LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Intro. We are relative Linux newbies (with Linux since Summer 1998). We run mostly RedHat and Mandrake −> the solutions might not be directly applicable to other Linux distributions (although most of them probably will be). Hope this helps; we try to be as practical as possible. Of course, we provide no warranty whatsoever. If you spotted a bad error or would like to contribute a part on a topic of your choice, we would like to hear from you.

General description of this Guide. A complete reference for new Linux users who wish to set up and administer their own Linux home computer, workstation and/or their home or small office network. The answers are meant to be simple, with just sufficient detail, and always supported with a readily usable example. The work is still in progress, but we hope the Guide can be helpful already. We welcome your corrections, advice, criticism, links, translations, and CONTRIBUTIONS. Pls note that there are no ad banners on our pages.

Conventions:
<> = single special or function key on the keyboard. For example `<Ctrl>` indicates the "control" key.
italic = name of a file or variable you probably want to substitute with your own.
`fixed width` = commands and filenames.

Part 0: For the Undecided (Linux Benefits)
If you are wondering what the Linux pros and cons are, and whether Linux is for you.

Part 1: Before Linux Installation
What distribution should I use, how to obtain it, Linux hardware requirements, how to partition your hard drive, about dual boot, which packages to install, which graphical user interface (GUI) to install (gnome or kde?), and how to login for the very first time.

Part 2: Linux Resources, Help and Some Links
How to access the Linux documentation (from under MS Windows or Linux), what are Linux help commands, where to find the geek dictionary, + pointers to some Linux newsgroups and websites.

Part 3: Basic Operations FAQ
After you installed Linux, here are answers to some questions that Linux newbie users/administrators may have when trying to perform every−day tasks: what are the file name conventions, how to run a program, shut down my computer, set up the path, add users, remove users, make your passwords and system more secure, work with file permissions, schedule jobs with "at" and cron, change your shell prompt, print symbols in the text mode, use color in the text mode, redirect input/output, write a simple shell script, install a new program ...

Part 4.1: Boot−time issues
Some info on LILO and GRUB boot managers, how do I choose the operating system which boots on default, hints on configuration of the boot loaders, "uninstalling" Linux ...

Part 4.2: Drives
Where are my drives, how to access them, configure user access, get the zip drive recognized, set 32−bit hard drive IO, increase the limit on the number of opened files, add a new hardrive, manage the swap space ...

Part 4.3: X−windows
How to switch between text and graphical consoles, set up my video card, monitor and mouse for the X−server, setup a graphical login prompt, change a default desktop, have multiple sessions of Xwindows running at the same time, use Xwindow remotely, install TrueType fonts from my MS Windows partition to Linux, how to copy−paste under X and in the text mode, how to use VNC.

Part 4.4: Basic Configurations
Real basics on how to configure the printer and soundcard, bits about configuration files, daemons, and device files.

Part 4.5: Networking
Setting up a network, ppp (connection over the phone), remote access to your computer, ftp and html server, e−mail, how my computer can get hacked ...

Part 5: Linux Shortcuts and Commands
Maybe this should have come first. A practical selection of Linux shortcuts and commands in a concise form. Perhaps this is everything that a computer−literate newbie Linuxer really needs. **Highly recommended.**

Part 6: Linux applications (proprietary or not)
Essential and/or famous Linux applications with some hint/comments: word processing, spreadsheet, database, latex. Extensive info on how to set up and use a CD recorder to write data, audio, and mixed mode CDs.

Part 7: Learning with Linux (commands for more esoteric work or programming)
Review of some more advanced or less useful commands/tools to get you started with fancier text processing, encryption, digital signatures (gpg), simple programming plus some info on the Linux console tools that can help you learn about computers. Under development so perhaps not so good: grep, regular expressions, sed, gawk, sort, ascii codes, linux built−in c compiler and tools, perl, python, tcl/tk, "Reverse Polish Notation" (RPN) calculator, scilab, wine ... working on it.

Appendix A: How to upgrade the kernel (by Alesh Mustar)
All you need to know to upgrade the Linux kernel (currently unmaintained)

Licence, Acknowledgments and log of changes.

The master copy of this page is http://sunsite.dk/linux−newbie/ (Denmark, Europe) hosted free by SunSite at Aalborg University. There are some official mirrors listed below.
Mirror: http://slayernetworking.com/newbie (Seattle, Washington, USA) hosted by Joe Morthland (Skull) (new).
Mirror: http://dbstreams.ca/mirrors/linux−newbie/ (Canada) hosted free by Kenan Bektas of DB Streams Inc.
Mirror: http://www.linsup.com/newbie/ (Australia) hosted free by linsup.com.
Only the link to the title page (e.g., http://sunsite.dk/linux−newbie/) is (rather) guaranteed not to change. Links to individual chapters may break because filenames may need to change in the future (as they did in May 2002).

Translations

A Hebrew translation (partial, pdf only) is available locally here (new).
A Portugese translation is available at http://www.onlinux.com.br/dicas/lnag/index.htm (Brasil) and locally here contributed by Ronaldo T Morais <rtoledo@onlinux.com.br>.
A French translation (in progress, Part 1−3 so far) is available at http://www.gerelia.asso.fr/lnag.html (France). Thanks to David <David.Lecat@gerelia.asso.fr> (new).
A Russian translation (new) is avialable at http://linuxbegin.by.ru/lnag_ru/ . An older Russian translation (ver. 0.10) is available here locally or at http://www.college.balabanovo.ru/rider/book/ (Russia).
A Polish translation is available at http://www.wzz.org.pl/~lnag/pl/ (Poland).
A Chinese translation (Big−5) of the "Linux Shortcuts and Commands" (ver. 0.32) is available (locally) here.
A Chinese translation (GB) of 3 parts is avialable (locally) here.

Downloading for Printing and Viewing Off−Line

The following (automatically generated using htmldoc) files are available for downloading:
*The **pdf** version of the Linux Newbie Guide is here (~600 kB, best for printing and off−line reading) and same zipped here, (slightly smaller, ~400 kB).*
*The **postscript** version of the Linux Newbie Guide is here. (big, ~ 1 MB) and zipped here (smaller, ~400 kB).*
*The **postscript** "2−logical−pages−on−1−physical−page" is here and zipped here (environmentally friendly for printing but very small print!).*
The pdf "2−logical−pages−on−1−physical−page" is here and zipped here (environmentally friendly for printing but very small print!).
*The **html zip** version of the Linux Newbie Guide is here (~300 kB, includes all the local html files).*

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Part 0: For the Undecided (Linux Benefits)

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section:

0.1 Fundamentally, why Linux?

If you truly enjoy working with computers, Linux is the operating system of your dreams. It is more fun than any other computer operating system around. However, the reason why Linux is truly revolutionary is that it is Open Software. Our science and technology works owing to the free availability of information and peer review. Would you fly a plane that was based on secret "science" and an unreviewed design, a plane at the internals of which nobody but the manufacturer could look? Then why would you trust a computer program containing secret parts and algorithms? Open−source Linux is ideally suited for a mission−critical application−−its security and power are based on robust solutions which anyone can view, criticize, or improve on. It is the implementation of the scientific method in computing.

The making of horseshoes, good glass, or measuring time were once closely guarded trade secrets. Science and technology exploded 500 years ago thanks to the sharing of knowledge by the means of printing. In the early days of printing, many of those who dared to share were assassinated for revealing "trade secrets." Linux is for the computer age what Gutenberg was for writing. Hopefully there will be no assassinations this time :−) . Linux does clash with those who claim the "ownership" of information, trying to push time back 500 years.

0.2 Is Linux for me?

Only you can answer this question. Linux is a mature, powerful, secure and extremely versatile UNIX−like operating system. The power and versatility come with a price−−you may need to be computer−literate in order to set−up and maintain Linux. Linux is relatively easy to use once the operating system and applications are set up properly. So, your mother will also be able to use Linux, if you set up an easy graphical account for her and put the proper icons/menus on her GUI desktop. Linux is secure, so your mother will not be able to damage the system no matter how hard she tries−−unless it's with a hammer :−) .

Linux is quite different from MS Windows, so do not expect that if you can get around MS Windows, Linux will be straightforward for you. You may need to learn. On the other hand, if you come from UNIX, Linux will be easy for you. If you don't know much about computers or you don't enjoy them, chances are Linux administration is not for you. If you don't know your hardware, Linux installation may be a challenge.

0.3 Linux is difficult for newbies.

This may be true. But the real question is: do you really want to learn it? None of the authors has a computer science background, yet we use Linux everyday and we love it.

0.4 What are the benefits of Linux?

Linux can give you:

o A modern, very stable, multi−user, multitasking environment on your inexpensive PC hardware, at no (or almost no) monetary cost for the software. Linux is a rich and powerful platform−−don't think of it as a "poor people" operating system. Out−of−box Linux has as much capability as MS Windows NT with $5000 in software add−ons, is more stable, and requires less powerful hardware for comparable tasks.

o Standard platform. Linux is VERY standard−−it is essentially a POSIX compliant UNIX. (Yes, Linux is a best−of−the−breed UNIX. The word "UNIX" is not used in conjunction with Linux because "UNIX" is a registered trademark.) Linux includes all the UNIX standard tools and utilities.

o Unsurpassed computing power, portability, and flexibility. A Linux cluster recently (April 1999) beat a Cray supercomputer in a standard benchmark. Linux is most popular on Intel−based PCs (price of the hardware), but it runs very well on numerous other hardware platforms, from toy−like to mainframes. One distribution (Debian) expresses the idea like this: "Linux, The Universal Operating System." Linux can be customized to perform almost any computing task.

o Advanced graphical user interface. Linux uses a standard, network−transparent X−windowing system with a "window manager" (typically KDE or GNOME).

o Dozens of excellent, free, general−interest desktop applications. This include a range of web browsers, email programs, word processors, spreadsheets, bitmap and vector graphics programs, file managers, audio players, CD writers, some games, etc.

o Thousands of free applets, tools, and smaller programs. "Small is beautiful" goes well with Linux philosophy. The small Linux tools and applets often work in tandem to perform more complex tasks.

o Hundreds of specialized applications built by researchers around the world (astronomy, information technology, chemistry, physics, engineering, linguistics, biology, ...). In many fields, Linux seems like "the only" operating system in existence (try to find out what your friend astronomer runs on her computer). The software in this category is typically not very easy to use, but if you want the power, it is the best software that humanity has in these areas. Doubtful? Have a look at:
http://SAL.KachinaTech.COM/Z/2/index.shtml for examples.

o Scores of top−of−the line commercial programs including all the big databases (e.g., Oracle, Sybase, but no Microsoft's). Many (most?) of these are offered free for developers and for personal use.

o A truly great learning platform. If you are a parent, you should be really glad your daughter/son does Linux−−s/he will surely learn something of lasting value. If you are a teacher, you should consider the installation of Linux at your school. "It is indeed a strange world when educators need to be convinced that sharing information, as opposed to concealing information, is a good thing" (http://edge−op.org/grouch/schools.html). You select Linux if you care to provide education, not training. The better the university, the greater the chance their computer department uses Linux in teaching. For example, under Linux, you can immediately begin modifying and compiling for yourself a spreadsheet application which is in every bit as advanced and capable as MS Excel. Linux puts you right on the cutting edge (in technology, project management, QA, methodology of science). Many teachers won't use Linux in schools because they are lacking in computer education themselves (at least that's what I see).

o Excellent networking capability built into your operating system. You think you don't need a network? Once you try home networking, you will never be able to live without it! How about connecting the two or more computers that you have at home and sharing your hard drives, CDROM(s), sound card(s), modem, printer(s), etc.? How about browsing the net on two or more machines at the same time using a single Internet connection? How about playing a game with your son over your home network? Even your old 386 with Win3.11 may become useful again when connected to your Linux Pentium server and when it is able to use your network resources. All necessary networking software comes with standard Linux, free, just setup is required. And it is not second−rate shareware−−it is exactly the same software that runs most of the Internet (the Apache software runs more than 50% of all Internet web servers and Sendmail touches some 70% of all e−mail). The pleasure of home networking is something I was able to discover only owing to Linux.

o Connectivity to Microsoft, Novel, and Apple proprietary networking. Reading/writing to your DOS/MS Windows and other disk formats. This includes "transparent" use of data stored on the MS Windows partition of your hard drive(s).

o State−of−art development platform with many best−of−the−kind programming languages and tools coming free with the operating system. Access to all the operating system source codes, should you require it, is also free. The "C" compiler that comes standard with Linux can compile code for more platforms than (probably) any other compiler on earth. Perl, Python, Guile, Tcl, Ruby, powerful "shell" scripting, and even an assembler also come as standard with Linux.

o Freedom from viruses, "backdoors" to your computer, software manufacturer "features," invasion of privacy, forced upgrades, proprietary file formats, licensing and marketing schemes, product registration, high software prices, and pirating. How is this? Linux has no viruses because it is too secure an operating system for the viruses to spread with any degree of efficiency. The rest follows from the open−source and non−commercial nature of Linux: Linux evolved itself by "bazar−like" mechanisms to encapsulate the best computing practices, code legibility and correctness, security, flexibility, usefulness, coolness, performance.

o The operating platform that is guaranteed "here−to−stay." Since Linux is not owned, it cannot possibly be put out of business. The Linux General Public License (GPL) insures that development/maintanance will be provided as long as there are Linux users. There is a great number of highly−educated Linux users and tens of thousands of actively developed projects.

o A platform which will technically develop at a rapid pace. This is insured by the modern, open−software development model which Linux implements: "build−on−the−back−of−the−previous−developer" and "peer−review−your−code" (as opposed to the anachronistic closed−software model: "always−start−from−scratch" and "nobody−will−see−my−code"). Even if the current "Linux hype" died out, Linux will develop as it did before the media hype started. Open source development does have its peculiarities: the development appears rather slow (vertically) but it proceeds on a very wide front, dangerous security bugs are fixed almost upon discovery, there are typically several alternatives for a program of similar functionality. Linux depth cannot be overestimated.

If you wanted to learn first−hand about the General Public License, check these famous GNU documents:

http://www.gnu.org/copyleft/gpl.html
http://www.gnu.org/gnu/linux−and−gnu.html
http://www.gnu.org/philosophy/categories.html#TheGNUsystem

In a nutshell, the GNU General Public Licence (GPL) allows anybody to:

− use the software at no charge, without any limitations,
− copy, and distribute or sell unmodified copies of the software in the source or binary form,
− modify, and distribute or sell a modified version of the software as long as the source code is included and licenced under the GPL,
− sell support for the software.

What this license *does not* allow to do is to modify the software and then distrubute a binary−only version of the software (without the source code). Speaking plainly, the GPL licence just forbids stealing somebody else's software for incorporation into a closed, commercial−only product. However, you may incorporate GPL software in a propriatory computer program if you obtain a permission from the author. Excluded from the use of GPL are persons who have been found to violate GPL.

The license under which Linux is distributed is probably the most important part of it. It is designed to perpetuate the freedom of information. Other important open−source projects include science and law (hardly a joke). The Linux method is really nothing new−−it is simply the application of the scientific method to software: you get information free, you add your ideas and make your living, and finally, you leave it free. However, some big corporations and their lawyers seem to be trying hard to change this, to push us back in time, to the dark ages, when information was kept "proprietary." Hence, you see in newspapers some famous Linux−connected persons involved in all kinds of struggles.

> To get a flavour for the value of Linux, here are some prices for commercial software as listed at www.amazon.com. All prices are in $USA, as listed on 2001−02−03, with discounts. Roughly equivalent Linux software is included on almost any Linux CD (but with no restrictions on the number of clients). In addition, the hardware for Linux is MUCH cheaper, since Linux can run all services on a single server:
>
> Microsoft Windows 2000 Server (5−client)−−$848.99; Microsoft Exchange 2000 Server (5−client)−−$1,279.99; Microsoft Outlook 2000 (1−client)−−$94.99; Systems Management Server 2.0 (10−Cals)−−$994.99; Proxy Server 2.0−−$886.99; Microsoft SQL Server 2000 Standard Edition (5−client)−−$1,229.99; Microsoft SQL Server 2000 Standard Edition (1−user License)−−$4,443.99; Microsoft BackOffice Small Business Server 4.5 NT (Add−On 5−CAL)−−$264.99; Windows NT Server Prod Upgrade From BackOffice SBS Small Bus Server (25−client)−−$558.99; Microsoft Windows 2000 Advanced Server Upgrade (25−client)−−$3,121.99; Microsoft FrontPage 2000−−$129.99; Microsoft Internet Security and Acceleration Server −−$664.99; Site Server Commerce 3.0 (25−client)−−$4,092.99; Visual C++ 6.0 Professional Edition with Plus Pack−−$525.99; Microsoft Visual Basic Enterprise 6.0 with Plus Pack−−$1,128.99; Microsoft Visual Sourcesafe 6.0 CD−−$469.99; Microsoft Office 2000 Standard (1−client)−−$384.99; Adobe Photoshop 6.0−−$551.99; Microsoft Plus Game Pack−−$19.99.

The word "free" has two quite different meanings in the English language, and it sometimes leads to misconceptions about the free nature of Linux. These two meanings follow the Latin adjective "liber" and the adverb "gratis," and they are often illustrated with the phrases "free speech" and "free (of charge) beer." Most Linux software is free in both senses, but it is only the first sense which is essential to Linux.

0.5 What are the differences between Linux and UNIX?

Command−line−wise, almost none, although this has been changing (for better or worse). Linux has a much larger market appeal and following than any commercial UNIX. GUI−wise there are also no major differences−−Linux, as most other UNICES, uses an X−Windowing system.
The major differences:

− Linux is free, while many UNICES (this is supposed to be plural of UNIX), cost A LOT. The same for applications−−many good applications are available on Linux free. Even the same commercial application (if you wanted to buy one) typically costs much more for a commercial UNIX than for Linux.
− Linux runs on many hardware platforms, the commodity Intel−x86/IBM−spec personal computers being the most prominent. A typical UNIX is proprietary−hardware−bonded (and this hardware tends to be much more expensive than a typical PC clone).
− With Linux, you are in charge of your computer, whereas on most UNICES you are typically confined to be an "l−user" (some administrators pronounce it "loser").
− Linux feels very much like DOS/Win in the late 80s/90s, but is much sturdier and much richer, while a typical UNIX account feels like a mainframe from the 60s/70s.
− Some UNICES may be more mature in certain areas (for example, security, some engineering applications, better support of cutting−edge hardware). Linux is more for the average Joe who wants to run his own server or engineering workstation.

0.6 What are the differences between Linux and MS Windows?

Mouse−click−wise, almost none, once Linux is properly installed. Linux installation can be a challenge though, whereas MS Windows comes pre−installed with your computer.

The major differences:

− Linux is free, while MS Windows costs money. Same for applications.
− Linux file formats are free, so you can access them in a variety of ways. On MS Windows, the common practice it to make you lock your own data in secret formats that can only be accessed with tools leased to you at the vendor's price. How corrupt (or incompetent?) must the politicians who lock our public records into these formats be! "What we will get with Microsoft is a three−year lease on a health record we need to keep for 100 years" [http://news.bbc.co.uk/hi/english/health/newsid_1694000/1694372.stm].
− With Linux, you are unlikely to violate any licence agreement−−all the software is happily yours. With MS Windows you likely already violate all kinds of licenses and you could be pronounced a computer pirate if only a smart lawyer was after you (don't worry, most likely none is after you).
− MS Windows tries to be the "lowest−common−denominator" operating system (for better or worse), whereas Linux is built for more sophisticated, feature−hungry computer users (for better or worse).
− MS Windows is based on DOS, Linux is based on UNIX. MS Windows Graphical User Interface (GUI) is based on Microsoft−own marketing−driven specifications. Linux GUI is based on industry−standard network−transparent X−Windows.
− Linux beats Windows hands down on network features, as a development platform, in data processing capabilities, and as a scientific workstation. MS Windows desktop has a more polished appearance, smoother general business applications, and many more games for kids (these are not better games though−−Linux games tend to be more sophisticated).
− Linux is more feature−rich than you could imagine. Heard on the Internet: "Two big products came from the University of California: UNIX and LSD. And I don't think it's a coincidence."

0.7 I don't believe in free software, etc.

And do you believe in the Internet? The Internet and Linux share underlying ideas and have common roots. Do you remember the disbelief about the Internet a few years ago, the endless, seemingly unbeatable arguments that free Internet cannot exist? "Who pays for that, anyway?"

The reality is simple. Cooperation and good will can benefit many at the same time: your gain is not my loss. The Internet works fine and is expanding at a rapid pace. So does Linux.

Here is the opinion of an IBM executive: "The reason we are so excited about Linux is we believe Linux can do for applications what the Internet did for networks" (http://linuxtoday.com/news_story.php3?ltsn=2000−08−17−001−04−PS−EL). IBM just (May 2002) spent 1 billion dollars making Linux run on all their hardware platforms (mainframes, workstations, PCs, laptops).

0.8 "There ain't no such thing as a free lunch"

"The economic paradigm which makes this true depends upon scarcity of resources. Software resources are only scarce because we all keep software proprietary and secret. But not Linux! When I give you my software, it may create an opportunity cost for me, but I get to keep it even after I've given it to you. It is a free lunch only rivaled in history by the loaves and the fishes." (Brett Bazant <bbazant@shaw.wave.ca> (http://linuxtoday.com/cgi−bin/showtb.pl?tbsn=12450&sn=5418).

0.9 I need high security. With commercial software, I can sue them if things go wrong.

Don't count on suing. Things go wrong on many MS Windows NT machines every day, and there are no damages awarded by courts. Read your MS Windows license agreement to find out that there is no guarantee whatsoever that ANYTHING will work. Trying to sue would be a waste of your money.

Linux also provides no guarantees, although it is far more secure than any version of MS Windows. If you are really security−sensitive , you can use high−security tools built by companies that rely on the availability of the source code to design and test their security features (e.g., Kryptokom in Germany provides high security firewalls). The "security in obscurity" implemented in MS Windows has repeatedly been demonstrated to be a naive approach.

"Risk aversion is what dictates you use Linux and other open products, rather than NT. The risks with NT are entirely out of your control, and there is nobody you could sue if anything goes wrong. Why people still believe the myth that Windows in any form offers any bit of accountability "more" than Linux remains a complete riddle to me." (David Kastrup, Research Engineer, Bochum, Germany, "Internet Week," http://www.techweb.com/se/directlink.cgi?INW19990329S0050).

0.10 I need standards. Big software corporations (Microsoft) provide standards.

Perhaps that's what people would expect from large corporations, but the reality is rather different. Once, big companies loved inventing nuts that could be undone only by their own service shops. Did these nuts become standard? Hardly. They didn't because there was no public benefit involved, and they couldn't because they were patented. Luckily, now we have open and free standards for nuts. A "proprietary standard" is such a ridiculous oxymoron that it is hard to believe that educated people can believe in it. (Currently, marketing types use the term "de facto standard" or "industry standard" to cover up the ugliness of the lack of standards.)

An example from the computer field. The "standard" MS Word file format has changed numerous times over the recent years. This keeps happening probably for a good business reason: as soon as other companies "reverse−engineer" the current Word format, Microsoft changes it. There are even sub−formats (an MS "fast−save" anybody?). It is also completely closed−−Microsoft does not publish any specifications. How can the user benefit from this in a longer term? What is the Microsoft guarantee that MS Word 6.0 file format will still be legible in 2020?

"... Microsoft's standards are both proprietary and arbitrary− the stealth incompatibility of Office 97 file formats with older versions of Office or the subversion of Open standards like XML with proprietary extensions that require Internet Explorer 5, MS Active server and so on, are sober reminders of what the company does to a market." (Xavier Basora, http://www.osopinion.com/Opinions/XavierBasora/XavierBasora47.html).

"... Microsoft's monopoly doesn't guarantee that your current MS Office will work with any previous or future MS Office. This is in spite of any number of Microsoft apologists arguing that the benefit of Microsoft's monopoly has been a standard for productivity applications." (Wesley Parish, http://www.osopinion.com/Opinions/WesleyParis/WesleyParish10.html).

To add to the confusion, companies typically do not "standardize" on file formats but on the applications that are supposed to produce them. It is like standardizing on a manufacturer of nuts instead of on nuts. How is this supposed to work if the file manufacturer keeps changing the specification to drive their sales?

"We need standardized, open file formats so that users can exchange documents between platforms. The actual word processing software used to generate these documents shouldn't even be an issue." (Ted Clark, http://linuxtoday.com/news_story.php3?ltsn=2000−09−29−004−06−OP−MR−0010).

There are a few text/document oriented file formats that are quite definitely more standard than MS Word file format: ASCII, XML (with non−propriatory stylesheets), HTML, SGML, LaTeX, TEX, PostScript, pdf, dvi ... and all of them have excellent support under Linux. The MS Word file format can also be read/written very well under Linux by OpenOffice (and a number of other applications) to cover your current needs. Advanced, "universal," open−source document formats (XML−based) is being worked on by an independent organization. The story is similar like with other proprietary computing "standards" (*.giff vs. *.png anyone?).

Linux, by its very nature, is based on true, published and free standards because "open source" makes the full specifications available to everybody (competitors or not). I believe that the urge for open standards is the very driving force behind Linux. Some people feel that they cannot afford to trust their algorithms and data to a commercial entity, let alone one that repeatedly demonstrated it is untrustworthy.

Have a look at a draft of this Argentinean law for a taste of the future. It sounds like the Argentineans may be the first to decide that their public records cannot be held hostage by a commercial entity: (source: http://slashdot.org/articles/01/04/28/010216.shtml): "... Public National Organizations mentioned in article 1 of this law, will not be allowed to use programs that store data in non−public format ...". Several other counteries are contempleting or enacting legistlations requiring storage of data in public file formats.

There is a strong perception in the Linux community that there is a serious problem with the computing "standards" championed by large software vendors. This includes their standards for our "static" data , as well as the knowledge embedded in our computer codes. Can we afford to have somebody decide for us when, how, and at what cost we can access our work? This problem is ignored and even aggravated by people who are paid to take care of it. Linux is a grass−root answer.

0.11 I Need MS Windows for Reading Writing MS Word Documents

In a large corporate environement, you may have little choice−−they locked themselves by cheerful productions of non−portable forms, templates, visual basic−driven web pages and other "tools".

In a smaller environment, you can use OpenOffice.org suit (OO) that runs on Linux, Windows, Mac, Solaris, with full file−level compatiblity. It can be downloaded and installed for free (no restrictions whatsoever) so nobody should really complain about the file format (some control freaks still will). Just to make sure, it can import and export MS Word and Excel documents of reasonable complexity very well. However, its native file format is fundamentally much better (and non−propriatory). Feature−by−feature, it can do almost anything MS Office can, plus some extras. Depending on whom you ask, the ease of use veries between "50% more difficult" to "20% easier" (measured on experienced MS Office users). Very complex documents are best transferes as pdf, and OO can make them on the fly.

So, probably you do not need MS Office any more. Download your OO for MS Windows and Linux at: http://www.openoffice.org/

0.12 MS Windows popularity insures that it is "here to stay".

This is likely true. Nintendo is probably also "here to stay." However, I like computing; therefore, I choose a computer with a powerful operating system, not a lowest−common−denominator piece designed for "everybody."

Linux is quite positively here−to−stay because of its open−source nature (Linux cannot possibly be put out−of−business). It is a standard selected for countless projects that are not going to go away, and some of them are quite "mission−critical." Try the International Space Station, for which Linux is the operating system (http://www2.linuxjournal.com/lj−issues/issue59/3024.html).

Plus, never underestimate the strength of the Linux community. Linux is "here to stay" at least for the computer avant−garde. Many Linuxers do not even want Linux to become very popular because they fear it could "dumb down" the elite Linux platform.

0.13 But LINUX may fork into many different systems ...

This is a typical argument of the type spread by those specializing in the marketing tactics known as "fear, uncertainty and doubt" (FUD) [about the competing product].

"Forking" in this context means "branching a computer program," so as to create parallel "subversions" of the program, and consequently fragment Linux.

There is very little (if any) evidence of harmful forking of any software included with a typical Linux distribution. Where forking did occur, it has always turned beneficial. Quite possibly, this is because although there are no artificial barriers to fork software under Linux, there are also no artificial barriers to merge the best pieces back.

The theoretical background on how forking software can be good for its development might have been actually given quite some time ago by the German philosopher Georg Wilhelm Friedrich Hegel (1770−1831), with his concept of dialectic development. E.g., in "Phenomenology of Spirit", Hegel concludes: "... the schism incipient in a party, which seems a misfortune, expresses its fortune rather."

0.14 Linux is a cult

The Linux community has repeatedly been labeled "religious zealots" by journalists whose well−established computer magazines received massive feedback after they had published highly unfair articles on Linux. So yes, the Linux community is numerous, literate, and willing to express its opinions. And many computer journalists/magazines know that Linux means less money for them (users pay less for their computing and the associated advertisement, while expecting more). Does this explain the "zealots"?

Face it, you salespeople pretending to be journalists. There is hardly any integrity left in the computing press. How many words on Linux did your PC Magazine (or whatever) publish by 1999−01−01? Wasn't Linux at least an interesting technology by that time? It surely was, yet you conspired to keep your readership in the dark, selling your journalistic integrity for money. And now, after Linux has surfaced in the mainstream (non−computer) media, you keep writing misleading articles about it saying "yah, but it will/cannot/may ...." whatever (trying the "fear, uncertainty and doubt" tactics to kill it). And adding "Microsoft is already ...", continuing to write about the vaporware and the future paradise in the face of the increasingly stealthy, unstable, pricey, architecturally unsound computer platform, whose greatest achievement has been exhorting unheard−of−before money by denying inter−operatibility, and killing any existing or proposed standard (by "embracing" and then proprietary−extending it). Whom do you serve? Surely not your readers.

I worded it as strongly as I could. Am I a zealot? Or am I just trying to voice my disapproval for the self−serving actions of the computer "powers−that−be"?

You think "self−serving" is ok in business? How pathetic must your business be! I always thought that business was a social contract in which we exchange good values, for a mutual benefit. As I read history, societies use to hang / guillotine / electrocute those members who really persisted in their self−serving business. Well, times have changed. A bit for the better, a bit for the worse :)))

0.15 The total cost of ownership (TCO) of Linux is high

Nobody really knows how to calculate the "total cost of ownership" of a general piece of software. So a good lawyer + accountant can prove whatever point they are paid to make, and they regularly do.

Let me try a simple estimate of how much the average total cost of the ownership of MS Windows is. Let's add the fortunes accumulated by all the MS Windows software makers. Add all the salaries of all generic Windows programmers, consultants, support and training personnel, IT management, etc. Now, add the losses customers must surely have suffered while the software corporations were presenting them with "features" so as to achieve their current monopolistic status. Divide this figure by the number of years (whatever timeframe you selected), and the number of MS Windows users (only in the countries in which software is normally paid for). Here is the TCO of MS Windows. However you count it, it will be many thousands of good US dollars per average joe per year. You didn't pay that much money? Well, you must have, it has just been hidden from you. Yes, developed countries waste billions every year on software.

How much did Linux cost? Hardly anything. The number of users is much lower, too, but you will be hard pressed to come up with $10 per user per year.

Yet, in my opinion, the total cost is not what matters the most. What value did I receive for my money? You would have to calculate the total value of ownership (TVO?), then subtract from it the total cost of ownership (TCO) to obtain the "net benefit of the ownership."

I guess accountants only talk about the TCO for software "necessary for doing business," and thus skip the issue of value and benefit. There is no value in the normal commercial software, it is just the necessity for doing business these days. Well, Linux satisfies my computing necessities at zero monetary cost, and the personal pleasure and learning value is just great.

0.16 Linux is idealistic "dreaming"; it is business that rules the world nowadays

Think of Linux as a consortium. Businesses/individuals get together to address a common computing need or problem. They may chip in labour or money, hire a technical leader, or otherwise cooperate to make Linux address their requirements. The solution is totally theirs for keeps, and it does not have to cost a lot−−they can re−use the pre−existing Linux software pieces. They may cooperate to overcome the advantage that a big "industry leader" may have and use against their interests.

Linux is the end−product of activities of many such loose "consortiums" who "scratch their needs." So Linux is a business, but it is not necessarily about selling software−−it is about access to reasonably−priced software that matches your need, solves your problem, sells your hardware or service, and which is totally yours (the licence never expires, and you will never be cut off from the source code).

0.17 Linux sux etc.

Then do not use Linux. You are not doing anybody a favor by using Linux. GNU/Linux is free and powerful software, but only for those who like or need it. There are alternative operating systems for you to choose from and they may better match your requirements. Although most Linuxers enjoy the growth and welcome new users, some are not very happy about it because "the crowd and commerce can spoil the hackers' paradise we created." Therefore, you really aren't doing anybody a favour by using Linux.

In this context, it may be worthwhile to briefly summarize Linux strengths and weaknesses: Linux is owned by its fans (your piece of ownership comes free with your free subscription to the fan club), definitely very powerful and feature−rich, highly configurable, as flexible as you want it to be (comes with complexity), low on the cost of hardware, comes with any networking bell−and−whistle known to man, requires a computer literate administrator, some essential desktop applications are still behind commercial offerings on other platforms (e.g., spreadsheet and word processing), a number of excellent end−user applications come "standard" and free with the operating system, the graphical user interface is very nice but still not as polished as Apple or MS offerings, Linux is highly standard (UNIX, POSIX), open file formats used all along, thousands of programs available for free download (although the ease of use and quality of these varies vastly). And most of all, Linux is enjoyable!

Next: Before Linux Installation
Back to Top Page

Part 1. Before Linux Installation

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section:

1.1 Which Linux distribution should I use?

Linux distribution is a coherent collection of free software with the Linux kernel (operating system) at its center.

The differences between the various Linux distributions ("distros") are minor: the installation program, choice of the bundled tools/applications, arrangement of a few things on the hard drive (most of Linux is still at the same, standard hard drive location in all distributions). Whichever distribution you decide to install, you will end up with essentially the same Linux.

We mostly use "Red Hat Linux" (also called RedHat or RH) and Mandrake (sometimes called MDK) for the following reasons:

1. They are both very popular (both an advantage for a newbie and a testimony to their quality).
2. They are both general−purpose distributions.
3. They both come with relatively easy setup programs.
4. Both Mandrake and RedHat contributions to Linux are "open software" (this means that all the software written by the packaging corporations and included on the distribution CDs is licensed under the General Public License, GPL, so that it can be legally copied, given away, reused, etc.).
5. Both Mandrake and RedHat can be obtained very cheaply or free if you don't care for commercial support. This is a consequence of (4).
6. Mandrake is based on RedHat, so both are very similar. Software packages for RedHat typically work on Mandrake (and vice versa) without problems. However, Mandrake is a bit more automatized and makes a somewhat nicer desktop than RedHat, and requires Pentium processor on default (RH will run on a i386).

In short, as a newbie, you can safely bet on "RedHat" or "Mandrake" unless you like something else or have specialized needs, or your environment suggests using something else (e.g., if you have an experienced guru nearby, or a bunch of friends who are using Linux, you may want to use the same distribution − makes getting help a whole lot easier).

The most recent distributions we recommend (Nov. 2002) is RedHat 8.0 or Mandrake 9.0. These are excellent distributions. Be sure to specify the most recent version if ordering your software from a dealer−−many dealers like to clear their inventory by sending you an older version (this applies not only to Linux). Generally, development under Linux is fast, and you don't want to waste your time with

older distributions. The authors of this guide have no connection to RedHat, Mandrake (or any other Linux distributor) whatsoever.

Our recommendation of Mandrake and RedHat for newbies does not mean that other distributions don't offer benefits or unique features which may surpass Mandrake or RedHat in specific areas. We do believe that we benefited from exposure to a different distribution because it helped us understand Linux better.

We tried Debian and we liked it very much. It was probably as easy as RedHat, but Debian seems less common (hence, being newbies, we picked up RedHat). The great benefit of Debian is that it is 100% non−commercial (put together by volunteer hackers, the true Linux way) and it probably most strictly adheres to Linux standards (it probably sets the standards too). Another great benefit is that Debian crams on their numerous distribution CDs thousands of tools and applications−−easily much more than any other distribution. All these tools/apps are nicely "packaged" (for ease of installation) and tested for compatibility. This makes Debian distro look monumental, safe, conservative, and always slightly outdated. So yes, we would not have a problem recommending Debian as a great general−purpose Linux distribution. Debian calls itself "The Universal Operating System" for a good reason. At any time, Debian carries 3 versions. (1) The "stable" version (sometimes called "potato"), and we would not recommend it, unless you are really paranoid on stability and don't mind quite outdated packages. (2) The Debian "testing" version (sometimes called "woody") is probably as stable as the latest RedHat, and more stable than your current Mandrake. It is much more up do date than Debian "stable". Debian Woody is the version we like. (3) If you don't mind occasional trouble, you can also the the third branch called "unstable", which is likely quite up−to−date.

Corel was once working on their own Linux distribution apparently geared towards a nice and easy platform to run the Corel suite of office applications: WordPerfect wordprocessor, QuattroPro spreadsheet, Corel Presentations, Paradox database, CorelDraw artist package.... The Corel Linux was based on the Debian distribution. It looked initially quite promissing, but it is unclear to me what Corel has done with it (was paid by Microsoft to drop it?). In brief, Corel Linux is dead now, and I would never recommend it to anybody because it it a dead−end. The only reason to mention it here is that Corel Linux once received lots of publicity, so you may still hear about it.

Slackware seems to be favorite among "cutting−edge hackers" who like being close to the operating system and hardware−−we did not use it so this is hearsay. We would have trouble recommending Slackware for Linux newbies. Our reviewer Bill Staehle says: "The real 'reason' for a newbie to avoid Slackware is that it is much more command line oriented, and lacks some of the 'cutsie slick and drool' tools that the other distributions have." However, we received feedback from Linux newbies who use Slackware and it works very well for them. It seems that Slackware is relatively simple and cool because of the lack of automation. Therefore, with a bit of effort, a computer−literate administrator can actually understand what is going in her operating system (this is not something I can always say about Mandrake, or MS Windows for that matter). Perhaps Slackware is to Linux what DOS is for MS Windows :) .

S.u.S.E distribution (http://www.suse.com) is very popular in Europe. It surely looks German−−a solid, general−purpose distro with an easy setup and an excellent reputation. Many users swear by SuSe. We couldn't find cheap Suse CDs though but it appears you can download it.

Caldera is another, well−known and respected distribution. It is said to be aiming at corporate users, have the most fancy installation program, a set of advanced (and pricey) remote configuration tools, and other corporate goodies. In Aug.2000, Caldera purchased SCO Unix (the original UNIX, including the UNIX trademark) which gives them an even more "corporate" look in my eyes. Caldera does not seem to be putting too much of their work into the Linux community, nor to care too much about the home Linux users, so I would not consider it for my home use.

There are "localized" versions of Linux for specific countries or languages (Korean, Chinese, Japanese...)−−they likely contain (on default) all the hacks and docs (documentation) that the users in these countries want to see. Says Bill Staehle: "You may want to mention the Conectiva Linux distribution, loosely based on RH from Brazil. As such, it is in Portugese, and is also available in Spanish. Try: http://www.conectiva.com.br/". I heard several good things about Conectiva, so if Portugese or Spanish was my language, I would probably give it a try.

There are also "special purpose" distributions, e.g. the "real−time" editions of Linux (might be useful if you are in for automation, robotics, fast speed data acquisition, etc.), very small distros (if you like the idea of running Linux from a single floppy which can be useful for system security or recovery), Linux for embedded systems (if you wanted to customize Linux as a small "special purpose" device, which could be good for the next−generation stereo, MP3 player, palm computer, or a fancy cellular phone), parallel computing and clustering systems (might be great if you plan to do your own weather forecasting :−) or at least nuclear explosion simulations :p), etc. Here the differences will be larger, but these distributions are not meant to be "general purpose". As a newbie, you likely don't want to start with any of these, although you might be tempted to. (They surely show Linux strength and viability−−Linux runs on toys, even a wrist watch, as well as computer clusters that make the currently fastest systems in the world.)

The distribution you need is of course specific to the hardware platform you have. This means that for your PC hardware containing an Intel 386 processor, or Intel 486, or Intel Pentium, or Intel 586, or Intel 686, or Cyrix, or K6, AMD, or similar, you need the binary distribution called "Intel" or "386" or x86. [Unless you are prepared to start with your own compilation of the Linux source code, which is not typical for a newbie :−)] . This happens because there are binary distributions for other hardware platforms too: PowerPC, Alpha, Apple, IBM mainframe, "Intel StrongARM", Transmeta, and perhaps a dozen more−−you don't want to get those binaries for

your PC clone; they surely will not work on a PC machine with an "Intel" or "AMD" processor inside. If you have no−Intel hardware, you may want to search the Internet to find who supports it (chances are Debian does, they seem to support even the most exotic ones. Then, you need to obtain "Debian ARM" or "Debian Motorola 680x0"or "Debian PowerPC" or "Debian SPARC ", ...).

In short, although newbies get confused with the multiple Linux distributions, there are reasons to have different distros. They should be viewed as a Linux strength rather than weakness. Linux is simply filling all application and hardware platform niches.

This guide concentrates on RedHat and Mandrake for the PC (Intel) platform. Many of the answers will work fine on other distributions or platforms, but we did not try them.

> Which Linux Distribution should I select for my old computer(s)? Quick answer: Debian, Slackware, or perhaps BasicLinux (current version), or an older version of RedHat, Mandrake, or SuSE. Justification: RedHat, Mandrake, SuSE, Caldera, and TurboLinux are optimized/suitable for hardware current at the date of their release. They may be difficult or impossible to install on older machines mostly due to the memory contraints and speed. Debian and Slackware are suitable for most older hardware as well.

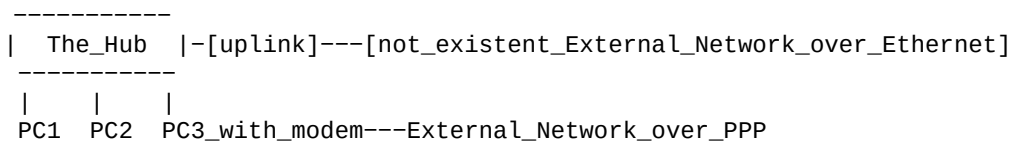
1.2 What are the Linux hardware requirements?

"Out−of−box" Linux should run on a 386SX−based PC with 8 MB of memory, but such a low−end computer is practical for text−only applications (no X−window). A 486 with 16 MB memory and 600 MB free (unpartitioned) hard drives work under X−windows but don't expect it to fly at all. My 586−133 MHz with 64 MB of memory runs acceptable under Linux with X. My 1.33 GHZ "Athlon" (AMD processor) with 256 MB of memory is a real pleasure to run with an instantaneous response even when running many large applications concurrently. I would not buy today a computer with less than 256 MB of memory (Dec.2001).

My 486−33 MHz with 8 MB memory and 1 GB hard drive has too little memory to run adequately stand−alone under GUI, but is still useful in my home network environment running as an X−terminal (a 486−class machine also performs just adequately stand−alone if it has at least 16 MB of memory but sometimes memory for old computers is hard to obtain at the price you would think it is worth). My old portable 386−SX−20 MHz Toshiba with 9 MB memory and 120 MB hard drive runs "legacy applications" under MS Windows 3.11 and it connects to our Linux home network and is thus still useful. We tried older Debian Linux on this Toshiba too, and it runs fine in text mode. (Pls note that Mandrake requires a Pentium processor.)

If you are willing to jump a few extra hoops, you should be able to install and run Linux on as little as 4 MB of memory, but this is probably not worth the effort for the general purpose home Linux machine. I would say: get at least 32 MB of memory, and if possible 128 or 256 MB −−more memory can make a difference in performance when running several large GUI applications concurrently. Memory is cheap these days. Please note that many current distribution have problems running their installation programs on older computers with a small amount of memory (although once installed, Linux will typically run just fine). If you require more help on installing Linux on a low−memory computer, try: http://7thguard.net/files/DebianHOWTO.txt

Networking is where Linux really shines, so consider getting 10−base−T Ethernet cards−−they are not very expensive and will be perfect to connect your two or more home computers together. Also, look around for old Ethernet cards which MS Windows deems obsolete−−they can be bought for a really low price and they will work great under Linux. To connect just two computers, a cross−over cable for direct Ethernet−card to Ethernet−card connection is sufficient ("networking for the poor"). To connect more than 2 computers together, you need a hub (~US$30 to US$80) and normal (not cross−over) cables. (If you have extra Ethernet cards, you may also consider installing more than 1 Ethernet card on a computer, use direct connections using the cross−over cables, and save the expense of a hub. But it adds a configuration complexity to your system. The 10−base−T system uses "giant phone" (RJ45)−type connectors and all machines are connected to one box (called the hub). The hub has an extra connection (called "uplink") which I will use if I ever have a permanent "over−Ethernet" connection to the outside world. Here is a schematics for a straight−forward home network arrangement:



> Here, I show a local private network consisting of PC1, PC2 and PC3, connected through a hub. Since I do not have "External_Network_over_Ethernet" on my home hub "uplink", PC3 provides my connection to the outside world (over a modem). Therefore, PC3 is called the "gateway" for all computers on my local private network (except PC3 itself). I enable the firewall software on PC3 PPP network interface, and let PC3 know how to dial out and connect to the outside. The outside world can only see PC3. As far as they can tell, PC1 and PC2 do not exist. My local ethernet network is "trusted" because only trusted people have physical access to PC1, PC2 and PC3. ("PPP" stands for "Point−to−Point Protocol" and it is a standard for communicating over phone lines.)
>
> You can, of course, build a more complicated network with Linux. A PC can have 2 (or more) ethernet cards. It may then work as bridge between 2 (or more) networks. The PC will act as a gataway for all traffic between

between the 2 networks. The networks do not have to be known to the outside world ("local private networks") and sit behind a firewall enabled on a gateway computer. The outside world will only know about 1 computer of mine, the "gateway" to the external network. Other computers will still be able to communicate with the outside world, but all the traffic will appear outside to originate from one, very busy computer−−the gateway.

Here is another suggestion on setting up a different kind of network, using a very much older type hardware, which uses coaxial cables (like for the cable TV). For this, no hub is necessary. Because this networking scheme is older, it can be assembled using cards and parts that are sometimes available for free:

(edited for space) From: John.Edwards@brunel.ac.uk Subject: Linux Guide−a suggestion

Hi. Many older 10Mbps network cards (and some newer ones as well) have a BNC connector and you can usually pick up old co−axial cabling when companies upgrade to UTP. Add a T piece for each machine and a 50−ohm terminator at each end (about 1 pound or $1.50 each) and you have a home network that will happily support more machines than you probably have room for. And most importantly−−no expensive hub (or cheap hub that can cause trouble). There are other advantages to co−ax as well, it's tougher to break and more resistant to noise from other equipment.

Disadvantages: There is a limit of 185 meters per network segment of thin co−ax, 30 machines per network, and you're stuck at 10Mbps, but I don't see any small home network needing more than that. Also if one cable goes down then the whole network stops, this shouldn't happen often unless someone unplugs a cable section. You can disconnect the T piece from a PC without harming the rest though.

Quick diagram, T for a T piece and Term for a terminator:

```
Term−T−−−−−T−−−−−T−−−−−T−Term
     |     |     |     |
     PC    PC    PC    PC
```

The various parts connect together using BNC connectors similar to a TV & video connector but with a bayonet that secures the two sockets together. For more details see the /usr/doc/HOWTO/Ethernet−HOWTO

The most straight forward and modern, however, is to get 10−base−T ethernet cards for your computers and a hub to connect them.

1.3 Will my hardware work under Linux?

Not every piece of PC hardware is supported under Linux, but most are, particularly the more standard, older, and popular ones. This applies to SCSI adapters, CDROMs, writable and rewritable CDs (CD−R and CD−RW), video cards, mice, printers, modems, network cards, scanners, Iomega drives, etc.

The most notable exceptions are the so−called Winmodems (=MS Windows modems also called "software modems"). Avoid these like fire−−they are a bit less expensive than full modems, but they are crippled (some processing is done by the main computer CPU instead of by the modem), and there is little chance you will have a Winmodem running on Linux right away (for more info on Winmodems, see http://www.idir.net/~gromitkc/winmodem.html). External modems are never "Winmodems" so if in doubt, purchase an external modem (external modems are more expensive, but they don't drain your PC power supply, are easily portable between machines, look better, and show modem activity). Additional points to consider with modems: "Older externals using a Rockwell Protocol that don't work too well. Also, the newer USB modems are not currently (March 2001) well supported. See the winmodem page." [source: B.Staehle].

Another area of potential problems is the video card. If you have a recent "cutting edge" 3D or uncommon card, you may want to check its compatibility at http://www.Xfree86.org.

Zip drives of all kinds are supported fine.

I wouldn't count on Linux supporting a parallel port (non−SCSI) scanner, no matter if the manufacturer claims TWAIN (="Technology Without An Interesting Name", no joke here) compatibility.

So the short answer is yes, in all likelihood your standard PC will run Linux with no problems. You don't invest much when trying Linux, so probably the easiest way to make sure is to attempt an installation on your existing hardware. There are Linux hardware compatibility lists at http://hardware.redhat.com/hcl/genpage2.cgi and http://metalab.unc.edu/LDP/HOWTO/Hardware−HOWTO.html if you want to check your newer or less popular hardware.

When purchasing new hardware, I would always check its Linux compatibility on the above lists. You can also ask your supplier if the hardware is supported under Linux, but I would take the answer with a grain of salt−−too many companies have incompetent sales personnel/technical support. When purchasing a new computer, I would consider a system with Linux pre−installed. A number of major suppliers offer systems (particularly large ones) with Linux, but many don't. You can always get a system with

Linux−preinstallled from a smaller vendor.

If you are an adventurous person, as I am, I would pay no attention to the remarks above, chances are 90−10 that the hardware will work.

If a piece of hardware of yours is (apparently) not supported in your current Linux distribution, don't give up. Chances are that: (1) It is supported, but you don't know how to set it up. (Solution: stay around with Linux for a few weeks, don't waste your time, when you get some understanding of how your system works, then you may be able to set it up.) (2) You have to go through a more complex setup to support the hardware (for example some cryptic command or a kernel re−compile, which is not as difficult as it seems). (3) An updated (different?) distribution already supports it "out−of−box" (you can usually order it for US$1.99). (4) There is already an upgrade somewhere on the Internet, you have to find it, download it, and figure out how to install it. (4) The upgrade will be available next month−−Linux development goes really fast!

1.4 How do I download Linux?

Do yourself a favor and do not download Linux. Buy an installation CD instead. Linux can be downloaded completely from the Internet, but it is a very large and sophisticated operating system. The download may take hours or days of download time, and you may encounter problems and frustrations, e.g., due to errors in the downloaded files.

If you do have a speedy Internet connection (definitely not a 56k−modem, but perhaps cable modem) and you are not a complete newbie, a Linux download may be an option to you after all. Try http://www.linuxiso.org/ for ready−to−burn CD images (ISO format) of your selected Linux distributions.

1.5 How do I get a Linux CD?

Many possible ways. (1) Buy the "RedHat" or Mandrake CDs from Linuxmall (http://www.LinuxMall.com/), or Cheapbytes (http://cart.cheapbytes.com/cgi−bin/cart)−−last time I checked, "the unofficial" RedHat GPL was US$2.99 or something like that for a 2 CD package + shipping and handling. They will mail you bare CDs. You get no printed manual, no support, no boot diskette, but the price is right, and the manual and tools to make a boot diskette are on the first (installation) CD. I purchased several packages from "Cheapbytes" and they always arrived fast, were of good quality, and there were no problems with my credit card charge (the authors have no connection to "cheapbytes" whatsoever). (2) Buy the boxed "official Red Hat" or "Mandrake" from the same place on the Internet or from your favorite software supplier; prices start at around US$40−−you will get the printed manual, e−mail or telephone installation support (60 days?), the boot diskette, additional CDs with "bundled" commercial applications, and perhaps other goodies (free updates?). If your time is worth lots of money, you may opt for more advanced technical support at higher price. (3) Copy the installation CDs from your friend. This is perfectly legal and ok−−Linux is free. If you have a Linux CD, don't be shy to loan it to your neighbor. (4) Check your library, local bookstore, or http:/www.amazon.com. Several Linux handbooks come with an attached CD containing a full Linux distribution. This is a good way to start with Linux because it is definitely a good idea to have a nice Linux handbook. With Linux' countless utilities, I need a handbook all the time. The drawback is that the books often include versions of Linux which are quite dated. Perhaps consider the "official" Linux with a handbook? (5) Visit a Linux "installfest" when one is organized in a place near you. Local Linux "gurus" will install Linux on your computer free (bring the computer) and you will likely be able to get a Linux CD too (why don't you bring some empty CD−R to the fest?). Check for the Linux User Group on the net to see when the nearest to you plans an installfest. Good way to meet other Linuxers too.

> Here is a more comprehensive list of places to obtain Linux CDs with their location, so you can find something near to you (after B.Staehle) : http://www.ixsoft.de (low price CDs in Europe); http://linuxservice.de (another source in Germany); http://www.polo.demon.co.uk/emporium.html (The Linux Emporium); http://www.linux−emporium.co.uk; http://definite.ukpost.com/ (Definite Linux Systems); http://www.mallind.demon.co.uk/ (GPL + official distros); http://www.amush.cx/linux/ (GPL distributions in UK only); http://www.linux123.co.uk/ (GPL + official distros in .uk); http://www.kihi.com.au/bowtie/ (Bowtie Software − cheap CDs in OZ); http://EverythingLinux.com.au/ (cheap CDs in OZ); http://www.lsl.com.au/ (cheap CDs in OZ); http://www.arles.ns.ca (official distros and BSD in .ca); http://www.affinity−systems.ab.ca/ (Official distros − hardware in .ca); http://www.warpedsystems.sk.ca (GPL + official distros, custom built system CA); http://www.softcopy.on.ca/ (cheap CDs in CA); http://www.libranet.com (Libranet Linux Vancouver, BC, CA); http://www.linuxwarehouse.co.za/ (low cost and official in South Africa); http://linuxcentral.com (Clinton Township MI 48035); http://www.lsl.com (Chesterfield, MI 48047, USA); http://www.cheapbytes.com (Lodi, CA 95241, USA); http://www.infomagic.com (Flagstaff AZ 86004, USA); http://www.tummy.com/krud/ (Fort Collins, CO 80525, USA); http://www.pieceby.com/ (Hudson, NH 03051, USA); http://www.linuxmall.com (Aurora, CO 80046−0190, USA); http://www.linux−now.com (Clarion, PA 16214, USA); http://www.xcomputing.com (San Francisco, CA 94134, USA); http://www.ccsoft.cc/linux/ (Santa Rosa, CA 95401, USA); http://www.storeanywhere.com/ (Brooklyn, NY 11235, USA); http://www.linuxcomponents.com (Owings Mills, MD 21117, USA).

1.6 I have Linux Installation CDs but no install floppy. What do I do?

If your computer can boot from the CD drive (most older computers cannot), you don't need a boot diskette to install Linux. Have a look at your BIOS setup; the boot sequence can often be set up there (the default is often floppy followed by hard drive). My computer has the CD drive specified as the first boot device in the BIOS yet still cannot boot from the CD drive. So the BIOS setup does not necessary reflect the capability of your machine. If you can boot from the CD drive, just insert the RedHat CD into the CD

drive and reboot the computer to enter the RedHat Linux installation program.

If you don't know how to access your BIOS setup, read this paragraph. The BIOS setup can typically be entered at boottime by pressing the proper key at the right moment (often when a prompt is briefly displayed). Most often, it is the <Del> key. Here is a list of key combinations used by popular BIOSes: Acer notebooks: <F2> during Power−On Self−Test (POST). American Megatrends (AMI): <Del> during Power−On Self−Test. Award: <Del>, or <Ctrl><Alt><Esc>. Compaq: <F10> after the square appears in the top right corner of the screen during boot−up. Dell: <Ctrl><Alt><Enter>. DTK: <Esc> during Power−On Self−Test. IBM Aptiva 535: <F1> while the square with the wavy lines is displayed in the upper right corner during power−on. IBM PS/2: <Ctrl><Alt><Del>, then <Ctrl><Alt><Ins> when the cursor is in the top right corner Mr. BIOS: <Ctrl><Alt><S> during Power−On Self−Test. Packard Bell: For some models, <F1> or <F2 > during Power−On Self−Test. Phoenix: <Ctrl><Alt><Esc> or <Ctrl><Alt><S>, or <Ctrl><Alt><Enter>.

If your computer cannot boot from the CD drive, make an install boot diskette from under DOS or the MS Windows DOS mode. (You have go to to "Shutdown" and "Restart in MS−DOS mode", not just run a DOS window).

It is important that you have a perfectly good floppy (without even one bad cluster). The program that makes the diskette does not check if the floppy was written properly. Also, don't count on the DOS `FORMAT` utility finding a faulty floppy−−it probably won't. If I were you, I would make two or three boot floppies at once−−you may be surprised how many diskettes have problems. For me, the third floppy worked! If your install diskette does not boot, make another one−−it definitely should boot.

Here are the commands. To make the boot floppy run:

```
F:\dosutils\rawrite.exe −f F:\images\boot.img −d a: −n
```

To make the supplemental (optional) diskette run:

```
F:\dosutils\rawrite.exe −f F:\images\supp.img −d a: −n
```

This assumes your CDROM is the DOS "F:" drive, and your floppy is "A:", adjust the commands if the drive letters are different on your system.

The commands above run the utility "rawrite" and specify the input file ("disk image", after the option "−f") and the destination drive (after the option "−d"), and suppresses the prompt to insert a floppy (option "−n"). You may find it easier to run `rawrite` without any argument−−it will interactively prompt you for the input image (pick the file name as in the commands above) and the destination drive letter.

From under Linux, you can make a boot disk by mounting the RedHat CDROM and typing the commands (as root user):

```
cd /mnt/cdrom/images/
dd if=boot.img of=/dev/fd0
```

[The ``dd" command copies files. The above command specifies that the input file ("if") is ``boot.img" and the output file ("of") is /dev/fd0, which is the first floppy drive, i.e. the floppy drive number zero (if you want to write to your second floppy drive, use /dev/fd1).]

1.7 What do I need to read before installation?

It is VERY helpful to get some UNIX orientation if you don't have any. Buy a good Linux manual or dust your old Unix handbook. Almost all Unix concepts apply in Linux, and almost all UNIX commands will run fine under Linux. I find manuals for MS Windows useless (click this, click that, look at the screenshot), but manuals for Linux/UNIX are typically great (give you an understanding of the system, a lasting benefit).

You may want to learn about your hardware: how many and what size hard drives you have, the type, number, order and size of all partitions on each drive, where your DOS/Windows partitions are, which one is the DOS/Windows boot partition (if you plan to have dual boot), what type of mouse you have, what video card and with how much memory, what monitor (max synchronization frequencies), etc.

Go to BIOS setup to see the number and geometry of your hard drives. Run DOS "fdisk" to display your hard drive(s) partition table(s), and print it. Watch your system boot to learn about the type of your video card and the amount of video memory. Boot MS Windows, go to the control panel−devices and write down the sound card, modem, network card types and settings (name, type, IRQ, i/o address, DMA channel). Read the label underneath your mouse to see the type of mouse you have. (Next time you buy a mouse, get a Linux−ready 3−button Logitech or similar−−Linux makes good use of all three buttons.) Dust off your monitor manual to find out the maximum synchronization frequencies (vertical and horizontal) that your monitor supports. Never use frequencies out of the monitor specification−−this may damage your monitor.

You may want to browse the RedHat or Mandrake manual. If you don't have the printed copy, an html version is on your CD so you can read it using any web browser, e.g. Netscape for Windows. Look here to see how to access this manual and some additional reading material which is on your Linux CD.

1.8 Can I have MS Windows and Linux installed on the same computer?

Yes, you can. Many Linuxers use a dual boot. This is typically achieved by installing MS Windows on one hard drive partition and Linux on another partition. Linux comes with a simple boot manager called LILO (or a more sophisticated one called GRUB), which will let you choose, at boot time, the operating system you boot. Install MS Windows first and Linux only afterwards or else the MS Windows installation program will disable your access to Linux. Have a Linux boot floppy ready if you need to re−install MS Windows−−MS Windows will surely disable your access to Linux and you will have to boot Linux from the floppy and then re−run the command `lilo` to be back in business.

