Introduction to Linux

A Hands on Guide

Machtelt Garrels

Garrels.be

<tille wants no spam at garrels dot be>

1.27 Edition

20080606

Table of Contents

Table of Contents

Table of Contents

Introduction

1. Why this guide?

Many people still believe that learning Linux is difficult, or that only experts can understand how a Linux system works. Though there is a lot of free documentation available, the documentation is widely scattered on the Web, and often confusing, since it is usually oriented toward experienced UNIX or Linux users. Today, thanks to the advancements in development, Linux has grown in popularity both at home and at work. The goal of this guide is to show people of all ages that Linux can be simple and fun, and used for all kinds of purposes.

2. Who should read this book?

This guide was created as an overview of the Linux Operating System, geared toward new users as an exploration tour and getting started guide, with exercises at the end of each chapter. For more advanced trainees it can be a desktop reference, and a collection of the base knowledge needed to proceed with system and network administration. This book contains many real life examples derived from the author's experience as a Linux system and network administrator, trainer and consultant. We hope these examples will help you to get a better understanding of the Linux system and that you feel encouraged to try out things on your own.

Everybody who wants to get a "CLUE", a Command Line User Experience, with Linux (and UNIX in general) will find this book useful.

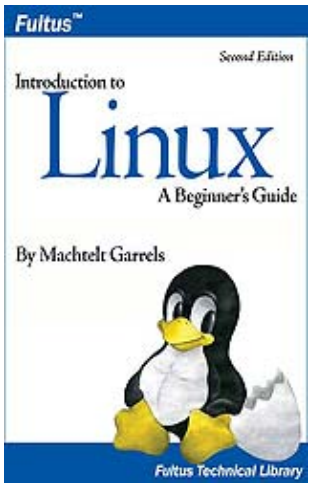
3. New versions and availability

This document is published in the Guides section of the Linux Documentation Project collection at http://www.tldp.org/guides.html; you can also download PDF and PostScript formatted versions here.

The most recent edition is available at http://tille.garrels.be/training/tldp/.

The second edition of this guide is available in print from Fultus.com Books as paperback Print On Demand (POD) book. Fultus distributes this document through Ingram and Baker & Taylor to many bookstores, including Amazon.com, Amazon.co.uk, BarnesAndNoble.com and Google's Froogle global shopping portal and Google Book Search.

Figure 1. Introduction to Linux front cover



The guide has been translated into Hindi by:

- Alok Kumar
- Dhananjay Sharma
- Kapil
- Puneet Goel
- Ravikant Yuyutsu

Andrea Montagner translated the guide into Italian.

4. Revision History

Revision History

Revision 1.27 2008-06-06 Revised by: MG
updates.

Revision 1.26 20070919 Revised by: MG
Comments from readers, license.

Revision 1.25 20070511 Revised by: MG
Comments from readers, minor updates, E-mail etiquette, updated info about availability (thanks Oleg).

Revision 1.24 2006-11-01 Revised by: MG
added index terms, prepared for second printed edition, added gpg and proxy info.

Revision 1.23 2006-07-25 Revised by: MG and FK
Updates and corrections, removed app5 again, adapted license to enable inclusion in Debian docs.

Revision 1.22 2006-04-06 Revised by: MG
chap8 revised completely, chap10: clarified examples, added ifconfig and cygwin info, revised network apps.

Revision 1.21 2006-03-14 Revised by: MG
Added exercises in chap11, corrected newline errors, command overview completed for chapter 9, minor corrections in chap10.

Revision 1.20 2006-01-06 Revised by: MG
Split chap7: audio stuff is now in separate chapter, chap11.xml. Small revisions, updates for commands like aptitude, more on USB storage, Internet telephony, corrections from readers.

Revision 1.13 2004-04-27 Revised by: MG

- Chapter 7: Configuring your graphical, text and audio environment, settings for the non-native English speaking Linux user, tips for adding extra software.
- Chapter 8: Converting files to a printable format, getting them out of the printer, hints for solving print problems.
- Chapter 9: Preparing data to be backed up, discussion of various tools, remote backup.
- Chapter 10: Overview of Linux networking tools and user applications, with a short discussion of the underlying service daemon programs and secure networking.
- Chapter 11: Sound and video, including Voice over IP and sound recording is discussed in this chapter.
- Appendix A: Which books to read and sites to visit when you have finished reading this one.
- Appendix B: A comparison.
- Appendix C: If you ever get stuck, these tables might be an outcome. Also a good argument when your boss insists that YOU should use HIS favorite shell.

The standard installation process allows users to choose between different basic setups, such as a workstation, where all packages needed for everyday use and development are installed, or a server installation, where different network services can be selected. Expert users can install every combination of packages they want during the initial installation process.

The goal of this guide is to apply to all Linux distributions. For your own convenience, however, it is strongly advised that beginners stick to a mainstream distribution, supporting all common hardware and applications by default. The following are very good choices for novices:

- Fedora Core
- Debian
- SuSE Linux
- Mandriva (former MandrakeSoft)
- Knoppix: an operating system that runs from your CD-ROM, you don't need to install anything.

Downloadable ISO-images can be obtained from LinuxISO.org. The main distributions can be purchased in any decent computer shop.

1.6. Summary

In this chapter, we learned that:

- Linux is an implementation of UNIX.
- The Linux operating system is written in the C programming language.
- "De gustibus et coloribus non disputandum est": there's a Linux for everyone.
- Linux uses GNU tools, a set of freely available standard tools for handling the operating system.

1.7. Exercises

A practical exercise for starters: install Linux on your PC. Read the installation manual for your distribution and/or the Installation HOWTO and do it.

Read the docs!

Most errors stem from not reading the information provided during the install. Reading the installation messages carefully is the first step on the road to success.

Things you must know BEFORE starting a Linux installation:

- Will this distribution run on my hardware?

  Check with http://www.tldp.org/HOWTO/Hardware-HOWTO/index.html when in doubt about compatibility of your hardware.
- What kind of keyboard do I have (number of keys, layout)? What kind of mouse (serial/parallel, number of buttons)? How many MB of RAM?
- Will I install a basic workstation or a server, or will I need to select specific packages myself?
- Will I install from my hard disk, from a CD-ROM, or using the network? Should I adapt the BIOS for any of this? Does the installation method require a boot disk?
- Will Linux be the only system on this computer, or will it be a dual boot installation? Should I make a large partition in order to install virtual systems later on, or is this a virtual installation itself?

Chapter 2. Quickstart

In order to get the most out of this guide, we will immediately start with a practical chapter on connecting to the Linux system and doing some basic things.

We will discuss:

- Connecting to the system
- Disconnecting from the system
- Text and graphic mode
- Changing your password
- Navigating through the file system
- Determining file type
- Looking at text files
- Finding help

2.1. Logging in, activating the user interface and logging out

2.1.1. Introduction

In order to work on a Linux system directly, you will need to provide a user name and password. You always need to authenticate to the system. As we already mentioned in the exercise from Chapter 1, most PC-based Linux systems have two basic modes for a system to run in: either quick and sober in text console mode, which looks like DOS with mouse, multitasking and multi-user features, or in graphical mode, which looks better but eats more system resources.

2.1.2. Graphical mode

This is the default nowadays on most desktop computers. You know you will connect to the system using graphical mode when you are first asked for your user name, and then, in a new window, to type your password.

To log in, make sure the mouse pointer is in the login window, provide your user name and password to the system and click OK or press **Enter**.

Careful with that root account!

