From DOS/Windows to Linux HOWTO

Table of Contents

From DOS/Windows to Linux HOWTO.....1
- By Guido Gonzato, ggonza at tin.it.....1
- 1.Introduction.....1
- 2.For the Impatient.....1
- 3.Meet bash.....1
- 4.Files and Programs.....1
- 5.Using Directories.....2
- 6.Floppies, Hard Disks, and the Like.....2
- 7.What About Windows?.....2
- 8.Tailoring the System.....2
- 9.Networking: Concepts.....2
- 10.A Bit of Programming.....2
- 11.The Remaining 1%.....2
- 12.The End, for Now.....3
- 1.Introduction.....3
- 1.1 Is Linux Right for You?.....3
- 1.2 It Is. Tell Me More.....4
  - Introductory Concepts.....4
  - Getting Help.....5
- 1.3 Conventions.....5
- 2.For the Impatient.....6
- 3.Meet bash.....7
- 4.Files and Programs.....7
- 4.1 Files: Preliminary Notions.....7
- 4.2 Symbolic Links.....8
- 4.3 Permissions and Ownership.....9
- 4.4 Files: Translating Commands.....10
  - Examples.....11
- 4.5 Running Programs: Multitasking and Sessions.....12
- 4.6 Running Programs on Remote Computers.....13
- 5.Using Directories.....14
- 5.1 Directories: Preliminary Notions.....14
- 5.2 Directories Permissions.....14
- 5.3 Directories: Translating Commands.....15
  - Examples.....15
- 6.Floppies, Hard Disks, and the Like.....16
- 6.1 Managing Devices the DOS Way.....16
- 6.2 Managing Devices the UNIX Way.....16
- 6.3 Backing Up.....18
- 7.What About Windows?.....18
- 8.Tailoring the System.....19
- 8.1 System Initialisation Files.....19
- 8.2 Program Initialisation Files.....20
- 9.Networking: Concepts.....21
- 10.A Bit of Programming.....21
- 10.1 Shell Scripts: .BAT Files on Steroids.....21
- 10.2 C for Yourself.....23

Table of Contents

10.3 X11 Programming....24
  Multi−Platform Programming....24
11.The Remaining 1%....25
11.1 Running DOS/Windows Apps....25
11.2 Using tar and gzip....25
11.3 Installing Applications 26
11.4 Tips You Can't Do Without....27
11.5 Where to Find Applications 27
11.6 A Few Things You Couldn't Do....27
11.7 Practicing UNIX under DOS/Win....29
11.8 Common Extensions and Related Programs....30
11.9 Converting Files....30
11.10 Free Office Suites....30
12.The End, for Now 31
12.1 Copyright....31
12.2 Disclaimer....31

From DOS/Windows to Linux HOWTO

By Guido Gonzato, `ggonza at tin.it`

Version 1.3.5. 31 August 2000.

This HOWTO is dedicated to all the (soon to be former?) DOS and Windows users who have decided to switch to Linux, the free UNIX clone. The purpose of this document is to help the reader translate his or her knowledge of DOS and Windows into the Linux environment, as well as providing hints on exchanging files and resources between the two OSes.

1.Introduction

- 1.1 Is Linux Right for You?
- 1.2 It Is. Tell Me More
- 1.3 Conventions

2.For the Impatient

3.Meet bash

4.Files and Programs

- 4.1 Files: Preliminary Notions
- 4.2 Symbolic Links
- 4.3 Permissions and Ownership
- 4.4 Files: Translating Commands
- 4.5 Running Programs: Multitasking and Sessions
- 4.6 Running Programs on Remote Computers

5.Using Directories

- 5.1 Directories: Preliminary Notions
- 5.2 Directories Permissions
- 5.3 Directories: Translating Commands

6.Floppies, Hard Disks, and the Like

- 6.1 Managing Devices the DOS Way
- 6.2 Managing Devices the UNIX Way
- 6.3 Backing Up

7.What About Windows?

8.Tailoring the System

- 8.1 System Initialisation Files
- 8.2 Program Initialisation Files

9.Networking: Concepts

10.A Bit of Programming

- 10.1 Shell Scripts: .BAT Files on Steroids
- 10.2 C for Yourself
- 10.3 X11 Programming

11.The Remaining 1%

- 11.1 Running DOS/Windows Apps
- 11.2 Using tar and gzip
- 11.3 Installing Applications
- 11.4 Tips You Can't Do Without
- 11.5 Where to Find Applications
- 11.6 A Few Things You Couldn't Do
- 11.7 Practicing UNIX under DOS/Win
- 11.8 Common Extensions and Related Programs
- 11.9 Converting Files

- 11.10 Free Office Suites

12.The End, for Now

- 12.1 Copyright
- 12.2 Disclaimer

1.Introduction

1.1 Is Linux Right for You?

Let's start politically correct. Throughout this document I say ``Linux", but I mean ``GNU/Linux". Please go to http://www.gnu.org/gnu/linux−and−gnu.html to see why.

You want to switch from the DOS/Windows world to Linux? Good idea: Linux is technically superior to DOS, Windows 9x and even Windows NT. But beware: it might not be useful for you. These are the main differences between DOS/Windows and Linux:

- Windows runs Microsoft Office and lots of games; is perceived to be easy to install and configure; is notoriously unstable; performs poorly; crashes are frequent.
- Linux runs StarOffice, scores of technical software and fewer games; can be tricky to install and configure; is rock solid; performs impeccably; crashes are extremely rare.

It's up to you to decide what you need. Furthermore, Linux gives you power, but it takes some time to learn how to harness it. Thus, if mostly need commercial sw, or if you don't feel like learning new commands and concepts, you had better look elsewhere. Be aware that many newcomers give up because of initial difficulties.

Work is underway to make Linux simpler to use, but *don't expect to be proficient with it unless you read a lot of documentation and use it at least for a few months*. Linux won't give you instant results. In spite of these warnings, I'm 100% confident that if you are the right user type you'll find in Linux your computer Nirvana. By the way, Linux + DOS/Win can coexist happily on the same machine.

Prerequisites for this howto: I'll assume that

- you know the basic DOS commands and concepts;
- Linux, possibly with X Window System (X11 for short), is properly installed on your PC;
- your shell (the equivalent of `COMMAND.COM`) is `bash`.

Unless specified, all information in this work is aimed at bad ol' DOS. There is information about Windows here and there, but bear in mind that Windows and Linux are totally different, unlike DOS that is sort of a UNIX poor relation.

Please also note that this work is neither a complete primer nor a configuration guide!

The latest version of this document is available in several formats on http://www.linuxdoc.org.

1.2 It Is. Tell Me More

You installed Linux and the programs you needed on the PC. You gave yourself an account (if not, type `adduser yourname`*now!*) and Linux is running. You've just entered your name and password, and now you are looking at the screen thinking: ``Well, now what?"

Now, don't despair. You're almost ready to do the same things you used to do with DOS/Win, and many more. If you were running DOS/Win instead of Linux, you would be doing some of the following tasks:

- running programs and creating, copying, viewing, deleting, printing, renaming files;
- CD'ing, MD'ing, RD'ing, and DIR'ring your directories;
- formatting floppies and copying files from/to them;
- tailoring the system;
- surfing the Internet;
- writing .BAT files and programs in your favourite language;
- the remaining 1%.

