ABC of UNIX

J-C. Soetens
jean-christophe.soetens@u-bordeaux.fr

Translation v1 : **Bassey Oboho**

Contents

1 Introducing UNIX

Unix was born in 1969 at Bell Laboratories (subsidiary of ATT) under the impetus ok Ken Thompson and Dennis Ritchie to meet a custom internal. Their objective was to develop a system interactive operating system for small machines equipped with possibilities comparable to those of large systems. Unix will then be written in C language (invented by Dennis Ritchie) From 1973, which will therefore make it portable on a large number of computers.

UNIX is a multi-tasking, multi-user operating system. As an operating system, its main role is to ensure an optimal distribution of resources (memory, processor (s), disks ...) between the different tasks of the different users.

The main functions of UNIX are:

- *Login and logout*: once the administrator of the system has registered a user, the system is in charge of checking the identity of the user when he wants to connect. A command interpreter (shell) allowing a dialogue user / system is then automatically activated.
- *Resource management* of the installation and sharing these resources between users and different tasks.
- *Data management*: this consists of the organization, maintenance and access to storage units (memory, hard disks, magnetic tapes ...)
- *Communication between users*: this is for example mail électronic or file transfers.
- *Programming environment*: these are the compilers (C, C ++, Fortran ...), text editors, programming assistance tools (debuggers ...).

1.1 Development of a working session

The *work session* is the time interval between the user login and logout. On connection, the system will ensure the identity of the user and if it is recognized, a dialog protocol (a shell) will automatically be established. The disconnection consists in indicating to the operating system that the session of work is finished.

1.2 Connection procedure

It is during this step that the user will have to identify himself from the system. This identification takes place in the way next :

- Display of the message **login:** after which you must enter your username (user id or uid).
- Display of the message **Password:** after which you must enter his password. This is not displayed while typing for prevent someone else from seeing it.

After connection, various messages from the administration of the system (words of the day, presence of mail in the box letters, etc ...) are displayed. The user is indeed ready to work when it receives the system prompt consisting of a marker at the beginning of the line (the prompt). This marker is variable depending on the machines (ex : **$** ou **nom_machine**[**numero**] > and can be redefining the user.

1.3 Password change

At the first connection or for safety, the user may have to change his password. This change is made with the command **passwd**. You can change your password by first entering the password current password then twice the new password. Example usage by user "gromit":
$ passwd
Changing password for "gromit"
gromit's Old password:
gromit's New password:
Enter the new password again:
If the new password is rejected, the reason may be that the second new password entry (for confirmation) was different from the first or, possibly, that the new password does not answer not to security requirements.

1.4 Logout procedure

This depends on the type of session that has been opened. If a graphical environment (such as the one created by X Windows, the multi-window environment) is in place, it usually exists a "logout" menu (or "exit") which allows you to exit this environment and thus to end the work session. Sometimes also this menu does not allow just quitting the graphical environment. Then you have to proceed to the disconnection step below. In the absence of a graphical environment, a simple command such as **logout** or **exit** is sufficient to end the working session.

2 Command interpreter

Once logged in, the user can submit commands to the system. *Any command entered will be interpreted by the interpreter of commands (or shell).* The term shell has been choosen for express the idea of an interface wrapping the kernel of the system and which establishes a communication between this kernel and the user.

There are many UNIX command interpreters. The main ones (found on most systems) are the Bourne-shell (sh), the C-shell (csh), the Korn Shell (ksh) and the T-CShell (tcsh). The choice of the interpreter activated on connection is made by the administrator of the system upon user registration and in most cases it this is the C-shell or the Korn-shell.

2.1 Command syntax

name _command [`options`] [*arguments*]

- the separator character between the different elements of the command is blank (Space key on the keyboard).
- options usually start with the character - (sign minus) followed by one or more key letters. These options will modify the command behavior. Square brackets around arguments and options mean that these are optional.
- the arguments specify the objects (files or variables) on which command will apply.

While entering the command, the user can correct his typing using the Delete or Backspace keys. When the user has when the entry is finished, it submits the command to the system by pressing the Enter key. Some shells have *mechanisms reminder of* commands which allow r úse (as is or after modification) a command previously used.

2.2 Elementary commands

ls	get the list of files in the current directory.
ls `-l`	`l` like long, gives all file attributes.
ls `-la`	`a` like all, also list files starting ç starting with the character ”.”
cat *f1*	display the contents of the file f1 on the screen.
cp *f1 f2*	copy from file *f1* to file f2 (**c** o **p** y).
mv *f1 f2*	rename the file f1 to f2 (**m** o **v** e).
rm *f1*	destroy file f1 (r e m ote).