It is generally considered a bad idea to connect (graphically) using the *root* user name, the system adminstrator's account, since the use of graphics includes running a lot of extra programs, in root's case with a lot of extra permissions. To keep all risks as low as possible, use a normal user account to connect graphically. But there are enough risks to keep this in mind as a general advice, for all use of the root account: only log in as root when extra privileges are required.

After entering your user name/password combination, it can take a little while before the graphical environment is started, depending on the CPU speed of your computer, on the software you use and on your personal settings.

working, whatever happens. On Linux, you will virtually never see irritating messages like *Out of memory, please close some applications first and try again*, because of this extra memory. The swap or virtual memory procedure has long been adopted by operating systems outside the UNIX world by now.

Using memory on a hard disk is naturally slower than using the real memory chips of a computer, but having this little extra is a great comfort. We will learn more about swap when we discuss processes in Chapter 4.

Linux generally counts on having twice the amount of physical memory in the form of swap space on the hard disk. When installing a system, you have to know how you are going to do this. An example on a system with 512 MB of RAM:

- 1st possibility: one swap partition of 1 GB
- 2nd possibility: two swap partitions of 512 MB
- 3rd possibility: with two hard disks: 1 partition of 512 MB on each disk.

The last option will give the best results when a lot of I/O is to be expected.

Read the software documentation for specific guidelines. Some applications, such as databases, might require more swap space. Others, such as some handheld systems, might not have any swap at all by lack of a hard disk. Swap space may also depend on your kernel version.

The kernel is on a separate partition as well in many distributions, because it is the most important file of your system. If this is the case, you will find that you also have a */boot* partition holding your kernel(s) and accompanying data files.

The rest of the hard disk(s) is generally divided in data partitions, although it may be that all of the non-system critical data resides on one partition, for example when you perform a standard workstation installation. When non-critical data is separated on different partitions, it usually happens following a set pattern:

- a partition for user programs (*/usr*)
- a partition containing the users' personal data (*/home*)
- a partition to store temporary data like print- and mail-queues (*/var*)
- a partition for third party and extra software (*/opt*)

Once the partitions are made, you can only add more. Changing sizes or properties of existing partitions is possible but not advisable.

The division of hard disks into partitions is determined by the system administrator. On larger systems, he or she may even spread one partition over several hard disks, using the appropriate software. Most distributions allow for standard setups optimized for workstations (average users) and for general server purposes, but also accept customized partitions. During the installation process you can define your own partition layout using either your distribution specific tool, which is usually a straight forward graphical interface, or **fdisk**, a text-based tool for creating partitions and setting their properties.

A workstation or client installation is for use by mainly one and the same person. The selected software for installation reflects this and the stress is on common user packages, such as nice desktop themes, development tools, client programs for E-mail, multimedia software, web and other services. Everything is put together on one large partition, swap space twice the amount of RAM is added and your generic workstation is complete, providing the largest amount of disk space possible for personal use, but with the disadvantage of possible data integrity loss during problem situations.

3.2. Orientation in the file system

3.2.1. The path

When you want the system to execute a command, you almost never have to give the full path to that command. For example, we know that the **ls** command is in the `/bin` directory (check with **which -a *ls***), yet we don't have to enter the command **/bin/ls** for the computer to list the content of the current directory.

The `PATH` environment variable takes care of this. This variable lists those directories in the system where executable files can be found, and thus saves the user a lot of typing and memorizing locations of commands. So the path naturally contains a lot of directories containing `bin` somewhere in their names, as the user below demonstrates. The **echo** command is used to display the content ("$") of the variable `PATH`:

```
rogier:> echo $PATH
/opt/local/bin:/usr/X11R6/bin:/usr/bin:/usr/sbin/:/bin
```

In this example, the directories `/opt/local/bin`, `/usr/X11R6/bin`, `/usr/bin`, `/usr/sbin` and `/bin` are subsequently searched for the required program. As soon as a match is found, the search is stopped, even if not every directory in the path has been searched. This can lead to strange situations. In the first example below, the user knows there is a program called **sendsms** to send an SMS message, and another user on the same system can use it, but she can't. The difference is in the configuration of the `PATH` variable:

```
[jenny@blob jenny]$ sendsms
bash: sendsms: command not found
[jenny@blob jenny]$ echo $PATH
/bin:/usr/bin:/usr/bin/X11:/usr/X11R6/bin:/home/jenny/bin
[jenny@blob jenny]$ su - tony
Password:
tony:~>which sendsms
sendsms is /usr/local/bin/sendsms

tony:~>echo $PATH
/home/tony/bin.Linux:/home/tony/bin:/usr/local/bin:/usr/local/sbin:\
/usr/X11R6/bin:/usr/bin:/usr/sbin:/bin:/sbin
```


Note the use of the **su** (switch user) facility, which allows you to run a shell in the environment of another user, on the condition that you know the user's password.

A backslash indicates the continuation of a line on the next, without an **Enter** separating one line from the other.

In the next example, a user wants to call on the **wc** (word count) command to check the number of lines in a file, but nothing happens and he has to break off his action using the **Ctrl+C** combination:

```
jumper:~> wc -l test

(Ctrl-C)
jumper:~> which wc
wc is hashed (/home/jumper/bin/wc)

jumper:~> echo $PATH
/home/jumper/bin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin:\
/usr/bin:/usr/sbin:/bin:/sbin
```

The use of the **which** command shows us that this user has a `bin`-directory in his home directory, containing a program that is also called **wc**. Since the program in his home directory is found first when searching the

3.4.2. The tools

3.4.2.1. The chmod command

A normal consequence of applying strict file permissions, and sometimes a nuisance, is that access rights will need to be changed for all kinds of reasons. We use the **chmod** command to do this, and eventually *to chmod* has become an almost acceptable English verb, meaning the changing of the access mode of a file. The **chmod** command can be used with alphanumeric or numeric options, whatever you like best.

The example below uses alphanumeric options in order to solve a problem that commonly occurs with new users:

```
asim:~> ./hello
bash: ./hello: bad interpreter: Permission denied

asim:~> cat hello
#!/bin/bash
echo "Hello, World"

asim:~> ls -l hello
-rw-rw-r--    1 asim    asim    32 Jan 15 16:29 hello

asim:~> chmod u+x hello

asim:~> ./hello
Hello, World

asim:~> ls -l hello
-rwxrw-r--    1 asim    asim    32 Jan 15 16:29 hello*
```


The **+** and **-** operators are used to grant or deny a given right to a given group. Combinations separated by commas are allowed. The Info and man pages contain useful examples. Here's another one, which makes the file from the previous example a private file to user *asim*:

```
asim:~> chmod u+rwx,go-rwx hello

asim:~> ls -l hello
-rwx------    1 asim    asim    32 Jan 15 16:29 hello*
```

The kind of problem resulting in an error message saying that permission is denied somewhere is usually a problem with access rights in most cases. Also, comments like, "It worked yesterday," and "When I run this as root it works," are most likely caused by the wrong file permissions.

When using **chmod** with numeric arguments, the values for each granted access right have to be counted together per group. Thus we get a 3-digit number, which is the symbolic value for the settings **chmod** has to make. The following table lists the most common combinations:

Table 3-9. File protection with chmod

Command	Meaning
chmod *400* file	To protect a file against accidental overwriting.
chmod *500* directory	To protect yourself from accidentally removing, renaming or moving files from this directory.

3.5. Summary

On UNIX, as on Linux, all entities are in some way or another presented to the system as files with the appropriate file properties. Use of (predefined) paths allows the users and the system admin to find, read and manipulate files.

We've made our first steps toward becoming an expert: we discussed the real and the fake structure of the file system, and we know about the Linux file security model, as well as several other security precautions that are taken on every system by default.

