Erik’s linux page

Linux information for beginners and advanced users

Erik Forsberg

Erik’s linux page: Linux information for beginners and advanced users

by Erik Forsberg

Hi and welcome to my page about Linux - most of it from the beginners point of view, but with some more or less advanced tips too. I try to update this page often,, but since I’m quite a busy person, sometimes it may take a month or two. Please sign (http://www.lysator.liu.se/~forsberg/cgi/guestbook.cgi/) my Guestbook (http://www.lysator.liu.se/~forsberg/guestbook.html) and tell me what you think about this page if you’re visiting the HTML version. Or, send me a mail, by clicking here (mailto:forsberg@lysator.liu.se)

Please check out the The Linux documentation project (http://sunsite.unc.edu/LDP) to find even more valuable information.

This page is a member of the *Linux webring. Go to:* [Home (http://www.webring.org/cgi-bin/webring?ring=linux;home)][List (http://www.webring.org/cgi-bin/webring?ring=linux;list)][Next (http://www.webring.org/cgi-bin/webring?ring=linux;id=129;next)][Prev (http://www.webring.org/cgi-bin/webring?ring=linux;id=129;prev)][Random (http://www.webring.org/cgi-bin/webring?ring=linux;random)]

If you’ve been here before, you will notice the layout has changed. I’m now using the DocBook DTD instead of the linuxdoc DTD. I hope you like the new style, it’s smaller pages so hopefully it’s faster for modem users.

Thanks to a cool guy in China there’s now a chinese version of this document available here (http://forceps.163.net/content/document/forsberg/index.html).

Since the most frequently asked question I get in mail regarding this page is "Where can I download a printable copy of this document?" I decided I should add a more easily located place. Here it is. (http://www.lysator.liu.se/~forsberg/linux/getting-this-document.html)

Version: $Id: index2.sgml,v 1.1.1.1 2004/09/19 12:25:14 forsberg Exp $ Last change of any part of the document: mån sep 20 20:09

Table of Contents

List of Tables

Chapter 1. About Erik Forsberg, and the document

Version: $Id: Chapter1.sgml,v 1.1.1.1 2004/09/19 12:25:09 forsberg Exp $

1.1. General information

Well.. This page isn't supposed to be one of these "Hello, I'm Jim, and this is my very fat cat Sara" pages.. but if you're interested, look at this page about him (/~forsberg/omerik.html)

You think this page is boring ? Almost no pictures, flashing text, background images, colours etc ?? Well.. this is what the web was in the old good days, when the internet was an information network, not a garbage can.. It's still possible to find small oasis of valuable information, but it's getting harder..

The information you find here, is in no order, with no structure, and directly out of my very weird brain :-) Watch out, or you'll be like me.. Right now, it's quite incomplete, but I'll try to write a section now and then. If you want to contribute with a text about something, feel free to mail me at the address you'll find if you read a few more lines..

If you're a new Linux user, I recommend the "Linux Installation and Getting Started Guide 2.3", a part of the Linux Documentation Project, check this url (http://sunsite.unc.edu/LDP/) for more information. A lot of the information on my pages are there too, most of it probably better written and formatted ;)

1.2. Authors address

Please give me feedback, any kind of it is welcome. Mislinks, errors, things you think should be here, misspells, whatever, just mail them ! If you want to, you can write your own section, if I think it should be here, I'll add it, with your name on it. Text or SGML format please. The e-mail address of the author is forsberg (at) lysator dot liu dot se.

1.3. Disclaimer

Standard Disclaimer follows: I don't take any responsibility for anything happening to you, your computer, your house, your neighbours, anybody/anything else, after reading this document or any other page written by me or anybody else. I won't put anything I know won't work here, but I don't guarantee it does.

All trademarks mentioned in this document are owned by their respective owners. Linux is a registered trademark of Linus Torvalds.

1.4. Getting this document

There are various versions of this document:

- PDF (Portable Document Format) (ELP.pdf)
- The sgml source (gzip compressed) (linuxpagesrc.tar.gz)
- compressed postscript (index.ps.gz)
- zip compressed RTF (yes, you can read it with your lousy word !!) (indexrtf.zip)
- zip compressed text file (indextxt.zip)

1.5. Changes in this document

The version number of this document is in the form A.B.C, where a change in A stands for a major rewrite or adding a new sect. Increasing of B indicates a new sect1 and increasing of C a new sect2 or small change, bug fix. Perhaps I'll even follow that system !! :)

Table 1-1. List of changes in this document.

Date	*What happened ?*
Dec 1997	First upload of this page
1998-02-14	I start tracking changes :)
1998-02-15	Started writing the script tutorial
1998-02-20	Added alias tip. Added guest book. Continued the script tutorial.
1998-02-21	Did some error correction.
1998-02-21	Added section on gpm. Added section on VT's.
1998-03-17	Added section on file rights.
1998-04-06	Added daemon section.
1998-04-10	Added small tip about Emacs.
1998-04-11	Added some URL's in link section.
1998-05-20	Added section about root directories.
1998-05-29	Where to get Linux. Some other small additions.
1998-05-31	Section about PPP. A few new links, minor corrections.
1998-06-07	Various minor modifications.
1998-09-05	Added some commands, and some fixes.
1998-09-13	Possible to download on and off files.
1998-11-13	New ftp server. New guestbook. Some links, and some corrections
1999-02-03	New main URL, some links. Not enough time.
1999-02-12	More info about ssh. Live date update in shell script section :-)

1999-03-28	Updated the todo list :-)
1999-07-22	Mostly administrative work and layout changes. Small section about modules
1999-07-22	New main section about compiling kernel. Removed version numbers, I’m now keeping track of the versions via RCS instead, for my own reference. Minor changes.
2000-05-02	Small section about how to access installed commands.
2001-12-17	Fixed links to alternate versions of document
2002-08-18	Fixed PDF version. Thanks to Lester Sussman for giving me the idea
2004-09-20	First update in two years. I seldom have time for this project anymore. Fixed a bad link and removed some obsole stuff.

1.6. TODO

Some plans for this document. Please mail me if you have ideas for this document.

Description of some useful tools. Descriptions of some of my hacks. Some words about the major distributions. More about printers. More links ? Some words about compiling kernels. Package systems (rpm, deb). How modules work. Also some stuff about how to find further documentation. Manual pages, Info pages, include files and good internet places. A chapter about X window system and the basics how it works might be good to.

Chapter 2. Some general words about Linux, mostly for new users.

Version: $Id: Chapter2.sgml,v 1.1.1.1 2004/09/19 12:25:13 forsberg Exp $

The operating system I run is Linux, a free 32 bit truly multitasking multiuser OS. It's fast, it's powerful, it has a TUI (Text User Interface), it has a GUI (Graphic User Interface), I can say positive things about it all day :-)

Here I'll try to assemble some information that is good to know for the new user. It's all availiable in other documentation, but if you don't know where to look, it's hard to find something useful. There will also be some tips and tricks for a more experienced user.

2.1. Some words about Linux..

What to say more about Linux ?? Linux isn't easy for the beginner, you have to dig deep into the documentation to understand it, and use it. After you've learnt, you get a lot more out of your computer, than you get with a commercial and not truly 32 bit "multitasking" OS (no names, I won't say those words on this page)

Linux runs on a variety of hardware platforms. Most Linux machines of today run som kind of Intel processor, or an Intelclone such as K6. Linux also runs on Digital Alpha, MIPS, ARM, Motorola 68x00, PowerPC, Sparc and even on PalmPilot !

Linux isn't easy to install and setup, but it isn't impossible, if you have some talent for computers, and can read English texts. It's getting easier, with a growing community of Linux users ready to help you with your problems. That's one of Linux's real advantages over other OS's.. if you ask a question in one of the numerous discussion groups on computer networks as Internet and fidonet, you get answers. Good answers, from people who know what they talk about.

Isn't there *anything* negative to say ? Of course there is.. I've already said it's a hard beginners operating system, but, perhaps that's an advantage. Personally, I don't think it's possible to combine user friendliness, and powerfulness in a computer. If you want the raw power, you still have to understand a few advanced commands. There is a very good GUI, The X window system, but it's a bit hard to setup..

2.2. Is there any software for Linux ?

This is another general misunderstanding; people think since Linux is free, there can't be any good software running on the OS. *Completely wrong !* Quite the opposite, there is a large number of software packages, available for free, with the capabilities of commercial software packages for other operating systems. Most software is distributed as source code, so a small knowledge about programming, especially 'C', is good when you're running Linux. Nowadays, with modern distributions such as Debian and Redhat, there are packages, containing pre compiled software, with installation scripts placing the files at the right place, often asking you questions about how the software should be setup.

Lately, a lot of commercial packages has been ported to Linux. Including software like Oracle's databases, SAP R/3, Staroffice and Informix databases just to mention a few.

2.2.1. Sure it's nice with free software, but where do I get any support ?

Ask someone in a newsgroup, a fido discussion, an e-mail list or just e-mail the author.. somewhere there is someone who have or had the same problem you have. He or she will most probably help you, for free !

2.2.2. Aren't there any commercial software ?

Sure there is, and it's coming more and more... Check out http://www.linux.org/apps/applications.html for some good examples..

2.3. Some words about partitions, Linux way of accessing (E)IDE disks, Linux way of using partitions and some other talk..

When newbies ask me about installing Linux, there's a lot of confusion about partitioning your HD, and how Linux access your HD and so on.. I'll try to explain some of it..

2.3.1. What's all this about partitions, boot sectors etc ?

If you're going to understand the text below, you should know what a primary and extended/logical partition is. You should also know how Linux names partitions located on different drives.

This applies to (E)IDE drives, the controller type sitting in most PC's. I'll check out how SCSI works, could someone help me with the information.

2.3.1.1. Linux way of accessing hardware.

In Linux, and most UNIX operating systems, you access almost all the hardware by reading/writing virtual files, mostly located in the /dev directory. This might seem a strange method, but when you get used to it, it is really natural.