From under Linux, you will be able to read from and write to your MS Windows drive partitions so that the data exchange between MS Windows− and the Linux−based program is seamless. You will also be able to use your existing MS Windows−based resources: sound files, backgrounds, pictures, fonts, etc. (First check if it does not violate your license agreement though, smile. For products that are on rent to you from Microsoft, it probably does. With my Linux computers, I am proud to have no pirated software on my system whatsoever.)

1.9 How do I partition my hard drive?

Before Linux installation, you might really want to know what a hard drive partition is. The concern is that you delete your MS Windows partition when you really don't want to−−you want two separate partitions to dual boot. This means: MS Windows is on one partition, Linux is on a separate partition. You do not normally install Linux on free space on your MS Windows−allocated partition(s). It is possible to install Linux on a MS Windows partition, but we do not recommend it.

If you plan a dual boot (Linux and MS Windows on the same computer), first use your DOS/Win utility `FDISK` to make the MS Windows partition(s). Leave part of the hard drive(s) unpartitioned for Linux. You will make and format the Linux partitions during your RedHat (or Mandrake or whatever else) installation. Linux will recognize the free space on the harddrive.

Make the MS Windows partition "primary" and "bootable". Install, configure, and test your MS Windows before Linux installation. If you plan to run Linux only, you need just a clean hard drive (no partitions) to start with.

It is possible to have only one Linux partition (plus one for MS Windows if you dual−boot). But it is better to have more partitions so that you can keep users' data separate from the rest of the operating system. This way, if something ever goes wrong, or if you have to reformat or re−install the operating system, you don't lose the users' data. (You can perform a complete Linux re−install without losing the contents of the `/home` directory that contains all user data if you skip the "re−format" option given to you during installation. But for that, the `/home` directory must be on its own partition.)

During the Linux setup, you will be asked to partition the available space on your hard drive(s). There are many possible ways to partition, depending on your hard drive space, requirements, and taste. I like Linux hard drive partitions like this (for a modest total of 2 GB of hardrive space which I give to Linux in this example):

```
mount point    type     size
/              ext2     300 MB
/usr           ext2     1200 MB
/home          ext2     380 MB
swap           swap     120 MB
```

In the above example, I dedicate 300 MB for the root partition that holds the base of the Linux operating system. I allocate 1200 MB to the mount point that will be visible on my filesystem as the `/usr` directory and will contain the user's programs (the programs that don't come with the base operating system and I install later, for example StarOffice). I dedicate 380 MB for the partition that will be visible as the directory `/home` and will contain the setting and data of all users on the machine. And I allocate 120 MB to a "raw" partition for the operating system to use as the virtual memory (extension of the physical, silicon memory on the hard drive, so−called swap). If your kernel is lower than 2.2 (this is the case with standard RH5.2 and earlier), your swap partition cannot be larger than approximately 127 MB. The rule of thumb is that the swap should be about twice the amount of the physical memory (RAM). If you need more (e.g. if you have lots of physical memory, or you expect to run custom programs with really large data structures) you might want to create a larger swap partition during the installation (or several smaller swap partitions) or add a swap file(s) later.

2 GB is a respectible amount of disk space and should be sufficient for users who like having many applications. (This is because Linux applications tend to be slimmer than their MS Windows equivalents). However, if you try to install everything that's available on the modern distribution CDs, you will surely run out of disk space. My experience is that however large the hard drive space, it will get filled and I regret I don't have more :−) .

If my space on the hard drive is really restricted, I may consider a two−partition setup like this (for a lean 650 MB total dedicated to Linux):

```
mount point    type     size
/              ext2     600 MB
swap           swap     50 MB
```

In this example, I dedicate 600 MB to the base of the operating system, applications, and user documents/data, and allow 50 MB for the swap partition (for the operating system to use as the virtual memory). The 50−MB swap should be quite sufficient for medium duty operations. The limitation of 600 MB for the operating system, applications and user data means that you will have to be very selective as to which applications you install or else you risk running out of hard drive space. Try pressing <F1> when installing the optional software that comes on the Red Hat CD−−it will give you a short description of what the software does so you could perhaps decide if you really need it. (Don't worry too much if you miss something you need, you can install the missing parts later). You can easily finish the RedHat installation with 200 MB free on your Linux partition (out of 600 MB used in this example) if you make reasonable choices. Please note that "bundling together" the root partition "`/`" and the `/home` directory will likely save you some disk space, but it is not the safest solution.

It is possible to install Linux on even less disk space than in the example above, but you will have to be really picky as to what you install.

For a larger available hard drive space, I may consider the following setup (for a comfortable total of 15 GB dedicated to Linux):

```
mount point    type     size
/              ext2     800 MB
/usr           ext2     5000 MB
/usr/local     ext2     3000 MB
/home          ext2     5200 MB
swap           swap     1000 MB
```

Please note that the the mount points can reside on different physical hard drives. Linux agglomarates all the hard drive space into a single directory tree.

Another consideration when setting up the partitions on older computers (486?). Many older BIOSes have the restriction that the boot partition cannot extend beyond the 1024th cylinder on your first physical hard drive. To overcome this limitation, simply make the first (bootable) partition so that it ends before the cylinder number 1023 (this makes this partition max approximately 512 MB in size, which is plenty for the "/" root partition). Once Linux boots, the BIOS restriction does not matter any more because Linux takes over the hardware management and it can access the partition(s) beyond the cylinder number 1023.

When installing and using Linux, your drives appear as devices with the following names: hda−−first IDE drive (stands for "hard drive a", i.e. the master drive on the first IDE interface), hdb−−second IDE drive (i.e., the slave drive on the first IDE interface), hdc−−third IDE drive (i.e. the master drive on the second IDE interface), hdd−−fourth IDE drive (i.e. the slave drive on the second IDE interface). The numbers mean the partitions on the physical drives: "hda1" means the first IDE hard drive (hd a), first partition (1); "hda2" is the first IDE hard drive, second partition; "hda3"−−the first IDE hard drive, third partition; (and so on if you have more than 3 partitions on the first IDE hard drive); "hdb1"−−second IDE hard drive, first partition (or just "hdb" if it is the CDROM installed as a slave on your first IDE interface). "hdc1"−−third IDE hard drive, first partition, etc. SCSI drives have analogous names but start with the letters "sd" (="SCSI drive"), followed by the letter indicating the SCSI interface and by the number indicating the SCSI device id. For example, "sda4" means "first SCSI interface, id number 4". If you have an external zip drive attached to your parallel port, it will appear as SCSI device "sda4" (zip drives work in a SCSI−emulation mode).

The listing of partitions that your Linux setup program presents to you during installation will include any MS Windows partitions which you have. For example, I have the following MS Windows partition:

```
mount point    type     size        comment

[no mount]     vfat     1200 MB     ["Win C drive, hda1]
/mnt/dos_hdd2  vfat     1600 MB     ["Win D drive, hdd2]
```

Don't erase these MS Windows partitions during your Linux installation if you want a dual boot. If you erase the MS Windows partition, MS Windows is gone from your system! If not sure, backup your data from your MS Windows partitions before Linux installation. "msdos", "fat" and "vfat" and "ntfs" are typical filesystems used by DOS and MS Windows 3.x/95/98/NT.

As a quick reference, here is a brief summary of the standard linux partition types ("filesystems") with a short description. I copied the info from the linux manual pages: `man fs` and `man mount` (with some additions after I had a look at the source code files at

`/usr/src/linux/fs`). The **underlined** filesystems are the ones that you are more likely to use. Other filesystems (not listed below) are available as add−ons (for example journaling filesystems, compressed, encrypted, ...).

minix is the filesystem used in the Minix operating system, the first to run under Linux. It has a number of shortcomings: a 64MB partition size limit, short filenames, a single time stamp, etc. It remains useful for floppies and RAM disks.

ext is an elaborate extension of the minix filesystem. It has been completely superseded by the second version of the extended filesystem (ext2) and will eventually be removed from the kernel.

ext2 is the high performance disk filesystem used by Linux for fixed disks as well as removable media. The second extended filesystem was designed as an extension of the extended file system (ext). ext2 offers the best performance (in terms of speed and CPU usage) of the filesystems supported under Linux. In short, ext2 is the main (default, typical) Linux filesystem.

ext3 is an extension of the ext2 filesystem with journaling. It is backwards and forward compatibile with ext2. It means that ext2 can be converted into ext3 without reformatting or data loss (just re−mounting the partion is required). ext3 can be changed back to ext2, also without data loss. I use ext3 extensively since Oct.2001−−it is simple and trouble−free. It is included as an installation "option" since RedHat 7.2 and Mandrake 8.0. It is highly recommended that you use this filesystem.

xiafs was designed and implemented to be a stable, safe filesystem by extending the Minix filesystem code. It provides the basic most requested features without undue complexity. The xia filesystem is no longer actively developed or maintained. It is used infrequently.

msdos is the filesystem used by DOS, Windows, and some OS/2 computers. msdos filenames can be no longer than 8 characters followed by an optional period and 3 character extension.

umsdos is an extended DOS filesystem used by Linux. It adds capability for long filenames, UID/GID, POSIX permissions, and special files (devices, named pipes, etc.) under the DOS filesystem, without sacrificing compatibility with DOS.

vfat is an extended DOS filesystem used by Microsoft Windows95 and Windows NT. VFAT adds capability for long filenames under the MSDOS filesystem.

proc is a pseudo−filesystem which is used as an interface to kernel data structures rather than reading and interpreting /dev/kmem. In particular, its files do not take up disk space. See `man 5 proc`.

iso9660 is a CD−ROM filesystem type conforming to the ISO 9660 standard. Two extensions (listed below) are automatically supported.

High Sierra −−Linux supports High Sierra, the precursor to the ISO 9660 standard for CD−ROM filesystems. It is automatically recognized within the iso9660 filesystem support under Linux.

Rock Ridge −−Linux also supports the System Use Sharing Protocol records specified by the Rock Ridge Interchange Protocol. They are used to further describe the files in the iso9660 filesystem to a UNIX host, and provide information such as long filenames, UID/GID, POSIX permissions, and devices. It is automatically recognized within the iso9660 filesystem support under Linux.

hpfs is the High Performance Filesystem, used in OS/2. This filesystem is read−only under Linux due to the lack of available documentation.

sysv is an implementation of the SystemV/Coherent filesystem for Linux. It implements all of Xenix FS, SystemV/386 FS, and Coherent FS.

nfs is the network filesystem used to access disks located on remote computers.

smb is a network filesystem that supports the SMB protocol, used by MS Windows for Workgroups, Windows NT, and Lan Manager. To use smb fs, you need a special mount program, which can be found in the ksmbfs package, found at ftp://sunsite.unc.edu/pub/Linux/system/Filesystems/smbfs. [Standard linux command "smbmount" will also do.]

ncpfs is a network filesystem that supports the NCP protocol, used by Novell NetWare.

devpts−−is a pseudo file system, traditionally mounted on /dev/pts. In order to acquire a pseudo terminal, a process opens /dev/ptmx; the number of the pseudo terminal is then made available to the process and the pseudo terminal slave can be accessed as /dev/pts/<number>.

fat −−is not a separate filesystem, but a common part of the msdos, umsdos and vfat filesystems.

UFS −−is a file system widely used in different operating systems.

swap−−is a special partition type used for swapping data from memory to hard drive.

raiserfs−−is a brand new journaling filesystem available as standard with Linux kernel version 2.4.1 up (January 2001).

hfs (=hierarchical files system)−−MacIntosh filesystem. It is a late beta version., i.e., not recommended for use with critical data, unless read−only.

ntfs−−MS Windows NT filesytem. It is still "experimental" under Linux, i.e. not recommeded for production machines, unless read−only (Aug.2001).

1.10 The MS Windows partition occupies my whole harddrive. Can I shrink/split it without a re−install?

Possibly. There is a utility called `FIPS.EXE` on my RedHat CD that does just that. Check the directory `\dosutils\fipsdocs\` on your RedHat CD for documentation. If I were you, I would back−up my essential data before doing anything to my partitions. There are also commercial utilities to change the partition size without destroying its contents.

My personal preference is to do a clean re−install of MS Windows on a single, dedicated partition. I leave some space on the hard drive unpartitioned so I can use it later for Linux. My fair division of hard drive space between MS Windows and Linux is 50/50. Linux programs tend to be smaller but they include (as standard) components that MS Windows offers only with many thousand of dollars of add−ons: e.g., servers (not just clients) for telnet, ftp, http, and mail, several databases, programming languages, graphics processing programs ...

1.11 How do I start the installation?

Insert the installation boot diskette into your floppy drive, the RedHat CD to the CDROM, and reboot. If you can boot from the CD, insert the RedHat CD into your CDROM drive and reboot.

You may also start the installation from DOS (or real DOS mode under MS Windows), by running `EZSTART.BAT` which is on my RedHat CD from Cheapbytes.

Most installers give you an option between text and graphical install. You need to select "text" if your computer memory is restricted.

1.12 Is the Linux installation difficult?

It was not for me. It seems that for most newbies, it is fairly straight forward and painless. Depending on your hardware and installation choices, it typically takes 1 h to 2.0 hours. [Expect longer or possible problems for slow systems with very restricted memory−−it took a whole night to install RH6.0 on my 486−33 MHz with 8 MB memory, the system pausing for 5 minutes at a time appearing to do nothing, yet it installed ok.] Upgrades from previous installations take longer and tend to be more problematic.

However, some newbies reported that the installation was a "total nightmare" to them (hardware problems? lack of experience?). If you encounter problems, my advice would be to install a plain−vanilla system, without struggling with the highest resolution on your fancy video card or other bleeding−edge hardware which you might have. Anything can be added/configured later, after you get more understanding of how things work on your system. Even a re−install is always an option for a newbie (it seems Linux gurus think it is a shame to ever re−install). It seems that many newbies have problems because they specify too high screen resolutions (which may be not supported or supported only with some extra tune−up). Again, it may not be wise to break your whole installation for support of a single device−−the support can be added/tuned−up later.

1.13 Which packages should I install?

Linux software comes in "packages". For example, my Linux Mandrake 7.0 installation CD contained 1002 packages. Mandrake 7.2 packs 2 CDs of software−−my installation of Mandrake 7.2 put 1123 packages on the hard drive. Mind you, this is not all the software available for Linux−−just a selection made by people who put the Mandrake distribution together. Mandrake tends to pack more software than RedHat.

No matter what distribution or version, the CD contains packages that make the base operating system (kernel, libraries, a selection of command−line configuration and maintenance tools, etc) a rich selection of networking "clients" and servers" with appropriate configuration and monitoring tools, some end−user text mode applications, base X−windowing system, at least one GUI desktop (most likely several), and likely a slew of GUI applications.

The installation program (either RedHat or Mandrake) will ask you which packages to install. If you select "workstation installation", then the packages normally found on servers will be omitted from your installation (for example, the Apache web server may be skipped). If you choose "server installation", then typically the end−user desktop applications will not be installed (for example, the GIMP graphical utility may be omitted). You can also choose to install "everything", and this is my favorite option for a home computer installation. Finally, you may opt to make your own selection of packages to install−−read on.

It is definitely a very bad idea to select installation of packages/programs on the basis of how interesting their names sound−−some packages have rather unusual names and I would never guess what they do. You could cripple your system by omitting the installation of an essential package (e.g., a library). You might also be disappointed when insisting to run some cool−named, cutting−edge piece of

software ("version 0.1") that happened to be included on the distribution CD. In general, you might be annoyed by the functionality (or lack of it) that your "customized Linux" exhibits. Being a newbie, it sometimes pays to trust the defaults selected by your distribution creator.

Therefore, for my final "production" installation, I would stay away from the tempting installation option "expert install−−select packages manually" unless I wish to install everything anyway. For starters, I like the safe "max default installation", however this installation option is called on your CD.

If you don't install a package and later find that you need it−−don't panic. It can easily be installed later. Read on.

1.14 Which GUI desktop should I install, KDE or GNOME?

Disk−space permitting, definitely both. You can later decide if you prefer KDE, GNOME or another desktop, but whatever your choice, you definitely want both the KDE and the GNOME libraries installed. Once you have the libraries installed, KDE programs can be run under GNOME and vice versa, which is great because there are nice applications written using either library. As far as the amount of disk space is concerned, the "desktop" is only a small part of the KDE and GNOME systems so you don't save much space by omitting the desktop and trying to install "libraries only". Both GNOME and KDE come with a set of nice programs and tools, so it is definitely worth it to install both desktops in full. I never heard that the two adversely interfered with each other. For every−day work, I use the KDE desktop, because it feels more solid than GNOME. If you like more "cutting edge" and "cooler", go GNOME, but don't complain if things don't always work quite that well.

I would also install the other "alternative windows managers". They hardly take any space (some are really tiny) yet they can be useful under some circumstances. You can run any KDE or GNOME application from under any of them, as long as KDE and GNOME libraries are installed.

KDE is more power hungry. On older hardware (e.g., 133 MHz Pentium) I prefer GNOME to KDE. Other windows managers are ligther than either KDE or GNOME. Therefore, on really modest hardware, I would choose one of the "alternative" windows managers.

1.15 I finished the installation. How do I log−in for the very first time?

As root. "root" is the only account that exists after the initial installation (newer installation programs do prompt you to create a regular user account during the installation). Example text mode login:

```
my_machine_name login: root
Password:  my_password
```

In the example above, I typed the word "root" at the login prompt. After that, I entered the password that I chose during the initial Linux installation. The password did not appear on the screen when I typed it (for security). After I login, I find myself in a text−mode terminal.

If you installed the GUI login screen, the login procedure looks similar, but occurs on an X−window screen (if you occasionally have problems typing there, perhaps remember that your mouse cursor must be above the dialog box. The X login screen implements the "focus−follows−mouse" policy). After a successful login, my default GUI desktop is launched.

"root" is a special account with an absolute power over the system, and it is used for system administration. You surely want to create at least one more "user" account later to do regular (not system administration) work. Read on to learn how to do it.

1.16 How do I crash Linux?

As root, you can do whatever you want, including an accelarated system crash. Try (reconsider if you really want to crash):

```
cp /dev/zero /dev/mem
```

As root, you can even erase all the files on your system with a similarly innocuously looking one−liner (don't do it):

```
rm −fr /
```

This is not to say that Linux is an easy crash, but that the system administrator ("root") has the complete power over the system. You can make MS Windows unusable by trashing some files from C:\WINDOWS OR C:\WINDOWS\SYSTEM directory. An important distinctions that makes Linux resiliant is that the user and administrative accounts are separated. Regular users can touch only the files they own, and similarly the programs users run can only junk user−owned areas, no matter how buggy or malicious a program might

be. The separation of "administrative" and "user" accounts adds to system complexity, yet it also makes the Linux system truly multi−user. This is unlike the old MS Windows approach. With the latest version of MS Windows, Microsoft moves more towards the UNIX approach. An old saying comes to mind: "those who do not know UNIX are destined to re−invent it".

Conclusion: do not use the root account for routine work. Add a regular−user account as your first administrative task and use this account for your experimenting with Linux. Here is how to do it (as root):

```
adduser johnbrown

passwd johnbrown

[enter a good long password with a number in the middle]

[reenter the same password without a mistake]

exit

[login as johnbrown]
```

The root account is for administration and setup only. As root, I would not do things/run programs that I do not exactly understand what they do. At least not on a computer that I plan to use for real work. Really.

My learning path was as follows:

1. Install Linux.

2. Play around, experiment (root and not root, do cool things, setup stuff. I want to be able to predict the behaviour of my system−−only then do I know that I understand it).

3. Re−install.

4. Unless I don't have any more time (I never have enough), goto 2.

Once properly installed on good hardware, command−line Linux is legendarly stable−−it can run for months or even years. As a newbie, you can almost bet that a funny system behaviour results from either your actions as root, or from flakey hardware.

1.17 Can I use Graphical User Interface (GUI) all the time?

I imagine it is possible to use GUI exclusively under Linux, but I don't think it would be very efficient for administration. The GUI under Linux is very nice, but it pales in comparison with the possibilities available under the command line. To make sure, I use the GUI every day both under Linux and MS Windows, and I find GUI great for program launching, and other routine tasks. Yet, for system housekeeping or automation, GUI is inflexible. The command line interface (CLI) is a richer interface to me, and it also gives me some understanding of the working of my system.

Two quotes to amplify this point: "Using a graphical interface is fairly easy because it limits you to a few basic operations that you can learn quickly. But if those basic operations aren't what you need, then you may have problems". "One characteristic of a user−friendly system is that it does what the user wants. In other words, the most user−friendly system isn't necessarily the simplest one." [http://linux.oreillynet.com/pub/a/linux/2001/11/15/learnunixos.html]

I find that a mix of GUI and command line is perfect to cover all my needs. If you are determined to use GUI only and are not willing to learn any command line options, you might consider delaying your Linux installation until the GUI tools are better developed, unless you have somebody to help you administer your computer for now.

On the other hand, if you have a computer−agnostic girlfriend or boyfriend, and all s/he does is execute perhaps half−a−dozen different programs, you can set up a nice GUI screen for him with the icons or buttons or menus he requires. This, after some intial encouragement, may make him accept Linux.

From this Guide's point of view, command line is certainly simpler for documentation. Icons and menus are meant to be customizable and therefore your icons and menus may be quite different from mine. Also, to set up an icon or command, I need to know the command that stands behind the icon. In short, understanding of the command line is indispensible for setup and any work beyond trivialities, even under GUI. We include no screenshots in this guide−−we found them ourselves useless in learning about computers.

1.18 How do I upgrade a Linux distribution?

For a full system upgrade (updated distribution CDs, .e.g., RH7.0−>RH7.2), "follow the time−honoured principle of upgrading Unix systems: do a fresh, 'clean' install and add back your data. Yes, we're talking about reformatting your partitions and installing from scratch." (the quote from: http://www.northernjourney.com/opensource/newbies/newb025.html).

Thoughtful partitioning of your hard drive will facilitate future upgrades (your data files can be preserved in your "home" directory).

A checklist I made for myself when upgrading:

− Log in as root.

− Make a copy of the `/etc` directory to some place where it won't be distroyed (e.g., zipdrive). This will let me have a look at your previous setup in case I encounter problems.

− Make a backup of any valuable data in the /home directory. This is just in case something went really wrong, for example, if I happened to format a wrong partition.

− Determine and write down the mountpoints on your filesystem using the "diskfree" command:

```
df
```

For example, on my system it shows that the `/home` my home directory is mounted on a separate hard drive partition called `/dev/hda9`. and the directory `/usr/local` is on `/dev/hda8`.

− Perform the installatation of Linux, but preserve and don't reformat the partitions `/home` and `/usr/local`. This is critical if you want to preserve your data.

− Boot the new installation and check that it works.

− For each user on the system, create a login with the old name and old user ID number, for example,

`ls /home/maria/* −l` (check the user ID for account "maria" in the file listing of her home directory),
`useradd maria −u 503` (create a user acount "maria" with with an example uid "503").

− From each user home directory, delete the potentialy troublesome setup files, e.g.: ".kde", ".ICE*", etc. Good oportunity to delete any junk too. The files may be troublesome because the new version of kde is likely to perform better with it most recent settings (it will create them on first startup). E.g.,

```
cd /home/maria
rm −fr .kde
```

− Make sure that each remaining file in the home directories belongs to the appropriate user. For example, I may do:

```
cd /home/maria
chown −R maria *
chgrp −R maria *
```

Go to part 2: Linux Resources, Help and Some Links
Back to Main Page

Part 2. Linux Resources, Help and Some Links

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas
The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section (Linux Resources, Help and Some Links):

2.1 Any Linux reading materials?

This guide is not sufficient?

The RedHat Linux distribution CDs contain lots of documentation. Part of it is in html format and part in plain text format. You can read it all from under DOS or MS Windows before you install Linux. For example, a soft−copy of the RedHat manual can be viewed with any MS Windows−based html browser, e.g. Netscape for Windows or MS Internet Explorer. Just access the file `D:\doc\rhmanual\manual\index.html`. (assuming your CDROM is drive D under MS Windows).

Also, check the directory `\doc\LDP` for the excellent Linux Documentation Project manuals. For example, you can browse the Linux System Administrators' Guide by accessing the file `\doc\LDP\sag\sag.html` with your favorite html browser.

Also, check the directory `\doc\HOWTO` for the HOWTO documents, the directory `\doc\HOWTO\mini` for the MINIHOWTOs and the directory `\doc\FAQ` for a set of FAQs on different topics (FAQ="frequently asked questions"). For example, these commands will let you read the Linux−FAQ document (plain−text format) from under DOS:

```
D:
cd \doc\FAQ\txt
edit Linux−FAQ
```

Under Linux, you can read the same documentation from the CD using, for example, this command:

```
lynx /mnt/cdrom/doc/rhmanual/manual/index.html
```

This will start lynx, a simple text−mode html browser, to view the RedHat manual. Please note that under Linux, the CD must be mounted first, and the example above assumes that the mountpoint is the directory `/mnt/cdrom/`. You can also use Netscape for Linux, StarOffice or any other html browser to view the RedHat manual and other documentation in the html format. You can read plain−text documents from the CD under Linux using, for example, these commands:

```
cd /mnt/cdrom/doc/FAQ/txt/
less Linux−FAQ
```

(The `less` command lets you scroll through the contents of a text file.)

After installing linux, the documentation, whatever part of it you installed, is in the directory `/usr/doc/` or `/usr/share/doc`. If you didn't install the documentation, consider installing everything now, it may be worth it. For example, the directory `/usr/doc/LDP` contains the Linux Documentation Project manuals. These commands will let you browse the Linux System Administrators' Guide:

```
cd /usr/share/doc/LDP/sag
lynx sag.html
```

Also, check `/usr/share/doc/HOWTO` for the HOWTO documents, and `/usr/share/doc/HOWTO/mini` for the MINIHOWTOs.

The location of the documentation is sometimes `/usr/doc`.

For more or updated documentation, see http://www.ibiblio.org/mdw/index.html

2.2 Is there a help command?

Most Linux commands can be run with the "−−help" option. For example, this command will give you concise help on the Linux `cp` (copy) command:

```
cp −−help | less
```

More extensive info is accessed from the command line using the so−called manual pages `man` *`topic`*. For example:

```
man cp
```

will display the manual page for the "cp" (copy) command. The manual pages are the standard "help" system under Linux, and contain a wealth of detailed, very technical information, but typically require some effort to be understood by a newbie.

The `man` command uses a simple utility called `less` that lets you scroll through a text. Use arrow keys to scroll, press "q" to quit. Actually, `less` can do more than this. Press "h" for help when running `less`, or learn more about `less` using the command

```
man less
```

There is also the info command `info` *`topic`*. For example:

```
info cp
```

will give you the help for the "cp" (copy) command. Often `info` contains information similar to `man`, but more up−to−date. Unfortunately, the `info` navigating utility is not very intuitive, so I use man pages more often. There is also `pinfo` (a substitute for the `info` interface, perhaps easier to use than `info`).
If you don't remember exactly the name of the command that you need to use, try `apropos`. For example, to obtain a list of commands which have something to do with "copy", I execute this from the command line:

```
apropos copy
```

The command `whatis` is similar to `apropos`, but matches only keywords, whereas `apropos` searches the complete database (keywords and their description). As a result, `whatis` tends to produce a shorter (perhaps more relevant) output.

In some menu driven programs, for example when configuring your system services using `ntsysv` (or `setup`, or `linuxconf`), you may press F1 for info about what the particular service does.
The list of bash built−in commands can be obtained by typing `help` on the command line. Then help on any specific bash built−in command can be obtained by issuing, for example:

```
help cd
```

Bash is the standard command line "shell", i.e., the Linux equivalent of the DOS command−line processor "COMMAND.COM".

The KDE environment includes a GUI−based "help browser", which can be started by clicking the appropriate icon on the "Kpanel" (the system bar, normally at the bottom of the screen). This browser can be used to access the KDE−specific help as well as the system manual pages. The Gnome desktop contains a similar help system.

If you want to learn about the many packages that come on your CDs in rpm format, you may want to use the GUI−based kpackage (type `kpackage` in an X−terminal) to browse through the packages, display the info that they contain, and install them if you wish (the installation has to be done as root). In place of `kpackage`, older distributions use `glint` (RH5.2) or `gnorpm` (RH6.0), which are slower and less convenient.

2.3 Any dictionary of terms?

This one is a rather maximalist one : "The New Hackers Dictionary" aka "Jargon file": http://www.tuxedo.org/~esr/jargon/jargon.html. It is not only an excellent resource, but also highly entertaining reading. Recommended.

To add entertainment to entertainment, here is another link I like: "A Girl's Guide to Geek Guys": http://www.lairgauche.com/geekguy.html. If you are of the other sex, you might prefer: "A Guy's Guide to Geek Girls": http://www.eecis.udel.edu/~masterma/GuideToGeekGirls.html.

A rather complete list of computer−related abbreviations and acronyms is found at http://www.geocities.com/ikind_babel/babel/babel.html

2.4 Web Search

Currently, the best websearch engine is Google, amazing what you can find with it. Google is *wow* fast, because it runs on Linux, no kidding. Try: http://www.google.com/. For a test, do an egosurf (type in the search box: your last name and a word of your choice). Google can be used to find almost anything relevant to Linux (or anything else) on the net. Just type−in a few keywords to find the Linux documentation you need.

2.5 Newsgroups

This can be an intimidating place to be−−the world's strangest wackos seem to be all represented in the newsgroups. I just choose to ignore the stupid or offensive postings or e−mails. For the malicious ones, I make an exception and inform the system administrator at their originating e−mail provider. Advertisements which I receive after posting to a newsgroup get deleted before reading−−I know I am not the only one doing this, so please mark your subject line clearly if you want your e−mail to be read, particularly if your e−mail address contains the string "aol".

Despite their drawbacks, newsgroups can be a very efficient way of finding the information you need.

Before going to the newsgroups, I would highly recommend the Google news archives (http://groups.google.com/ , once known as DejaNews). This is a huge archive of newsgroup postings and you can search it using nice search tools. This way, you can often find an answer to your question without going through tons of trash, and without exposing yourself to anger after posting a question which "was already asked ten times this week". You may be surprised by the amount of information available through the google archive.

There are several newsgroups devoted to Linux and they seem much better than other newsgroups (maybe they are better policed by the Linux experts?). Here is a short list:

news:comp.os.linux.announce (moderated−−the postings are done by a moderator, who reviews them prior to the posting. Inspect the footer of any message for info on how to post.)
news:comp.os.linux.setup
news:comp.os.linux.hardware
news:comp.os.linux.security
news:comp.os.linux.misc (miscellaneous)
news:comp.os.linux.advocacy (Use this one for discussion of pros and cons of Linux and perhaps a comparison of Linux with other operating systems. This is an excellent newsgroup if you like getting into endless arguments).
news:alt.linux.sux (Here you can read/write really all opinions on Linux.)
news:comp.os.linux.networking
news:comp.os.linux.x (X−windows)
news:comp.os.unix (general UNIX newsgroup)

Please note that there is a newsgroup etiquette ("netiquette"), and you risk rejection and perhaps expose yourself to flames if you choose to break it. The major points:

- Don't post on a topic that is unrelated to the subject of the newsgroup;
- Don't post to many newsgroups at the same time (cross−post);
- Use plain ASCII, don't post attachments, pictures, html, etc.;
- Don't advertise (particularly commercial products);
- Don't write UPPER CASE ONLY − THAT GETS YOU KILL FILED.
- Use a simple descriptive subject ("HELP" isn't going to work) and briefly explain your problem. Include distribution and version number, and identify the hardware (USR 56K modem is wrong − list the model number).

To read newsgroups (also called usenet), you have to configure your access to a newsserver. The simplest may be to use mozilla ("edit"−"preferences"−"mail and newsgroups") and specifying the news server (your Internet Service Provider, ISP, should have given the name of the server) and then add the appropriate newsgroup to your list of local "mailboxes". If you don't know the name of the news server, try: "news.my_isp_provider_name.and_domain", or perhaps just "my_isp_provider_name.and_domain".

For news reading, I prefer `knode&` (type in the X terminal). Installation and learning newsgroups was certainly worth my effort.

2.6 Any Linux Internet links?

There are surely thousands of Internet sites devoted to Linux. Here are some Linux links which I like, in no particular order. If you need something else, you should find a useful pointer on one of these pages.

http://sunsite.dk/linux−newbie/	Master site for this document (LNAG). Bookmark it.
http://www.linuxdoc.org/docs.html	Linux Documentation Project−−Home for the many FAQs, Howtos, Minihowtos and Guides. Always up−to−date.
http://www.kalug.lug.net/linux−admin−FAQ/	Linux Admin FAQ (the non−Newbie).
http://members.aa.net/~swear/pedia/learning−linux.html	Gary's Encyclopedia−−Learning Linux. Bookmark it.
http://jgo.local.net/LinuxGuide/	Josh homepage. Good resource for learning Linux.
http://www.control−escape.com/	This site seems good for newbies!
http://www.linuxninja.com/linux−admin/	Linux administration made easy (LAME). Recommended.
http://metalab.unc.edu/mdw/index.html#guide	Lots of Linux documentation. Bookmark it.
http://www.frankenlinux.com	Another help site for newbies
http://www.easyfeed.com/~jgo/LinuxGuide/	Yet another newbie guide
http://www.slashdot.org/	Discussions for nerds, hackers, gurus, etc. (= /.)
http://www.freshmeat.org/	Update on today's releases of Linux software
http://linuxtoday.com/	Linux news−−excellent daily reading. Bookmark it.
http://www.llp.fu−berlin.de/	"The Linux Lab Project." Data acquisition and other interesting material for those in science.
http://www.linuxberg.com/	Linuxberg. Big portal. They have everything there. I like their rating of Linux software and am installing only packages that received 5 penguins ;−) . Bookmark it.
http://counter.li.org/linuxcounter_eng.html	The Linux counter. Register yourself as a linuxer!
http://www.ap.univie.ac.at/users/havlik/Album/Linux−Counter/	See Dennis Havlik's impressive maps on Linux growth and geographical distribution.
http://www.cl.cam.ac.uk/users/iwj10/linux−faq/index.html	Linux FAQ.
ftp://sunsite.unc.edu/pub/Linux/	Tons of Linux software at the Sunsite archive. Bookmark it.
http://stommel.tamu.edu/~baum/linuxlist/linuxlist/linuxlist.html	Linux applications.
http://www.boutell.com/lsm/	Linux applications.
http://www.linuxlinks.com/Software/	Linux applications.
http://directorysearch.mozilla.org/Computers/Operating_Systems/Linux/	Great new portal (better than yahoo) with excellent links for Linux newbies.
http://dir.yahoo.com/.../Unix/Linux/	Yahoo's entries for Linux. Looks very corporate−they refuse to add this guide!
http://www.debian.org/	Debian Linux site.
http://hardware.redhat.com/hcl/genpage2.cgi	Linux hardware compatibility list.
http://metalab.unc.edu/mdw/links.html	Lots of useful Linux links
http://www.cse.unsw.edu.au/~conradp/linux/	Scores of excellent links.
http://www.linuxstart.com/documentation/	More links to Linux documentation.
http://www.linuxlinks.com/	Even more Linux links.
http://www.gnu.org/	Master GNU site (GNU's−Not−Unix. This is a recursive definition).
http://www.redhat.com/	The Red Hat site. It is typically too busy to bother.
http://www.cs.Helsinki.FI/u/torvalds/	Linus Torvalds home page.

2.7 Source code−−the ultimate resource

The ultimate reference under Linux is the source code. If you installed it (comes with standard distributions; we really recommend its installation if you have enough hard drive space), it is in `/usr/src/linux`(the kernel source) and `/usr/src/RPM/sources`(the source code for the balance of the rpm packages). How can the source code be of use to a newbie? Well, it contains all the comments and documentation down to the smallest detail. For example, later in this guide, we show how to read/set up some of the kernel runtime parameters via the /proc filesystem. You can read the complete documentation for all the available parameters using:

```
less /usr/src/linux/Documentation/proc.txt
```

To install kernel sources, I would select the appropriate rpm package during my main installation. To install sources for other packages that came with my distribution, I would put the "Source CD" into the cd drive and do something like (as root, with RedHat CD):

[install the source code for the gnumeric spreadsheet from the cd to the harddrive]
```
su
cd /mnt/cdrom/SRPMS/
rpm -ivh gnume<Tab>
```
[unzip the sourcecode which I just installed]
```
cd /usr/src/RPM/SOUR<Tab>
tar -xvzf gnumer<Tab>
```
[read the code for statistical functions in gnumeric]
```
cd gnumeric/src/functions
less fn-stat.c
```

This is truly the ultimate reference on how a particular spreadsheet function works, no kidding.

Go to part 3: Basic Operations FAQ
Back to the main page

Part 3: Basic Operations FAQ

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section:

3.1 Basics

3.1.1 Filenames

Linux is case−sensitive. For example: `myfile`, `Myfile`, and `myFILE` are three different files. Your password and login name are also case−sensitive. (This follows tradition since both UNIX and the "c" programming language are case−sensitive.) Naming conventions for files and directories are identical. All the files and directories which I create (for myself, as a user) are lower−case, unless there is a very special reason to make it different. Most of Linux commands are also all lower case.

Filenames under Linux can be up to 256 characters long and they normally contain letters, numbers, "." (dots), "_" (underscores) and "−" (dashes). Other characters are possible but not recommended. In particular, it is not recommended to use special metacharacters: "*" (asterisk), "?" (question mark), " " (space), "$" (dollar sign), "&" (ampersand), any brackets, etc. This is because metacharacters have special meaning to the Linux shell (shell is something like `COMMAND.COM`, the command processor under DOS). It is possible to have a space in the filename, but we don't recommend it either−−we use underscore "_" instead.

It is not possible at all to have '/' (slash) as a part of the filename because '/' is used to represent the top of the directory tree, and as a separator in the pathnames (the same as '\' is in DOS).

Like in DOS, I cannot have a file called `.` or a file called `..` (dot or two dots)−−they mean "current" and "parent to the current" directory respectively, exactly like in DOS.

Here is the meaning of some metacharacters:
* = Matches any sequence of zero or more characters, except for "." (a dot) at the beginning of a filename.
? = Matches any single character.
[abC1] = Matches a single character in the enumerated set. In this example the set contains: 'a', 'b', 'C', and '1'.
[a−z] = Matches any lower−case letter.
[A−F] = Matches any upper−case letter from A to F.
[0−9] = Matches any single digit.
[a−zA−Z0−9] = Matches any letter (lower or upper case) or any digit.

The character \ (backslash) is also special. It makes the subsequent special character aquire literal meaning (read on).

Examples. This command will list any filename in the current directory, with the exception of filenames starting with "." (dot):

```
ls *
```

An equivalent to this command is to type just `ls` or `dir` (without the "*"). Files with names starting with "." are not shown because "." as the first character of a filename is not matched by "*". Think of files with names starting with "." as an equivalent of DOS hidden files. Use `ls −a` (list with the option "all") or `ls .*` to see these "dot" files. The "dot−files" are common in the user home directories and they typically contain user−level configurations.
This command will list any file (in the current directory) that contains a dot (except files starting with a dot):

```
ls *.*
```

This command will list any filename that contains two dots (except those starting with a dot):

```
ls *.*.*
```

Please note that Linux does not have "filename extensions" the way DOS does, but you can still use them. For example, I can have a file my_text.txt.zip. Some other DOS−kind file−naming features are completely absent ("Micros~1.doc" comes to mind).
This command will find (on the whole filesystem) any file with the extension "htm" optionally followed by any one more character:

```
locate *.htm?
```

This command will show all filenames in the current directory that start with "a" or "b", or any capital letter:

```
ls [abA−Z]*
```

This command will list any file starting with "a" and ending with "n"

```
ls a*n
```

Command line autocompletion. This is a great command line feature−−I use the [Tab] key a lot to save on typing. It makes it brisk to deal with long and complicated filenames. For example using such a filename on the command line is really not a problems, if I use autocompletion:

```
dir Eurosong\ 2000\ Olson\ Brothers\ −\ Fly\ on\ the\ wings\ of\ love\ \(denmark\).mp3
```

I just type

```
dir Eu<Tab>
```

and if there are no other files starting with "Eu", the rest of the filename is automatically typed for me. Otherwise, I would have to look at my choices (which are printed for me) and type one or two more characters to make the filename unambiguous. The backslashes in the name of the example song above show that the spaces are "literal", i.e., they spaces are part of the filename.

Problems with weird filenames. Most of these problems can be solved using autocompletion. Additionally, to manipulate files with names that do contain metacharacters, I may use a pair of '' (two apostrophes), so that the metacharacters are quoted and therefore the shell does not interpret their meaning. For example, to rename a file `my file*` (contains space and asterisk), I would issue:

```
mv 'my file*' filename_without_weird_characters.txt
```

Please note that I use a pair of ' (apostrophes) for quoting. Quoting with a pair of " " (quotation marks) is generally weaker than quoting with ' ' . If you use " (quotation marks) some metacharacters may get interpreted by the shell (altering their meaning).

Following UNIX tradition, on Linux, one may create files with names contaning almost any character, including non−printable (control) characters. Those are very infrequent, but if you encounter such a file, it can make you feel really weird. I would rename such a file using a carefully positioned metacharacter. I would use ls first to try if my action indeed targets the desired file, and then rename the file (using the move "mv" command):

```
ls −l myfile*y.html
mv myfile*y.html myfile.html
```

(I assume that the non−standard character(s) are between the letters e and y.)

As an example of the perhaps weirdest problems that you might face when using non−recommended characters in a filename, try creating a file with a name starting with a dash and then remove it−−there seems to be no way to do it (because a dash normally introduces command options). E.g., the command

```
dir > −junk
```

will create such a funny file (like in DOS, the symbol ">" redirects the output from the `dir` command to a file named "`−junk`"). Since the regular way of removing the file `−junk` does not work, I use:

```
rm ./−junk
```

The "dot slash" at the beginning means "the current directory" and here serves just the purpose of hiding the leading dash so it is not interpreted as introducing an option to the `rm` command. The point here is that I would rather stick to traditional naming conventions than face the occasional complications.

Besides using autocompletion, apostrophes and quotes, I can manipulate files with weird names using \ (backslash). Backslash hides the special meaning of the subsequent character. For example, i can create a weird file with the name `*?[` using the following command:

```
touch \*\?\[
```

(The `touch` command creates an empty file or, if the file exists, updates its date/time of last modification.)

3.1.2 What are the different directories for?

Linux filesystem tree is large and complicated. It will vastly improve your skills if you familiarize yourself with it.

Briefly, typical Linux contains five filesystems. These filesystems can reside on a single or different physical hard drives and/or hard drive partitions, depending on the size and need of your system. (A single filesystem can also be distributed between different physical devices, if needed.)

The root "`/`" filesystem contains basic operating system and maintenance tools. The content of this filesystem should be sufficient to start up the system and perform emergency maintenance and repairs if they were necessary.
`/usr` filesystem contains all commands, libraries, documentation, and other files that do not change during normal operation. This will also contain major applications that come with your Linux distribution, for example Netscape.
`/var` filesystem contains files that change: spool directories, log files, lock files, temporary files, and formatted (on use) manual pages.
`/home` filesystem contains user files (users' own settings, customization files, documents, data, mail, caches, etc). The contents of this directory should be preserved on an operating system upgrade.
`/proc` filesystem contains entirely illusionary files. They don't really exist on the disk and don't take up any space there (although `ls −l` will show their size). When viewing them, you really access information stored in the memory. It is used to access information about the system.

The parts of the **root filesystem** are:
`/bin`−−executables (binaries) needed during bootup that might be used by normal users.
`/sbin`−−executables (system binaries) not intended for use by general users (users may still use them, but this directory is not on their PATH).
`/etc`−−system−wide configuration files for your operating system.
`/root`−−the home directory of the system administrator (called super−user or root).
`/dev`−−device files. Devices appear on Linux as files so that hardware is abstracted and it is easy to write to them or read from them.

`/mnt`−−mount points for removable media (floppy, cdrom, zipdrive), partitions of other operating systems (e.g. MS Windows), network shares, and anything else that is mounted on the file system temporarily. It normally contains a separate subdirectory for each mounting share. The contents of these drives/shares appear in these subdirectories−−there are no drive letters on Linux.
`/lib`−−shared libraries for programs that reside on the root filesystem and kernel modules.
`/boot`−−files used by the bootstrap loader (LILO or GRUB), the thing that loads first when the computer is booted and perhaps gives you the option of which operating system to boot, if you have more than one OS on your computer). It typically also contains the Linux kernel (compressed, file `vmlinuz)`, but this can be stored somewhere else, if only LILO is configured to know where it is.
`/opt`−−optional large applications, for example kde under RedHat 5.2 (under RedHat 6.0, kde is distributed as any other X−windows distribution, main executables are in the `/usr/bin` directory).
`/tmp`−−temporary files. This directory may clean up automatically.
`/lost+found`−−files recovered during the filesystem repair.

The most interesting parts of the **/usr filesystem** are:
`/usr/X11R6`−−X−windows system (version 11, release 6).
`/usr/X11`−−the same as /usr/X11R6 (it is a symbolic link to /usr/X11R6).
`/usr/X11R6/bin` −−lots of small X−windows apps, and perhaps symbolic links to the executables of some larger X−windows applications that reside in their own subdirectories somewhere else).
`/usr/doc`−−Linux documentation (on newer systems, this moved to /usr/share/doc).
`/usr/share` −−Data independent from your computer architecture, e.g., dictionary words.
`/usr/bin` and `/usr/sbin`−−similar to their equivalents on the root filesystem (`/bin` and `/sbin`), but not needed for basic bootup (e.g. during emergency maintenance). Most commands will reside here.
`/usr/local`−−the applications installed by the local administrator (perhaps each application in a separate subdirectory). After the "main" installation, this directory is empty. The contents of this directory should survive normal re−installation or upgrade of the operating system.
`/usr/local/bin`−−perhaps smaller "user"−installed executables, plus symbolic links to the larger executables contained in separate subdirectories under `/usr/local` .

It is important to understand that all directories appear in a single directory tree, even if the directories are contained on different partitions, physical drives (including floppies, etc), or even if they are distributed over the network. Therefore, there are no DOS−type "drive letters" under Linux. What would be a "drive" under DOS or MS Windows, appears on Linux as a subdirectory in a special "mounting" location.

The directory system is well−established and standard on most Linux distributions (the small differences are being currently addressed by the Linux Standard Base). It is also quite similar to that found on typical commercial UNIX systems.

To summarize:

- Users always save their files to the directory `/home/user_login_name` (and its subdirctories)
- The local administrator most likely installs the "additional" software under the directory `/usr/local` and makes a link to the main executable in /usr/local/bin.
- System settings are all in the directory /etc.
- It is not a good idea to temper with the content of the root directory ("/") or of the directory `/usr`, unless I really know what I want. These directories are best left as they came with my Linux distribution.
- Most tools and applications installed on my system are in the directories: /bin, /sbin, /usr/bin, /sbin, /usr/X11/bin, /usr/local/bin.
- All the files are in single directory tree. There are no drive letters.

More about the `/proc` filesystem (only for really curious).

The `/proc` "pseudo" file system is a real−time, memory−resident file system that tracks the state of the operating system kernel and the processes running on your computer. The `/proc` file system is totally virtual, i.e., it is not written on any particular disk or other persistent media, it exists only in the computer memory, and it is constantly updated to reflect any changes to your system. The size of the `/proc` directory is always zero and the last modification time is the current date. In some cases, it is possible to change your system settings by manually changing the contents of files in the `/proc` filesystem. Many Linux utilities use the `/proc` filesystem as the source of their information, e.g., `dmesg`, `ps`, `top`.

Contents of the `/proc` filesystem.

Directories with numerical names like "1" "170" "4908" are IDs of the processes running on your computer. Each directory contains several files, e.g.,: `cmdline` (contains the entire command line that was used to envoke the process), `cwd` (symbolic link to the cwd of the process), `environ` (the environment variables defined for this particular process in the form VARIABLE=value), `exe` (a symbolic link to the executable file that the current process is linked to), `fd` (a list of the file descriptors opened by the process),`maps` (a named pipe that can be used to access the process memory), `root` (a symbolic link to the directory which is the root file system for the particular process), `stat` (info on the status of the process).

Other files in the `/proc` filesystem:
`/proc/cpuinfo` −−information about the processor, such as its type, make, model, and performance.
`/proc/devices` −−list of device drivers configured into the currently running kernel.

`/proc/dma` −−DMA channels being used at the moment.
`/proc/filesystems` −−filesystem types configured into the kernel.
`/proc/interrupts` −−interrupts in use, and how many of each there have been.
`/proc/ioports` −−I/O ports in use at the moment.

For example, I can read the cpu info on my system using the following command:
`cat /proc/cpuinfo`

3.1.3 How do I run a program?

Typing the name of the executable on the command line doesn't help? There are three possibilities.

The first possibility: I did not type the name of the executable correctly. Check the case−−Linux is case sensitive! For example, typing "Pico" or "PICO" will not start the `pico` editor.

The second possibility: maybe the program is not on my PATH? Under Linux (or any UNIX), an executable must be on your PATH to run it, and the current directory is not on my PATH. Type the full path to the executable in front of the executable name, or do:

```
cd the_program_directory
./program_name
```

I must put the dot and slash in front of the program name or the program will not execute. (This is a security feature not to put one's current directory on the path. It makes "trojan horses" more difficult. A "trojan horse" is a malicious program that pretends to be something different than it really is.) The dot means "the current directory", and the slash "/" is a separator between the directory name and the filename (exactly as "\" in DOS).
I may check my path using:

```
echo $PATH
```

To learn how to change your PATH, or add your current directory to it, see the next answer.
If my executable is lost somewhere in the directory tree, I may want to find it using (for example):

```
find / −name "netscape"
```

to find a file named "netscape", starting the search from the root directory "/". You may be able to achieve the same result faster using:

```
locate netscape
```

(Locate runs faster because it relies on a pre−built database of files on your system. This database is updated by a background "cron" process that normally runs at night, so don't count on `locate` to find a file if you regularly switch off your computer for the night, or you are searching for a file that you have just installed.)
Please note that the PATH is normally different for root than for the regular users (root's PATH includes `/sbin` and `/usr/sbin` whereas users' don't). Therefore users cannot execute commands located in the "sbin" directories unless they specify the full path to the command. Also, if you become a superuser by executing the `su` command, you inherit the user's PATH, and to execute the command located in sbin, you need to specify the full path.
Conversely, if I need to learn where an executable which is on my PATH is located on your system (i.e., the executable runs by typing its name anywhere in the system, but I would like to know where it is located), I may use something like this:

```
which netscape
```

which will show the full PATH to the executable program called "netscape" (if one exists).

The third possibility: maybe the file is not executable. If it should be, change the permissions to make it executable. E.g. (as root or the user who owns the file):

```
chmod a+x my_file
```

will make the file "my_file" executable for all users. Check if it worked using:

```
ls −l  my_file
```

Read here if you don't understand the output of this command or the whole "third possibility".

Please note that under Linux (or UNIX), the file extension (for example .exe or .com or .bat) does not make the file executable. The file needs an "executable file access mode" which is not unlike a "file attribute" under DOS.

3.1.4 How can I change the PATH?

Typically, you don't have to change your PATH, but it very useful to understand what PATH is.

The PATH is the list of directories which are searched when you request the execution of a program. You can check your PATH using this command:

```
echo $PATH
```

which, on my system , shows the PATH for the user "yogin" to be:

```
/opt/kde/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/yogin/bin
```

The ":" is a separator, therefore the above PATH represents a list of directories as follows:

```
/opt/kde/bin
/usr/local/bin
/bin
/usr/bin
/usr/X11R6/bin
/home/yogin/bin
```

Here is the output from the command "echo $PATH" run on my system on the account "root":

```
/opt/kde/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin
```

You can change the PATH for all users on the system by editing the file `/etc/profile` and adjusting (as root) the line starting with "PATH=". I do it using the `pico` editor (as root):

```
pico −w /etc/profile
```

(The option −w turns off the wrap of long lines.)
Re−login for the change to take effect. To set up the PATH for an individual user only, edit the file `/home/user_login_name/.bash_profile` (please note the dot in front of the filename−−files starting with a dot are normally invisible, you have to use `ls −a` to see them).
If you really want to have the current directory on your PATH, add "." (dot) to your PATH. When used in the place when directory name is expected, a dot means "the current directory". The specification for the path in `/etc/.bash_profile` may then look like this:

```
PATH="$PATH:$HOME/bin:."
export PATH
```

This command takes the contents of the environmental variable called PATH (as set for all users in `/etc/profile`), and appends to it the name of your home directory as set by the variable HOME with an attached "/bin" and then a dot. Finally, the command assigns the resulting string back to the variable called PATH. It is necessary to use the command "export" after modifying PATH or any other user−environment variable, so that the variable is visible outside of the script that sets it.

3.1.5 How can I shutdown my computer?

Close all your programs saving the data as desired. From your GUI main menu (e.g., "K"), select "Logout". Then, from the logon screen, select: "System"−"Shutdown".

Alternatively, from a text terminal, press `<Ctrl><Alt><Del>` (the "three−finger salute", you press the three keys simultaneously), wait for the shutdown process to complete, and turn off your machine only after it starts rebooting again. If you are in X−windows, first switch to a text terminal by pressing `<Ctr><Alt><F1>` (three keys simultaneously).

Never turn off your machine without the proper shutdown or else you may have disk error messages next time you boot. (Typically, the errors resulting from improper shutdown will be repaired automatically during the next boot, but occasionally more serious problem may result, and then you may need to repair the files manually or re−install!)

If you prefer your computer to go to a halt after you press <Ctrl><Alt><Del> (instead of the default reboot), you can set this up by editing the file /etc/inittab. This file specifies something like this:

```
# Trap CTRL−ALT−DELETE
ca::ctrlaltdel:/sbin/shutdown −t3 −r now
```

As root, replace the option "−r" to "−h" so that the same fragment reads:

```
# Trap CTRL−ALT−DELETE
ca::ctrlaltdel:/sbin/shutdown −t3 −h now
```

The line starting with "#" is just a comment (it is for humans, it does not have any effect on the computer). The option "−t3" tells the shutdown command to wait 3 seconds before it starts killing processes. The options "−r" and "−h" stand for "reboot" and "halt" respectively, so they perform a shutdown to reboot or a shutdown to a system halt.

Root can also use the shutdown command directly. This command can be used for either local or remote shutdown of your computer, but is used mostly for remote shutdown when the local keyboard is not available so you cannot use <Ctrl><Alt><Del>. It can also be very useful if a program hangs so that the keyboard is no longer functional. For example:

```
telnet name_of_machine_with_no_operable_keyboard
```
[login as a user]
```
su
```
[give password]
Now either execute `ps axu |more,` find the process id of the offending command in the ps output and do

```
kill pid_of_offending_process
```

or reboot your machine with:

```
/sbin/shutdown −rn now
```

This command will shutdown really fast, bypassing standard (longer) shutdown procedure−−useful when the system becomes really buggy (the option −n will make "shutdown" kill all the processes before rebooting).

Please note that for security reasons, you cannot login to a remote machine as root (e.g., over the telnet). You have to login as a user and then execute su and give a password to become a super user (root).

The shutdown command may also be used to execute a shutdown later. E.g. (as root):

```
/sbin/shutdown −r 23:59
```

will reboot the system 1 minute before midnight. I could also use:
```
/sbin/shutdown −r +1
```

to shutdown 1 minute from now. I can cancel a scheduled shutdown with:

```
/sbin/shutdown −c
```

If the shutdown command is too long for you, you may want to try these two commands, which do exactly what their names suggest (as root):

```
reboot
halt
```

A fancy way to shut down your computer is to switch your system to the runlevel 0 (for halt) or runlevel 6 (for reboot). Try it using (as root):

```
init 0
```

The meaning of the different runlevels is explained in the file /etc/inittab and here.

3.1.6 How do I deal with a hanged program?

Buggy programs do hang under Linux. A crash of an application should not, however, affect the operating system itself so it should not be too often that you have to reboot your computer. Linux servers are known to run for more than a year without a reboot. In our experience, a misbehaving operating system may be a sign of hardware or configuration problems: we repeatedly encountered problems with the Pentium processor overheating (the fan on the Pentium did not turn as fast as it should or it stopped altogether, the heat sink on the Pentium was plugged with dirt), bad memory chips, different timing of different memory chips (you may try re−arranging the order of the chips, it might help), wrong BIOS setup (you should probably turn off all the "advanced" options, Linux takes care of things by itself). The "signal 11" error message is typically (99%) associated with hardware problems and is most likely to manifest itself when you perform computing−intensive tasks: Linux setup, kernel compilation, etc. If your Pentium has the tendency to overheat (very common for early Pentiums), here are some tips to keep it cool, particulary during hot weather: clean the processor heat sink, replace the processor fan, operate the computer with the cover off and aim an extra fan inside, increase the processor "wait−state" in the computer BIOS, don't overclock, decrease useless load, e.g., replace this super−fancy screen saver with a blank screen.
Not really hanged. Some programs might give the uninitiated impression of hanging, although in reality they just wait for user input. Typically, this happens if a program expects an input filename as a command line argument and no input filename is given by the user, so the program defaults to the standard input (which is console). For example, this command

```
cat
```

may look like it's hanged but it waits for keyboard input. Try pressing <Ctrl>d (which means "end−of−file") to see that this will satisfy the `cat` command. Another example: I have seen many questions on the newsgroups about the "buggy" `tar` command that "hangs" when trying to uncompress a downloaded file, for example:

`tar −zxv my_tar_file` [wrong!]

This waits for user input too, since no option "−f filename" was specified so "my_tar_file" was not recognized as a filename. The correct command is:

```
tar −zxvf my_tar_filename
```

Please note that the filename must follow immediately after the option "f" (which stands for "filename). This WILL NOT work (very common mistake):
`tar −zxfv my_tar_file` [wrong!]

Any program (hanged or not) can be killed.

A text−mode program in the foreground can often be killed by pressing <Ctrl>c. This will not work for larger applications which block the <Ctrl>c, so it is not used on them accidentally. Still you can get back in control either by sending the program to the background by pressing <Ctrl>z (no guarantee this will work) or switching to a different terminal, for example using <Ctrl><Alt><F2> and login as the same user that hanged the program (this should always work). Once you are back in control, find the program you want to terminate, for example:

```
ps
```

This command stands for "print status" and shows the list of programs that are currently being run by the current user. In the ps output, I find the process id (PID) of the program that hanged, and now I can kill it. For example:

```
kill 123
```

will kill the program with the process id (PID) of "123".
As user, I can only kill the processes I own (this is, the ones which I started). The root can kill any process. To see the complete list of all processes running on the system issue:

```
ps axu | more
```

This lists all the processes currently running (option "a"), even those without the controlling terminal (option "x"), and together with the login name of the user that owns each process ("u"). Since the display is likely to be longer than one screen, I used the "more" pipe so that the display stops after each screenful.

The `kill` command has a shortcut `killall` to kill programs by name, for example:

```
killall netscape
```

will kill any program with "netscape" in its name, while

```
killall pppd
```

will surely disconnect any dial−up connection by killing the ppp daemon.

X−windows−based programs have no control terminals and may be easiest to kill using this (typed in an X−terminal):

```
xkill
```

to which the cursor changes into something looking like a death sentence; you point onto the window of the program to kill and press the left mouse button; the window disappears for good, and the associated program is terminated.

A shortcut to the last command is to press <Ctrl><Alt><Esc>, to which the cursor changes into something looking like a death sentence−−you point at the window of the offending program, click your mouse, and the window closes and the program is gone.