The shell is the most important tool for interaction with the system. We learned several shell commands in this chapter, which are listed in the table below.

Table 3-10. New commands in chapter 3: Files and the file system

Command	Meaning
bash	GNU shell program.
cat `file(s)`	Send content of file(s) to standard output.
cd `directory`	Enter `directory`. **cd** is a **bash** built-in command.
chgrp *`newgroup`* `file(s)`	Change the group ownership of `file(s)` to *newgroup*
chmod *`mode`* `file(s)`	Change access permissions on `file(s)`
chown *`newowner`*[:[newgroup]] `file(s)`	Change file owner and group ownership.
cp `sourcefile targetfile`	Copy `sourcefile` to `targetfile`.
df `file`	Reports on used disk space on the partition containing `file`.
echo *`string`*	Display a line of text
export	Part of **bash** that announces variables and their values to the system.
file `filename`	Determine file type of `filename`.
find *`path expression`*	Find files in the file system hierarchy
grep *`PATTERN`* `file`	Print lines in `file` containing the search pattern.
head `file`	Send the first part of `file` to standard output
id	Prints real and effective user name and groups.
info *`command`*	Read documentation about **command**.
less `file`	View `file` with a powerful viewer.
ln `targetfile linkname`	Make a link with name `linkname` to `targetfile`.
locate *`searchstring`*	Print all accessible files matching the search pattern.
ls `file(s)`	Prints directory content.
man *`command`*	Format and display online (system) manual pages for **command**.
mkdir `newdir`	Make a new empty directory.
mv `oldfile newfile`	Rename or move `oldfile`.
newgrp *`groupname`*	Log in to a new group.
pwd	Print the present or current working directory.

3.6.3. Tour of the system

- Change to the `/proc` directory.
- What CPU(s) is the system running on?
- How much RAM does it currently use?
- How much swap space do you have?
- What drivers are loaded?
- How many hours has the system been running?
- Which filesystems are known by your system?
- Change to `/etc/rc.d | /etc/init.d | /etc/runlevels` and choose the directory appropriate for your run level.
- What services should be running in this level?
- Which services run in graphical mode that don't run in text mode?
- Change to `/etc`
- How long does the system keep the log file in which user logins are monitored?
- Which release are you running?
- Are there any issues or messages of the day?
- How many users are defined on your system? Don't count them, let the computer do it for you!
- How many groups?
- Where is the time zone information kept?
- Are the HOWTOs installed on your system?
- Change to `/usr/share/doc`.
- Name three programs that come with the GNU *coreutils* package.
- Which version of **bash** is installed on this system?

3.6.4. Manipulating files

- Create a new directory in your home directory.
- Can you move this directory to the same level as your home directory?
- Copy all XPM files from `/usr/share/pixmaps` to the new directory. What does XPM mean?
- List the files in reverse alphabetical order.
- Change to your home directory. Create a new directory and copy all the files of the `/etc` directory into it. Make sure that you also copy the files and directories which are in the subdirectories of `/etc`! (recursive copy)
- Change into the new directory and make a directory for files starting with an upper case character and one for files starting with a lower case character. Move all the files to the appropriate directories. Use as few commands as possible.
- Remove the remaining files.
- Delete the directory and its entire content using a single command.
- Use **grep** to find out which script starts the Font Server in the graphical run level.
- Where is the *sendmail* server program?
- Make a symbolic link in your home directory to `/var/tmp`. Check that it really works.
- Make another symbolic link in your home directory to this link. Check that it works. Remove the first link and list directory content. What happened to the second link?

3.6.5. File permissions

- Can you change file permissions on `/home`?
- What is your standard file creation mode?
- Change ownership of `/etc` to your own user and group.

4.1.5.3. Signals

Processes end because they receive a signal. There are multiple signals that you can send to a process. Use the **kill** command to send a signal to a process. The command **kill -l** shows a list of signals. Most signals are for internal use by the system, or for programmers when they write code. As a user, you will need the following signals:

Table 4-2. Common signals

Signal name	Signal number	Meaning
SIGTERM	15	Terminate the process in an orderly way.
SIGINT	2	Interrupt the process. A process can ignore this signal.
SIGKILL	9	Interrupt the process. A process can not ignore this signal.
SIGHUP	1	For daemons: reread the configuration file.

You can read more about default actions that are taken when sending a signal to a process in **man 7 *signal***.

4.1.6. SUID and SGID

As promised in the previous chapter, we will now discuss the special modes SUID and SGID in more detail. These modes exist to provide normal users the ability to execute tasks they would normally not be able to do because of the tight file permission schemes used on UNIX based systems. In the ideal situation special modes are used as sparsely as possible, since they include security risks. Linux developers have generally tried to avoid them as much as possible. The Linux **ps** variant, for example, uses the information stored in the `/proc` file system, which is accessible to everyone, thus avoiding exposition of sensitive system data and resources to the general public. Before this, and still on older UNIX systems, the **ps** program needed access to files such as `/dev/mem` and `/dev/kmem`, which had disadvantages because of the permissions and ownerships on these files:

```
rita:~> ls -l /dev/*mem
crw-r-----    1 root     kmem       1,   2 Aug 30 22:30 /dev/kmem
crw-r-----    1 root     kmem       1,   1 Aug 30 22:30 /dev/mem
```

With older versions of **ps**, it was not possible to start the program as a common user, unless special modes were applied to it.

While we generally try to avoid applying any special modes, it is sometimes necessary to use an SUID. An example is the mechanism for changing passwords. Of course users will want to do this themselves instead of having their password set by the system administrator. As we know, user names and passwords are listed in the `/etc/passwd` file, which has these access permissions and owners:

```
bea:~> ls -l /etc/passwd
-rw-r--r--    1 root     root     1267 Jan 16 14:43 /etc/passwd
```

Still, users need to be able to change their own information in this file. This is achieved by giving the **passwd** program special permissions:

```
mia:~> which passwd
passwd is /usr/bin/passwd
```

The GNU **time** command in `/usr/bin` (as opposed to the shell built-in version) displays more information that can be formatted in different ways. It also shows the exit status of the command, and the total elapsed time. The same command as the above using the independent **time** gives this output:

```
tilly:~/xml/src> /usr/bin/time make
Output written on abook.pdf (222 pages, 1595027 bytes).
Transcript written on abook.log.

Command exited with non-zero status 2
88.87user 1.74system 1:36.21elapsed 94%CPU
                                (0avgtext+0avgdata 0maxresident)k
0inputs+0outputs (2192major+30002minor)pagefaults 0swaps
```

Refer again to the Info pages for all the information.

4.3.3. Performance

To a user, performance means quick execution of commands. To a system manager, on the other hand, it means much more: the system admin has to optimize system performance for the whole system, including users, all programs and daemons. System performance can depend on a thousand tiny things which are not accounted for with the **time** command:

- the program executing is badly written or doesn't use the computer appropriately
- access to disks, controllers, display, all kinds of interfaces, etc.
- reachability of remote systems (network performance)
- amount of users on the system, amount of users actually working simultaneously
- time of day
- ...

4.3.4. Load

In short: the load depends on what is normal for your system. My old P133 running a firewall, SSH server, file server, a route daemon, a sendmail server, a proxy server and some other services doesn't complain with 7 users connected; the load is still 0 on average. Some (multi-CPU) systems I've seen were quite happy with a load of 67. There is only one way to find out - check the load regularly if you want to know what's normal. If you don't, you will only be able to measure system load from the response time of the command line, which is a very rough measurement since this speed is influenced by a hundred other factors.

