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.

All commands yield a status value on termination. This value can be tested (e.g., in most shells the variable *?* contains the status of the last executed command) to see whether the command completed successfully. A zero exit status is conventionally used to indicate success, and a nonzero status means that the command was unsuccessful. (Details of the exit status can be found in wait(2).) A nonzero exit status can be in the range 1 to 255, and some commands use different nonzero status values to indicate the reason why the command failed.

NOTES

Linux is a flavor of UNIX, and as a first approximation all user commands under UNIX work precisely the same under Linux (and FreeBSD and lots of other UNIX-like systems).

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. That is faster and more powerful, but requires finding out what the commands are. Below a bare minimum, to get started.

Login

In order to start working, you probably first have to login, that is, give your username and password. See also login(1). The program*lo gin* now starts 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 to the *shell*, the command interpreter. It is not built-in, but is just a program and you can change your shell. Everybody has her own favorite one. The standard one is called *sh*. See also**ash(1),** bash(1), **csh(1), zsh(1),** chsh(1).

A session might go like

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
%
```

and 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, here the terminal screen.)

The command *cp* (from copy) will copy a file. On the other hand, the command *mv* (from move) 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 cd / and pwd and cd and pwd.

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* detac hes 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 sho w you which processes are active and what numbers these processes have. The command*kill* allo ws 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/shar e/doc/howto/en* and use a browser if you find HTML files there.

SEE ALSO

standards(7)

COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man-pages/.