If your X−windows system crashes so that it cannot recover, or you just get stuck, it may be the easiest to kill the X−server by pressing <Ctrl><Alt><BkSpace>. After that, it might be a good idea to run `ps axu`, find any possible X−programs that might still be running, and kill them. If you don't do this, and there really is a misbehaving program that caused your X−windows to crash, it might cause trouble again.
If you have programs in the background, the operating systems will object your logging out, and issue a message like "There are stopped jobs". To override and logout anyway, just repeat the `logout` (or `exit`) command −−the background program(s) will be automatically terminated and you will be logged out.
Core files. When a program crashes, it often dumps a "core" into your home directory. This is accompanied by an appropriate message. A core is a memory image (plus debugging info) and is meant to be a debugging tool. If you are a user who does not intend to debug the program, you may simply delete the core:

```
rm core
```

or do nothing (the core will be overwritten when another core is ever dumped). You can also disable dumping the core using the command:

```
ulimit −c 0
```

Checked if it worked using:

```
ulimit −a
```

(This shows "user limits", the option "−a" stands for "all".) To make the option of disabling core dumps permanent for all users, edit the file `/etc/profile` (as root), where `ulimit` is set, and adjust the setting. Re−login for the changes to `/etc/profile` to take effect.
If you would like to see how a core file can be used, try (in the directory where you have a core file):

```
gdb −c core
```

This launches GNU debugger (gdb) on the core file "core" and displays the name of the program that created the core, signal on which the program was terminated, etc. Type "quit" to exit the debugger. To learn the meaning of different signals, try:

```
cat /usr/include/bits/signum.h |more
```

3.1.7 Command options

Most commands accept numerous "options". An option can be introduced with an "−" (dash). For example:

```
dir −l
```

shows me the listing of the current directory but in a long format (the default format is "short"). Multiple options can be introduced in two, equivalent ways:

```
dir −l −a
```

or

```
dir −la
```

Either of the above commands will show me the listing of the current directory in the long file format (option −l), and include all files in the listing, i.e., also the hidden files (option −a).

Most popular options are marked with one letter. This follows the traditional UNIX way of invoking options. There is also a new style, which looks like this:

```
dir −−help
```

Here, a single option is more than one character long, and it must be introduced with two dashes. The above command displays a brief help for the dir command, including the listing of all options. Because there are so many of those (more than a screenful), I would probably do:

```
dir −−help | more
```

3.2 Users, passwords, file permissions, and security

3.2.1 Home directories, root, adding users

The (almost) only place on the harddrive that normal users (non−root) can write to is their home directory, which is `/home/user_login_name`.

This "home" directory is for all user files: settings, program configuration files, documents, data, netscape cache, mail, etc. As a user, you can create subdirectories under your home directory to keep yourself organized. Other users cannot read your files or write to your home directory unless you give them permission to do so.
Normal users can also see, read and execute many other files on the system (besides their home directory), but normally they cannot modify or remove (delete) them.

The "root" (also called "super user") is a special administrative account that has the power to modify any file on the system. It is not a good idea to habitually work on your system as root−−if you do so, your mistakes can cost you dearly. Set up and use a normal user account for everyday work for yourself, another user account for your son, and yet another for your wife. The root account is typically the only account that exists on Linux after the initial installation. Thus you have to explicitly create "user" accounts for normal work for you Linux system.

A user account can be created by "root" using, for example:

```
adduser joe
passwd joe
[type the password for the user joe]
[retype the password for the user joe so as to avoid mistakes]
```

In the example above, first I logged in as root. Then, on the command line, I issued the command "adduser" with the parameter (argument) "joe". This created the account "joe" on my Linux computer. Then, I issued the command "passwd joe" to change the password for the user "joe" to something fairly secure. Now, I can tell "joe" what her initial password is, and she can login and change the password to her liking. Please note that the account name (user login name, "joe") and the password are case−sensitive.

Root can change any user's password, although s/he cannot read it. [Passwords are encrypted using a one−way encryption algorithm and only this encrypted version is stored on the system, in the file `/etc/passwd` (older systems) or `/etc/shadow` (newer systems), and the "open" version of the password is never stored. When you login, the password you type is encrypted again using the same one−way algorithm and compared with the already encrypted version stored in `/etc/passwd` or `/etc/shadow`.]

The separation of the administrator and user makes Linux systems secure and robust−−it even makes viruses under Linux difficult (the programs that a user runs can write only to his/her own directories, and therefore cannot affect the vital parts of the operating system). It is customary that the user changes his/her password immediately after the first login, for example:

```
passwd
(current) UNIX password: pass_OLD
New UNIX password: pass_NEW
Retype New UNIX password: pass_NEW
```

In reality, the password will not appear on the screen as you type it (for security reasons). Take your time if you are changing the password for the very first time−−it can be difficult to type "blind".

On the Linux system, the same password is used to:

- login on the text terminal,
- login from a graphical (GUI) screen into your desktop (KDE or GNOME),
- unlock a locked text terminal,
- unlock a password−protected screen saver on a GUI (for example, KDE or GNOME).

3.2.2 About password security

Weak passwords are probably the most common source of security problems. Even at home, you may expose yourself to serious trouble because somebody may be able to hack your computer when you browse the Internet and read/delete your files, or use your computer to do something really nasty to the local police computer network. Therefore, keep all your login names/passwords secure, even at home. Once somebody logs into your computer (even as an ordinary user), he may find it quite easy to gain root access (depending on how well−maintained/up−to−date your system is vs. how good a hacker s/he is).

Here are some examples of hazardous passwords:
− No password (possible!).
− The word "password" (wow, this one is really weak!).
− Your login name (The login and the password the same? Hmm.).
− Your first name or the first name of your daughter, son, husband, wife, girlfriend, or any other first name. The number of first names in use is quite limited−−just check the paperback book "what to name your baby". Don't assume that a first name you think of is secure because you are from India−−Canada is really a multinational society and the typical namelist seems to cover all kinds of first names.
− Your last name or any other last name. The number of last names is surprisingly limited! Just check the US census data to see that your "rare" last name from the abamamahaba island is very well represented in the US 89,000 of the most frequent last names (e.g., http://www.census.gov/genealogy/www/freqnames.html). Or just check the Toronto telephone book. Another proof that we are all one family :))
− The nickname of your dog, wife, canary or computer. (Very few nick names humans use, much fewer than last names!)
− Name of your favourite sports team, celebrity, toothpaste, or detergent. Avoid names of popular soccer teams like fire. Same with rock bands (music).
− Date of your birth, social security number, etc; Sequences of digits can be easily probed.
− Name of your company, department, workgroup, etc.
− Password written in the calendar on your desk or on the side of your computer.
− A password which you also use in an insecure public place, for example an Internet store or a mailing list. In general, you should use different passwords for places controlled by different organizations.
− Any word which is in the English dictionary. The English dictionary does not contain as many words as it might seem. A not−so−skillful hacker can easily set a program to encrypt all dictionary words (100,000? that's under 1 MB!) and then compare all the encrypted strings to your encrypted password. As a matter of fact, tools for the "dictionary attack" are readily available on the Internet. Try the program `crack` yourself to find how easy it is. Swear words or "cool" (colloquial) expressions make the password particularly vulnerable for cracking.
− Any other word, last name, first name, pet or swear word, no matter in what language. For a cracker, to cover most languages is only a small overhead if he already covered one. How many significant languages are out there? 40? The cracker just grabs a few more files and appends it to his cracking list. The point here is that the subset of words that humans normally use if far far below the theoretical limit of the random combination of characters.
− Any of the above with an addition of a number/letter at the beginning or the end. "yuoping1" is really a very weak password.

A good password is relatively long (minimum 6 characters, some experts even recommend minimum 10 characters), contains a mixture of letters (upper and lower case, if possible), numbers and special characters, and is changed quite regularly (8−16 weeks?).

Unfortunately, the better the password, the harder it is to remember. I solved this problem for myself by taking 10 minutes to invent my personal password "scheme". Say, I always start and end with the monkey (@) sign, and use two words connected with an exclamation mark, the last letter of each word is capitalized, e.g., "@whitE!housE@". Seems like an adequate password, and it is easy to remember once I know what my password rule is. If you are a memory genius, you may consider truly excellent passwords generated with `mkpasswd` :))

The system administrator can set the password policy (minimum length, requirement of special characters, password expiry) through the utility included in this configuration program (run as root):

```
linuxconf
```

under the menu "user account"−"policies"−"password & account policies". Normal users won't be able to set a password which is too short, is a dictionary word, or does not contain the prescribed number of non−alphanumeric characters (but root can change any password to anything s/he likes, s/he will only be given a warning).
Also make sure that any file that contains any password of yours (e.g., `/root/.kde/share/config/kppprc`) has proper,

secure permissions so that it cannot be read by anybody. For example, most likely you want:

```
chmod 600 kppprc
```

If you use an "over the phone" Internet connection for just a couple of hours a week, you may be fine even with a relatively weak password on your system. But please really reconsider your system security if you use a cable modem, or are otherwise connected to the Internet for a significant amount of time.

Most computer semi−literate use amazingly weak passwords. "Around 50 percent of computer users ***base*** passwords on the name of a family member, partner or a pet. Thirty percent look to a pop idol or sporting hero," reports CNN (http://www.cnn.com/2002/TECH/ptech/03/13/dangerous.passwords/index.htmll). Please note the underlined ***base***. Appending a digit to an obvious word hardly makes the password more secure.

3.2.3 I forgot the root password

Even if I never forget any passwords, I would still study this issue in detail because it can give me a hint on how my mother might be reading my ICQ chats history :−)
First method. The easiest way to solve your "forgotten root password" problem is to boot your Linux in the single−user mode, namely at the "lilo"prompt (during bootup) type:

```
linux single
```

This will make you "root" without asking for a password. Now, being root, you may change the root password using this command (no knowledge of the old password required):

```
passwd
```

If it strikes you as insecure, that's because no computer system is secure if other people have physical access to your hardware. Nevertheless, I did not like the "linux single" hole on my home computer and plugged it by adding the following lines to my `/etc/lilo.conf` file (at the end of the "image=" section):

```
password="my_password"
restricted
```

[This "lilo" password is required when, at the LILO prompt during bootup, somebody enters the word "linux" with any parameter (normal bootup without any parameters will still be possible without a password).] For the changes to `/etc/lilo.conf` to take effect, I must re−run the command `lilo` . Since my lilo password is not encrypted, I must make `/etc/lilo.conf` readable only for root:

```
chmod 600 /etc/lilo.conf
```

Second Method. Another way to solve the "lost−root−password" problem is to boot your computer from the Linux boot diskette or the CD. Then find your Linux root partition on the hard drive, mount it, and edit the file `/etc/shadow`. (I can do it because after booting from the floppy, I become root without being asked for a password.) In the password file, I erase the encrypted password for root (for example, using the pico editor), so it is empty.

> Information about a user account is kept in plain−text files: `/etc/passwd` and `/etc/shadow`.
>
> The file `/etc/passwd` contains the "world−readable" information about all accounts on my computer Each line in this file contains information about one account. Each line has 7 colon−delimited fields (this means 8 entries separated by colons): login name, the letter "x", the numerical user ID, the numerical primary group ID for the user, a comment field (for example, the full name of the user), the user's $HOME directory, the name of the shell (meaning the program that is run at login).
>
> The balance of information about accounts on my computer is stored in the file `/etc/shadow`. This file is more secure because normally only root can read it. In this file, each line describes "shadow" information about one account, and has 9 colon−delimited fields: login name, encrypted password, days since Jan 1 1970 that password was last changed, days before password may be changed, number of days after which the password must be changed, number of days before password expiration to warn the user, number of days after password expiry that account is disabled, number of days since Jan 1 1970 that account is disabled, and a reserved field.
>
> Some (older) UNIX or Linux systems do not contain the file `/etc/shadow` and store the encrypted user password in the second field of each line of the file `/etc/passwd` (the field which on newer systems contains

just the letter x).

For example, my `/etc/shadow` entry for "root" account may look like this:

```
root:$1$BuPbmLAz$1G7.evIChyqaEI0TlZp0F.:11071:0:99999:7:−1:−1:134540356
```

and after the password is erased, it looks like this:

```
root::11071:0:99999:7:−1:−1:134540356
```

Now, the root account has no password, so I can reboot the computer and, at the login prompt, type "root" and for password just press ENTER (empty, no password). After a successful login, I immediately set the password for root using the command:

```
passwd
```

> Apparently, despite deleting the password from `/etc/shadow` , the Debian distribution will not let you log in "passwordless" (enhanced security?). In such a case, what needs to be done is to replace the password in /etc/shadow with an encrypted password from another account, where you know the password. After that, you can login since you know the password.
>
> E−mailing an encrypted password may be also a secure way to set up an account for somebody remote: "I am setting up an ftp account for you on my server. Email me your encrypted password." After you receive the encrypted password, you insert it into the appropriate field in `/etc/shadow`. Now, the user can log in, since she knows the password, but nobody else can.

To make the "floppy access" to my system a little bit more difficult, I considered running a computer without a floppy drive :−) Unfortunately, Linux CDs are bootable these days. I set up my boot sequence (in the BIOS setup) so that the system boots from the hard drive before floppy and CDROM are tried, and added an "administrative" password on changes to the BIOS settings. Still, I worry that these BIOS passwords are very easily crackable, or that one could remove the small battery that sustains the BIOS setting. One could also remove my harddrive and connect it to another computer for reading :−) . I am thinking about installing an "encrypted file system" which is now available on Linux, but considering all the trouble associated with it, perhaps I will settle on locking my room :−) . If all this sounds paranoid to you, it probably is−−it just illustrates the point there is little computer security, even under Linux, if the potential cracker has a physical access to your hardware.

3.2.4 I forgot my user password

If a regular (non−root) user forgets his/her password, this is not a problem since root can change any password. For example (as root):

```
passwd barbara
```

will prompt for a new password for the user "barbara" (no knowledge of the old password required by root). If a regular user (non−root) wants to change his/her password, s/he will be asked for the old password first. (This is a security feature so nobody changes your password if you have left your terminal unattended.)

3.2.5 Disabling or removing a user account

A user account can be temporarily disabled or permanently removed.

To temporarily disable (lock) a user account, there is no need to change his/her password. Just put an asterisk "*" at the beginning of the second field (before the encrypted password) in the file `/etc/shadow` . The "*" means that no login is permitted for this account. When you want to restore the account, you just erase the asterisk and the user account is back in operation, with its old password.

> Here is an example entry from the file `/etc/shadow` with the password disabled for user "peter":
>
> ```
> peter:*$1$narMEFm6$fhA1puOU422HiSL5aggLI/:11193:0:99999:7:−1:−1:134539228
> ```

I could also lock a user account with the following command:

```
passwd peter −l
```

and unlock it with

```
passwd peter −u
```

To irreversibly remove a user account from my home computer, I do the following:
− login as root

− change my identity to the user to be removed, to check if there is any new important mail:

```
su doomed_user_login_name
mail
logout
```

− delete the user account and group

```
userdel doomed_user_login_name
groupdel doomed_user_login_name
```

Remove the user affiliation to any supplementary groups:

```
usermod −G doomed_user_login_name doomed_user_login_name
```

− force−delete the user home directory with all its contents including any subdirectories:

```
rm −fr /home/doomed_user_login_name
```

3.2.6 I have file permission problems. How do file ownership and permissions work?

Linux (the same as any UNIX) is a secure, multiuser operating system, and this creates a level a complexity with "files permissions". Trouble with file permissions can lead to unexpected and nasty problems. Understanding file permissions is of uttermost importance to be able to administer any multiuser operating system (be it UNIX, WinNT, or Linux). My advice would be: learn the system of Linux (or any UNIX) file permission conventions; you will not regret it.

File owners. Each file (or directory) belongs to an owner (normally a login name) and to a group. The owner is typically the person who created (or copied) the file. The group often consists of one person−−the owner, and has a name identical to that of the owner, but it does not need to be so. A file can be removed (erased) only by the owner of the file, or a member of the group that owns the file, or the root. Other users, however, may be able to modify or erase the contents of the file if they are given permission to do so−−read on. The owner and group that owns the file will be shown in the output from the `ls −l` command (="list in the long format"). For example, the command:

```
ls −l junk
```

produced this output on my screen:

```
−rwx------   1 yogin    inca           27 Apr 24 14:12 junk
```

This shows the file "junk", belonging to the owner "yogin" and to the group "inca".

The ownership of a file can be changed using the commands `chown` (change owner) and `chgrp` (change group), which are normally executed by root:

```
chown peter junk
chgrp peter junk
ls −l junk
```

After executing the above 3 lines, the command `ls−l junk` produces this output on my screen:

```
−rwx------   1 peter    peter           27 Apr 25 20:27 junk
```

Changing file ownership comes handy if you move/copy files around as root for use by other users. At the end of your housekeeping you typically want to hand the file ownership over to the proper user.

File permissions. Now, an owner of a file can make the file accessible in three modes: read (r), write (w) and execute (x) to three classes of users: owner (u), members of a group (g), others on the system (o). You can check the current access permissions using:

`ls −l filename`
If the file is accessible to all users (owner, group, others) in all three modes (read, write, execute) it will show:

```
−rwxrwxrwx
```

Skip the first "−" (it shows the type of file, and is "−" for normal files, "d" for directories, "l" for links, "c" for character devices, "b" for block devices, "p" for named pipes i.e. FIFO files, "f" for stacks i.e. LIFO files). After the initial "−" character, the first triplet shows the file permission for the owner of the file, the second triplet shows the permissions for the group that owns the file, the third triplet shows the permissions for other users. A "no" permission is shown as "−". Here is an output from the `ls −l` command on a file that is owned by root, for which the owner (root) has all permissions, but the group and others can only read and execute:

```
drwxr−xr−x   2 root     root        21504 Apr 24 19:27 dev
```

The first letter "d" shows that the file is actually a directory.

You can change the permissions on a file which you own using the command `chmod` (="change mode"). For example, this command will add the permission to read the file "junk" to all (=user+group+others):

```
chmod a+r junk
```

In the command above, instead of "a" (="all"), I could have used "u", "g" or "o" (="user", "group" or "others"). Instead of "+" (="add the permission"), I could have used "−" or "=" ("remove the permission" or "set the permission"). Instead of "r" (="read permission"), I could have used "w" or "x" ("write permission" or "execute permission").
Second example. This command will remove the permission to execute the file "junk" from others:

```
chmod o−x junk
```

Instead of letters, one can also use numbers to specify the permissions. To understand how it works, look at this:

```
execute=1
write=2
read=4
```

The total permission for a class of users is the sum of the three. Thus:

```
0 = no permissions at all(neither to write, nor to read nor to execute)(common)
1 = execute only (seems unusual)
2 = write only  (seems unusual)
3 = write and execute (seems unusual)
4 = read only (common)
5 = read and execute (common)
6 = read and write (common)
7 = read, write and execute (common).
```

The permission for all three classes of users (owner, group, others) is obtained by gluing the three digits together one by one. For example, the command:

```
chmod 770 junk
```

will give the owner and the group the completto of permissions, but no permissions to others. The command:

```
chmod 666 junk
```

gives all three classes of users (owner, group, others) the permissions to read and write (but not execute) the example file named "junk". Please note the "666". It is quite often used and, for at least one person I know, it is proof that Linux (any UNIX for that matter) is the work of the devil >:−0.
This command:

```
chmod 411 junk
```

would give the owner the permission to read only, and the group and others to execute only. This one does not seem useful, but might be funny, at least for those North American Linux users who dial 411 (telephone number) for directory assistance. Mail me if you can think of any other funny permissions (perhaps 007?).
The numerical way of representing file permissions is called "octal" because the numbers have the base 8 (the decimal system's base is 10). The highest digit in the octal system is 7 (the octal system has eight digits: 0 to 7, analogous to the decimal system having ten digits: 0 to 9). The octal representation is really a convenient notation for the binary representation of file permissions, where each permission is flagged as "set" or "denied" with a one or zero and the total is represented as a string of zeroes and ones, as in this diagram:

```
user class:                                    owner  group others
example permissions:                            rwx    rw-    r--
absent permissions:                             ---    --x    -wx
binary representation of the permissions:       111    110    100
octal representation  of the binary:              7      6      4
```

Permissions for directories.
The meaning of the permissions is different for directories than it is for "normal" files. For normal files: r=permission to read the contents of the file, w=permission to modify the contents of the file, and x=permission to execute the file.
For directories: r=permission to list the filenames in the directory, w=permission to create or delete files in the directory, and x=permission to access the directory. Otherwise, the permissions are set the same way for directories as they are for normal files.

Default file permissions with umask. When a new file is created, it is given default permissions. On my system, these are:

```
-rw-r--r--
```

This means that files created by a user can be read and written by this user; the group and the others can only read the file. Still, on my default RedHat system, users cannot read the files in the other users' home directories because the permissions on the home directories are:

```
drwx------
```

I can check the default file permissions given to my newly created files using:

```
umask -S
```

(The option "-S" stands for "symbolic" and tells `umask` to display the permissions in an easy-to-read form, instead of the default numeric mode.)
I can change the default file permissions for newly created files using a command like:

```
umask u=rw,g=,o=
```

which will give the owner the read and write permissions on newly created files (r+w), and no permission to the group and others.
Using numbers to set default permissions with umask is more tricky. The number shows the permissions that you take away for users (opposite to `chmod`). Thus:

```
umask 000
```

will give full permissions to everybody on newly created files. The next example gives read and write permissions to the owner, and zero permissions for everybody else (seems that's what one might want):

```
umask 177
```

To make the settings permanent for all users on the system, adjust the appropriate line(s) in the file `/etc/profile` .

3.2.7 My mp3 player chokes. The sound is kind of interrupted (how to set suid).

The MP3 player might not be given enough processor power (it requires a lot of it). It could be that your computer is lousy. Or you might be running too many cpu-intensive programs at the same time. Or, most likely, you may need to run the player with a higher priority. (The priority of a program can be set with the command `nice` -- see `man nice` or `info nice`). Try to run the player as root--programs run by root are given higher priority than those run by normal users. If this solves the "interrupted music" problem, set the "suid" on the executable so all users are given the "effective user id" of the file ower (normally root) when running it, for example:

```
chmod a+s /usr/bin/xmms
```

will do the trick for the xmms program. The output from

```
ls -l /usr/bin/xmms
```

on my computer is now:

```
-rwsr-sr-x  1 root  root  908k Feb 22  2000 /usr/bin/xmms
```

The first "s" indicates that the substitute−user−id (suid) bit is set. The second "s" indicates that the substitute−group−id (sgid) is also set. Thus anybody who executes `xmms` is given the effective user id of the program owner and effective group id of the owner group, which in the example above is the user "root" and the group "root".

Setting the suid for a program could possibly become a security hole in your system. This is unlikely the case on a closed home network and when setting suid for a program of which the origin is well traceable. However, even at home, I wouldn't suid a piece of code of which the origin is uncertain, even if the setup instructions urged me to do so. Also, it is definitely a very bad idea to suid too many executables on your system−−it defies the whole idea of UNIX security.
Some programs do, however, require suid for proper functioning, for example `kppp` (the popular modem "ppp" connection utility under the KDE graphical−user−interface desktop). This is because they require direct access to the hardware−−something only root is allowed to.

If you have constant problems with a smooth performance of your system, or some "real time hardware" (e.g., CD writer) tends to crash, try to reduce the number of daemons on your Linux system. Run (as root) `setup` (RH specific command) and disable all the "services" that you don't really require. Ultimately, you can switch to the command line, shut down the GUI (command `init 3` as root), and then the performance should surely be better even.

For those who need (like) their Linux to be a "universal" operating system (workstation, server, office computer, game box, mulimedia, etc, everything at the same time), there are dedicated Linux kernel patches: "low latancy patch" and "pre−emptive kernel patch" which aggresively atack the "latency" problem that overloaded systems exhibit.

3.3 Job scheduling with "&", "at", "batch", and cron

3.3.1 How do I execute a command in the "background"?

Using the "&" at the end of the command. For example, this will start `licq` (an icq client) in the x−terminal in the background, so that after issuing the command, my x−terminal is not blocked:

```
licq &
```

The process identification number, *job_number*, is printed on the screen, so you can use it with related commands. The related commands are `fg` *`job_number`* (="foreground", bring the background process back to my immediate view/control, restart it if it was stopped), `bg` *`job_number`* (="background", send the process to the background, restart if it was stopped, exactly as if it was started using &), `<Ctrl>z` (send the current foreground process to the background and stop it), `jobs` (list the active jobs), `kill` *`process_ID`* (terminate the process, use the conmmand `ps` to find the *process_ID* of the process to kill).

To make a background process keep running after you disconnect, you may use the nohup (="no hungup"), for example:

```
nohup make &
```

that maybe compiling a large program.

3.3.2 How do I execute a command at specified time (using "at" or "batch")?

The `at` command will execute the command(s) you specify at the date and time of your choice. For example, I could start playing music from my CDROM at 7 o'clock in the morning:

```
at 7:00
cdplay<Ctrl>d
```

In the example above, I entered the first line "at 7:00" on the command line and then pressed ENTER. To this, the `at` command displayed a prompt "at>". At this prompt, I entered my command "cdplay" and then pressed the control key and "d" simultaneously to finish the input. If instead of pressing <Ctrl>d , I pressed "ENTER", the next "at>" prompt would appear, at which I would be able to enter the next command to be executed right after "cdplay", also at 7:00. And so on, I could have had many commands scheduled for execution one by one starting at 7:00. After typing the last command, I would finish the input with <Ctrl>d. Think of the <Ctrl>d as sending "end−of−file" to the current input. Don't press <Ctrl>d twice because this will log you out−−that's what <Ctrl>d does when entered straight on the Linux command line.

You can list the job you scheduled for execution using:

```
at −l
```

which will give you the numbered list of the jobs waiting.

If you changed your mind, you can remove a job from this list. For example:

```
atrm 8
```

will remove the job with the number eight on the list.
I could also schedule a job for execution much later, for example:

```
at 23:55 12/31/00
startx
```

would start my X−windowing system right on time for the new millennium (5 minutes before midnight on 31 of December 2000). If you cannot execute the `at` command, check if the at daemon ("atd") is loaded (as root, use ntsysv). If you cannot execute the `at` command as a regular user although it works for root, check if the empty file `/etc/at.deny` exists and there is no file `/etc/at.allow`. This should be the default setup and it permits all the users to execute `at`. If you want only certain users to use `at`, create a file `/etc/at.allow` and list these users there.
For other options, check:

```
man at
```

If you wish to perform a processor−intensive job in the background when the system load is low, you may choose to use the `batch` command. For example, I could run setiathome (a program crunching data to help in search of extraterrestrial intelligence, SETI) using:

```
batch
at>setiathome<Ctrl>d
```

In this example, I entered the command `batch` and then, at the "at>" prompt, I entered the command which I wanted to be executed in the background. The job tries to start immediately, but goes ahead only when the system load is under 0.8 You can check the system load by inspecting the contents of the (virtual) file `/proc/loadavg` . For example:

```
cat /proc/loadavg
```

When a batch job finishes, the output is sent to me via e−mail.

3.3.3 How do I set up cron?

Cron (a Linux process that performs background work, often at night) is set up by default on your RedHat system. So you don't have to do anything about it unless you would like to add some tasks to be performed on your system on a regular basis or change the time at which cron performs its duties.

Please note that some of the cron work might be essential for your system functioning properly over a long period of time. Among other things cron may:
− rebuild the database of files which is used when you search for files with the `locate` command,
− clean the /tmp directory,
− rebuild the manual pages,
− "rotate" the log files, i.e. discard the oldest log files, rename the intermediate logs, and create new logs,
− perform some other checkups, e.g. adding fonts that you recently copied to your system.

Therefore, it may not be the best idea to always switch your Linux machine off for the night−−in such a case cron will never have a chance to do its job. If you do like switching off your computer for the night, you may want to adjust cron so it performs its duties at some other time.

To find out when cron wakes up to perform its duties, have a look at the file `/etc/crontab`, for example:

```
cat /etc/crontab
```

It may contain something like this:

```
# run−parts
01 * * * * root run−parts /etc/cron.hourly
02 4 * * * root run−parts /etc/cron.daily
```

```
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

You can see that there are four categories of cron jobs: performed hourly, daily, weekly and monthly. You can modify those or add your own category. Here is how it works.

The columns in the entries show: minute (0−59), hour (0−23), day of month (1−31), month of year (1−12), day of week (0−6−−Sunday to Saturday). The "*" means "any valid value".

Thus, in the example quoted, the hourly jobs are performed every time the computer clock shows "and one minute", which happens every hour, at one minute past the hour. The daily jobs are performed every time the clock shows 2 minutes past 4 o'clock, which happens once a day. The weekly jobs are performed at 22 minutes past four o'clock in the morning on Sundays. The monthly jobs are performed 42 minutes past four o'clock on the first day of every month. The directory with the script file that contain the command(s) to be executed is shown as the last entry on each line.

If you wanted your jobs to be performed at noon instead of 4 in the morning, just change the 4s to 12s. Cron wakes up every minute and examines if the `/etc/crontab` has changed so there is no need to re−start anything after you make your changes.
If you wanted to add a job to your cron, place a script which runs your job (or a link to your script) in the directory `/etc/cron.hourly` or `cron.daily` or `/etc/cron.weekly`, or `/etc/cron.monthly` .

Here is an example of an entry in `/etc/crontab` which causes a job to be performed three times a week (Mon, Wed, Fri):

```
02 4 * * 1,3,5 root run-parts/etc/cron.weekly
```

An example seen on usenet showing how to automatically email a log file (edited for space):

> Re: help in crontab
> From: Dean Thompson <Dean.Thompson@csse.monash.edu.au> Date: 2001−03−03 16:35
> Newsgroups: comp.os.linux.admin,comp.os.linux.networking,comp.os.linux.security
> > How can I set the job mail abc@abc.com < /var/log
> > every day in the /etc/crontab −e file ?
> You could try the following entry and see if you meet with any success:
> 0 0 * * * (/bin/mail abc@abc.com < /var/log/messages) > /dev/null 2>&1

3.4 Shell

3.4.1 What is a shell and do I want to use a different one?

A shell is the program that interprets what you type on the command line and decides what to do with it. A shell can also be invoked in a non−interactive way, for example to execute a pre−typed list of commands contained in a text file (a "shell script"). Think of a shell as the equivalent of the DOS "command.com" (command−line interpreter) and the shell script files as the equivalent of the DOS batch files (*.bat). In comparison with their DOS cousins, the Linux shell and scripting are on steroids.

There are several shells available on the Linux system (if you installed them): `bash` ("Bourne Again" shell), `sh` (Bourne shell, standard on many UNIX systems), `csh` (C shell, with a syntax akin to the "c" programming language, available on most UNIX systems), `pdksh` (public domain Korn shell), `tcsh` (tiny C shell, often used on small systems), `sash` (stand−alone shell, could be used when libraries are not available), `ash`, `zsh`, and perhaps a couple more.

The default shell on my system (and most probably on yours too) is `bash` , which is an excellent and standard shell, and I really cannot see a reason why a newbie like myself would want to change it. `bash` is fully backwards−compatible with the Bourne shell (the most popular shell on UNIX) and incorporates many enhancements and best features from other shells. From a newbie perspective, the different shells are included with Linux for historical reasons and backwards−compatibility of shell scripts that may require a particular shell to run. [Some shells may be useful if you write programs targeted for specialized "embedded" devices, that might run a "tiny" shell.]

You can determine the shell you are running using:

```
echo $SHELL
```

If you wanted to try another shell, type, for example:
`tcsh`
which will start the tiny c shell. When done, type
`exit`

which will return you to the previous shell (using `exit` on your first shell will log you out). You can find out how many shells you stacked on each other by displaying the "shell level" environmental variable:

```
echo $SHLVL
```

In the above command, the "$" means "expand the value of a shell environment variable", "SHLVL" is the variable name, and "echo" is a command that prints things.

The shell for each user is specified as the last field in the password file `/etc/passwd` . If you really wanted to change it, edit (as root) this file and replace the "/bin/bash" with the shell of your choice.

3.4.2 How do I customize my shell prompt?

On my machine, the prompt may look like this:

```
[stan@marie stan]$ _
```

Here "stan" is my login name, "marie" is the name of the computer, the second "stan" is the name of my current working directory, and "_" represents the cursor.

The prompt is set by the environmental variable called PS1. To display the current setting, I can use:

```
echo $PS1
```

The system−wide setting of the prompt (for all users on the system) is in the file `/etc/bashrc` which on my system contains such a line:

```
PS1="[\u@\h \W]\$ "
```

To customize the prompt, I can edit the file `/etc/bashrc` (as root) and insert almost any text inside the quotation marks. Here is the meaning of some special codes I may also choose to use:

```
\u   −    username of the current user (= $LOGNAME),
\h   −    the name of the computer running the shell (hostname),
\H   −    entire hostname,
\W   −    the base of the name of the current working directory,
\w   −    the full name of the current working directory,
\$   −    display "$" for normal users and "#" for the root,
\!   −    history number of the current command,
\#   −    number of the current command (as executed in the current shell),
\d   −    current date,
\t   −    current time (24−hr),
\T   −    current time (12−hr) − bash 2.0 only,
\@   −    current time (AM/PM format) − bash 2.0 only,
\s   −    name of the shell,
\a   −    sound alarm (beep),
\j   −    number of jobs the user has,
\n   −    new line,
\\   −    backslash,
\[   −    begin a sequence of non−printable characters,
\]   −    end a sequence of non−printable characters,
\nnn −    the ASCII character corresponding to the octal number nnn.
$(date) − output from the date command (or any other command for that matter),
```

Here is an example on how to add colour. See the next chapter for details about colour:

```
PS1="\[\033[1;32m\][\u@\h \W]\$\[\033[0m\] "
```

There is also the second−level prompt, set by a variable called PS2. The shell uses the second level prompt when it expects additional input, and on my system the secondary prompt is "> ". I don't worry too much about PS2, but if I did I could set it the same way as PS1. There are even PS3 and PS4, but these are rarely seen.

3.4.3 Colour on text terminal

Colour on the text terminal can be produced using the "ANSI escape sequences". For example:

```
echo −e "\033[44;37;5m ME \033[0m COOL"
```

The above sets the background to blue, foreground white, blinking video, and prints " ME ", then resets the terminal back to defaults and prints " COOL". The "−e" is an option specific to the `echo` command−−it enables the interpretations of the special characters. The "\033[" introduces the escape sequence. The "m" means "set attribute" and thus finishes the sequence. The actual codes in the example above are "44;37;5" and "0".

Change the "44;37;5" to produce different colour combinations−−the number/order of codes do not matter. The codes to choose from are listed below:

```
Code  Action/Color
−−−−−−−−−−−−−−−−−−−−−−−−−−
 0    reset all attributes to their defaults
 1    set bold
 2    set half−bright (simulated with color on a color display)
 4    set underscore (simulated with color on a color display)
 5    set blink
 7    set reverse video
22    set normal intensity
24    underline off
25    blink off
27    reverse video off
30    set black foreground
31    set red foreground
32    set green foreground
33    set brown foreground
34    set blue foreground
35    set magenta foreground
36    set cyan foreground
37    set white foreground
38    set underscore on, set default foreground color
39    set underscore off, set default foreground color
40    set black background
41    set red background
42    set green background
43    set brown background
44    set blue background
45    set magenta background
46    set cyan background
47    set white background
49    set default background color
```

Other interesting codes:

```
\033[2J      clear screen
\033[0q      clear all keyboard LEDs (won't work from Xterm)
\033[1q      set "Scroll Lock" LED
\033[2q      set "Num Lock" LED
\033[3q      set Caps Lock LED
\033[15;40H  move the cursor to line 15, column 40
\007         bell (beep)
```

LEDs (="Light Emitting Diods) are the lights on the keyboard which indicate if <CapsLock>, <NumLock> and <ScrollLock> are engaged.

See `man console_codes` for more.

3.4.4 How do I print symbols on the console, in a text mode application, and in X?

The procedure described here may give me fast access to the PC extended character set (codes 128−255) and is quite portable in the PC world: it works in MS Windows, DOS (if you have an ANSI driver installed), and inside any text mode Linux application (including right on the shell command line). I found it was worth my time to memorize the codes for the few characters I tend to use the most.

It works like this. Make sure that <NumLock> is on. Then press <Alt> and hold it. While <Alt> is pressed, key in on the numeric keypad these four digits: 0181. Now release <Alt> and the Greek letter mu "µ" appears. I find quite useful the following characters from the PC character set encoding: 176 ° (degree), 177 ± (plus minus), 178 ² (square), 179 ³ (power 3), 181 µ (Greek mu), 0183 · (multiplication sign), 232 è (French accent agrave), 233 é (French accent aigu) 228 ä (German a−umlaut), 243 ó (Polish u−zamkniete), 248 ø (Scandinavian o−bar) 252 ü (German u−umlaut). Some other characters are also possible, here is the full listing:

```
128 ? 147 ? 166 ¦ 185 ¹ 204 Ì 223 ß 242 ò
129 ? 148 ? 167 § 186 º 205 Í 224 à 243 ó
130 , 149 * 168 ¨ 187 » 206 Î 225 á 244 ô
131 f 150 − 169 © 188 ¼ 207 Ï 226 â 245 õ
132 ? 151 − 170 ª 189 ½ 208 Ð 227 ã 246 ö
133 ? 152 ~ 171 « 190 ¾ 209 Ñ 228 ä 247 ÷
134 ? 153 ? 172 ¬ 191 ¿ 210 Ò 229 å 248 ø
135 ? 154 s 173 - 192 À 211 Ó 230 æ 249 ù
136 ^ 155 > 174 ® 193 Á 212 Ô 231 ç 250 ú
137 ? 156 ? 175 ¯ 194 Â 213 Õ 232 è 251 û
138 S 157 ? 176 ° 195 Ã 214 Ö 233 é 252 ü
139 < 158 ? 177 ± 196 Ä 215 × 234 ê 253 ý
140 ? 159 Y 178 ² 197 Å 216 Ø 235 ë 254 þ
141 ? 160   179 ³ 198 Æ 217 Ù 236 ì 255 ÿ
142 ? 161 ¡ 180 ´ 199 Ç 218 Ú 237 í
143 ? 162 ¢ 181 µ 200 È 219 Û 238 î
144 ? 163 £ 182 ¶ 201 É 220 Ü 239 ï
145 ? 164 ¤ 183 · 202 Ê 221 Ý 240 ð
146 ? 165 ¥ 184 ¸ 203 Ë 222 Þ 241 ñ
```

Now, if I really want to, I can have a file with a name `µm·°C±b³.` MS Windows, DOS ANSI, and Unicode differ slightly in some of the above characters, but the useful "core" remains the same. See http://www.hclrss.demon.co.uk/demos/ansi.html if you want to know the details of the differences. Linux uses the Unicode standard.

Under X, the above key combinations will not work. But I may use:

```
kcharselect&
```

or

```
gcharmap&
```

to select a Unicode character and copy it into my application. Not all unicode characters are available yet, but many are. From Unicode pages other than page 0, the characters may display or not, depending on your application and the availability of the glyph in your font. For example, I can surely use the following characters in most KDE applications (if they display on your browser, depends on your browser AND the availability of a suitable Unicode font):

Greek (Unicode page 3, char 913 to 969): ???????????? ???????????????? ???????????????? ??????????
Russian: (Unicode page 4, chars 1040 to 1103): ????????? ????????????? ????????????? ????????????? ?????????????? ??
and many others. You can find common Unicode codes (numerical) and their html symbolic ("character entity") references at http://www.hclrss.demon.co.uk/demos/ent4_frame.html.

3.4.5 How do I write a simple shell script?

Create a text (ASCII) file which will contain the shell script. For example, I would use the pico editor to write a script that runs the program `tar` with all the parameters usually necessary to uncompress tarballs downloaded from the Internet (I never seem to remember the `tar` options). I decided to call my script "untar":

```
pico untar
```

Since the file "untar" did not exist in my current directory, it was created by the `pico` text editor. Now, I type in the content of my script:

```
#!/bin/bash
echo this is the script file $0
echo untarring the file $1
# this calls tar with options −xvzf (extract,
#    verbose, filter through gzip, input filename)
tar −xvzf $1
```

I save the file with <Ctrl>o and exit with <Ctrl>x

The first line of the script, starting with "#!" (called pound−bang), is special−−it tells the shell what program should be used to interpret my script. In this example, the script is to be interpreted by the bash shell `/bin/bash`. The first line must start with #! or the script will never run (the file will be interpreted as just a text file). Other lines starting with # are comments for the author (readers, users) of the shell and are totally ignored by the computer.

The $0, $1, $2 ... in my script are the parameters passed to my script. For example, if I ran a script called "myscript" with seven parameters like this:

```
myscript a b c d e f g
```

then $0 would be seen inside "myscript" as having the value "myscript", $1 would have the value "a", $2 would be "b", $3 would be "c", etc.
On the second and third line of my example script, the command `echo` prints on the screen everything that follows on the same line, expanding $0 and $1 to the values of the parameters passed to the script. The fourth and fifth line contains a comment I wrote to myself to remind myself what I was trying to achieve, just in case I ever had to modify my script. The last line performs the actual work.
Once the script is written, I make the file executable to the file owner ("u"=user):

```
chmod u+x untar
```

and my script is ready to run like this:

```
./untar my_tar.tar.gz
```

Linux scripting is definitely rich, flexible, powerful and can be complex. However, it does not require special knowledge to write simple scripts for automation of common tasks. You just put together a group of often used commands, one by one, into a file. I use scripting because I am too lazy to type the same groups of commands over and over again.

A really simple sequence of commands can also be typed into a text file and passed to shell for straight execution using:

```
source my_file
```

[No need for the initial "pound bang" or executable permission.]

3.4.6 Meaning of quotes

Normally, these characters are special to the shell:

```
\ ' " ` < > [ ] ? | ; #  $  ^ & * ( ) = <Space> <Tab> <Newline>
```

There are four different types of quotes: backslash (\), single quotes (apostrophes, '), double quotes (quotation marks, "), and backquotes (`).

The backslash \ means: disable the special meaning of the subsequent character.

Quoting with " (two apostrophes) means: quote exactly, disabling any special characters inside the quotes.

Quoting with "" means: disable the special characters inside the quotes except for `$ ` \`

The pair `` (two backquotes) means: do a command substitution inside the backquotes first. So what is inside the backquotes is executed by the shell first, and then the output is passed to the command outside the quotes. The same can also be acomplished with

`$(command)` which nests better than backquotes.

Examples. I can create a funny directory called "*" by either \ quoting or " quoting:

```
mkdir \*
mkdir '*'
```

This hides the special meaning of the "*" from the shell (without the quote it would mean "all files in the current directory").

3.4.7 Input/output redirection

There are three important input−output streams: standard input ("stdin"), standard output ("stdout"), and standard error output ("stderr"). They all default to the console ("console" means the keyboard for the input and the screen for the output), but they can be redirected.

To redirect the standard output I use ">". For example:

```
dir my_dir > filelisting.txt
```

will redirect the standard output of the `dir` command into the textfile `filelisting.txt` and nothing should appear on my screen. The file can be subsequently edited (e.g. with `pico filelisting.txt`) or embedded into a document.

To redirect the standard error, I need to use the construct "2>". For example:

```
dir my_dir 2> errorlisting.txt
```

The above will send the normal output onto the screen and nothing to the file unless `dir` produces an error. On error, nothing may go to my screen, and the file `errorlisting.txt` will contain the error message, which might be something like:

```
dir: my_dir: Permission denied
```

Finally, I can redirect both standard output and standard error to a file using:

```
dir my_dir > file_and_error_listing.txt 2>&1
```

which first redirects the standard output to a textfile, and then redirects the standard error to the same location as the standard output. A bit twisted, how it works, but it works.

In the examples above, if the file (to which to redirect) already existed, it will be overwritten. To append to an existing file, I use ">>" as in these examples:

```
dir my_dir >> filelisting.txt
dir my_dir 2>> errorlisting.txt
dir my_file >>file_and_error_listing.txt 2>&1
```

If you are puzzled by the "2>" symbol, here, briefly, is how to rationalize it. The standard streams have standard descriptors. "0" is standard input, "1" standard output and "2" is standard error.

```
dir my_dir > file.txt
```

is short for

```
dir my_dir 1> file.txt
```

and therefore the example below redirects the standard error:

```
dir my_dir 2> file.txt
```

One can also use the symbol "|" to send ("pipe") the output from one command as input for another command. In this popular example, the output from `dir` is piped to `more` (`more` pauses the display after each screenful):

```
dir | more
```

One can also split the output so it goes both to a file and the screen using "tee":

```
dir | tee filelisting.txt
```

It is called "tee" by the analogy to the "T"−letter−shape fitting that pipefitters use, and which divides flow.

This section so far dealt with redirecting standard output. Redirecting standard input is not nearly as useful as redirecting the output, but it can be done using a construct like this:

```
cat < my_file
```

There is also something called in−line redirection of the standard output, realized with "<<". Forget about it, seems of no use to me. Yet, here is an example if you really ever needed it (here, the ">" stands for the secondary prompt):

```
cat << my_marker
> my_line_from_the_keyboard
> another line_from_the_keyboard
> my_marker  [the marker of my choice ends the in−line redirection].
```

Apart from redirection to regular files and "filters" (as shown in the examples above), one can redirect to/from devices and other special files. Some examples follow.

An example of redirection to a device file. The following command displays the listing of files on the fourth text terminal:

```
dir > /dev/tty4
```

An example of redirection to a special "FIFO" file. This command sends the message "you are lucky" to the lucky ICQ user UIN 77777777 (assuming you are connected to the icq server with your licq program):

```
echo message 77777777 "you are lucky" > ~/.licq/licq_fifo
```

> The above works because the file "licq_fifo" in your licq directory is a special "fifo" (first−in−first−out) queue file. How could the above ever be more useful than sending a message using the pretty licq GUI front−end? For example, you could write a short script to impress fellow icq users with multiple (identitcal) messages:
>
> ```
> #!/bin/bash
> echo Messaging UIN:  $1  Message:  $2   Times: $3
> # The next command puts puts your licq in the status "on−line, invisible".
> echo 'status *online' > ~/.licq/licq_fifo
> c=0
> while  [ $c −le $3]
> do
> echo message $1 $2 > ~/.licq/licq_fifo
> c=`expr $c + 1`
> echo $c " "
> done
> echo 'status offline' > ~/.licq/licq_fifo
> echo "all done"
> ```
>
> The above example may give you an idea how one can use licq for automation, owing to the smart licq communication model (the fifo file) and simple file redirection.

3.4.8 Shell special characters (metacharacters)

Normally, these characters have special meaning to the shell:

```
\ ' " ` < > | ; <Space> <Tab> <Newline> ( ) [ ] ? # $ ^ & * =
```

Here is the meaning of some of them:

`\` `' "` and `'` are used for quoting and were described before.

`<` and `>` are used for input/output redirection and were described before.

`|` pipes the output of the command to the left of the pipe symbol "|" to the input of the command on the right of the pipe symbol.

`;` separates multiple commands written on a single line.

`<Space>` and `<Tab>` separate the command words.

`<Newline>` completes a command or set of commands.

`( )` enclose command(s) to be launched in a separate shell (subshell). E.g. `( dir )`.

`{ }` enclose a group of commands to be launched by the current shell. E.g. `{ dir }`. It needs the spaces.

`&` causes the preceding command to execute in the background (i.e., asynchronously, as its own separate process) so that the next command does not wait for its completion.

`*` when a filename is expected, it matches any filename except those starting with a dot (or any part of a filename, except the initial dot).

`?` when a filename is expected, it matches any single character.

`[ ]` when a filename is expected, it maches any single character enclosed inside the pair of [ ].

`&&` is an "AND" connecting two commands.

> *`command1 && command2`* will execute *command2* only if *command1* exits with the exit status 0 (no error). For example: `cat file1 && cat file2` will display file2 only if displaying file1 succeeded.

`||` is an "OR" connecting two commands.

> *`command1 || command2`* will execute *command2* only if *command1* exits with the exit status of non−zero (with an error). For example: `cat file1 || cat file2` will display file2 only if displaying file1 didn't succeed.

`=` assigns a value to a variable.

> Example. This command:
> `me=blahblah`
> assigns the value "blahblah" to the variable called "me". I can print the name of the variable using:
> `echo $me`

`$` preceeds the name of a variable to be expanded.

> The variables are either assigned using "=" or are one of the pre−defined variables (which cannot be assigned to):
> $0 name of the shell or the shell script being executed.
> $# number of the positional parameters to the command
> $1 the value of the first positional parameter passed to the command. $2 is the second positional parameter passed to the command. etc. up to $9.
> $* expands to all positional parameters passed to the command
> $@ expands to all positional parameters passed to the command, but individually quoted when "$@" is used.
> See `man bash` if you really need more.

3.5 Package installation and rpm package manager

3.5.1 How do I install a program I downloaded from the Internet?

The answer depends on what kind of package you downloaded. You can avoid many installation headaches if you download programs in the form of Red Hat binary packages *.rpm (that's the format I select if given a choice).

INSTALLATION OF REDHAT BINARY PACKAGES

o If the program I want to install is a RedHat binary package (*.rpm), I can use either a command line, or a GUI utility. I like to use the command−line utility because it is fast and trouble−free. The RedHat package manager installation utility is called `rpm` . First I read the info on the package content (optional):

```
rpm −qpi my_new_file.rpm
```

This queries (mode "q", must be the first letter after the dash) the yet uninstalled package (option "p") so that it displays the info (option "i") which the package contains. If I want to install the program, I run (as root):

```
rpm −ihv my_new_file.rpm
```

The above command does the installation job. It runs `rpm` telling it to install the package (mode "i", must be the first letter after the dash) while printing to the screen more information than usual (option "h"=display "hashes" to show the unpacking progress, option "v" = be verbose). The contents of the package are distributed to the directories where they belong (rpm knows where they belong). After this installation, the program is ready to run, I just have to know the executable name and its location. If I have trouble finding the executable, this lists all the files that the package contains together with their destination directories:

```
rpm −qpl my_new_file.rpm
```

This queries (option "q") the yet uninstalled package (option"p") so that it displays the listing (option "l") of all the files the package contains.

The GUI front−ends to `rpm` are: `gnopro` (the old version, that comes with RH6.0 is confusing, but newer versions are much improved), `kpackage` (available only with the more recent distributions), and the old `glint` (very slow, comes with RH5.2).

Troubleshooting. rpm is supposed to be an intelligent software package manager. If the installation fails I read the error message and may be able to figure what to do:
(1) Installation failed because I have an earlier version of the same package and the versions confilict. Solution: don't install, but "upgrade" the package.

```
rpm −Uvh my_new_file.rpm
```

(2) Installation failed because another package is needed first. I have to find the missing package and install it first, and then retry the installation. In extreme cases, I may choose to ignore the missing dependencies (I really should know what I am doing here else the software may malfunction):

```
rpm −ivh −−nodeps my_new_file.rpm
```

or perhaps even:

```
rmp −ivh −−nodeps −−force my_new_file.rpm
```

INSTALLATION FROM A SOURCE−CODE TARBALL

o If what I downloaded from the net is a Linux source code in the form of a compressed tarball (*.tar.gz or *.tgz), the installation procedure is longer and more troublesome than with the binary−only rpm. I typically install the program as root.

First, I change my current working directory to `/usr/local` :

```
cd /usr/local
```

Second, I decompress the tarball that I downloaded from the net:

```
tar −xvzf /home/the_dir_where_the_tarball_is/my_tarball.tar.gz
```

This extracts (option "x") the contents of the *.tar.gz (or *.tgz) tarball, unzips it (option "z"), while talking to me more than usual (option "v" = verbose). Please note that the option "f" means "file", so the filename must immediately follow the letter "f". The contents of the tarball are extracted into a subdirectory which `tar` creates under my current working directory, which in the typical case is `/usr/local/` . The tarball knows what the new subdirectory should be called.

If the tarball is not compressed (e.g., *.tar), I may use:

```
tar −xvf /home/the_dir_where_the_tarball_is/my_tarball.tar
```

Third, I have to figure how the new directory is called, then I `cd` into it:

```
dir
cd the_new_program_subdir
```

Since some of the directories have long names, I use the great autocompletion option to save on typing−−I just type the first few letters and then press <TAB> .

Fourth, most programs are compiled by executing these three commands:

```
./configure
make
make install
```

The above commands can take some time to complete (1 min? 0.5 h?). If any of them fail, it might be an idea to read the README or INSTALL or whatever info is provided with the new program. Some programs may require customization of the environment (e.g. definition of their path) or installation of an additional library, or yet something else. It can sometimes be a pain. Very simple programs might not need the "./configure" or/and "make install" step, in which case "make" alone will do.

Fifth, if everything goes well, I find the new executable which I just compiled. The names of executables display in green when running this command:

```
ls −−color
```

Now, I can run the executable, for example:

```
./the_executable
```

Some programs automatically install the executable to /usr/local/bin, so I may want to try:

```
/usr/local/bin/the_executable
```

Sixth, if I plan to run the program more often, I create a symbolic link to the executable from the directory `/usr/local/bin` :

```
cd /usr/local/bin
ln −s /usr/local/the_new_program_subdir/the_executable .
```

This way, the executable (actually, a symbolic link to it) is on my PATH and it can be run by simply typing its name (no need to type the full path to the executable any more). Some programs will install the executable (or a link to it) in a "bin" directory in which case you skip the last step.

INSTALLATION FROM SOURCE CODE RPM PACKAGE

o There are also programs distributed as "source code rpm" packages. They require installation of the *.rpm package with the "rpm" utility as described in the first part of this chapter. But since the "rpm" installs the source code (typically in the C language source code), I then have to compile the source code by executing the same: "./configure ; make ; make install" sequence as for the sourcecode distributed as tarballs (see the previous answer).

Go to Part 4.1: Startup Issues (LILO and GRUB)
Back to the Main Index

Part 4: Linux Newbie Administrator FAQ

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

4.1: Startup Issues (LILO and GRUB)

Contents of this section:

4.1.0 LILO and GRUB

Both the newer GRUB and the older LILO are boot loaders. They make it possible for you to select the operating system to boot at the boot time. Most (all?) of the booting problems described in this section can likely be overcome by installing the most recent Linux kernel and the latest GRUB boot loader. GRUB is better than LILO because LILO relies on the absolute hard drive addresses to find the boot image, while GRUB understands the filesystems and looks for a file containing the boot image. We recommend using GRUB when given a choice during the installation.

The main GRUB configuration file is `/boot/grub/menu.lst` (or `grub.conf`, on my system one is a symbolic link to the other). Here are some comments on the items found in this file:

Lines starting with the # mark are comments.

```
timeout 5
```

the above setting starts booting the default operating system after 5 seconds unless a key is pressed

```
default 0
```

the above setting makes the default operating system to be the first found in the menu list. I could use "default 3" to have the 4th menu item the default.

#

```
title linux
kernel (hd0,2)/boot/vmlinuz root=/dev/hda3  mem=64M
hdc=ide−scsi
```

The above two lines define a boottime menu item, and set the boot action for this item.
The first line names the menu item "linux".
The second line specifies that the kernel is located on the first physical hard drive (hd0), the third partition (2), the boot image is the file /boot/vmlinux
Also on the second line, the following options are passed to the kernel:
root=/dev/hda3 (i.e., make the root partition the 3d partition on the first hard drive (hda)),
mem=64M (i.e., force using 64 megabytes of physical memory.).
hdc=ide−scsi (use SCSI emulation on my CD ROM, because it is a CD writer).
#

```
initrd /boot/initrd−2.4.17−custom.img
```

define the file which contains the modules needed at the boot time, as the modules load to the "initial ram disk" (initrd).
I had to re−create mine (because I recomplied the kernel) using mkinitrd /boot/initrd−2.4.7−10custom.img 2.4.7−custom

Good documentation for GRUB is available using `info grub`

4.1.1 Linux cannot detect all my memory

If you have more than 64 megabytes of physical memory, Linux kernel ver. 2.0.36 or lower will use, by default, only the first 64 MB. To see how much memory Linux uses on your system, type:
`cat /proc/meminfo`
or
`free`

You can check your version of Linux kernel with:
`uname −a`

The last popular kernel with the "memory problem", 2.0.36, comes with RedHat 5.2. My RedHat 6.0 came with kernel 2.2.5−15 so it does not have the "memory problem" any more.

To get more than 64 MB memory recognized on RH5.2, you have to edit (as root) the file `/etc/lilo.conf`, and add a line like this just before your first "image=" statement:
`append="mem=80M"`

If you have an amount of memory different than 80 MB, adjust the above line. For any changes in `/etc/lilo.conf` to take effect, you *must* re−run the program
`lilo`

(watch if it runs without errors) and reboot. After the reboot, you can check if your adjustment worked using either of these two commands:
`cat /proc/meminfo`
`free`

For testing purposes, or if you are having problems, the option of specifying the amount of memory at the LILO prompt is useful:

[type at LILO prompt during bootup] `linux "mem=16M"`

Occasionally, I hear the advice to skip the upper few megabytes if you have problems enabling all your memory, or the machine locks up. E.g., enable only 78 out of your 80 MB. This is apparently the case for some SCSI controllers that use the very upper chunk of the main memory. Take it for what it's worth.

Occasionally on some systems, Linux recognizes only 16 MB of memory. This is usually linked to the setting "memory hole at 15−16 MB" enabled in the BIOS setup (the solution is to disable this BIOS setting). It is probably a good idea to disable all "advanced" features in your BIOS setup anyway (for example, the BIOS virus detection seems to be a common source of problems).

Mixture of memory chips with different timings can also lead to memory recognition problems or to system crashes (the solution is to replace the memory chips so that the timing of all memory chips is the same).

4.1.2 LILO displays only LI (or LIL) and hangs

I quote from my good handbook "Red Hat Linux Unleashed" by Kamran Hussain, Timothy Parker, et al., published by SAMS Publishing:

"When LILO loads itself, it displays the word LILO. Each letter is printed before or after performing some specific action. If LILO fails at some point, the letters printed so far can be used to identify the problem. [...]

LI [...] This is caused either by geometry mismatch or by moving `/etc/lilo/boot.b` without running the map installer.

LIL [...] This is typically caused by media failure or geometry mismatch."

The geometry means the number of sectors/heads/cylinders used in the hard drive configuration of your BIOS. Hope this helps!

It is a very good idea to have a handbook for Linux or at least a general UNIX handbook. Handbooks for Windows

are useless, handbooks for Linux are great! "Red Hat Linux Unleashed" is a very good handbook but I am sure there are many other equally good ones.

With a LILO error like above, you can boot your machine using a Linux or DOS boot floppy. There seems to be several general possibilities to correct such a LILO error, depending on what is wrong:

1. If LILO simply got corrupted (does not seem very common), you can remove and re−install it. You can remove LILO by running under Linux:
`lilo −u /dev/hda`

or, under DOS:
`FDISK/MBR`

which rewrites the hard drive master boot record (MBR), in which LILO resides, and replaces it with "clean" DOS stuff. You will lose access to Linux if you rebooted your computer after removing LILO (if this happened, you can boot Linux from the floppy and re−install LILO on top of the DOS MBR).

To re−install LILO, simply re−run the command `lilo` (as root).

2. Specify the option
`linear`

at the top of your `/etc/lilo.conf` file. This is particularly useful for large drives (>8 GB). See `man lilo.conf` for details. The option "linear" is safe−−it should not affect a properly working system, so you can specify this option is any case.

You may also want to play with the method by which BIOS accesses your harddrive. Fore example, turn the LBA ("linear or large block access") mode on/off in your BIOS to see if this helps.

Instead of the option "linear" you may try specifying the option (helpful to overcome the 1024 cylinder limit with larger harddrives and newer BIOSes):
`lba32`

This is a new option so it won't work with the stock LILO supplied with RH6.1 or lower. Use the latest Mandrake or RedHat if having the "LI" kind of problems−−hard drives are bigger and bigger, and BIOS makers put new tricks to support them.

3. Look into your BIOS setup and figure out how the specified hard drive geometry does not match your hard drive. From under Linux, you can display the hard drive geometry using (for example, for the first IDE hard drive):
`hdparm −g  /dev/hda`

You can typically easily find the recommended manufacturer geometry on the web using Google to search for your harddrive model number.

4. Put LILO on another partition (different hard drive) and, using fdisk, make this partition bootable (if your system supports booting from another drive). Or swap your harddrives so that the one that is better supported by your old BIOS comes as the first hard drive on your first IDE interface (DOS drive "C").

5. Maybe you prefer to use "loadlin" instead of "lilo". From under DOS, locate your CDROM and see the program `/dosutils/loadlin.exe`. It boots Linux from DOS.

6. Get rid of LILO and use GRUB. Mandrake 7.2 contains GRUB as the default boot loader.

4.1.3 How can I change the operating system that LILO boots on default?

This can be set in the lilo configuration file `/etc/lilo.conf` . Mine (lilo version 0.21.5.1−4MDK) looks like this:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
vga=normal
```

```
default=linux
keytable=/boot/us.klt
lba32
prompt
timeout=50
message=/boot/message
menu-scheme=wb:bw:wb:bw
image=/boot/vmlinuz
        label=linux
        root=/dev/hda3
        append=" mem=96M"
        read-only
image=/boot/vmlinuz
        label=failsafe
        root=/dev/hda3
        append=" mem=96M failsafe"
        read-only
other=/dev/hda1
        label=windows
        table=/dev/hda
other=/dev/fd0
        label=floppy
        unsafe
```

The four "label=" entries define the names of the boot choices. The default operating system to boot is specified by the option "default=" at the top of the file. In the absence of the "default", the first label to appear in `/etc/lilo.conf` is booted by default.

