

NAME

intro – introduction to user commands

DESCRIPTION

Section 1 of the manual describes user commands and tools, for example, file manipulation tools, shells, compilers, web browsers, file and image viewers and editors, and so on.

NOTES

Linux is a flavor of UNIX, and user commands under UNIX work similarly under Linux (and lots of other UNIX-like systems too, like FreeBSD).

Under Linux, there are GUIs (graphical user interfaces), where you can point and click and drag, and hopefully get work done without first reading lots of documentation. The traditional UNIX environment is a CLI (command line interface), where you type commands to tell the computer what to do. This is faster and more powerful, but requires finding out what the commands are and how to use them. Below is a bare minimum guide to get you started.

Login

In order to start working, you'll probably first have to open a session. The program **login**(1) will wait for you to type your username and password, and after that, it will start a *shell* (command interpreter) for you. In case of a graphical login, you get a screen with menus or icons and a mouse click will start a shell in a window. See also **xterm**(1).

The shell

One types commands into the *shell*, the command interpreter. It is not built-in; it is just another program. You can change your shell, and everybody has their own favorite one. The standard one is called *sh*. See also **ash**(1), **bash**(1), **chsh**(1), **csch**(1), **dash**(1), **ksh**(1), **zsh**(1).

A session might look like this:

```
knuth login: aeb
Password: *****
$ date
Tue Aug  6 23:50:44 CEST 2002
$ cal
      August 2002
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

$ ls
bin  tel
$ ls -l
total 2
drwxrwxr-x   2 aeb          1024 Aug  6 23:51 bin
-rw-rw-r--   1 aeb           37 Aug  6 23:52 tel
$ cat tel
maja      0501-1136285
peter     0136-7399214
$ cp tel tel2
$ ls -l
total 3
drwxr-xr-x   2 aeb          1024 Aug  6 23:51 bin
-rw-r--r--   1 aeb           37 Aug  6 23:52 tel
-rw-r--r--   1 aeb           37 Aug  6 23:53 tel2
$ mv tel tel1
```

```
$ ls -l
total 3
drwxr-xr-x  2 aeb          1024 Aug  6 23:51 bin
-rw-r--r--  1 aeb           37 Aug  6 23:52 tel1
-rw-r--r--  1 aeb           37 Aug  6 23:53 tel2
$ diff tel1 tel2
$ rm tel1
$ grep maja tel2
maja      0501-1136285
$
```

Here typing Control-D ended the session.

The `$` here was the command prompt—it is the shell's way of indicating that it is ready for the next command. The prompt can be customized in lots of ways, and one might include stuff like username, machine name, current directory, time, and so on. An assignment `PS1="What next, master? "` would change the prompt as indicated.

We see that there are commands *date* (that gives date and time), and *cal* (that gives a calendar).

The command *ls* lists the contents of the current directory—it tells you what files you have. With a *-l* option it gives a long listing, that includes the owner and size and date of the file, and the permissions people have for reading and/or changing the file. For example, the file "tel" here is 37 bytes long, owned by aeb and the owner can read and write it, others can only read it. Owner and permissions can be changed by the commands *chown* and *chmod*.

The command *cat* will show the contents of a file. (The name is from "concatenate and print": all files given as parameters are concatenated and sent to "standard output" (see **stdout(3)**), here the terminal screen.)

The command *cp* (from "copy") will copy a file.

The command *mv* (from "move"), on the other hand, only renames it.

The command *diff* lists the differences between two files. Here there was no output because there were no differences.

The command *rm* (from "remove") deletes the file, and be careful! it is gone. No wastepaper basket or anything. Deleted means lost.

The command *grep* (from "g/re/p") finds occurrences of a string in one or more files. Here it finds Maja's telephone number.

Pathnames and the current directory

Files live in a large tree, the file hierarchy. Each has a *pathname* describing the path from the root of the tree (which is called `/`) to the file. For example, such a full pathname might be `/home/aeb/tel`. Always using full pathnames would be inconvenient, and the name of a file in the current directory may be abbreviated by giving only the last component. That is why `/home/aeb/tel` can be abbreviated to `tel` when the current directory is `/home/aeb`.

The command *pwd* prints the current directory.

The command *cd* changes the current directory.

Try alternatively *cd* and *pwd* commands and explore *cd* usage: "`cd`", "`cd .`", "`cd ..`", "`cd /`", and "`cd ~`".

Directories

The command *mkdir* makes a new directory.

The command *rmdir* removes a directory if it is empty, and complains otherwise.

The command *find* (with a rather baroque syntax) will find files with given name or other properties. For example, "`find . -name tel`" would find the file *tel* starting in the present directory (which is called `.`). And "`find / -name tel`" would do the same, but starting at the root of the tree. Large searches on a multi-GB disk will be time-consuming, and it may be better to use **locate(1)**.

Disks and filesystems

The command *mount* will attach the filesystem found on some disk (or floppy, or CDROM or so) to the big filesystem hierarchy. And *umount* detaches it again. The command *df* will tell you how much of your disk is still free.

Processes

On a UNIX system many user and system processes run simultaneously. The one you are talking to runs in the *foreground*, the others in the *background*. The command *ps* will show you which processes are active and what numbers these processes have. The command *kill* allows you to get rid of them. Without option this is a friendly request: please go away. And "kill -9" followed by the number of the process is an immediate kill. Foreground processes can often be killed by typing Control-C.

Getting information

There are thousands of commands, each with many options. Traditionally commands are documented on *man pages*, (like this one), so that the command "man kill" will document the use of the command "kill" (and "man man" document the command "man"). The program *man* sends the text through some *pager*, usually *less*. Hit the space bar to get the next page, hit q to quit.

In documentation it is customary to refer to man pages by giving the name and section number, as in **man(1)**. Man pages are terse, and allow you to find quickly some forgotten detail. For newcomers an introductory text with more examples and explanations is useful.

A lot of GNU/FSF software is provided with info files. Type "info info" for an introduction on the use of the program *info*.

Special topics are often treated in HOWTOs. Look in */usr/share/doc/howto/en* and use a browser if you find HTML files there.

SEE ALSO

ash(1), bash(1), chsh(1), csh(1), dash(1), ksh(1), locate(1), login(1), man(1), xterm(1), zsh(1), wait(2), stdout(3), man-pages(7), standards(7)