You'll be glad to know that these tasks can be accomplished under Linux in a fashion similar to DOS. Under DOS, the average user uses very few of the 100+ commands available: the same, up to a point, applies to Linux.

Introductory Concepts

The best way to learn something new is to get your feet wet. You are strongly encouraged to experiment and play with Linux: unless you login as ``root", you can't damage the system that way. A few points:

- first of all, how to quit Linux safely. If you see a text mode screen, press <CTRL−ALT−DEL>, wait for the system to reboot, then switch off the PC. If you are working under X Window System, press <CTRL−ALT−BACKSPACE> first, then <CTRL−ALT−DEL>. *Never* switch off or reset the PC directly: this could damage the file system;
- unlike DOS or Windows, Linux has built−in security mechanisms. Files and directories have permissions associated to them; as a result, some cannot be accessed by the normal user; (see Section Permissions and Ownership). DOS and Windows, on the contrary, will let you wipe out the entire contents of your hard disk;

- there's a special user called ``root'': the system administrator, with full power of life and death on the machine. If you work on your own PC, you'll be root as well. Working as root is *dangerous*: any mistake can seriously damage or destroy the system just like with DOS/Windows. Don't work as root unless absolutely necessary;
- much of the complexity of Linux comes from its extreme configurability: virtually every feature and every application can be tailored through one or more configuration files. Complexity is the price to pay for power;
- redirection and piping are a side DOS feature, a very inportant one and much more powerful under Linux. Simple commands can be strung together to accomplish complex tasks. I strongly suggest that you learn how to use them.

Getting Help

There are many ways to get help with Linux. The most important are:

- *reading the documentation*−−−I mean it. Although the HOWTO you are reading may serve as an introduction to Linux, there are several books that you really should read: at least, Matt Welsh's ``Linux Installation and Getting Started'' (http://www.linuxdoc.org/LDP/gs/gs.html) and the Linux FAQ (http://www.linuxdoc.org/FAQ/Linux−FAQ/index.html). Feel a guilty conscience until you have read at least one of them;
- the documentation of the packages installed on the machine is often found in subdirectories under /usr/doc/;
- to get some help about the ``internal commands'' of the shell, type `help` or, better, `man bash` or `info bash`;
- to get help about a command, type `man command` that invokes the manual (``man'') page of `command`. Alternatively, type `info command` that invokes, if available, the info page pertinent of `command`; info is a hypertext−based documentation system, perhaps not intuitive to use at first. Finally, you may try `apropos command` or `whatis command`. With all of these commands, press `q' to exit.
- finally, on the Internet: the right place for getting help is Usenet, like news:comp.os.linux.setup. Please don't email me for help, because I'm quite overloaded.

1.3 Conventions

Throughout this work, examples will often follow the following format: `<...>` is a required argument, while `[...]` an optional one. Example:

```
$ tar −tf <file.tar> [> redir_file]
```

`file.tar` must be indicated, but redirection to `redir_file` is optional.

``RMP'' means ``please Read the Man Pages for further information''. I can't stress enough how important reading the documentation is.

When the prompt of a command example is #, the command can only be performed by root.

2.For the Impatient

Want to strike out? Have a look at this table:

DOS	Linux	Notes
ATTRIB (+-)attr file	chmod <mode> file	completely different
BACKUP	tar -Mcvf device dir/	ditto
CD dirname\	cd dirname/	almost the same syntax
COPY file1 file2	cp file1 file2	ditto
DEL file	rm file	beware - no undelete
DELTREE dirname	rm -R dirname/	ditto
DIR	ls	not exactly the same syntax
DIR file /S	find . -name file	completely different
EDIT file	vi file	I think you won't like it
	jstar file	feels like dos' edit
EDLIN file	ed file	forget it
FORMAT	fdformat, mount, umount	quite different syntax
HELP command	man command, info command	same philosophy
MD dirname	mkdir dirname/	almost the same syntax
MORE < file	less file	much better
MOVE file1 file2	mv file1 file2	ditto
NUL	/dev/null	ditto
PRINT file	lpr file	ditto
PRN	/dev/lp0, /dev/lp1	ditto
RD dirname	rmdir dirname/	almost the same syntax
REN file1 file2	mv file1 file2	not for multiple files
RESTORE	tar -Mxpvf device	different syntax
TYPE file	less file	much better
WIN	startx	poles apart!

If you need more than a table of commands, please refer to the following sections.

3.Meet bash

Good news: with Linux you type much less at the prompt, because the `bash` shell types for you whenever possible, and features cool line editing capabilities. To begin with, the arrow−up key recalls previous command lines; but there's more. Pressing <TAB> completes file and directory names, so typing

```
      $ ls /uTABloTABbTAB
```

is like typing

```
      $ ls /usr/local/bin
```

If there were ambiguities, as typing

```
      $ ls /uTABloTABiTAB
```

`bash` stops because it doesn't know if you mean /usr/local/info or /usr/local/include. Supply more characters then press <TAB> again.

Other useful key presses are <ESC−BACKSPACE> that deletes a word to the left, while <ESC−D> deletes a word to the right; <ESC−F> moves the cursor one word to the right, <ESC−B> to the left; <CTRL−A> moves to the beginning of the line, <CTRL−E> to the end. The <ALT> key is equivalent to <ESC>.

Enough for now. Once you get used to these shortcuts, you'll find the DOS prompt very annoying...

4.Files and Programs

4.1 Files: Preliminary Notions

Linux has a structure of directories and files very similar to that of DOS/Win. Files have filenames that obey special rules, are stored in directories, some are executable, and among these most have command switches. Moreover, you can use wildcard characters, redirection, and piping. There are only a few minor differences:

- under DOS, file names are in the so−called 8.3 form; e.g. `NOTENOUG.TXT`. Under Linux we can do better. If you installed Linux using a file system like ext2 or umsdos, you can use longer filenames (up to 255 characters), and with more than one dot: for example, `This_is.a.VERY_long.filename`. Please note that I used both upper and lower case characters: in fact...
- upper and lower case characters in file names or commands are different. Therefore, `FILENAME.tar.gz` and `filename.tar.gz` are two different files. `ls` is a command, `LS` is a mistake;
- Windows users, beware when using long file names under Linux. If a file name contains spaces (not recommended but possible), you must enclose the file name in double quotes whenever you refer to it. For example:

  ```
  $ # the following command makes a directory called "My old files"
  $ mkdir "My old files"
  $ ls
  My old files    bin     tmp
  ```

  Further, some characters shouldn't be used: some of those are `!*$&#`.
- there are no compulsory extensions like .COM and .EXE for programs, or .BAT for batch files. Executable files are marked by an asterisk ``*' at the end of their name when you issue the `ls -F` command. For example:

  ```
  $ ls -F
  I_am_a_dir/   cindy.jpg    cjpg*   letter_to_Joe    my_1st_script*  old~
  ```

  The files `cjpg*` and `my_1st_script*` are executables, that is ``programs''. Under DOS, backup files end in .BAK, while under Linux they end with a tilde `~'. Further, a file whose name starts with a dot is considered as hidden. Example: the file `.I.am.a.hidden.file` won't show up after the `ls` command;
- DOS program switches are obtained with `/switch`, Linux switches with `−switch` or `−−switch`. Example: `dir /s`<tt> becomes `ls −R`. Note that many DOS programs, like `PKZIP` or `ARJ`, use UNIX−style switches.