2.3.1.2. (E)IDE controllers, and partitions on hard disks

Every IDE controller can have 2 disks, one master and one slave. In new home computers of today, there's often a (E)IDE controller, which provides the possibility of attaching 4 drives. Every drive can have four primary partitions, and a larger number of logical partitions. A partition is a piece of the hard disk, with a size you decide.

2.3.1.2.1. The difference between Primary and Logical partitions

With DOS/windows and a few other operating systems, you can only see one primary partition at one specific moment. Running DOS or windows, they are named C:. There can be four primary partitions, a PC limit as stupid as the 640 kb DOS memory limit, there should be no foreseeable limits in the soft/hardware you use.. that's my opinion. Anyway, due to information I have, Linux can see all primary partitions at one time, and use them.

There can be a larger number of logical partitions. An operating system can see several logical partitions at once, making it possible to access them all at the same moment. This is why one should use logical partitions where possible. Some operating systems must have a primary partition to start from, including DOS, and windows 3.xx and 95. I don't think that's clever.

2.3.1.3. The device names of different hard disks/partitions in the Linux file system.

So, now when we know the difference between a master and a slave, and a primary and a logical partition, we can understand the system which Linux uses to name the different drives/partitions in it's file system.

the base name for a (E)IDE controlled disk is /dev/hd*?*. The *?* is a single letter. The system goes like this:

- /dev/hda - master disk on first controller
- /dev/hdb - slave disk on first controller
- /dev/hdc - master disk on second controller
- /dev/hdd - slave disk on second controller

You get it ?? GOOD ! Now, naming the partitions is quite easy, you just attach a number to the device. Look below for some partitions on /dev/hda

- /dev/hda1 - master disk on first controller, first primary partition.
- /dev/hda2 - master disk on first controller, second primary partition.
- /dev/hda5 - master disk on first controller, first logical partition.
- /dev/hda6 - master disk on first controller, second logical partition.

You get that too ?? VERY GOOD ! Not ?? Read it once more :-)

2.3.2. OK, now I understand what a partition is.. how do I use it ? How do I access C: ? (Or, the mount thing)

Linux doesn't present the user with stupid drive letters, it has a much more intelligent way of doing that.

2.3.2.1. A mounted file system..

The advantages of a mounted file system is many.. You decide where you want the space, you can mount different file systems without problems..

2.3.2.2. How does it work ?

Well.. first you need a root file system, the base of all directories, named '/'. The directory separator in Linux/UNIX is '/', not '\' as in DOS.. during system startup you tell the kernel you want as an example /dev/hda5 as root file system. After that, you can tell the kernel, you want /dev/hda1 as /dosc and you access that drive exactly like any other directory. This way, you can mount just a few directories from a local hard disk, and keep shared volumes with large amounts of data at a central server. There's a protocol for sharing files over a network, the Network File System, and you won't notice the difference accessing that directory, from accessing a directory at your hard disk.

I hope you understand.. perhaps not a good explanation.

2.3.2.3. Two examples

To mount your floppy disk, located at /dev/fd0 and formatted with the MS DOS filesystem at /floppy you'll have to execute `mount -t msdos /dev/fd0 /floppy`. The `-t msdos` tells mount what filesystem to expect, sometimes you don't have to give this, it'll find out itself. Don't forget to `umount /floppy` before removing the floppy.

To mount your CDROM, let's say it's a ATAPI CDROM located at the second EIDE controller as master, you'll execute the following command. `mount -t iso9660 /dev/hdc /cdrom`. Make sure /cdrom exists, otherwise it won't work.

2.3.3. Using MSDOS floppies - the easy way

Sometimes you have to exchange data with users not running Linux. In such situations MSDOS floppy disks are quite useful. Now, how do you easily access those disks from Linux? Mtools (http://mtools.linux.lu/) is the answer! It allows formatting, copying and deletion of files on MSDOS formatted floppies. All this without mounting the filesystem, though you can of course mount a MSDOS filesystem from within Linux if you want to.

2.3.3.1. Formatting a MSDOS floppy

The first thing you might want to do when you need to use MSDOS disks, is to format it. This is done using the `mformat` command.

Finding the correct parameters to feed to mformat can be tricky. The best way is probably to use the `minfo`program with an existing (already formatted floppy). As an example, here is what minfo thinks one of my MSDOS floppies looks like

```
hostname:erik ~ % minfo a:
device information:
===================
filename="/dev/fd0"
sectors per track: 18
heads: 2
cylinders: 80

mformat command line: mformat -t 80 -h 2 -s 18 a:

bootsector information
======================
banner:"(k’\\IHC"
sector size: 512 bytes
cluster size: 1 sectors
reserved (boot) sectors: 1
fats: 2
max available root directory slots: 224
small size: 2880 sectors
media descriptor byte: 0xf0
sectors per fat: 9
sectors per track: 18
heads: 2
hidden sectors: 0
big size: 0 sectors
physical drive id: 0x0
reserved=0x0
```

```
dos4=0x29
serial number: B03212D4
disk label="NO NAME    "
disk type="FAT12   "
```

As you can see, it gives the correct parameters to format a floppy disk the same way using mformat

2.3.4. Moving, Copying or deleting files on a MSDOS floppy

Copying and moving are quite similar operations. Use the `mcopy` and `mmove` commands. The syntax is what you would expect. Ie `mcopy *.exe a:` works just the way copy would do under some DOS clone.

Deleting files is done with the `mdel` command. 'nough said :-)

2.4. Some other commands for the beginner

On request from people visiting this page, I'll list some commands the new user might want to know. Most of this is also availiable in the DOS to Linux HOWTO, check out the Linux Documentation Project (http://sunsite.unc.edu/LDP/) for HOWTO's and other very interesting material.

Table 2-1. Some useful (?) Linux commands.

DOS Command	*Linux Command*	*Comment*
dir	ls	List files in the current directory
dir	ls -l	List files, now with user right details and dates and other perhaps interesting stuff.
Not present in DOS	ps	Lists processes you've started. Give it the argument *aux* for all the processes on the system.
type	cat	List a file
more	more or less or most or..	List files longer than the screen length. more is standard, less is very common, most is not as common.
help	man or whatis or apropos	Commands to learn more about the system. run man man sometime.
fdisk	fdisk	A very funny game, don't play it too often =)
chkdisk	fsck	Fsck is often run automatically when it's needed, at bootup.
nc	mc	mc is a very good Norton Commander clone.

I can't remember any more dos commands =)

2.5. More talk about commands and programs

For the new User, it might be confusing finding out what files are programs and which are not. Under MS-DOS or Windows, all executable files are named *.exe, *.com or *.bat . This is not the case under Linux (or other Unices). Instead, files are executable if they have the executable bit set in the file rights. Read more about this in the Section 2.6. This is also the basis for shell scripts about which you can read in the Section 2.10 . Creating a shell script is in brief just a question of deciding what interpreter you want, writing some commands and then execute a **chmod**command to make it executable.

2.5.1. I just installed a program, but now I can’t find it. Help!

When you install a program using either some package tool (RPM or DEB for example) or by compiling it by hand it will most often be placed some directory in your path. Se Section 2.9 for some discussion about the PATH and other environment variables. This makes it possible for you to execute the program just by writing it’s name in a shell.

If the program was a X window system program, chances are great that your package system installed it in the menus of your window manager. Try restarting your windowmanager and browse the menus to see if you find the program. It might be there even if it isn’t a X program - most programs can be run in a xterm.

2.5.2. I don’t remember the name of a command. Help!

Since Linux is used by users that know that computers can do a lot of things more convenient for the user. One example of this is the command completion feature most shells have. For example, pressing a single letter and one or two tabs will in most shells show you all commands in your path that begins with that letter. This makes it easier to find a command that you know start with some characters.

The manual pages are also very helpful. Try running the apropos command with some word as parameter. **apropos file** for example will show you all the commands that have to do with files in some way. That’s a lot of commands..

2.6. Dealing with user rights in the filesystem.

Linux, like any advanced operating system, has a access control system for the file system. That means, not all users can do what they want with a specific file. Every file has a owner, a group owner, each with their respective right to do certain things on that file, and rights for the rest of the world. If you do a normal ls -l in some directory, you’ll get something like this.

```
-rw-r--r--   1 erik     erik          444 Feb 14 22:24 Makefile
-rw-r--r--   1 erik     erik         3507 Feb 14 17:44 erik.html
```

This might look cryptic the first time you see it, but after a while it’s as natural as it can be. I’ll try to explain the fields.

2.6.1. The rights for this file.

`-rw-r--r--` describes the rights the *owner, group and everyone else* has on this file. As you can see, there is 10 characters, the first one tells the type of the file, if it’s a directory there is a ’d’ instead of a ’-’, there is a few other types, but this is what you need to know right now. After the classifying character, there is three groups of characters, each of them with three characters in them. They describe the rights for the owner, the group and everyone, one character group for each of them. The basic rights are *Read Write Execute*. So, the string `rwx` says right to read, write and execute the file. If you have execution right on a directory, you may cd into it. Anyway, a string like the one in the example `-rw-r--r--` means that the owner of this file has read and write rights, the group has only read and the so has the rest of the systems users. Of course, root can do whatever with the file too, since root is God in the UNIX world (Root really can do everything, like deleting your /bin directory, ask me, I know...). The other fields in the listing says how many links to this file there are, who is the owner, what group it is in, the file size and date and the filename.

2.6.1.1. What is a group ?

Perhaps I should explain that too.. As you probably know, you have a username, and most of your rights depend on that username. Sometimes it's practical to have common rights for some users, per example 5 people developing the same program should have read/write permissions to the source code, but everybody else shouldn't. The sysadmin of the system may setup a special group for those programmers, and then they can happily share their files.

2.6.2. Changing the rights for a file...

So, now when you know what the cryptic fields mean, you'd probably like to change them, wouldn't you ? The magic command is `chmod` with the correct parameters given. There is two ways to tell chmod what rights you want for a particular file (or a bunch of files specified with a wildcard);

2.6.2.1. ...The character way or...