Don't forget to re-run the command
`lilo`

after any changes to the `/etc/lilo.conf` file.

There are also GUI utilities to configure lilo. For example, try, as root, in an X terminal:
`klilo &`

4.1.4 The LILO prompt stays too short (or too long) on the screen during the bootup

Add or adjust the line
`delay=100`

right before the first "image=" or "append=" statement in your `/etc/lilo.conf` file. (Newer versions of lilo may use a "timeout" option instead.) The number is the time of delay in tenths of a second (0.1 s), so in the example above the delay will be 10 seconds. Don't forget to re-run `lilo` after making any changes to the `/etc/lilo.conf` file, or your changes will not be enabled.

4.1.5 Uninstalling Linux

If you really wanted to "uninstall" Linux, you could run the following two commands (from under DOS or MS Windows):

```
LOCK C:
FDISK/MBR
```

which will get rid of LILO--it overwrites the master boot record (MBR) of your first hard drive, where LILO resides. The "lock" command allows "raw" writing to disk, which is normally disallowed on more recent DOS versions as an antiviral measure. The problem with FDISK/MBR is that it does not report back any success or failure, so it is better to proceed it with the "lock" command. After this you can remove the Linux partitions using the DOS "FDISK" utility to re-claim the hard drive space.

Apparently, MS FDISK does not always cope with removing the Linux partitions. In this case, I may use linux fdisk. The simplest may be to boot from the Linux installation floppy/CD, and to remove the partition using the Linux partitioning tool when it pops up during the "installation" procedure. After that I abort the "installation" and

Linux is gone.

If you still have problems, here are the ultimate solutions for zeroing the MBR (after: http://www.linuxgazette.com/issue63/okopnik.html, edited for space):

* *

Note: The following advice will completely wipe your Master Boot Record (MBR), which contains all your partition information. DO NOT DO THIS unless you know that this is exactly the result you want − it will leave your hard drive in an unbootable state, in effect bringing it back to "factory−fresh", i.e., empty of data and requiring partitioning and formatting.
* *

Linux−based solution. If you can boot Linux − say via boot floppy − you can simply invoke "dd":

```
dd if=/dev/zero of=/dev/hda bs=512 count=1
```

This fills up the MBR with zeros. Obviously, you have to be root to do this.

DOS−based solution . Boot with a DOS floppy that has "debug" on it; run

```
debug
```

At the '−' prompt, "block−fill" a 512−byte chunk of memory with zeroes:

f 9000:0 200 0

Start assembly mode with the 'a' command, and enter the following code:

mov dx,9000
mov es,dx
xor bx,bx
mov cx,0001
mov dx,0080
mov ax,0301
int 13
int 20

Press <Enter> to exit assembly mode, take a deep breath − and press "g" to execute, then "q" to quit "debug". Your HD is now in a virgin state, and ready for partitioning and installation.

Go to Part: 4.2 − Accessing my drives
Back to Main Page

4.2: Accessing my drives

Contents of this section:

4.2.1 Where are my drives?

Linux shows all the directories in one directory tree, irrespectively of what drives/hardware they are found on. Generally, this is a much better solution than the traditional DOS/Windows model−−it completely abstracts the file system from the underlying hardware. You will appreciate this if you ever have to re−arrange or expand your hardware or add network resources. But for the users who are accustomed to the DOS way of dealing with drives, it adds some extra complexity.

To be brief, don't search for drive letters. There are none under Linux; the content of your disks appears as subdirectories on your single Linux filesystem (directory tree). On default, the content of removable media does not appear automatically in these subdirectories−−you have to "mount" your drives. See the next answers for details. You should also unmount a drive before ejecting the media.

You can access (read and write) a variety of drives and file systems from under Linux. This includes native Linux partitions, DOS and MS Windows partitions (on hard drives or floppies), ZIP and Jazz drives, and CDROM disks. Many less common file system types are also supported. This means that you can download your Linux software using Netscape for Windows, save the downloaded file on your MS Windows hard drive partition, and then boot Linux and copy the downloaded software from the Windows partition on your harddrive to the Linux partition, and finally install the software under Linux.

4.2.2 How can I access my CDROM?

Mount it. The mounting adds all the directories and files from your CD to your Linux directory tree so you can easily access them without the drive letter.

As root, you can mount the CDROM with a command like this:

```
mount −t auto /dev/cdrom /mnt/cdrom
```

If this works, the contents of your CD appears in the directory `/mnt/cdrom`

Chances are this command will not work for you right away−−you may have to customize it. Here is how it works.

The command tells the operating system to mount a filesystem autodetecting the filesystem type ("−t auto"). The device is `/dev/cdrom`. The mountpoint (the directory where to which "mounting" takes place) is `/mnt/cdrom`. This directory must exist and be empty. If it does not exist, create it with:

```
mkdir /mnt/cdrom
```

If the mounting command fails, maybe the device `/dev/cdrom` does not exist on your system? Try

```
ls −l /dev/cdrom
```

/dev/cdrom is just a convenient symbolic link to a real "device" that is mapped onto your hardware. On an IDE system, chances are your real CD rom is on /dev/hdb. Therefore, try `/dev/hdb` instead of `/dev/cdrom` in the `mount` command above:

```
mount −t auto /dev/hdb /mnt/cdrom
```

If this fails, you can try `/dev/hdc` or `/dev/hdd`, if your CD is an IDE CDROM on the second IDE interface. If none of them is your CDROM, maybe you don't have IDE but a SCSI CDROM? Then try `/dev/sda`, `/dev/sr0`, etc.

A short listing of possible drives could include:
`hda` −− the master drive on the first IDE interface (that's always the first hard drive)
`hdb` −− the slave drive on the first IDE interface (you must have at least two hard drives for that)
`hdc` −− the master drive on the second IDE interface (if you have two IDE interfaces on your computer, most newer computers do)
`hdd` −− the slave drive on the second IDE interface (if you have one)
`sda` −− the first SCSI drive
`sdb` −− the second scsi drive ("sdc" is the third scsi drive, etc. There can be many scsi drive on a system).
`sr0` −− the first scsi CD drive (sometimes called scd0)
`sr1` −−the second scsi CD drive (sometimes called scd1), (sr2 is the third scsi CD drive, etc. There can be many

scsi CD drives on the system).

It is a good idea to have a device `/dev/cdrom` anyway because some programs assume that it exists. If it does not exist on your system, you may create it as a symbolic link using, for example:

```
ln −s /dev/hdb /dev/cdrom
```

if your cdrom is the `/dev/hdb` drive.

If you cannot mount because "the device is already mounted or directory busy", perhaps the mountpoint `/mnt/cdrom` is your current directory. You have to change the directory to somewhere else in order to be able to mount to it; for example change the current directory to the root directory by issuing this command:

```
cd /
```

To unmount a mounted CD, exit the directory /mnt/cdrom and type as root:

```
umount /mnt/cdrom
```

Your CDROM may refuse to eject the media if it is not unmounted. Also, you may have problems mounting the next CD if the previous one was not unmounted. If you cannot unmount because "the device is busy", perhaps /mnt/cdrom (or any subdirectory underneath) is your current directory? You need to change your current directory to somewhere else out of the mountpoint in order to unmount the device.

4.2.3 How to mount a floppy, zip drive, DOS/Windows partition, or a network drive?

Very much the same as CDROM−−see the previous answer if you did not read it.

Floppy. I can mount my floppy (as root) with:

```
mount −t auto /dev/fd0 /mnt/floppy
```

Again, make sure that the directory /mnt/floppy exists and is empty. Also, `/mnt/floppy/` cannot be your current directory.

After a successful mount, the files from the floppy appear in the directory `/mnt/floppy/` . All the users will be able to read the files, but only root will be able to modify/delete the files. Please read further if you wanted the users to be able to write to the floppy.

To unmount a floppy (you *must* do this before ejecting the disk!) use:

```
umount /mnt/floppy
```

If you cannot unmount because "the device is busy", perhaps the `/mnt/floppy/` directory is your current directory. Exit it by typing (for example):

```
cd
```

which will change your current directory to your home directory.

Zipdrive. I mount the parallel port external zipdrive (scsi emulation) with:

```
mount −t vfat /dev/sda4 /mnt/zipdrive
```

The "−t vfat" is used here because zip disks come preformatted in the vfat filesystem, which is the filesystem of MS Windows with the long filename support. You won't be able to eject the disk without unmounting it. Again, the directory must exist, be empty, and must not be your current working directory (see the previous answer).

I can mount an internal IDE zipdrive using:

```
mount −t vfat /dev/hdd4 /mnt/zipdrive
```

On my system, this is the second drive on the second IDE interface, hence "hdd"−−replace it with "hdb" or "hdc"

if necessary on your system.

A tip from Alvaro Reguly <alvaro@reguly.net>. "I have a ATAPI Zip Drive (recognized as ATAPI Floppy) so to make it work with Debian and kernel 2.4.3 I had to switch my BIOS setting from "Autodetect" to "None" (just the Zip channel of course), and mount it using

```
mount −t vfat /dev/hdb
```

(without the trailing 4!) "

All zipdrives (internal SCSI and IDE, external SCSI and parallel port) but the USB are supported under Linux (April 1999). See forward in this chapter for info on how to manually load a module (driver) for zipdrives if one does not load automatically on your system.

DOS/Windows partition. I use a dual boot system with both Linux and MS Windows on the same computer. I can access files on the DOS/Windows partition after mounting it with the following command:

```
mount −t vfat /dev/hda1 /mnt/dosdrive
```

Again, you may have to customize this command depending on what partition your DOS filesystem is. The "hda1" means the first IDE hard drive (hd a), first partition (1); "hda2" is the first IDE hard drive, second partition; "hda3"−−the first IDE hard drive, third partition; "hdb1"−−second IDE hard drive, first partition (or just "hdb" if it is the CDROM installed as a slave on your first IDE interface). "hdc" is the third IDE drive, hdd is the fourth IDE drive. SCSI drives have analogous names but start with letters "sd", followed by the letter indicating the SCSI interface, followed by the number indicating the SCSI device id . For example sda4 means "first SCSI interface, id number 4".

To mount so that all the users can read and write, you may want to try:

```
mount −t vfat −o user,rw,exec,umask=000 /dev/hda1 /mnt/dosdrive
```

This uses options (−o user,rw,exec,umask=000) to give absolutely everybody all the permission to all files on your DOS /dev/hda1 partition (you should ask yourself if this is really safe on your system). If users still can't write to the DOS partitions, perhaps the permissions on your mountpoint need to be set. This command (executed by root) will set up the permissions on the mountpoint `/mnt/dosdrive` so that all users will be given rights to read, write and execute:

```
chmod a=rwx /mnt/dosdrive
```

Network File System (NFS). This is great for direct access to files that reside on another Linux computer. For mounting of a remote filesystem as NFS, first check if the NFS service is enabled (use the program `setup`). NFS also requires permission from the other computer. To configure the permissions on the server machine, run as root:

```
netconf
```

and adjust the setting under "Exported File Systems" menu.

If you prefer to do it manually, the permissions are set in the file `/etc/exports` . My `/etc/exports` looks like this:

```
/usr hacker(ro) mars(ro)
/home hacker(rw) mars(rw)
/mnt hacker(rw) mars(rw)
```

This gives the machines called hacker and mars the permission to mount the directories `/usr/` (read−only access), `/home` and `/mnt` (read−write).

If you set up your NFS properly, you should now be able to mount a network directory using a command like this:

```
mount −t nfs mars:/home /mnt/mars_home
```

This mounts the contents of the directory `/home/` on a machine called "mars" into the directory `/mnt/mars_home/` (which must exist and be empty).

Many operating systems know NFS, but MS Windows doesn't. Therefore MS Windows remote shares have to be dealt with differently. See the next answer for details.

4.2.4 How to mount a remote MS Windows filesystem through Samba?

A remote MS Windows filesystem can be mounted onto a Linux filesystem through the Samba protocol (Samba must be installed, go here if it isn't). Type a command like this (as root):

```
smbmount //mars/windows /mnt/mars_windows −c marie
```

This mounts the MS Windows resource called windows from the MS Windows machine called mars. The mountpoint on the client computer is `/mnt/mars_windows/`. The option "−c" specifies that the samba server is a machine called marie (this should not be necessary, but it is on my system).

For the above to work, the permission must be given on the MS Windows machine for sharing the directory or drive as a resource. To do this, on the MS Windows machine, enable the filesharing using the "control panel−network", then launch the "Windows Explorer", click the right mouse button on the drive or directory to share, click on properties, switch to the page "sharing", give yourself the permission and give the resource a name.

To unmount an MS Windows directory use:

```
smbumount /mnt/mars_windows
```

If you have problems, see:

```
man smbmount
```

4.2.5 Any quick way to access a file on a DOS/Windows floppy?

Use "mtools", no mounting required. For example, I can use the `mdir` command to quickly inspect the contents of the root directory on my DOS floppy:

```
mdir a:\
```

I can also use `mcopy` to copy the file "autoexec.bat" from the root directory on the floppy to my current directory on Linux:

```
mcopy a:\autoexec.bat .
```

You have to be root to be able to write to a floppy.

Type "mtools" to see the supported commands in the rich mtools set, which parallel the most popular DOS commands (for example: `mformat`, `mtype`, `mren`, `mmove`, `mdel`, `mrd`, `mattrib`, ...), and use manual pages if you have problems using them. For example:

```
man mtype
```

will show me how to display the contents of a text file on a DOS partition.

To access DOS drives other than a: or b:, you have to configure mtools so as to indicate which devices are associated with other DOS "drive letters". This is quite easy−−you just edit and modify the file `/etc/mtools.conf`. I typically use pico to do it (as root):

```
pico /etc/mtools.conf
```

For example, my `/etc/mtools` contains a line like this:

```
drive c: file="/dev/hda1"
```

which instructs the mtools that the partition "/dev/hda1" will be called "c:" The setup of `/etc/mtools.conf` requires just uncommenting (removing the "#" at the beginning of the line) and adjusting the appropriate entry.

4.2.6 Mounting works when I am root. Can a normal user mount?

You have to edit the file `/etc/fstab` as root to give the normal users the permission to mount a particular drive. For example I can use the `pico` text editor to do this:

```
pico −w /etc/fstab
```

The option "−w" turns off the long line wrap.

Here is the content of my /etc/fstab:

```
/dev/hda2 / ext2 defaults 1 1
/dev/hdc3 /home ext2 defaults 1 2
/dev/hdc2 /usr ext2 defaults 1 2
/dev/hdc4 swap swap defaults 0 0
/dev/fd0 /mnt/floppy auto noauto,users,rw 0 0
/dev/cdrom /mnt/cdrom auto noauto,user,ro 0 0
/dev/sda4 /mnt/zipdrive vfat noauto,user,rw,exec 0 0
/dev/hda1 /mnt/dosdrive vfat noauto,user,rw 0 0
none /proc proc defaults 0 0
hacker:/mnt/cdrom /mnt/hacker_cdrom nfs noauto,user,ro 0 0
hacker:/mnt/floppy /mnt/hacker_floppy nfs noauto,user,rw 0 0
hacker:/home /mnt/hacker_home nfs noauto,user,rw 0 0
hacker:/usr /mnt/hacker_usr nfs noauto,user,rw 0 0
```

Each line contains six space−delimited fields (this means that each line has six entries separated by white space). The first field is the name of the device. The second field is the mountpoint (an existing directory on your Linux system to which the resource will be mounted). The third is filesystem type. For removable media that may contain filesystems of several types, I use the option "auto" to let Linux probe which filesystem is currently present there. (The order in which they are probed is determined by the content of the file `/etc/filesystems` . You may want to make sure that it specifies "vfat" before "msdos" or the long DOS filenames may be cut short.) The fourth field contains options: "auto" = mount the filesystem on the system startup; "rw" = read and write allowed; "ro" = read only, "user" = users have the permission to mount this filesystem (one can also use "users" to allow a user to mount and another user to unmount−−otherwise only the user that mounted the filesystem can unmount it), "exec" execution of programs is permitted from this filesystem. The number in the field 5 specifies if the filesystem is to be backed up during a system backup, the number in the field 6 determines if to check up the filesystem integrity during bootup. The hacker stuff in my `/etc/fstab` are filesystems on another computer (called "hacker") on my home network and it serves here as an example of how to mount network resources. Check `man fstab` for more info.

For example, if regular (non−root) users have the permission to mount the cdrom (the "user" option is specified), they can mount it using a command like this:

```
mount /mnt/cdrom
```

The command which the root uses for mounting (see here) will not work for a regular user because the regular user is restricted by the options in `/etc/fstab` and therefore s/he cannot specify simultaneously both the device and the mountpoint.

For a regular user to be able to write to a disk or execute a program on it, s/he must also be given the appropriate permission on the "mountpoint" directory. For example, this will give all the users all the permissions (read, write, execute) on the directory `/mnt/floppy` :

```
chmod a+rwx /mnt/floppy
```

Now (also the "rw" option is specified for the floppy in the `/etc/fstab`) the user will be able to write to a floppy. If the "exec" option was enabled in the `/etc/fstab`, the user would also be able to execute programs from the floppy.

Please note that the DOS vfat file system doesn't know about the file permissions the way Linux does. Linux manages this during mounting by giving the default file permissions on the mounted filesystem: the user who mounted the filesystem will be the owner of all files and will be given the right to write to the filesystem (if "rw" was specified in fstab) but other users can only read. If you wanted to change this behaviour, you could use the

"umask=" option so that the appropriate line in your `/etc/fstab` may look like this example:

```
/dev/sda4 /mnt/zipdrive vfat noauto,users,rw,exec,umask=000 0 0
```

This gives absolutely everybody all the permissions on your zipdrive (mounting, unmounting, read, write, execute).

To summarize, the file `/etc/fstab` is the place to keep your defaults on how to mount filesystems and what kind of access is allowed for users. You really want to customize it to simplify mounting on your system. Linux default mounting scheme is restrictive so as to be secure, you may want to remove some restrictions when setting up Linux at home.

4.2.7 Mounting command is too long, how can I simplify it with an alias?

An alias is an abbreviation of a more complex or often used command. For creating aliases, I edit, as root, the file `/etc/bashrc` . This way the aliases are available for all the users on the system. (For creating user−specific aliases, I edit the file `.bashrc` in the user home directory.) The relevant part of my `/etc/bashrc` looks like this:

```
alias cdrom="mount −v /mnt/cdrom"
alias ucdrom="umount −v /mnt/cdrom"
alias dosdrive="mount −v /mnt/dosdrive"
alias udosdrive="umount −v /mnt/dosdrive"
alias zipdrive="mount −v /mnt/zipdrive"
alias uzipdrive="umount −v /mnt/zipdrive"
alias floppy="mount −v /mnt/floppy"
alias ufloppy="umount −v /mnt/floppy"
```

The option "−v" stands for "verbose", i.e., it tells Linux to talk to me a lot during mounting. For the aliases to take effect, the user has to re−login. Now the user can mount the floppy using this simple command:

```
floppy
```

and s/he can unmount it using

```
ufloppy
```

4.2.8 Can I mount automatically?

Yes, you can automatically mount a filesystem as you access it and unmount when you stop using it. It works similar to what you have experienced under MS Windows. Yet, if you used removable media extensively under DOS or Windows, you must have noticed that automounting is not entirely foolproof.

There are two utilities for automounting under Linux, and they are callled "supermount" and "automount".

Supermount. Mandrake 7.2 gives you the option of using "supermount" as a setup option. So the simplest way to "supermount" is to install latest Mandrake and select this option. My /etc/fstab on a computer running Mandrake may contain the following lines:

```
/dev/hda3 / ext2 defaults 1 1
none /dev/pts devpts mode=0620 0 0
/dev/hda4 /home ext2 defaults 1 2
/mnt/cdrom /mnt/cdrom supermount fs=iso9660,dev=/dev/cdrom 0 0
/mnt/floppy /mnt/floppy supermount fs=vfat,dev=/dev/fd0 0 0
/mnt/zip /mnt/zip supermount fs=vfat,dev=/dev/zip 0 0
none /proc proc defaults 0 0
/dev/hdb2 /usr ext2 defaults 1 2
/dev/hdb5 swap swap defaults 0 0
```

In the example above, you may notice that I selected to supermount 3 filesystems: cdrom, floppy and zipdrive. I can edit the file /etc/fstab manually (e.g. with pico) or use the `supermount` command to customize the supermount to my needs.

Automount. To set up "automount", I first run the programs `ntsysv` (as root) and make sure that automount service ("autofs") is enabled.

Then, I configure automount by editing the files `/etc/auto.master` and `/etc/auto.misc`, e.g. (as root):

```
pico /etc/auto.master
```

My /etc/auto.master looks like that:

```
/misc /etc/auto.misc --timeout 1
```

This says that my automount devices will be mounted in the directory `/misc` (which must exist and be empty). My automount drives will automatically unmount one second after I stop using them (for example, after I exit the directory). This is a short time--you may choose a longer one. The detailed config file is `/etc/auto.misc` . Here is mine:

```
kernel -ro,soft,intr ftp.kernel.org:/pub/linux
cdrom -fstype=auto,ro :/dev/cdrom
floppy -fstype=auto,rw :/dev/fd0
zipdrive -fstype=vfat,rw :/dev/sda4
dosdrive -fstype=vfat,ro :/dev/hda1
hacker_cdrom -fstype=nfs,ro hacker:/mnt/cdrom
hacker_floppy -fstype=nfs,rw hacker:/mnt/floppy
hacker_usr -fstype=nfs,ro hacker:/usr
```

Each line consists of 3 space delimited fields. The first field is the "key" which will be the name of the subdirectory (under /misc) to which the device will be mounted. This directory must NOT exist. It will not be visible when I use the command `ls` , but I can "cd" to it and my device will then mount. Don't ask me why it is so, and how to use this automount in GUI. I don't know. The hacker stuff in my auto.misc is the cdrom and floppy from another computer in my home network.

I automount to the directory `/misc` (not `/mnt`) so that I can also mount filesystems manually, without using automount, to the directory `/mnt`.

4.2.9 How do I get my parallel-port (external) Zip drive recognized?

RedHat 6.0 and 6.1 The zipdrive (zip100 drive) installation did not work during my upgrade to RedHat 6.0 (the installation program said that it couldn't find the zipdrive). So, after the installation was completed, I issued the following commands to insert the modules for parallel port zip drive into the kernel (as root):

```
/sbin/insmod parport
/sbin/insmod ppa
```

To have these two lines executed automatically after each bootup, I added them at the end of the file `/etc/rc.d/rc.local` (this file is something like autoexec.bat on DOS).

If this still does not work for you, you may also want to edit the file `/etc/conf.modules`. Mine contains such a line:

```
alias parport_lowlevel parport_pc
```

and there is no line mentioning the "ppa" module.

For the newer Zip250 drive, I have the following two lines executed from my `/etc/rc.d/rc.local` file:

```
/sbin/insmod parport
/sbin/insmod imm
```

4.2.10 Can I set 32-bit hard drive I/O?

Newer Linux distributions (e.g., Mandrake 7.0) can automatically turn on the hard drive optimization: 32 bit input/outup (I/O) and direct memory access (DMA). Here is how to turn on the hard drive optimization manually. (Based on http://hardwarezone.community.com.sg/main.htm by Edward Choh.)

This procedure worked fine for me, yet be warned that it can possibly harm the content of your harddrive, so do not do it if you are a real PC newbie, don't feel geeky today, or have a weak heart−−I can't guarantee it will work for you.

The turning on of the 32−bit I/O and DMA has to be done by root, and I did it in a single−user mode (to minimize the damage to the file system if something went wrong and I had to reboot). I definitely would not do it on a system currently running many programs or X−windows, and would have a current backup of any precious data.

To boot your computer in a single−user mode, I type this at the lilo prompt (during bootup):

```
linux single
```

Say, I would like enable the 32−bit I/O on my first IDE harddrive, which is "hda". First, I time the harddrive current performance, and note the score:

```
hdparm −t /dev/hda
```

Now, I display my current I/O and DMA settings:

```
hdparm −c /dev/hda
```

[my system showed 0, meaning that the 32−bit I/O is turned off and the default 16−bit access is used]

```
hdparm −d /dev/hda
```

[my system showed 0 again, meaning that the harddrive DMA access is turned off].

Now, I turn on the 32 bit IO and DMA:

```
hdparm −c 1 /dev/hda
hdparm −d 1 /dev/hda
```

Now, I can time the performance of the harddrive again to compare the score with the original one:

```
hdparm −t /dev/hda
```

If everything worked ok, and the performance has improved, I can "commit" the new settings, so they can survive a soft reset:

```
hdparm −k 1 /dev/hda
```

To have the new settings in effect every time you reboot the machine, you may add a line at the end of the file `/etc/rc.d/rc.local` (this file is something like `AUTOEXEC.BAT` in DOS):

```
hdparm −c 1 −d 1 −k 1 /dev/hda
```

If something did not work as expected, or the performance did not really improve, I can reboot at any time and the old settings will be in effect as long as I did not perform the last operation.

I performed this tune−up on 4 hard drives on my home network. It was a success on 3 newer hardrives: the performance improved by 30−300% and at least one computer "feels" faster than before. One harddrive (which is always flaky) hanged the computer hard during the performance test and I had to reset the machine (no damage done).

4.2.11 I reached the limit on the number of opened files (error message)

You can increase the limit via the /proc file system. This file system is entirely virtual−−it is just a "window" to see or set some parts of the Linux kernel. To read the maximum number of simultanously opened files on my system, I use the following command:

```
cat /proc/sys/fs/file−max
```

On my system (Mandrake 7.2), the limit is 8192. To increase it, I use (as root):

```
echo 16000  > /proc/sys/fs/file-max
```

You may also want to increase the limit on a related kernel variable:

```
echo 30000 > /proc/sys/fs/inode-max
```

To make the changes permanent, add the above lines at the end of your startup script `/etc/rc.d/rc.local`

To learn more about the /proc Linux kernel interface, the meaning of the variables it contains, and their recommended values, you may wish to read (if you installed the Linux kernel source codes, which is a great resource even for a newbie):

```
less /usr/src/linux/Documentation/proc.txt
```

or (on RedHat 8.0)

```
man proc
```

4.2.12. I attached a new hard drive. What do I do to start using it?

0. Plan the layout of the filesystem with the new drive
1. Partition the new hard drive
2. Format the new partitions
3. Test the new space
4. Copy data from old to new partition (optional)
5. Edit /etc/fstab
6. Reboot
7. Remove old data (optional)

Here is a longer version for my recipe.

0. Plan the new filesystem. Where would you like to use the new space? Do
`df`

to print a summary of free/used space on each of the existing mounted partitions. Do
`du`

on selected directories to find their size.

For example, I would consider using new hard drive space in one of the following mount points:
`/usr/local`
`/home`
`/home/share/downloads`
`/usr/local/mp3s`
`/usr/local/dos_data`

Here is why.

`/usr/local` is suposed to survive any upgrade of Linux. It is nice to have it on a separate partition because I can even reformat other partitions without affecting my local contents stored in /usr/local. I surely want it of the type "ext2" or perhaps "reiserfs".

`/home` contains user data. Surely, it is the data that deserves the most care. It is obviously supposed to survive any upgrade of Linux. Wow, obviously I want it on a separate partition. The type is normally "ext2" or "raiserfs".

`/usr/local/mp3s` is a non−standard Linux directory. I may keep my MP3 (music) files there. Those tend to be large.

`/usr/local/dos_data`. Another non−standard directory. If I dual boot, I would consider making an extra partion of the type "DOS FAT32" or similar so as to share files between MS Windows and Linux transparently (both ways). I would configure all the Windows−based programs to use this "drive" as the default location for all user−generated files. I could even have "mp3s", "cds" and other such directories in this location. The serious drawback of this approach−−MS Windows may insist on messing up with this partion on re−install.

1. Partition the new hard drive. For example, if my new harddrive is the slave on the second IDE interface (perhaps the "fourth" IDE drive), I could use:

```
cfdisk /dev/hdd
```

or the more old−fashioned (and standard) tool:

```
fdisk /dev/hdd
```

If your drive is not "hdd" adjust the above command as needed.
hda −− first ide master (whole disk)
hdb −− first ide slave
hdc −− second ide master
hdd −− second ide slave
sda −− first scsi (whole disk)
sdb −− second scsi (whole disk)
...
sdp −− sixteenth scsi (whole disk)

For other disks, consult `/usr/src/Linux/Documentation/devices.txt`.

Most of the time, I want my Linux partitions to be of the type ext2 ("Linux").

Partitioning can be tricky−−if you never have done it, read man fdisk and man cfdisk. It is very easy to delete a partition with all your data. Make sure you know which disk you are working with!

fdisk or cfdisk does not make any changes to the hard drive until I write the new partition layout. So if I make a bad mistake, I can always quit without writing. I write the layout to the drive only when I am completely done.

2. Format each partition. For example, to format the first partition, while checking for bad blocks (−c), I would do:

```
mkfs −c −t ext2 /dev/hdd1
```

3. Test the new partitions around. Mount the new partitions manually (the mount directory must exist and be empty). Copy a bunch of files to each partition. View/edit a couple of random files. Delete them all.

4. Copy data. Optional−−only if you would like to move data from an old partition to a new partition. Go to the single−user mode (init 1). Mount the new partition manually. Copy the data from the old partition to the new partition. Careful with the old data, you probably don't want to lose it if you made a mistake, so I wouldn't delete it yet−−I rename the top level directory appropriately. E.g.,

```
cp −R /usr/local/ /mnt/hdd1/
mv /usr/local/ /usr/local.old.backup_of_2001−04−21
```

5. Edit the file /etc/fstab. Modify it to reflect your new filesystem layout. Perhaps, insert the mountpoint for the new partition(s) or modify any old mountpoints as needed. For example, if moving `/usr/local` to its own partition, I would need to add to add a line like this:

```
/dev/hdd1 /usr/local ext2 defaults 1 2
```

6. Reboot and test. The alternative to reboot is to unmount old and mount new mount points. For example:

```
umount /usr/local
mount −a
```

but hard reboot may be a more rigorous test of the new layout.

7. Remove old data. After a few days, when I have the confidence everything is really working fine, I delete the old data (e.g., the directory /usr/local.oldbackup_of_2001−04−21).

4.2.13 Swap space

Swap is an extension of the physical memory of the computer. Most likely, you created a swap partition during the initial RedHat setup. You can verify the amount of swap space available on your system using:
`cat /proc/meminfo`
The general recommendation is that one should have: at least 4 MB swap space, at least 32 MB total (physical+swap) memory for a system running command−line−only, at least 64 MB of total (physical+swap) memory for a system running X−windows, and swap space at least 1.5 times the amount of the physical memory on the system.
If this is too complicated, you might want to have a swap twice as large as your physical (silicon) memory, but not less than 64 MB.
If you ever need to change your swap, here are some basics.

4.2.13.1 Swap partitions

You can have several swap partitions. [Older Linux kernels limit the size of each swap partition to up to approximately 124 MB, but the linux kernels 2.2.x up do not have this restriction.] Here are the steps to create and enable a swap partition:
− Create the partition of the proper size using `fdisk` (partition type 82, "Linux swap").
− Format the partition checking for bad blocks, for example:
`mkswap −c /dev/hda4`
You have to substitute /dev/hda4 with your partition name. Since I did not specify the partition size, it will be automatically detected.
− Enable the swap, for example:
`swapon /dev/hda4`
To have the swap enabled automatically at bootup, you have to include the appropriate entry into the file `/etc/fstab`, for example:
`/dev/hda4 swap swap defaults 0 0`
If you ever need to disable the swap, you can do it with (as root):
`swapoff /dev/hda4`

4.2.13.2 Swap files

Swapping to files is usually slower than swapping to a raw partition, so this is not the recommended permanent swapping technique. Creating a swap file, however, can be a quick fix if you temporarily need more swap space. You can have up to 8 swap files, each with size of up to 16 MB. Here are the steps for making a swap file:
− Create a file with the size of your swap file:
`dd if=/dev/zero of=/swapfile bs=1024 count=8192`
This physically creates the swap file `/swapfile`, the block size is 1024 bytes, the file contains 8192 blocks, the total size is about 8 MB. [The `dd` command copies files. In the example above, the input file (if) was `/dev/zero`, the output file (of) was `/swapfile` . You cannot use the `cp` (copy) command for creating a swap file because the swap file must be physically continuous on the hard drive.]
− Set up the file with the command:
`mkswap /swapfile 8192`
− Force writing the buffer cache to disk by issuing the command:
`sync`
− Enable the swap with the command:
`swapon /swapfile`
When you are done using the swap file, you can turn it off and remove:
`swapoff /swapfile`
`rm /swapfile`
You may also want to see the nice info written by Linus Torvalds himself:
`man mkswap`

Go to Part: 4.3 − Working with X−windows

4.3: Working with X−windows

Contents of this section:

4.3.1 How to switch between text and graphical consoles?

Pressing the key combination `<Ctrl><Alt><F1>` will switch you to to the first text console at any time. `<Ctrl><Alt><F2>` will switch you to the second text console, `<Ctrl><Alt><F3>` to the third text console, etc, up to `<Ctrl><Alt><F6>`, for the total of 6 text consoles. `<Ctrl><Alt><F7>` will switch you to the first graphical user interface (GUI) console if one is running. `<Ctrl><Alt><F8>` to the second GUI console, etc., up to `<Ctrl><Alt><F11>` for the total of 5 GUI consoles. The 12th console is either used as the 6th GUI (RedHat 6.1) or a place to which kernel messages are continually displayed (Mandrake 7.0, really cool feature). Typically none or only the first GUI console is running.

`<Ctrl><Alt><F1>` means: "Press the left <Ctrl> and <Alt> keys and hold them. Now press <F1>. Release <F1>. Release <Ctrl> and <Alt> keys."

Thus, sitting at a Linux computer I can have many consoles opened at the same time, and I can switch between them using the hot keys as described above. I have to log in on each console to be able to use it−−I may log in as the same user (multiple times), or different users. Each login session is quite separate; they should not interfere with each other (the X sessions sometimes may if you log in as the same user twice). The first 6 consoles are text−based, command−line terminals (CLI, "command line interface") and are named tty1, tty2 ... tty6 (historical name, "tty" stands for "teletypewriter"). The subsequent consoles are graphical (GUI). These are all *local* consoles−−my local linux computer is truely muliuser and multitasking.

You can connect to a Linux computer remotely, over a network. While connected, you can have a program run on the remote Linux server and the display sent to your terminal on your "local" console. This local console can be Linux−based or another operating system−based. One cannot run programs remotely on an MS Windows server, but it is often used with Linux. It is really helpful to be able to distinguish if your program is run locally or remotely.

The full−screen text consoles are terminals by themselves. Under the GUI consoles, you can also create "slave" pseudo−terminals (in a window) on demand−−they will be called pts0, pts1, ...

Sitting at a text terminal, you can determine the name of your terminal using the following command:

```
tty
```

I can determine the computer on which my current session is located using:

```
uname −a
```

The name of the command "uname" is derived from "UNIX name". It shows the operating system name, the server name, the version of the operating system kernel, and the time of the compilation of the kernel.

You can close any text terminal by typing "exit" inside it.

The Linux GUI console is quite similar to other GUIs you might have used, e.g., MS Windows, but there are also numerous important differences. I love multiple "desktops" to stay organized−−the default setup in KDE offers 4 desktops (it can be customized to between 1 and 16). I run many programs and never close them, so to stay organized I keep a connectivity application (netscape with all its windows, knode newsreader, and licq) together

on Desktop2, abiword, staroffice and a text editor on Desktop3, games and konqueror with helpfiles on Desktop4, and leave Desktop1 for the more occassional chores. One can switch between desktops by clicking on the "desktop pager" on the "K−panel" or using <Ctrl><TAB>. To switch between applications on a same desktop, I may click the application window, or click its icon representation on the "applicaton panel", or use <Alt><TAB> to toggle between the applications. <Ctrl><Esc> will give me a list of the processes currently run on the local machine ("localhost").

4.3.2 How do I setup video card, monitor and mouse for the X−server?

This should be set−up during your Linux initial installation unless you skipped the step. To set it up now, you may try, as root, one of these text−mode configurators (as root):

```
Xconfigurator
XF86Setup
xconf
```

Under RedHat, you can also run the command `setup` (as root) and access Xconfigurator from there.

To setup X−windows under Linux, you may need to know your hardware. You may want to dust your monitor manual to see what maximum synchronization frequencies (vertical and horizontal) your monitor supports. The message when the computer boots may give you a clue about what type of video card you have and with how much video memory. Also running these commands will likely provide helpful information:

```
lspci
SuperProbe
```

Read the label underneath your mouse to find out about the mouse type. Next time you buy a mouse, get a 3−button "Linux−ready" Logitech or similar (Linux makes good use of all three mouse buttons). A standard (clone or not) mouse always makes a good sense−−I would never buy an unusual mouse because it may requires a weird driver or otherwise be a installation/functional pain.

During testing of the X−server, if the screen goes blank, displays funny lines, or otherwise obviously does not function as designed, kill it fast with `<Ctrl><Alt><BkSpace>` and re−check your monitor sync frequencies. Running too high frequencies can be harmful to your monitor.

If you really have problems, set up a plain vga X server first (resolution 640x480 pixels, 16 or 256 colours). You can fine−tune it later, after you get some understanding of how things work on your system, or perhaps with the help of some nicer setup tools available under X.

After setting up X, you can start it manually using:

```
startx &
```

The "&" makes your command run in the background so that your text terminal is not blocked. You could also use:

```
init 5
```

which will switch your system to runlevel 5, which means "the graphical user interface run level". To start X automatically (or not, your choice) on the system reboot, read the next few paragraphs.

4.3.3 Can I have a GUI login prompt?

To start your X−server automatically on the system start−up and display a graphical login prompt, you have to change (as root) just one character in the file `/etc/inittab` . This file specifies something like:

```
id:3:initdefault:
```

Change it to

```
id:5:initdefault:
```

This sets up the default runlevel to 5, which is X−Windows. The meaning of the different runlevels is explained in the same `/etc/inittab` file :

```
0 − halt (Do NOT set initdefault to this)
1 − Single user mode
2 − Multiuser, without NFS (The same as 3, if you do not have networking)
3 − Full multiuser mode
4 − unused
5 − X11
6 − reboot (Do NOT set initdefault to this)
```

You can change the runlevel from the command line. E.g., this command (has to be executed as root):

```
init 6
```

will reboot your computer, while the following command would switch your computer to a single−user mode:

```
init 1
```

To find out which runlevel I am currently at, I use the command `runlevel`.

To fine−tune the appearance of my X login screen, I can use (under X):

```
kcontrol &
```

and select "System"−"Login Manager". I like a login screen with an analog clock, big font, the login name of the last user already typed in, and the focus pre−set on the password field in the dialog box.

4.3.4 How do I install kde (e.g., on RedHat 5.2)?

This section is only of interest only for those who run an older distribution.

RedHat 5.2 does not install kde by default because at the time of the RH5.2 release there was a problem with the license for a library that the kde uses. Otherwise, kde is a very good GUI, the library license problem is now solved, and the RedHats 6.0 up include kde (alongside the more ornamental GNOME). Still, the kde binaries are on your RedHat 5.2 CDs, you just have to install them yourself. (If you don't have the CDs, you can download the binaries from the kde site on the Internet.)

First check if your X−server works by executing:

```
xinit
```

[The X−server is a bare−bone X−windows system, without a "Windows Manager." You can execute your X−windows programs from here by typing the program name (with leading ./ or full path) in the X−terminal window, but you will not be able to move or resize the windows, add icons, etc.]

You can exit your X−server by typing exit in the X−terminal window, or pressing <Ctrl><Alt><Bkspace> to kill the X−windows server. (The last solution is perfect should your windows ever hang−don't reboot in such a case.) If your X−server does not work, see the next answer.

Now, login as root. Mount the RH5.2 CD:

```
mount /mnt/cdrom
```

Go to the proper directory:

```
cd /mnt/cdrom/kde/distribution/RedHat/i386/binary
```

It is useful to use the command line autocompletion (press Tab) when typing long paths or filenames.

Read the README file:

```
cat README−2rh51−rpms | more
```

Use the `rpm` "RedHat Package Manager" to install the necessary packages

```
rpm −iv packagename
```

The packages have filenames ending with .rpm. First install the qt libraries, then kde support, then kde libs, then kde base. If you choose the wrong order, the dependency check will fail and the package will not install (rpm will issue a message). This is not serious, just re−install the required package first, and then try the next package again. After you are done with the base, you may want to install all other packages for kde (util, admin, network, games, graphics, multimedia)−−they are not big, so you may consider installing them all. Finally, just to make sure that you installed everything type:

```
rpm −ivh  *.rpm
```

The options "vh" print some extra info. You will get some messages like "the package is already installed" If there is more than a screenful of them, you can scroll back using <Shift><PgUp>. If you really don't like the command−line−based rpm package manager, you may install the same packages using a GUI front to rpm called glint (available only in RH5.2). Just type "glint" in the X−windows terminal.

Now tell your system that kde is to be your default X−windows manager. In the user home directory, create an .Xclients file:

```
pico .Xclients
```

which contains just one line:

```
/opt/kde/bin/startkde
```

Type in the line and save the file. (Adjust the line as required so the location of the `startkde` file is correct.) Now, make the file executable to all users:

```
chmod a+x .Xclients
```

Check if the permissions were changed:

```
ls −l .Xclients
```

[Files with a dot at the beginning are not displayed by a regular ls command, there are something like hidden files under DOS. You must use its name or `ls −a` .]

If you created the file as root not the user, change the owner and the group of the file to the proper user:

```
chown user_name .Xclients
chgrp user_name .Xclients
```

That's it. Now typing `startx` should start your X−server with the kde as the windows manager.

4.3.5 How can I change my default desktop to KDE (or Gnome or yet another)

In my home directory, I create (or edit if it exists) the file `.xsession` using my favourite pico editor:

```
pico .xsession
```

[Pls note the dot at the beginning of the filename, files with names starting with dots are normally "invisible".] On my RedHat 6.2 system, the file contains just one line:

```
exec startkde
```

KDE clearly works best for me, although it feels heavy on older hardware or under a load. Here is my list of windows managers available on the RedHat or Mandrake installation CD:

`startkde`	(to run kde. on some systems, the command may be `kde`)
`gnome−session`	(to run Gnome)
`xfce`	(to run XFce, my favourite "lightweight" desktop)

```
afterstep      (to run afterstep)
AnotherLevel   (to run AnotherLevel)
fvwm2          (to run fvwm2)
fvwm           (to run fvwm)
```

Of course, the alternative windows manager will run only if it is installed on your system. The above windows managers are available on RH/Mandrake CDs for you to decide if you want to install them. I use almost exclusively KDE, although the other managers may be smaller and faster. Gnome is a famous X−windows project which is said to be more advanced and is prettier than KDE, but it is still quite buggy, so perhaps not recommended unless you don't mind occasional trouble. RH6.x contains both major X−windows systems, Gnome and KDE.

4.3.6 Can I have multiple sessions of X running at the same time?

Yes, you can. When you issue the first `startx` command on your system, it opens the first X−session on the default display 0. The second X−session must be opened on a different display. For example, this will open a second X−session on the display 1:

```
startx −− :1
```

You can have up to 6 concurrent X sessions. Use <Ctrl><Alt><F7> to switch to display 0, <Ctrl><Alt><F8> to second screen, etc. up to <Ctrl><Alt><F12> Try `man startx` if you need more info.

In a similar fashion, you can open another bare X−server session without a window manager. This will open one on display 2:

```
xinit −−  :2
```

On this bare−bone X−display I can run a different windows manager (so as to have two different ones running at the same time) by typing in the X−terminal window one of these (see the previous answer for more details):

```
startkde
gnome−session
xfce
afterstep
AnotherLevel
fvwm2
fvwm
```

4.3.7 Can my sister have second GUI login prompt so she does not have to kill my X−session to start hers?

To enable several concurrent GUI logins on different local consoles under RedHat (RedHat uses program `gdm` for graphical logins), I had to edit the file: `/etc/X11/gdm/gdm.conf` . I have the following entry at the end of this file to enable 4 login terminals <Ctrl><Alt><F7> to <Ctrl><Alt><F10>:

```
[servers]
3=/usr/bin/X11/X vt10
2=/usr/bin/X11/X vt9
1=/usr/bin/X11/X vt8
0=/usr/bin/X11/X vt7
```

Having four GUI lets me run KDE and GNOME at the same time on one computer with two GUI terminals spare, so my sister can login despite my having screensavers with password−protection.

To enable several concurrent GUI logins on different local consoles under Mandrake (Mandrake uses kdm for graphical logins), I had to modify two files: `/etc/X11/xdm/Xservers` to include something like:

```
:0 local /usr/X11R6/bin/X :0
:1 local /usr/X11R6/bin/X :1
:2 local /usr/X11R6/bin/X :2
```

and `/etc/X11/xdm/xdm-config` to copy all the settings for display 0 to display 1 and 2 so that it includes this:

```
DisplayManager._0.authorize:    true
DisplayManager._1.authorize:    true
DisplayManager._2.authorize:    true
DisplayManager._0.setup:        /etc/X11/xdm/Xsetup_0
DisplayManager._0.startup:      /etc/X11/xdm/GiveConsole
DisplayManager._0.reset:        /etc/X11/xdm/TakeConsole
DisplayManager._1.setup:        /etc/X11/xdm/Xsetup_0
DisplayManager._1.startup:      /etc/X11/xdm/GiveConsole
DisplayManager._1.reset:        /etc/X11/xdm/TakeConsole
DisplayManager._2.setup:        /etc/X11/xdm/Xsetup_0
DisplayManager._2.startup:      /etc/X11/xdm/GiveConsole
DisplayManager._2.reset:        /etc/X11/xdm/TakeConsole
```

This enables me to run gnome and kde at the same time on a single computer with the third GUI terminal spare.

4.3.8 How to X−window remotely?

− Start X−server on the local machine, e.g.

```
xinit
```

− From the x−terminal give the remote machine the permission to display on your local screen:

```
xhost name_of_the_remote_server
```

In the really secure environment of my house, I could even give all servers the permission to display on my screen using (don't do it when connnected to the Internet):

```
xhost +
```

− Telnet the remote server.

− Start an X−program on the remote server directing the display on your local screen, for example, you may start a window manager:

```
startkde −display local_machine_name:0.0 &
```

The symbol "&" puts the command in the background, so that your telnet window is still available to you.
The 0.0 means "display zero, screen 0", which is your first screen on the first display and makes sense since you can have many concurrent sessions of X running on your computer with Linux.

You don't have to specify the "−display" option if your environment variable DISPLAY specifies the correct location on your current terminal, which is the case on my systems by default, but not on everybody else's as I am told. You can check your DISPLAY setting using:

```
echo $DISPLAY
```

− After I finish my remote X session, I restore the access control to my X−server using:

```
xhost −name_of_the_remote_server
```

or

```
xhost −
```

> Example. This sequence of commands will run Netscape on the remote machine called marie, directing the display to the X−server with X−windows manager which runs on the local machine hacker:
>
> ```
> startx
> ```

```
xhost marie
telnet marie
[login]
netscape −display hacker:0.0 &
[do my stuff]
[logout]
xhost −marie
```

In principle, you can run a program on any computer on the network, and display the output on any other (not necessarily the one you are sitting at).

I use remote X−windowing a lot to run fat programs (kde, Word Perfect 8, and Netscape) on a slim machine (486−33, 8 MB mem) which would not be able to run those by itself. It is also a convenient and fast way to work with files on a remote system for which the nfs mount is not set up.

X−windows was designed to run remotely over the network. Remote X−windowing is a very powerful tool, on top of being quite a pleasant experience. Try it out.

You can even run a program on a remote Linux (or any Unix) computer and redirect the display to a local MS Windows machine if you install an X−windowing program for MS Windows. For a good overview of choices, see: http://www.linuxworld.com/linuxworld/lw−2000−09/lw−09−legacy_1.html

4.3.9 How do I install TrueType fonts from my MS Windows partition?

Some distributions come with a TrueType font server but no (or a limited choice of) TrueType fonts. You can install your own TrueType fonts though. Here is how I did it manually. Mandrake includes a GUI utility to transfer you MS Windows fonts to Linux, so you don't have to bother with the procedure below.

0. From under K−menu (KDE), select "System"−"Font Manager" (or equivalent) and note what fonts you have installed.

1. On the command line, check if the "free type" font server is installed:

```
rpm −q freetype
```

This queries (option "q") the rpm package manager for the package "freetype". If the package is installed, go to next step. If "freetype" is not installed, install it now from your distribution CD. "freetype" was installed on my system after a "full" RH installation.

2. As root, make a directory that is to hold your TrueType fonts:

```
cd /usr/X11R6/lib/X11/fonts
mkdir TrueType
```

This directory is referred to in the configuration file `/etc/X11/XF86Config` so make sure that the name of the directory is exactly as shown. If you would like to name the directory differently, you have to edit `/etc/X11/XF86Config` and make appropriate adjustments. My "default installation" RedHat contained such a line:

```
FontPath "/usr/X11R6/lib/X11/fonts/TrueType"
```

3. As root, copy your *.ttf files from the original location to the TrueType font directory that you just created. I took some TrueType from my MS Windows partition, you may need to use a different source location:

```
cd /usr/X11R6/lib/X11/fonts/TrueType
cp /mnt/dos_hda1/windows/fonts/my_private_fonts/*.ttf .
```

Before copying any fonts, make sure that it does not violate your licence agreement.

4. As root, run the following commands:

```
cd /usr/X11R6/lib/X11/fonts/TrueType
ttmkfdir > fonts.dir
```

```
cp fonts.dir fonts.scale
```

5. Close all X−windows applications and log out from X−windows.

6. As root, restart your X−font server (or reboot your computer):

```
/etc/rc.d/init.d/xfs stop
/etc/rc.d/init.d/xfs start
```

7. Log back onto your KDE, and from under K−menu, select "System"−"Font Manager" to see if the fonts installed correctly.

This parts is based on: http://www.computerbits.com/archive/20001000/linux0010.htm

4.3.10 How do I copy−paste?

Under X−windows, use your mouse: highlight the text to be copied, switch to the location where to copy, and press the middle mouse button to paste. This works nice and fast, as long as during the switching you don't disselect the original text. If your mouse has only two buttons (no middle button), use "both buttons together" or perhaps the right button (which combination works depends on your setup). This is the traditional "X−Windows style" copying.

Many GUI applications (but not all) also support the Mac/MS−Windows−style "copy−paste": Select the text. Use the menu item "edit"−"copy" (either from the pull−down menu, or a local menu activated with the <RightMouseButton>). Switch to the location where to copy to. Use the menu item "edit"−"paste". This works fine for applications which use the same toolkit (e.g. KDE or GNOME) but does not always work across toolkits (e.g., from a GNOME application to the KDE application).

As a keyboard shortcut for the last method, I can use <Ctrl><c> for copying the highlighted text and <Ctrl><v> for pasting. Text can be highlighted without mouse using <Ctrl><Shift><RightArrow>.

You may also use the cut−paste history. Try running `klipper` (in X−terminal, KDE).

The two copying methods are supposed to be separate; therefore, they should not mutually interfere.

To capture the content of a window or the entire screen to a graphics file, I use `knapshot`. Alternatively, I can use `<Alt><PrintScreen>` to take a snapshot of the current window into the clipboard, and `<Ctrl><Alt><PrintScreen>` to take a snapshot of the entire desktop into the clipboard.

To catch contents of a text console (outside of the GUI console), I could use in X terminal (probably as root):

```
cat /dev/vcs1
```

and then copy and paste whatever I need from the X terminal with a mouse. I need to adjust the number in "vcs1" if my terminal to capture is not terminal 1.

The text−based consoles support the mouse if you run the `gpm` daemon. Type `gpm` to test it−−it will run fine if your mouse is appropriately configured. (You may want to run `mouseconfig` to configure your mouse.) To have `gpm` start automatically on the system startup and stay running, select the gpm daemon using the `ntsysv` utility. Use gpm exactly the same as the GUI cut−paste: highlight the text to be copied, move the text cursor to the "copy to" location, and then press the middle mouse button (or both buttons at once for a two−button mouse) to paste.

4.3.11 How do I Display and Control a Remote Desktop using VNC

(VNC = Virtual Network Computing). A very useful application−−don't miss it.

VNC is a cross−platform utility that allows me to display a remote graphical desktop over a standard network connection. For example, I can use VNC on an MS Windows PC to display an X−window environment of my mighty Linux server downstairs, or the other way around. VNC will even run over a 56k modem networking, but probably only for fun or in emergency (too slow a connection for normal work).

Recent Mandrake or RH will have vnc on their distributions CD. The MS Windows version you have to download yourself. See http://www.uk.research.att.com/vnc/ for download information and more details.

On Linux, VNC consists of four commands: vncserver, vncviewer, vncpasswd, and vncconnect. I typically need just two of them: vncserver and vncviewer. A brief description of the commands follows.

`vncserver`
The server that has to be running on the host (remote) computer. You start the server as the user whose desktop will be displayed (don't run the server as root or somebody else somebody may kidnap your computer!).

`vncviewer`
The local application wich connects to the vncserver and displays the remote environment. You need to know the password and ip address of the server to connect.

`vncpasswd`
Password selection utility for vncserver. The server won't run without password (good behaviour). Therefore, if you don't select one, it will prompt you. Hence, I don't need to explicitly run `vncpasswd`.

`vncconnect`
Tells vncserver to connect to a listening VNC viewer on the given computer and port. This way I can avoid giving anybody a password.

`Xvnc`
A "master" program that I don't really need to run directly (`vncserver` and `vncviewer` are scripts which call `Xvnc`).

For a list of all available options I run:
`Xvnc -help`

It is not recommenced to run the VNC server as root due to potential security issues. If you need root privileges, login as a user and then execute `su`

Two examples of "typical" sessions follow.

Example 1. Sitting at an MS Window computer, I can display an X environment from my Linux server, using the following sequence:

[start a DOS terminal and type in the following command]
`telnet` *`my_linux_server_name`*
[log in to your user account on Linux and type in it the following command]
`vncserver`
[provide a really good password of your choice when prompted; mine was "357+Simon&Garfunkel"]
[re−enter the same password for confirmation]
[watch the messages and note the screen number on which the server is started; mine was ":4"]
[From the "Start" menu on the MS Windows computer, select "Programs" − "Vnc" − "Run VncViewer"
[in the input box that appears, type the server ip address and screen number as shown on the next line]
`my_linux_server_ip_address:4`
[in the input box that appears type the password as follows]
`357+Simon&Garfunkel`
[an X−windows desktop should now appear on top of your MS Windows desktop]
[do your work as you normally would in Xwindows]
[when done, switch to the telnet session window and type in it the following two commands]
`vncserver − kill :4`
`logout`

Example 2. Sitting at my Linux X desktop, I can display and remotely control an MS Windows computer screen. Hopefully, nobody else is using this MS Windows computer at the same time, because I move its mouse pointer.

[Walk to the MS Windows computer because you probably cannot telnet it]
[From the "Start" menu, select "Programs" − "Vnc" − "Run WinVnc (app mode)"
[From the "System Tray", click the mouse right button on the "Vnc" icon, and select "Properties"]
[In the dialog box that appears, fill in the password. Leave the screen number on "auto".]
[Walk back to your Linux desktop]
[Start an X terminal and type in it]
`vncviewer ms_windows_server_name_or_ip`
[When prompted, type in the password]
[a MS Windows desktop should now appear on top of your X]

[do your work as you normally would on MS Windows]
[When done, right click on the Vnc icon in the system tray and select "Close VNC".]

Go to Part: 4.4 − Basic Configurations (Printer, soundcard...)

4.4: Basic Configurations

Contents of this section:

4.4.1 How to setup my soundcard?

Try to run (as root)

```
sndconfig
```

Unless you have a very fancy sound card, this will work for you. At the end of the setup, Linus says how he pronounces "Linux". (On RedHat, "sndconfig" can be also run via the "setup" utility−−type `setup`.)
You may want to try your soundcard and cdrom using a command line cdplayer. Put a music CD to your CDROM and type:

```
cdplay
```

If this does not work, maybe you don't have `/dev/cdrom`? Check if you can mount a data CD as root (look here) and create the device `/dev/cdrom` by linking it to the appropriate drive (most likely `/dev/hdb`), for example (as root):

```
ln −s /dev/hdb /dev/cdrom
```

If `cdplay` works for root, but does not work for a regular user, you may need to give (as root) everybody the permissions to read and write to the the file `/dev/cdrom` or review the permissions on /mnt/cdrom, or modify /etc/fstab as explained earlier, e.g.:

```
chmod 666 /dev/cdrom
```

(The directory `/dev` is where all your devices appear as files.)
To play third song, try:

```
cdplay play 3
```

You can also use the command `cdp` for rudimentary command line interface to cdplay, but perhaps you prefer the interfaces available from under X−windows (e.g., from KDE "K" menu, choose: Multimedia−"CD Player").

To stop the music either press the button on your CDROM or issue one of these commands:

```
eject
cdplay stop
```

Troubleshooting. If you are having problems with a sound card, a manual configuration is an option. Here is my setup for a SoundBlaster16−compatibile ("no name") soundcard that persistently played at half speed (too slow and with "low voice") because it was misdetected. The resource to read turned out to be: /usr/src/linux−xxx/Documentation/sound (hope you installed the kernel source code so that you have the documentation). The file to edit is `/etc/modules.conf`. The critical line in this file (after manual edition) is:

```
options sb esstype=1688 io=0x220 irq=5 dma=0 dma16=5 mpu_io=0x330
```

The change that I had to make was to insert the "esstype=" option. After modification, the best to test your setup is to cold reboot (shutdown to a halt, and then recycle the power).

4.4.2 How do I setup my printer?

Start an X−terminal (perhaps by pressing the proper button) and type in it (as root or you will be prompted for the root password):

```
printtool &
```

This program does a complete printer setup, you just have to fill up the information about your type of printer and where it is hooked up.

Specifying the proper printer port is the most important part. If you don't know which one is yours try: **on RedHat 5.2:** lp1 (this is the first parallel port on RH5.2) or lp2 (this is the second parallel port on RH5.2) or lp3 (this is the third parallel port on RH5.2); **on RedHat 6.0 (or later):** lp0 (this is the first parallel port on RH6.x) or lp1 (this is the second parallel port on RH6.x) or lp2 (this is the third parallel port on RH6.x). After upgrading from RH5.2 to 6.0, the printing stopped working because the name of the parallel ports changed. I had to re−run the printool and adjust the port. The numbering of ports changed to bring it in line with numbering of other devices, which always starts from 0.

Try printing an ASCII test−page straight to the port. Only when this works set up the bells and whistles.