You can now jump to Section Translating Commands from DOS to Linux, but if I were you I'd read on.

4.2 Symbolic Links

UNIX has a type of file that doesn't exist under DOS: the symbolic link. This can be thought of as a pointer to a file or to a directory, and can be used instead of the file or directory it points to; it's similar to Windows shortcuts. Examples of symbolic links are `/usr/X11`, which points to `/usr/X11R6`; `/dev/modem`, which points to either `/dev/ttyS0` or `/dev/ttyS1`.

To make a symbolic link:

```
$ ln −s <file_or_dir> <linkname>
```

Example:

```
$ ln −s /usr/doc/g77/DOC g77manual.txt
```

Now you can refer to `g77manual.txt` instead of `/usr/doc/g77/DOC`. Links appear like this in directory listings:

```
$ ls −F
g77manual.txt@
$ ls −l
(several things...)           g77manual.txt −> /usr/doc/g77/DOC
```

4.3 Permissions and Ownership

DOS files and directories have the following attributes: A (archive), H (hidden), R (read−only), and S (system). Only H and R make sense under Linux: hidden files start with a dot, and for the R attribute, read on.

Under UNIX a file has ``permissions'' and an owner, who in turn belongs to a ``group''. Look at this example:

```
$ ls −l /bin/ls
−rwxr−xr−x  1  root  bin  27281 Aug 15 1995 /bin/ls*
```

The first field contains the permissions of the file `/bin/ls`, which belongs to root, group bin. Leaving the remaining information aside, remember that `−rwxr−xr−x` means, from left to right:

`−` is the file type (`−` = ordinary file, `d` = directory, `l` = link, etc); `rwx` are the permissions for the file owner (read, write, execute); `r−x` are the permissions for the group of the file owner (read, execute); (I won't cover the concept of group, you can survive without it as long as you're a beginner ;−) `r−x` are the permissions for all other users (read, execute).

The directory `/bin` has permissions, too: see Section Directories Permissions for further details. This is why you can't delete the file `/bin/ls` unless you are root: you don't have the permission to do so. To change a file's permissions, the command is:

```
$ chmod <whoXperm> <file>
```

where who is `u` (user, that is owner), `g` (group), `o` (other), X is either `+` or `−`, perm is `r` (read), `w` (write), or `x` (execute). Common examples of `chmod` use are the following:

```
$ chmod +x file
```

this sets the execute permission for the file.

```
$ chmod go−rw file
```

this removes read and write permission for everyone but the owner.

```
$ chmod ugo+rwx file
```

this gives everyone read, write, and execute permission.

```
# chmod +s file
```

this makes a so−called ``setuid'' or ``suid'' file−−−a file that everyone can execute with its owner's privileges. Typically, you'll come across root suid files; these are often important system files, like the X server.

A shorter way to refer to permissions is with digits: `rwxr−xr−x` can be expressed as 755 (every letter corresponds to a bit: `−−−` is 0, `−−x` is 1, `−w−` is 2, `−wx` is 3...). It looks difficult, but with a bit of practice you'll understand the concept. root, being the superuser, can change everyone's file permissions. RMP.

4.4 Files: Translating Commands

On the left, the DOS commands; on the right, their Linux counterpart.

ATTRIB:	chmod
COPY:	cp
DEL:	rm
MOVE:	mv
REN:	mv
TYPE:	more, less, cat

Redirection and plumbing operators: `< > >> |`

Wildcards: `* ?`

`nul: /dev/null`

`prn, lpt1: /dev/lp0 or /dev/lp1; lpr`

Examples

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>ATTRIB +R FILE.TXT             $ chmod 400 file.txt
C:\GUIDO>COPY JOE.TXT JOE.DOC           $ cp joe.txt joe.doc
C:\GUIDO>COPY *.* TOTAL                 $ cat * > total
C:\GUIDO>COPY FRACTALS.DOC PRN          $ lpr fractals.doc
C:\GUIDO>DEL TEMP                       $ rm temp
C:\GUIDO>DEL *.BAK                      $ rm *~
C:\GUIDO>MOVE PAPER.TXT TMP\            $ mv paper.txt tmp/
C:\GUIDO>REN PAPER.TXT PAPER.ASC        $ mv paper.txt paper.asc
C:\GUIDO>PRINT LETTER.TXT               $ lpr letter.txt
C:\GUIDO>TYPE LETTER.TXT                $ more letter.txt
C:\GUIDO>TYPE LETTER.TXT                $ less letter.txt
C:\GUIDO>TYPE LETTER.TXT > NUL          $ cat letter.txt > /dev/null
        n/a                             $ more *.txt *.asc
        n/a                             $ cat section*.txt | less
```

Notes:

- `*` is smarter under Linux: `*` matches all files except the hidden ones; `.*` matches all hidden files (but also the current directory \``.`' and parent directory \``..`': beware!); `*.*` matches only those that have a \``.`' in the middle or that end with a dot; `p*r` matches both \`peter' and \`piper'; `*c*` matches both \`picked' and \`peck';
- when using `more`, press <SPACE> to read through the file, \`q' to exit. `less` is more intuitive and lets you use the arrow keys;
- there is no `UNDELETE`, so *think twice* before deleting anything;
- in addition to DOS' `< > >>`, Linux has `2>` to redirect error messages (stderr); moreover, `2>&1` redirects stderr to stdout, while `1>&2` redirects stdout to stderr;
- Linux has another wildcard: the `[]`. Usage: `[abc]*` matches files starting with a, b, c; `*[I-N1-3]` matches files ending with I, J, K, L, M, N, 1, 2, 3;
- `lpr` <file> prints a file in background. To check the status of the print queue, use `lpq`; to remove a file from the print queue, use `lprm`;
- there is no DOS−like `RENAME`; that is, `mv *.xxx *.yyy` won't work. A REN−like command is available on ftp://metalab.unc.edu/pub/Linux/utils/file;
- use `cp -i` and `mv -i` to be warned when a file is going to be overwritten.

4.5 Running Programs: Multitasking and Sessions

To run a program, type its name as you would do under DOS. If the directory (Section Using Directories) where the program is stored is included in the PATH (Section System Initialisation Files), the program will start. Exception: unlike DOS, under Linux a program located in the current directory won't run unless the directory is included in the PATH. Escamotage: being `prog` your program, type `./prog`.

This is what the typical command line looks like:

```
$ command [-s1 [-s2] ... [-sn]] [par1 [par2] ... [parn]] [< input] [> output]
```

where `-s1`, ..., `-sn` are the program switches, `par1`, ..., `parn` are the program parameters. You can issue several commands on the command line:

```
$ command1 ; command2 ; ... ; commandn
```

That's all about running programs, but it's easy to go a step beyond. One of the main reasons for using Linux is that it is a multitasking os−−−it can run several programs (from now on, processes) at the same time. You can launch processes in background and continue working straight away. Moreover, Linux lets you have several sessions: it's like having many computers to work on at once!

- To switch to session 1..6 on the virtual consoles, press <ALT−F1> ... <ALT−F6>
- To start a new session in the same v.c. without leaving the current one, type `su - <loginname>`. Example: `su - root`. This is useful, for instance, when you need to perform a task that only root can do.
- To end a session, type `exit`. If there are stopped jobs (see later), you'll be warned.
- To launch a process in background, add an ampersand '&' at the end of the command line:

  ```
  $ progname [-switches] [parameters] [< input] [> output] &
  [1] 123
  ```

  the shell identifies the process with a job number (e.g. `[1]`; see below), and with a PID (Process Identification Number; 123 in our example).
- To see how many processes there are, type `ps ax`. This will output a list of currently running processes.
- To kill (terminate) a process, type `kill <PID>`. You may need to kill a process when you don't know how to quit it the right way.... Unless you're root, you can't kill other people's processes. Sometimes, a process will only be killed by `kill -SIGKILL <PID>`. In addition, the shell allows you to stop or temporarily suspend a process, send a process to background, and bring a process from background to foreground. In this context, processes are called ``jobs''.
- To see how many jobs there are, type `jobs`. Here the jobs are identified by their job number, not by their PID.

- To stop a process running in foreground, press <CTRL−C> (it won't always work).
- To suspend a process running in foreground, press <CTRL−Z> (ditto).
- To send a suspended process into background, type `bg <%job>` (it becomes a job).
- To bring a job to foreground, type `fg <%job>`. To bring to foreground the last job sent to background, simply type `fg`.
- To kill a job, type `kill <%job>` where <job> may be 1, 2, 3,...

Using these commands you can format a disk, zip a bunch of files, compile a program, and unzip an archive all at the same time, and still have the prompt at your disposal. Try this with Windows, just to see the difference in performance (if it doesn't crash, of course).

4.6 Running Programs on Remote Computers

To run a program on a remote machine whose name is `remote.machine.edu`:

```
$ telnet remote.machine.edu
```

After logging in, start your favourite program. Needless to say, you must have a shell account on the remote machine.

If you have X11, you can even run an X application on a remote computer, displaying it on your X screen. Let `remote.machine.edu` be the remote X computer and let `local.linux.box` be your Linux machine. To run from `local.linux.box` an X program that resides on `remote.machine.edu`, do the following:

- fire up X11, start an `xterm` or equivalent terminal emulator, then type:

  ```
  $ xhost +remote.machine.edu
  $ telnet remote.machine.edu
  ```

- after logging in, type:

  ```
  remote:$ DISPLAY=local.linux.box:0.0
  remote:$ progname &
  ```

  (instead of `DISPLAY...`, you may have to write: `setenv DISPLAY local.linux.box:0.0`. It depends on the remote shell.)

Et voila! Now `progname` will start on `remote.machine.edu` and will be displayed on your machine. Don't try this over the modem though, for it's too slow to be usable. Moreover, this is a crude and insecure method: please read the ``Remote X Apps mini−HOWTO'' at http://www.linuxdoc.org/HOWTO/mini/Remote−X−Apps.html.

5.Using Directories

5.1 Directories: Preliminary Notions

We have seen the differences between files under DOS/Win and Linux. As for directories, under DOS/Win the root directory is `\`, under Linux it is `/`. Similarly, nested directories are separated by `\` under DOS/Win, by `/` under Linux. Example of file paths:

```
DOS:    C:\PAPERS\GEOLOGY\MID_EOC.TEX
Linux:  /home/guido/papers/geology/middle_eocene.tex
```

As usual, `..` is the parent directory and `.` is the current directory. Remember that the system won't let you `cd`, `rd`, or `md` everywhere you want. Each user has his or her stuff in a directory called `home', given by the system administrator; for instance, on my PC my home dir is `/home/guido`.

5.2 Directories Permissions

Directories, too, have permissions. What we have seen in Section Permissions and Ownership applies to directories as well (user, group, and other). For a directory, `rx` means you can `cd` to that directory, and `w` means that you can delete a file in the directory (according to the file's permissions, of course), or the directory itself.

For example, to prevent other users from snooping in `/home/guido/text`:

```
$ chmod o-rwx /home/guido/text
```

5.3 Directories: Translating Commands

```
DIR:            ls, find, du
CD:             cd, pwd
MD:             mkdir
RD:             rmdir
DELTREE:        rm −rf
MOVE:           mv
```

Examples

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR                            $ ls
C:\GUIDO>DIR FILE.TXT                   $ ls file.txt
C:\GUIDO>DIR *.H *.C                    $ ls *.h *.c
C:\GUIDO>DIR/P                          $ ls | more
C:\GUIDO>DIR/A                          $ ls −l
C:\GUIDO>DIR *.TMP /S                   $ find / −name "*.tmp"
C:\GUIDO>CD                             $ pwd
        n/a − see note                  $ cd
        ditto                           $ cd ~
        ditto                           $ cd ~/temp
C:\GUIDO>CD \OTHER                      $ cd /other
C:\GUIDO>CD ..\TEMP\TRASH               $ cd ../temp/trash
C:\GUIDO>MD NEWPROGS                    $ mkdir newprogs
C:\GUIDO>MOVE PROG ..                   $ mv prog ..
C:\GUIDO>MD \PROGS\TURBO                $ mkdir /progs/turbo
C:\GUIDO>DELTREE TEMP\TRASH             $ rm −rf temp/trash
C:\GUIDO>RD NEWPROGS                    $ rmdir newprogs
C:\GUIDO>RD \PROGS\TURBO                $ rmdir /progs/turbo
```

Notes:

- when using `rmdir`, the directory to remove must be empty. To delete a directory and all of its contents, use `rm −rf` (at your own risk).
- the character `~' is a shortcut for the name of your home directory. The commands `cd` or `cd ~` will take you to your home directory from wherever you are; the command `cd ~/tmp` will take you to `/home/your_home/tmp`.
- `cd −` ``undoes'' the last `cd`.

6.Floppies, Hard Disks, and the Like

There are two ways to manage devices under Linux: the DOS way and the UNIX way. Take your pick.

6.1 Managing Devices the DOS Way

Most Linux distributions include the Mtools suite, a set of commands that are perfectly equivalent to their DOS counterpart, but start with an `m': i.e., `mformat`, `mdir`, `mdel`, `mmd`, and so on. They can even preserve long file names, but not file permissions. If you configure Mtools editing a file called /etc/mtools.conf (a sample is provided in the distribution), you can also access the DOS/Win partition, the CD−−ROM, and the Zip drive. To format a fresh disk though, the `mformat` command won't do. As root, you'll have to issue this command beforehand: `fdformat /dev/fd0H1440`.

You can't access files on the floppy with a command like, say, `less a:file.txt`! This is the disadvantage of the DOS way of accessing disks.