If you prefer this way, you tell chmod what you want by combining characters, basically *u, g, o and a* is used, they tell chmod for whom you want to change rights, u for user, g for group, o for others and a for all. You combine that character with another character specifying what right to change, *r, w and x* most commonly used. r for read, w for write and x for execute. As an example, if you want to add execute rights for the user owning the file named file1 you execute `chmod u+x file1`. Removing the groups write rights on the same file would be `chmod g-w file1`. Really logical, isn't it? Make a dummy file and experiment, and read the man page `man chmod`.

2.6.2.2. ...The numerical way

This is the way I use, since I've grown up with it ;). You tell chmod what you want by giving it numbers, one for each of *user*, *group* and *others*. You calculate the numbers by looking at the read, write and execute flags as binary bits. *rwx* would be a *7* if you look at an enabled r as a 4, an enabled w as a 2 and an enabled x as a 1. *rw-* is 6, since 4 (enabled r) + 2 (enabled w) + 0 (disabled x) = 6. You get it ? Not ? Use the character way :) Anyway, if you combine one number for the user, one for the group and one for others, you'll get three numbers. So, for *rwxr-xr-x* you'd do `chmod 755`. There's probably about a million better ways to describe this, but..

2.7. Booting Linux with LILO

When you start using Linux, you probably want your old operating system left there, to use if you get struck in some problem. You won't need another operating system because Linux can't do something. Linux can do everything, it's just a matter of configuration and time :-) Anyway, if you want your old system as a backup, a smart way of doing this, is to install Linux to another partition, and use LILO to choose the operating system you want to run at the moment, at system boot. For more detailed information than this text gives you, check the lilo man page. 'man lilo' and 'man lilo.conf'

2.7.1. How LILO works

LILO (*LI*nux *LO*ader), places itself in a boot sector of your hard disk. It could be the MASTER boot record of your hard disk, ie the first sector of the disk, making it execute as the first thing the BIOS does after checking things are OK. It could also be a partition boot record, ie the start of a partition, making it execute when another boot loader, perhaps the OS/2 Bootmanager, passes the control to it. This way, you could have a more fancy boot loader there for your mother to choose from, and then LILO will do the Linux boot.

LILO has a configuration file, called /etc/lilo.conf, where you tell it how to behave. I'll give you an example of such a file in the next section. After you've written your lilo.conf, run the command 'lilo' and it will write the boot sector, or give you an error message if you've done something wrong. It's really important that you have a boot, because if you do something wrong you have to edit the lilo.conf file, and try doing that when you can't start Linux ! Installing some operating systems (Win95 for example) will erase the information lilo writes to the Master Boot Record, and you'll have to boot Linux with a boot disk to be able to reexecute lilo to rewrite the information.

2.7.2. An example lilo.conf file, for a computer with win95 and Linux installed.

I'll give an example here, basically copied from the lilo man page, how to setup lilo for a computer with windows 95 on the first primary partition, and Linux on the first logical partition. All text after a '#' is comments.

```
# example /etc/lilo.conf
boot  =  /dev/hda     # place lilo in MBR of first disk
delay = 40            # Wait 4 seconds for user to press Ctrl or Shift
                      # This way your mother won't have to do anything to
                      # boot her windows 95
other = /dev/hda1     # The windows 95 partition
    label = windows
    table = /dev/hda

image = /boot/zImage-2.0.33 # A Linux kernel, located in the /boot directory
     root  = /dev/hda2     # The partition that will be mounted as root
     label = linux         # The name you should type in to boot this kernel.
```

After entering that into your /etc/lilo.conf, run the command `lilo`. If everything is what it should be, it should respond with `Added windows * Added linux` The star indicates that windows is the default alternative, that will boot if you do nothing.

When you reboot (remember having a rescue disk if something goes wrong), lilo will print 'LILO' at your screen, wait for seconds for you to do something, and then boot windows 95. If you push down shift or Control, LILO will give you a prompt, where you can enter 'linux' and push enter, then it will boot the linux kernel you've specified. I hope it'll work for you !

2.8. Virtual Terminals - VT's

OK, so now you sit there with your Linux prompt, doing nice things.. try pressing Alt+F2.. Wow, a brand new login prompt ! Try pressing ALT+F3.. Another one. Press Alt+F1, back at the old one.. this way you can do several things at once, logging in as yourself at tty1 and perhaps as root on tty2, this way making your system administration effective. Usually there are 6 *Virtual Terminals*. This might seem obvious, but I'm writing this document with the new user in mind.

2.9. Environment variables

Perhaps good to know is how bash handles environment variables. This is bash specific I know, but as I've said before, bash is the most common shell. I'd like to have information how other shells does.

2.9.1. What is an environment variable good for ?

It's a fast way of accessing some short configuration options from every program. It's easy to change during a session and is easy to view.. or something, well, they exist and it works quite well..

2.9.2. How do I set/unset an environment variable, and how do I view them ?

In bash, you use export `<variable>="<value>"` (example: `export PATH="/bin:/usr/bin"`) to set a variable. To remove it completely you use `unset <variable>`. With `printenv` you can see all the variables.

2.9.3. What more or less important variables are there ?

There are a lot of them, some of the most common being

Table 2-2. Some environment variables

Variable	*Meaning*
PS1	Let's you decide how your prompt should behave
PATH	Defines the PATH (surprise ! :))
LOGNAME	Shows which user is logged in on the current terminal
HOSTNAME	Contains your hostname
PAGER	Contains the local pager, to view textfiles
$$	Gives the Process ID of 'this' process.
$?	Gives the errorlevel of the last command. Useful in shell scripts
$0	The name of the command executing right now, also useful in shell scripts
$1	The first cmdline parameter to 'this' process.
$2	The second...

2.10. Shell scripts

Another thing good to know is how to write shell scripts. They are commonly used in the setup and in the automation of system tasks. In DOS and similar OS’s there are a simple shell script in the batch files, with .BAT filename extension. The writing of Linux shell script is similar, with some enhancements.

2.10.1. Type in a test script

To begin, type in the following with your favourite editor, and save it with filename "first" in your home directory.

```
#!/bin/sh
# This is a comment
# The first script I’ve ever created.
echo -n "My first script runs at "
date
```

2.10.2. Make the script run.

To make Linux understand it’s a shell script, do a chmod on the file. Do a `chmod 755 first` at your prompt (if you don’t understand what that means, go read the Section 2.6). If you have colour support in your ls, the file will have another colour.

Now type `first` and watch the output. It should say something like this:

```
My first script runs at Sun Feb 15 02:33:35 CET 1998
```

(Of course it should give you the correct date. If you don’t have the correct date, use xntpd. Read Section 2.16.2.10.

Well, actually it’s lying, since this isn’t my first script, not at all :-)

2.10.3. What did we do ?

Now let’s check out what the cute little script did...

2.10.3.1. Beginning of file

A shell script always starts with something like this:

```
#!/bin/sh
```

Telling your shell what program to execute to parse the script. /bin/sh should be availiable on most Linux/UNIX systems, on my system a symbolic link to /bin/bash.

2.10.3.2. Comments

Lines beginning with '#' are considered comments. Always comment your scripts, otherwise you won't remember after a month why you did things that way. Try to keep version information and date of last modification there too.

2.10.3.3. Commands

In the shell script you can use commands just like you would type them in at your prompt. Two examples of that are in the example script, *echo* and *date* they are both examples of small unix programs doing their specific tasks very well.

2.10.3.3.1. echo

echo does what it's named.. echoing something to the screen. If you give it the -n variable (as in the example script) it won't put any newline after it's output. There are few other parameters, check out the man page.

2.10.3.3.2. date

Prints the current system time and date.

2.10.4. if, for, case

Bash of course has support for these..

2.10.4.1. if

This is one of the most useful commands, when you want to check if a file is where it should be, if a command executed successfully, and whatever else you usually use a if command for. Type in the following example in second and `chmod 755` it.

```
#!/bin/sh

ping -c 2 localhost
if [ $? != 0 ] ; then
    echo "Couldn't ping localhost, weird"
    fi

ping -c 2 veryweirdhostname.noend
if [ $? != 0 ] ; then
    echo "Surprise, Couldn't ping a very weird hostname.."
    fi

echo "The pid of this process is $$"
```

(Yes, I do believe a lot in examples :)) OK, let's check out what we're doing. `ping` is a net command that checks if a host is up and running, and how long time it takes for a packet to travel the way to the host. If it finds the host and succeeds pinging it, it will leave an errorlevel of 0, else it will leave another, higher errorlevel. That's common practice, if something goes well, it usually leaves errorlevel 0, else it gives something else.

2.10.5. for

For is useful when you want to do something on every file in a directory, the following will do a `file` on every file with a .sgml extension in the current directory. `file` is a very useful command trying to determine what type a file is.

```
#!/bin/bash
for sgmlfile in *.sgml ; do
    file $sgmlfile
    done
```

I’m sure you can find a better example, please mail it :)

2.10.6. case

Case is very useful when you want to give your shell scripts commandline parameters. Almost all scripts in the SysV init dirs has start, stop and perhaps restart as possible parameters. That’s done this way;

```
#!/bin/sh

case "$1" in
    start)
    echo "Parameter was start"
    ;;

    stop)
    echo "Parameter was stop"
    ;;

    reload)
    echo "Parameter was reload"
    ;;

    *)
    echo "Usage: ‘basename $0‘ {start|stop|reload}"
    ;;
    esac
```

This example also shows how to include a command in a string, very useful feature of UNIX. Put a command inside two "‘" and it’s output will be inserted. In this case, `basename $0` removes any directories from the name of the script. So if the script is named /scripts/third, it won’t say "/scripts/third {start|stop|reload}" but "third {start|stop|reload}".

Linux scripting is wonderful, if you compare with DOS or Windows NT or something similar. There are other more powerful scripting languages than bash’s, for example perl and python, I don’t know too much about either of them, but as fast as I do, I’ll write a small chapter.