If you are setting up a remote printer, make sure that your machine has the permission to use the remote printer. The permissions are set in the file `/etc/hosts.lpd` (more secure) or `/etc/hosts.equiv` (less secure) on the machine to which the printer is attached. These files simply list the names of the remote computers that can use a local printer, one computer name per line. Mine looks like this:

```
hacker
mars
```

The file `/etc/hosts.lpd` did not exist on my system, so I created it.
For quick information about the printers on your machine, you may want to view the file `/etc/printcap` :

```
cd /etc/
cat printcap
```

Here is the meaning of some codes that I see in my `/etc/printcap`:

:	Field separator (separates the entries in the file).
\	(at the end of line) Continuation on the next line.
lp	Name of the printer. "lp" is the name of the default printer on your machine. Subsequent printers are often, by default, given the the names lp0 or lp1, ... (or whatever you like) but this should not be confused with the name of the devices (parallel ports) to which they are connected.
sd=/var/spool/lpd/lp	My spool directory (sd).
mx#0	Maximum size of print jobs (mx) in blocks. "0" means no limit.
sh	I want headers to be suppressed (sh). Header is the page with your name that prints before your printing job (waste of paper if you print at home).
rm=mars	Name of the remote machine (rm), which on my system is called "mars (my printer is connected to a different computer).
rp=lp or lp=/dev/lp0	Name of the remote printer (rp), which is the name of the printer on the remote machine ("lp" on "mars" on my home network) or the name of the device on the local machine. "/dev/lp0" is the first parallel port on RH6.x (it used to be /dev/lp1 on RH5.2, the numbering of parallel ports changed).
if=/var/spool/lpd/lp/filter	Input filter (if). Your printing job will be formatted by this "filter" before it is sent to the printer.
sf	Suppress the form feed (sf) that is normally sent when printing is completed (use it if your printer keeps printing an empty page at the end of each jobs).

The printer is controlled using the command `lpc` (as root). Type "?" to see the options. This program is notorious for its peculiarities,

so don't get discouraged easily. The printer queue can be viewed with `lpq` and cleaned up with `lprm` , both of which work for a user (not only root). You can print from the command line using the command `lpr`. Under KDE, you can control the printer queue from the program available under the "K−button"−"Utilities"−"Printer Queue".

Most printers will work perfectly under Linux, but some may not utilize their full capablity due to lack of information/drivers from the vendors. Therefore, when purchasing a new printer, you may want to consult the database of Linux printers: http://www.linuxprinting.org/database.html. In brief, it is a good bet is to select (http://www.linuxprinting.org/suggested.html):

− For inexpensive colour printing: an Epson Stylus, for example: Stylus C80 (better) or Stylus C60 (cheaper) (Dec.2001). HP inkjets are generally less recommended than Epson's. Please note that "inkjet−type" printers are (in general) "not−so−great" for black−and−white printing. Also, they are meant to be "personal" printers and do not handle well high volumes. Yet they can offer excellent colour output, particularly on good quality paper. Kids love inkjets.
− For low−end laser printing: a Lexmark or Brother printer. Many Hewlett−Packard (HP) laser printers will also work perfectly, but one has to be more careful when selecting an HP printer due to their more limited support. Lower−cost laser printers are always black−and−white, but they offer excellent quality text printouts. You may avoid some headaches if you select a printer which supports "Postscript".

4.4.3 Word Perfect 8 does not have a driver for my printer

If you installed your printer in KDE using the printtool and it had a driver which works fine, set up Word Perfect to print using the "passthru postscript" driver.

4.4 4 Where are the setup and configuration files?

System−wide settings settings are stored in the `/etc` directory. User−specific settings are stored in the user home directory `/home/user_login_name`.

Here is a listing of some system−wide configuration files that I use most often:

SHELL DEFAULTS
`/etc/bashrc` − system−wide default functions and aliases for the bash shell
`/etc/profile` − system−wide defaults for bash shell, including system−wide environment variables.

ADMINISTRATIVE SETTINGS
`/etc/passwd` − contains passwords and other information concerning users who are registered to use the system. It can be modified by root directly, but it is preferable to use a configuration utility such as passwd to make the changes. A corrupt /etc/passwd file can easily render a Linux box unusable.
`/etc/shadow` − contains "shadow" information for the passwd file, i.e., the information pieces which "the world" does not have permission to read.
`/etc/group` − similar to /etc/passwd but for groups.
`/etc/crontab` − setup for "cron", which runs commands periodically (hourly, daily, weekly, monthly, etc.).
`/etc/inittab` − runs different programs and processes on startup.
`/etc/issue` − message that accompanies login prompt. This is often overwritten by the rc.local script.
`/etc/issue.net` − same as above, but used when login is attempted over the network.
`/etc/motd` − "message of the day" file, displayed after a user logs in.
`/etc/rc.d/rc.local` − the last script to execute on the system bootup. I put the commands which customize my local machine at the end of this file. It works like DOS "autoexec.bat".

NETWORK CONFIGURATION
`/etc/hosts` − contains a list of host names and absolute IP addresses.
`/etc/hosts.allow` − hosts allowed to access Internet services
`/etc/hosts.deny` − hosts forbidden to access Internet services
`/etc/resolv.conf` − setups for a list of domain name servers used by the local machine
`/etc/inetd.conf` − configures the inetd daemon to tell it what TCP/IP services your machine should run.
`/etc/exports` − specifies hosts to which file systems can be exported using NFS (network file system). `man exports` contains information on how to set up this file for remote users.

HARDWARE CONFIGURATION
`/etc/conf.modules` − setup for the linux kernel modules. Modules are like "device drivers" under MS Windows or DOS.
`/etc/fstab` − contains information on partitions and filesystems used by system to mount different partitions and devices on the directory tree.
`/etc/mtab` − shows currently mounted devices and partitions and their status.
`/etc/lilo.conf` − configuration file for `lilo` boot loader.

`/boot/grub/grub.conf` − configuration file for `grub` boot loader.
`/etc/printcap` − setup for printers.
`/etc/termcap` − ASCII database defining the capabilities and characteristics of different consoles, terminals, and printers. You typically don't want to change these.
`/etc/X11/XF86Config` − X configuration file. For XFree version 4.xx, the file is `/etc/X11/XF86Config−4` (if it does not exist, then XF86Config is tried).

4.4.5 What are all the device files?

Devices appear as files in the directory `/dev`. They can be read, or written to, if you have the permission to do so. The listing of the file reveals some important details about the device, for example:

```
ls −l /dev/ttyS3
```

on my system produces the following output:

```
crwxr−xr−x    1 root     tty        4,  67 Mar 13 22:59 ttyS3
```

The initial "c" indicates a character device. "b" would mean "block device", "p"=FIFO device, "u"=unbuffered character device, "d"=directory, "l"=symbolic link. The numbers "4, 67" mean that the device major number is 4 and the minor number is 67. To make some devices usable to all users on your system, you may need to set the proper permissions. For example:

```
ls −l /dev/usb/scanner0
chmod 666 /dev/usb/scanner0
```

Here is a list of some common devices:

`/dev/ttyS0` − the first serial port. The mouse is typically connected here.
`/dev/ttyS1` − the second serial port. This may well be the device to which your modem is connected.
`/dev/ttyS2` and `/dev/ttyS3` the third and fourth serial port (typically not present, but your internal modem may well be configured as one of these).
`/dev/modem` − the serial modem. In the typical case, a symbolic link to /dev/ttyS1, /dev/ttyS2, /dev/ttyS3 or /dev/ttyS0, depending to which serial port your modem is connected.
`/dev/mouse` − mouse. In the typical case, a symbolic link to /dev/ttyS0 or similar (see above), depending to which serial port your mouse is connected.
`/dev/lp0` − printer on the first parallel port. That's where normally printers are connected.
`/dev/lp1` − printer on the second parallel port (typically not present).
`/dev/fd0` − first floppy disk drive (almost always present).
`/dev/fd0H1440` − driver for the first floppy drive in the high−density mode (1440 kB). Generally, this (or a driver with a device with a similar descriptive name) is invoked when formatting a floppy drive to a particular density. Slackware also comes with drivers that allow for formatting a 3.5" diskette with up to 1.7MB of space. Red Hat and Mandrake do not contain these device drivers files by default.
`/dev/fd1` − second floppy disk drive.
`/dev/hda` − first IDE hard drive (whole drive). Most hard drives on IBM−compatibile PCs are IDE.
`/dev/hdb` − second IDE hard drive (whole drive). On many computers, the IDE cdrom drive is attached here.
`/dev/hdc` − third IDE drive (whole drive). On many computers, the IDE cdrom drive is attached here.
`/dev/cdrom` − a symbolic link to the appropriate drive interface, typically /dev/hdc or /dev/hdb (a CDROM) or /dev/scd0 (a CD−R/RW writer).
`/dev/hda1` − the first partition on the first IDE hard drive. /dev/hda2 is the second partion on the first IDE hard drive. As one could guess, /dev/hdd8 would be the eight partition on the fourth IDE hard drive.
`/dev/tty1` − the first text console. /dev/tty2 is the second text console, etc.
`/dev/dsp` − digital audio, i.e., the sound card. "dsp" stands for "digital signal processing".
`/dev/sndstat` − do `cat /dev/sndstat` to learn about the status of your sound devices.
`/dev/null` − used when you want to send output into oblivion.
`/dev/random` − used to read pseudo−random numbers. Do `cat /dev/random` to display garbage−looking characters on your screen. There is also /dev/urandom to generate lower−quality random sequences.
`/dev/sda` −the first SCSI drive (whole drive). On a home machine, you are unlikely to have any SCSI drives (expensive).
`/dev/sdb` − the second scsi drive ("sdc" is the third scsi drive, etc. There can be many scsi drive on a system).
`/dev/sda1` − the first partition on the first scsi drive.
`/dev/sr0` − the first scsi CD drive (sometimes called /dev/scd0). If you have an ATAPI CD writer, it will also be likely here.
`/dev/sr1`−is the second scsi CD drive (sometimes called /dev/scd1), (`/dev/sr2` is the third scsi CD drive, etc. There can be many scsi CD drives on the system).
`/dev/usb/scanner0` − a usb scanner. Try: `less /usr/src/linux/Documentation/usb/scanner.txt` for an info on scanner configuration from scratch.

For more info try:
```
less /usr/src/linux/Documentation/devices.txt
man MAKEDEV
```

As explained in /usr/src/linux/Documentation/devices.txt, I may need to create some symbolic links to device files locally to configure my system. This is merely a tabulation of existing practice, and does not constitute a recommendation. However, if the links exist, they should have the following uses:

```
/dev/mouse      Current mouse port***
/dev/tape       Current tape device
/dev/cdrom      Current CD-ROM device***
/dev/cdwriter   Current CD-writer device (but my RedHat have /dev/cdrecorder)
/dev/scanner    Current scanner device
/dev/modem      Current dialout (modem) port***
/dev/root       Current root filesystem
/dev/swap       Current swap device
```

The *** mark the symbolic links that are surely present on my Mandrake system. For example, if having problems with mouse I would do something like (as root):
`ls -l /dev/mouse`
[see if the mouse device is present and where it points]
`ln -s /dev/ttyS0 /dev/mouse`
[create a symbolic link so that /dev/mouse point to the first serial port]

For SCSI (and ATAPI) devices, /dev/tape and /dev/cdrom should point to the ``cooked" devices (/dev/st* and /dev/sr*, respectively), whereas /dev/cdwriter and /dev/scanner should point to the appropriate generic SCSI devices (/dev/sg*).

Non-transient sockets and named pipes may exist in /dev. Common entries are:
```
/dev/printer    socket     lpd local socket
/dev/log        socket     syslog local socket
/dev/gpmdata    socket     gpm mouse multiplexer
```

Some Linux daemons

Daemons are "resident"programs that periodically wake up, check your system and may perform certain functions. They do not take any input and don't normally produce any output. Your Linux system is likely set to run quite a number of daemons. Most of them can be (dis)selected by running the program `ntsysv` (RedHat) as root and checking the appropriate box. The short description of each daemon is available under `netsysv` by pressing <F1>. If the daemon you need is not listed in `ntsysv`, you need to insert your RedHat/Mandrake installation CD and install the appropriate package. The alternative to `ntsysv` may be `tksysv` (type as root, in X terminal), which is perhaps more flexible, but way more complicated (it lets you set up the list of daemons to run in each runlevel). Another, simpler and even more powerful+flexible+difficult-to-use tool is `/sbin/chkconfig`.

Here is a short list of popular daemons with a brief description:

`anacron` - checks `cron' jobs that were left out due to down time and executes them. Useful if you have cron jobs scheduled but don't run your machine all the time--anacron will detect that during bootup.
`amd` - automount daemon (automatically mounts removable media).
`apmd` - Advanced Power Management BIOS daemon. For use on machines, especially laptops, that support apm.
`arpwatch` - keeps watch for ethernet/ip address pairings.
`atd` - runs jobs queued by the "at" command.
`autofs` - control the operation of automount daemons (competition to amd).
`bootparamd` - server process that provides information to diskless clients necessary for booting.
`crond` - automatic task scheduler. Manages the execution of tasks that are executed at regular but infrequent intervals, such as rotating log files, cleaning up /tmp directories, etc.
`cupsd` - the Common UNIX Printing System (CUPS) daemon. CUPS is an advanced printer spooling system which allows setting of printer options and automatic availability of a printer configured on one server in the whole network. The default printing system of Linux Mandrake.
`dhcpd` - implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP).
`gated` - routing daemon that handles multiple routing protocols and replaces routed and egpup.
`gpm` - useful mouse server for applications running on the Linux text console.
`httpd` - daemon for the Apache webserver.
`inetd` - listens for service requests on network connections, particularly dial-in services. This daemon can automatically load and unload other daemons (ftpd, telnetd, etc.), thereby economizing on system resources. Newer systems use `xinetd` instead.

`isdn4linux` − for users of ISDN cards.
`kerneld` − automatically loads and unloads kernel modules.
`klogd` − the daemon that intercepts and displays/logs the kernel messages depending on the priority level of the messages. The priority is (copied from `/usr/include/linux/kernel.h`):

```
KERN_EMERG      "<0>"  system is unusable
KERN_ALERT      "<1>"  action must be taken immediately
KERN_CRIT       "<2>"  critical conditions
KERN_ERR        "<3>"  error conditions
KERN_WARNING    "<4>"  warning condition
KERN_NOTICE     "<5>"  normal but significant condition
KERN_INFO       "<6>"  informational
KERN_DEBUG      "<7>"  debug-level messages
```

The messages typically go to the appropriately named files in the directory /var/log/kernel.

`kudzu` − detects and configures new or changed hardware during boot.
`keytable` − loads selected keyboard map.
`linuxconf` − the linuxconf configuration tool. The automated part is run if you want linuxconf to perform various tasks at boottime to maintain the system configuration.
`lpd` − printing daemon.
`mcserv` − server program for the Midnight Commander networking file system. It provides access to the host file system to clients running the Midnight file system (currently, only the Midnight Commander file manager). If the program is run as root the program will try to get a reserved port otherwise it will use 9876 as the port. If the system has a portmapper running, then the port will be registered with the portmapper and thus clients will automatically connect to the right port. If the system does not have a portmapper, then a port should be manually specified with the −p option (see below).
`named` − the Internet Domain Name Server (DNS) daemon.
`netfs` − network filesystem mounter. Used for mounting nfs, smb and ncp shares on boot.
`network` −activates all network interfaces at boot time by calling scripts in /etc/sysconfig/network−scripts .
`nfsd` − used for exporting nfs shares when requested by remote systems.
`nfslock` − starts and stops nfs file locking service.
`numlock` − locks numlock key at init runlevel change.
`pcmcia` − generic services for pcmcia cards in laptops.
`portmap` − needed for Remote Procedure Calls. Most likely, you need it for running network.
`postfix` − mail transport agent which is a replacement for sendmail. Now the default on desktop installations of Mandrake (RedHat uses sendmail instead).
`random` − saves and restores the "entropy" pool for higher quality random number generation.
`routed` − daemon that manages routing tables.
`rstatd` − kernel statistics server.
`rusersd`, `rwalld` − identification of users and "wall" messaging services for remote users.
`rwhod` − server which maintains the database used by the rwho(1) and ruptime(1) programs. Its operation depends on the ability to broadcast messages on a network.
`sendmail` − mail transfer agent. This is the agent that comes with Red Hat.
`smbd` − the SAMBA (or smb) daemon, a network connectivity services to MS Windows computers on your network (hard drive sharing, printers, etc).
`squid` − An http proxy with caching. Proxies relay requests from clients to the outside world, and return the results. You would use this particular proxy if you wanted to use your linux computer as a gateway to the Internet for other computer on your network. Another (and probably safer at home) way to do it, is to set up masquarading.
`syslogd` − manages system activity logging. The configuration file is /etc/syslog.conf .
`smtpd` − Simple Mail Transfer Protocol, designed for the exchange of electronic mail messages. Several daemons that support SMTP are available, including sendmail, smtpd, rsmtpd, qmail, zmail, etc.
`usb` − daemon for devices on Universal Serial Bus.
`xfs` − X font server.
`xntpd` − finds the server for a NIS domain and stores the information about it in a binding file.
`ypbind` − NIS binder. Needed if computer is part of Network Information Service domain.

Go to Part: 4.5 − Networking

4.5: Networking

Contents of this section:

4.5.1 Would it be worth it to set up my home network?

This is an excellent idea. It will let you use the machines that are in your closet now because they were not powerful enough to run standalone. The benefits include sharing hard drives, zipdrives, CDROMs, modem, printers, even soundcards, running programs remotely (the text or graphics mode), browsing the Internet on all computers at the same time over one phone connection. If you ever lose control of your machine, you may also be able to shut it down remotely and thus avoid possible problems−−see the answer on shutting down for details.

4.5.2 How to set up my home network?

Hardware. Your hardware must be set up properly. Your network card should have been set up during the initial RedHat installation. If you added your card later, chances are it was autodetected and configured during a subsequent bootup (by `kudzu`). If it wasn't, you may want to set up your network card now. If you have more than one network card on your computer, you will also need to set up the second cards manually, since Linux will autodetect only 1 network card.

Under Linux, most drivers for network cards are implemented as modules. So setting up a card manually involves just inserting the proper module with its parameters. You don't need to recompile the kernel, nor even reboot your computer (unless you have an uncommon card). To figure out what module(s) and parameters you need, you may want to consult the documentations that comes with the kernel source code:

```
less /usr/src/linux−2.4/Documentation/networking/net−modules.txt
```

If needed, you can list all the modules available for your kernel using something like:

```
modprobe −l | more
```

If you need more help, see the Linux Network Administrator Guide (file `/usr/share/doc/LDP/nag` on your system or check http://www.tldp.org/LDP/nag/). This excellent guide is known as nag.

There are many ways of inserting a module into a kernel. For a network card, the simplest is to start, as root, `netconf` and specify the module name there. An alternative is to start an X−terminal, execute "su" (to make yourself a root) and then run `/usr/bin/kernelcfg`

In most cases, you don't have to specify the parameters (IRQ and address) for your module−−the module will know what they should be. However, the parameters were a problem during my setup of two network cards on one computer−−you must make sure that you don't have any hardware conflicts. A common source of problems is that the card wants to configure on IRQ 5, which is occupied by the SoundBlaster, or IRQ 3 which conflicts with the second serial port (COM2, cua1, ttyS1). Inspecting the files `/proc/interrupts`, `/proc/ioport` and reading bootup messages may help.

For example, my WD8013 card (same as SMC Elite and SMC Elite plus, according to nag) runs under IRQ 10 (set by a jumper on the card and I specified the IRQ in the kernel module setup), under the address 0x300 and uses the "WD" module. My "SMC EtherEZ" card (no jumper settings on the card) runs under IRQ 9, address 0x240 and uses "SMC EtherEZ" module. Please make sure you don't omit the leading "0x" in the address−−it means "hexadecimal" and must be there else the number will be interpreted as decimal.

After the module is inserted, you may want to inspect the file `/proc/modules` to see if the module is indeed loaded (or run `lsmod` as root). The module configuration file is `/etc/modules.conf` so if you encounter difficulties (for example, I had difficulty removing modules inserted by mistake), just edit and adjust this file manually, e.g., using `pico`.

Network configuration. After setting up the network cards and connecting the cables, set up the network by running (as root):

`netconf`

This program has help! `netconf` can also be run under GUI, but I did not really try it. If you need more understanding of how networking works, you may want to read the previously mentioned nag. Also, reading this material may be of help: http://ieee.uow.edu.au/~mjp16/wylug−netlinux/notes.html.
It also contains very clear examples on how to set up a small office network that, like our home setup, has only part−time connection to the outside world.

How and what to fill up in `netconf` depends on your network. You surely want to fill up "Basic Host Information" (enable the first ethernet interface, eth0, fill in the name, aliases and IP number of your local computer) and the "Information About Other Hosts" (names, aliases and IPs of other computers on your home network). This information goes to the files: `/etc/hosts` and `/etc/sysconfig/network`, so you may want to inspect these files and adjust them manually.

When setting up the network, don't mess up with the "loopback driver" which has the IP 127.0.0.1. It is always there−−it is the IP through which the computer talks to itself.

If you don't have an IP address (as will typically be the case for a home network with no permanent connection to the outside world), you may want to invent one. It does not matter very much what it is since when connecting to the outside world by your modem, you will be dynamically allocated an IP address (a machine can have many IP addresses at the same time). Your invented IP must be formally correct and the net mask must match the class of the network (class A, B or C). See the chapter on IP addresses in the already mentioned Network Administrator Guide (nag) (`/usr/doc/LPD/nag` on your system, or if you don't have it, download it now from http://metalab.unc.edu/mdw/index.html#guide.

For a home network, you might want to invent a class C network (up to 254 machines, the smallest) IP number which has the first three digits between 192 to 223. The last three digits identify the machine on your class C network and must be between 1 and 254 (don't use 0, it means "whole network" or 255 which is the "broadcast address"). The middle two sets of digits can be anything from 0 to 255. Thus 223.223.223.1 is the first machine on the class C network 223.223.223.0 (the last zero signifies the whole network), with the broadcast address 223.223.223.255. The network mask for a class C network is always 255.255.255.0 (unless you subdivide your network into smaller "subnet", which is not discussed here).

Rather than completely inventing an IP number, it might be safer/better to use one of the numbers reserved for "private networks". For me, the IP number 192.168.1.1 works just fine. This way, your "invented" IP address is guaranteed never to interfere with any IP that may exist in the world.

Typically, the first machine on a network is the one that is expected to have the connection to the outside world (since it was connected first, but there is no standard for that). So, I filled up the GATEWAY to xxx.xxx.xxx.1 (my first machine) on all machines, except for the machine xxx.xxx.xxx.1, where I left this field blank. Actually, although I left the field blank, `netconf` inserted the gateway 0.0.0.0 into the `/etc/sysconf` file. This was a source of an annoying error message during the loading of the network card on the bootup. To get rid of the message, I edited `/etc/sysconfig/network` and set it to something like this:

```
GATEWAYDEV=
GATEWAY=
```

[You don't want a gateway on your ethernet interface on this machine if it does not lead to the outside world. The ppp interface on this first machine will be set up as a default gateway once you connect through your modem, e.g. using kppp.]

The name of a computer is entirely arbitrary−−the main user normally chooses a short word s/he likes. The domain name of the home network is also entirely arbitrary, unless you have a permanent connection in which case a domain name is registered to you. Try to invent something that does not exist yet−−it could make your life easier once you have a permanent connection.

As for the DNS (domain name server, also called "named" = name daemon), RedHat 5.2 and 6.0 comes with DNS preinstalled as "caching−only" so it is easy to configure as such. You may also choose not to use local DNS at all−−if your local DNS is looking up an "outside" server and can't find it, it can be a real show−stopper (the machine can appear to be hanged for up to a few minutes). To use local DNS, the "named" service must be enabled−− check this by running `setup`. To set up the caching−only DNS, fill up the appropriate boxes in `netconf`. E.g., I filled "nameserver 1" to my first machine on all computers (I entered the loopback address 127.0.0.1 on the machine xxx.xxx.xxx.1, and the proper xxx.xxx.xxx.1 address on all other machines).

No routes to other networks and hosts were required in my network, since I don't have other local networks. So I left this field blank.

Other than setting up the hardware correctly and filling up the info under `netconf` on each computer, as described above, I did not have to do anything on the standard RedHat to get my network working.

Reboot all computers one by one in any order (this is not necessary, but won't hurt you) and watch the boot messages (if they scroll too fast off screen, use <Shift><PgUp> to scroll up, or use `dmesg` from the command line to view them later). Did your cards configure correctly? Use the command `route` (as root) to see if the eth0 interface is running. Use the `ping` command to test the connections between individual machines. Try to `telnet` your local computer to see it the loopback−only (lo) interface works:

```
telnet name_of_the_machine_you_are_sitting_at
```

After a successful login, you can exit the telnet session by typing

```
exit
```

Finally, try to telnet another computer on your network:

```
telnet name_of_a_remote_machine
```

If this works on all machines, your eth0 network interface is set up.

After setting up your ppp and connecting to your Internet Service Provider (ISP), you will have another network interface (ppp0) and then will be able to telnet any machine in the world.

4.5.3 I have problems configuring my ppp dial out

GUI. If I were you, I would install kde and use `kppp`. kppp is really easy to configure and run. To set up your ppp dial out, run `kppp` (it is under the "K" menu "Internet, or you can run it from the xterm by typing `kppp`), press the "setup" button, create an account, and fill out the information required: telephone number, authentication protocol, and your Internet Service Provider's (ISP) domain name and the Domain Name Server (DNS) number of your ISP. All this information should have been provided to you by your ISP.

Next, I check if a device called /dev/modem exists and points to the right port. If necessary, I create it by symbolically linking it to the device /dev/ttyS1 (as root):

```
ls −l /dev/modem
ln −s /dev/ttyS1 /dev/modem
```

ttyS1 should work if your modem is on the port that DOS calls COM2. Use ttyS0 for COM1 and ttyS2 for COM3 and ttyS3 for COM4. The modem will not dial at all if a wrong port is chosen. For non−standard serial port setups, see the command `setserial`.

With old versions of kppp, you may get an error message complaining about a "lock". The solution then is to make sure that the file `/etc/ppp/options` is empty by editing it (as root):

```
pico /etc/ppp/options
```

and deleting the word "lock", then saving the file. This problem does not exist in the more recent versions of kppp.

RedHat 6.0 required one additional step: setting the "suid" ("substitute user id") so that "kppp" runs with the effective user id of root (because it needs to access hardware directly). Without it, kppp complains that "it was not properly set up" and "can't create lock file". This has to be done as root:

```
cd /usr/bin/
chmod a+s kppp
```

Troubleshooting. If your modem refuses to dial on the port that you are positive is chosen properly, maybe the modem is not set up properly (or maybe it is a "winmodem"? Then throw it away and buy a proper modem).

For example, in one instance, I had to run kppp setup, edit the "modem commands" and input `ATZ1` as the

"initialization string" (instead of the default ATZ for a standard Hayes−compatible modem). Otherwise, this particular modem would not dial.

If your telephone line requires pulse dialing (instead of the default touch−tone dialing), you may need to change the modem dialup command from "ATDT" (="ATtantion Dial Touchtone") to "ATDP" (="ATtantion Dial Pulse"). These commands work with any standard Hayes−compatible modem.

If your modem dials correctly and you are able to connect, but your authentication fails, perhaps your Internet Service Provider (ISP) uses a different authentication protocol. Call them and ask what authentication protocol they use. Or try "pap", "terminal−based" or "chap" (in your kppp setup) until you find the one which works with your ISP.

In one instance, I had a problem with the reliability of establishing a connection (the error would pop up saying something like: "time−out for the pppd startup", and the connection would establish only once every few trials). The problem was solved by changing the "flow control" option (in the kppp "setup" under "device") from "CRTSCTS" to XON/XOFF. (Still CRTSCTS is the recommended flow control method in most cases.)

Random disconnects (after some time of correct connection) can have many causes. (1) They may be caused by "glitchy" drops of "data terminal ready" (DTR) signal or "carrier" signal. e.g., due to a noisy line. Most modems respond to that by hanging up. To change this default behaviour, you may need to add to your "modem initialization string" something like `S10=50` . This sets the duration of DTR loss (in 0.01 s) after which hangup is executed (check your modem manual, "US Robotics" modems may need something like `S25=200`). (2) Call−waiting feature on your phone line may disconnect you when somebody tries to call you. (3) Old phone cables and dirty or corroded phone plugs or sockets are a common source of problems. Check the connections, replace the cables. Run the cables further away from sources of great electrical noise. (4) Too high modem speed for your village long and noisy phone cables. Drop the modem speed (or move to a city). (5) Many ISPs will diconnect you after some period of inactivity (30 min?).

If you keep having problems setting up ppp, you may want to try `minicom` to see if you can get your modem working from there. `minicom` is something like `PROCOMM` for Linux. It should be present on your system if you chose to install it during your RedHat initial setup. Here is a post from a newsgroup comp.os.linux.help which explains how to start ppp manually using `minicom` (edited for space):

From: mark <balthazaar@one.net.au> Subject: Re: pppd problem with kppp
BachuZ wrote:
>>Also, for an experiment, try using minicom to connect to your ISP, start ppp manually ... this can prove buggy scripts. >how would u do that?

Easy!! If your ISP doesn't allow a manual logon then you might be in trouble. Every ISP I've ever used does allow this, so.. 1. Start minicom. 2. dial your ISP. 3. Log in. 4. AFter your ISP starts PPP, quit minicom with ALT−Q (or whatever the sequence is to 'quit without reset'). 5. start pppd, eg:

```
pppd −d −detach /dev/modem  115200 &
```

OK, PPP will be running. Try pinging your ISP or another known IP address. That will test everything is OK. BTW, this is all in the PPP−HOWTO. If you can get PPP running this way, then you have a scripting problem. If PPP doesn't work, you have a PPP configuration problem. Cheers.

Command line. If you would like to start your ppp from the command line, run `netconf` (as root) to configure your first ppp interface (ppp0). The information you must enter is similar to what you entered when setting up kppp (have a look above!): the proper device for the modem port, modem initialization and dialup strings, the telephone number of your Internet Service Provider (ISP), the proper authentication protocol (by entering the login name and password into the right slot). In older versions, the `netconf` utility lacked a place to enter the IP addresses of my ISP DNS server, so I edited the file `/etc/ppp/pap−secrets` (I use PAP authentication protocol) and added the two DNS IP addresses at the end of the setup line which was created by `netconf` so it looked like this:

```
# added by linuxconf
my_login_name ppp0 my_password 111.111.111.111 222.222.222.222
```

When done with the settings, I could start my ppp0 interface using the command (as root, unless I specified in `netconf` that normal users can start the interface):

```
ifup ppp0
```

and shut it down with

```
ifdown ppp0
```

Setting up the command line ppp was not more difficult on my machine than running kppp and the connection is more reliable for me. There is lots of command line scripts to start/stop ppp, but they apparently are not so easy to set up and use, and many newbies seem to have problems with them. Badly misfired ppp connections can be killed without rebooting using (as root):

```
killall pppd
```

If this minimalistic setup of ppp does not work for you, here are some useful links:

Roderick A. Anderson < raanders@altoplanos.net > wrote:

I have a web page on setting up diald to work with RedHat Linux 5.x that works for me every time. It is at http://home.altoplanos.net/~raanders/diald.html

Bill Unruh < unruh@physics.ubc.ca > wrote:

I just wanted to bring your attention to the page http://axion.physics.ubc.ca/ppp−linux.html for detailed instructions for setting up ppp. This is especially for cases in which the remote side uses (perhaps without the ISP even telling you) PAP or CHAP. While kppp is useful, there are a number of situations where it can fail.

4.5.4 How to browse the net from my networked computer without a modem?

Another computer on your network must have a modem (or another Internet connection) though :−). Set up IP masquerading. This way, all requests going from your network to your Internet Service Provider (ISP) appear to have originated from a single computer, and your ISP will let them through.

ON REDHAT 5.2, simple masquerading required just one command (on the computer with the modem):

```
/sbin/ipfwadm −F −p m
```

This sets up masquerading as your default forwarding policy of your IP firewall, and therefore is insecure but probably ok for a home user. (The danger is that if somebody hacked your computer, s/he can use it as an annonymous forwarder to hide his identity. Whatever malicious the hacker does to anybody, you take the blame. The hacker can even set his route to "tunnel" back to your network thus concealing his identity from you.) For more info, please check the file `/usr/doc/HOWTO/mini/IP−Masquerade`. A more secure setup is shown here:

```
ipfwadm  −F −p deny
ipfwadm  −F −a m −S 192.168.1.1/32 −D 0.0.0.0/0
ipfwadm  −F −a m −S 192.168.1.3/32 −D 0.0.0.0/0
ipfwadm  −F −a m −S 192.168.2.0/24 −D 0.0.0.0/0
```

This sets up the default policy to "deny" and explicitly masquerades two machines with IPs 192.168.1.1 and 192.168.1.3. It also masquerades any machine from the network 192.168.2.0. The number /32 stands for point−to−point networking (this means "machine−to−machine"), the option /24 identifies a class C network. The −D 0.0.0.0/0 identifies the default route that the machines to be masqueraded use to go out to the Internet.

ON **REDHAT 6.0** THE NAME OF THE COMMAND is `ipfwadm−wrapper` (instead of `ipfwadm`) and I had to use the second, more secure method (setting up masquerading as the default policy does not seem to work on my system any more). Also, with newer kernels (2.0.34 and later) forwarding is disabled by default and must be turned on using:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Actually, `ipfwadm−wrapper` is a wrapper because it lets me use the old rules of setting up the firewall policies using the brand new firewalling kernel code. Doing something like this may work better for you:

```
ipchains −P forward DENY
ipchains −A forward −s 192.168.1.0/24 −j MASQ
```

If you would like to have this command(s) always executed on your system startup, add it as the last line(s) to the file `/etc/rc.d/rc.local` . This file is something like AUTOEXEC.BAT in DOS. As always, it is recommended to read the manual page and other documentation to see what the command(s) does and what are the other options:

```
less /usr/doc/HOWTO/mini/IP−Masquerade
man ipfwadm
man ipchains
```

ON REDHAT 7.1 (KERNEL 2.4.x) the firewalling can be set up using the new `iptables` command. You can still use the old ipchains, provided you don't run iptables at the same time. So perhaps use ntsysv to make sure ipchains is enabled, and iptables is disabled.

4.5.5 How to use Samba?

Samba (smb) is for Linux−MS Windows networking. It is a program that makes a Linux computer pretend to be a MS Windows NT server, and thus lets your MS Windows 3.1/95/98−based computers connect to the network. Samba not only replaces WinNT−−it is acclaimed to do a much better job than WinNT!

One of the three machines on my home network is a dual Linux/Win95 boot. I configured my network so that if I boot Win95, another Linux machine acts as a Samba server. This way, the Win95 machine has access to the network printer, shared directories on Linux, can telnet, browse the Internet through a modem on Linux, etc. I can also access the files on the remote Win95 machine from Linux computers.

It is necessary to configure Samba only on one Linux machine.

First, I ran `setup` as root, choose "System Services" (RH6.0) or "ntsysv" (RH5.2) from the menu, and make sure that the "smb" service is enabled. (The program `setup` has help if you press F1. Just in case you were curious what the different services are for.) If smb is not listed there, maybe you did not install it during your RedHat setup? You may want to put your RedHat CD into the CDROM, mount the CD, start glint (RH5.2) or gnorpm (RH6.0) installation utility (as root) and add Samba to your system.

The second part is to configure Samba. This is relatively simple since the Samba configuration is done through a single, well commented file: `/etc/smb.config` . The minimum setup includes specifying the workgroup name. Note that if you don't fill up the "host allow" option, all hosts are allowed, which is probably ok for the home network. The other options in the example `/etc/smb.conf` supplied with your RedHat CD may require adjustment to suit your particular needs, but they didn't have to be changed to get a Windows machine connected to my Samba server.

Check that the options

```
domain master = yes
domain logons = yes
```

are enabled in `/etc/smb.conf`. You probably want them.

Under RH6.0, you may also perform a basic samba setup using (as root):

```
netconf
```

The third step is to configure your MS Windows 95 (or whatever) for networking. This is done exactly as if you were connecting to a WinNT server. Make sure to enable the networking ("client for Microsoft Network") and fill up the workgroup name. To login on the network when booting MS Windows, use your Linux user ID and password. To see if it worked, click on the icon "Network Neighborhood"−−your Linux server should be listed there, and underneath you should see the shared directories and printers that you chose to share in the file `/etc/smb.conf`.

If you can't see or use the public directories, make sure that you created them and set the proper read/write permissions for all users.

MS Windows 95b and above (95c, 98, and newer NTs) may use password encryption. This will make your logins from the Window's machine fail, and you may need to enable this option in the `/etc/smb.conf` file:

```
encrypt passwords = yes
```

On the Linux server, you can start, stop, restart and check the Samba status using these commands (as root):

```
samba start
samba status
samba restart
samba stop
```

You need to re−start samba after making changes to your /etc/smb.conf file.

You can browse the net using your Netscape for Windows if you are connected to your Linux computer through Samba and the Linux machine is currently connected to the Internet. To do this, the IP masquerading must be set up on the Linux machine with the modem (described here) and you must enable the Samba dns name resolution in `/etc/smb.conf`:

```
dns proxy = yes
```

and then tell Windows to enable the dns server, specify the Linux server name and IP address (in ControlPanel−Network−TCP/IP).

You can also mount a remote Windows directory onto your Linux filesystem. Look here to see how.

The above described just a minimal Samba setup. You can get more information from:

`man samba; man smb.conf; man smbclient; man smbmount`
`less /usr/doc/HOWTO/SMB−HOWTO` (under RH5.2)
documentation in `/usr/doc/samba−2.0.3/doc` (under RH6.0)
`http://www.sfu.ca/~yzhang/linux/samba/` (samba minihowto)
`http://www.germanynet.de/teilnehmer/101,69082/samba.html`

and also by studying the `file /etc/smb.conf`.

4.5.6 Sendmail

Sendmail is the Internet standard mail−transport system and the default mail−transport on RedHat Linux (Mandrake uses PostFix instead). As a user, you don't use sendmail directly−−sendmail is the underlying server engine that manages the mail on your machine in the background, for all users. To read/send mail in the text mode, you probably want to use pine or elm (choose one and stay with it−−it can be inconvenient to manage two separate mail boxes). In KDE, you may choose to use the "mail client"(kmail) available from the K−menu. To communicate with the outside world through your ppp dial−out, you probably would like to use the mailer that is built into your Netscape and which communicates directly with your remote Internet−Service−Provider−based mailbox (bypassing the mail server facility on your local computer).

Sendmail is very flexible and robust, but also notoriously difficult to manage if you needed to customize it to your specific needs. Luckily, Red Hat (5.2 or 6.x) comes with the sendmail that runs out−of−box (with some limitations though).

On my home system (default Red Hat setup), I can send mail to another user on the same machine with no problem (e.g. using pine). I can send mail locally or anywhere in the world from any mailer once I am connected to my Internet Service Provider through the modem (IP masquerading enabled). But when I am not connected, the mail sits in the queue and waits for the Internet connection, even if the mail is to be sent to another computer on my home network (a minor annoyance). It gets sent once I connect. (It happens because sendmail is trying to do a DNS lookup and this is not available on my system−−RedHat default DNS is cache−only.) If you really want to avoid this on RH6.x, you may use `netconf` (as root) and specify that sendmail is not to use DNS at all (Linuxconf that comes with RH5.2 does not give you the choice). I do use DNS.

Under RedHat 6.x, you may also want to use `netconf` to configure other sendmail options using (as root). I entered the name of my Internet Service Provider and the names of other machines on my home network under "mail deliver system"−"relay to hosts". I also added the names of my home network machines under "relay for by

name".

You should also be aware of the limitation of your simple setup that arises if you send e−mail from your home network (for example using pine), to the world beyond your network. If you invented the IP address and your domain name is unregistered, there is no chance you will receive a reply. Your outgoing mail is given a "reply" address in the form: "user_login_name@machine.domain". This is NOT ok since your domain name does not exist, according to any DNS in the outside world, hence no way for any reply to ever get to you. To overcome this, you may use the Netscape mailer to communicate to the outside of your home network. Netscape uses the settings you enter in its "edit−preferences" to communicate directly to your ISP−based mailbox (which resides on a registered server of your ISP) and thus bypasses your unregistered−home−network−based e−mail system. Another possibility is to specify the correct address in the "reply to" field. If you do it in Netscape, you can setup your NetscapeMail to use your local Linux computer as the mail server, and this way be able to send e−mail from your NetscapeMail also to the computers on your home network (not only in the "outside world"). The option "reply−to" can also be set in the KDE "mail client" setting, if you use the KDE mailer. Also in "pine" you can specify the reply−to address under "setup−configure", "customized−hdrs" with something like:

```
Reply−to:  joe@joe_net.net
```

where "joe@joe_net.net" is your good reply−to e−mail address.

If you do wish to fetch your mail from your Internet provider−based mailbox onto your account without the help of Netscape, you may consider installing `fetchmail` (see `man fetchmail`).

4.5.7 Simple web server (running Apache)

To set up a simple apache web server was extremely easy under RedHat 6.0. First of all, I made sure to install the apache web server rpm package during my RedHat installation. If you didn't do it, you can put the RH CD in your CDROM now, mount it, and install the package apache−*.rpm (instead of the "*" the name also contains the version number and the platform). If I know what I want to install, it is simple do it (as root):

```
cd /mnt/cdrom/RedHat/RPMS
rpm −ivh apache*.rpm
```

Then, I run "setup" (as root) and make sure that the httpd daemon (under "system services") is enabled. [A daemon is a program that sits in the background and wakes up when it is needed. In the case of httpd, it gets awakened when somebody calls on your http server.]

Now, the httpd will start automatically every time I boot the computer. I can also start it manually using (as root):

```
/etc/rc.d/init.d/httpd start
```

and shut it down using:

```
/etc/rc.d/init.d/httpd stop
```

I should already have the directory: `/home/httpd` (check if it exists), and under it, the subdirectory `html`−−this is the "root directory" for people accessing my computer from the web ("their root directory" means that they will not be able to access any directories above the directory `/home/httpd/html/` on your system). On default, this directory contains some html files and manual that the apache installation program put there. So I should now be able to connect to my web server from another machine on my home network. For example I would type on the Netscape "Location" line:

```
http://my_http_server_name
```

and be able to browse the "apache" manual.

To put my own content on my web server, I move the "apache" sample files somewhere else, and copy or link my html files (the ones which I want to display to the public) to the directory `/home/httpd/html/` (don't forget to include the the file `index.htmll` , this is the one that appears first when somebody connects to your server).

The apache configuration files and log files are in the directory `/etc/html/` if you wanted to view/customize them. The log file can be viewed in "real time" using this simple command (as root):

```
tail −f /etc/httpd/logs/access_log
```

[The `tail` command normally displays the end ("tail") of a text file. With the option "−f", "tail" keeps displaying the end of the log file as it grows−−really handy to monitor the log file and see who logs onto your server.]

For graphical setup of the Apache server, try this (in an X−window, as root, if you installed "Comanche" from your RedHat CD):

```
comanche
```

4.5.8 Simple ftp server

With older my RedHat Linux distribution (RH<7.1), setting up an ftp server could not be simpler−−it just works out−of−box. This is because the ftp service is enabled on default as one of the standard services (as is telnet and gopher), in the `file /etc/inetd.conf`. Here is the relevant part of my `/etc/inted.conf`:

```
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd −l −a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
gopher  stream  tcp     nowait  root    /usr/sbin/tcpd  gn
```

The second part of my ftp server setup is in the file `/etc/passwd` which defines the ftp account:

```
ftp:*:14:50:FTP User:/home/ftp:
```

The ":" is a field separator. The first field is the account (user) name "ftp", the "*" in the second field indicates that the password is disabled (nobody can login under the "ftp" user name), the user id is 14, group id is 50, "FTP User" is a comment, the home directory is `/home/ftp`, the last field is empty (for "normal" user accounts, it specifies the name of the shell for the user).

Because this setup was already done for me by RedHat, anybody can ftp my computer and either login as a user (will be prompted for password and directed to his/her home directory), or login as "anonymous" and give his/her e−mail address as a password. Any user can also enter something like this on the Netscape "location" line:

```
ftp://my_computer_name
```

and connect automatically (Netscape will take care of sending the "anonymous" user name and the e−mail address as password).

The "anonymous" ftp users are directed to the directory `/home/ftp` , which appears to be a root directory to them (they cannot access any directory above it). I put the files I want to serve in the subdirectory `/home/ftp/pub` .

The directory /home/ftp/bin contains the commands that the remote users are able to execute. On my system these are: `compress`, `cpio`, `gzip`, `ls`, `sh`, `tar`, `zcat`; all with execute−only (111) permissions.

The directory `/home/ftp/etc` contains the setup files necessary for the anonymous account to function (edited `passwd`, `group`, `ld.so.cache`). The directory `/home/ftp/lib` contains the libraries (I guess these libraries are used by the commands that the anonymous ftp users are allowed to run).

Red Hat 7.1 uses `xinetd` in place of the older inetd, and most of the network services are disabled on default. If you cannot telnet to yourself or another network service you need is disabled, you may want to inspect the files in the directory `/etc/xinetd.d`, and edit the file with the name of the service, so that it contains: disable = no . This is for for security reasons−−you have to choose the services you need and enbable just those. Don't enable ftp unless you require it−−ftp used to have quite a few security glitches in the past. If you enable network services, make sure you conservatively setup the files /etc/hosts.allow and /etc/host.deny for security.

Example file `/etc/xinetd.d/tpfp` showing the service disabled:

```
service tftp
{
        disable = yes
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
```

```
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot
}
```

If you don't have this file, do `cat` to see what services you installed. The daemon for most services will start automatically on system startup if this startup is enabled using command `setup` (as root).

4.5.9 How can one access my computer from the outside world when I am on the net using phone connection?

The only difficulty is that your IP address is dynamically allocated to you by your Internet Service Provider (ISP) from their IP address pool, and therefore the IP address is not the same every time you connect (unless you made specific arrangements with your ISP). To telnet, ftp, or access your web pages (served by your appache web server) from the outside world, one has to know your current IP address. To find out my current IP address, I use this "interface configuration" command which, when run without any parameters, just displays info on all active network interfaces present on your machine:

```
/sbin/ifconfig
```

On my machine this displays three paragraphs of information on: eth0 (the first ethernet network interface that leads to other computers on my home network), lo (the loopback-only interface, the one with IP 127.0.0.1, this one must be present on every machine), and ppp0 (the first point-to-point protocol interface). My current IP address, assigned to me by my ISP, is displayed under the ppp0 heading. (Your Linux machine can have multiple IP addresses assigned at the same time, so if you have a "static" IP that you use on your home network, it is still valid but visible only on your home network.)

Once I know the IP address, I can send it through ICQ or e-mail to a friend, who can then, for example, telnet or ftp my computer (s/he must have an account on my machine) and run a program on my linux machine, or enter http://my_ip_address on the "location" line in the browser to browse my home web pages, etc. If the friend has Xwindows on his/her local machine, s/he can even run a GUI program on a my server and direct the display on his computer.

I can also write a short script that will automatically notify my friend when I am connecting to the Internet and enter the name of the script under kppp-setup-account-edit-dial-"execute program upon connect". Here is my script which notifies me at work when somebody in my house is going on-line (I entered the text into a text file and made the file executable using `chmod o+x file_name`):

```
#!/bin/bash
sleep 15
/sbin/ifconfig | mail -s notification my_email_address
```

The first line of this script tells my computer to interpret this text file as a bash shell script. The second line makes the script wait 15 seconds (just to make sure that the e-mail is not sent before the ppp connection is fully established). The third line executes the ifconfig command and pipes the output to the mail utility that sends it to my_email_address under the subject "notification".

A more flexible way to access your home computer remotely is to configure it as a dial-up ppp server (as opposed to the dial-in client that you use when you connect to your ISP). If somebody has a simple recipe how to do it, please drop a line.

To summarize, unlike MS Windows 3.x/95/98 which severely restricts traffic to your computer, Linux is very network oriented and it is easy to make all kinds of network connections both FROM and TO your Linux computer. The powerful networking features are generally considered a Linux a strength but, from a real newbie point of view, they can be a problem (see the next question).

4.5.10 Can my home computer get hacked?

Unfortunately, this is perfectly possible and attempts to do so are quite common. Every time you are connected to your Internet Service Provider (ISP) you are at risk. Read the previous answer if you would like to know how it is possible. Obviously, the risk is much higher if you have a permanent Internet connection (e.g. cable modem), and it is lower if your connections are more transient (as typical with short-duration, over-the phone modem connections).

The real danger is that the intruder, if s/he is able to login onto your machine on any account, may find (may know of) a "local security exploit" and get root access. This is particularly possible if you are a real newbie administrator and/or your machine is not really security oriented (you are at home, aren't you−−who would care about security, you think!).

To protect yourself, just never let a stranger log onto your computer. Have fairly long passwords that contain both numbers and letters for all accounts on your computer. Change the passwords occasionally. The best way to enforce the password policies on all users of your computer is to run (as root, available on RH6.0) `linuxconf` and under "password and account policies" change the minimum password length to 6 or more characters, the minimum number of non−alpha characters (i.e., not−letters) to 1 or 2, the number of days after which the password must be changed to something like 90 or less, and set a warning about password expiry to something like 7 days before the expiration. Check here (FAQ2.htm#pass_security) for more info on weak passwords. Absolutely NEVER create an account with no password, or with a silly weak password. Do not habitually work on your computer as "root"−−if you run a program with a known "security hole" as root, somebody may find a way to hack you. Older Linux distribution have known security holes, so use an updated version if you let untrusted people log onto your computer, or if you run "server side" network services (e.g., ftp or http server).

It is also an excellent idea to occasionally screen the files that contain a record of all the logins onto your computer: `/var/log/secure` (the most recent log) `/var/log/secure.1` (older log) `/var/log/secure.2` (yet older log), etc. There are also other useful log files in the directory `/var/log` that you might want to view, check them out from time to time. The most typical "warning" sign is a scanning of the ports on your computer: there are repeated entries on connection request from the same IP number to your system telnet, ftp, finger and other ports−−somebody tried to learn more about your system.

If you never use remote connections to your home Linux machine, it is an excellent idea to restrict the rights to use the "server side" network services (all the network services are listed in the file `/etc/inetd.conf`) to the machines on your home network. The access is controlled by two files:`/etc/hosts.allow` and `/etc/hosts.deny` . These access−control files work as follows. When an outside connection is requested, the file `/etc/host.allow` is scanned first and if the name of the machine from which the connection is requested is matched, the access is granted (irrespectively of any entry in `/etc/host.deny`). Otherwise, the file `/etc/host.deny` is scanned, and if the name of the machine from which the connection is requested is matched, the connection is closed. If no matches are found in either file, the permission is granted.

B. Staehle (a Linux modem guru) wrote to me to advice not to install network services at all. "If your network services are not configured properly, you may wind up with your computer owned by some script kiddie. A newbie should _never_ be allowing services (ftp, telnet, www) to the world. If you "must" install these, make sure to only permit connections from systems you control. The file `/etc/hosts.deny` should contain

```
ALL: ALL
```

and `/etc/hosts.allow` should only have

```
ALL: 127.0.0.1
```

to permit connections only from that named host. Do NOT use hostnames! " <end of Bill advice>.

Indeed, my `/etc/host.deny` is exactly as adviced above (ALL: ALL), but my `/etc/hosts.allow` two extra trusted computers to connect to all my network services, and another computer to access telent and ftp: (the IP numbers are fake):

```
ALL:  127.0.0.1, 100.200.0.255, 100.200.69.1
in.telnetd, in.ftpd: 100.200.0.2
```

In the examples above "ALL: ALL" stands for "ALL services, ALL Hosts", meaning "connections to any local network service" coming from "any host".

For more info, check the excellent "Linux Network Administrator Guide" which is surely present on your RedHat (or whatever) distribution CD. I printed this book and had it hardcovered.

To verify which services your computer offers to the outside world, you may want to use a web−based tool. Go to: http://scan.sygatetech.com/ and click on "scan now".

> Here are some other places that may be able to scan you: http://crypto.yashy.com/
> http://davidovv2.homestead.com/freetoolsservices.html http://privacy.net/ http://scan.sygatetech.com/

http://security1.norton.com/us/intro.asp http://suicide.netfarmers.net/
http://trojanscanner.com/cgi−bin/nph−portscanner http://www.doshelp.com/dostest.htm
http://www.dslreports.com/secureme/ http://www.dslreports.com http://www.earthlink.net/freescan/
http://www.grc.com http://www.hackerwhacker.com/ http://www.nessus.org
http://www.netcop.com/newscan/fullscan.html http://www.privacyscan.org
http://www.sdesign.com/cgi−bin/fwtest.cgi http://www.sdesign.com/securitytest/index.htmll
http://www.securityspace.com/ http://www.vulnerabilities.org/nmapemail.html http://grc.com
http://www.dslreports.com/scan http://www.dslreports.com/security/sec025.htm

For security reasons, it is also a good idea not to advertise the OS/version that you use. I replaced the contents of the file `/etc/issue` and `/etc/issue.net` which on my computer read:

```
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.14−5.0 on an i586
```

with something like this:

```
WARNING: THIS IS A PRIVATE NETWORK
UNAUTHORIZED USE IS PROHIBITED AND ALL ACTIVITIES ARE LOGGED
IBM S/390 LINUX
```

This blends a joke with a little bit more security (I hope).

The contents of the files `/etc/issue` and `/etc/issue.net` are recreated at every reboot (when the script `/etc/rc.local` is run). So, to make the changes permanent, I can make these files read−only for all users (as root):

```
chmod a=r /etc/issue*
```

Instead of the last command, I could have edited (as root), the script `/etc/rc.d/rc.local` and commented out 5 lines with ### so that the relevant part reads:

```
# This will overwrite /etc/issue at every boot. So, make any changes
# want to make to /etc/issue here or you will lose them when you reboot
### echo "" > /etc/issue
### echo "$R" >> /etc/issue
### echo "Kernel $(uname −r) on $a $SMP$(uname −m)" >> /etc/issue
### cp −f /etc/issue /etc/issue.net
### echo >> /etc/issue
```

Another good security measure is to disable ping. Ping is a sonar−like response that your computer sends back when inquired by another computer. It is mostly useful for setup and debugging, to probe whether your machine is available on the network. It can also be used for probing your machine and/or attacking it by flooding with ping requests ("ping of death"). To disable my machine response to pingging from the net, I use the IP masquarading. I took and slightly modified the following command and explanation from http://www.securityfocus.com/focus/linux/articles/linux−securing2.html:

```
ipchains −A input −p icmp −−icmp−type echo−request −i ppp0 −j REJECT −l
         (1) (2)    (3)           (4)                 (5)     (6)    (7)
```

The ipchains flags explained:

1. (A)ppend a new rule.
2. The chain to apply the rule to, in this case the rule will apply to ingress (input) packets.
3. (P)rotocol to apply the rule to. In this case, it is icmp.
4. ICMP type, in this case all icmp echo requests will be blocked. "ICMP echo" means ping.
5. Interface name. In this case, it is the first over−the−phone connection, ppp0.
6. Target, or what should actually be done with the packet in question.
7. Log all packets matching the rules criteria to system log file.

IP masquarading was described in more detail in the chapter on masquerading of this Guide.

Another security precaution I take. I occassionally check if somebody hasn't installed a "root kit" on my system. I use the utility "chkrootkit" (very small, 25k, download from http://www.chkrootkit.org/). After downloading the tarball do:

```
su  [provide password]
cd /usr/local
```

```
tar xvzf /home/my_name/chkrootkit.tar.gz
cd /usr/local/chkro<tab>
make
./chrootkit
```

The last command actually runs the search for a rootkit on my system. "Rootkit" is a software hidden backdoor that somebody who gained once access to your system (as "root") could install in order to listen, monitor, protect her access, etc.

Go to Part: 5 − Linux Shortcuts and Commands

Part 5: Linux Shortcuts and Commands

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section:

Intro. This is a practical selection of the commands we use most often, find useful, and which came on our Linux distribution CDs (RedHat or Mandrake). Press `<Tab>` on the empty command line to see the listing of all available commands (on your PATH). On my small home system, it says there are 3786 executables on my PATH. Many of these "commands" can be accessed from your favourite GUI front−end (probably KDE or Gnome) by clicking on the right menu or button. They can all be run from the command line (unless you didn't install the package, but they all came on our CDs). Programs that require GUI have to be run from under the GUI, for example from a terminal opened in kde or gnome (e.g., `konsole` or `xterm`). Some more advanced (less useful for a newbie?) tools are described in the Part Learning with Linux of this Guide.

Notes for the UNIX Clueless:

1. LINUX IS CASE−SENSITIVE. For example: `Mozilla`, `MOZILLA`, `mOzilla`, and `mozilla` would be four different commands (but of the four, only `mozilla` is available on my system). Also `my_filE`, `my_file`, and `my_FILE` are three different files. Your user login name and password are also case sensitive. (This goes with the tradition of UNIX and the "c" programming language being case sensitive.)
2. Filenames can be up to 256 characters long and can contain letters, numbers, "." (dots), "_" (underscores), "−" (dashes), plus some other non−recommended characters.
3. Files with names starting with "." are normally not shown by the `ls` (list) or `dir` command. Think of these "dot" files as "hidden". Use `ls −a` (list with the option "all") to see these files.
4. "/" is an equivalent to DOS "\" (root directory, meaning the parent of all other directories, or a separator between a directory name and a subdirectory or filename). For example, try `cd /usr/doc`
5. Under Linux, all directories appear under a single directory tree (there are no DOS−style drive letters). This means directories and files from all physical devices are merged into this single−view tree.
6. In a configuration file, a line starting with # is a comment. When changing a configuration file, don't delete old settings−−comment out the original lines with #. Always insert a short comment describing what you have done (for your own benefit!).
7. Linux is inherently multi−user. Your personal settings (and all other personal files) are in your home directory which is `/home/your_user_login_name`. Many settings are kept in files with names starting with a dot "."so as to keep them out of your way (see point 3 above).
8. System−wide settings are kept in the directory `/etc` .

9. Under Linux, as in any multiuser operating system, directories and files have an owner and set of permissions. You will typically be allowed to write only to your home directory which is `/home/your_user_login_name`. Learn to use the file permissions else you will be constantly annoyed with Linux.
10. Command options are introduced by a dash, "−", followed by a single letter (or −− when the option is more than one letter). Thus "−" is an equivalent of DOS's switch "/". For example, try `rm --help.`
11. Type `command&` (the command name followed by an &) to start a command in the background. This is usually the preferred way of starting a program from the X−window terminal.

5.1 Linux essential keyboard shortcuts and sanity commands

`<Ctrl><Alt><F1>`
Switch to the first text terminals. Under Linux you can have several (6 in standard setup) terminals opened at the same time. This is a keyboard shortcut, which means: "press the control key and the alt key, hold them. Now press <F1>. Release all keys."

`<Ctrl><Alt><Fn>` (n=1..6)
Switch to the nth text terminal. (The same could be accomplished with the rarely used command `chvt n`. "chvt" stands for "change virtual terminal"). In text terminal (outside X), you can also use `<Alt><Fn>` (the key `<Ctrl>` is not needed).

`tty`
Print the name of the terminal in which you are typing this command. If you prefer the number of the active terminal (instead of its name), it can be printed using the command `fgconsole` (="foreground console").

`<Ctrl><Alt><F7>`
Switch to the first GUI terminal (if X−windows is running on the 7th terminal, where it typcially is).

`<Ctrl><Alt><Fn>` (n=7..12)
Switch to the nth GUI terminal (if a GUI terminal is running on screen n−1). On default, the first X server is running on terminal 7. On default, nothing is running on terminals 8 to 12−−you can start subsequent X server there.

`<Tab>`
(In a text or X terminal) Autocomplete the command if there is only one option, or else show all the available options. On newer systems you may need to press `<Tab><Tab>`. THIS SHORTCUT IS GREAT, it can truely save you lots of time.

`<ArrowUp>`
(In a text or X terminal) Scroll and edit the command history. Press `<Enter>` to execute a historical command (to save on typing). `<ArrowDown>` scrolls back.

`<Shift><PgUp>`
Scroll terminal output up. This works also at the login prompt, so you can scroll through your bootup messages. The amount/usage of your video memory determines how far back you can scroll the display. `<Shift><PgDown>` scrolls the terminal output down.