Keep in mind that different systems will behave different with the same load average. For example, a system with a graphics card supporting hardware acceleration will have no problem rendering 3D images, while the same system with a cheap VGA card will slow down tremendously while rendering. My old P133 will become quite uncomfortable when I start the X server, but on a modern system you hardly notice the difference in the system load.

4.3.5. Can I do anything as a user?

A big environment can slow you down. If you have lots of environment variables set (instead of shell variables), long search paths that are not optimized (errors in setting the path environment variable) and more of those settings that are usually made "on the fly", the system will need more time to search and read data.

In X, window managers and desktop environments can be real CPU-eaters. A really fancy desktop comes with a price, even when you can download it for free, since most desktops provide add-ons ad infinitum. Modesty

```
mike:~> cat wishlist
more money
less work
Thu Feb 28 20:23:07 CET 2002
```

The **date** command would normally put the last line on the screen; now it is appended to the file `wishlist`.

5.2. Advanced redirection features

5.2.1. Use of file descriptors

There are three types of I/O, which each have their own identifier, called a file descriptor:

- standard input: 0
- standard output: 1
- standard error: 2

In the following descriptions, if the file descriptor number is omitted, and the first character of the redirection operator is <, the redirection refers to the standard input (file descriptor 0). If the first character of the redirection operator is >, the redirection refers to the standard output (file descriptor 1).

Some practical examples will make this more clear:

ls > dirlist 2>&1

will direct both standard output and standard error to the file `dirlist`, while the command

ls 2>&1 > dirlist

will only direct standard output to `dirlist`. This can be a useful option for programmers.

Things are getting quite complicated here, don't confuse the use of the ampersand here with the use of it in Section 4.1.2.1, where the ampersand is used to run a process in the background. Here, it merely serves as an indication that the number that follows is not a file name, but rather a location that the data stream is pointed to. Also note that the bigger-than sign should not be separated by spaces from the number of the file descriptor. If it would be separated, we would be pointing the output to a file again. The example below demonstrates this:

```
[nancy@asus /var/tmp]$ ls 2> tmp

[nancy@asus /var/tmp]$ ls -l tmp
-rw-rw-r--  1 nancy nancy 0 Sept  7 12:58 tmp

[nancy@asus /var/tmp]$ ls 2 > tmp
ls: 2: No such file or directory
```

The first command that *nancy* executes is correct (eventhough no errors are generated and thus the file to which standard error is redirected is empty). The second command expects that `2` is a file name, which does not exist in this case, so an error is displayed.

All these features are explained in detail in the Bash Info pages.

5.2.2. Examples

5.2.2.1. Analyzing errors

If your process generates a lot of errors, this is a way to thoroughly examine them:

command 2>&1 | less

This is often used when creating new software using the **make** command, such as in:

```
andy:~/newsoft> make all 2>&1 | less
--output ommitted--
```

5.2.2.2. Separating standard output from standard error

Constructs like these are often used by programmers, so that output is displayed in one terminal window, and errors in another. Find out which pseudo terminal you are using issuing the **tty** command first:

```
andy:~/newsoft> make all 2> /dev/pts/7
```

5.2.2.3. Writing to output and files simultaneously

You can use the **tee** command to copy input to standard output and one or more output files in one move. Using the `-a` option to **tee** results in appending input to the file(s). This command is useful if you want to both see and save output. The **>** and **>>** operators do not allow to perform both actions simultaneously.

This tool is usually called through a pipe (|), as demonstrated in the example below:


```
mireille ~/test> date | tee file1 file2
Thu Jun 10 11:10:34 CEST 2004

mireille ~/test> cat file1
Thu Jun 10 11:10:34 CEST 2004

mireille ~/test> cat file2
Thu Jun 10 11:10:34 CEST 2004

mireille ~/test> uptime | tee -a file2
 11:10:51 up 21 days, 21:21, 57 users,  load average: 0.04, 0.16, 0.26

mireille ~/test> cat file2
Thu Jun 10 11:10:34 CEST 2004
 11:10:51 up 21 days, 21:21, 57 users,  load average: 0.04, 0.16, 0.26
```

5.3. Filters

When a program performs operations on input and writes the result to the standard output, it is called a filter. One of the most common uses of filters is to restructure output. We'll discuss a couple of the most important filters below.

- List the devices in `/dev` which are currently used by your UID. Pipe through **less** to view them properly.
- Issue the following commands as a non-privileged user. Determine standard input, output and error for each command.

    - **cat nonexistentfile**
    - **file /sbin/ifconfig**
    - **grep *root* /etc/passwd /etc/nofiles > grepresults**
    - **/etc/init.d/sshd *start* > /var/tmp/output**
    - **/etc/init.d/crond *start* > /var/tmp/output 2>&1**
    - Now check your results by issuing the commands again, now redirecting standardoutput to the file `/var/tmp/output` and standard error to the file `/var/tmp/error`.
- How many processes are you currently running?
- How many invisible files are in your home directory?
- Use **locate** to find documentation about the kernel.
- Find out which file contains the following entry:

```
root:x:0:0:root:/root:/bin/bash
```

  And this one:

```
system:          root
```

- See what happens upon issuing this command:

  > time; date >> time; cat < time
- What command would you use to check which script in `/etc/init.d` starts a given process?

6.5. Exercises

This chapter has only one exercise: start the Vim tutor by entering **vimtutor** in a terminal session, and get started.

You may alternatively start **emacs** and type **Ctrl+H** and then **T** to invoke the self-paced Emacs tutorial.

Practice is the only way!

```
                killproc ICanSeeYou
                echo
                rm -f /var/lock/subsys/ICanSeeYou
                rm -f /var/run/ICanSeeYou/ICanSeeYou.pid
                ;;
        status)
                status ICanSeeYou
                ;;
        restart)
                $0 stop
                $0 start
                ;;
        *)
                echo "Usage: $0 {start|stop|restart|status}"
                exit 1
esac

exit 0
```

First, with the **.** command (dot) a set of shell functions, used by almost all shell scripts in `/etc/rc.d/init.d`, is loaded. Then a **case** command is issued, which defines 4 different ways the script can execute. An example might be **ICanSeeYou *`start`***. The decision of which case to apply is made by reading the (first) argument to the script, with the expression *$1*.

When no compliant input is given, the default case, marked with an asterisk, is applied, upon which the script gives an error message. The **case** list is ended with the **esac** statement. In the *start* case the server program is started as a daemon, and a process ID and lock are assigned. In the *stop* case, the server process is traced down and stopped, and the lock and the PID are removed. Options, such as the `daemon` option, and functions like `killproc`, are defined in the `/etc/rc.d/init.d/functions` file. This setup is specific to the distribution used in this example. The init scripts on your system might use other functions, defined in other files, or none at all.

Upon success, the script returns an exit code of zero to its parent.

This script is a fine example of using functions, which make the script easier to read and the work done faster. Note that they use **sh** instead of **bash**, to make them useful on a wider range of systems. On a Linux system, calling **bash** as **sh** results in the shell running in POSIX-compliant mode.

The **bash** man pages contain more information about combining commands, **for**- and **while**-loops and regular expressions, as well as examples. A comprehensible Bash course for system administrators and power users, with exercises, from the same author as this Introduction to Linux guide, is at http://tille.garrels.be/training/bash/. Detailed description of Bash features and applications is in the reference guide Advanced Bash Scripting.

7.3. The graphical environment

7.3.1. Introduction

The average user may not care too much about his login settings, but Linux offers a wide variety of flashy window and desktop managers for use under X, the graphical environment. The use and configuration of window managers and desktops is straightforward and may even resemble the standard MS Windows, Apple or UNIX CDE environment, although many Linux users prefer flashier desktops and fancier window managers. We won't discuss the user specific configuration here. Just experiment and read the documentation using the built-in Help functions these managers provide and you will get along fine.

usually in `/etc/X11`.