6.2 Managing Devices the UNIX Way

UNIX has a different way to handle devices. There are no separate volumes like A: or C:; a disk, be it a floppy or whatever, becomes part of the local file system through an operation called ``mounting''. When you're done using the disk, before extracting it you must ``unmount'' it.

Physically formatting a disk is one thing, making a file system on it is another. The DOS command `FORMAT A:` does both things, but under Linux there are separate commands. To format a floppy, see above; to create a file system:

```
        # mkfs −t ext2 −c /dev/fd0H1440
```

You can use `dos`, `vfat` (recommended) or other formats instead of `ext2`. Once the disk is prepared, mount it with the command

```
        # mount −t ext2 /dev/fd0 /mnt
```

specifying the right file system if you don't use `ext2`. Now you can address the files in the floppy using `/mnt` instead of A: or B:. Examples:

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR A:                         $ ls /mnt
C:\GUIDO>COPY A:*.*                     $ cp /mnt/* .
C:\GUIDO>COPY *.ZIP A:                  $ cp *.zip /mnt
C:\GUIDO>EDIT A:FILE.TXT                $ jstar /mnt/file.txt
C:\GUIDO>A:                             $ cd /mnt
A:> _                                   /mnt/$ _
```

When you've finished, before extracting the disk you *must* unmount it with the command

```
# umount /mnt
```

Obviously, you have to `fdformat` and `mkfs` only unformatted disks, not previously used ones. If you want to use the drive B:, refer to `fd1H1440` and `fd1` instead of `fd0H1440` and `fd0` in the examples above.

Needless to say, what applies to floppies also applies to other devices; for instance, you may want to mount another hard disk or a CD−−ROM drive. Here's how to mount the CD−−ROM:

```
# mount −t iso9660 /dev/cdrom /mnt
```

This was the ``official'' way to mount your disks, but there's a trick in store. Since it's a bit of a nuisance having to be root to mount a floppy or a CD−−ROM, every user can be allowed to mount them this way:

- as root, do the following:

  ```
  # mkdir /mnt/floppy ; mkdir /mnt/cdrom
  # chmod 777 /mnt/floppy /mnt/cd*
  # # make sure that the CD−ROM device is right
  # chmod 666 /dev/hdb ; chmod 666 /dev/fd*
  ```
- add in /etc/fstab the following lines:

  ```
  /dev/cdrom      /mnt/cdrom  iso9660 ro,user,noauto          0       0
  /dev/fd0        /mnt/floppy vfat    user,noauto             0       0
  ```

Now, to mount a DOS floppy and a CD−−ROM:

```
$ mount /mnt/floppy
$ mount /mnt/cdrom
```

/mnt/floppy and /mnt/cdrom can now be accessed by every user. Remember that allowing everyone to mount disks this way is a gaping security hole, if you care.

Two useful commands are `df`, which gives information on the mounted file systems, and `du dirname` which reports the disk space consumed by the directory.

6.3 Backing Up

There are several packages to help you, but the very least you can do for a multi−volume backup is (as root):

```
# tar −M −cvf /dev/fd0H1440 dir_to_backup/
```

Make sure to have a formatted floppy in the drive, and several more ready. To restore your stuff, insert the first floppy in the drive and do:

```
# tar −M −xpvf /dev/fd0H1440
```

7.What About Windows?

The ``equivalent'' of Windows is the graphic system X Window System. Unlike Windows or the Mac, X11 wasn't designed for ease of use or to look good, but just to provide graphic facilities to UNIX workstations. These are the main differences:

- while Windows looks and feels the same all over the world, X11 does not: it's much more configurable. X11's overall look is given by a key component called ``window manager'', of which you have a wide choice: `fvwm`, basic but nice and memory efficient, `fvwm2−95`, `Afterstep`, `WindowMaker`, `Enlightenment`, and many more. The w.m. is usually invoked from `.xinitrc`;
- your w.m. can be configured so as a window acts as under, er, Windows: you click on it and it comes to foreground. Another possibility is that it comes to foreground when the mouse moves over it (``focus''). Also, the placement of windows on the screen can be automatic or interactive: if a strange frame appears instead of your program, left click where you want it to appear;
- most features can be tailored editing one or more configuration files. Read the docs of your w.m.: the configuration file can be `.fvwmrc`, `.fvwm2rc95`, `.steprc`, etc. A sample configuration file is typically found in /etc/X11/window−manager−name/system.window−manager−name;

- X11 applications are written using special libraries (``widget sets''); as several are available, applications look different. The most basic ones are those that use the Athena widgets (2−−D look; `xdvi`, `xman`, `xcalc`); others use Motif (`netscape`), others still use Tcl/Tk, Qt, Gtk, XForms, and what have you. Nearly all of these libraries provide roughly the same look and feel as Windows;
- the feel, unfortunately, can be incoherent. For instance, if you select a line of text using the mouse and press <BACKSPACE>, you'd expect the line to disappear, right? This won't work with Athena−−based apps, but it does with other widget sets;
- how scrollbars and resizing work depends on the w.m. and the widget set. Tip: if you find that the scrollbars don't behave as you would expect, try using the central button or the two buttons together to move them;
- applications don't have an icon by default, but they can have many. Most w.m. feature a menu you recall by clicking on the desktop (``root window''); needless to say, the menu can be tailored. To change the root window appearance, use `xsetroot` or `xloadimage`;
- the clipboard can only contain text, and behaves strange. Once you've selected text, it's already copied to the clipboard: move elsewhere and press the central button to paste it. There's an application, `xclipboard`, that provides for multiple clipboard buffers;
- drag and drop is an option, and is only available if you use X11 applications and/or w.m. that support it.

This said, good news for you. There are projects that aim at making X11 look and behave as coherently as Windows. Gnome, http://www.gnome.org, and KDE, http://www.kde.org, are simply awesome. Most likely your distribution uses either or both. You won't regret your Windows desktop anymore!

8.Tailoring the System

8.1 System Initialisation Files

Two important files under DOS are `AUTOEXEC.BAT` and `CONFIG.SYS`, which are used at boot time to initialise the system, set some environment variables like PATH and FILES, and possibly launch a program or batch file. Additionally, Windows has the infamous registry−−−one of the worst ideas ever conceived in computer science.

Under Linux there are lots of initialisation files, some of which you had better not tamper with until you know exactly what you are doing; they reside in the /etc tree. All configuration can be done editing plain text files. If all you need is setting the PATH and other environment variables, or you want to change the login messages or automatically launch a program on login, have a look at the following files:

```
FILES                                   NOTES

/etc/issue                              sets pre−login message
/etc/motd                               sets post−login message
```

```
/etc/profile                        sets $PATH and other variables, etc.
/etc/bashrc                         sets aliases and functions, etc.
/home/your_home/.bashrc             sets your aliases + functions
/home/your_home/.bash_profile   or
/home/your_home/.profile            sets environment + starts your progs
```

If the latter file exists (note that it is a hidden file), it will be read after the login, and the commands therein will be executed.

Example−−−look at this `.bash_profile`:

```
# I am a comment
echo Environment:
printenv | less   # equivalent of command SET under DOS
alias d='ls −l'   # easy to understand what an alias is
alias up='cd ..'
echo "I remind you that the path is "$PATH
echo "Today is `date`"  # use the output of the command 'date'
echo "Have a good day, "$LOGNAME
# The following is a "shell function"
ctgz() # List the contents of a .tar.gz archive.
{
  for file in $*
  do
    gzip −dc ${file} | tar tf −
  done
}
# end of .profile
```

`$PATH` and `$LOGNAME`, you guessed right, are environment variables. There are many others to play with; for instance, RMP for apps like `less` or `bash`.

Putting this line in your /etc/profile will provide the rough equivalent of `PROMPT $P$G`:

```
export PS1="\w\\$ "
```

8.2 Program Initialisation Files

Under Linux, virtually everything can be tailored to your needs. Most programs have one or more initialisation files you can fiddle with, often as a `.prognamerc` in your home dir. The first ones you'll want to modify are:

- `.inputrc`: used by `bash` to define key bindings;
- `.xinitrc`: used by `startx` to initialise X Window System;
- `.fvwmrc`: used by the window manager `fvwm`.

- `.joerc, .jstarrc`: used by the editor `joe`;
- `.jedrc`: used by the editor `jed`;
- `.pinerc`: used by the mail reader `pine`;
- `.Xdefault`: used by many X programs.

For all of these and the others you'll come across sooner or later, RMP. Perhaps I could interest you in the Configuration HOWTO, http://www.linuxdoc.org/HOWTO/Config−HOWTO.html?

9.Networking: Concepts

Not only is ``Dialup Networking'' available under Linux, it's also more stable and quicker. The name of the game is ``PPP'', the protocol employed for connecting to the Internet using modems. All you need is a tool that dials out and makes the connection.

To retrieve your mail from the ISP's server you need a tool called ``email fetcher'' that uses the POP protocol; when the mail is fetched it will appear as though it had been directly delivered to your Linux box. You'll then use a MUA (Mail User Agent) like `pine`, `mutt`, `elm` or many others to manage it.

While under Windows the dialer is automatically invoked when you launch an Internet application, under Linux the path is the other way round: you dial first, then launch the application. A thing called `diald` provides the usual behaviour. Installing and configuring dialup networking used to be one of the most difficult things to do under Linux, but not anymore: please consult the Configuration HOWTO.

Finally, a word about ``Network neighborhood'': you can make your Linux workstation appear as Windows NT/9x in a local network of Windows machines! The magic word is Samba: not the lively Brazilian dance, but an implementation of the SMB protocol for Linux. Go to http://samba.anu.edu.au/samba.

10.A Bit of Programming

10.1 Shell Scripts: .BAT Files on Steroids

If you used .BAT files to create shortcuts of long command lines (I did a lot), this goal can be attained by inserting appropriate alias lines (see example above) in `profile` or `.bash_profile`. But if your .BATs were more complicated, then you'll love the scripting language made available by the shell: it's as powerful as good ol' QBasic, if not more. It has variables, structures like while, for, case, if... then... else, and lots of other features: it can be a good alternative to a ``real'' programming language.

To write a script−−−the equivalent of a .BAT file under DOS−−−all you have to do is write a standard ASCII file containing the instructions, save it, then make it executable with the command `chmod +x <scriptfile>`. To execute it, type its name.

A word of warning. The system editor is called `vi`, and in my experience most new users find it very difficult to use. I'm not going to explain how to use it; please consult Matt Welsh's book or search for a tutorial on the net. Suffice it here to say that:

- to insert some text, type `i` then your text;
- to delete characters, type <ESC> then `x`;
- to quit `vi` whithout saving, type <ESC> then `:q!`
- to save and quit, type <ESC> then `:wq`.

A good beginner editor is `joe`: invoking it by typing `jstar` you'll get the same key bindings as the DOS/Win editor. `jed` in WordStar or IDE mode is even better. Please consult Section Where to Find Applications to see where to get these editors.

Writing scripts under `bash` is such a vast subject it would require a book by itself, and I will not delve into the topic any further. I'll just give you an example of shell script, from which you can extract some basic rules:

```
#!/bin/sh
# sample.sh
# I am a comment
# don't change the first line, it must be there
echo "This system is: `uname -a`" # use the output of the command
echo "My name is $0" # built-in variables
echo "You gave me the following $# parameters: "$*
echo "The first parameter is: "$1
echo -n "What's your name? " ; read your_name
echo notice the difference: "hi $your_name" # quoting with "
echo notice the difference: 'hi $your_name' # quoting with '
DIRS=0 ; FILES=0
for file in `ls .` ; do
  if [ -d ${file} ] ; then # if file is a directory
    DIRS=`expr $DIRS + 1`  # DIRS = DIRS + 1
  elif [ -f ${file} ] ; then
    FILES=`expr $FILES + 1`
  fi
  case ${file} in
    *.gif|*jpg) echo "${file}: graphic file" ;;
    *.txt|*.tex) echo "${file}: text file" ;;
    *.c|*.f|*.for) echo "${file}: source file" ;;
    *) echo "${file}: generic file" ;;
  esac
done
echo "there are ${DIRS} directories and ${FILES} files"
ls | grep "ZxY--%%WKW"
if [ $? != 0 ] ; then # exit code of last command
  echo "ZxY--%%WKW not found"
fi
echo "enough... type 'man bash' if you want more info."
```

10.2 C for Yourself

Under UNIX, the system language is C, love it or hate it. Scores of other languages (Java, FORTRAN, Pascal, Lisp, Basic, Perl, awk...) are also available.

Taken for granted that you know C, here are a couple of guidelines for those of you who have been spoilt by Turbo C++ or one of its DOS kin. Linux's C compiler is called `gcc` and lacks all the bells and whistles that usually accompany its DOS counterparts: no IDE, on−line help, integrated debugger, etc. It's just a rough command−line compiler, very powerful and efficient. To compile your standard `hello.c` you'll do:

```
        $ gcc hello.c
```

which will create an executable file called `a.out`. To give the executable a different name, do

```
        $ gcc −o hola hello.c
```

To link a library against a program, add the switch −l<libname>. For example, to link in the math library:

```
        $ gcc −o mathprog mathprog.c −lm
```

(The `−l<libname>` switch forces `gcc` to link the library `/usr/lib/lib<libname>.so`; so `−lm` links `/usr/lib/libm.so`).

So far, so good. But when your prog is made of several source files, you'll need to use the utility `make`. Let's suppose you have written an expression parser: its source file is called `parser.c` and #includes two header files, `parser.h` and `xy.h`. Then you want to use the routines in `parser.c` in a program, say, `calc.c`, which in turn #includes `parser.h`. What a mess! What do you have to do to compile `calc.c`?

You'll have to write a so−called `Makefile`, which teaches the compiler the dependencies between sources and objects files. In our example:

```
# This is Makefile, used to compile calc.c
# Press the <TAB> key where indicated!

calc: calc.o parser.o
<TAB>gcc −o calc calc.o parser.o −lm
# calc depends on two object files: calc.o and parser.o

calc.o: calc.c parser.h
<TAB>gcc −c calc.c
# calc.o depends on two source files
```