`<Ctrl><Alt><+>`
(in X−windows) Change to the next X−server resolution (if you set up the X−server to more than one resolution). For multiple resolutions on my standard SVGA card/monitor, I have the following line in the file `/etc/X11/XF86Config` (the first resolution starts on default, the largest resolution determines the size of the "virtual screen"):
`Modes "1024x768" "800x600" "640x480" "512x384" "480x300" "400x300" "1152x864"Z`
Of course, first I had to configure the X server, either by using Xconfigurator, xf86config, or manually by edition the file `/etc/X11/XF86Config`, so that it supports the above resolutions (mostly the matter of uncommenting the line that defines my video chipset, and specifying the synchronization frequencies my monitor supports). `XFdrake` (Mandrake configuration utility) can do it from GUI. See also the commands `xvidtune` and `xvidgen`.

`<Ctrl><Alt><−>`
(in X−windows) Change to the previous X−server resolution.

`<Ctrl><Alt><Esc>`
(in X−windows, KDE) Kill the window I am going to click with my mouse pointer (the pointer changes to something like a death symbol). Similar result can be obtained with the command xkill (typed in X−terminal). Useful when an X−window program does not want to close (hangs?).

`<Ctrl><Alt><BkSpc>`
(in X−windows) Kill the current X−windows server. Use if the X−windows server cannot be exited normally.

`<Ctrl><Alt><Del>`
(in text terminal) Shut down the system and reboot. This is the normal shutdown command for a user at the text−mode console. Don't just press the "reset" button for shutdown!

`<Ctrl>c`
Kill the current process (works mostly with small text−mode applications).

`<Ctrl>d`
(pressed at the beginning of an empty line) Log out from the current terminal. See also the next command.

`<Ctrl>d`
Send [End−of−File] to the current process. Don't press it twice else you also log out (see the previous command).

`<Ctrl>s`
Stop the transfer to the terminal.

`<Ctrl>q`
Resume the transfer to the terminal. Try if your terminal mysteriously stops responding. See the previous command.

`<Ctrl>z`
Send the current process to the background.

`exit`
Logout. I can also use `logout` for the same effect. (If you have started a second shell, e.g., using `bash`, this command will make you exit the second shell, and you will be back in the first shell, not logged out. Then use another `exit` to logout.)

`reset`
Restore a screwed−up terminal (a terminal showing funny characters) to default setting. Use if you tried to "cat" a binary file. You may not be able to see the command as you type it, but it still will work.

`<MiddleMouseButton>`
Paste the text which is currently highlighted somewhere else. This is the normal "copy−paste" operation in Linux. It a fast and powerful supplement to the widely−known GUI "copy−paste" menu−based operation. (It doesn't work inside older versions of Netscape which use the Mac/MS Windows−style "copy−paste" exclusively. It does work in the text terminal if you enabled "gpm" service using "setup". It also works inside any dialog boxes, etc.−−really convenient!) It is best used with a Linux−ready 3−button mouse (Logitech or similar) or else set "3−mouse button emulation". The <MiddleMouseButton> is normally emulated on a 2−button mouse by pressing both mouse buttons simultanously.

`~`
(tilde character) My home directory (normally the directory `/home/my_login_name`). For example, the command `cd ~/my_dir` will change my working directory to the subdirectory "*my_dir*" under my home directory. Typing just "cd" alone is an equivalent of the command "cd ~". I keep all my files in my home directory.

`.`
(dot) Current directory. For example, `./my_program` will attempt to execute the file "my_program" located in your current working directory.

`..`
(two dots) Directory parent to the current one. For example, the command `cd ..` will change my current working directory one one level up.

Some additional KDE keyboard shortcuts (useful, but non−essential)
`<Alt><Tab>` Walk through windows. To walk backwards: `<Alt><Shift><Tab>`
`<Ctrl><Tab>` Walk through desktops. To walks backwards: `<Ctrl><Shift><Tab>`
`<Ctrl><Esc>` Show the table of processes running on my system. Allow me to kill any of the processes I started (or send other signals to them).
`<Alt><F1>` Access the K−menu ("Equivalent to MS Windows "Start" menu).
`<Alt><F12>` Emulate the mouse using the arrow keys on the keyboard.
`<Alt><LeftMouseButton>` Drag a window to move it. Normally, I move a window by dragging its top title bar, but occassionally I manage to get it off the screen. With this shortcut, I can drag by any part of the window.
`<Alt><PrintScreen>` Take a snapshot of the current window into the clipboard.
`<Ctrl><Alt><PrintScreen>` Take a snapshot of the entire desktop into the clipboard.
`<Ctrl><Alt><l>` Lock the desktop.
`<Ctrl><Alt><d>` Toggle hide/show the desktop (great to hide the Solitaire game when your boss walks in).

`<Alt><SysRq><command_key>`
(Non−essential.) This is a group of key combinations implemented at the Linux kernel level (a low level). It means, chances are these key combinations will work most of the time. The combinations are meant for debugging purposes and in an emergency (mostly developers); you should try other, safer solutions first. The key <SysRq> is also knows on PC as <PrintScreen>. The combinations can be enabled/disabled by setting the relevant kernel variable to "1" or "0", e.g. : `echo "1" > /proc/sys/kernel/sysrq`
`<Alt><SysRq><k>` Kill all processes (including X) which are running on the currently active virtual console. This key combination is know as "secure access key" (SAK).
`<Alt><SysRq><e>` Send the TERM signal to all running processes except init, asking them to exit.
`<Alt><SysRq><i>` Send the KILL signal to all running processes except init. This may be more successful in killing runaway processes than the previous key combination, but it may cause some of them to exit abnormally.
`<Alt><SysRq><l>` Send the KILL signal to all processes, including init. The system will not be functional.
`<Alt><SysRq><s>` Run an emergency sync (cache write) on all mounted filesystems. This can prevent data loss.
`<Alt><SysRq><u>` Remount all mounted filesystems as read−only. This has the same effect as the sync combination above, but with one important benefit: if the operation is successful, fsck won't have to check all filesystems after a computer hardware reset.
`<Alt><SysRq><r>` Turn off keyboard raw mode. This can be useful when your X session hangs. After issueing this command you may be able to use <CTRL><ALT><DEL>.
`<Alt><SysRq><b>` Reboot immediately without syncing or unmounting your disks. Your will likely end up with filesystem errors.
`<Alt><SysRq><o>` Shut the system off (if configured and supported).
`<Alt><SysRq><p>` Dump the current registers and flags to your console.
`<Alt><SysRq><t>` Dump a list of current tasks and their information to your console.
`<Alt><SysRq><m>` Dump memory info to your console.
`<Alt>SysRq><digit>` The digit is '0' to '9'. Set the console log level, controlling which kernel messages will be printed to your console. For example, '0' will cause only emergency messages like PANICs or OOPSes displayed on your console.
`<Alt><SysRq><h>` Display help. Also, any other unsupported <Alt><SysRq><key> combination will display the same help.

5.2 Help commands

`any_command` `−−help |more`
Display a brief help on a command (works with most commands). For example, try `cp −−help |more`. "−−help" works similar to DOS "/h" switch. The "more" pipe is needed when the output is longer than one screen.

`man` *`topic`*
Display the contents of the system manual pages (help) on the topic. Press "q" to quit the viewer. Try `man man` if you need any advanced options. The command `info` *`topic`* works similar to `man` *`topic`*, yet it may contain more up−to−date information. Manual pages can be hard to read−−they were written for UNIX programmers. Try *`any_command`* `−−help` for a brief, easier to digest help on a command. Some programs also come with README or other info files−−have a look to the directory `/usr/share/doc`. To display manual page from a specific section, I may use something like: `man 3 exit` (this displays an info on the command `exit` from section 3 of the manual pages) or `man −a exit` (this displays man pages for `exit` from all sections). The `man` sections are: Section 1−User Commands, Section 2−System Calls, Section 3−Subroutines, Section 4−Devices, Section 5−File Formats, Section 6−Games, Section 7−Miscellaneous, Section 8−System Administration, Section 9, Section n−New. To print a manual page, I use: `man topic | col −b | lpr` (the option col −b removes any backspace or other characters that could make the printed manpage difficult to read).

`info` *`topic`*
Display the contents of the info on a particular command. `info` is a replacement for `man` pages so it contains the most recent updates to the system documentation. Use <Space> and <BkSpace> to move around or you may get confused. Press "q" to quit. A replacement for the somewhat confusing info browsing system might be `pinfo` − try if you like it any better.

`apropos` *`topic`*
Give me the list of the commands that have something to do with my topic.

`whatis` *`topic`*
Give me a short list of commands matching my topic. `whatis` is similar to `apropos` (see the command above)−−they both use the same database. But `whatis` searches keywords, while `apropos` also searches the descriptions of the keywords.

`help` *`command`*
Display brief info on a bash (shell) built−in command. Using `help` with no *command* prints the list of all bash built−in commands. The shortest list of bash built−in commands would probably include: `alias, bg, cd, echo, exit, export, fg, help, history, jobs, kill, logout, pwd, set, source, ulimit, umask, unalias, unset.`

`kdehelp`
`kdehelpcenter`
(in X−terminal, two commands, use the one that works on your system). Browse the whole system help using the graphical KDE help navigator. Normally, KDE help is invoked by pressing the appropriate icon on the KDE control panel. Use `gnome−help−browser` for the GNOME equivalent.

5.3 System info

`pwd`
Print working directory, i.e., display the name of my current directory on the screen.

`hostname`
Print the name of the local host (the machine on which I am working). Use `netconf` (as root) to change the name of the machine.

`whoami`
Print my login name.

`id` *`username`*
Print user id (uid) and his/her group id (gid), effective id (if different than the real id) and the supplementary groups.

`date`
Print the operating system current date, time and timezone. For an ISO standard format, I have to use: `date −Iseconds`
I can change the date and time to 2000−12−31 23:57 using this command: `date 123123572000`
or using these two commands (easier to remember):
`date −−set 2000−12−31`
`date −−set 23:57:00`
To set the hardware (BIOS) clock from the system (Linux) clock, I can use the command (as root): `setclock`

> The international (ISO 8601) standard format for all−numeric date/time has the form: 2001−01−31 (as in Linux default "C" localization). You can be more precise if you wish using, for example: 2001−01−31 23:59:59.999−05:00 (representing I milisecond before February 2001, in a timezone which is 5 hours behind the Universal Coordinated Time (UTC)) . The most "kosher" representation of the same point in time could be: 20010131T235959,999−0500. See the standard at ftp://ftp.qsl.net/pub/g1smd/8601v03.pdf.

`time`
Determine the amount of time that it takes for a process to complete + other process accounting. Don't confuse it with the `date` command (see previous entry). E.g. I can find out how long it takes to display a directory content using: `time ls`. Or I can test the time function with `time sleep 10` (time the commands the does nothing for 10 seconds).

`clock`
`hwclock`
(two commands, use either). Obtain date/time from the computer hardware (real time, battery−powered) clock. You can also use one of this commands to set the hardware clock, but `setclock` may be simplier (see 2 commands above). Example: `hwclock −−systohc −−utc` sets the hardware clock (in UTC) from the system clock.

`who`
Determine the users logged on the machine.

`w`
Determine who is logged on the system, find out what they are doing, their processor ussage, etc. Handy security command.

`rwho −a`
(=remote who) Determine users logged on other computers on your network. The rwho service must be enabled for this command to run. If it isn't, run `setup` (RedHat specific) as root to enable "rwho".

`finger` *`user_name`*
System info about a user. Try: `finger root` . One can use `finger` with any networked computer that exposes the finger service to the world, e.g., I can do (try): `finger @finger.kernel.org`

`last`
Show listing of users last logged−in on your system. Really good idea to check it from time to time as a security measure on your system.

`lastb`
("=last bad") Show the last bad (unsuccessful) login attempts on my system. It did not work on my system, so got it started with:

```
touch /var/log/btmp
```

> "There's a good reason why /var/log/btmp isn't available on any sane set−up − it's a world−readable file containing login mistakes. Since one of the most common login mistakes is to type the password instead of the username, /var/log/btmp is a gift to crackers." (Thanks to Bruce Richardson). It appears the problem can be solved by changing the file permissions so only root can use "lastb":
> `chmod o−r /var/log/btmp`

`history | more`
Show the last (1000 or so) commands executed from the command line on the current account. The "| more" causes the display to stop after each screenful. To see what another user was doing on your system, login as "root" and inspect his/her "history". The history is kept in the file `.bash_history` in the user home directory (so yes, it can be modified or erased).

`uptime`
Show the amount of time since the last reboot.

`ps`
(="print status" or "process status") List the processes currently run by the current user.

`ps axu | more`
List all the processes currently running, even those without the controlling terminal, together with the name of the user that owns each process.

`top`
Keep listing the currently running processes on my computer, sorted by cpu usage (top processes first). Press `<Ctrl>c` when done.

> PID = process identification.
> USER=name of the user who owns (started?) the process.
> PRI=priority of the process (the higher the number, the lower the priority, normal 0, highest priority is −20, lowest 20.
> NI=niceness level (i.e., if the process tries to be nice by adjusting the priority by the number given). The higher the number, the higher the niceness of the process (i.e., its priority is lower).
> SIZE=kilobytes of code+data+stack taken by the process in memory.
> RSS=kilobytes of physical (silicon) memory taken.
> SHARE=kilobytes of memory shared with other processes.
> STAT=state of the process: S−sleeping, R−running, T−stopped or traced, D−uniterruptable sleep, Z=zombie.
> %CPU=share of the CPU usage (since last screen update).
> %MEM=share of physical memory.
> TIME=total CPU time used by the process (since it was started).
> COMMAND=command line used to start the task (careful with passwords, etc., on command line, all permitted to run "top" may see them!

`gtop`
`ktop`
(in X terminal) Two GUI choices for top. My favourite is `gtop` (comes with gnome). In KDE, ktop is also available from the "K"menu under "System"−"Task Manager".

`uname −a`
(= "Unix name" with option "all") Info on your (local) server. I can also use `guname` (in X−window terminal) to display the info more nicely.

`XFree86 −version`
Show me the version of X windows I have on my system.

`cat /etc/issue`
Check what distribution you are using. You can put your own message in this text file−−it's displayed on login. It is more common to put your site−specific login message to the file `/etc/motd` ("motd"="message of the day").

`free`
Memory info (in kilobytes). "Shared" memory is the memory that can be shared between processes (e.g., executable code is "shared"). "Buffered" and "cashed" memory is the part that keeps parts of recently accessed files−−it can be shrunk if more memory is needed by processes.

`df −h`
(=disk free) Print disk info about all the filesystems (in human−readable form).

`du / −bh | more`
(=disk usage) Print detailed disk usage for each subdirectory starting at the "/" (root) directory (in human legible form).

`cat /proc/cpuinfo`
Cpu info−−it shows the content of the file `cpuinfo`. Note that the files in the `/proc` directory are not real files−−they are hooks to look at information available to the kernel.

`cat /proc/interrupts`
List the interrupts in use. May need to find out before setting up new hardware.

`cat /proc/version`
Linux version and other info.

`cat /proc/filesystems`
Show the types of filesystems currently in use.

`cat /etc/printcap |more`
Show the setup of printers.

`lsmod`
(= "list modules". As root. Use `/sbin/lsmod` to execute this command when you are a non−root user.) Show the kernel modules currently loaded.

`set|more`
Show the current user environment (in full). Normally too much to bother.

`echo $PATH`
Show the content of the environment variable "PATH". This command can be used to show other environment variables as well. Use `set` to see the full environment (see the previous command).

`dmesg | less`
Print kernel messages (the content of the so−called kernel ring buffer). Press "q" to quit "less". Use `less /var/log/dmesg` to see what "dmesg" dumped into this file right after the last system bootup.

`chage −l my_login_name`
See my password expiry information.

`quota`
See my disk quota (the limits of disk usage).

`sysctl −a |more`
Display all the configurable Linux kernel parameters.

`runlevel`
Print the previous and current runlevel. The output "N5" means: "no previous runlevel" and "5 is the current runlevel". To change the runlevel, use "init", e.g., `init 1` switches the system to a single user mode.

> Runlevel is the mode of operation of Linux. Runlevel can be switched "on the fly" using the command `init`. For example, `init 3` (as root) will switch me to runlevel 3. The following runlevels are standard:
>
> 0 − halt (Do NOT set initdefault to this)
> 1 − Single user mode
> 2 − Multiuser, without NFS (The same as 3, if you do not have networking)
> 3 − Full multiuser mode
> 4 − unused
> 5 − X11
> 6 − reboot (Do NOT set initdefault to this)
>
> The system default runlevel is set in the file: `/etc/inittab` .

`sar`
View information extracted the system activity log file (/var/log/sarxx where xx is the current day number). `sar` can extract many kinds of system statistics including CPU load averages, i/o statistics, and network trafic statistics for the current day and (usually) several days backs.

5.4 Basic operations

`ls`
`dir`
List the contents of the current directory. The command `dir` is an alias to `ls` so these two commands do exactly the same thing. The file listing is normally color−coded: dark blue= directories, light grey = regular files, green = executable files, magenta = graphics files, red = compressed (zipped) files, light blue = symbolic links, yellow = device files, brown = FIFO ("First−In First−Out" named pipes).

`ls −al |more`
List the content of the current directory, all files (also those starting with a dot), and in a long form. Pipe the output through the "more" command, so that the display pauses after each screenful. The `ls` command has several very useful options. Some of these may have shortcuts (aliases) to avoid clumsy typing. Try `ll` (="long ls", an alias to `ls −l`). Another option I use quite often is `ls −ad` (list all the subdirectories in my current directory, but don't list their contents).

`cd` *`directory`*
Change directory. Using "cd" without the directory name will take you to your home directory. "`cd −` " will take you to your previous directory and is a convenient way to toggle between two directories. "`cd ..`" will take me one directory up (very useful).

`./program_name`
Run an executable in the current directory. The ./ is needed when the executable is not on my PATH. An executable which is on my PATH is simply run using: *`program_name`*

`shutdown −h now`
(as root) Shut down the system to a halt. Mostly used for a remote shutdown. Use `<Ctrl><Alt><Del>` for a shutdown at the console (which can be done by any user).

`halt`
`reboot`
`init 6`
(as root, three commands) Halt or reboot the machine. Used for remote shutdown, simpler to type than the previous command. Also great if the computer "hangs" (I lose control over the keyboard)−−I telnet to it from another machine on the network and remotely reboot it. I use `<Ctrl><Alt><Del>` for normal shutdown at the console of a local computer.

`vlock`
(Not present on older versions of RedHat.) Lock a local (text mode) terminal. I can also use `vlock −a` to lock all terminals (probably not a good idea). The best is probably to log out. You don't use vlock in GUI−−the windows managers come with password−protected screensaver and a locking utility (the small icon with padlock in KDE, the keyboard shortcut `<Ctrl><Alt><l>`).

5.5 File management

`cp` *`source destination`*
Copy files. E.g., `cp /home/stan/existing_file_name .` will copy a file to my current working directory. Use the "−R" option (stands for "recursive") to copy the contents of whole directory trees, e.g. , `cp −R my_existing_dir/ ~` will copy a subdirectory under my current working directory to my home directory.

`mcopy` *`source destination`*
Copy a file from/to a DOS filesystem (no mounting of the DOS filesystem is necessary). E.g., `mcopy a:\autoexec.bat ~/junk`. See `man mtools` for other commands that can access DOS files without mounting: `mdir`, `mcd`, `mren`, `mmove`, `mdel`, `mmd`, `mrd`, `mformat` .... We don't use the mtool commands that often−−operations on DOS/MS Windows files can be performed using regular Linux commands after you mount the DOS/MS Windows filesystem.

`mv` *`source destination`*
Move or rename files. The same command is used for moving and renaming files and directories.

`rename` *`string replacement_string filename`*
Flexible utility for changing parts of filenames. For example:
`rename .htm .html *.htm`

`ln` *`source destination`*
Create a hard link called *destination* to the file called *source*. The link appears as a copy of the original files, but in reality only one

copy of the file is kept, just two (or more) directory entries point to it. Any changes to the file are automatically visible throughout. When one directory entry is removed, the other(s) stay(s) intact. The limitation of the hard links are: the files have to be on the same filesystem, hard links to directories or special files are impossible.

`ln -s source destination`
Create a symbolic (soft) link called "destination" to the file called "source". The symbolic link just specifies a path where to look for the "real" file. In contradistinction to hard links, the source and destination do not have to be on the same filesystem. In comparison to hard links, the drawback of symbolic links are: if the original file is removed, the link is "broken"--it points to nowhwere; symbolic links can create circular references (like circular references in spreadsheets or databases, e.g., "a" points to "b" and "b" points back to "a"). In short, symbolic links are a great tool and are very often used (more often than hard links), but they can create an extra level of complexity.

`rm files`
Remove (delete) files. You must own the file in order to be able to remove it (or be "root"). On many systems, you will be asked for a confirmation of deletion; if you don't want this, use the "-f" (=force) option, e.g., `rm -f *` will remove all files in my current working directory, no questions asked.

`mkdir directory`
Make a new directory.

`rmdir directory`
Remove an empty directory.

`rm -r files`
(recursive remove) Remove files, directories, and their subdirectories. Careful with this command as root--you can easily remove all files on the system with such a command executed on the top of your directory tree, and there is no undelete in Linux (yet). But if you really wanted to do it (reconsider), here is how (as root):
`rm -rf /*`

`rm -rf files`
(recursive force remove). As above, but skip the prompt for confirmation, if one is set on your system. Careful with this command particularly as root--see the command above.

`mc`
Launch the "Midnight Commander" file manager (looks like "Norton Commander" for Linux). According to some computer dinosaurs, this is the best file manager ever.

`konqueror &`
(in X terminal) Launch the KDE file manager. Perhaps this is the utltimate for file managment. Much better that the MS "Windows Explorer". It embeds web browsing, pdf viewing, and more. Really cool.

`xwc`
(in X terminal). Another excellent file manager (called "X Win Commander"). Faster than konqueror, but not as loaded with features.

`nautilus &`
(in X terminal). A really cool file manager. Slower than konqueror, but offers me goodies like icon-preview of the content of files (!). It even "previews" the contents of sound files! Speedwise, it runs great on my 1.33 GHz computer, but I don't use it on my 133MHz computer.

5.6 Viewing and editing files

`cat filename | more`
View the content of a text file called "filename", one page a time. The "|" is the "pipe" symbol (on many American keyboards it shares the key with "\"). `more` makes the output stop after each screenful. For long files, it is sometimes convenient to use the commands `head` and `tail` that display just the beginning and the end of the file, or `less` that enables scrolling up and down. If you happened to use `cat` a binary file and your terminal displays funny characters afterwards, you can restore it with the command `reset`.

`cat filename | less`
`less filename`
(two commands, use either) Scroll a content of a text file. Press q when done. "less" is roughly an equivalent to "more" , the command you know from DOS, but often "less" is more convenient than "more" because it lets me scroll both up and down.

`head` *`filename`*
Print first 10 lines of the (long) text file.

`tail` *`filename`*
Print last 10 lines of a long or growing text file. Use `tail −f filename` for tail to follow the file as it grows−−really handy for continuing inspection of log files.

`pico` *`filename`*
Edit a text file using the simple and standard text editor called `pico`. Use `<Ctrl>x` to exit. There are many text editors for Linux, including several GUI−based. A brand new clone of pico (GPLed) is `nano`.

`pico −w` *`filename`*
Edit a text file, while disabling the long line wrap. Handy for editing configuration files, e.g. `/etc/fstab`.

`kwrite`
(in X terminal) Very nice, "advanced text editor". Supports veritical text selection!

`kate`
`kedit`
`gedit`
(in X terminal). Simple yet nice text editors (GUI based).

`gxedit`
(in X terminal) Another multi−purpose, feature packed text editor. This one even has timed backup.

`latte`
(in X terminal) "Code" editor, i.e., plain text editor meant for writing programs.

`nedit`
(in X terminal) Another programmer editor. Very nice and loaded.

`bluefish`
(in X terminal) html editor (source with syntax highlighting and maaaany tools and options).

`ispell` *`filename`*
Spell check an ASCII text file. AbiWord, WordPerfect, StarOffice and other word processors come with "as−you−type" spellchecking, so you really don't have to worry about the simple ispell unless you need it. Newer Linux distributions (e.g., RH7.0) contain an improved spellchecking module called aspell, yet the above command will still work.

`look` *`thermo`*
Look up the dictionary on your system (`/usr/share/dict/words`) for words which start with "thermo".

`wvHtml` *`ms_word_document.doc > filename.html`*
Convert a MS Word document to the html file format.

5.7 Finding files

`find / −name "`*`filename`*`"`
Find the file called "filename" on your filesystem starting the search from the root directory "/". The "filename" may contain wildcards (*,?).

> The find command is very powerful. It has many options that will let you search for files in a variety of ways e.g., by date, size, permissions, owner, ... Yet some search queries can take you more than a minute to compose. See `info find`. Here are some more complex examples for using find to accomplish some useful tasks.
>
> `find $HOME −name core −exec rm −f {} \;`
> The above command finds files named "core", starting from your home directory. For each such file found, it perform the action "rm −f" (force−deleting the file). The {} stands for the file found, and the "\" terminates the command list.
>
> `find /dev −user "peter" |more`
> The above command prints the filename for all devices owned by user "peter". Printing the filename is the default "action" of find, so it does not have to be specified if this is all I need.

```
find /home/peter −nouser −exec ls −l {} \; −ok chown peter.peter {} \;
```
Find files without a valid owner in the /home/peter directory. List the file in a long format. Then prompt to change the ownership to the user "peter" and the group "peter". You probably need to be root to hand over the ownership of a file.

`locate` *`filename`*
Find the file name which contains the string "filename". Easier and faster than the previous command but depends on a database that normally rebuilds at night, so you cannot find a file that was just saved to the filesystem. To force the immediate update of the database, I may do (as root): `updatedb&`.

`which` *`executable_name`*
Show me the full path to the executable that would run if I just typed its name on the command line. For example, this commmand:
`which mozilla`
on my system produces:
`/usr/bin/mozilla`

`whereis` *`command`*
Print the locations for the binary, source, and manual page files of the command "*`command`*".

```
rgrep −r 'celeste' . |more
grep −r 'celeste' . |more
```
(Two commands, use the one that works on your system.) Search all files in the current directory and all its subdirectories (the option "−r" stands for "recursive") for the example string "celeste". Print the filename and the line in the file that contains the searched string.

`kfind &`
(in X terminal). A GUI front−end to `find` and `grep`. Very nice. The & at the end of the command makes `kfind` run in the background so that the X terminal remains available.

5.8 Basics of X−windows

`xinit &`
Start a barebone X−windows server (without a windows manager). The "&" makes the command run in the background.

`startx &`
Start an X−windows server and the default windows manager. Works like typing "win" under DOS with Win3.1.

`startx −− :1 &`
Start another X−windows session on the display 1 (the default is opened on display 0). You can have several GUI terminals running concurrently. Switch between them using <Ctrl><Alt><F7>, <Ctrl><Alt><F8>, etc.

`xterm`
(in X terminal) Run a simple X−windows terminal. Typing `exit` will close it. There are other, more advanced "virtual" terminals for Xwindows. I like the popular ones: `konsole` and `kvt` (both come with kde) and `gnome−terminal` (comes with gnome). If you need something more fancy−looking, try `Eterm`. For something plain and fast, I could select `rxvt`.

```
startkde
gnome−session
xfce
afterstep
AnotherLevel
fvwm2
fvwm
```
(in X terminal, 7 different commands, use the one which starts your fav windows manager) Start your favourite windows manager in an X terminal on bare X server.

5.9 Network apps

`mozilla &`
(in X terminal) Run the mozilla web browser. The current version is Mozilla 1.0.1 (Oct. 2002), and it is very nice. Mozilla is a descendant of netscape (`netscape` is on older linux systems). Good alternatives are also `konqueror` and `galeon` (type `konqueror&` or `galeon&` in your Xterminal).

`mozilla −display` *`host`*`:0.0 &`
(in X terminal) Run mozilla on the current machine and direct the output to machine named "host" display 0 screen 0. Your current

machine must have a permission to display on the machine "host" (typically given by executing the command `xhost` *`current_machine_name`* in the xterminal of the machine host. Other X−windows program can be run remotely the same way.

`lynx` *`file.html`*
View an html file or browse the net from the text mode. Although lynx's look or convenience of use is not as great as GUI−based browser, it is light−weight, almost always works, and does not require any configuration, as long as your networks is functional.

`konqueror &`
(in X terminal) File manager and web browser in one. Very nice, in many very comptetitive to mozilla. Comes with KDE.

`pine`
A good, old−fashioned, text−mode mail reader. Another old−fashioned and standard one is `elm`. Your mozilla mail will read the mail from your Internet account. `pine` will let you read the "local" mail, e.g. the mail your son or a cron process sends to you from a computer on your home network. The command `mail` could also be used for reading/composing mail, but it would be inconvenient−−it is meant to be used in scripts for automation.

`mutt`
A really basic but extremally useful and fast mail reader.

`mail`
A basic operating system tool for e−mail. Look at the previous commands for a better e−mail reader. `mail` is good if you wanted to send an e−mail from a shell script.

`kmail &`
(in X−terminal) Nice, GUI mail program. I use kmail, it is much better than netscape mail. I can have multiple accounts and retrieve mail from the smtp (local) server and pop3 servers (internet service provider) to the same mailbox. Simple and elegant. Supports digital signatures.

`licq &`
(in X terminal) An icq "instant messaging" client. Another good one is `kxicq`. Older distributions don't have an icq client installed, you may have to do download one and install it.

`knode &`
(in X terminal) Start my favourite newsgroup (usenet) reader. It is MUCH better than the netscape's built−in reader.

`talk` *`username1`*
Talk to another user currently logged on your machine (or use "`talk` *`username1@machinename`*" to talk to a user on a different computer) . To accept the invitation to the conversation, type the command "`talk` *`username2`*". If somebody is trying to talk to you and it disrupts your work, your may use the command "`mesg n`" to refuse accepting messages. You may want to use "`who`" or "`rwho`" to determine the users who are currently logged−in. `talk` is one of the old−fashioned "standard" UNIX tools, yet it still can be cool and useful in some situations.

`telnet` *`server`*
Connect to another machine using the TELNET protocol. Use a remote machine name or IP address. You will be prompted for your login name and password−−you must have an account on the remote machine to login. Telnet will connect you to another machine and let you operate on it as if you were sitting at its keyboard (almost). Telnet is not very secure−−everything you type moves through the networks in open text, even your password! A competent system administrator on a computer "on−route" can read what you type. Use `ssh` (requires some setup) for encrypted transmission.

`rlogin` *`server`*
(=remote login) Connect to another machine. The login name/password from your current session is used; if it fails you are prompted for a password.

`rsh` *`server`*
(=remote shell) Yet another way to connect to a remote machine. The login name/password from your current session is used; if it fails you are prompted for a password.

`ssh` *`servername −l username`*
(=secure shell) Connect to a server (remote login) using a secure connection. `ssh` is secure because encrypts all the data transfered over the network using a pair of RSA"public−private" keys. If you don't specify the username, your current user name is assumed.

> Both the client and the server must have ssh service (daemon) running. They are normally available on newer Linux distributions (e.g., RH7.0). Before using ssh, some setup may be necessary. The user creates his/her RSA key pair (for encryption) by running the command `ssh−keygen`. This stores the private key in the file $HOME/.ssh/identity and the public key in $HOME/.ssh/identity.pub in the user's

home directory. To allow automatic login, the user should copy the identity.pub to $HOME/.ssh/authorized_keys in his/her home directory on the remote machine After this, the user can log in without giving the password. The most convenient way to use RSA authentication may be with an authentication agent. See `man 1 ssh-agent` for more information. If automathic authentication methods fail, ssh prompts the user for a password. The password is sent to the remote host for checking; however, since all communications are encrypted, the password cannot be seen by someone listening on the network.

From: Benjamin Smith <bens@benjamindsmith.com> (edited for space):

I recently got openssh 2.9.2p1 up and running, along with the password-free login option. It took some doing and none of the howtos covered this. Would you like the "magic tidbit" that makes it all work? Here it is: "the default is to SSH2 and DSA keys, which you generate with 'ssh-keygen -d' and it goes into ~/.ssh/id_dsa.pub, which you would copy to remotehost:.ssh/authorized_keys2" Use this instead of the usual "authorized_keys" file given in the howtos, and VOILA! It actually works.

`ftp` *`server`*
Ftp another machine. (There is also `ncftp` which adds extra features and `gftp` for GUI .) Ftp is good for copying files to/from a remote machine. Try user "anonymous" if you don't have an account on the remote server. After connection, use "?" to see the list of available ftp commands. The essential ftp commands are: `ls` (see the files on the remote system), `ASCII`, `binary` (set the file transfer mode to either text or binary, important that you select the proper one), `get` (copy a file from the remote system to the local system), `mget` (get many files at once), `put` (copy a file from the local system to the remote system), `mput` (put many files at once), `bye` (disconnect). For automation in a script, you may want to use `ncftpput` and `ncftpget`, for example:
`ncftpput -u my_user_name -p my_password -a remote.host.domain remote_dir *local.html`
"ncftp" seems to have a problem if your computer is behind a firewall--you need to configure the file /home/usr_name/.ncftp/firewall. Alternatively, you may use "lftp" to accomplish the same, for example:
`lftp -e "mput -a *local.html" -u my_user_name,my_password ftp://remote.host.domain`
For keeping mirrors of ftp directories, one can use `fmirror`

`wget -m --no-parent http://sunsite.dk/linux-newbie`
Copy files from web sites. The example above uses the option -m (=mirror) to retrieve a complete set of files from the master site of this guide. The option "--no-parent" limits the retrieval to the files in the given directory and its subdirectories.

`minicom`
Minicom program for serial port "terminal emulation". Looks and works like "Procomm" or "Telix". It is useful for testing and debugging your serial communication.

`rx`
Receive files using the Zmodem, Ymodem, or Xmodem protocol. Xmodem requires a filename. Use `rx --help` for more info. Who uses these protocols any more anyway?

"I use Zmodem regularly. I have two computers running (SuSE) Linux, a laptop and a desktop. The desktop computer does not have access to an internet connection. So, in order to get files I downloaded from one computer to the other, I send them over via a null-modem cable, using Minicom and the Zmodem protocol. This way I can even connect my laptop from work running Win2000 to my linux machine using Reflexion (a win32 terminal emulation prog)" (from Berry Vos, B.Vos@getronics.com, 2001 08 28).

5.10 File (de)compression

`tar -zxvf` *`filename.tar.gz`*
(=tape archiver) Untar a tarred and compressed tarball (*.tar.gz or *.tgz) that you downloaded from the Internet.

`tar -xvf` *`filename.tar`*
Untar a tarred but uncompressed tarball (*.tar).

```
tar czvpf /var/backups/mybackup.tar.gz /home
cd /; tar xzvpf /var/backups/mybackup.tar.gz '*/myfile.rtf'
```
(as root) Create a backup of /home to a compressed file. The second command shows how to restore a file from the backup. This won't include "dotfiles" (the files or directories with names starting with a dot) in my tarball. To tar everything, I would do:
`tar cvzf` *`filename.tgz`* `* .[a-zA-Z]*`

`gunzip` *`filename.gz`*
Decompress a zipped file (*.gz" or *.z). Use `gzip` (also `zip` or `compress`) if you wanted to compress files to this file format. Note the funny pronounciation of "gun zip".

`zcat filename.gz | more`
(=zip cat) Display the contents of a compressed file. Other utilities for operating on compressed files without prior decomprssion are also available: `zless`, `zmore`, `zgrep`, ...

`bunzip2 filename.bz2`
(=big unzip) Decompress a file (*.bz2) zipped with bzip2 compression utility. Used for big files.

`unzip filename.zip`
Decompress a file (*.zip) zipped with a compression utility compatible with PKZIP for DOS.

`zip filename.zip filename1 filename2`
Compress two files "filename1" and "filename2" to a zip archive called "filename.zip".

`unarj e filename.arj`
Extract the content of an *.arj archive.

`lha e filename.lha`
Extract the content of an lharc archive.

`uudecode −o outputfile filename`
Decode a file encoded with `uuencode`. uu−encoded files are typically used for transfer of non−text files in e−mail (uuencode transforms any file into an ASCII file).

`cat filename | mimencode −o filename.mime`
`cat filename.mime |mimencode −u −o filname`
(2 commands.) Encode and then decode back a file to/from the mail−oriented Internet standard for 7−bit data transfer called "mime". On older distributions, the command that does the work (`mimencode`) is called `mmencode`. Usually, you don't have to bother with these commands, your mailer should do the mime encoding/decoding in a transparent way.

`ar −x my_archive.a file1 file2`
(=archiver). Extract files `file1` and `file2` from an archive called `my_archive.a`. The archiver utility `ar` is mostly used for holding libraries.

`ark &`
(in X terminal). A GUI (Qt−based) archiver application. Perhaps that's everything what you need to manage your compressed files. An alternative is `gnozip`.

5.11 Process control

`ps`
(="print status" or "process status") Display the list of currently running processes with their process ID (PID) numbers. Use `ps axu` to see all processes currently running on your system (also those of other users or without a controlling terminal), each with the name of the owner. Use "top" to keep listing the processes currently running.

`any_command` `&`
Run any command in the background (the symbol "&" means "run the preceding command in the background"). The *`job_number`* is printed on the screen so you can bring the command in the foreground (see below) if you want. I use "&" often when starting a GUI program from an X−terminal.

`jobs`
List my background or stopped processes and show their job numbers.

`fg job_number`
Bring a background or stopped process to the foreground.

`bg job_number`
Place a process in the background, so it is exactly as if it had been started with &. This will restart a stopped background process. The current foreground process can often be stopped with `<Ctrl>z`. If you have stopped or background jobs, you have to type `exit` twice in row to log out.

`batch`
at`>updatedb<Ctrl>d`
Run any command (usually one that is going to take more time to complete) when the system load is low. I can logout, and the process will keep running. When the command completes, an email will be sent to me with the output. In the example above, the "at>" represents a prompt, the command to run is `updatedb`, and the `<Ctrl><d>` terminates my input to batch (I could start many commands to run, separated by <Enter>).

`at 17:00`
Execute a command at a specified time. You will be prompted for the command(s) to run, until you press <Ctrl>d. The associated commands are `atq` (display the queue of processes started with `at`) and `atrm` (remove a process from the "at queue").

`kill` *`PID`*
Force a process shutdown. First determine the PID of the process to kill using `ps`.

`killall` *`program_name`*
Kill program(s) by name. For example, `killall pppd` will disconnect your dialup network.

`nohup` *`program_name`*
(=no hungup). Run *`program_name`* so that it does not terminate when you log out. Output is redirected to the file `nohup.out` in your home directory. You surely do not want to run an interactive program under `nohup`.

`xkill`
(in X terminal) Kill a GUI−based program with mouse. (Point with your mouse cursor at the window of the process you want to kill and click.)

`kpm`
(in X terminal) KDE process manager.

`lpc`
(as root) Check and control the printer(s). Type "?" to see the list of available commands.

`lpq`
Show the content of the printer queue. Under KDE (X−Windows), you may use GUI−based "Printer Queue" available from "K"menu−Utilities.

`lprm` *`job_number`*
Remove a printing job "job_number" from the queue.

`nice` *`program_name`*
Run *program_name* adjusting its priority. Since the priority is not specified in this example, it will be increased by 10 (the process will run slower), from the default value (usually 0). The lower the number (of "niceness" to other users on the system), the higher the priority. The priority value may be in the range −20 to 19. Only root may specify negative values. Use `top` to display the priorities of the running processes.

`renice −18` *`PID`*
(as root) Change the priority of a running process to minus 18. Normal users can only adjust processes they own, and only up from the current value (make them run slower). One could also `renice +10 −u peter` to make user peter use fewer cpu clicks so that other user don't suffer when he runs his computing−intensive tasks.

`<Ctrl>c`, `<Ctrl>z`, `<Ctrl>s`, and `<Ctrl>q` also belong to this chapter but they were described previously. In short they mean: stop the current command, send the current command to the background, stop the data transfer, resume the data transfer.

`lsof`
List the opened files. If I am a root, all files will be listed. I can limit myself to files opened by processes owned by the first console if I use `lsof /dev/tty1` . To list only network files (useful for a security audit), I would do `lsof −i` (as root).

`watch −n 60` *`my_command`*
Execute *`my_command`* repeatedly at 60−second intervals (the default interval is 2 seconds).

5.12 Some administration commands

`su`
(=substitute user id) Assume the superuser (=root) identity (you will be prompted for the password). Type "exit" to return you to your previous login. Don't habitually work on your machine as root. The root account is for administration and the `su` command is to ease your access to the administration account when you require it. You can also use "su" to assume any other user identity, e.g. `su barbara` will make me "barbara" (password required unless I am the superuser).

`alias ls="ls −−color=tty"`
Create an alias for the command "ls" to enhance its format with color. In this example, the alias is also called "ls" and the "color" option is only evoked when the output is done to a terminal (not to files). Put the alias into the file `/etc/bashrc` if you would like

the alias to be always accessible to all users on the system. Aliases are a handy way to customize your system. Type "`alias`" alone to see the list of aliases for your account. Use `unalias` *`alias_name`* to remove an alias.

`cat /var/log/httpd/access_log`
Show who connected to your http (apache) server since the last time the log file was "rotated" (normally rotated once a day, when cron runs). The previous log file is `access_log.1`, the yet previous `access_log.2`, etc.

`cat /var/log/secure`
(as root) Inspect the important system log. It is really a good idea to do it from time to time if you use Internet access.

`ftpwho`
(as root) Determine who is currently connected to your ftp server.

`printtool`
(as root in X−terminal) Configuration tool for your printer(s). Settings go to the file `/etc/printcap` and (strangely) `/var/spool/lpd`.

`setup`
(as root) Configure mouse, soundcard, keyboard, X−windows, and system services. There are many distibution−specific configuration utilities, `setup` is the default on RedHat. Mandrake 7.0 offers very nice `DrakConf` .

`linuxconf`
(as root, either in text mode or in the X terminal). You can access and change hundreds of network setting from here. Very powerful−−don't change too many things at the same time, and be careful with changing entries you don't understand. ReadHats network configuration utility `netconf` is a subset of `linuxconf`, therefore it is simpler and sometimes easier to use.

`mouseconf`
(as root). Simple tool to configure your mouse (after the initial installation). Mandrake includes also an alternative `mousedrake`.

`kudzu`
(as root). Automatically determines and configures your hardware. If having mysterious problems with your mouse (or other serial hardware), you may want to disable kudzu, so it does not run on the system startup (kudzu messed up my system so I could not have my mouse working). You can run it manually when you need it.

`timeconfig`
(as root) Set the timezone for your system. My computer hardware clock (BIOS setup) keeps time in UTC (Coordinated Universal Time, which was once called GMT or the Greenwich Mean Time). This way, I avoid any possible problems associated with switching timezones due to the daylight savings time, transfering files across the globe through the network, or a physical travel. It is customary to keep time on a server computers in UTC to avoid time ever going "backwards" (which could cause problems). Timestamps on files are always kept in UTC and displayed in the local time using the time zone information. For example, many applications (e.g., compilers, databases) depend on being able to distinguish a newer file from an older one by comparing their timestamps. It is important to keep the timezone correct. The only reason why I could select to keep BIOS time in the local time is to avoid problems when when dual booting from the same computer, and when the other operating system (MS Windows?) does not know how to handle UTC. Then, I let my Linux server know about this by checking the box "Hardware clock set to GMT", so that Linux can backcalculate the UTC which it needs.

`setclock`
(as root). Set your computer hardware clock from the current linux system time. Use the command "date" first to set up the linux system time. E.g., I could change the date and time to 2000−12−31 23:57 using this command:
`date 123123572000`
and then write the time to the hardware clock using:
`setclock`

`dateconfig&`
(in X−terminal, as root else you will be asked for the root password). An excellent GUI utility to set my operating system and hardware clock and timezone, and tell my BIOS to keep time in UTC. I don't need the previous two commands.

`xvidtune`
(in X−terminal). Adjust the settings for your monitor display for all resolutions so as to eliminate black bands, shift the display right/left/up/down, etc. (First use the knobs on your monitor to fit your text mode correctly on the screen). Then use xvidtune to adjust the monitor frequencies for each resolution so it fits well in your secree. To make the changes permanent, display the frequencies on the screen and then transfer them to the setup file `/etc/X11/XF86Config`. On newer monitors, you may really prefer to adjust your monitor using the built−in monitor settings−−xvidtune is for older monitors which do not have the capability to remember their settings.

`kvideogen`
(in X−terminal). Generate "modelines" for customized resolutions of your screen. After you generated the setup text (the "modelines"), you can copy−paste it to the X−windows setup file `/etc/X11/XF86Config` (or /etc/X11/XF86Config−4 if you use X−server version 4.xx). See also the keyboard shortcut `<Ctrl><Alt><+>`

`SVGATextMode 80x25x9`
`SVGATextMode 80x29x9`
(as root) Change the text resolution in the text terminal. In the above example (second line) I changed the text screen to 80 columns x 29 lines with characters 9 pixels high. The first line defines a resolution that always works, so that if the second command did not work on my system, I can press <ArrowUP> twice and <Enter> to regain control over my screen. The possible modes depend on your video card and your monitor synchronization frequencies−−I needed to edit (as root) the file `/etc/TextConfig` and (un)comment the proper lines to let SVGATextMode know what my system supports.

`SuperProbe`
(as root). A utility to determine the type of the video card and the amount of its memory.

`cat /var/log/XFree86.0.log`
A log file for X that can be useful to determine what is wrong with your X setup. The "0" in the filename stands for "display 0"−−modify the filename accordingly if you need log for displays "1", "2", etc.

`lspci`
Show info on your motherboard and what cards are inserted into the pci extension slots. My older computer has ISA slots (or EISA) slots, no pci.

`lsdev`
Display info about your hardware (DMA, IRQ, IO ports).

`lsof|more`
List files opened on your system.

`kernelcfg`
(as root in X terminal). GUI to to add/remove kernel modules. Module is like a device driver−−a piece of Linux kernel that provides support for a particular piece of hardware or functionality. You can do the same from the command line using the command `insmod.`

`lsmod`
(= list modules). List currently loaded kernel modules. A module is like a device driver−−it provides operating system kernel support for a particular piece of hardware or feature.

`modprobe −l |more`
List all the modules available for your kernel. The available modules are determined by how your Linux kernel was compliled. Almost every possible module/feature can be compiled on linux as either "hard wired" (perhaps a bit faster, but non−removable), "module" (maybe a bit slower, but loaded/removable on demand), or "no" (no support for this feature at all). The modules which your kernel supports (with which it was compiled) are all as files under the directory `/lib/modules` (and the subdirectories) so browsing it may give you a clue if you are lost. If your kernel does not support a module you require, you may need to re−compile your kernel with this module enabled (this is rare because the "stock" RedHat or Mandrake Linux kernels come with almost all common and non−experimental modules pre−compiled. Still, if you have a bleeding edge hardware ...).

`modprobe sb`
Load the soundblaster (sb) module. Use the previous command to find other kernel modules there are to load.

`insmod parport`
`insmod ppa`
(as root) Insert modules into the kernel (a module is roughly an equivalent of a DOS device driver). Normally, I use "modprobe" (see the previous command) to insert modules. This example shows how to insert the modules for support the external parallel−port 100−MB zip drive (it appears to be a problem to get the external zip drive to work in any other way under RH6.0 and 6.1). For the 250−MB external zip, I use the `imm` module instead of `ppa`.

`rmmod` *`module_name`*
(as root, not essential). Remove the module *module_name* from the kernel.

`depmod −a`
(as root) Build the module dependency table for the kernel. Not essential unless you modified /etc/modules and don't wish to reboot.

`setserial /dev/cua0 port 0x03f8 irq 4`
(as root) Set a serial port to a non−standard setting. The example here shows the standard setting for the first serial port (cua0 or ttyS0). The standard PC settings for the second serial port (cua1or ttyS1) are: address of i/o port 0x02f8, irq 3. The third serial port (cua2 or ttyS2): 0x03e8, irq 4. The forth serial port (cua3 or ttyS3): 0x02e8, irq 3. Add your setting to `/etc/rc.d/rc.local` if you want it to be set at the boot time. See man setserial for good a overview.

`tunelp`
(as root, rarely needed) Tune up your parallel ports.

`/sbin/chkconfig −−level 123456 kudzu off`
(as root)A tool to check/enable/disable system services which will automatically start under different runlevels. Typically, I just use RedHat `ntsysv` utility if I need to enable/disable a service in the current runlevel, but `chkconfig` does give me an extra flexibility. An alternative tool is `tksysv` (X−based). The example above shows how to disable kudzu service so it does not start up at any runlevel (it messes up mouse on one of my computers). To list all the services started/stopped under all runlevels, I use:
`chkconfig −−list | more`
To check the current status of services, I may use:
`service −−status−all`
To start a service right now, I may use something like (starts an ftp server):
`service wu−ftpd start`
To re−start samba networking (e.g., after I changed its configuration), I may use:
`service smb restart`

`symlinks −r −cds /`
(as root) Check and fix the symbolic links on my system. Start from / and progress through all the subdirectories (option −r="recurse") and change absolute/messy links to relative, delete dangling links, and shorten lengthy links (options −cds). If my filesystem spreads over different hard drive partitions, I need to re−run this command for each of them (e.g., `symlinks −r −cds /usr`).

`cd /usr/src/linux−2.4.7−10`
`make xconfig`
(as root in X terminal). A nice GUI front−end for configuration of the kernel options in preparation for compilation of your customized kernel. (The directory name in the example contains the version of my Linux kernel so you may need to modify the directory name if your Linux kernel version is different than 2.4.7−10 used in this example. You need the "Tk" interpreter to run "make xconfig", and the kernel source code installed.) The alternatives to "make xconfig" are: "make config" (runs a scripts that asks you questions in the text mode) and "make menuconfig" (runs a text−based menu−driven configuration utility).
Try: `less /usr/share/doc/HOWTO/Kernel−HOWTO` for more information.
After configurating the options for the new kernel with "make xconfig", I may proceed with compilation of the new kernel by issuing the following commands:
`make clean` (this is optional; it cleans the old object files, may lengthen compilation, may prevent problems in some situations)
`make dep`
`make bzImage`
The last command will take some time to complete (maybe 10 min or 2 h, depending on your hardware). It produces the file `arch/386/boot/bzImage`, which is your new Linux kernel. Next:
`make modules`
`make modules_install`
Now you have the new modules installed in `/lib/modules/KernelName`.

> Don't rename the module directory if you want to run multiple kernels−−the kernel must be able to find its "matching" modules. If I want to change the kernel name, I have to edit the main kernel makefile (e.g., /usr/src/linux−2.2.14/Makefile) and change the lines right at the top. Mine (default RH7.2) are:
>
> ```
> VERSION = 2
> PATCHLEVEL = 4
> SUBLEVEL = 7
> EXTRAVERSION = −10custom
> ```
>
> The kernel name for the currently running kernel can be displayed using `uname −r` . Mine (default RH7.2) is "2.4.7−10custom".
> The configuration for my "original" RedHat kernel is in the file /boot/config−2.4.18−14 (RedHat 8.0), while some addtional "custom" kernel configurations are in the directory /usr/src/linux−x.x.x/configs. I can load any of those from a dialog box in available from "make xconfig".

Now I can install the new kernel. The installation involves copying the new kernel (while renaming it) into the `/boot` directory:
`cp arch/386/boot/bzImage /boot/vmlinuz−2.4.7−10custom`
`cp System.map /boot/System.map−2.4.7−10custom`
and making changes to `/etc/lilo.conf` or `/boot/grub/grub.conf` so I can select at the boot time which kernel (the old or the new) to boot. It is strongly advised that you preserve the old kernel as a boot option (in case the new kernel refuses to boot).

If you use initrd (initial ram disk) for two−stage booting, you may also need to create an image with modules used by the kernel during startup:
`mkinitrd /boot/initrd−2.4.7−10custom.img 2.4.7−custom`
See this for details on kernel patching. Quick reference:
`cd /usr/src/linux−2.4.7−10`
`patch −E −p1 < /home/download/the_patch_to_apply`
It may also be helpful to read: `/usr/doc/HOWTO/Kernel−HOWTO` and perhaps `man depmod`. Configuration, compilation and installation of a new kernel is quite simple but it CAN lead to problems. Compilation of a kernel is also a good way to test your hardware, because it involves considerable amount of computing. If your hardware is "flaky", you may receive the "signal 11" error (then read the `/usr/doc/FAQ/txt/GCC−SIG11−FAQ`).

`ldconfig`
(as root) Re−create the bindings and the cache for the loader of dynamic libraries ("ld"). You may want to run `ldconfig` after an installation of new dynamically linked libraries on your system. (It is also re−run every time you boot the computer, so if you reboot you don't have to run it manually.)

`mknod /dev/fd0 b 2 0`
(=make node, as root) Manually create a device file. This example shows how to create a device file associated with your first floppy drive and could be useful if you happened to accidentally erase it. The options are: b=block mode device, c=character mode device, p=FIFO device, u=unbuffered character mode device. The two integers specify the major and the minor device number. I normally wouldn't know the parameters which `mknod` requires. So to make devices, I first read `man MAKEDEV` to figure the name of the device and then run the script `/dev/MAKEDEV` which knows about Linux devices by their names−−see the next command. If the mentioned manual page does not help, I may refer to the ultimate documentation included with the kernel source code:
`less /usr/src/linux/Documentation/devices.txt`

`cd /dev`
`./MAKEDEV audio`
(as root). Restore the "audio" device that I just somehow screwed up. Also see the previous command.

5.13 Hard Drive/Floppy Disk Utilities

`fdisk /dev/hda`
(= "fixed disk". As root.) Linux hard drive partitioning utility (DOS has a utility with the same name). In the example above, I specified that I would like to partition the first harddrive on the first IDE interface, hence "hda". If I were you, i would backup any important data before using fdisk on any partition. I do not not know anybody who likes `fdisk` (either Linux or DOS edition)−−I prefer easier to use `cfdisk`, see next command.

`cfdisk /dev/hda`
(as root) Hard drive partitioning utility, menu−based. Easier to use then the plain−vanilla `fdisk` (see the previous command). Physical drives can contain primary partitions (max 4 per disk), and logical partitions (no restriction on number). A primary partition can be bootable. Logical partitions must be contained within "extended partitions"; extended partitions are not usable by themselves, they are just a container for logical partitions. When partitioning a disk, I typically: (1) create a primary partition (2) make the primary partition bootable (3) create an extended partition, (4) create logical partition(s) within the extended partition.

`sfdisk −l −x |more`
(as root) List the partition tables (including extended partitions) for all drives on my system.

`parted /dev/hda`
A partition manipulation utility for Linux (ext2), and DOS (FAT and FAT32) hard drive partition. It is for creation, destroying, moving, copying, shrinking, and extending partitions. You should really like to backup your data and carefully read `info parted` before using it.

`fdformat /dev/fd0H1440`
`mkfs −c −t ext2 /dev/fd0`
(=floppy disk format, two commands, as root) Perform a low−level formatting of a floppy in the first floppy drive (/dev/fd0), high density (1440 kB). Then make a Linux filesystem (−t ext2), checking/marking bad blocks (−c). Making the filesystem is an equivalent to the high−level formatting. I can also format floppies to different (also non−standard) densities; try `ls /dev/fd0<Tab> .` I am also able to format to the default density (normally 1440k) using `fdformat /dev/fd0`.

`badblocks /dev/fd01440 1440`
(as root) Check a high−density floppy for bad blocks and display the results on the screen. The parameter "1440" specifies that 1440 blocks are to be checked. This command does not modify the floppy. `badblocks` can be also used to check the surface of a hard

drive but I have to unmount the filesystem first to do a full read−write check:
`mount` [to find out which device contains the disk partition I wish to check for bad blocks]
`umount /dev/hda8` [unoumnt the selected partition]
`badblocks −n /dev/hda8` [check the selected partition in a non−destructive read−write mode, so that my data is not erased!]
`mount /dev/hda8` [mount the partition back since no info on bad blocks was printed]
If bad blocks are found, they can be marked on the hard drive so that will not be used using:
`e2fsck −c /dev/hda8`

`fsck −t ext2 /dev/hda2`
(=file system check, as root) Check and repair a filesystem, e.g., after an "unclean" shutdown due to a power failure. The above example performs the check on the partition hda2, filesystem type ext2. You definitely want to unmount the partitions or boot Linux in the "single mode" to perform this (type "linux single" at the LILO prompt or use `init 1` as root to enter the single user mode). If errors are found during the filesystem checkup, I accept the defaults for repair.

`tune2fs −j /dev/hda2`
(as root, only for kernel that support ext3−−RH7.2) Adjust the tuneable parameter of an ext2 filesystem. The example above shows how to add a journal to a disk partition (hda2 in this example), effectively converting the file system to ext3 (journaling) filesystem. To complete the transition, you must also edit the file `/etc/fstab` and change the filesystem type from ext2 to ext3, else you may run into problems−−ext2 will not mount an uncleanly shut down journaled filesystem! To check what is the type of the filesystem use `mount` (with no arguments) or `cat /etc/mtab`. If you need more information on ext3 setup, try: http://www.symonds.net/~rajesh/howto/ext3/ext3−5.html.
Other options of tune2fs let you me add a volume label, adjust the number of mounts after which the filesystem check is performed (maximal mount count), or turn on time−based filesystem checks instead (less often used).

`dd if=/dev/fd0H1440 of=floppy_image`
`dd if=floppy_image of=/dev/fd0H1440`
(two commands, dd="data duplicator") Create an image of a floppy to the file called "floppy_image" in the current directory. Then copy `floppy_image` (file) to another floppy disk. Works like DOS "DISKCOPY".

`mkbootdisk −−device /dev/fd0 2.4.2−3`
Make an emergency boot floppy. You are typically asked if you would like to make a boot disk during the system installation. The above command shows how to make it after install, on the first floppy drive (`/dev/fd0`). Your kernel name (needed in the command, here 2.4.2−3) can be determined either by running `uname −a` or `ls /lib/modules`.

5.14 Management of user accounts and files permissions

`useradd` *`user_name`*
`passwd` *`user_name`*
(as root) Create a new account (you must be root). E.g., `useradd barbara` Don't forget to set up the password for the new user in the next step. The user home directory (which is created) is `/home/`*`user_name`*`.` You may also use an equivalent command
`adduser` *`user_name`*

`ls −l /home/peter`
`useradd peter −u 503 −g 503`
(as root). Create an account to match an existing directory (perhaps from previous installation). If the user ID and the group ID (shown for each file) were both 503, I create an account with a matching user name, the user ID (UID) and the group ID (GID). This avoids the mess with changing the ownership of user files after a system upgrade.

`userdel` *`user_name`*
Remove an account (you must be a root). The user's home directory and the undelivered mail must be dealt with separately (manually because you have to decide what to do with the files). There is also `groupdel` to delete groups.

`groupadd` *`group_name`*
(as root) Create a new group on your system. Non−essential on a home machine, but can be very handy even on a home machine with a small number of users.

> For example, I could create a group "friends", using
> `groupadd friends`
> then edit the file `/etc/group`, and add my login name and the names of my friends to the line that lists the group, so that the final line might look like this:
> `friends:x:502:stan,pete,marie`
> Then, I can change the permissions on a selected file so that the file belongs to me AND the group "friends".
> `chgrp friends` *`my_file`*

Thus, the listed members of this group have special access to these files that the rest of the world might not have, for example read and write permission:

`chmod g=rw,o= my_file`

The alternative would be go give write permission to everybody, which is definitely unsafe even on a home computer.

`groups`
List the groups to which the current user belongs. Or I could use `groups john` to find to which groups the user john belongs.

`usermod`
`groupmod`
(as root) Two command−line utilities to modify user accounts and groups without manual editing of the files `/etc/passwd` `/etc/shadow` `/etc/group` and `/etc/gshadow`. Normally non−essential.

`userconf`
(as root) Menu−driven user configuration tools (password policy, group modification, adding users, etc). Part of linuxconf package, but can be run separately.