Notes:

- All shells distinguish between lowercase letters and uppercase for commands and file names (unlike the from efunt MS-DOS).
- We have on-line help under UNIX allowing access to the rules of use and functionality of a command. For it, just issue the command:
  man *name _command.*

2.3 Special shell characters

When entering an order (**name _command** [`options`] [*arguments*]), all or part of the *argument* can be designated as c on subtle with the help of special characters:

?	denotes any any character.
∗	denotes a sequence (which may be empty) of characters.
[...]	designates a character from the list.
[^...]	designates a character that does not belong to the list.

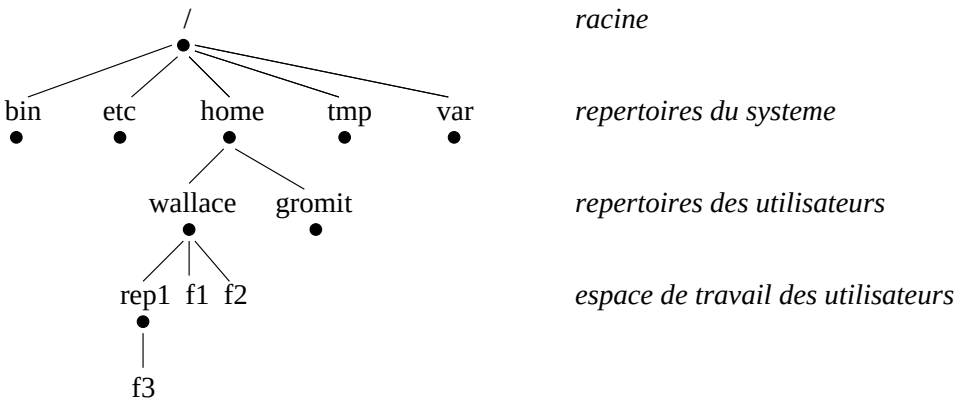
Example: either a current directory such as:
ls gives `file1.bin file1.txt file2.txt file10.txt file.txt readme zzz`
so

ls *file1* ∗	gives `file1.bin file1.txt file10.txt`
ls *file* ∗ *.txt*	gives `file1.txt file2.txt file10.txt file.txt`
ls *file* [*0-9*]∗ *.txt*	gives `file1.txt file2.txt file10.txt`
ls *???*	give zzz

3 Tree structure of UNIX files

The information unit managed by the system is the file. According to their use, the files are called directories or all files. short. A directory is a catalog of files containing their characteristics like access rights, size, date of creation... *The backbone of the system is a tree structure* of files (files) and directories (directories). Each user can create At will in his workspace of new

files and directories.



3.1 Files and directories

Important Notions:

/ is the name of the tree root directory.

. designates the current directory.

.. designates the parent directory (just above).

Current directory: position in the tree at a given time.

Home directory: root of the tree structure of a user (home directory) and current directory each time connection. For example / home / gromit is the home directory of growled.

Absolute name: Uniquely denotes a file in starting from the root. Each file has a unique absolute name in the system. For example / home / wallace / rep1 / f3

Relative name: designates a file from the current directory. For example dir1 / f3 if the current directory is / home / wallace. Two files with the same name can coexist if they are in two different directories because they have an absolute name different.

3.2 Navigate in the tree structure

cd *name _directory*	allows to change directory (**c** hange **d** irectory).
cd ..	allows you to go back to the above directory.
cd	without argument, the command takes you to your home directory.
mkdir *rep1*	create the directory *rep1* (**m** a **k** e **dir** ectory).
rmdir *rep1*	delete the directory *rep1* (**r** e **m** ote **dir** ectory).

Notes:

- to create a file *f1* in a directory x is equivalent to (i) create the file *f1* and (ii) add the name *f1* in the list of names contained in x
- creating a directory y in a directory x is equivalent to (i) add the name y in the list of names contained in x and (ii) create a y file and associate an empty list with it.

3.3 Access rights to files

Each file (or directory) has a set of attributes defining the access rights to this file for all system users.

3.3.1 User classes

There are 3 classes of users who can optionally access To a file:

- the owner of the file (User).
- the group to which the owner belongs (Group).
- the others (Others).

When it is created, a file belongs to its author. The file owner can then distribute or restrict the access rights to this file as we will see later.

3.3.2 Types of access

For each user class, there are 3 types of access to a data file:

- in reading (Read).
- in Write.
- in execution (eXecute).

At the directory level, these rights mean:

- Read: right to list the files found in this directory.
- Write: right to create or delete a file located there.
- executed: right to traverse this directory.