```
parser.o:  parser.c parser.h xy.h
<TAB>gcc −c parser.c
# parser.o depends on three source files

# end of Makefile.
```

Save this file as `Makefile` and type `make` to compile your program; alternatively, save it as `calc.mak` and type `make −f calc.mak`, and of course RMP. You can invoke some help about the C functions, that are covered by man pages, section 3; for example,

```
$ man 3 printf
```

To debug your programs, use `gdb`. `info gdb` to learn how to use it.

There are lots of libraries available; among the first you may want to use are `ncurses` (textmode effects), and `svgalib` (console graphics). Many editors can act as an IDE; `emacs` and `jed`, for instance, also feature syntax highlighting, automatic indent, and so on. Alternatively, get the package `rhide` from ftp://metalab.unc.edu:/pub/Linux/devel/debuggers/. It's a Borland IDE clone, and chances are that you'll like it.

10.3 X11 Programming

If you feel brave enough to tackle X11 programming (it's not that difficult), there are several libraries that make writing X11 programs a breeze. The main sites to visit are those of GTK+, http://www.gtk.org, and Qt, http://www.troll.no. Gtk+ is a C−based widget set originally written for the graphic package The GIMP (http://www.gimp.org), and is used by the Gnome environment. Kdeveloper is based on C++−based Qt, used by KDE. Most likely, you'll use one of these.

Some of the best tools for visual programming are Kdevelop for Qt, http://www.kdevelop.org, and Glade for GTK+, http://glade.pn.org. This page has more information: http://www.free−soft.org/guitool/.

Multi−Platform Programming

Wouldn't it be nice if you could write code that compiled seamlessly under Linux *and* Windows using `gcc`? As of this writing, there are some widget sets that allow for more−or−less stable multi−platform programming. As far as stability and completeness are concerned though, I would say that the choice is narrowed down to only one: FLTK, the Fast Light Tool Kit http://www.fltk.org. It's amazingly small, quick, and stable. It also has a semi−visual builder called Fluid.

11.The Remaining 1%

Much more than 1%, actually...

11.1 Running DOS/Windows Apps

Yes, you can to some extent run DOS and Windows applications under Linux! There are two emulators that are quite good: Dosemu (http://www.dosemu.org) and Wine (http://www.winehq.com). The latter is getting better release after release, and the list of runnable applications is getting larger. It even runs Word and Excel!

11.2 Using tar and gzip

Under UNIX there are some widely used applications to archive and compress files. `tar` is used to make archives−−−it's like `PKZIP` or `Winzip`but it doesn't compress, it only archives. To make a new archive:

```
$ tar cvf <archive_name.tar> <file> [file...]
```

To extract files from an archive:

```
$ tar xvf <archive_name.tar> [file...]
```

To list the contents of an archive:

```
$ tar tf <archive_name.tar> | less
```

You can compress files using `compress`, which is obsolete and shouldn't be used any more, or `gzip`:

```
$ compress <file>
$ gzip <file>
```

that creates a compressed file with extension `.Z` (`compress`) or `.gz` (`gzip`). These programs can compress only one file at a time. To decompress:

```
$ compress −d <file.Z>
$ gzip −d <file.gz>
```

RMP.

There are also the `unarj`, `zip` and `unzip` (PK??ZIP compatible) utilities. Files with extension `.tar.gz` or `.tgz` (archived with `tar`, then compressed with `gzip`) are as common in the UNIX world as .ZIP files are under DOS. Here's how to list the contents of a `.tar.gz` archive:

```
$ tar ztf <file.tar.gz> | less
```

11.3 Installing Applications

First of all: installing packages is root's work. Most Linux applications are distributed as a `.tar.gz` archive, which typically will contain a directory aptly named containing files and/or subdirectories. A good rule is to install these packages from `/usr/local` with the command

```
# tar zxf <archive.tar.gz>
```

reading then the README or INSTALL file. In most cases, the application is distributed in source, which you'll have to compile; often, typing `make` then `make install` will suffice. If the archive contains a `configure` script, run it first. Obviously, you'll need the `gcc` or `g++` compiler.

Other archives have to be unpacked from /; this is the case with Slackware's `.tgz` archives. Other archives contain the files but not a subdirectory − careful not to mess things up! Always list the contents of the archive before installing it.

Debian and Red Hat have their own archive format; respectively, `.deb` and `.rpm`. The latter is widely used by many distributions; to install an `rpm` package, type

```
# rpm −i package.rpm
```

11.4 Tips You Can't Do Without

Backscrolling: pressing <SHIFT + PAG UP> (the grey key) allows you to backscroll a few pages, depending on how much video memory you have.

Resetting the screen: if you happen to `more` or `cat` a binary file, your screen may end up full of garbage. To fix it, blind type `reset` or this sequence of characters: `echo CTRL−V ESC c RETURN`.

Pasting text: in console, see below; in X, click and drag to select the text in an `xterm` window, then click the middle button (or the two buttons together if you have a two−button mouse) to paste. There is also `xclipboard` (alas, only for text); don't get confused by its very slow response.

Using the mouse: if you installed `gpm`, a mouse driver for the console, you can click and drag to select text, then right click to paste the selected text. It works across different VCs.

Messages from the kernel: have a look at `/var/adm/messages` or `/var/log/messages` as root to see what the kernel has to tell you, including bootup messages. The command `dmesg` is also handy.

11.5 Where to Find Applications

If you're wondering whether you can replace your old and trusted DOS/Win application with a Linux one, I suggest that you browse the main Linux software repository: ftp://metalab.unc.edu/pub/Linux. Other good starting places are the ``Linux Applications and Utilities Page'' [http://www.xnet.com/~blatura/linapps.shtml](http://www.xnet.com/~blatura/linapps.shtml), the ``official'' Linux page http://www.linux.org, and http://freshmeat.net.

11.6 A Few Things You Couldn't Do

Linux can do an awful lot of things that were cumbersome, difficult or impossible do to with DOS/Windows. Here's a short list that may whet your appetite:

- `at` allows you to run programs at a specified time;
- `awk` is a simple yet powerful language to manipulate data files (and not only). For example, being `data.dat` your multi field data file,

  ```
  $ awk '$2 ~ "abc" {print $1, "\t", $4}' data.dat
  ```

  prints out fields 1 and 4 of every line in `data.dat` whose second field contains ``abc''.
- `cron` is useful to perform tasks periodically, at specified date and time. Type `man 5 crontab`.
- `file <filename>` tells you what `filename` is (ASCII text, executable, archive, etc.);

- `find` (see also Section Directories: Translating Commands) is one of the most powerful and useful commands. It's used to find files that match several characteristics and perform actions on them. General use of `find` is:

  ```
  $ find <directory> <expression>
  ```

  where <expression> includes search criteria and actions. Examples:

  ```
  $ find . −type l −exec ls −l {} \;
  ```

  finds all the files that are symbolic links and shows what they point to.

  ```
  $ find / −name "*.old" −ok rm {} \;
  ```

  finds all the files matching the pattern and deletes them, asking for your permission first.