`passwd`
Change the password on your current account. If you are root, you can change the password for any user using: `passwd` *`user_name`*

`chfn`
(="change full name"). Change the information about you (full name, office number, phone number, etc). This information is displayed when the `finger` command is run on your *login_name*.

`chage −M 100` *`login_name`*
(= "change age"). Set the password expiry to 100 days for the user named *login_name* .

`quota username`
`setquota username`
`quotaon /dev/hda`
`quotaoff /dev/hda`
A set of commands to manage user disk quotas. Normally not used on a home computer. "Disk quota" means per−user limits on the usage of disk space. The commands (respectively) display the user quota, set the user quota, turn the quota system on the for a given filesystem (/dev/hda in the above example), turn the quota system off. "Typical" Linux distros I have seen set on default: no limits for all users, and the quota system is off on all filesystems.

`kuser`
(as root, in X terminal) Manage users and groups using a GUI. Nice and probably covering most of what you may normally need to manage user accounts.

`chmod perm filename`
(=change mode) Change the file access permission for the files you own (unless you are root in which case you can change any file). You can make a file accessible in three modes: read (r), write (w), execute (x) to three classes of users: owner (u), members of the group which owns the file (g), others on the system (o). Check the current access permissions using:
`ls −l` *`filename`*
If the file is accessible to all users in all modes it will show:
`rwxrwxrwx`
The first triplet shows the file permission for the owner of the file, the second for the group that owns the file, and the third for others ("the rest of the world"). A "no" permission is shown as "−".
When setting permissions, these symbols are used: "u"(=user or owner of the file), "g"(=group that owns the file), "o"(=others), "a" (=all, i.e., owner, group and others), "="(=set the permission to), "+"(=add the permission), "−"(=take away the permission), "r"(=permission to read the file), "w"=(write permission, meanning the permission to modify the file), "x"(=permission to execute the file).

For example, this command will **add** the permission to read the file `junk` to all (=user+group+others):

`chmod a+r junk`

This command will remove the permission to execute the file `junk` from others:

`chmod o−x junk`

Also try here for more info.
You can set the default file permissions for the new files that you create using the command `umask` (see `man umask`).

`chown new_ownername filename`
`chgrp new_groupname filename`
Change the file owner and group. You should use these two commands after you copy a file for use by somebody else. Only the owner of a file can delete it.

`lsattr files`
List attributes for the file(s). Not very often used because the most interesting attributes are still not implemented. The attributes can be changed using the `chattr` command. The attributes are: A (=don't update atime when the file is modified), S (=synchronous updates), a (=append only possible to this file), c (=file compressed on the kernel level, not implemented yet), i (=immutable file), d (=no dump), s (=secure deletion), and u (undeletable, not implemented yet). An interesting usage may be to make a file undeletable even by root (until s/he clears the attribute).

`sudo /sbin/shutdown −h now`
(as a regular user, I will be prompted for my user password) Run the command "shutdown" (or another command which you have been given permission to run by your system administrator). With sudo, the administrator can give selected users the rights to run selected commands, without handing out the root password. The file `/etc/sudoers` must be configured to contain something like:
`my_login_name my_host_computer_name = /sbin/shutdown`

`pwck`
`grpck`
(as root, two commands). Verify the integrity of the password and group files.

`pwconv`
`grpconv`
(as root) Unlikely you need these commands. They convert old−style password and group files to create the more−secure "shadow" files.

5.15 Program installation

`rpm −ivh package_name−version.platform.rpm`
(as root) Install a package (option "i", must be the first letter after the dash), while talking to me a lot (option "v'=verbose) and printing "hashes" to show installation progress (option "h"). rpm stands for "Redhat Package Manager".

`rpm −Uvh package_name−version.platform.rpm`
(as root) Upgrade (option "U", must be the first letter after the dash) a package, while being verbose (option "v") and displaying hashes ("h").

`rpm −ivh −−force −−nodep package_name−version.platform.rpm`
(as root) Install the package ignoring any possible conflicts and package dependency problems.

`rpm −e package_name`
(as root) Uninstall (option "e"=erase) the package package_name. Please note the absence of "−version.platform.rpm" at the end of the package name (the package name is the same as the name of the *.rpm file from which the package was installed but without the dash, version, platform and "rpm").

`rpm −qpi package_name−version.platform.rpm`
Query (option "q", must be the first letter after the dash) the yet uninstalled package (option "p") so that it displays the info (option "i") which the package contains.

`rpm −qpl package_name−version.platform.rpm`
Query (option "q", must be the first letter after the dash) the yet uninstalled package (option "p") so that it displays the listing (option "l") of all the files the package contains.

`rpm −qf a_file`
Find the name of the installed package to which the file "a_file" belongs or belonged. Useful if I accidentally erased a file and now I need to find the right package and re−install it.

`rpm −qi package_name`
Query the already installed package so that it displays the info about itself. Please note the absence of "−version.platform.rpm" at the end of the package name.

`rpm −qai | more`
Query all the packages installed on my system so that they display their info. On my simple system, I have ~600 packages installed so obviously, I must have a lot of time to read all their info. To count your packages, try:
`rpm −qa | grep −c ''`
To find a particular package, try:
`rpm −qa | grep −i the_string_to_find`
(The option −i makes grep ignore the case of the characters, so upper or lower case letters will match.)

`rpm −Va`
Verify (the option "V") all the packages (option "a") installed on my system. This lists files that were modified since the installation. Here is the legend for the output:

```
.      Test passed
c      This is a configuration file
5      MD5 checksum failed
S      File size is different
L      Symbolic link has changed
T      File modification time changed
D      Device file is modified
U      User that owns the file has changed
G      Group that owns the file has changed
M      File mode (permissions and/or file type) has been modified
```

`kpackage`
`gnorpm`
`glint`
(in X terminal, as root if you want to be able to install packages, 3 commands) GUI fronts to the Red Hat Package Manager (rpm). "glint" comes with RH5.2 and seems obsolete now. `gnorpm` is the "official" RedHat GUI package installer, older versions are very slow and confusing but the newer version (the one that comes with RH7.0) is vastly improved. `kpackage` is the "official" KDE program and has been pretty good all along. Use any of them to view which software packages are installed on your system and the what not−yet−installed packages are available on your RedHat CD, display the info about the packages, and install them if you want (installation must be done as root).

5.16 Accessing drives/partitions

`mount`
See here for details on mounting drives. Examples are shown in the next commands.

`mount −t auto /dev/fd0 /mnt/floppy`
(as root) Mount the floppy. The directory `/mnt/floppy` must exist, be empty and NOT be your current directory. No setup in `/etc/fstab` is necessary because you supplied the command with all the information required and you are a root. The type of the filesystem will be autodetected.

`mount −t auto /dev/cdrom /mnt/cdrom`
(as root) Mount the CD. You may need to create/modify the `/dev/cdrom` file depending where your CDROM is. The directory `/mnt/cdrom` must exist, be empty and NOT be your current directory.

`mount /mnt/floppy`
(as user or root) Mount a floppy as user. The file `/etc/fstab` must be set up to do this. The directory `/mnt/floppy` must not be your current directory.

`mount /mnt/cdrom`
(as user or root) Mount a CD as user. The file `/etc/fstab` must be set up to do this. The directory `/mnt/cdrom` must not be your current directory.

`umount /mnt/floppy`
Unmount the floppy. The directory `/mnt/floppy` must not be your (or anybody else's) current working directory. Depending on your setup, you might not be able to unmount a drive that was mount by somebody else.

`mount /mnt/hda1 /mnt/dos_drive1`
Mount a DOS (MS Windows) partition from your local hard drive.

5.17 Network administration tools

`netconf`
(as root) A very good menu−driven setup for your network.

`ping` *`machine_name`*
Check if you can contact another machine (give the machine's name or IP), press <Ctrl>C when done (without <Ctrl>c, the command keeps going). As all Linux commands, `ping` has options, including the "ping of death" attack, when it seems you can ping some servers so they die−−try the the opitons −f and −s.

`route −n`
Show the kernel routing table.

`host` *`host_to_find`*
`nslookup` *`host_to_find`*
`dig` *`ip_to_find`*
(Three commands, use any.) Query your default domain name server (DNS) for an Internet name (or IP number) *host_to_find*. This way you can check if your DNS works. You can also find out the name of the host of which you only know the IP number.

`traceroute` *`host_to_trace`*
Have a look how your messages trace to *`host_to_trace`* (which is either a host name or IP number).

`mtr` *`host_to_trace`*
(as root) A powerful and nice tool that combines the functionality of the older ping and traceroute (RH7.0)

`nmblookup −A` *`ip_address`*
Status of a networked MS Windows machine (with an NetBIOS name). This command is an equivalent of Windows `nbtstat` command.

`ipfwadm −F −p m`
(for RH5.2, see the next command for RH6.0) Set up the firewall IP forwarding policy to masquerading. (Not very secure but simple.) Purpose: all computers from your home network will appear to the outside world as one very busy machine and, for example, you will be allowed to browse the Internet from all computers at once.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipfwadm−wrapper −F −p deny
ipfwadm−wrapper −F −a m −S xxx.xxx.xxx.0/24 −D 0.0.0.0/0
```
(three commands, RH6.0). Does the same as the previous command. Substitute the "x"s with digits of your class "C" IP address that you assigned to your home network. See here for more details.

```
ipchains −P forward DENY
ipchains −A forward −s xxx.xxx.xxx.0/24 −d 0.0.0.0/0 −j MASQ
```
(two commands, RH7.0). Same as previous commands, but works under RH7.0.

`ipchains −L`
List all firewall rules. Use to check if your firewalling setup works.

`iptables −L`
Linux kernel 2.4.x uses new firewalling "iptables". The above example lists the firewall rules.

`firewall−config`
(as root, in Xterm). A GUI for building your custom firewall.

`ifconfig`
(as root) Display info on the network interfaces currently active (ethernet, ppp, etc). Your first ethernet should show up as eth0, second as eth1, etc, first ppp over modem as ppp0, second as ppp1, etc. The "lo" is the "loopback only" interface which should be always active. Use the options (see `ifconfig −−help`) to configure the interfaces.

`ifup` *`interface_name`*
(`/sbin/ifup` to run as a user) Startup a network interface. E.g.:
```
ifup eth0
ifup ppp0
ifup ppp1
```

Users can start up or shutdown the ppp interface only when the permission is given in the ppp setup (using `netconf`). To start a ppp interface (dial−up connection), I normally use kppp available under the KDE "K" menu (or by typing `kppp` in an X−terminal).

`/etc/rc.d/init.d/network restart`
Restart the network using its normal initialization script (the same which is used during bootup). Useful if you just have manually made changes to your network configuration. Any other service listed in init.d can be stopped, started, or restarted in a similar way (call the script with an options `stop`, `start` or `restart`).

`ifdown` *`interface_name`*
(`/sbin/ifdown` to run it as a user). Shut down the network interface. E.g.: `ifdown ppp0` Also, see the previous command.

`netstat | more`
Displays a lot (too much?) information on the status of your network.

`/usr/sbin/mtr −−gtk`
(as root, in X windows if you wish the nice gtk−based interface). Network diagnostic tool combining the capabilities of traceroute and ping. Comes with RH7.0.

`nmap` *`ip_number`*
Map the ports on the machine with *ip_number*. REALLY useful to establish the security of your network configuration as you can see the opened ports. `nmap` is included on the RH7.0 "Linux PowerTools" CD, as is a convenient GUI front end, "nmapfe". nmap can also do operating system "fingerprinting". Normally, people (and their ISPs) don't like their computer ports being scanned (they view it as possbily probing before an attack) so they may complain if they find out−−learn how to use nmap on your own computers else you will soon hear from your ISP (the complaints will go to them). How do I know this?

`ethereal`
(as root, in Xterminal) Network analyzer−−view the network trafic going through your computer. Included on the RH7.0 "Linux PowerTools" CD. Using ethereal may be unethical in some situations, and unauthorized use in the workplace could be a fireable offence.

`tcpdump −i ppp0 −a −x`
(as root) Print all the network traffic going through the first over−the−phone interface (ppp0) as ascii and hexadecimal. Probably too much printout. `tcpdump` is a rather raw tool and it can be useful for building more "customized" tools for listening to/log what you need.

5.18 Music−related commands

`cdplay play 1`
Play the first track from a audio CD. Use `cdplay` to play the whole CD. Use `cdplay stop` when had enough.

`eject`
Get a free coffee cup holder :))). (Eject the CD ROM tray). This command defaults to the cdrom, but could be used to eject other removable media by specifying the mount point or device. E.g., I can eject the zipdisk from a parallel−port (external) zipdrive (as root) using: `eject /dev/sda4`

`play` *`my_file.wav`*
Play a wave file.

`rec` *`my_file.wav`*
Record a wave file from my microphone.

`mpg123` *`my_file.mp3`*
Play an mp3 file.

`mpg123 −w` *`my_file.wav my_file.mp3`*
Create a wave audio file from an mp3 audio file. Useful if you wanted to write a regular audio CD from mp3s−−you have to convert the mp3s to the *.wav format first. Don't be surprised the conversion is slow−−decompressing mp3s is very processor intensive.

`xmms &`
(in X terminal) Nice GUI mp3 player.

`freeamp &`
(in X terminal) Another GUI mp3 player.

`lame input_file output_file`
MP3 encoder. You may need to download and install it yourself (standard Linux distributions avoid supplying it because of disagreement about patents on the mp3 compression technique).

`knapster`
(in X terminal) Start the program to downoload mp3 files that other users of napster have displayed for downloading. You may share your mp3s too. Really cool, while it lasts. Gnutella and FreeNet will soon replace them−>it gets even cooler.

`cdparanoia −B "1−"`
(CD ripper) Read the contents of an audio CD and save it into wavefiles in the current directories, one track per wavefile. The "1−" means "from track 1 to the last". −B forces putting each track into a separate file.

`grip&`
(in X terminal) A GUI to ripping (see the previous command).

`playmidi my_file.mid`
Play a midi file. `playmidi −r my_file.mid` will display text mode effects on the screen.

`sox audio_file another_format_audio_file`
(="SOund eXchange") Convert from almost any audio file format to another (but not mp3s). See `man sox` for the list of supported audio file formats (many). `sox` also lets you add special effects to your sound file.

`kscd`
(in X terminal) CD player.

`kmidi`
(in X terminal) MIDI player.

`kmid`
(in X terminal) MIDI/caraoke player.

`kmix`
(in X terminal) Sound mixer.

`studio&`
(in Xterminal) Sound Studio−−edit sound files, add effects, etc. Available on the on the PowerTools CD, RH7.x.

`extace&`
(in Xterminal) Sound visualization utility.

`festival −−tts my_file.txt`
Say the content of the `my_file.txt` file (ascii text). "festival" is a speach synthesizer that comes on the RedHat 7.0 "Linux PowerTools" CD. To say something from the command line, you need to start up "festival" and then, at the "festival>" prompt, type the appropriate command ("scheme" language interpreter), as in this example (**bold** represents the prompt):
`festival`
festival`>(SayText "good dog, really good dog")`
festival`> (quit)`

5.19 Graphics−related commands

`kghostview my_file.ps`
(in X terminal) Display a postscript (or pdf) file on screen. I can also use the older−looking `ghostview` or `gv` for the same end effect. I can print the postscript file from the viewer too.

`xpdf my_file.pdf`
(in X terminal) View a pdf file. For viewing pdf files, I prefer the Adobe Acrobat Reader for Linux (it is faster):
`acroread my_file.pdf`
It can be downloaded from: http://www.adobe.com/products/acrobat/readstep2.html

`enscript my_file.txt −U 2`
Convert a text file to postscript and print it to the default printer. I could also send the output to a postscript file:
`enscript my_file.txt −U 2 −o my_file.ps`
The option −U 2 makes `enscript` print 2 logical pages on one physical page which saves me paper, and creates more convenient,

compact printouts. You may also select four pages per page, more makes the printout kind of difficult to read. `enscript` is really flexible, see `man enscript` to select from among the many formatting options.

`ps2pdf my_file.ps my_file.pdf`
Make a pdf (Adobe portable document format) file from a postscript file.

`mpage −2 my_file.ps > new_file.ps`
Print the postscript file `my_file.ps`, outputting two logical pages on one physical page. Save the output to the file `new_file.ps`.

`ps2ps file.ps new_file.ps`
`psnup −nup 2 −pletter new_file.ps new_file2.ps`
Another way of making a postscript file containing 2 logical pages on one physical page. First, I used the "postscript distiller" `ps2ps` to make the postscript file simpler (at the cost of it becoming much larger). Then, I used the `psnup` utility to make `new_file2.ps` which contains 2 logical pages per one physical page. I could have also put 4 or 8 logical pages per one physical page.

`gimp`
(in X terminal) A humble looking but very powerful image processor. Takes some learning to use, but it is great for artists, there is almost nothing you can't do with gimp. Use your mouse right button to get local menus, and learn how to use layers. Save your file in the native gimp file format *.xcf (to preserve layers for future editing) and only then flatten it and save as png (or whatever) for use. "Learning how to make proper selection is the key."

`gphoto`
(in X terminal) Powerful photo editor and camera image acquisition program.

`kpaint`
(in X terminal) Simple bitmap paint program ("paintbrush"−type).

`xfig`
(in X terminal) A simple drawing program. Useful for making elementary sketches or diagrams.

`dia`
(in X terminal) A tool for drawing diagrams from pre−built components.

`display` *`my_picture`*
(in X terminal) Display a picture for viewing only. You can also type `display &` and select file from the menu to view the image, rotate it , change its colour, apply certain effects, etc. `display` is part of ImageMagick package, together with several other utilities described below.

`identify −verbose` *`my_picture`*
Give me a description of an image file *my_picture*: format, type, class, size in pixels, number of colours, size in bytes, etc.

`convert −geometry 60x80` *`my_picture.gif out_small_picture.gif`*
Scale a picture to a size 60x80 pixels. See a few line down for an example how to use convert to convert between different graphical file formats.

`animate −delay 6x5` *`pic1 pic2 pic3`*
Keep showing two or more pictures in sequence. In the example above, the picture files are named pic1, pic2 and pic3, the delay between pictures is 0.06 second, and the whole presentation sequence is repeated in 5 seconds.

`combine pic1 pic2 combined_pic.miff`
Combine two or more images to another image. You can for example put a logo on every image.

`montage −geometry 800x600+0+0 my_picture montage.miff`
Create a tiled image from `my_picture` so that the total size is 800x600 pixels, with 0x0 border. The output goes to the file montage.miff.

`zgv my_picture`
Display a picture for viewing on a vga screen (no X necessary).

`giftopnm` *`my_file.giff > my_file.pnm`*
`pnmtopng` *`my_file.pnm > my_file.png`*
Convert the proprietary giff graphics into a raw, portable pnm file. Then convert the pnm into a png file, which is a newer and better standard for Internet pictures (better technically plus there is no danger of being sued by the owner of giff patents).

`xwd −out my_cupture_screen_file.xwd`
(in X terminal) Capture the contents of X−windows screen into a graphics X−windows "dump" file (*.xwd). You can later convert the xwd file into your favourite format using the convert utility. Unde KDE, you can also use the keyboard shortcuts `<Alt><PrintScreen>` or `<Ctrl><Alt><PrintScreen>` to copy the current window or the entire desktop into the clipboard.

`convert my_capture_screen_file.xwd my_capture_screen.jpg`
Convert the X−windows screen dump file (*.xwd) into the *.jpg file format. The convert utility can convert graphics from/to many different file formats.

`import −display 192.5.100.10:0 −window root my_file.jpeg`
Capture the contents of the root screen from X−windows runnning on server 192.5.100.10 display 0. The output file is my_file.jpeg (change the file format by giving it an appropriate filename extension). You need to have the permission to write to the screen in order to be able to capture its content (the permission to everybody can be given by running `xhost +` in the X−terminal). See `man import` for options.

`ksnapshot`
(in X terminal) GUI−based utility to capture screen contents.

`xine frankenstein.avi &`
(in X terminal). Watch the movie from the file "frankenstein.avi". Looks better than on a TV :))

5.20 Small games

Many small games are probably installed on your system. Here is just a sample that installed from my standard Linux distribution CD.

`kpat`
(in X terminal) Patience card game. `sol` (fast) and `pysol` (slow but loaded) are two other choices. My favourite is: `sol −−variation=freecell&`

`xboing`
(in X terminal). Very nice, pin−ball game.

`xboard`
(in X terminal) Chess. Plays too well for me :(

`konquest &`
(in X terminal) Compete with your son in a conquest of a galaxy. Nice board game.

`kmines`
(in X terminal) Minesweeper.

`civserver`
`civclient`
(in X terminal) Startup server for the FreeCivilization game (first command). Afterwards, when the server is already running, start up the client (second comamand). Somebody else starts another client−−and you play. FreeCiv came on my RH7.0 CDs.

`fgfs`
"Flight Gear" flight simulator.

Go to Part 6: Essential Linux applications (proprietary or not)
Back to Top Page

Part 6: Some Essential Linux Applications

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section:

Intro. This part covers only application we frequently use or like. There are thousands of Linux programs. If you are unsatisfied with our lean choice, try: http://www.linuxapps.com/ or http://stommel.tamu.edu/~baum/linuxlist/linuxlist/linuxlist.html or http://www.boutell.com/lsm/ or http://www.linuxlinks.com/Software/

6.1 Word processing

6.1.1 OpenOffice.org /StarOffice Suite

OpenOffice is a complete office suite: word processor, spreadsheet, presentation program, drawing program, graphing module, and editor for mathematical equations. It sports the best, and the most feature−packed word processor and spreadsheet for Linux. Highly recommended. OpenOffice is included on the more recent Linux distribution CDs (Nov.2002). The latest version can be downloaded (free) from http://www.openoffice.org/ (large, ~70 MB download, probably not practical with a modem). The current (non−developer) version of OpenOffice is 1.0.1 (Oct. 2002). OpenOffice is cross−platform: it runs on Linux, MS Windows, Solaris, and Mac OS X, with full file−level compatibility.

Brief history. StarOffice used to be a commercial program ("Star Division", Germany). It was purchased by Sun Microsystems and the source code donated to the open source community under General Public Licence (Aug.2000). It is being rapidly developed by programmers many of whom are still associated with / paid by Sun. The open−source version is called "OpenOffice.org". Sun occassionally releases its own product based on a recent stable built of OpenOffice and calls it "StarOffice". Thus "OpenOffice.org" and "StarOffice" are basically the same products, with some (minor) feature additions in StarOffice. OpenOffice is officially called OpenOffice.org because of some trademark issues.

Description. OpenOffice looks and acts very much like MS Office for Windows. This includes richness of features, large size (requires considerable amount of disk space, memory, and processor speed), and well−burried features (may require some careful mouse−clicking to access some items). OpenOffice may not be worth your trouble without at least 64 MB of physical memory; the more memory the better. Open Office is very stable, although it sometimes displays weird artefacts ("ghosts") on my screen. It has a good file−level compatibility with MS Office: read and write MS Word, MS Excel and MS PowerPoint file formats. Natively, it uses a ground−breaking xml open file format: the text and pictures are zipped together into one file. When I unzip the file (`unzip my_file.sxv`), I can extract the original pictures−−something MS Office cannot possibly do (with sometimes serious consequences for document management).

OpenOffice does not look as "sexy" as some other Linux office alternatives. Yet, it is a real productivity workhorse and its polish is rapidly improving. In brief, we highly recommend StarOffice/OpenOffice to cover even demanding office needs. Feature−for−feature,

it matches almost anything found in MS Word or MS Excel, and adds some extras (long missing in MS Office).

Best of all, OpenOffice sports an open and beautifully designed file format which is rapidly becoming a standard (only unimaginative or corrupt decisions makers would insist on storing their data in a file format that can be read exclusively by a product of one company). This file format is suitable for serious uses because it can be parsed with third−party tools.

Installation. The installation of OpenOffice/StarOffice can be confusing. It goes like this:

− Make sure you have enough hard drive space. To check the space use the `df` ("disk free") command:

```
df −h
```

This displays a report on the used and available hard drive space in a human−legible form (option −h). At minimum, you need some 350 MB of free disk space (of which, ~100 MB you can release after installation).

− Decompress the downloaded file. I did it as root in the `/usr/local` directory for "local server" installation, but you may also choose /home/your_login for "personal" installation:

```
cd /usr/local
tar −xvf StarOffice5.2.tgz
```

Substitute the filename "StarOffice5.2.tgz" with the name of the file you downloaded.

− As root, run the setup program for a "local server" with the DOS−style `/net` switch:

```
cd /usr/local/OpenOffice641
./setup /net
```

[Without the "/net" or "−net" switch, OpenOffice will perform a personal installation (into your home directory!), and then only one user will be able to run it, plus your home directory will be clogged.]

− After this "network" installation, each user has to perform her own installation to put some personal files into their "home" directories. This is done by running (as a user, without the `/net` switch):

```
cd /usr/local/OpenOffice641
./setup
```

To run any of the OpenOffice component from the command line (in X terminal), I could use:

`ooffice&`	(word processor)
`oowriter&`	(same as above)
`oocalc&`	(spreadsheet)
`oodraw&`	(vector drawing program)
`ooimpress&`	(presentation program similar to MS PowerPoint)
`oosetup&`	(installation program)
`oopadmin&`	(printer administration utility)
`oomath&`	(equation editor, it is not typical to run it stand−alone)

Hints. ooofice comes with a extensive, contex−sensitve help (press <F1>). Here, we are going to collect some quick reference on using oowriter (just started Nov.2002).

- Consider using "styles" for formatting documents. Unless you are writing a very simple document, avoid "physical formatting". Press <F11> for a full style list. Inspect the drop box on the toolbar (left−hand side) for a list of paragraph styles used in the current document.

- The styles are divided into the following groups:
  - ♦ Character styles (applies to a letter or a group of letters)
  - ♦ Paragraph styles (a paragraph extends from <Enter> to <Enter>)
  - ♦ Frame styles (a frame a box containing text, graph, picture, etc.)
  - ♦ Page styles
  - ♦ Numbering styles (for bullet and numbered lists)

  The most important are the paragraph styles. I use them to format chapter headings, captions, table headings, etc. To apply a style to a paragraph, I place the cursor in the paragraph to be modified, and then double−click on the

name of the style in the "Stylelist". To modify a style (or create a new one), I use the menu "Format"−"Style−"Catalog".

- For numbered chapter headings, I set the numbering system under the menu "Tools"−"Outline Numbering".
- To insert a caption (for a table, figure, etc.), I position the cursor over the table (figure, etc) and use the menu "Insert"−"Caption". I can modify the caption numbering level with the button "Options" in the dialog box. It can also be modified later in the dialog box which appears when I click on the caption number.
- To update all the fields (numbers for heading, captions, table of contents, etc.), I press <Ctrl>a and then <F9>. This select the entire document and then updates the fields. Alternatively, I can use the menu "Tools"−"Update"−"Update All".
- To display an outline of the document, I press <F5>.
- To modify a default document template, make a "default" document and save it as "Default" ("File"−"Templates"−"Save").
- To spell−as−you−type, enable it under menu "Tools"−SpellCheck"−"AutoSpellCheck". To spellcheck of a word, position a cursor over it and press <F7>.
- For "type−ahead" autocompletion, I may press <Enter> to accept the selection when it appears.
- Cells in tables can be split. (If you are coming from MS Word, you can now start using split cells again.)
- Pressing <F2> produces a "formula bar". In tables, you can calculate results based on the numbers in other cells. The status bar shows the cell references. Outside of tables, one can use the formula bar to make simple "on−the−spot" calculations.
- You can insert a pre−formated and numbered "dummy equation" into the document by typing fn at a beginning of a paragraph and pressing <F3>. (This uses the "auto text" feature of the word processor.)
- Mail marging is described in http://documentation.openoffice.org/HOW_TO/word_processing/writer2_EN.html

Some useful keyboard shortcuts (most apply across the entire OpenOffice, not only the wordprocessor):

<Ctrl>x Cut	<F1> Help	<Ctrl><Space> Non−breaking space
<Ctrl>c Copy	<F2> Formula bar	<Ctrl><Shift>− Non−breaking hyphen
<Ctrl>v Paste	<F3> Autotext completion	<Ctrl><Enter> Manual (hard) page−break
<Ctrl>a Select All	<F4> Data source view on/off	<Shift><Enter> Line−break without paragraph change
<Ctrl>f Find (and replace)	<Shift><F4> Toggle asolute−relative references in a spreadsheet formula.	<Ctrl><Shift><Enter> Manual column break (in multi−columnar document)
<Ctrl>z Undo	<F5> Navigator on/off	<Insert> Insert/overwrite mode on/off
<Ctrl><Shift>p Superscript	<F7> Spellcheck	<Home> Go to beginning of the line
<Ctrl><Shift>b Subscript	<F9> Update fields (or recalculate spreadsheet)	<End> Go to end of the line
	<F11> Stylist on/off	<Ctrl><Home> Go to beginning of the document
	<F12> Numbering on/off	<Ctrl><End> Go to end of the document

Selection with mouse:
<LeftMouseButton> Select text, cells, etc.
<Shift><LeftMouseButton> Extend the current selection.

Dragging with Mouse:
`<LeftMouseButton>` Drag the selection to move it.
`<Ctrl><LeftMouseButton>` Copy the selection to the dragged location.

Some codes used inside equations:

Element type	Example
Fractions	1 over {2+3} {a + b} over {d + e}
Superscripts	a^2 + b^c = c^2 a^n a^m = a^{n+m}
Subscripts	a_1, a_2, ... a_n K_r=K_0 e^{−E_a over {kT}}
Roots	sqrt {a+b} nroot 3 {a+b}
Greek letters	%alpha %beta %gamma %delta %epsilon %varepsilon %zeta %eta %theta %vartheta %iota %kappa %lambda %mu %nu %xi %omicron %pi %varpi%rho %varrho %sigma %varsigma %tau %upsilon %phi %varphi %chi %psi %omega newline %ALPHA %BETA %GAMMA %DELTA %EPSILON %ZETA %ETA %THETA %IOTA %KAPPA %LAMBDA %MU %NU %XI %OMICRON %PI %RHO %SIGMA %TAU %UPSILON %PHI %CHI %PSI %OMEGA
Common relationships	= approx equiv def sim simeq prop <> < <= << > >= >> <> dlarrow drarrow dlrarrow towards parallel ortho leslant geslant transl transr
Common operators	+ − +− −+ cdot times and or in notin
Arrows	lefarrow rightarrow uparrow downarrow
Other common symbols	infinity emptyset
Sum and product	sum X_n sum from 1 to n X_n prod X_n
Integrals and derivatives	int f(x) dx int from 1 to 2 f(x) dx {partial x} over {partial t}

6.1.2 abiword

(type `abiword` or `AbiWord` in an X−terminal) AbiWord (http://www.abisource.com). It is a good light−weight wordprocessor. Really worth trying for simple word processing needs. Although still fairly light on features, it is quite useful to me, e.g. it supports spelling−as−you−type without the overhead of StarOffice. It is under heavy development and both versions for Linux an MS Windows are available.

6.1.3 kword

kword is still in developement and we would not recommend using it for anything important−−it may crash. However, it looks like the coolest office application of the three main (GPL) office suites. It is frame−based, like "framemaker" (hearsay, never used framemaker) which makes it easy to use and powerful for desktop publishing. Pretty feature−rich (certainly more features than in abiword).

To run `kword` in another language, I can do something like this (in X terminal, runs kword in the Dutch language):

```
exec sh −c "KDE_LANG=nl kword" &
```

Here is a list of some useful "standard" keyboard shortcuts. These shortcuts are used inside any kde−based applications (where applicable):
`<Alt>` Access the top menu.
`<Alt><a character>` Quick access to the items in the top menu. The *character* is the underlined letter in the menu name. For

example (for the English−language menus): <Alt>f −− "File" menu; <Alt>e −− "Edit" menu; <Alt>v −− "View" menu; <Alt>i −− "Insert" menu. <Alt>o −− "Format" menu.
<Ctrl>x Cut
<Ctrl>c Copy
<Ctrl>v Paste
<Ctrl>a Select All

6.1.4 lyx and latex

(Type lyx in an X−windows terminal). lyx is a front end (WYSIWYG, running under X−Windows) for Latex. [There is also Klyx, which is a "K−desktop" variant of Lyx, but it is not updated any more.] Latex has for years been the heavy−duty document preparation and typesetting program, particularly popular in academia because it is good with equations, can handle very large documents, etc.

The good news is that even if you do not know what Latex is, you may still be able to use lyx. Think of lyx as a word processor, although its philosophy is different from that of other popular word processors, and therefore it may require an adjustment of your mindset. Latex (and lyx) philosophy is to type in the text, define the "styles" and leave the formatting to the typesetting program. This means you never adjust the spacing (between words, sentences, paragraphs, chapter, etc.) manually. When done with typing of your document, you "compile" your text to create a device independent file ("*.dvi"). The *.dvi file can be viewed using a dvi viewer and printed. The quality of the output is usually outstanding, but its creation process is typically somewhat more frustrating than using a regular word processor.

The strength of Latex is the excellent quality of the printouts, its capability to cope with long, complex documents (technical books, math, etc.), availability of all foreign characters and even rarely used symbols, its portability across many different platforms, and the popularity of the file format. Its weakness is the relative usage complexity.

lyx is free and it is included on your Mandrake or RedHat CD for you to try. As almost any piece of Linux software, you can also download it from Linuxberg: http://idirect.linuxberg.com/kdehtml/off_word.html or any other fine Linux software depository on the Internet.

If instead of easier lyx, you wanted to try straight, hard−core Latex, here is some intro to get you started:

- Use your favorite plain−text editor to create a Latex document, spell check it, etc., save the text file with the extension "*.tex". Read on to see my sample Latex document.
- Invoke Latex to "compile" the text file into a "*.dvi" ("device independent") file by typing on the command line:

  ```
  latex my_latex_file.tex
  ```

- Print the "my_latex_file.dvi" file which was created by the previous command by invoking the dvi to postscript utility, that on default send the output to the lpr printer:

  ```
  dvips my_file.dvi
  ```

- You can also save the output to postscript file by typing:

  ```
  dvips −o output_file.ps my_file.dvi
  ```

  The option −o introduces the output file.

- You can also create a pfd file using

  ```
  dvipdf my_file.dvi output_file.pdf
  ```
  or
  ```
  ps2pdf my_file.ps my_file.pdf
  ```

- You can view any of the files (dvi, ps, or pdf), for example using (in X terminal):

  ```
  kdvi my_file.dvi&
  ```
  or
  ```
  kghostview my_file.pdf&
  ```

Here is my sample Latex file:

```
% Any line starting with "%" is a comment.
% "\" (backslash) is a special Latex character which introduces a Latex
% command.
\documentclass[10pt]{article}
\begin{document}
% Three commands are present in every Latex document. Two of them are
% above and one at the very end of this sample document.
This is a simple document to try \LaTeX. Use your favorite plain text
editor to type in your text. See how the command \LaTeX produces the
\LaTeX logo. Here is the end of the first paragraph.
Here starts the second paragraph (use one or more empty lines in your
input file to introduce a new paragraph).
The document class of this sample is ``article'' and it is defined at the
very beginning of the document. Other popular classes are ``report'',
``book'' and ``letter''.
Please note that the double quote is hardly ever used, utilize
two ` to begin a quote and two ' to close it. This nicely formats the
opening and closing quotes.
Here are different typefaces:
{\rm This is also roman typeface. It is the default typeface.}
{\bf This is bold typeface. }
{\em This is emphasize (italic) typeface.}
{\sl This is slanted typeface, which is different from the italic.}
{\tt This is typewriter typeface.}
{\sf This is sans serif typeface.}
{\sc This is small caps style.}
You can itemize things:
\begin{itemize}
\item one
\item two
\item three
\end{itemize}
You can also enumerate things:
\begin{enumerate}
\item one
\item two
\item three
\end{enumerate}
Try some foreign letters and symbols:
\aa \AA \o \O \l \L \ss \ae \AE \oe \OE \pounds \copyright \dag \ddag \S
\P. There are also three dashes of different length: - -- ---.
Try some accents over the letter ``a'': \'{a} \`{a} \"{a} \^{a} \~{a}
\={a} \.{a} \b{a} \c{a} \d{a} \H{a} \t{a} \u{a} \v{a}. Other letters can
be accented in a similar way.
The pair of ``\$'' marks a math context. Many special characters are
available only in the ``math'' context. For example, try the Greek
alphabet:
Small: $ \alpha \beta \gamma \delta \epsilon \varepsilon \zeta \eta
\theta \vartheta \iota \kappa \lambda \mu \nu \xi o \pi \varpi
\rho \varrho \sigma \varsigma \tau \upsilon \phi \varphi
\chi \psi \omega $
Capital: $ A B \Gamma \Delta E Z H \Theta I K \Lambda M \Xi \Pi P
\Sigma T \Upsilon \Phi X \Psi \Omega $
Try some equations: $ x^{y+1} + \sqrt{p \times q}=z_{try_subscripts} $
\begin{center}
$ \frac{x \times y}{x/2+1}=\frac{1}{3} $
\end{center}
\LaTeX math commands are very similar to those in the old ``Word Perfect''
equation editor.
Use the verbatim mode to print the 10 special symbols which normally have
special meaning in \LaTeX: \verb|%${}_#&^~\|. The special symbols must be
contained between any two identical characters which in the example above
is |. Most of these special symbols can also be printed by preceding the
character with a backslash: \% \$ \{ \} \_ \# \& \^.
```

```
% This command ends the document (this is the third one that *must* be
% present in every document).
\end{document}
```

6.1.5 WordNet (dictionary / thesaurus /synonym / antonym finder)

As a dictionary / thesaurus, I use WordNet (type `wn` in text terminal). It did not come on RH 7.0 CDs, so I had to download (10 MB) and install it myself. Really worth it. Try: http://www.cogsci.princeton.edu/~wn/. RedHat 8.0 came with `wn` pre−installed, and a GUI frontend to it:

```
ktheasurus&
```

6.2 Spreadsheet

oocalc&

I currently use the good spreadsheet called calc, which is part of OpenOffice.org. It can be run by clicking an appropriate menu item from your favourite desktop ("K"−−"Office"−−"OpenOffice.org Calc") or typing in an X terminal:

```
oocalc&
```

I am a very heavy spreadsheet user, so here are some other promising programs I keep my eyes on. In my opinion, Linux does not have an excellent spreadsheet program yet, but oocalc can do a lot.

Users can even define their own function in oocalc. For example, i can write in the editor ("Tools"−"Macros"−"Edit") such a function:

```
REM  *****  BASIC  *****
REM This function calculates an area of a donut with radii r1 and r2
Function my_function(r1,r2)
REM return the value using the variable called like the function
my_function=abs(pi()*r1^2 − pi()*r2^2)
End function
```

and then use it from my spreadsheet using something like: `=my_function(2;3)`

Currently (Nov.2002, ver.1.01) oocalc supports 256 columns (A .. IV), 32000 rows (1 .. 32000), and up to 256 sheets ("sheet1", "sheet2", etc, named dynamically and be re−named). Work is in progress to increase the number of rows and columns. It has hundreds of build−in functions (covering compatiblity with anything found in MS Excel).

The user interface is sometimes awkward. For example, adding a new data series to a chart requires highlighting a spreadsheet range, and then dragging and dropping it onto the chart with a mouse. Still, the chart component supports (for XY graphs) two Y axes, two X axes, good selection of line types, bitmaps for data points, error bars, regression fits, etc. Really powerful if you learn how to use it−−to my taste too much of careful mouse clicking is required. I would really enjoy a giant dialog box with all the options for the chart typed into it for me to modify when required.

6.2.1 `gnumeric`

(in X terminal) Nice spreadsheet, part of GNOME, included with standard RH distributions (RH6.0 or higher). Although still fairly incomplete, it is quite usable. Gnumeric is under heavy development and definitely has the potential to become really great in the near future−−it already has a lot of built−in functions, but its printing is unreliable−−major pain.

6.2.2 `kspread`

Kspread is another highly promising spreadsheet. It is part of the KDE ("koffice") since KDE2. Still not ready for a serious use. Both gnumeric's and kspread's file format is xml (the already standard, next−generation, "enhanced html"). This file format is definitely good news if you ever experienced problems with the incompatibility of the MS−Windows−based spreadsheet file formats. Like html, xml is legible to humans. The spreadsheet files are compressed, so to view the contents on the console, I can do something like: `zless my_gnumeric_file.gnumeric`

A comparison of the file sizes for a simple spreadsheet book (containing just one formula, copied 10 000 times) demonstrates that the human−legible file format is not necessary a liablity, and that MS Excel is bloated:

```
Gunmeric 1.09                    27,136
 OOCalc 1.01                      57,756
```

```
kspread 1.2RC1                  90,560
 MS Excel 2000                  549,888
 QuattroPro 9 for MS Windows    155,648
```

6.3 Databases

If you are a database person, you will be pleased to see that Linux is very well covered in this area.
postgreSQL is a high−powered database available on Mandrake and RH CD (free, unrestrictive BSD license).

mySQL http://www.MySQL.com/ GPL database, simpler and easier than postrgreSQL, yet very very capable. Favourite among many database developers. Like postgreSQL, mySQL is not meant to be just a personal database, and it may be hard to use it as such.

There are also commercial databases which are free for personal use, e.g. **Sybase** for Linux (http://www.sybase.com:80/sqlserver/linux/aselinux_install.html) and **Interbase** (http://www.interbase.com/downloads/products.html). There is also **Oracle** for Linux: http://platforms.oracle.com/linux/index_lin.htm. For an Oracle−Linux howto, see: http://jordan.fortwayne.com/oracle/index.html

6.4 CAD

QCAD (GPL): http://www.qcad.org (simple but useful for small drawing or learning)
OCTREE (free for non−commercial applications): http://www.octree.de/html/frames/eng/f_octree.htm
VariCAD (proprietary commercial, a free 15−day trial version available for download, geared towards mechanica design): http://www.varicad.com/
VARKON (LGPL): http://www.varkon.com/
Microstation (proprietary): http://www.microstation.com/academic/products/linux.htm−−the academic edition of Microstation includes the Linux version of their excellent CAD system (better than AutoCad).

There is also something called "LinuxCAD" but it appears to be a rip−off (not recommended).

6.5 Web browsers: Mozilla, Konqueror, Galeon, and Lynx

All newer linux distributions (Nov. 2002) include as the main web browser "mozilla". Type in an X−terminal:

```
mozilla&
```

This is a state−of−art browser and one cannot wish for much more (except maybe some speed). It is a decendant of the famous Netscape.

Other choices for an Internet browser are: KDE−based `konqueror` and `galeon`. I use `galeon`.

If your ISP connection is really slow, you may prefer a text−based browser:

```
lynx
```

Don't expect it to look as fancy a GUI−based browser−it is text−mode based. The good think about it is that it is always works and is fast. Great for a quick look at an html file.

On an older Linux, you probably have installed a 4.xx version of Netscape ("a tried and true browser"). To run it, try (in X−terminal):

```
netscape&
```

To compose html pages (including this guide), I use mozilla (WYSiWYG view or code view), or netscape (WYSiWYG view or code view), or WebMaker (code view).

6.6 Writing CD−Rs: cdrecord and cdparanoia

Disclaimer: Copying copyrighted material is illegal. Do NOT use the instructions below for anything illegal.

INTRO

Writing CDs used to be tricky, but these days I simply use (as root, in X terminal):

```
xcdroast &
```

If it works for you as it works for me, you don't need to read any further.

Perhaps still useful if xcdroast does not work on your system, here are the steps I once followed to write CDs using command−line tools. Please note that `xcdroast` is just a graphical front−end to the command line tools described below, so it will not work if the commands do not work. My only complaint s were that my low−cost, no−name "12x 8x 32x" CD−RW writes at top speed of 12x (for CD−R) but reads only at the speed of 2x (instead of 8x forCD−RW), and rips audio at 1x, no matter what I do.

SETUP

o All setup has to be done as root. Newer distributions (e.g. RedHat 8.0) may require no setup at all.

o Check your boot files if they pass a parameter to the kernel with the information that you have the ide−scsi drive: "hdb=ide−scsi". If required, add to the file

`/etc/lilo.conf` or `/boot/grub/grub.conf` (depending which boot loader your system uses):

the option "hdb−ide−scsi" so that the line looks like this:

```
append="hdb=ide−scsi"    #(for /etc/lilo.conf, at the end of the Linux "image" section)
kernel /boot/vmlinuz−2.4.7−10 ro root=/dev/hda6 hdb=ide−scsi      #(for /boot/grub/grub.conf)
```

Adjust the line above if your CD writer is not "hdb" (second drive on the first IDE interface). It makes your IDE−ATAPI CD−W(R) to be seen on your Linux system as a SCSI device. (It is not really a SCSI device, it is an IDE device, it just pretends to be SCSI.) Run `lilo` after making any changes to `/etc/lilo.conf` . Grub does not need re−running.

o Add the loop devices to the `/dev/` directory if it is not present. This is not a obligatory, but a nice feature if you plan creating your own data CDs. The loop device will let you mount a CD image file (as if it was a already a filesystem) to inspect its content. The loop devices don't exist on my hard drive after Linux RedHat installation, so I create them using:

```
cd /dev/
./MAKEDEV loop
```

o Make sure that appropriate modules are loaded to the kernel using:

```
lsmod
```

If required, add these two lines at the end of the file `/etc/rc.d/rc.local` so that the needed kernel modules are automatically loaded on system startup:

```
/sbin/insmod ide−scsi
/sbin/insmod loop
```

These two kernel modules are needed for SCSI emulation of IDE drives and to support the loop devices, respectively.

o Check, create or modify the device `/dev/cdrom` so it now points to the correct device , most likely:

```
ls −l /dev/cdrom
ln −s /dev/scd0 /dev/cdrom  #(if required)
```

You may need to do this because "/dev/cdrom" pointed to an IDE device (probably `/dev/hdb`) but now this changes since your CD−R is going to be in SCSI emulation mode.

There is also /dev/cdwriter that you may want to point to /dev/cdrom although it is not necessary on a single CD drive system:

```
ln −s /dev/cdwriter /dev/cdrom
```

o Reboot so that the changes to `/etc/lilo.conf` take effect. Check if your CD−R(W) still works properly for normal reading.

o Check if the program cdrecord is installed, e.g.,:

```
cdrecord −scanbus
```

If it is not installed, download the program "cdrecord" from your favorite Linux software repository (e.g., http://idirect.linuxberg.com/). Then install the source code, compile it, install the program, and make symbolic links so that the executable are easy to run (the installation would be much easier if you found a binary *.rpm file):

```
cd /usr/local
tar −xvzf /the_path_to_which_you_downloaded/cdrecord−1.6.1.tar.gz
ls
cd cdrecord−1.6.1
make
make install
ls /opt/schily/bin/
ln −s /opt/schily/bin/* /usr/local/
```

The program `cdrecord` is a spartan, command line utility for writing CD. There are several GUI front ends to it, but they will be useless if the underlying cdrecord does not work properly. My advice: use command line for some time−−you get to understand how things work, get flexibility, and reliable results. Then you can install GUI front ends to make CD covers, and make things easier for Windows−educated users on your system.

o See if your cdwriter is recognized. If it is, it should now show in the output from this command:

```
cdrecord −scanbus
```

CREATING DATA CDs

o Create a CD image containing your data:

```
mkisofs −r −o cd_image input_data_directory
```

This makes an International Standard Organization (ISO) standard 9660−type filesystem containing the files from `input_data_directory`, but writes the filesystem to an ordinary file on the hard drive. This output file is an "image" of the new CD which I am creating. The option "−o" indicates that the parameter that follows is the output filename of this image. The option "−r" enables "Rock Ridge" extensions to the ISO protocol so that file attributes are saved, and it sets the file permissions so all the files on the CD are publicly readable (can be read by all user, not only the file owner). The filenames are abbreviated to the "8.3" DOS−type length but, since Linux supports so called "Rock Ridge" extensions to ISO9660, it also writes the full names and all the file permissions as well−−this way the new filesystem is portable across all popular operating systems (DOS, MS Windows, Linux, UNIX, etc)−−really convenient to the user.

The input data directory can be assembled from differenet directories and files from all−over your filesystem using symbolic links (saves harddrive space because the data is not copied), but if you do it you probably want to tell `mksiofs` to follow symbolic links using the option `−f`:

```
mkisofs −r −f −o cd_image input_data_directory_containing_symlinks
```

o You may want to inspect the CD image file by mounting it through the loop device:

```
mount −t iso9660 /dev/loop0 cd_image /mnt/cdrom
[now the content of the file should appear in /mnt/cdrom]
cd /mnt/cdrom
[inspect the file mounted through the loop device]
```

When done with inspection, change your working directory away from the mountpoint and unmount the file:

```
cd
umount /mnt/cdrom
```

o If everything worked, you may burn your data CD:

```
cdrecord −v speed=8 dev=0,0,0 −data cd_image
```

The first number in "dev=" stand for the scsi bus number (the first one is 0, second bus is 1, ...), device id on the scsi bus (between 0 and 7), and the scsi lun number (always 0) respectively. You must customize them: the first two numbers can be read in the output from `cdrecord −scanbus`, the third number is always 0. Make sure to use the correct numbers or you may write to a wrong drive and corrupt your data.

The timing of writing to CD−Rs is very important, or an error may occur (the laser cannot be switched on and off at will). Therefore avoid doing intensive tasks during creating a CD, e.g. don't create or erase large files on the hard drive. My old system (RH6.2) will not permit me to start new tasks when using `cdrecord` .

In a similar way, I can burn a CD from an ISO CD image downloaded from the Internet. One source (a Debian FAQ) recommends the following command (as root) to burn the image "binary−i386−1.iso" to a CD:

```
nice −−18 cdrecord −eject −v speed=2 dev=0,6,0 −data −pad binary−i386−1.iso
```

This assigns a very high priority ("niceness" of minus 18) to the CD burning task (thus minimizing the possiblity of an error).

CREATING AUDIO CDs

o Audio tracks have to be in files of *.cdr (I guess it is the same as *.cdda.raw), *.wav (wave), or *.au format before you can write them to a CD.

o The utility `sox` converts between the various audio file formats (`sox` understands quite a few of them). For example, I can convert a .wav file to a .cdr file:

```
sox my_file.wav my_file.cdr
```

You don't need to do the conversions manually − `cdrecord` supports *.wav and *.au directly (it does a conversion from *.wav or *.au to *.cdr "on the fly"). This is very convenient because audio files tend to be large.

o Audio CDs don't contain a filesystem, they store "raw data". This means that you cannot mount an audio CD. Also, each track is written separately, i.e., as if it was a different "partition" on the CD.

o To read audio tracks from an audio CD and write them to suitable files on your hard drive (typical format is *.raw or *.wav) , you need a "cd ripper". A popular CD ripper is "cdparanoia".

> If cdparanoia is not installed, download it. The installation from source goes as follows (I use the autocompletion <Tab> shortcut when typing the long filenames):
>
> ```
> cd /usr/local
> tar −xvzf /the_path_to_which_you_downloaded/cdparanoia−III−alpha9.6.src.tgz
> ls
> cd cdparanoia−III−alpha9.6
> ./configure
> make
> make install
> ```

To rip the first track from an audio CD, I can use:

```
cdparanoia 1
```

which will put the first track from the CD into the wave file "cdda.wav" in the current directory.

To rip tracks 1 to 2 from an audio CD to a "raw" file format, I can use:

```
cdparanoia −B −p "1−2"
```

The option −B specifies to use a "batch" mode, so that each track is put into a separate file (this is probably what you want, otherwise all tracks would be placed in one output file). The "−p" option specifies output in raw format. The files are named `track1.cdda.raw` and `track2.cdda.raw` .
To rip all tracks from an audio CD, each track to a separate *.wav file, while forcing reading speed 4x, I can use:

```
cdparanoia −S 4 −B "1−"
```

Make sure you have sufficient free space on your hard drive. You can use use the space on your DOS partition (if you have dual boot).

o To write suitable audio files to a CD−R(W), I can use:

```
cdrecord −v speed=8 dev=0,0,0 −pad −dao −audio track*
```

o Some audio CDs do not have gaps between individual audio tracks. The easiest way to make a copy of such an audio cd, is to use the utility "cdrdao".

To copy a disk to the file "data.bin" (on my harddrive), and the table of contents to the file "toc−file.toc", I can use this command:

```
cdrdao read−cd −−device 0,0,0 −−buffers 64 −−driver generic−mmc−raw −−read−raw
toc−file.toc
```

To burn the CD from the files I just created, I can use:

```
cdrdao write −−device 0,0,0 −−buffers 64 −−driver generic−mmc −−speed 12 toc−file.toc
```

o Note on re−writeable CDs. Some stereos will not play re−writeable CDs because of the size of the pits on the CDs. For example, my home stereo (JVC) cannot read re−writable CDs (CD−RW) at all, although it will read write−once disks (CD−Rs). Therefore, re−writable CDs may be good to store data but not audio (unless I plan to play them exclusively on my computer).

CREATING MIXED−MODE CDs

Mixed−mode CDs (meaning CDs which contain both data and audio, often game CDs) are not a problems, e.g.:

`mount −t iso9660 /dev/cdrom /mnt/cdrom` (mount the data part of the mixed−mode CD)
`mkisofs −r −o cd_image /mnt/cdrom` (make an ISO filesystem from the data on the CD).
`umount /mnt/cdrom` (unmount the CD)
`cdparanoia −B "2−"` (rip the content of all audio tracks on the CD, except the first track since it is data)
`cdrecord −v speed=2 dev=0,0,0 −data cd_image −audio track*` (write the data and audio files, piece by piece)

MAKING A COPY OF THE ENTIRE CD

Most CDs can be copied by first copying all data (for data CDs) or all tracks (for audio CDs) onto the hard drive as described before, but some CDs cannot.

For example, these kinds of data (not audio) CDs need to be treated differently: bootable CDs (like Linux installation CD), CDs that require the label, disk with errors, etc. For data CDs, I use these commands to make an exact copy:

```
dd if=/dev/cdrom of=cd_image
cdrecord −v speed=2 dev=1,0,0 −data cd_image
```

The `dd` command copies the input file (if), which in this case is the device `/dev/cdrom` to the output file (of) which in this example is a file called `cd_image` (on the hard drive in the current working directory). The second command copies the file `cd_image` that was created by the `dd` command onto an empty CD.
For data disk with error, you might want to try:

```
dd conv=noerror,notrunc if=/dev/cdrom of=cd_image
cdrecord −v speed=2 dev=1,0,0 −data cd_image
```

The option "conv=noerror,notrunc" specifies that the potential read errors are to be ignored, and files not truncated on error.
For audio CDs, I use these command to make a copy:

`cdparanoia −B "1−"` (rip the content of all audio tracks on the CD, from track 1 on. The tracks are saved into files in the current directory and named: track01.cdda.wav, track02.cdda.wav, etc.)

`cdrecord −v speed=2 dev=1,0,0 −audio track*` (write all the audio files to the CD, one by one. The tracks are separated by a 2 s gap).

To copy an audio CDS in the most acurate way, `man cdrecord` recommends doing this:

```
cdda2wav −v255 −D2,0 −B −Owav
cdrecord −v dev=2,0 −dao −useinfo  *.wav
```

To make an exact copy of mixed mode CDs:

`dd if=/dev/cdrom of=cd_image` (The `dd` command will output an error message when the the data has ended and audio started. This is expected and ok).
`cdparanoia −B "2−"` (rip the content of all audio tracks on the CD, except the first track since it is data)

`cdrecord −v speed=2 dev=1,0,0 −data cd_image −audio track*` (Write the data and subsequent audio files, piece by piece.)

RE−WRITABLE CDs

Re−writable CDs (CD−RW) are used the same way as regular write−once CDs (CD−R), but you have to blank re−writable disks before you will be able to re−use them, e.g.:

```
cdrecord −v speed=2 dev=1,0,0 blank=fast
```

To see other (more thorough and slower) options for blanking, use:

```
cdrecord blank=help
```

For example this thorough blanking can take 0.5 hour on my system, but is not really necessary unless the old data is confidential:

```
cdrecord dev=0,0,0 blank=disk
```

Again, older stereos often will not play CD−Rs.

SIMPLIFYING LONG COMMANDS WITH AN ALIASES

To simplify writing long commands required by `cdrecord` (or `cdrdao`), I may want to define a global alias by placing the following line in the file `/etc/bashrc`:
`alias cdrecord="cdrecord −v speed=2 dev=1,0,0"`
Re−login for the changes in `/etc/bashrc` to take effect. After creating this alias, I can record a CD using the following shortened command (no need to specify the CD writer speed and device name all the time):
`cdrecord −audio track*`

6.7 Automating creation of graphs with gnuplot

`gnuplot` is good for automating generation of graphs for numerical data and/or mathematical functions. For "interactive" generation of graphs, I prefer any spreadsheet. As old−fashioned as gnuplot may look, it can be quite handy if you want to periodically re−generate a graph or visualize (for inspection) massive amounts of data from a graph "template". `gnuplot` is flexible (many options available, including 3d plots) but one needs to take your time to learn it. Setting up a complex graph can take me 2 hours (but it's ok, if the graph is to be re−used many times over). The best help is to start `gnuplot`, and on the "gnuplot>" prompt, type "help". gnuplot is available for Linux and MS Windows.

My data sets are stored in text (ASCII, *.dat) files. My "graph templates" are stored in gnuplot "command" files (*.gnu). The output goes to a graphics file (*.png) which can be printed or imported to any word processor.

To generate a graph from an example gnuplot command file "make_graphs.gnu", I can do:
`gnuplot make_graphs.gnu`

To display the graph, I would do (in X terminal):
`display my_graph.png`

My example "make_graphs.gnu" that generates an x−y graph follows.