If you see that the file `/etc/X11/XF86Config` is present on your system, a full description can be found in the Info or man pages about `XF86Config`.

Because of licensing issues with XFree86, newer systems usually come with the *X.Org* distribution of the X server and tools. The main configuration file here is `xorg.conf`, usually also in `/etc/X11`. The file consists of a number of sections that may occur in any order. The sections contain information about your monitor, your video adaptor, the screen configuration, your keyboard etcetera. As a user, you needn't worry too much about what is in this file, since everything is normally determined at the time the system is installed.

Should you need to change graphical server settings, however, you can run the configuration tools or edit the configuration files that set up the infrastructure to use the XFree86 server. See the man pages for more information; your distribution might have its own tools. Since misconfiguration may result in unreadable garbage in graphical mode, you may want to make a backup copy of the configuration file before attempting to change it, just to be on the safe side.

7.4. Region specific settings

7.4.1. Keyboard setup

Setting the keyboard layout is done using the **loadkeys** command for text consoles. Use your local X configuration tool or edit the *Keyboard* section in `XF86Config` manually to configure the layout for graphical mode. The `XkbdLayout` is the one you want to set:

```
    XkbLayout       "us"
```

This is the default. Change it to your local settings by replacing the quoted value with any of the names listed in the subdirectories of your `keymaps` directory. If you can't find the keymaps, try displaying their location on your system issuing the command

locate keymaps

It is possible to combine layout settings, like in this example:

```
Xkblayout       "us,ru"
```

Make a backup of the `/etc/X11/XF86Config` file before editing it! You will need to use the *root* account to do this.

Log out and reconnect in order to reload X settings.

The Gnome Keyboard Applet enables real-time switching between layouts; no special pemissions are needed for using this program. KDE has a similar tool for switching between keyboard layouts.

7.4.2. Fonts

Use the **setfont** tool to load fonts in text mode. Most systems come with a standard `inputrc` file which enables combining of characters, such as the French "é" (meta characters). The system admin should then add the line

Note that this is not the CUPS web interface and only works for printers supporting this feature. Check the documentation of your printer.

If your job ID is not there and not on the printer, contact your system administrator. If your job ID is listed in the output, check that the printer is currently printing. If so, just wait, your job will get done in due time.

If the printer is not printing, check that it has paper, check the physical connections to both electricity and data network. If that's okay, the printer may need restarting. Ask your system admin for advice.

In the case of a network printer, try printing from another host. If the printer is reachable from your own host (see Chapter 10 for the **ping** utility), you may try to put the formatted file on it, like `file.ps` in case of a PostScript printer, using an FTP client. If that works, your print system is misconfigured. If it doesn't work, maybe the printer doesn't understand the format you are feeding it.

The GNU/Linux Printing site contains more tips and tricks.

8.4. Summary

The Linux print service comes with a set of printing tools based on the standard UNIX LPD tools, whether it be the SystemV or BSD implementation. Below is a list of print-related commands.

Table 8-1. New commands in chapter 8: Printing

Command	Meaning
lpr or **lp**	Print file
lpq or **lpstat**	Query print queue
lprm or **cancel**	Remove print job
acroread	PDF viewer
groff	Formatting tool
gv	PostScript viewer
printconf	Configure printers
xdvi	DVI viewer
xpdf	PDF viewer
***2ps**	Convert file to PostScript

8.5. Exercises

Configuring and testing printers involves being in the possession of one, and having access to the *root* account. If so, you may try:

- Installing the printer using the GUI on your system.
- Printing a test page using the GUI.
- Printing a test page using the **lp** command.
- Print from within an application, for example Mozilla or OpenOffice, by choosing File->Print from the menu.
- Disconnect the printer from the network or the local machine/print-server. What happens when you try to print something?

9.1.1.2. Incremental backups with tar

The **tar** tool supports the creation of incremental backups, using the `-N` option. With this option, you can specify a date, and **tar** will check modification time of all specified files against this date. If files are changed more recent than date, they will be included in the backup. The example below uses the timestamp on a previous archive as the date value. First, the initial archive is created and the timestamp on the initial backup file is shown. Then a new file is created, upon which we take a new backup, containing only this new file:

```
jimmy:~> tar cvpf /var/tmp/javaproggies.tar java/*.java
java/btw.java
java/error.java
java/hello.java
java/income2.java
java/income.java
java/inputdevice.java
java/input.java
java/master.java
java/method1.java
java/mood.java
java/moodywaitress.java
java/test3.java
java/TestOne.java
java/TestTwo.java
java/Vehicle.java

jimmy:~> ls -l /var/tmp/javaproggies.tar
-rw-rw-r-- 1 jimmy   jimmy   10240 Jan 21 11:58 /var/tmp/javaproggies.tar

jimmy:~> touch java/newprog.java

jimmy:~> tar -N /var/tmp/javaproggies.tar \
-cvp /var/tmp/incremental-javaproggies.tar java/*.java 2> /dev/null
java/newprog.java

jimmy:~> cd /var/tmp/

jimmy:~> tar xvf incremental-javaproggies.tar
java/newprog.java
```

Standard errors are redirected to `/dev/null`. If you don't do this, **tar** will print a message for each unchanged file, telling you it won't be dumped.

This way of working has the disadvantage that it looks at timestamps on files. Say that you download an archive into the directory containing your backups, and the archive contains files that have been created two years ago. When checking the timestamps of those files against the timestamp on the initial archive, the new files will actually seem old to **tar**, and will not be included in an incremental backup made using the `-N` option.

A better choice would be the `-g` option, which will create a list of files to backup. When making incremental backups, files are checked against this list. This is how it works:

```
jimmy:~> tar cvpf work-20030121.tar -g snapshot-20030121 work/
work/
work/file1
work/file2
work/file3

jimmy:~> file snapshot-20030121
```

9.2.2. Making a copy with a CD-writer

On some systems users are allowed to use the CD-writer device. Your data will need to be formatted first. Use the **mkisofs** command to do this in the directory containing the files you want to backup. Check with **df** that enough disk space is available, because a new file about the same size as the entire current directory will be created:

```
[rose@blob recordables] df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/hde5              19G   15G  3.2G  82% /home

[rose@blob recordables] du -h -s .
325M    .

[rose@blob recordables] mkisofs -J -r -o cd.iso .
<--snap-->
making a lot of conversions
<--/snap-->
98.95% done, estimate finish Fri Apr  5 13:54:25 2002
Total translation table size: 0
Total rockridge attributes bytes: 35971
Total directory bytes: 94208
Path table size(bytes): 452
Max brk space used 37e84
166768 extents written (325 Mb)
```

The `-J` and `-r` options are used to make the CD-ROM mountable on different systems, see the man pages for more. After that, the CD can be created using the **cdrecord** tool with appropriate options:

```
[rose@blob recordables] cdrecord dev 0,0,0 -speed=4 cd.iso
Cdrecord 1.10 (i686-pc-linux-gnu) (C) 1995-2001 Jörg Schilling
scsidev: '0,0,0'
scsibus: 0 target: 0 lun: 0
Linux sg driver version: 3.1.20
Using libscg version 'schily-0.5'
Device type    : Removable CD-ROM
Version        : 0
Response Format: 1
Vendor_info    : 'HP      '
Identification : 'CD-Writer+ 8100 '
Revision       : '1.0g'
Device seems to be: Generic mmc CD-RW.
Using generic SCSI-3/mmc CD-R driver (mmc_cdr).
Driver flags   : SWABAUDIO
Starting to write CD/DVD at speed 4 in write mode for single session.
Last chance to quit, starting real write in 0 seconds.
Operation starts.
```

Depending on your CD-writer, you now have the time to smoke^H^H^H^H^H eat a healthy piece of fruit and/or get a cup of coffee. Upon finishing the job, you will get a confirmation message:

```
Track 01: Total bytes read/written: 341540864/341540864
          (166768 sectors).
```

There are some graphical tools available to make it easier on you. One of the popular ones is **xcdroast**, which is freely available from the X-CD-Roast web site and is included on most systems and in the GNU directory. Both the KDE and Gnome desktop managers have facilities to make your own CDs.