2.11. Linux configuration

2.11.1. How ? Global and User settings

Almost all configuration in Linux is made by editing text files located at various places. Often there are one global text file, defining global defaults for all users on your system, and one user specific file, defining settings for a specific user. This way, every user can tweak his settings in every detail, or just use the default configuration made by the system administrator. In my opinion a perfect system !

2.11.2. Where ?

Most global configuration files are located in the /etc directory, and it's subdirectories. Users configuration files are usually located in their home directories, very often as files or directories with names that begin with a '.' You won't see them with a normal `ls`, but with a `ls -a`. Not all programs have user specific files, but userprograms often have.

2.12. The system init files (or, "Where is the autoexec.bat file ?")

(this section is true only for systems with SysV init, as Debian GNU/Linux and RedHat Linux, not Slackware).

When the system boots, it reads some files to setup itself. After the kernel has booted, it starts a process called *init*, with PID 1. Init reads it's main configuration file called /etc/inittab to get the default runlevel, and a few other settings.

2.12.1. runlevels

A linux system has a few different runlevels, each configured for a specific task. A Debian system normally goes into runlevel 2 when it boots, starting up daemons and more to make the system accessible in different ways and by several users at once. If you start the system in runlevel 1, you'll get a singleuser system for maintainance. Setting the runlevel to 0 will do a system halt, and runlevel 6 will reboot the machine after taking down daemons, umounting file systems and other important things before a shutdown. The runlevels I give here as example are specific for a Debian system, a RedHat or Slackware of whatever distribution have different runlevels for the different modes of operation.

2.12.2. Runlevel specific files

For each of the runlevels, there are files that specify what to do when entering that runlevel. Or, to be more exact, the files are links to files in another directory, this way you won't have 6 copies of each file when the file is used in 6 runlevels.

The files have names that start with either *S* or *K*, then a number, and after that a descriptive word. The S or K stands for *Start* or *Kill*, the number stands for when the file is executed, and the rest of the filename is there for uniqeness and completeness. Take the following file list for example, it's from my /etc/rc2.d, ie files to execute for runlevel 2.

```
S10sysklogd     S20gpm          S20ssh          S30netstd_misc  S99xdm
```

```
S12kerneld      S20iplogger     S20timecount    S89atd
S13dhcpc        S20lpd          S20xfs          S89cron
S15netstd_init  S20ppp          S25netstd_nfs   S91apache
S18netbase      S20sendmail     S26firewall     S99rmnologin
```

The file S10syslogd will be executed first, and the file S99xdm will be executed as the last of the files for runlevel 2. This way you may decide when in the boot process something is done.

On a Debian system, the files are located in /etc/rc?.d, where the ’?’ stands for the runlevel. The symbolic links mostly points to files in the directory /etc/init.d. Those files are scripts that take *start* or *stop* as arguments, some of them have *reload* too. When init reads in the /etc/rc2.d and finds the S10syslogd, it will see that’s a symbolic link to /etc/init.d/sysklogd and execute /etc/init.d/sysklogd with the argument *start*. After that it will do the same thing with S12kerneld and so on.. If the filename is K10sysklogd, it will execute /etc/init.d/sysklogd with argument *stop* instead. You get it ? Me to :) RedHat has it’s files in /etc/rc.d/rc?.d and /etc/rc.d/init.d, AFAIK !

2.13. BASH’s init files

Now when we know where configuration files are located, it might be interesting to take a look at the configuration for bash, the most common shell on Linux machines. There are, as there should be, a global and a user file; the global file named /etc/profile and the user file named .bash_profile (well, this is not completely true.. there is other files, .bashrc and .inputrc, but right now I won’t deal with them..).

2.13.1. What is setup in the files ?

Environment variables such as PATH, PS1 and such.. aliases, umask and other mysterious things, perhaps some programs is run to do things you want to be executed at your login..

2.14. Customizing your prompt.

This is a little bells and whistles, but very nice to know..

When you’ve installed your Linux distribution (I recommend Debian), your prompt probably won’t look as you want it. The key to changing this is the PS1 environment variable. This is specific for the common shell bash (I think). You make your own prompt by combining some codes in the variable, just like in the (old good ?) DOS days... here is what you can use:

Table 2-3. Codes to use to format your prompt

\a	Bell (makes Linux beep every time it shows a prompt, not recommended !)
\d	Todays date
\e	ESC character, useful if you want to send ansi sequences
\h	Your hostname, to the first ’.’
\H	Your fully qualified hostname

\n	newline
\s	The name of the shell
\t	The current time in 24 hour format
\T	The current time in 12 hour format
\@	The current time in 12 hour am/pm format
\u	The username of the current user
\v	The version of bash
\w	The current working directory.
\W	The basename of the current working directory (ie /usr/local will be local)
\!	The history number of this command
\#	The command number of this command
\\	A backslash

My own PS1 says "\h:\u \w $", giving me hostname:user working dir $ (I really need all that information, since often I have ssh sessions to a few machines open, and it's REALLY good knowing what machine you currently work at). The best way to set this is in the shell init files.

2.15. Using your mouse - gpm

So far, we've just been using our keyboard, typing in mysterious commands, looking at the screen with a total amazement waiting for response. Somewhere to the right (or perhaps left, if you're left-handed) there's a weird shaped thing with a cable attached to it, doing nothing at the moment. They usually call it a mouse, or pointing device. Can you use that thing in Linux text mode ? The answer is yes.

2.15.1. The mouse server, gpm.

When you installed your Linux, most probably a *mouse server* called gpm was installed and perhaps even configured. Gpm is a daemon watching the mouse, and when detecting movement, showing up a mouse pointer on the screen. Gpm is usually started in the init scripts (remember reading about that earlier in this document ? :)) Command to start gpm could be `gpm -t ms -m /dev/ttyS1` if your mouse is of Microsoft type attached to COM2. If you have a PS/2 mouse, try something like `gpm -t ms -m /dev/psaux`. Anyway, if your mouse server isn't working, read the manual. `man gpm` will probably give some useful information.

2.15.2. OK, it's running, what can I do ?

You move your mouse and a pointer moves on the screen ? Great ! But what is it good for ?

The basic function of gmp is as a cut and paste utility. You have one Virtual terminal open with a file listing, and want to open one of the files in an editor in another Virtual terminal. Mark the file in the first terminal (hold left button down and mark, then release it) switch to the other one, and paste the text with the right button. Very useful, when you know about it, you'll miss it if it isn't there.

gmp can do other stuff to, if you invoke it with the -S parameter, and then hold your right button and triple-click you left, a message will appear.. if you press one of the buttons again within three seconds, your machine will reboot. Very handy for programmers if the keyboard locks, and he can’t telnet in to do a clean reboot. It’s possible to specify what should be done instead of rebooting.

2.16. Daemons ?

A Linux system, as any UNIX system, has quite a few daemons hanging around, doing nice little things for you. This section will try to explain what a daemon is.

2.16.1. What’s a daemon ?

A daemon is a program executing in the background, doing a specific task, usually something important for the system. They are most often started at system boot, and stopped at system halt, running all the time the system is up. Normally, they don’t interact directly with the user, instead you tell them how to behave with a configuration file. They are most often well written, so they won’t consume much CPU power when they are just waiting for something to happend.

2.16.2. What daemons are there ?

I’ll describe some of the most common daemons, in no particular order.

2.16.2.1. inetd

The "internet super server" is one of the most common daemons. At startup, it reads it configuration file /etc/inetd.conf and then starts listening for incoming internet connections. When someone tries to access your system with, as an example, the telnet protocol, inetd checks it’s configuration to see what to do if a telnet request comes, and executes the correct program.

2.16.2.2. crond

Crond is a very useful daemon, making it possible to execute things at a specified time, or at an interval. You submit a configuration file to it, and every minute it wakes up from it’s CPU friendly sleep, and checks if something should be done right this minute. Really useful, if you ask me.

2.16.2.3. syslogd

This daemon takes care of logging information from various programs, and stores them to the correct place. It’s configuration file, /etc/syslog.conf, makes it possible to have logs from pppd in one file, and logs from sendmail in another. It’s also possible to have the log on a VT, or even on another machine via TCP/IP.

2.16.2.4. klogd

Takes care of messages from the kernel.

2.16.2.5. atd

A daemon waking up every minute to see if something should be done at that time. Kind of similar to crond, but a job submitted to atd (with the command *at*) is only executed once, while a job maintained by cron is executed periodically.

2.16.2.6. rpc.portmap

This daemon acts a little like inetd for SUN RPC calls such as NFS. Sounded like greek ? Read the nfs man page :-)

2.16.2.7. sshd, httpd, nntpd..

These are various internet services, such as secure shell, http (web), nntp (news) running as separate daemons instead of being started by inetd. This makes them a bit faster, but they consume more memory/CPU.

2.16.2.8. gmp

See Section 2.15, but basically handling your mouse in consolemode.

2.16.2.9. xdm

Handling X server, remote or local. If you have a fancy login prompt in X, this is the daemon handling that.

2.16.2.10. xntpd

Synchronising your time with a distant time server, keeping your clock within milliseconds to world time. This is very important in environments where you share files via NFS.

2.17. There are so many directories, I don’t understand the file structure !

The directory structure in Linux most often has the same structure as any UNIX. There’s a specification specifying how it should be built, the FHS. I’ll try to explain what the directories under ’/’ does.

/bin	This is where the system stores important executables, needed for the startup of the system.
/boot	The kernel is often placed here, together with some other boot files, this directory should be on the first 1024 cylinders of the disk, or the disk should be LBA translated, otherwise LILO can’t boot the kernel.
/cdrom	This is a directory I have, as a mount point for my CDROM reader.