3.3.3 Control and visualization of access rights

For this, we use the command **ls** `-l`.

```
home / wallace > ls -l
total 3
-rw-r - r-- 1 wallace usr 12 Dec 18 12:14 f1
-rw-r - r-- 1 wallace usr 297 Dec 18 12:14 f2
drwxr-xr-x 2 wallace usr 512 Dec 18 12:14 rep1
```

The first character specified if the file is a directory (character `d`) or a file (character -). The 9 characters following identify the access rights (presence of the right if letter `r, w` or `x`) in the order of user (u), group (g) and others (o).

3.3.4 Modification of access rights

Only the owner of a file can modify his access rights. For that, it uses the command **chmod** with the syntax next: **chmod** `who` ± `what` *name _file*
with `qui = u, g, o, a` and `what = r, w, x`, − to remove rights and + to add more.

Examples:
`home / wallace` > **chmod** `or` *f1*
`home / wallace` > **chmod** `g + w` *f2*
`home / wallace` > **chmod** `go-rx` *rep1*

The result of these commands is:

```
home / wallace > ls -l
total 3
-rw-rw ---- 1 wallace usr 12 Dec 18 12:14 f1
-rw-rw-r-- 1 wallace usr 297 Dec 18 12:14 f2
drwx ------ 2 wallace usr 512 Dec 18 12:14 rep1
```

4 Inputs / outputs and pipe

Usually the commands read the standard input and / or write to standard output. *By default, the standard input is the keyboard and the standard output is the screen.* It is possible to redirect this standard input and output to files using > and < characters.

4.1 Redirection of standard output

So that the result of a command is stored in a file at the instead of appearing on the screen, use the syntax:

name _command [`options`] [*arguments*] > *file _output*

If the file *file _output* does not exist, it is created and will contain the order result. If it existed before, its content is is overwritten. If we want this content is preserved and add the result of a command, you must use the redirection >>

4.2 Redirection of standard input

Likewise, instead of providing data by entering it on the keyboard, these data can be read from a file with the syntax:

name _command [`options`] [*arguments*] < *file _output*

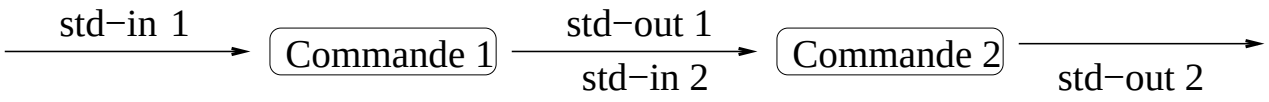
Example:
wc `-l` *file1*: indicates on the screen (standard output) the number of lines of the file file1.
ls `-l` *file1* > temp; **wc** `-l` *temp*: same result than the previous command.

4.3 Pipe

In the same order of ideas as the redirection of standard / output, we can use the *mechanism of pipe* (tube) which allows to take the standard output of a first command and redirect it to the standard input of a second order. The syntax uses the character | as follows:

name _command1 [`options`] [*arguments*] | **name _command2** [`options`] [*arguments*]

This sequence of commands can be sh ématized as follows:



Example:
ls `-l` *file* ∗ | **wc** `-l`

5 Search tools

Unix offers powerful file search or search tools in files.

5.1 Search for strings in a file

The **grep** command displays the lines of the data files in arguments which contain a given pattern:

grep `options` pattern *files*

Example:
grep subroutine * *.f*
allows to display on the screen all the lines of the files of the current directory with extension ".f" and containing the pattern "subroutine".

Basic options:
`-i` ignores the lowercase / uppercase distinction.
`-v` display lines which do not contain the pattern.
`-n` print out lines and their numbers that do not contain the pattern.

5.2 Search for a file

The **find** command descends recursively in subtrees of directories, seeking to apply to files specified by one or more selection criteria (name, type, date of modification, etc.), a given

command. **find** is used as ç on:

find *list _of _r éxpression directories*

where *list _of _ directories* is the list of tree roots To browse and *expression* is a series of options expressing the file selection criteria and actions to apply to them. When the criterion is true, the action is executed.

Example of selection criteria:

`-name "pattern"`	true if a current filename contains the pattern string.
`-user username`	true if a current file belongs to user username.
`-mtime n`	true if the file has been modified in the last n days (+ n to express "n and more" and -n to express "n and less").

Example of actions to be performed on the selected files:

`-print`	display the name of the current file.
`-ls`	list of information about the current file.

Examples:
find. -name "* .f" -print
allow to search recursively from the current directory "." all files with the ".f" extension.

find / home / gromit -name "*" -print | xargs -i grep -i "Wrong Trousers"
allows recursive search from directory / home / gromit all files and display the lines that contain the string of "Wrong Trousers" characters without distinguishing between upper and lower case.