  ```
  $ find . −perm +111
  ```

  finds all the files whose permissions match 111 (executable).

  ```
  $ find . −user root
  ```

  finds all the files that belong to root. Lots of possibilities here−−−RMP.
- `grep` finds text patterns in files. For example,

  ```
  $ grep −l "geology" *.tex
  ```

  lists the files *.tex that contain the word ``geology''. The variant `zgrep` works on gzipped files. RMP;
- **regular expressions** are a complex but darn powerful way of performing search operations on text. For example, `^a[^a−m]X{4,}txt$` matches a line that starts with `a', followed by any character except those in the interval a−m, followed by 4 or more `X', and ends in `txt'. You use regular expressions with advanced editors, `less`, and many other applications. `man grep` for an introduction.
- `script <script_file>` dumps the screen contents on `script_file` until you issue the command `exit`. Useful for debugging;
- `sudo` allows users to perform some of root's tasks (e.g. formatting and mounting disks; RMP);
- `uname −a` gives you info about your system;
- `zcat` and `zless` are useful for browsing and piping gzipped files without decompressing them. For example:

  ```
  $ zless textfile.gz
  $ zcat textfile.gz | lpr
  ```

- The following commands often come in handy: `bc, cal, chsh, cmp, cut, fmt, head, hexdump, nl, passwd, printf, sort, split, strings, tac, tail, tee,` `touch, uniq, w, wall, wc, whereis, write, xargs, znew.` RMP.

11.7 Practicing UNIX under DOS/Win

Believe it or not, there are fine tools that provide a UNIX−like environment under DOS/Windows! One is the Djgpp suite (http://www.delorie.com/djgpp/) for DOS, while Cygwin (http://www.cygnus.com/cygwin) is a more complex port for Win32. Both include the same GNU development tools and utilities as Linux; you won't get the same stability and performance, though.

If you'd like to have a taste of Linux, try out Djgpp. Download and install the following files (as of this writing, the latest version is 2.02): `djdev202.zip, bnu281b.zip, bsh1147b.zip, fil316b.zip, find41b.zip, grep22b.zip, gwk303b.zip, lss332b.zip, shl112b.zip.`. Installation instructions are provided, and you can find assistance on news:comp.os.msdos.djgpp.

In particular, using `bash` under DOS/Win is a whiff of fresh air. To configure it properly, edit the supplied file `BOOT.BAT` to reflect your installation, then put these files in your home directory (in the Windows partition) instead of those provided:

```
# this is _bashrc

LS_OPTIONS="−F −s −−color=yes"
alias cp='cp −i'
alias d='ls −l'
alias l=less
alias ls="ls $LS_OPTIONS"
alias mv='mv −i'
alias rm='rm −i'
alias u='cd ..'
```

```
# this is _bprof
if [ −f ~/_bashrc ]; then
  . ~/_bashrc
fi
PS1='\w\$ '
PS2='> '
CDPATH="$CDPATH:~"
# stuff for less(1)
LESS="−M−Q"                    # long prompt, silent
LESSEDIT="%E ?lt+%lt. %f"      # edit top line
VISUAL="jed"                   # editor
LESSCHARSET=latin1             # visualise accented letters
export PS1 PS2 CDPATH LS_OPTIONS LESS LESSEDIT LESSOPEN VISUAL LESSCHARSET
```

11.8 Common Extensions and Related Programs

You may come across scores of file extensions. Excluding the more exotic ones (i.e. fonts, etc.), here's a list of who's what:

- `1 ... 8`: man pages. Read them with `groff −Tascii −man <file.1>`.
- `arj`: archive made with `arj`.
- `dvi`: output file produced by TeX (see below). `xdvi` to visualise it; `dvips` to turn it into a PostScript `.ps` file.
- `gz`: archive made with `gzip`.
- `info`: info file (sort of alternative to man pages). Get `info`.
- `lsm`: Linux Software Map file. It's a plain ASCII file containing the description of a package.
- `ps`: PostScript file. To visualise or print it get `gs` and, optionally, `ghostview` or `gv`.
- `rpm`: Red Hat package. You can install it on any system using the package manager `rpm`.
- `taz, tar.Z`: archive made with `tar` and compressed with `compress`.
- `tgz, tar.gz`: archive made with `tar` and compressed with `gzip`.
- `tex`: text file to submit to TeX, a powerful typesetting system. Get the package `tex`, available in many distributions.
- `texi`: texinfo file, can produce both TeX and info files (cp. `info`). Get `texinfo`.
- `xbm, xpm, xwd`: graphic file.
- `Z`: archive made with `compress`.

11.9 Converting Files

If you need to exchange text files between DOS/Win and Linux, be aware of the ``end of line" problem. Under DOS, each line of text ends with CR/LF (that is, ASCII 13 + ASCII 10), with LF under Linux. If you edit a DOS text file under Linux, each line will likely end with a strange−−looking `M' character; a Linux text file under DOS will appear as a kilometric single line with no paragraphs. There are a couple of tools, `dos2unix` and `unix2dos`, to convert the files.

If your text−−only files contain accented characters, make sure they are made under Windows (with Notepad) and not under plain DOS; otherwise, all accented characters will be screwed up.

11.10 Free Office Suites

Yes, you can have for free what would otherwise cost a lot of money!

StarOffice (http://www.sun.com/staroffice.) is currently the only choice, though Koffice is down the pipeline (http://www.koffice.org). StarOffice is big and slow, but very good anyway: it offers a lot of functionality not found in Microsoft Office. It can also read and write Word and Excel files, although the conversion isn't always perfect.

Another good package is Corel WordPerfect, a free edition of which is available for download. Need I say more? Go fetch it: http://www.corel.com.

12.The End, for Now

Congratulations! You have now grasped a little bit of UNIX and are ready to start working. Remember that your knowledge of the system is still limited, and that you are expected to do more practice with Linux to use it comfortably. But if all you had to do was get a bunch of applications and start working with them, what I included here is enough.

I'm sure you'll enjoy using Linux and will keep learning more about it−−−everybody does. I bet, too, that you'll never go back to DOS/Win! I hope I made myself understood and did a good service to my 3 or 4 readers.

12.1 Copyright

Copyright (c) by Guido Gonzato, `ggonza at tin.it`. This document may be distributed only subject to the terms and conditions set forth in the LDP License at http://www.linuxdoc.org/COPYRIGHT.html, except that this document must not be distributed in modified form without the author's consent.

If you have questions, please refer to the Linux Documentation Project home page, http://www.linuxdoc.org

12.2 Disclaimer

This document is provided ``as is". I put great effort into writing it as accurately as I could, but you use the information contained in it at your own risk. In no event shall I be liable for any damages resulting from the use of this work.

Many thanks to Matt Welsh, the author of ``Linux Installation and Getting Started", to Ian Jackson, the author of ``Linux frequently asked questions with answers", to Giuseppe Zanetti, the author of the book ``Linux", to all the folks who emailed me suggestions, and especially to Linus Torvalds and GNU who gave us Linux.

Feedback is welcome. For any requests, suggestions, flames, etc., feel free to contact me.

Enjoy Linux and life,

Guido `=8-)`