/dev	This is a really interesting directory :). In there, every device on your computer (serial ports, hard disks etc..) has a file, a device file. When you write to the file /dev/ttyS0, the kernel takes care of what you write, and write it to the first serial port of the system. This is a very nice system, since everything in the system can be treated as files.
/dosc	This is a common name for your DOS *C:*. On every system I've seen, the DOS primary partition has been mounted as /dosc. Don't ask me why !
/etc	Global configuration files, often has some subdirs, like /etc/ppp for your dialup PPP internet configuration.
/floppy	My mount point for floppydisks.
/home	Users's home directories, every user has a directory in this directory.
/lib	Shared libraries, important for the system startup. Share libraries are used by many executables at the same time, saving memory and diskspace.
/lost+found	If you run a fsck, and it finds some weirdness, the result goes here.
/mnt	This is another non-standard, but common directory. It exist there for temporary mounts, ie if you want to mount some partition you normally not mount, you can mount it here.. Well, that's at least how I use it.
/proc	This is a really important directory. It exists only as a mount point on the harddisk, the rest is virtual information filled in by the kernel. Here you find information about all your processes, and other information such as which interupts and memory adresses are in use. An easy way to fetch that information, it's just files, as the rest of the UNIX system.
/root	This is the home directory of the root user.
/sbin	Executables the root needs to use, that should be availiable at boot.
/tmp	Temporary files. Everybody can write here, but only the owner of the file may remove the file. Root may do it too, of course.

/usr	This directory has a lot of subdirs. There is a /usr/bin, a /usr/sbin and so on. /usr doesn't need to be there directly at system boot, but can be mounted a bit later. That way, you can have a small root partition and a bigger /usr and perhaps /home that you mount later. /usr/bin has the same type of programs in it as /bin, but they aren't that essential for the system to work.One subdir of /usr is /usr/local, also having a bin and sbin and lib directory. In local you can place local extensions of the system, and if you upgrade your system the upgrade shouldn't do anything about local, so your changes will still be there.
/var	var stands for Variable. This directory holds files that could be a lot bigger in a short time, and get smaller really fast too. Typical examples are mail and news spools.

Well, that's about it. Of course this is my version, if you have any feedback, make sure to send a mail to forsberg@lysator.liu.se (mailto:forsberg@lysator.liu.se)

2.18. Setting up a PPP connection to your ISP

A lot of the useful information about Linux is on the net, I think I've said that before :) Most of the people I know (including myself) don't have a high speed internet connection at home, but rather a dialup connection to some Internet Service Provider. The most common way to connect to the net via a ISP is by PPP (Point to Point Procotol). PPP support is included in most common distributions.

Here I'll try to describe a ppp setup that should work for almost all distributions. If you're from Sweden, you might want to check out http://www.lysator.liu.se/~forsberg/ instead, since there you can find this information in Swedish, together with examples for most of the ISP's in Sweden. The original scripts were created by Simon Josefsson jas@pdc.kth.se (mailto:jas@pdc.kth.se).

2.18.1. Basic setup.

As a start, your kernel must have ppp support, and you must have a pppd that isn't too old. Try to execute the command 'pppd' at a root prompt, you should get a lot of funny characters, not the message "This kernel lacks kernel support". I you get the message, something is wrong ! =)

By the way, the funny characters stops after a short while, just be patient.

2.18.2. Files.

Start by creating an empty file named /etc/ppp/options. This can be done by executing `:> /etc/ppp/options` at a root prompt. Make a symlink named /dev/modem to the com port your modem is placed at, that is /dev/ttyS0 if your modem sits on COM1, /dev/ttyS1 if it's connected to COM2.. Do this by executing `# ln -s /dev/ttyS0 /dev/modem` . First, make sure the link doesn't exist, your distribution may have done that work at installation.

2.18.2.1. /etc/ppp/on

Save this as /etc/ppp/on (If read this online, you may retrieve the script by clicking here (on)

```
#!/bin/sh

# PPP-script made by Simon Josefsson <jas@pdc.kth.se>.

tmpfile=/tmp/ppp.script.$$
source /etc/ppp/provider
export TELEPHONE LOGIN LOGINSTR PASSWORD PASSWORDSTR PPPCOMMANDSTR PPPCOMMAND

if test -n "$USEPAP"; then
        PAP="name $PAPLOGIN remotename $USEPAP"
fi

umask 0177
cat<<EOF>$tmpfile
TIMEOUT         5
ABORT           BUSY
ABORT           "NO DIAL TONE"
ABORT           \nRINGING\r\n\r\nRINGING\r
ABORT           "NO CARRIER"
""              \r\rATZ
OK-+++\c-OK     ATH0
TIMEOUT         60
OK              ATDT$TELEPHONE
CONNECT         \n
TIMEOUT         40
$LOGINSTR       $LOGIN
$PASSWORDSTR    $PASSWORD
$PPPCOMMANDSTR  $PPPCOMMAND
EOF

exec /usr/sbin/pppd -detach debug lock modem crtscts /dev/modem 57600 \
        noipdefault netmask 255.255.255.0 defaultroute $PAP \
        connect "/usr/sbin/chat -v -f $tmpfile" disconnect "rm -f $tmpfile" &
```

Make it executable by executing `# chmod 755 /etc/ppp/on` . By the way, if your telephone switch don't understand tone dial, you should use ATDP instead of ATDT in the string "ATDT$TELEPHONE" above.

2.18.3. /etc/ppp/off

Save this as /etc/ppp/off (Or, get it by clicking here (off))

```
#!/bin/sh
if [ "$1" = "" ]; then
        DEVICE=ppp0
else
        DEVICE=$1
fi
if [ -r /var/run/$DEVICE.pid ]; then
        kill -INT ‘cat /var/run/$DEVICE.pid‘
        if [ ! "$?" = "0" ]; then
                rm -f /var/run/$DEVICE.pid
                echo "ERROR: Removed stale pid file"
                exit 1
        fi
        echo "PPP link to $DEVICE terminated."
        exit 0
fi
echo "ERROR: PPP link is not active on $DEVICE"
exit 1
```

Do the same chmod command to this file, as you did to the file named ’on’.

2.18.4. Nameservers, /etc/resolv.conf

If you want to be able to use the internet by typing names (like http://www.lysator.liu.se/~forsberg/) instead of numbers (like 10.38.0.18) you have to tell the linux resolver library how to get the information what number correspond to which names. This is done by writing nameserver statements to the file /etc/resolv.conf. Get ip adress of your nameserver from the information your ISP sent you, and write something like this in /etc/resolv.conf

```
nameserver 10.38.0.18
```

It’s possible to have up to three nameservers, they will be tried in the order they are written.

2.18.5. ISP Specific settings

Here comes the hard part :-) Now you must give pppd ISP specific settings, writing some words in /etc/ppp/provider and, if the ISP uses PAP or CHAP, a few words in /etc/ppp/pap-secrets or /etc/ppp/chap-secrets respective.

Here is a list of the statements you may use, with descriptions.

TELEPHONE=<phonenumber>	The telephone number to your ISP’s modem pool.
LOGINSTR=<login string>	The string that should trigger sending your user name, for example "ogin:--ogin:" works quite well in quite a few cases. PPPD will wait for the string ogin: and then send your account name.
LOGIN=<account>	Your account name.
PASSWORDSTR=<password string>	Similar to the LOGINSTR, this tells pppd what to wait for before sending your password. "assword:" will probably work for many systems.
PASSWORD=<password>	Your password.
PAPLOGIN=<account>	If your provider uses PAP or CHAP, put your login name here, instead of in the LOGIN keyword.
USEPAP=<providername>	If your provider uses PAP or CHAP, put the name of your provider here, and put the same name as providername in the secrets file, described later.

2.18.6. The secrets file.

If your provider uses PAP or CHAP, a line in the /etc/ppp/pap-secrets, or chap-secrets should be added. It should look like `<account> <providername> <password>` where account and providername is the same as you put in the /etc/ppp/provider with the PAPLOGIN and USEPAP statements.

2.18.7. Some examples of /etc/ppp/provider and /etc/ppp/pap-secrets

2.18.7.1. Provider not using PAP (or CHAP)

/etc/ppp/provider

```
TELEPHONE=555112233
LOGINSTR=ogin:--ogin:
LOGIN=mystupidaccountname
PASSWORDSTR=assword:
PASSWORD=MyTooEasyPassword
```

2.18.7.2. Provider using PAP (or CHAP)

/etc/ppp/provider

```
TELEPHONE=555221133
PAPLOGIN=mystupidaccountname
USEPAP=providername
```

/etc/ppp/pap-secrets

```
mystupidaccountname    providername   MyTooEasyPassword
```

Well.. I hope this helps.. =)

2.19. Doing things periodically - Using CRON

On a UNIX system there is always some things that should be done periodically, for example backups, locatedb updates and such things. This makes life easier, as you don't have to keep all the things you should do in your head, or on silly little notes on your desk. Not to mention the fact that you don't want to sit up all night doing backups.

2.19.1. The CRON daemon

The nice program that makes this possible is the cron daemon. It runs all the time the system is up and every minute it wakes up to see if there is something that should be done. If there is, it spawns a shell that executes some command. I'll describe the most used cron package on Linux systems, Vixie Cron. Check out http://www.vix.com for some more interesting software.

2.19.2. Crontab’s

To instruct the crond when to do what, you use crontabs. Like many other programs, cron has a system and a per user crontab. They both have a similar format, with some small differences. What you do in the crontab is to give the minute, hour, day of month and/or day of week a specific command should be run.