9.2.3. Backups on/from jazz drives, USB devices and other removables

These devices are usually mounted into the file system. After the mount procedure, they are accessed as normal directories, so you can use the standard commands for manipulating files.

In the example below, images are copied from a USB camera to the hard disk:

```
robin:~> mount /mnt/camera

robin:~> mount | grep camera
/dev/sda1 on /mnt/camera type vfat (rw,nosuid,nodev)
```

If the camera is the only USB storage device that you ever connect to your system, this is safe. But keep in mind that USB devices are assigned entries in `/dev` as they are connected to the system. Thus, if you first connect a USB stick to your system, it will be on the `/dev/sda` entry, and if you connect your camera after that, it will be assigned to `/dev/sdb` - provided that you do not have any SCSI disks, which are also on `/dev/sd*`. On newer systems, since kernel 2.6, a hotplug system called HAL (Hardware Abstraction Layer) ensures that users don't have to deal with this burden. If you want to check where your device is, type **dmesg** after inserting it.

You can now copy the files:

```
robin:~> cp -R /mnt/camera/* images/

robin:~> umount /mnt/camera
```

Likewise, a jazz drive may be mounted on `/mnt/jazz`.

Appropriate lines should be added in `/etc/modules.conf` and `/etc/fstab` to make this work. Refer to specific hardware HOWTO's for more information. On systems with a 2.6.x kernel or higher, you may also want to check the man pages for **modprobe** and `modprobe.conf`.

9.2.4. Backing up data using a tape device

This is done using **tar** (see above). The **mt** tool is used for controlling the magnetic tape device, like `/dev/st0`. Entire books have been written about tape backup, therefore, refer to our reading-list in Appendix B for more information. Keep in mind that databases might need other backup procedures because of their architecture.

The appropriate backup commands are usually put in one of the *cron* directories in order to have them executed on a regular basis. In larger environments, the freely available Amanda backup suite or a commercial solution may be implemented to back up multiple machines. Working with tapes, however, is a system administration task beyond the scope of this document.

9.2.5. Tools from your distribution

Most Linux distributions offer their own tools for making your life easy. A short overview:

- SuSE: YaST now includes expanded backup and restore modules.
- RedHat: the File Roller tool provides visual management of (compressed) archives. They seem to be in favour of the X-CD-Roast tool for moving backups to an external device.
- Mandrake: X-CD-Roast.

9.4.1.2. GNU Privacy Guard

On Linux systems you will find GnuPG, the GNU Privacy Guard, which is a suite of programs that are compatible with the PGP (Pretty Good Privacy) tools that are commercially available.

In this guide we will only discuss the very simple usage of the encryption tools and show what you will need in order to generate an encryption key and use it to encrypt data for yourself, which you can then safely store in a public place. More advanced usage directions can be found in the man pages of the various commands.

9.4.2. Generate a key

Before you can start encrypting your data, you need to create a pair of keys. The pair consists of a private and a public key. You can send the public key to correspondents, who can use it to encrypt data for you, which you decrypt with your private key. You always keep the private key, never share it with somebody else, or they will be able to decrypt data that is only destined for you. Just to make sure that no accidents happen, the private key is protected with a password. The key pair is created using this command:

```
willy@ubuntu:~$ gpg --key-gen
gpg (GnuPG) 1.4.2.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions.  See the file COPYING for details.

gpg: directory `/home/willy.gnupg' created
gpg: new configuration file `/home/willy/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/willy/.gnupg/gpg.conf' are not yet
 active during this run
gpg: keyring `/home/willy/.gnupg/secring.gpg' created
gpg: keyring `/home/willy/.gnupg/pubring.gpg' created
Please select what kind of key you want:
    (1) DSA and Elgamal (default)
    (2) DSA (sign only)
    (5) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n month
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the
user ID from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Willy De Wandel
Email address: wdw@mvg.vl
Comment: Willem
You selected this USER-ID:
    "Willy De Wandel (Willem) <wdw@mvg.vl>"
```

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

Passphrase:
```

Now enetr your password. This can be a phrase, the longer, the better, the only condition is that you should be able to remember it at all times. For verification, you need to enter the same phrase again.

Now the key pair is generated by a program that spawns random numbers and that is, among other factors, fed with the activity data of the system. So it is a good idea to start some programs now, to move the mouse cursor or to type some random characters in a terminal window. That way, the chances to generate a number that contains lots of different digits will be much bigger and the key will be more difficult to crack.

9.4.3. About your key

When your key has been created, you will get a message about the *fingerprint*. This is a sequence of 40 hexadecimal numbers, which is so long that it is very, very hard to generate the same key twice, on any computer. You can be rather sure that this is a unique sequence. The short form of this key consists of your name, followed by the last 8 hexadecimal numbers.

You can get information about your key as follows:

```
willy@ubuntu:~$ gpg --list-keys
/home/willy/.gnupg/pubring.gpg
------------------------------
pub     1024D/BF5C3DBB 2006-08-08
uid                    Willy De Wandel (Willem) <wdw@myw.wl>
sub     4096g/A3449CF7 2006-08-08
```

The *key ID* of this key is "BF5C3DBB". You can send your key ID and your name to a *key server*, so that other people can get this info about you and use it to encrypt data for you. Alternatively, you can send your public key directly to the people who need it. The public part of your key is the long series of numbers that you see when using the `--export` option to the **gpg** command:

gpg --export -a

However, as far is this guide is concerned, we assume that you only need your key in order to encrypt and decrypt data for yourself. Read the **gpg** man pages if you want to know more.

9.4.4. Encrypt data

Now you can encrypt a `.tar` archive or a compressed archive, prior to saving it to a backup medium or transporting it to the backup server. Use the **gpg** command like this:

gpg -e -r *(part of) uid* archive

The `-e` option tells **gpg** to encrypt, the `-r` option indicates who to encrypt for. Keep in mind that only only the user name(s) following this `-r` option will be able to decrypt the data again. An example:

```
willy@ubuntu:~$ gpg -e -r Willy /var/tmp/home-willy-20060808.tar
```

We will only discuss some common UNIX/Linux text tools in this section.

10.2.6.1. The host command

To display information on hosts or domains, use the **host** command:

```
[emmy@pc10 emmy]$ host www.eunet.be
www.eunet.be. has address 193.74.208.177

[emmy@pc10 emmy]$ host -t any eunet.be
eunet.be. SOA dns.eunet.be. hostmaster.Belgium.EU.net.
  2002021300 28800 7200 604800 86400
eunet.be. mail is handled by 50 pophost.eunet.be.
eunet.be. name server ns.EU.net.
eunet.be. name server dns.eunet.be.
```

Similar information can be displayed using the **dig** command, which gives additional information about how records are stored in the name server.

10.2.6.2. The ping command

To check if a host is alive, use **ping**. If your system is configured to send more than one packet, interrupt **ping** with the **Ctrl+C** key combination:

```
[emmy@pc10 emmy]$ ping a.host.be
PING a.host.be (1.2.8.3) from 80.20.84.26: 56(84) bytes of data.
64 bytes from a.host.be(1.2.8.3):icmp_seq=0 ttl=241 time=99.977msec
--- a.host.be ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 99.977/99.977/99.977/0.000 ms
```


10.2.6.3. The traceroute command

To check the route that packets follow to a network host, use the **traceroute** command:

```
[emmy@pc10 emmy]$ /usr/sbin/traceroute www.eunet.be
traceroute to www.eunet.be(193.74.208.177),30 hops max,38b packets
1 blob (10.0.0.1)
       0.297ms  0.257ms  0.174ms
2 adsl-65.myprovider.be (217.136.111.1)
       12.120ms 13.058ms 13.009ms
3 194.78.255.177 (194.78.255.177)
       13.845ms 14.308ms 12.756ms
4 gigabitethernet2-2.intl2.gam.brussels.skynet.be (195.238.2.226)
       13.123ms 13.164ms 12.527ms
5 pecbru2.car.belbone.be (194.78.255.118)
       16.336ms 13.889ms 13.028ms
6 ser-2-1-110-ias-be-vil-ar01.kpnbelgium.be (194.119.224.9)
       14.602ms 15.546ms 15.959ms
7 unknown-195-207-939.eunet.be (195.207.93.49)
       16.514ms 17.661ms 18.889ms
8 S0-1-0.Leuven.Belgium.EU.net (195.207.129.1)
       22.714ms 19.193ms 18.432ms
9 dukat.Belgium.EU.net (193.74.208.178) 22.758ms * 25.263ms
```

On some systems, **traceroute** has been renamed to **tracepath**.

10.3.7. The Domain Name System

All these applications need DNS services to match IP addresses to host names and vice versa. A DNS server does not know all the IP addresses in the world, but networks with other DNS servers which it can query to find an unknown address. Most UNIX systems can run **named**, which is part of the BIND (Berkeley Internet Name Domain) package distributed by the Internet Software Consortium. It can run as a stand-alone caching *nameserver*, which is often done on Linux systems in order to speed up network access.

Your main client configuration file is `/etc/resolv.conf`, which determines the order in which Domain Name Servers are contacted:

```
search somewhere.org
nameserver 192.168.42.1
nameserver 193.74.208.137
```

More information can be found in the Info pages on **named**, in the `/usr/share/doc/bind[-<version>]` files and on the Bind project homepage. The DNS HOWTO covers the use of BIND as a DNS server.

10.3.8. DHCP

DHCP is the Dynamic Host Configuration Protocol, which is gradually replacing good old **bootp** in larger environments. It is used to control vital networking parameters such as IP addresses and name servers of hosts. DHCP is backward compatible with **bootp**. For configuring the server, you will need to read the HOWTO.

DHCP client machines will usually be configured using a GUI that configures the **dhcpcd**, the DHCP client daemon. Check your system documentation if you need to configure your machine as a DHCP client.

10.3.9. Authentication services

10.3.9.1. Traditional

Traditionally, users are authenticated locally, using the information stored in `/etc/passwd` and `/etc/shadow` on each system. But even when using a network service for authenticating, the local files will always be present to configure system accounts for administrative use, such as the root account, the daemon accounts and often accounts for additional programs and purposes.

These files are often the first candidates for being examined by hackers, so make sure the permissions and ownerships are strictly set as should be:

```
bob:~> ls -l /etc/passwd /etc/shadow
-rw-r--r--    1 root     root         1803 Mar 10 13:08 /etc/passwd
-r--------    1 root     root         1116 Mar 10 13:08 /etc/shadow
```

10.3.9.2. PAM

Linux can use PAM, the Pluggable Authentication Module, a flexible method of UNIX authentication. Advantages of PAM:

- A common authentication scheme that can be used with a wide variety of applications.

- PAM can be implemented with various applications without having to recompile the applications to specifically support PAM.
- Great flexibility and control over authentication for the administrator and application developer.
- Application developers do not need to develop their program to use a particular authentication scheme. Instead, they can focus purely on the details of their program.

The directory `/etc/pam.d` contains the PAM configuration files (used to be `/etc/pam.conf`). Each application or service has its own file. Each line in the file has four elements:

- *Module*:
  - `auth`: provides the actual authentication (perhaps asking for and checking a password) and sets credentials, such as group membership or Kerberos tickets.
  - `account`: checks to make sure that access is allowed for the user (the account has not expired, the user is allowed to log in at this time of day, and so on).
  - `password`: used to set passwords.
  - `session`: used after a user has been authenticated. This module performs additional tasks which are needed to allow access (for example, mounting the user's home directory or making their mailbox available).

  The order in which modules are stacked, so that multiple modules can be used, is very important.
- *Control Flags*: tell PAM which actions to take upon failure or success. Values can be `required`, `requisite`, `sufficient` or `optional`.
- *Module Path*: path to the pluggable module to be used, usually in `/lib/security`.
- *Arguments*: information for the modules

Shadow password files are automatically detected by PAM.

More information can be found in the **pam** man pages or at the Linux-PAM project homepage.

10.3.9.3. LDAP

The Lightweight Directory Access Protocol is a client-server system for accessing global or local directory services over a network. On Linux, the OpenLDAP implementation is used. It includes **slapd**, a stand-alone server; **slurpd**, a stand-alone LDAP replication server; libraries implementing the LDAP protocol and a series of utilities, tools and sample clients.

The main benefit of using LDAP is the consolidation of certain types of information within your organization. For example, all of the different lists of users within your organization can be merged into one LDAP directory. This directory can be queried by any LDAP-enabled applications that need this information. It can also be accessed by users who need directory information.

Other LDAP or X.500 Lite benefits include its ease of implementation (compared to X.500) and its well-defined Application Programming Interface (API), which means that the number of LDAP-enabled applications and LDAP gateways should increase in the future.

On the negative side, if you want to use LDAP, you will need LDAP-enabled applications or the ability to use LDAP gateways. While LDAP usage should only increase, currently there are not very many LDAP-enabled applications available for Linux. Also, while LDAP does support some access control, it does not possess as many security features as X.500.

Since LDAP is an open and configurable protocol, it can be used to store almost any type of information relating to a particular organizational structure. Common examples are mail address lookups, central authentication in combination with PAM, telephone directories and machine configuration databases.

See your system specific information and the man pages for related commands such as **ldapmodify** and **ldapsearch** for details. More information can be found in the LDAP Linux HOWTO, which discusses installation, configuration, running and maintenance of an LDAP server on Linux. The author of this Introduction to Linux document also wrote an LDAP Operations HOWTO, describing the basics everyone should know about when dealing with LDAP management, operations and integration of services.

10.4. Remote execution of applications

10.4.1. Introduction

There are a couple of different ways to execute commands or run programs on a remote machine and have the output, be it text or graphics, sent to your workstation. The connections can be secure or insecure. While it is of course advised to use secure connections instead of transporting your password over the network unencrypted, we will discuss some practical applications of the older (unsafe) mechanisms, as they are still useful in a modern networked environment, such as for troubleshooting or running exotic programs.

10.4.2. Rsh, rlogin and telnet

The **rlogin** and **rsh** commands for remote login and remote execution of commands are inherited from UNIX. While seldom used because they are blatantly insecure, they still come with almost every Linux distribution for backward compatibility with UNIX programs.

Telnet, on the other hand, is still commonly used often by system and network administrators. Telnet is one of the most powerful tools for remote access to files and remote administration, allowing connections from anywhere on the Internet. Combined with an X server, remote graphical applications can be displayed locally. There is no difference between working on the local machine and using the remote machine.

Because the entire connection is unencrypted, allowing **telnet** connections involves taking high security risks. For normal remote execution of programs, Secure SHell or **ssh** is advised. We will discuss the secure method later in this section.

However, **telnet** is still used in many cases. Below are some examples in which a mail server and a web server are tested for replies:

Checking that a mail server works:

```
[jimmy@blob ~] telnet mailserver 25
Trying 192.168.42.1...
Connected to mailserver.
Escape character is '^]'.
220 m1.some.net ESMTP Sendmail 8.11.6/8.11.6; 200302281626
ehlo some.net
250-m1.some.net Hello blob.some.net [10.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-SIZE
250-DSN
```

10.4.4.4. Secure remote copying

The SSH suite provides **scp** as a secure alternative to the **rcp** command that used to be popular when only **rsh** existed. **scp** uses **ssh** for data transfer, uses the same authentication and provides the same security as **ssh**. Unlike **rcp**, **scp** will ask for passwords or passphrases if they are needed for authentication:

```
lenny /var/tmp> scp Schedule.sdc.gz blob:/var/tmp/
lenny@blob's password:
Schedule.sdc.gz  100% |*****************************| 100 KB 00:00

lenny /var/tmp>
```

Any file name may contain a host and user specification to indicate that the file is to be copied to/from that host. Copies between two remote hosts are permitted. See the Info pages for more information.

If you would rather use an FTP-like interface, use **sftp**:

```
lenny /var/tmp> sftp blob
Connecting to blob...
lenny@blob's password:

sftp> cd /var/tmp

sftp> get Sch*
Fetching /var/tmp/Schedule.sdc.gz to Schedule.sdc.gz

sftp> bye

lenny /var/tmp>
```


Secure copy or FTP GUIs

Don't feel comfortable with the command line yet? Try Konqueror's capabilities for secure remote copy, or install Putty.

10.4.4.5. Authentication keys

The **ssh-keygen** command generates, manages and converts authentication keys for **ssh**. It can create RSA keys for use by SSH protocol version 1 and RSA or DSA keys for use by SSH protocol version 2.

Normally each user wishing to use SSH with RSA or DSA authentication runs this once to create the authentication key in `$HOME/.ssh/identity`, `id_dsa` or `id_rsa`. Additionally, the system administrator may use this to generate host keys for the system.

Normally this program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but *.pub* appended. The program also asks for a passphrase. The passphrase may be empty to indicate no passphrase (host keys must have an empty passphrase), or it may be a string of arbitrary length.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, a new key must be generated and copied to the corresponding public keys.

We will study SSH keys in the exercises. All information can be found in the man or Info pages.

ssh-keygen	Generate authentication keys for Secure SHell.
telnet	Make an insecure connection to another hosts.
tracepath/traceroute	Print the route that packets follow to another host.
whois	Get information abotu a domain name.
xclock	X Window clock application, handy for testing remote display.
xhost	X Window access control tool.

10.7. Exercises

10.7.1. General networking

- Display network information for your workstation: IP address, routes, name servers.
- Suppose no DNS is available. What would you do to reach your neighbour's machine without typing the IP address all the time?
- How would you permanently store proxy information for a text mode browser such as **links**?
- Which name servers handle the redhat.com domain?
- Send an E-mail to your local account. Try two different ways to send and read it. How can you check that it really arrived?
- Does your machine accept anonymous FTP connections? How do you use the **ncftp** program to authenticate with your user name and password?
- Does your machine run a web server? If not, make it do so. Check the log files!

10.7.2. Remote connections

- From your local workstation, display a graphical application, such as **xclock** on your neighbour's screen. The necessary accounts will have to be set up. Use a secure connection!
- Set up SSH keys so you can connect to your neighbour's machine without having to enter a password.
- Make a backup copy of your home directory in `/var/tmp` on your neighbour's "backup server," using **scp**. Archive and compress before starting the data transfer! Connect to the remote host using **ssh**, unpack the backup, and put one file back on the original machine using **sftp**.

10.7.3. Security

- Make a list of open (listening) ports on your machine.
- Supposing you want to run a web server. Which services would you deactivate? How would you do that?
- Install available updates.
- How can you see who connected to your system?
- Make a repetitive job that reminds you to change your password every month, and preferably the *root* password as well.

Appendix B. DOS versus Linux commands

In this appendix, we matched DOS commands with their Linux equivalent.

As an extra means of orientation for new users with a Windows background, the table below lists MS-DOS commands with their Linux counterparts. Keep in mind that Linux commands usually have a number of options. Read the Info or man pages on the command to find out more.

Table B-1. Overview of DOS/Linux commands

DOS commands	Linux command
<command> /?	man <command> or command `--help`
cd	cd
chdir	pwd
cls	clear
copy	cp
date	date
del	rm
dir	ls
echo	echo
edit	vim or other editor
exit	exit
fc	diff
find	grep
format	mke2fs or mformat
mem	free
mkdir	mkdir
more	more or even less
move	mv
ren	mv
time	date

Glossary

This section contains an alphabetical overview of commands discussed in this document.

A

a2ps
Format files for printing on a PostScript printer, see Section 8.1.2.
acroread
PDF viewer, see Section 8.1.2.2.
adduser
Create a new user or update default new user information.
alias
Create a shell alias for a command.
alsaconf
Configure sound card using the ALSA driver, see Section 11.1.2.
alsamixer
Tune ALSA sound device output, see Section 11.2.2.3.
anacron
Execute commands periodically, does not assume continuously running machine.
apropos
Search the whatis database for strings, see Section 5.3.2.2.
apt-get
APT package handling utility, see Section 7.5.3.2.
arecord
Record a sound sample, see Section 11.2.3.
aspell
Spell checker.
at, atq, atrm
Queue, examine or delete jobs for later execution, see Section 4.1.2.2 and Section 4.4.3.
aumix
Adjust audio mixer, see Section 11.2.2.3.
(g)awk
Pattern scanning and processing language.

B

bash
Bourne Again SHell, see Section 3.2.3.2 and Section 7.2.5.
batch
Queue, examine or delete jobs for later execution, see Section 4.1.2.2.
bg
Run a job in the background, see Section 4.1.2.1.
bitmap
Bitmap editor and converter utilities for the X window System.
bzip2
A block-sorting file compressor, see Section 9.1.1.3.

dmesg
Print or control the kernel ring buffer.
du
Estimate file space usage.
dump
Backup file system, see Section 9.2.5.

E

echo
Display a line of text, see Section 3.2.1.
ediff
Diff to English translator.
egrep
Extended grep.
eject
Unmount and eject removable media, see Section 7.5.5.2.
emacs
Start the Emacs editor, see Section 6.1.2.1.
exec
Invoke subprocess(es), see Section 4.1.5.1.
exit
Exit current shell, see Section 2.2.
export
Add function(s) to the shell environment, see Section 3.2.1, Section 3.2.1.2 and Section 7.2.4.2.

F

fax2ps
Convert a TIFF facsimile to PostScript, see Section 8.1.2.
fdformat
Format floppy disk, see Section 9.2.1.1.
fdisk
Partition table manipulator for Linux, see Section 3.1.2.2.
fetchmail
Fetch mail from a POP, IMAP, ETRN or ODMR-capable server, see Section 10.3.2.3.
fg
Bring a job in the foreground, see Section 4.1.2.1.
file
Determine file type, see Section 3.3.1.2.
find
Find files, see Section 3.3.3.3.
firefox
Web browser, see Section 10.3.3.2.
fork
Create a new process, see Section 4.1.5.1.
formail
Mail (re)formatter, see Section 10.3.2.3.
fortune
Print a random, hopefully interesting adage.