```
# Comment are introduced with the hash (#)
# Stamp the graph with the current date and time
set timestamp "%Y−%m−%dT%T%z"
# This sets the graph resolution (the default is 100)
set samples 600
# Save the plot to a *.png file (make it colour)
set output "my_plot.png"
set terminal png color
#interesting terminals: png, x11, postscript, postscript eps, hpgl
set title "My Graph"   # Graph title
set xlabel "Distance [m]"      # title of x1 axis (bottom)
set x2label "Distance [feet]"  # title of x2 axis (top)
set ylabel "sin meters"        # title of y1 axis (left)
set y2label "log feet"         # title y2 axis (right)
```

```
set xtics  # control major tic marks on the axis
set x2tics; set ytics; set y2tics #commands can be separated with semicolons
set mytics # control minor tics on the axis, here I add them to the y axis
set xrange [0:15]          # Range for display on the x1 axis
set x2range [0:15.0/0.305] #Expressions are ok. This one converts meters to feet.
set yrange [*:*]           # The "*" sets the range to auto
set y2range [*:*]          # Range for the y2 axis
set nologscale  # or "set logscale x1x2y1y2" #Control logscale, linear scale is the default
set nogrid      # or "set grid" #Control gridlines, no grid is the default
set key outside # or "set nokey" #Control legend and its positions: "top", "bottom", "left"
# The following line creates the plot with 4 graph series
plot sin(x) axes x1y1, log(x) axes x2y2, "data.dat" using 1:2, \
   "data.dat" using 1:3
# Long lines can be split with \
# The third series uses columns 1 and 2 from the file
# The fourth plots the 3 column agains the 1st column from the data file.
```

Goto part 7: Learning with Linux

Part 7: Learning with Linux

LINUX NEWBIE ADMINISTRATOR GUIDE

ver. 0.193 2002−12−14 by Stan, Peter and Marie Klimas

The latest version of this guide is available at http://sunsite.dk/linux−newbie.

Contents of this section:

Intro: This page contains commands that are not essential for newbies but can be fun and/or very productive if you take the time to learn them. Above all, they can be very educational−−linux is a great learning platform. There are thousands of free programs available on Linux, but some of them may be missing on your system, depending on your distribution. You can always find the hompage with google, and then download them.

7.1 Linux Advanced Text Processing Tools

`/usr/games/banner −w79 "Happy Birthday, Marie" > marie.txt`
Create an ascii "banner" with the width of 79 characters. The output is sent to file `marie.txt`. Funny, old−fashioned tool. Another utilty for asci text art is figlet. E.g. `figlet "Funny!"` produces this on in my terminal (I always use a fixed−size font to display ascii art):

```
 _____                        _
|  ___|   _ _ __  _ __  _   _| |
| |_ | | | | '_ \| '_ \| | | | |
|  _|| |_| | | | | | | | |_| |_|
|_|   \__,_|_| |_|_| |_|\__, (_)
                         |___/
```

`script`
Log my current session in the text terminal into a text file `typescript` (the default filename). The log finishes when I type `exit` or press `<Ctrl>d`.

`emacs&`
(in X−terminal) The emacs text editor. Advanced and sophisticated text editor. Seems for gurus only: "emacs is not just an editor, it is a way of living". Emacs surely seems rich or bloated, depending on your point of view. There are likely 3 versions of emacs installed on your system: (1) text−only: type `emacs` in a text (not X−windows) terminal (I avoid this like fire); (2) graphical−mode: type `emacs` in an X−windows terminal (fairly usable even for a newbie if you take some time to learn it); and (3) X−windows mode: type "xemacs" in an X−windows terminal.

`vi`
The famous (notorious?) "vi" text editor (definitely not recommended for newbies). To exit "vi" (no changes saved) use these five characters: `<ESC>:q!<Enter>` I use the "kate&" (under X) or "pico" (command line) or "nano" (command line) text editors and don't ever need vi (well, unless I have to unmount the /usr subsystem and modify/edit some configuration files, then vi is the only editor avialable). To be fair, modern Linux distributions use `vim` (="vi improved") in place of vi, and vim is somewhat better than the original vi. The GUI version of vi is also available (type `gvim` in an X terminal). Here is one response I have seen to the criticism of vi interface being not "intuitive": "The only intuitive interface is the nipple. The rest must be learned." (Well, so much for MS Windows being an "intuitive" interface.)

> Experts do like `vi`, but `vi` is definitely difficult unless you use it very often. Here is a non−newbie opinion on `vi` (http://linuxtoday.com/stories/16620.html):

> "I was first introduced to vi in 1988 and I hated it. I was a freshman in college... VI seemed archaic, complicated and unforgiving... It is now 12 years later and I love vi, in fact it is almost the only editor I use. Why the change? I actually learned to use vi... Now I see vi for what it really is, a powerful, full featured, and flexible editor..."

For your entertainment, you might want to try the even more ancient−looking line−editor `ed` (just type `ed` on the command line). Tools like these, however "inconvenient" in interactive use, can be very useful for automation of manipulation of files from within another program.

> Brief Introduction to vim (="visual editor improved") which is a modern Linux version of vi. The main reason why a newbie like myself ever needs vi is for rescue−−sometimes it is the only editor available. The most important thing to understand about vi is a "modal" editor, i.e., it has a few modes of operation between which user must switch. The quick reference is below, the 4 essential commands are in bold.
>
> The commands to switch modes:
>
> The key Enters the mode Remarks
> **<ESC> command mode (get back to the command mode from any editing mode)**
> **i "insert" editing mode (start inserting before the current position of the cursor)**
> DO NOT PRESS ANY OTHER KEYES IN THE COMMAND MODE. THERE ARE MORE COMMANDS AND MODES IN THE COMMAND MODE!
>
> Copying, cutting and pasting (in the command mode):
>
> v start highlighting text. Then, move the cursor to highlight text
> y copy highlighted text
> x cut highlighted text
> p paste text that has been cut/copied
>
> Saving and quitting (from the command mode):
>
> **:w write (=save)**
> :w filename write the contents to the file "filename"
> :x save and exit
> :q quit (it won't let you if changes not saved)
> **:q! quit discarding changes (you will not be prompted if changes not saved)**

`nano`
This is a brand new (March 2001) GNU replacement for pico. Works and looks like pico, but it is smaller, better, and licenced as expected for a decent piece of Linux software (i.e., General Public Licence, GPL). Not included with RH7.0 or MDK7.2, but expect it soon.

`khexedit`
(in X terminal) Simple hexadecimal editor. Another hexadecimal editor is `hexedit` (text based, less user friendly). Hex editors are used for editing binary (non−ASCII) files.

`diff` *`file1 file2 > patchfile`*
Compare contents of two files and list any differences. Save the output to the file `patchfile`.

`sdiff` *`file1 file2`*
Side−by−side comparison of two text files. Output goes to the "standard output" which normally is the screen.

`patch` *`file_to_patch patchfile`*
Apply the patch (a file produced by `diff`, which lists differences between two files) called `patchfile` to the file `file_to_patch`. If the patch was created using the previous command, I would use: `patch file1 patchfile` to change `file1` to `file2`.

`grep` filter
Search content of text files for matching patterns. It is definitely worth learning at least the basics of this command.

> A simple example. The command:
>
> ```
> cat * | grep my_word | more
> ```

will search all the files in the current working directory (except files starting with a dot) and print the lines which contain the string "my_word".

A shorter form to achieve the same may be:

```
grep my_word * |more
```

The patterns are specified using a powerful and standard notation called "regular expressions".

There is also a "recursive" version of grep called rgrep. This will search all the files in the current directory and all its subdirectories for my_word and print the names of the files and the matching line:

```
rgrep −r my_word . | more
```

Regular expressions (regexpr)
Regular experessions are used for "pattern" matching in search, replace, etc. They are often used with utilities (e.g., `grep`, `sed`) and programming languages (e.g., perl). The shell command `dir`, uses a slightly modifed flavour of regular expressions (the two main differences are noted below). This brief writeup includes almost all the features of standard regular expression−−regexpressions are not as complicated as they might seem at first. Definitely worth a closer look at.

In regular expressions, most characters just match themselves. So to search for string "peter", I would just use a searchstring "peter". The exceptions are so−called "special characters" ("metacharacters"), which have special meaning.

The regexpr special characters are: "\" (backslash), "." (dot), "*" (asterisk), "[" (bracket), "^" (caret, special only at the beginnig of a string), "$" (dollar sign, special only at the end of a string). A character terminating a pattern string is also special for this string.

The backslash, "\" is used as an "escape" character, i.e., to quote a subsequent special character.
Thus, "\\" searches for a backslash, "\." searches for a dot, "*" searches for the asterisk, "\[" searches for the bracket, "\^" searches for the caret even at the begining of the string, "\$" searches for the dollar sign even at the end of the string.

> Backslash followed by a regular (non−special) character may gain a special meaning. Thus, the symbols \< and \> match an empty string at the beginning and the end of a word, respectively. The symbol \b matches the empty string at the edge of a word, and \B matches the empty string provided it's not at the edge of a word.

The dot, ".", matches any single character. [The `dir` command uses "?" in this place.] Thus, "m.a" matches "mpa" and "mea" but not "ma" or "mppa".

Any string is matched by ".*" (dot and asterisk). [The `dir` command uses "*" instead.] In general, any pattern followed by "*" matches zero or more occurences of this pattern. Thus, "m*" matches zero or more occurances of "m". To search for one or more "m", I could use "mm*".

> The * is a repetition operator. Other repetition operators are used less often−−here is the full list:
>
*	the proceding item is to be matched zero or more times;
\+	the preceding item is to be matched one or more times;
\?	the preceding item is optional and matched at most once;
\{n}	the preceding item is to be matched exactly n times;
\{n,}	the preceding item is to be matched n or more times;
\{n,m}	the preceding item is to be matched at least n times, but not more than m times.

The caret, "^", means "the beginning of the line". So "^a" means "find a line starting with an "a".

The dollar sign, "$", means "the end of the line". So "a$" means "find a line ending with an "a".

> Example. This command searches the file `myfile` for lines starting with an "s" and ending with an "n", and prints them to the standard output (screen):
>
> ```
> cat myfile | grep '^s.*n$'
> ```

Any character terminating the pattern string is special, precede it with a backslash if you want to use it within this string.

The bracket, "[" introduces a set. Thus [abD] means: either a or b or D. [a−zA−C] means any character from a to z or from A to C.

Attention with some characters inside sets. Within a set, the only special characters are "[", "]", "−", and "^", and the combinations "[:", "[=", and "[.". The backslash is not special within a set.

Useful categories of characters are (as definded by the POSIX standard): [:upper:] =upper−case letters, [:lower:] =lower−case letters, [:alpha:] =alphabetic (letters) meaning upper+lower, [:digit:] =0 to 9, [:alnum:] =alphanumeric meaning alpha+digits, [:space:] =whitespace meaning <Space>+<Tab>+<Newline> and similar, [:graph:] =graphically printable characters except space, [:print:] =printable characters including space, [:punct:] =punctuation characters meaning graphical characters minus alpha and digits, [:cntrl:] =control characters meaning non−printable characters, [:xdigit:] = characters that are hexadecimal digits.

Example. This command scans the output of the `dir` command, and prints lines containing a capital letter followed by a digit:

```
dir −l | grep '[[:upper:]][[:digit:]]'
```

`tr`
(=translation). A filter useful to replace all instances of characters in a text file or "squeeze" the white space.

Example :

```
cat my_file | tr 1 2 > new_file
```

This command takes the content of the file `my_file`, pipes it to the translation utility `tr`, the `tr` utility replaces all instances of the character "1" with "2", the output from the process is directed to the file `new_file`.

`sed`
(=stream editor) I use sed to filter text files. The pattern to match is typically included between a pair of slashes // and quoted.

For example, to print lines containing the string "1024", I may use:

```
cat filename | sed −n '/1024/p'
```

Here, `sed` filters the output from the cat command. The option "−n" tells sed to block all the incoming lines but those explicitly matching my expression. The `sed` action on a match is "p"= print.
Another example, this time for deleting selected lines:

```
cat filename | sed '/.*o$/d' > new_file
```

In this example, lines ending the an "o" will be deleted. I used a regular expression for matching any string followed by an "o" and the end of the line. The output (i.e., all lines but those ending with "d") is directed to `new_file`.
Another example. To search and replace, I use the `sed` 's' action, which comes in front of two expressions:

```
cat filename | sed 's/string_old/string_new/' > newfile
```

A shorter form for the last command is:

```
sed 's/string_old/string_new/' filename > newfile
```

To insert a text from a text file into an html file, I may use a script containing:

```
sed '/text_which_is_a_placeholder_in__my_html_file/r text_file_to_insert.txt'
index_master_file.html > index.htmll
```

`gawk`
(=GNU awk. The `awk` command is a traditional UNIX tool.) A tool for processing text files, in many respects similar to `sed`, but more powerful. Perl can do all that `gawk` can, and more, so I don't bother with `gawk` too much. For simple tasks, I use `sed`, for more complicated tasks, I use perl. In some instances, however, `awk` scripts can be much shorter, easier to understand and maintain, and faster than an equivalent perl program.

`gawk` is particularly suitable for processing text−based tables. A table consists of records (each line is normally one record). The records contain fields separated by a delimiter. Often used delimiters are whitespace (gawk default), comma, or colon. All gawk expressions have a form: `gawk 'pattern {action}' my_file`. You can ommit the patern or action: the default pattern is "match everything"

and the default action is "print the line". `gawk` can also be used as a filter (to process the output from another command, as used in our examples).

Example. To print lines containing the string "1024", I may use:

```
cat filename | gawk '/1024/ {print}'
```

Like in `sed`, the patterns to match are enclosed in a pair of "/ /".

What makes `gawk` more powerful than `sed` is the operations on fields. $1 means "the first field", $2 means "the second field", etc. $0 means "the entire line". The next example extracts fields 3 and 2 from lines containing "1024" and prints them with added labels "Name" and "ID". The printing goes to a file called "newfile":

```
cat filename | gawk '/1024/ {print "Name: " $3 "ID: " $2}' > newfile
```

The third example finds and prints lines with the third field equal to "peter" or containing the string "marie":

```
cat filename | gawk '$3 == "peter" || $3 ~ /marie/ '
```

To understand the last command, here is the list of logical tests in `gawk`: `==` equal, `!=` not equal, `<` less than, `>` greater than, `<=` less than or equal to, `>=` greater than or equal to, `~` matching a regular expression, `!~` not matching a regular expression, `||` logical OR, `&&` logical AND, `!` logical NOT.

`cvs`

Concurrent versions system. Try: `info cvs` for more information. Useful to keep the "source code repository" when several programmers are working on the same computer program.

`cervisia`

(in X−terminal). A GUI front−end to the cvs versioning system.

`file −z` *`filename`*

Determine the type of the file *filename*. The option −z makes `file` look also inside compressed files to determine what the compressed file is (instead of just telling you that this is a compressed file).

To determine the type of content, `file` looks inside the file to find particular patterns in contents ("magic numbers")−−it does not just look at the filename extension like MS Windows does. The "magic numbers" are stored in the text file `/usr/share/magic`−−really impressive database of filetypes.

`touch` *`filename`*

Change the date/time stamp of the file *`filename`* to the current time. Create an empty file if the file does not exist. You can change the stamp to any date using `touch −t 200201311759.30` (year 2002 January day 31 time 17:59:30).

There are three date/time values associated with every file on an ext2 filesystem:

− the time of last access to the file (atime)

− the time of last modification to the file (mtime)

− the time of last change to the file's inode (ctime).

Touch will change the first two to the value specified, and the last one always to the current system time. They can all be read using the stat command (see the next entry).

`stat` *`filename`*

Print general info about a file (the contents of the so−called inode).

`strings` *`filename`* `| more`

Display the strings contained in the binary file called *filename*. "strings" could, for example, be a useful first step to a close examination of an unknown executable.

`od`

(=octal dump). Display contents as octal numbers. This can be useful when the output contains non−printable characters. For example, a filename may contain non−printable characters and be a real pain. This can also be handy to view binary files.

Examples:

```
dir | od −c | more
```

(I would probably rather do: `ls −b` to see any non−printable characters in filenames).

```
cat my_file | od −c |more
od my_file |more
```

Comparison of different outputs:

Show 16 first characters from a binary (/bin/sh) as ASCII characters or backslash escapes (octal):

```
od −N 16 −c /bin/sh
```
output:
```
0000000 177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0
```
Show the same binary as named ASCII characters:
```
od −N 16 −a /bin/sh
```
output:
```
0000000 del   E   L   F soh soh soh nul nul nul nul nul nul nul nul nul
```
Show the same binary as short hexcadecimals:
```
od −N 16 −t x1 /bin/sh
```
output:
```
0000000 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
```
Show the same binary as octal numbers:
```
od −N 16 /bin/sh
```
output:
```
0000000 042577 043114 000401 000001 000000 000000 000000 000000
```

`wc`
(=word count) Print the number of lines, words, and bytes in the file.

Examples:
```
dir | wc
cat my_file | wc
wc myfile
```

`cksum` *`filename`*
Compute the CRC (="cyclic redundancy check") for file *filename* to verify its integrity.

`md5sum` *`filename`*
Compute a md5 checksum (128−bit) for file `filename` to verify its integrity.

`mkpasswd −l 10`
Make a hard−to−guess, random password of the length of 10 characters.

`sort −f` *`filename`*
Arrange the lines in filename according to the ascii order. The option −f tells `sort` to ignore the upper and lower character case. The ascii character set is (see `man ascii`):

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL '\0'	32	20	SPACE	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL '\a'	39	27	'	71	47	G	103	67	g
8	08	BS '\b'	40	28	(	72	48	H	104	68	h
9	09	HT '\t'	41	29	)	73	49	I	105	69	i
10	0A	LF '\n'	42	2A	*	74	4A	J	106	6A	j
11	0B	VT '\v'	43	2B	+	75	4B	K	107	6B	k
12	0C	FF '\f'	44	2C	,	76	4C	L	108	6C	l
13	0D	CR '\r'	45	2D	−	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x

```
25   19   EM          57   39   9         89   59   Y       121   79   y
26   1A   SUB         58   3A   :         90   5A   Z       122   7A   z
27   1B   ESC         59   3B   ;         91   5B   [       123   7B   {
28   1C   FS          60   3C   <         92   5C   \  '\\' 124   7C   |
29   1D   GS          61   3D   =         93   5D   ]       125   7D   }
30   1E   RS          62   3E   >         94   5E   ^       126   7E   ~
31   1F   US          63   3F   ?         95   5F   _       127   7F   DEL
```

If you wondered about the control characters, here is the meaning of some of them on the console (Source: `man console_codes`). Each line below gives the code mnemonics, its ASCII decimal number, the key combination to produce the code on the console, and a short description:

BEL (7, <Ctrl>G) bell (=alarm, beep).

BS (8, <Ctrl>H) backspaces one column (but not past the beginning of the line).

HT (9, <Ctrl>I) horizonal tab, goes to the next tab stop or to the end of the line if there is no earlier tab stop.

LF (10, <Ctrl>J), VT (11, <Ctrl>K) and FF (12, <Ctrl>L) all three give a linefeed.

CR (13, <Ctrl>M) gives a carriage return.

SO (14, <Ctrl>N) activates the G1 character set, and if LF/NL (new line mode) is set also a carriage return.

SI (15, <Ctrl>O) activates the G0 character set.

CAN (24, <Ctrl>X) and SUB (26, <Ctrl>Z) interrupt escape sequences.

ESC (27, <Ctrl>[) starts an escape sequence.

DEL (127) is ignored.

CSI (155) control sequence introducer.

`uniq`
(=unique) Eliminate duplicate lines in **sorted** input. Example: `sort myfile | uniq`

`fold −w 30 −s my_file.txt > new_file.txt`
Wrap the lines in the text file `my_file.txt` so that there is 30 characters per line. Break the lines on spaces. Output goes to `new_file.txt`.

`fmt −w 75 my_file.txt > new_file.txt`
Format the lines in the text file to the width of 75 characters. Break long lines and join short lines as required, but don't remove empty lines.

`nl myfile > myfile_lines_numbered`
Number the lines in the file `myfile`. Put the output to the file `myfiles_lines_numbered`.

`indent −kr −i8 −ts8 −sob −l80 −ss −bs −psl "$@" *.c`
Change the appearance of "C" source code by inserting or deleting white space. The formatting options in the above example conform to the style used in the Linux kernel source code (script `/usr/src/linux/scripts/Lindent`). See `man indent` for the description of the meaning of the options. The existing files are backed up and then replaced with the formatted ones.

`rev filename > filename1`
Print the file *filename*, each line in reversed order. In the example above, the output is directed to the file *filename1*.

`shred filename`
Repeatedly overwrite the contents of the file *filename* with garbage, so that nobody will ever be able to read its original contents again.

`paste file1 file2 > file3`
Merge two or more text files on lines using `<Tab>` as delimiter (use option "d=" to specify your own delimiter(s).

Example. If the content of `file1` was:

```
1
2
3
```

and `file2` was:

```
a
b
c
d
```

the resulting `file3` would be:

```
1     a
2     b
3     c
```

```
d
```

`join file1 file2 > file3`
Join lines of two files on a common field. `join` parallels the database operation "join tables", but works on text tables. The default is to join on the first field of the first table, and the default delimiter is white space. To adjust the defauls, I use options which I find using `man join`).

Example. if the content of `file1` was:
```
1 Barbara
2 Peter
3 Stan
4 Marie
```
and `file2` was:
```
2 Dog
4 Car
7 Cat
```
the resulting `file3` would be:
```
2 Peter Dog
4 Marie Car
```

`des −e plain_file encrypted_file`
(="Data Encryption Standard") Encrypt *`plain_file`*. You will be ask for a key that the program will use for encryption. Output goes to *`encrypted_file`*. To decrypt use
`des −d encrypted_file decrypted_file`.

`gpg`
"Gnu Privacy Guard"−−a free equivalent of PGP ("Pretty Good Privacy"). `gpg` is more secure than PGP and does not use any patented algorithms. `gpg` is mostly used for signing your e−mail messages and checking signatures of others. You can also use it to encrypt/decrypt messages. http://www.gnupg.org/ contains all the details, including a legible, detailed manual.

To start, I needed a pair of keys: private and public. The private key is used for signing my messages. The public key I give away so that others can use it to verify my signatures. [One can also use a public key to encrypt a message so it can only be read using my private key.] I generated my keypair using this command:
`gpg −−gen−key`

My keys are stored in the directory `~/.gnupg` (encrypted using a passphrase I supplied during the key generation). To list my public key in plain text file, I use:
`gpg −−armor −−export my_email_address > public_key_stan.gpg`

which created a file public_key_stan.gpg containing something like this:

```
−−−−−BEGIN PGP PUBLIC KEY BLOCK−−−−−
Version: GnuPG v1.0.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGiBDmnzEYRBACoN438rxANaMfCy5bfj6KWM0/TR6x6HZ0gpmhGeuouM/SOR2IU
/G30NdCuzHeFs93BhtY0IdzoEMtMyZHnvdhZC2bx/jhgaaMbEaSsXwRhVB0xVPYx
rHbsgSULHYzRFF34MS3/Lse3QWfWxzA7I0lbXB7nLwZKZqaNONRFRR42owCg60hV
TDPEB2N0llMyt12R4ZByFSsEAJ1tE7pb9b6TP7cw21vkIjc+BI2uSzn/B4vNlCWK
TTuZHVv0w0jFcbd8DB0/1tlZUOrIzLSqJyQDGiNn258+7LetQ+LKG/1YKbiAcosz
4QirBuLIeF2M9GuXYCwZypE3Dwv+4YupvybR31CgLTJ8p4sKqC5n0eSr2oSrtdHZ
yuJtA/9v2HcebOncfCNOK+cVRmcTB1Frl/Gh/vNCfeZyXaJxlqDfCU2vJHtBemiE
AtcfZHB/iHy0DM68LfRJSAIFAa5um9iWHh5/vWCGZLqtpwZ7kyMw+2D6CFkWATsy
wQA1g1VcGkNc14Crrd36qf60bI+b8pn2zDhwZtLsELsXyXkNhbQmU3RhbiBKIEts
aW1hcyA8U3RhbktsaW1hc0B3ZWJoYXJ0Lm5ldD6IVgQTEQIAFgUCOafMRgQLCgQD
AxUDAgMWAgECF4AACgkQt+ZBooH8bHd2kwCghAt9aKIk0mRJv+g7YcRPotVtrwkA
n1a4xEVEyaKgKoMaJnopf69K9+vouQENBDmnzH4QBADgFpLP+tWZPnVYg47cn+9b
XQRjdOtNsDE6BYH872/sR1oCrdH6k+gXFOiZxRZ3PElK2/olo59kh5xa9aBxNdEC
FuXJN0UelmhOFbDtqVksIqVWyYfXnLz+wtcXg0Q0L0q8vY4IuTzw2WkV6EkM+/x8
6UhA2XVaMJKBdRKFSVilbwADBQP+JCzLj5HDgpRvf+KM72nzSg7sp8Tki7nF9wNA
PODK0SeQgI3dwXYyF6AVenlETE/3xRWoYQN1bxVZsOex9vzqPrQC3dR0NBljd74r
kfXwUTl2fNQX4N9iuVCo2gCGbi5+gfEk1GhsWDsq0z40f+18k+XBdWmY8sCNiolT
tnvm1QeIRgQYEQIABgUCOafMfgAKCRC35kGigfxsd9SGAJ9/FWSkEfgbE/Yc46d8
Ef1gYg3I1ACff3oLeAMeGGO79gW6UGp9RJ6mRao=
=X1k2
−−−−−END PGP PUBLIC KEY BLOCK−−−−−
```

Now, I can e−mail my public key to the people with whom I want to communicate securely. They can store it on their pgp system using;

```
gpg −−import public_key_stan.gpg
```

Even better, I can submit my public key to a public key server. To find a server near me, I used:

```
host −l pgp.net | grep wwwkeys
```

and to submit the key, I did (can take a couple of minutes, and I am connected to the Internet):

```
gpg −−keyserver wwwkeys.pgp.net −−send−keys
linux_nag@canada.com
```

The "wwwkeys.pgp.net" is the key server I selected, and "linux_nag@canada.com" is my email address that identifies me on my local key ring. I need to submit myself only to one public key server (they all synchronize).

Now, I can start using gpg. To manually sign a plain text file `my_message`, I could use:

```
gpg −−clearsign my_message
```

This created file `my_message.asc` which may contain something like:

```
−−−−−BEGIN PGP SIGNED MESSAGE−−−−−
Hash: SHA1

Hello World!

−−−−−BEGIN PGP SIGNATURE−−−−−
Version: GnuPG v1.0.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

iD8DBQE5p9+3t+ZBooH8bHcRApn/AJ9kx9+pU3GJBuvJN9Bo3bW3ku/5PwCgquht
mfrPrt7PQtdmGox72jkY0lo=
=rtK0
−−−−−END PGP SIGNATURE−−−−−
```

To verify a signed message, I could do:

```
gpg −−verify my_message.asc
```

If the contents of the signed section in `my_message.asc` was even slightly modified, the signature will not check.

Manual signing can be awkward. But, for example, `kmail` can sign the electronic signatures automatically for me.

"docbook" tools

Docbook is the incoming standard for document depository. The docbooks tools are included with RH6.2 (and later) in the package "jade" and include the following converters: `db2ps`, `db2pdf`, `db2dvi`, `db2html`, `db2rtf` which convert docbook files to: postscript (*.ps), Adobe Portable Document Format (*.pdf), device independent file format (*.dvi), HyperText Markup Language (*.html), and Rich Text Format (*.rtf), respectively.

"Document depository" means the document is in a format that can be automatically translated into other useful formats. For example, consider a document (under development) which may, in the future, need to be published as a report, a journal paper, a newspaper article, a webpage, perhaps a book, I (the author) am still uncertain. Formatting the document using "hard codes" (fonts, font sizes, page breaks, line centering, etc.) is rather a waste of time−−styles vary very much between the particular document types and are publisher−dependent. The solution is to format the document using "logical" layout elements which may include the document title, chapter titles, subchapters, emphasis style, picture filenames, caption text, tables, etc. Thats what "docbook" does−−it is a description of styles (using xml, a superset of html, and a close relative of sgml)−−a so−called stylesheet. The logical layout is rendered to a physical appearance when the document is being published.

This section will be expanded in the future as we learn to use docbook.

7.2 Simple Programming under Linux

`perl`
Powerful and widely used scripting language, very popular among gurus. Perl looks cryptic yet it is quite straight−forward if you need to achieve simple tasks. Think of perl as a swiss−army knife for simple programming. Perl's syntax parallels that of the "C" language. Excellent implementation of the perl interpreter is available for MS Windows so you code can be cross−platform. Here is how Eric Reymond (famous Linux guru) describes perl: "Perl, of course, is the 800−pound gorilla of modern scripting languages. It has largely replaced shell as the scripting language of choice for system administrators, thanks partly to its comprehensive set of UNIX library and system calls, and partly to the huge collection of Perl modules built by a very active Perl community. The language is commonly estimated to be the CGI language behind about 85% of the ``live" content on the Net. Larry Wall, its creator, is rightly considered one of the most important leaders in the Open Source community, and often ranks third behind Linus Torvalds and Richard Stallman in the current pantheon of hacker demigods."

How do I write a simple perl script?

I may use `pico` (or any other text editor of my choice) to type in a simple perl script:

```
pico try_perl
```

The example script below does nothing useful, except illustrates some features of perl:

```
#!/usr/bin/perl −w
# a stupid example perl program
# the lines starting with # are comments except for the first line
# names of scalar variables start with $
$a=2;
$b=3;
# each instruction ends with a semicolon, like in "c"
print $a**$b,"\n";
$hello_world='Hello World';
print $hello_world,"\n";
system "ls";
```

The first line tells the shell how to execute my text file. The option "−w" causes perl to print some additional warnings, etc. that may be useful for debugging your script. The next 3 lines (starting with #) are comments. The following lines are almost self explanatory: I assign some values to two variables ($a and $b), put $a to power $b and print the result. The "\n" prints a new line, just like in the "c" programming language. Then I define another variable to contain the string "Hello World" and, in the next line, I print it to the screen. Finally, I execute the local operating system command "ls", which on Linux prints the listing of the current directory content. Really stupid script.

After saving the file, I make it executable:
`chmod a+x try_perl`

Now, I can run the script by typing:
`./try_perl`

Here is somewhat longer script that does something very simple yet useful to me. I take a long text file which is generatated by a data acquisition system. I need to erase every other line (or so) so that the file can be crammed into MS Excel (as required):

```
#!/usr/bin/perl −w
# Create a text file containing a selection of lines from an original file. This is needed
# so that data for manual postprocessing are fewer.
#
# Prompt the user for the filename, and the selection of lines to preserve in the output.
print STDOUT "Enter the filename: ";
chomp($infile=<STDIN>);
open(INFILE,"<$infile");  # open the file for reading.
print STDOUT "Enter the number of initial lines to preserve: ";
chomp($iskip=<STDIN>);    # the first lines may contain column headings etc
print STDOUT "Enter the skip: ";
chomp($skip=<STDIN>);
#
# The name of the output file is created automatically on the basis of the
# input file and the selection of lines. It is always of type CSV, so preserve is so.
$outfile=$infile.'−pro'.$iskip.'−'.$skip.'.csv';  #glue strings together using the dot
```

```
operator.
open(OUTFILE,">$outfile"); # open file for writing.
#
# write the "initial" lines to the output file.
for($a=0;$a<$iskip;$a++) {
    $line=<INFILE>;
    print OUTFILE $line;
}
#
# do the rest of the file
$c=0;$w=0;$skip++;
while($line=<INFILE>){
    $c++;
    if(!($c%$skip)) {    #use % for remainder of integer division
        print OUTFILE $line;
        $w++;
    }
}
#
close(OUTFILE);
print STDOUT "Read Lines: ", $c+$iskip," Wrote lines: ", $w+$iskip,"\n";
```

python

Modern and very elegant object oriented interpreter. Powerful and (arguably) more legible than perl. Very good (and large) free handbooks by G. van Rossum (the Python creator) are available on the net (try: http://www.python.org/doc/ for browsing or ftp://ftp.python.org for downloading).

How do I write a simple Python program?

Edit a text file that will contain your Python program. I can use the kde "kate" editor to do it (under X):

```
kate try_python.py &
```

Type in some simple python code to see if it works:

```
#!/usr/bin/env python
print 2+2
```

The first line (starting with the "pound−bang") tells the shell how to execute this text file−−it must be there (always as the first line) for Linux to know that this particular text file is a Python script. The second line is a simple Python expression.

After saving the file, I make it executable:

```
chmod a+x try_python.py
```

after which I can run it by typing:

```
./try_python.py
```

Python is an excellent, and very modern programming language. Give it a try, particularly if you like object oriented programming. There are numerous libaries/extensions available on the Internet. For example, scientific python (http://starship.python.net/crew/hinsen/scientific.html) and numeric python (http://sourceforge.net/projects/numpy) are popular libraries used in engineering.

Here is a slightly longer, but still (hopefully) self−explanatory python code. A quick note: python flow control depends on the code indentation−−it makes it very natural looking and forcing legibility, but takes an hour to get used to.

```
#!/usr/bin/env python
# All comments start with a the character "#"
# This program converts human years to dog years

# get the original age
age = input("Enter your age (in human years): ")
print           # print a blank line
```

```
# check if the age is valid using a simple if statement
if age < 0:
        print "A negative age is not possible."
elif age < 3 or age > 110:
        print "Frankly, I don't believe you."
else:
        print "That's the same as a", age/7, "year old dog."
```

`tcl`
(Pronounced "tickle".) Popular scripting language.

A simple tcl program?

```
#!/usr/bin/tclsh
puts stdout {Hello World!}
```

`wish`
(type in X−terminal) A front−end to Tk, an X−windows extension of tcl. Often used for building front−ends of a program.

How do I write a simple GUI program (using Tk)?

Tk is a GUI extension of the easy yet powerful tcl programming language. For example, I may use `pico` to create a text file that will contain a simple tk program:

```
pico try_tk
```

and type in a simple example of tk code to see if it works:

```
#!/usr/bin/wish
button .my_button −text "Hello World" −command exit
pack .my_button
```

The first line (starting with the "#!" pound−bang) tells the shell what utility to use to execute my text file. The next two lines are an example of a simple tk program. First, I created a button called "my_button" and placed it at the root of my class hierarchy (the dot in front of "my_button"). To the button, I tied the text "Hello World" and a command that exits the program (when the button is pressed). Last line makes my program's window adjust its size to just big enough to contain my button.

After saving the file, I make it executable:
`chmod a+x try_tk`

after which I can run it by typing (in the X−terminal, because it requires X−windows to run):
`./try_tk`

Tk is very popular for building GUI front ends.

`ruby`
A purely object−oriented scripting language. This language is a relative newcomer, but it is rapidly gaining popularity, and may well be the flavour of the future of programming.

To write a simple program in ruby, I open my favorite text editor and start a program with the following first line:
`#!/usr/bin/ruby`

Here is an example of a program that I wrote to help me understand the basics of the ruby language:

```
#!/usr/bin/ruby
#This is a comment
 a = Array.new
 print "Please enter a few words (type EXIT to stop):\n"

 i = 0
 while enterWord = STDIN.gets
   enterWord.chop!
   if enterWord == "EXIT"
```

```
            break
          end
          a[i] = enterWord
          i += 1
        end

      #sort the array
      for i in 0...a.length-1 do
        for j in i+1...a.length do
          if a[j] < a[i]
            tmp = a[i]
            a[i] = a[j]
            a[j] = tmp
          end
        end
      end

      #Output the results
      print "You entered "  + a.length.to_s + " entries.\n\n"
      for i in 0...a.length do
        print "Entry " + (i+1).to_s + ": "+ a[i] + "\n"
      end
```

I save my ruby script to file "myprogram". To execute it, I need to type on the command line:

```
./myprogram
```

`gcc` *`filename.c`*

GNU C compiler. Quite straight−forward if you know C. Extensive free manuals are available on the net.

How do I compile a simple C program?

Start your favourite text editor and type in your source code. For example, I may use `pico`:

```
pico hello.c
```

and type in the Kerningham and Richie (the creators of "c") intro example of a C program:

```
#include <stdio.h>
void main(void) {
printf("hello world\n");
}
```

I save the file and then invoke the GNU C compiler to compile the file "hello.c":

```
gcc hello.c
```

The gcc compiler produces an executable binary file "a.out", which I can run:

```
./a.out
```

`g++` *`filename.C`*

GNU C++ compiler. The capital "C" is often used for C++ sources. If you need an "integrated development envrionment" (IDE), `kdevelop` is really something you would probably like to look at.

How do I compile a simple C++ program?

Just like in c, I open a text editor and write my program. For example, using pico, I write the following program:

```
//This is a comment (to the end of line, C++ style)
#include <stdio.h>
#include <math.h>
#include <iostream.h>
#include <stdlib.h>

//define a function
double wheeldrop (double dGap, double dDiameter) {
    double dDrop, dRadius, dNotDrop;
```

```
        dRadius = dDiameter * 0.5;
        dDrop = dRadius − sqrt( (dRadius*dRadius)−(0.25*dGap*dGap) );

        return (dDrop);
    }   //end of the function

    //The function main is the entry point to the program
    void main(void) {
        double dGap, dDiameter, dDrop, dRadius, dNotDrop;  //variables

        for (;;) {                                          //infinite loop
            cout << "Please enter gap between track segments and \n"
          << "diameter of train wheel in inches (−1 −1 to exit): ";
            cin >> dGap >> dDiameter;

            if ((dGap == −1) && (dDiameter == −1))
                break;
            else if (dGap < dDiameter) {   //do calculations
                dDrop = wheeldrop (dGap, dDiameter);
                printf ("The wheel will drop %f inches.\n\n", dDrop);
            }
            else {
                printf ("Error, your train is going to crash.\n Gap bigger then
    wheel!\n\n");
            }
        }
    }
```

I save the source to the file "train.c", and then invoke the GNU C++ compiler to compile the file "train.c" to an executable called "traincalc":

```
g++ −o traincalc train.c
```

I can then run the executable by typing:

```
./traincalc
```

`kdevelop`
(type in X−terminal) Integrated development environment for K. It is really worth downloading (if it does not come with your distribution).

`glade`
(type in X−terminal) A graphical builder of user interfaces.

> "Glade is an interface builder developed by Damon Chaplin. It allows graphical and interactive construction of Gnome/Gtk graphical user interfaces. From Glade, the generated interface can be saved in a xml file or directly exported to C code to be included in a C source tree. Glade also allows to define the name of the handlers − functions − to be attached to the various event of the interface. For example the function (name) to be called when a specific menu item is pressed." (From: http://linuxtoday.com/news_story.php3?ltsn=2000−07−16−013−04−PS−GN)

What "C" functions are available for programming under Linux?
Too many for a newbie like myself. I started by studying the header files (*.h) in the directory `/usr/include` and all its subdirectories. To find a header file that contains a prototype for a given function (e.g., cosh) I would do something like:

```
cd /usr/include
grep −H "cosh" *.h
```

There are also many interesting libraries that are not a part of a typical distribution, e.g., GNU libraries for scientific computing (GSL): http://sources.redhat.com/gsl/. Also check: http://www.phy.duke.edu/~hsg/sci−computing.html.

`as`
`nasm`
`ndisasm`
(3 commands). Assembler, an alternative "native assembler" and a disassembler. Send in your newbie examples how to use those :)

E.g.: `ndisasm /bin/sh |more` produces a long output of "assembler mnemonics" from a binary on my system (/bin/sh in this example) but I cannot understand it anyway :(To learn more about nasm, you may want to see: file:///usr/share/doc/nasm−doc−0.98/html/nasmdoc0.html

Example Intel assembly for Linux 2.2.17 or higher:

```
;; hello.asm: Copyright (C) 2001 by Brian Raiter, under the GNU
;; General Public License (version 2 or later). No warranty.

BITS 32

  org 0x05936000

  db 0x7F, "ELF"
  dd 1
  dd 0
  dd $$
  dw 2
  dw 3
_start:  inc eax   ; 1 == exit syscall no.
  mov dl, 13       ; set edx to length of message
  cmp al, _start − $$
  pusha            ; save eax and ebx
  xchg eax, ebx    ; set ebx to 1 (stdout)
  add eax, dword 4 ; 4 == write syscall no.
  mov ecx, msg     ; point ecx at message
  int 0x80         ; eax = write(ebx, ecx, edx)
  popa             ; set eax to 1 and ebx to 0
  int 0x80         ; exit(bl)
  dw 0x20
  dw 1
msg:  db 'hello, world', 10
```

After saving this to a plain−text file `hello.asm`, I can build it to an output file "hello" and make the output executable using the command:

```
nasm −f bin −o hello hello.asm && chmod +x hello
```

and execute it using:

```
./hello
```

The example above is borrowed from http://www.muppetlabs.com/~breadbox/software/tiny/.

Why would somebody use assembler? After building from assemebler, this example executable is 56 bytes on my system. The "C" language example with identical functionality (see one page above) is 13.7 kB.

Here is brief info to help me understand the above program:

";" marks comments (to the end of the line).

"msg:" −− is an example of a label (like in fortan).

org (="origin")−−declares where in the memory the program begins (after it is loaded to memory for execution).

db, dd, dw are nasm "pseudoinstructions" used to insert initialized data into the output file.

"$" evaluates to the assembly position at the beginning of the line containing the expression; so you can code an infinite loop using "JMP $". "$$" evaluates to the beginning of the current section.

The general−purpose 32−bit registers in the 80x86 ("Intel") processor are: EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP. (The "E" stands for extended. It is there because the processor can instead "overlay" the registers and treat them as 16−bit registers with names: AX, BX, CX, CX, SI, DI, BP, and SP. Still underlying those, there are also eight 8−bit registeres: AL, AH, BL, BH, CL, CH, DL, DH. Here, the "L" and "H" stand for "high" and "low" byte.).

Mnemonics for some common 80x86 processor instructions:

```
Name  Syntax                    Comment
NOP   NOP                       No operation (do nothing).
MOV   mov destination, source   Move (copy, set)data.
XCHG  XCHG operand1,operand2    Exchange the values.
CMP   CMP operand1,operand2     Compare the two operands.
PUSH  PUSH source               Push onto stack.(Place the value on stack and increment the
stack pointer).
PUSHF PUSHF                     Push flags.
```

```
PUSHA PUSHA                    Push all general−purpose registers.
POP   POP destination          Pop from stack(take the value from stack, and decrement the
stack pointer). Pop is reverse to push.
POPF  POPF                     Pop flags.
POPA  POPA                     Pop all general−purpose registers.
INC   INC operand              Increment (increase by 1).
DEC   DEC operand              Decrement (decrease by 1).
ADD   ADD Dest,Source          Add.
ADC   ADC Dest,Source          Add with carry.
SUB   SUB Dest,Source          Subtract.
INT   INT number               Execute an interrupt.
CALL  CALL subroutine          Call a subroutine.
RET   RET                      Return from this (current, innermost) subroutine.
JMP   JMP destination          Jump (start executing code starting at the the address
"destination")
JE    JE destination           Jump if equal.
JNE   JNE destination          Jump if not equal.
JZ    JZ destination           Jump if zero.
JNZ   JNZ destination          Jump if not zero.
JP    JP destination           Jump if parity (parity is even).
JNP   JNP destination          Jump if no parity (parity is odd).
JPE   JPE destination          Jump if parity even.
JPO   JPO desitination         Jump if parity odd.
JCXZ  JCXZ destination         Jump if CX zero.
JECXZ JECXZ destination        Jump if ECX zero.
```

guile

An implementation of "Scheme" programming language. Scheme is a modern dialect of the LISP language (the one that has been promising the artificial intelligence for the last 40 years).

Silly examples for the guile interpreter.

```
guile
(+ 1 1)
(define a 2)
(/ a 3)
(= a 7)
(display "hello\n")
(system "ls")
(exit)
```

The first command runs the guile interpreter. The next four commands do addition, definition, division, and comparison using the so−called Polish notation (operator in front of the operants). See the section on reverse Polish notation on this page. The last command exits the guile interpreter.

g77

GNU FORTRAN. An on−line manual is available at: http://gcc.gnu.org/onlinedocs/g77/. If you are really into FORTRAN, you might also want to check: http://studbolt.physast.uga.edu/templon/fortran.html to find a FORTRAN compiler that suits your particular needs under Linux.

Silly example of a fortran code. It prints squares and cubes of numbers from 1 to 20:

```
PROGRAM TRY_FORTRAN
  INTEGER X
  PRINT 200, "X", "X^2", "X^3"
  DO X=1, 20
     PRINT 100, X, X**2, X**3
  END DO
100 FORMAT (I20, I20, I20)
200 FORMAT (A20, A20, A20)
END
```

To compile this file, I run the fortran compiler with the option that recognizes the "free−form" source code (I don't like the fixed−code source):

```
g77 −ffree−form try_fortran.f
```

and now I can run the resulting executable (which has the default name is a.out):

```
./a.out
```

`expect`
Scripting language for "programmed dialog". See `man expect`.

kylix
This is a brand−new (Feb.2001) commercial offering from Borland (aka Inprise). In short, it is a Linux port of the famous object−oriented Pascal ("Delphi"). kylix is unlikely to be on your Linux distribution CD, you must pay for it, but if you want the best rapid application development (RAD) platform with a code portablity between Linux, MS Windows and the Web, large number of pre−built components, etc., kylix is likely the best. In my opinion, Delphi is significanly better than MS Visual Basic.

javac
java
The Java language compiler and interpreter. Under Linux, both `javac` and `java` are actually scripts which call `kaffe` with appropriate options (try `cat /usr/bin/java`).

A trivial example for a java "standalone" program. I use my favourite text editor, e.g. `kate` (in X terminal) to type in the following java code:

```
/* Comments are marked like in C++
 * A java class to display "Hello World"
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World."); // Print "Hello World." followed by a newline
    }
}
```

I save this into a file named `try_java.java`. Now, I can compile it using:

```
javac try_java.java
```

This creates a file called `HelloWorldApp.class` which contains the "bytecode" (semi−compiled code which requires an interpreter to run). I can run it on the command line using:

```
java HelloWorldApp
```

For an example on how to embed a simple java applet into an html web page, have a look at http://java.sun.com/docs/books/tutorial/getStarted/cupojava/unix.html from which my java "standalone" program was borrowed.

`make`
Run the "make" utility to build (compile, link, etc) a project described in the `Makefile` found in the current directory.

`make` is used to bring a system "up to date", whenever a change in one file requires an action elsewhere. `make` is "intelligent" in that it will not make changes unless the change is required, as determined by the file modification date/time. Normally used for buiding software packages (compiling, linking ...), `make` is also used for other tasks, e.g., system administration. `Makefile` looks as follows:

```
target :  prerequisites
[Tab]commands
```

where `target` is usually a file (but does not have to be, it can be a "phony" target), and `prerequisites` are files on which `target` depends. If `target` does not exist or is older than any prerequisites, "commands" are executed. The first line above is called "the rule", the second "the action". Please note that any action line must start with the tab character. Here is an example Makefile that makes an executable file called "edit":

```
my_program : main.o command.o
    cc −o my_program main.o command.o
main.o : main.c defs.h
    cc −c main.c
command.o : command.c defs.h command.h
    cc −c command.c
clean :
    rm my_program main.o command.o
```

To use this `Makefile` to create an executable file called "my_program', I type: `make`. It works backwards to determine the dependencies, so first it compiles "command.c" to the object file "command.o", then it compiles "main.c" to "main.o", and finally it links "main.o" and "command.o" into the executable "my_program".
One could also use this makefile to delete the executable file and all the object files from the directory by typing: `make clean`. Since the target "clean" does not depend on any prerequisites, it is not normally executed unless explicitly called. The target "clean" is an example of a "phony" target.

`yes`
Generate a never−ending output of strings containing "yes" (it does end when <Ctrl><c> is pressed or when electricity goes off). Sounds like a silly utility, but it can be used to write simple programs on the command line. For example, the following amusing on−liner determines the frequency of digits in 100 000 radom numbers (the whole command is a single line):

```
yes | sed '100000q' | awk 'BEGIN{srand();u=6*log(10)}{printf"%e\n",rand()*exp(rand()*u)}'|
cut −c1 | sort | uniq −c
```

I hope this example does not scare you too much−−it surely shows that the old−fashioned UNIX command line can be as complicated (and powerful) as you wish to make it. If you are interested why the frequency of digits varies (it seems intuitively that it is should be constant if the numbers are random), try the website from which I borrowed the example:
http://www.newscientist.com/ns/19990731/letters4.html

7.3 Math Tools

`dc`
A command−line, arbitrary−precision "reverse Polish notation" (RPN) calculator.

dc is based on the concept of a stack, which is central to the operations of modern digital computer. A computer stack is not unlike a stack of kitchen plates, the last to come on stack, is the first to go out (this is also known as LIFO="last−in, first−out"). This contrasts with a queue (another important concept) where the first in is the first out (FIFO).

You can perform operations only on the number(s) which is on the top of the stack. The two basic operations are: push and pop (put on the top of stack, and retrieve from the top of stack). Unary operations pop one value off the stack ("unary" means "requiring one operand"). Binary operations pop two values off the stack ("binary" means "requiring two operands"). Tertiary operations pop three values off the stack ("tertiary" means "requiring three operands"). In all cases, the result is always pushed back onto the top of stack.

RPN calculators (regular, hand−held) are very popular among technically oriented people and in academia. The RPN notation never requires parentheses.

History. The parentheses−free logic was developed by Polish mathematician Jan Lukasiewicz (1878−1956) before the WWII. Originally, the operator preceded the values. For computer applications, it's been modified so that the operator goes after the values, hence "reversed" in the name "reversed Polish notation".

To exercise some operations on stack, try this:
`dc` [start the arbitrary precision reverse Polish notation calculator]
`1` [push "1" on the stack]
`2` [push another number on the stack]
`3` [push yet another number on the stack]
`4` [push yet another number on the stack]
`f` [print the entire stack; you should see 1 2 3 4]
`p` [print the number on the top of the stack without affecting the stack; you should see 4]
`+` [perform addition (binary operation), therefore pop two last values off the stack (4,3), and push the result (7) on the stack]
`p` [print the number on the top of the stack, i.e. the results of the last addition (7).].
`p` [print again the number on the top of the stack to see that the stack wasn't affected by printing (7)]
`*` [perform multiplication (binary operation), therefore pop two last values, and push the result (14)]
`p` [print the result of the multiplication (14)]
`P` [pop the last number off the stack (14)]
`p` [print the number on the top of the stack]
`2000` [push a large integer on the stack]
`k` [set the precision to the value which is on the top of the stack, i.e. 2000]
`1` [push another number on the stack]
`f` [print the content of the entire stack]
`701` [push another number on the stack]
`/` [divide last two numbers on the stack, i.e. "1/701" with 2000 decimal places of precision]
`p` [print the result of the last division]
`q` [quit the arbitrary precision reverse Polish notation calculator]

Please note that when using the reverse Polish notation (RPN) you never need parentheses. Try `man dc` to read about other capabilities of dc.

`bc`
An interactive arbitrary−precision calculator. Type "quit" to exit `bc`. Type "scale=20" (or so) before doing floating point divisions or you end up with the quotient instead of the floating−point result of the division.

`kcalc&`
`xcalc&`
(in X terminal) Standard, GUI calculators.

`e '2*3/4+sin(pi/2)'`
The "e" expression evaluator did not come on my RH7.x CDs. Yet, it is my favourite of all the "command line" calculators. Try: http://www.softnet.tuc.gr/~apdim/projects/e/ to download.

`expr 1 + 1 / 3`
Evaluate an integer−type expression. The "expr" is not meant to be a calculator, but is mostly for flow control in shell scripts. The above example will return the result "1", which is correct because 1/3 is 0 as far as integer division is concerned.

`octave`
Octave is a high−level interactive language for numerical computations, closely resembling "matlab". Included with any fine Linux distribution.

`scilab&`
(in X terminal) Another large and sophisticated system for numerical computing, somewhat resembling "matlab", but with a rather clampsy graphical interface. Don't even try it unless you have rather sophisticated math needs else you won't appreciate it. It is included on RH7.0 "powertools" CD. The hompage is http://www−rocq.inria.fr/scilab/

A silly example session showing some matrix algebra. My input is shown **bold**.

```
−−>a=[1 1 1;2 3 4]
 a  =
!  1.    1.    1. !
!  2.    3.    4. !

−−>b=[1 1; 2 2;3 3]
 b  =
!  1.    1. !
!  2.    2. !
!  3.    3. !

−−>c=a*b
 c  =
!  6.     6.  !
!  20.    20. !

−−>d=inv(c)
 d  =

   1.0E+14 *
!   11.258999  − 3.3776997 !
! − 11.258999    3.3776997 !

−−>
```

`head −c 8 /dev/random`
`cat /dev/random | od`
`cat /dev/urandom | memencode`
(3 commands.) Examples on how to generate random characters on the Linux command line by reading from the device "random" or "urandom". The first command produces approximately 8 characters by reading from the device "random", which generates high quality (difficult to predict) random numbers. It will become slow once the "entropy" on your computer is exhausted (e.g., when producing lots of random characters). The solution then is to wait or type something on the keyboard, move your mouse, switch the terminal, make your hard drive to read or write, etc., to generate more random noise ("entropy"). The second command keeps producing random characters, but displays them as octal numbers (using the "octal dump", od, filter). It has to be interrupted with <Ctrl><c>. The third command uses the device "urandom", which is faster then "random" when generating lots of random characters. But when the system enthropy is low, the randomness of its output from "urandom" might be compromised, yet it probably is still good for all but most demanding applications. The output is filtered to the mime format (the Internet mail−oriented 7−bit encoding

standard) so it is all printable ASCII. The detailed description of the theorry and implementation of the Linux algorithm for generating the random numbers can be found in the source code file://usr/src/linux/drivers/char/random.c on your Linux system.

`factor 10533858466222239345`
Find all the prime factors of an integer number. Factors of an integer are numbers by which the integer is divisible without a remainder. For example, the factors for 6 are: 1, 2, 3, and 6.

`R`
A programming language / environment for advanced statistical computing. Type "quit()" to exit.

`gnuplot`
Utility for creating graphs and plots. Very good for non−interactive (batch) work, but not very simple for interactive use. A good introduction to gnuplot can be found at http://www.duke.edu/~hpgavin/gnuplot.html.

7.4 Miscellaneous

How do I run an MS Windows Application (using "wine")?

You do not install Linux to run MS Windows applications. If you need MS Windows programs, you probably want to install a dual boot on your computer.

Still, the Linux−based "wine" library lets me execute some MS Windows binaries, although with a rather severe speed penalty. On my system (Wine installed), I can execute MS Solitaire by typing in the X−windows terminal:

```
wine /mnt/dos_hda1/windows/sol.exe
```

The `/mnt/dos_hda1` is the mount point of the harddrive partition that contains MS Windows, and it is mounted.

If you don't have `wine` installed, put your Mandrake cd into the cdrom, mount it, and then do something like this (as root):

```
cd /mnt/cdrom/Mandrake/RPMS/
rpm −ihv wine−991212−1mdk.i586.rpm
```

Mandrake packages are RedHat−compatible so you can use the Mandrake CD to install software that RedHat lacks.

Can I have a RAID if my computer has two or more IDE (or other) harddrives?

RAID = "redundant array of inexpensive drives". RAID can be used for a "on−the−fly" mirroring of one drive to another so as to protect your data and keep your system functioning in case of a disk failure. Linux comes with a set of RAID tools that let you custom−design a RAID system to suit your particular needs.The pieces of RAID on Linux are:

`mkraid` − initializes/upgrades RAID device arrays
`raid0run` − starts up old (superblock−less) RAID0/LINEAR arrays
`raidstart` − command set to manage md devices
`raidstop` − command set to manage md devices
`raidtab` − configuration file for md (RAID) devices

RAID operates by joining two or more disks into a single logical device. There are several layers of RAID:

RAID 0 layer ("striping") just joins two or more disks into a single logical device, without giving any redundancy. It is often used to join RAID 1 or RAID 5 layers. RAID 0 + RAID 1 is called RAID 10. RAID 0 + RAID 5 is called RAID 50.

RAID 1 (mirroring) combines two disks, each containing the same data.

RAID 4 combines three or more disks, with one of the disks dedicated to parity. If any disk fails, the whole logical device remains available, but with degraded performance. It is not used very often because of the performance.

RAID 5 combines three or more disks, with parity distributed accross the disks. Functionality similar to RAID 4 but apparently better performance.

Try http://www.osfaq.com/vol1/linux_softraid.htm for more information.

RH7.2 gives me an option to set up a software raid almost automatically during the initial operationg system installation procedure. The (simple) procedure is outlined at http://www.redhat.com/docs/manuals/linux/RHL−7.2−Manual/custom−guide/software−raid.html. Briefly, during RH installation, part "partition the hard drive" :

(1) Create new partition(s) of the type "software RAID" You will not be able to specify a mount point for the individual "RAID−type" partitions. You can specify the size for each partition, make it "growable", or force it to be the primary partition.

(2) Press the "Make RAID" button on the Disk Druid screen.

(3) Into the dialog box which appears enter: the mount point for the RAID array, the partition type for the RAID array, the RAID type (you can select between RAID 0, RAID 1, and RAID 5), the RAID member partitions (which you created in (1)), and the number of spares (for RAID 1 or 5). (The "spare" is the partition that will be automatically used should the the software RAID fail. In (1), you should have created at least one "RAID−type" partition + one "RAID−type" partition for each spare.)

(4) Click "OK", and inspect the "Drive Summary" if your RAID array appears correctly.

Note: "If you are making a RAID partition of /boot, you must choose RAID level 1 and it must use one of the first two drives (IDE first, SCSI second). If you are not creating a RAID partition of /boot, and you are making a RAID partition of /, it must be RAID level 1 and it must use one of the first two drives (IDE first, SCSI second)"

Network traffic shaping using `shapecfg`

Nice info can be found at: http://oreilly.linux.com/pub/a/linux/2000/08/24/LinuxAdmin.html
Unlikely I will really ever need traffic shaping on my home network, yet it makes an interesting exercise for the curious.

Go to Appendix: How to Upgrade the Kernel
Back to Main Menu

Licence, Acknowledgments, etc.

LNAG LICENCE

The Linux Newbie Administrator Guide (LNAG) ("The Guide") is distributed under the Open Content Licence (http://opencontent.org/openpub/) with the following addition:

THE GUIDE IS BEING INCREMENTALLY UPDATED. THEREFORE, TO PROVIDE THE BEST VALUE TO OUR READERS, UNLESS THE GUIDE HAS BEEN SUBSTANTIALLY MODIFIED BY AUTHOR(S) OTHER THAN THE ORIGINAL GUIDE AUTHORS, ANY DISTRIBUTOR SHALL DISTRIBUTE A REASONABLY RECENT VERSION OF THE GUIDE, I.E., ITS MOST RECENT VERSION OR A VERSION NOT OLDER THEN ONE YEAR ON THE DATE OF WEB SERVING, CD WRITING, OR HARD COPY PRINTING. THE MOST RECENT VERSION OF THE GUIDE IS AVAILABLE AT HTTP://SUNSITE.DK/LINUX−NEWBIE.

THIS LICENCE MEANS THAT ANY PUBLICLY ACCESSIBLE MIRROR MUST BE UPDATED AT LEAST ONCE A YEAR, IF A NEWER VERSION OF THE GUIDE IS AVAILABLE. PLEASE DO NOT CREATE A MIRROR IF YOU DO NOT INTEND TO UPDATE IT.

All the code examples are distributed under the GNU General Public Licence (http://www.gnu.org/copyleft/gpl.html).

The maintainers of this Guide (Stan and Peter Klimas) can be contacted by email: LINUX_NAG@CANADA.COM

Acknowledgments

The Linux Newbie Administrator Guide (LNAG) is hosted FREE OF CHARGE on the SunSite server at Aalborg University, Denmark (http://sunsite.dk/linux−newbie). Thanks to Esben Haabendal Soerensen <bart@sunsite.auc.dk>.

Thanks to "linsup.com" (http://linsup.com/) for hosting our free, timely updated Australian mirror at (http://linsup.com/newbie/). Thanks to Kenan Bektas, VP Engineering for hosting our free, timely updated North American mirror: (http://dbstreams.ca/mirrors/linux−newbie/). Thanks to Andamooka for hosting LNAG among their great free books. We have to figure how to update it.

Major help and advice was received from (alphabetical order): Alan W. Irwin <irwin@beluga.phys.uvic.ca>, Benjamin Smith <bens@saber.net>, Barbara Klimas <bklimas@magma.ca>, Ben McCosh <ben@albagroup.com>, Bill Staehle <staehle@netvalue.net>, Bill Unruh <unruh@physics.ubc.ca>, Brian Kelsay <ripcrd6@kcinter.net>, Ding−Hou Lee <dl105@columbia.edu>, Gary <swear@aa.net>, Greg Mizell <GMizell@peek−traffic.com>, Jaakko Alarto <pikkumyy@ikiliekki.org>, Jeff Greenlee <greenj@analogy.com>, jeff covey <jeff.covey@pobox.com>, Jo <rainbow@linuxfocus.org>, Juhapekka Tolvanen <juhtolv@st.jyu.fi>, Ken Foskey <waratah@zip.com.au>. Special thanks to the Portland Linux User Group for review!

Thanks to the several dozen others who sent comments, minor corrections, good suggestions, or good word to us. We always appreciate any feedback. We received no flames so far :−) . Sorry if we could not answer specific questions you sent us on your particularly annoying problem :((Sorry we are unable to provide free support for individual users. Sorry we do not provide any support to MS Windows users.

Thanks to the few thousands who developed GNU/Linux. Should I mention Richard Stalman and Linus Torvalds by name?

Other Matters

You can see that this guide was written by newbies for newbies. It should never be considered an authoritative source on any topic−−there are much more exhaustive docs, typically more difficult to read too :−), most of them available right on your Linux system in the directory `/usr/doc`. Please e−mail us immediately if you spot a mistake that can confuse or mislead a new Linux user−−this work is in progress and the current version may contain such mistakes. Don't use this guide if your life or well−being was to depend on it!

If you wanted to contribute a part (on a useful Linux topic of your choice), we would like to hear from you−−we will be happy to include a part with your name, etc. The only condition is that your advice be easy to follow by newbies like us.

If you create your own mirror, pls make sure to update it at least from time to time. LNAG is under development and inaccuracies are found and corrected on regular basis. Our licence requires updates (if available on http://sunsite.dk/linux−newbie) else please remove

your outdated mirror.

Hope this helps. Best regards, Stan and Peter Klimas (linux_nag@canada.com)