2.19.2.1. The system crontab

The system wide crontab is most often called /etc/crontab. This file should have only important jobs, the file is only writable by root. Mine looks something like this:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don’t have to run the ‘crontab’
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/etc/ppp

# m h dom mon dow user command
44 5 * * * root run-parts /etc/cron.daily

07 5 * * 7 root run-parts /etc/cron.weekly

00 5    1 * *   root    run-parts /etc/cron.monthly
```

As you can see, there is seven fields with information. They are, in correct order: minute, hour, day-of-month, month, day-of-week, user and command. Some of the fields contain stars (*) which means any value will fit.

The first line of my file tells cron that every day (the fields day-of-month, month and day-of-week have stars, so cron doesn’t care about them) at 05:44 in the morning I want the command "run-parts /etc/cron.daily" to be run. The command run-parts checks a directory for files that are executable by the user specified in the user field. It runs all of them, in some kind of order :) This is a good way of getting a lot of small things done without writing a big script. Makes it a lot easier maintaining the jobs. My /etc/cron.daily have the following files:

```
calendar*   standard*   sysklogd*   find*     man* netbase*    smail*
tetex-bin*
```

So every morning at 05:44 my computer starts doing things like rotating my logs, updating the manual index files, updating the updatedb database and other things that is quite good if you do once a day. At 05:44 I usually don’t use my computer anyway (I do actually sleep sometimes ! ;)) so these jobs don’t use any CPU - time I could need.

The other lines work just as the first, executing things every week and every month. There is a user field, if you put another user than root there, it will run the command with other rights than root. That is often recommendable by security reasons.

2.19.2.2. The per - user crontab

Every user on the system may write his own crontab. The job will be executed even though the user is not logged on. The format is similar to the system crontab, though there is not user field, since a user’s cron commands will always be executed as that user. My personal crontab looks like this:

```
10 5   * * 0 futility tool "+delete" "keep+1300" ; futility pack
```

```
5 3 * * * /home/erik/bin/mirroring
15 0-23/1 * * * /usr/bin/fscan
```

As you can see, it has three jobs. The first one goes every Sunday (Day of week field has the value 0, which is Sunday. I could write ’Sun’ there too) and does some packing of my fidonet message base.

The second goes every day at 5 minutes past 3 in the morning and mirrors some ftp archives.

The third executes 5 minutes past every hour 24 hours a day. The syntax 0-23/1 tells cron it should do it once an hour from hour 0 to hour 23. 0-14/2 would mean twice an hour from 00:00 to 14:00 in the afternoon.

2.19.3. The crontab command

When a user wants his own crontab, he has to install it. This is done with the command *crontab*. The syntax is quite simple, *crontab name-of-crontab* will install the file you specified as your new crontab. The command crontab has some other switches, to list, edit or remove crontabs, but the basic syntax is actually all you need to know - If you want to edit you crontab, do so and reinstall it with a new crontab command.

2.19.4. Environment variables and file rights.

There are a few things one should think of when trying to master cron. First cron doesn’t read your login files to set the path you’re used to, so a command that executes just fine when you test it may not run at all because cron can’t find some file. There are two solutions to this problem, the first is to use only absolute filenames (ie give the whole path to the files), the second to include something like

```
PATH = /usr/local/bin:/home/foo/bar
```

in your crontab. That way you set the environment variable PATH, so your scripts will find the files.

Another environment variable of interest is MAILTO. If you set that variable (the same way as you set PATH) cron will mail the output of your commands to the address specified. Cron will by default mail the output to the user the crontab belongs to, but if you want the mail to go somewhere else you may specify it in the variable. You may also give an empty MAILTO variable (by writing "MAILTO=" in your crontab). Cron won’t mail any mails to an empty address..

File rights can sometimes be a problem. If cron doesn’t want to execute your scripts or programs, try giving the group or world execution rights to the file. That may solve the problem. The irritating thing is cron won’t tell you it can’t execute, it just sits there doing nothing.

I hope that will help you in further cron adventures =)

2.20. Accessing a computer in a secure way - SSH

One of the really good things with UNIX computers is you can access them the same way wherever they are located. I can do the same things on my local computer, as I can do on a server in the US, by an interactive command line session. I can even run X window system programs on another computer, and get the window on my local screen.

2.20.1. telnet, rsh, rlogin

The old way of doing this is to use telnet, rlogin or rsh, all of them unencrypted standard protocols availiable at most UNIX computers. Sure they work, but you send your password in plaintext, not to mention all the data you enter during your session. That's not good, since network security is getting more important all the time.

2.20.2. ssh

There is an alternative. Secure Shell, developed in finland, encrypts everything you send, including your password. It has even more features, it can forward ports, and forward X window system connections. It also has quite a few ways of authentification.

2.20.2.1. Where do I get it ?

Most major distributions have a ssh package, though you may have to look in some special directory, since the US government by some reason doesn't permit export of programs with encryption, therefore those files has to be kept outside the US. Don't aks me why they do this, I can't find a reason. If you can't find a package, get the source from The ssh communications security (http://www.ssh.com), the company who developed it.

2.20.2.2. Version 1 or 2 ?

There are two major versions of the ssh protocol. Version 1, and version 2. Most systems in use runs version 1, it works well. The second version hasn't been out for a long time, and it has a license that gives some problems. Be sure to read the license for the ssh before you use it. Anyway, using version 1 works good.

2.20.2.3. Server and Client

The ssh protocol needs its own server, sshd, running. Well, it may be run from inetd, but it works best if it runs as a daemon.

To connect to a computer with sshd running, you use ssh, the client. You type something like *ssh hostname*, type your password, and gets connected.

Run ssh without arguments to see what command line parameters there are. One of the most used is -l which lets you specify your username on the remote machine.

2.20.2.4. Authentification

Ssh has a lot of ways to authenticate you to the server. The most basic one is by password, just the way you would do if you used telnet. The difference is your password is never sent in cleartext, it's encrypted, so a person listening to the network where you send your data can't fetch your password.

The second way is by RSAhost authentification. If the host you are connecting to has the host you are connecting from listed in either it's /etc/hosts.equiv or /etc/shosts.equiv or in .shosts or .rhosts in your home directory on the server host, it will permit your login if it recognizes the client hosts host key. A host key is uniqe for every single machine, and every time you connect to a new machine that machines host key is stored in the file $HOME/.ssh/known_hosts. This way of authentification closes security holes due to IP spoofing, DNS spoofing and routing spoofing.

A third way of authentification is a RSA based public-key technology. You generate a pair of keys (one private and one public) with the command *ssh-keygen* and place the public part of it in the file $HOME/.ssh/authorized_keys on every host where you want to login using this type of authentification. When you try to connect to one of these hosts the server and client finds out you have a .ssh/identity on your host, and a .ssh/authorized_hosts on the server. If they match, you are allowed access. If you encrypted your private key with a password, you'll have to type in that to get access.

A very good way of using RSA based public-key authentification is with the program ssh-agent. That program keeps your personal keys on it's mind, and if you have a passphrase for them you only need to enter them once, with the program ssh-add.

Another major feature is that the program automatically starts a new ssh-agent process on each host you logon. That way you can continue ssh:ing to other hosts, without irritating passwords. Does it sound like something that never happends ? Believe me, it does.

As you've probably foundout, this information is not at all complete, it isn't meant to be. When it comes to security related things, always read the manpages. I've only written enough to get you interested ;-)

Chapter 3. Some other Linux information, perhaps not for the new user

Version: $Id: Chapter3.sgml,v 1.1.1.1 2004/09/19 12:25:13 forsberg Exp $

I'll try to assemble some of my experiences here, dealing with some problems, software I've coped with, and other things..

3.1. Modules, who are they, and how do they work ?

A very nice feature of the Linux kernel is the modules. Modules are pieces of software that can be loaded and unloaded from the kernel at runtime. That way you can save kernel memory (and that's important for small machines, since kernel memory is never swapped out on disk), and keep your kernel clean when you're not working with some of your hardware.

Especially sound and network cards are really good to compile as modules, since sometimes they are a bit hard to configure, and when you have a module, you don't have to recompile and reboot your kernel to configure some irq or io setting. Quite a lifesaver, really !

3.1.1. The tools used.

The tools you use to handle modules are all located in /sbin. They are *insmod, modprobe, lsmod, depmod* and *rmmod.*. Notice the names, and compare them to some often used unix commands such as ls and rm and you'll immediatelly understand what each command will do for you. As always I recommend reading the manual pages. Anyway, here's a short table of what they do:

Command	*Function*
lsmod	List the modules currently loaded. A completely harmless command, even a normal user can run it.
rmmod	Removes a loaded module from memory. Checks dependencies, ie if a module needs another module, the other module may not be unloaded before the first one.
insmod	Inserts a specific module in memory. You may give a path, or just a filename with or without extension. In the latter case, insmod will search some standard module paths to find the module.
modprobe	Inserts a module in memory, but checks dependencies. If you modprobe module A, and module A needs module B to run, modprobe will insert module B and then module A.
depmod	Calculates dependencies between modules needed by the other module commands. This is most probably done by your init scripts at system startup, but some time you may need to execute a **depmod -a**

There is also a daemon, kerneld, that loads and unloads modules automatically. That way, your CDROM driver may be a module, and every time you try to access the device, the kernel will insert the modules needed. Very nice feature! Kerneld is replaced by a kernel process called kmod in the new stable kernel (The 2.2.XXX series of kernels). It has the same functionality.

3.1.2. How do I make modules?

When you compile your kernel, you choose to make things as modules, instead of into the kernel itself.

The modules will be installed when you do a **make modules ; make modules_install** after your kernel is compiled. They will reside in a directory called /lib/modules/xx.yy.zz where xx.yy.zz is your kernel version.

3.2. A script to turn off the ’beep’

When you push the wrong key in Linux, it beeps.. it even beeps sometimes when you press the right key.. If you get irritated, just run this script with a parameter ’off’ or ’on’ (I think it’ll only work under the bash shell)

```
#!/bin/sh
if [ "$1" = "off" ]; then
echo -e "\33[11;0]"
echo "Bell turned off."
fi

if [ "$1" = "on" ]; then
echo -e "\33[10;750]\33[11;250]"
echo "Bell turned on."
fi

if [ "$1" = "" ]; then
echo "Usage: $0 <on|off>"
fi
```

(If you’re reading this from the web, it might be more convenient to download the file by shift clicking here (bell))

3.3. Using your printer in an intelligent way

One good piece of software I’ve found is apsfilter, making it possible to print almost any kind of file with a simple command. It works with a large number of printers, including all postscript printers, epson printers, Deskjet printers and all printers supported by the ghostscript software.

You must have lpd/lpr installed, and ghostscript, and a few other things..

3.3.1. How it works.

Install it, and print files with the command lpr, you can print a large number of formats, including postscript and LaTeX and compressed files and usual textfiles and... and apsfilter converts to the right format and prints it..

wonderful :-) It works with more than one printer too, I have a Citizen ProJet II (emulating a HP Deskjet), and a Fujutsi DL 1100 (emulating an Epson), and they both work very well.

3.3.2. Where do I get it ?

Check out this page (http://www.freebsd.org/~andreas/#APSFILTER).

3.4. Slow loading X or X clients (netscape or similar) ?

On systems without math coprocessor, or perhaps just a system without the absolutely newest hardware, there are sometimes problems with X clients loading slowly.

One tips that perhaps might give some better performance is to look at the Fontpath section of your XF86Config, mine looks like this:

```
FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
FontPath "/usr/X11R6/lib/X11/fonts/freefont/"
FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
```

It's all in the Files section. Well, the trick is, the "Type1" and "Speedo" Paths' should be moved to the bottom of the list (like they are in my example), or perhaps commented out. If you have them in the beginning of the list it might slow down things. Try it !

3.5. BackSpace not working in X ?

Sometimes you find that you can't use backspace in a X program. I had problems especially with TCL/TK programs, where I had to type Ctrl-H instead. A solution that works for me is adding the line

```
keycode 22 = BackSpace
```

to a Xmodmap file, and run xmodmap on that..

3.6. No colours in your file listings ?

Something I do on every new Linux installation I do, is an alias for ls. First, let's explain how to use the alias command, available at least with bash and tcsh. Alias will put a table in memory, and when you type a command that matches, it will execute what you've specified instead. There's a command for this in DOS too, but I can't remember right now. `alias ls="ls --color"` will do the right thing in bash, I think `alias ls "ls --color"` is the correct syntax for tcsh, but not sure there. You can do a lot of things with alias..

Of course, you don't want to type `alias ls="ls --color"` every time you log on, so a good idea is to put the command in the file .bash_profile in your home directory. That way it'll be done every time you log on.

3.7. How to find a program for a specific purpose ?

People knowing nothing about Linux says there are no software for the operating system. As we all know, that’s as wrong as it could be, there are tons of programs, most of them free for you to use.

3.7.1. Freshmeat.net

The most useful site for finding software for Linux is Freshmeat.net (http://freshmeat.net).

3.8. Date and time in Emacs status bar

Emacs is a very good editor (Well, almost religion for some people :-)), with so much configuration possibilities one could configure it for years, and that’s good !

I want emacs to display the current date and time in it’s status bar, place this in your $HOME/.emacs and start emacs..

```
(setq display-time-day-and-date t
   display-time-24hr-format t)
(display-time)
```

And watch the magic ! It’ll now display Date, Time, system load and mail status in the status bar. Really useful, if you ask me.

3.9. Changing your caps lock into Ctrl in X

Have you ever been irritated that there’s a big key between tab and skift that you never use? If not, sometime in the future your left little finger might start complaining about having to reach all the way down to the Ctrl key every time you’re moving around in an emacs buffer. This small tip will let you redefine your Caps Lock into a Ctrl key in X.

Of course there is a solution for this. The X window system wasn’t designed by a gang of morons so the key mapping system is quite flexibel, though a little bit hard to understand. I won’t say I understand it :)

So, let’s say you wan’t to change your Caps Lock into a Ctrl key. The first thing to do is to glance through the manual page of **xmodmap**. When you’re done with that you’ll understand a little bit more of what’s happening. The next thing would be to execute the **xev** from an xterm or similar. Try putting the cursor in the window popping up, and press some keys up and down. A lot of information will pass by in the xterm you started the **xev** from. It will most probably look something like this:

```
KeyPress event, serial 21, synthetic NO, window 0x4800001,
    root 0x36, subw 0x0, time 921569335, (87,93), root:(796,823),
    state 0x0, keycode 66 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 characters:  ""

KeyRelease event, serial 21, synthetic NO, window 0x4800001,
    root 0x36, subw 0x0, time 921569438, (87,93), root:(796,823),
    state 0x4, keycode 66 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 characters:  ""
```

The interesting parts here is ’keycode’ and perhaps what’s inside the parenthesis. Be sure to note the keycodes of the keys you’re interested in. 66 is the keycode of my Caps Lock, depending on you hardware yours might be something else (most probably it’ll be 66 on a PC).

Now lets do some real action! You might want to start with executing **xmodmap -pke** and redirect the output to some file. That way you might be able to restore your keymap without restarting your X. After that, do this:

```
xmodmap -e ’keycode 66 = Control_L’
xmodmap -e ’clear Lock’
xmodmap -e ’add Control = Control_L’
```

And test if that did the trick. If you don’t get the desired effect, try some other combinations. You may want to put your Caps_Lock somewhere else, find a decent key and do this:

```
xmodmap -e ’keycode 117 = Caps_Lock’
xmodmap -e ’add Lock = Caps_Lock’
```

117 is one of the windowskeys on my Keyboard. Find a good key with xev as described before.

I assume you don’t want to do those commands every time you restart your X. (Personally I don’t restart X more often than once a week or so). The solution to this is to put something like this:

```
keycode 66 = Control_L
clear Lock
add Control = Control_L
keycode 117 = Caps_Lock
add Lock = Caps_Lock
```

(ie, the above commands without the xmodmap before each line). in a file called .Xmodmap or .xmodmap in your home dir. Then make sure the command **xmodmap .Xmodmap** is executed somewhere in your X startup files. If you have a good distribution that’s already automatically done. Debian for example will xmodmap a file called .Xmodmap. Check out the scripts in /etc/X11/ to see if it’s done in your dist, and what the filename should be.

That’s all folks! Now get used to your new key, or use the old one. Some variation is always good to prevent repetetive stress injuries caused by keyboards.

Chapter 4. Some links to other information sources

Version: $Id: Chapter4.sgml,v 1.1.1.1 2004/09/19 12:25:14 forsberg Exp $

4.1. Where to get Linux

There are several ways to get Linux, now when you know how supreme it is =) Mainly there are two ways; From the net, if you have a fast internet connection (or a friend with fast access..). From a CD, they're really quite inexpensive.

4.1.1. CDROM Vendors

Examples of companys selling CD's with Linux on, is Walnut Creek CDROM (http://www.cdrom.com) and Infomagic (http://www.infomagic.com/).

4.1.2. FTP sites containing Linux.

There are a lot of them ! First, check out the main webpage of the distribution you want and check out their instructions for getting the dist. They often have a list of mirrors containing the same files, that way you won't load the main server too much, and you'll get a faster transfer. An important Linux ftp site is sunsite.unc.edu (ftp://sunsite.unc.edu/pub/Linux). There are a lot of Linux programs and distributions there. Other important sites is tsx-11.mit.edu (ftp://tsx-11.mit.edu) and prep.ai.mit.edu (ftp://prep.ai.mit.edu). Learn how to find programs, especially on sunsite, since there is almost always what you want, it's just a matter of finding it.

4.2. Linux/UNIX links, in no particular order.

- Linux.COM - managed by VA Research this page seems to have potential. (http://www.linux.com)
- Userfriendly.org, very funny unix/sysadmin humor. Updated every day. This is one of the places I visit every day (http://www.userfriendly.org)
- Allexperts.com - The Unix section (http://www.allexperts.com/software/unix.shtml). A website with volunteer experts answering your questions.
- SunWorldOnline, very nice online magazine with UNIX information. (http://www.sun.com/sunworldonline/sun.index.html)
- The Linux documentation project, home of the manpages, the HOWTO's and some good books. They even link this page. (http://metalab.unc.edu/LDP/)
- Linux.ORG (http://www.linux.org)
- Linux v2 Information HQ, patches to the kernel. (http://www.linuxhq.com)
- The Linux Kernel Archives (http://www.kernel.org)
- The Debian GNU/Linux Distribution headquarters (http://www.debian.org) This is the linux distribution I run.

- RedHat, another Major linux distribution, though more commercial than Debian (http://www.redhat.com)
- Linux Weekly journal, really good site with the Linux news of the week (http://lwn.net)
- Slashdot (http://slashdot.org)I visit this page daily. "News for nerds, stuff that matters"
- My main page, with a FAQ for the swedish fido meeting R20_LINUX.> (http://www.lysator.liu.se/~forsberg)
- Linux Web Watcher. Keeps track of the change date of a lot of Linux related pages. (http://webwatcher.org)
- The Linux MallOnline shop with those really useful things you need as a true Linux user =) (http://www.linuxmall.com)
- Mango Girl’s Links for Linux newbies. Ackording to the author of this page, it sucks, but I think it’s quite a nice compilation of links :) (http://www.geocities.com/SiliconValley/Circuit/7737/) Her page says I’m supernice, apparently she doesn’t know what she’s talking about.. *evil laugh* ;)

4.3. Programming links (almost only about ’C’)

- libsmb (http://www-eleves.iie.cnam.fr/brodu/smblib/index.html)A C++ library for accessing windows computers on a network from Linux (and other unices). I’m doing some work on this sometimes
- Home of python, another script language (http://www.python.org)
- Unix Socket Programming. With the Socket FAQ, and a few good links. (http://kipper.york.ac.uk/~vic/sock-faq/)
- C/C++ Users Group (http://www.hal9k.com/cug)
- Programmers oasis (http://www.utu.fi:80/~sisasa/oasis/main.html)
- The programmers booklist, by Sunir Shah (http://intranet.ca/~sshah/booklist.html)
- The programmers virtual library (http://www.strangecreations.com/)
- The Virtual C++ Library (http://www.desy.de/user/projects/C++.html)
- The snippets collection, a lot of examples, FAQ’s and such things in a package (http://www.brokersys.com/snippets)
- Scott’s tutorial hotlist (http://cires.colorado.edu/people/peckham.scott/tutors.html)
- Programming in C, at Lysator, Linkoping university (http://www.lysator.liu.se/c)

4.4. Other links, useful places, useful programs etc..

- Home of LYNX, THE Webbrowser (http://lynx.browser.org)

Chapter 5. Compiling your kernel

Version: $Id: Chapter5.sgml,v 1.1.1.1 2004/09/19 12:25:08 forsberg Exp $

I will try to describe to the new user how to compile your own kernel. I won't describe any distribution specific tools, though there are som helpful commands in both RedHat and Debian AFAIK.

5.1. Why should I compile my own kernel? What do I need?

5.1.1. Why should I compile my own kernel?

If you've just installed Linux, there was a kernel with your distribution. Why should you recompile it?

Most often there is a reason for you to compile your own kernel. First, you might have some hardware not supported in your current kernel, forcing you to compile your own.

Second, you might want to compile your own kernel because the one you have has too much hardware support compiled in. As an example, if you have a system with only (E)IDE harddisks, you don't your kernel to support several SCSI controllers. The same goes for odd hardware like CDROM:s connected to soundblaster sound cards and other similar things.

Recompiling your kernel will make it smaller, taking upp less memory. On todays computers that's often not a problem, since they have enough memory anyway.

Security is another aspect. Sometimes some security related bug is found in the kernel code, and a fix is released as a patch or in the next kernel release. When you recompile your kernel your computer won't be vulnerable to that security problem any more.

5.1.2. OK, so now I've decided to compile my own kernel, what do I need?

5.1.2.1. A C compiler

You need a c compiler with friends. To be more specific, you need the GNU C compiler, gcc. Try **gcc** at your command line. If you get the message **gcc: No input files** you have gcc. Make sure it's not historical. If you don't have any gcc, find out how to install it.

5.1.2.2. The Linux kernel source

You need (of course) the Linux kernel source, availiable at a very large number of ftp sites around the world. Try pointing your web browser at http://www.kernel.org. Try finding a kernel archive in your country by appending your country code (Ie 'se' for Sweden, 'us' for the USA, 'fi' for Finland..) after the www, making the URL something like www.us.kernel.org . That will take less international bandwidth, and that's always a good thing.

Once you've got the source (that will take some time for a modem user, since it's quite big. At the moment I'm writing this, the 2.2.10 kernel is 13 Megabytes) you should unpack it. Preferably in /usr/src since that's the place it should be. So.. get the rights you need and go to /usr/src and execute **tar -zxvf /foo/bar/linux-2.2.10.tar.gz**. That will take some time.

In my humble opinion, when I say "get the rights you need" above, I don’t mean you should get **root** since that’s a user you shouldn’t use when it’s not absolutely neccessary. Fewer things broke that way.. So, it’s better if you check the rights needed and make yourself a member of the correct group.

5.1.2.3. Time.

Compiling the kernel will take some time, it’s quite a lot of source code. Depending of your computer, and how much you include in your kernel configuration it may take anything from less than 5 minutes to a day or two. (The first example is my current Intel PII 350, the second if you’re compiling on an i386. Don’t do that :-))

So, when you’re done with your kernel configuration; set the whole thing off and go get yourself a nice cup of tea, or something

5.2. So, now I have the source, what do I do?

We are now ready to do the real work. Configuring, complining and installing the kernel. I’m not going to describe every single configuration entry availiable, since that would be a novel by itself, and there’s already very good documentation inside the tools you use to configure the kernel.

5.2.1. Configuring the kernel

5.2.1.1. What tools are there to configure the kernel?

Configuration of the kernel can be done in four different ways.

First, there is the *very manual way*. This is when you edit the file /usr/src/linux/.config by hand. A very primitive way of doing things, don’t do this since there is better ways.

Second, there is **make config**. Execute **make config** in the /usr/src/linux directory, and the kernel configurator will ask you a lot of questions. You have the ability to read some help text about each question by pressing ’?’. This is a better way than the first one, but still quite boring.

Third, there is **make menuconfig**. This will present you with a text-based program where you make your choices by moving around in some menus. This is a good way of configuring the kernel if you computer doesn’t have the X window system, or if it’s to slow to run X.

Fourth, and last, is **make xconfig**. This will give you an X window system based configuration program that is really nifty. Very easy to use!

5.2.1.2. OK, I have some configuration tool up and running, what do I do next? There’s a lot of confusing things here!

You’re right. Kernel configuration can be very confusing. At present time (Linux 2.2.10) the kernel configuration has the following subcategories, each with a lot of choices.

Table 5-1. Kernel configuration subcategories with explanation

Category	*Explanation*

Code maturity level options	This lets you select if you want to be prompted for experimental drivers in the kernel. Most often you want that.
Processor type and features	Select your processor type, if you have an mathematic processor and some other important stuff. Be warned, if you don’t have a math coprocessor and don’t choose math emulation here, your kernel will not boot. The same warning goes for choosing the wrong processor type.
Loadable Module support	Lets you enable the support for modules in the kernel. You want this. Read Section 3.1
General Setup	A lot of general stuff, like if you want network support, if you have a PCI system and if you want Advanced Power Management.
Plug and Play support	Choose if you want the kernel to be able to auto-configure some devices.
Block devices	This is where you tell the kernel what kind of harddisk and other storage devices you have
Networking options	There’s a lot of more or less advanced networking options in the kernel. If you don’t know what to do, read the help or let it be.
QoS and/or fair queueing	This is another advanced network option. Let it be if you don’t know what it is.
SCSI support	If you have some type of SCSI devices in your system, you want to check out this entry.
SCSI low-level drivers	Here you add the adapter you have
Network device support	If you have a network card, you can choose it here
Amateur Radio support	This is for people with a radio connected to their computer.
IrDA subsystem support	If you want to use the IrDA port you might have on your computer, enable this.
Infrared-port device drivers	This also has to do with infrared devices
ISDN subsystem	If you access the internet (or something else) via ISDN, this is where to add support.
Old CD-ROM drivers (not SCSI, not IDE)	If you have some weird CDROM, for example one connected to a soundblaster sound card or something similar, there might be support for it here.
Character devices	Virtual terminals, serial ports and other similar things
Mice	Support for your mouse.
Watchdog Cards	A watchdog card is a piece of hardware checking your computer all the time. It communicates with the kernel, and if it doesn’t get any response it will reboot the computer. This is conventient if your computer is located where you can’t reach it.
Video for Linux	Support for videocapture and radios.
Joystick support	Guess! :)

Ftape, the floppy tape device driver	This is a driver for cheap tape drives connected to the floppy port in your computer.
Filesystems	Linux can read and write quite a few different filesystems..
Network File Systems	..There’s also support for a few network based file systems such as NFS, SMB and NCP.
Partition Types	Here you can add the ability to read several different partition types, such as the one Free/Open/NetBSD uses. That will let you mount disks from those operating systems.
Native Language Support	Add support for your language, and you will be able to see the filenames of foreign partitions correctly.
Console drivers	Console drivers gives you high resolution text consoles.
Sound	Add sound support
Additional low level sound drivers	Drivers for some special sound cards
Kernel hacking	Lets you enable the magic SysRq key. Read the manual.

Phew, that took it’s time to type in! :-)

I assume you are now even more confused. Anyway, the trick is to take your time and go through all the options availiable and try to decide what the answer should be. There is help, and if you are uncertaion, just let it be.

If your kernel doesn’t work, it’s easy to revert to the old one and reconfigure the kernel. Of course it takes time, but learning-by-doing is the only way.

When you’re done, save and exit and proceed to the next step..

5.2.2. Compiling the kernel

So, now our kernel is well configured and we are ready to compile it. This is done with the command **make bzImage** followed by **make modules** and **make modules_install**. If you haven’t enabled modules, you don’t need to do the last two steps. The whole process will, as I’ve mentioned earlier, take some time. If you didn’t read Section 3.1 earlier, do it now.

The kernel is now ready for installation.

5.2.3. Installing and booting the new kernel

To run the new kernel you have to reboot your machine. This is one of the few things software related you have to reboot to fix.

5.2.3.1. Installing the new kernel in LILO

Your newly compiled kernel is at /usr/src/linux/arch/i386/boot/bzImage . That’s not a good place for a kernel to be, so let’s move it. If you machine doesn’t have a Intel family processor, the i386 should be replaced by something appropriate, such as ’alpha’ or ’mips’.

Personally, I keep my kernels in /boot, but you are free to choose a place to copy the file above to. Keep in mind though that for most PC bioses to boot the kernel it has bo be located in the first 1024 cylinders of the harddisk.

Now install the kernel in the /etc/lilo.conf file, read Section 2.7 if you don't know how to do. Basically you just put the name of the image file in the top of the file, and a symbolic name too. LILO will boot the first image in the configuration file. Don't forget to run **/sbin/lilo** - otherwise the change won't take effect. I've done that mistake, several times :-) Always keep at least one kernel you know work in the lilo.conf, so you may rescueboot if your newly compiled kernel doesn't work.

Reboot, and see what happends. If you're unlucky, it won't boot at all, then restart and choose another kernel at the LILO command line. Watch the boot messages closely, too see that all your hardware is found. With the **dmesg** you can see what the kernel said, after it's booted.

Chapter 6. Thanks to..

Version: $Id: Chapter6.sgml,v 1.1.1.1 2004/09/19 12:25:08 forsberg Exp $

.. all the people in the Swedish R20_LINUX fido conference for their knowledge about the best operating system in the world. Also thanks to the people in the international fido meeting LINUX, also recommended for it's nice people and good answers. For information about the nice computer conference network fidonet, check out www.fidonet.org (http://www.fidonet.org).

Nowadays I spend far more time in the Electronic Conference system LYSKom (http://www.lysator.liu.se/lyskom) than in Fidonet. That's a very good source for all kinds of information, though in swedish..

Thanks to Ed Suda,Peter Åstrand, Jon Berg, John Paulsen, Wei Chen, Ron Price, Keith Bailey, Arnold L. Johnson, Damien Carbery, Eric C. and Noel Cruz for feedback and corrections.