Introduction to UNIX: basic concepts

P. Regnauld <regnauld@eu.org>

AfNOG 2006

1. Content

- history
- introduction to UNIX design (kernel, virtual memory, multi-user, filesystem, access-rights)
- basic concepts of UNIX (sessions, manuals, users, groups, processes, terminals, shells, signals, job control, ...)

2. History

- long history: over 25 years old (On the Early History and Impact of Unix[1] , Ronda Hauben).
- 1969 at Bell Laboratories (pun on «Multics») -- K. Thompson / B. Kernighan
- C language around 1974 (B. Kernighan / D.Richie)
- Ported to several platforms in C («Portable OS»)
- dsitributed to laboratories / Universities for small fee ($150)
- 1978: V7 is the first commercial version
- two main «branches»: BSD (4.4BSD) and SVSTEM V (SVR4), and now Linux
- most ported operating system -- free versions available

Today: over 50 variants (see UGU). Most known:

- Solaris -- Sun Microsystems
- HP-UX -- Hewlett Packard
- AIX -- IBM
- Irix -- Silicon Graphics, Inc.
- MacOS X / Darwin -- Apple

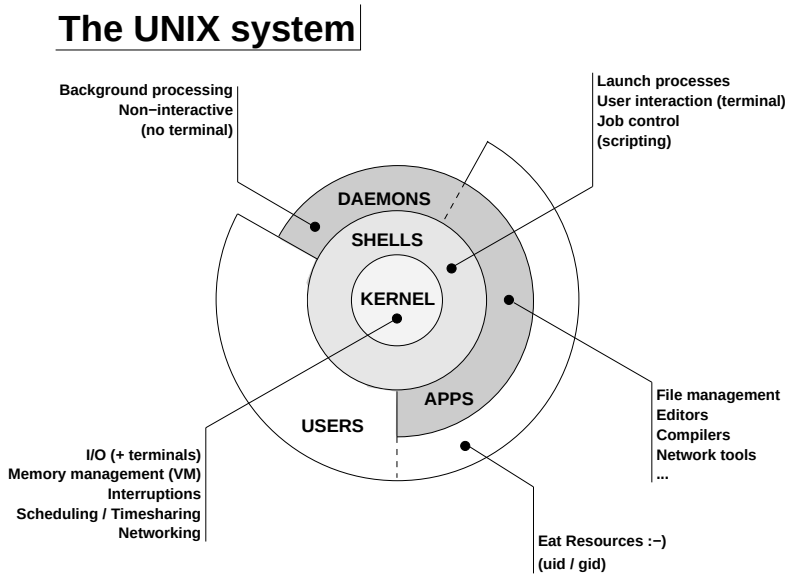
1. <URL:http://internet-history.org/archives/early.history.of.unix.html>

- GNU/Linux -- various (Debian, RedHat, Ubuntu, SuSE, Gentoo, Mandriva, ...)
- FreeBSD, NetBSD, OpenBSD

3. UNIX design

There are 4 main layers to the UNIX system

1. the kernel
2. the shell
3. applications
4. users



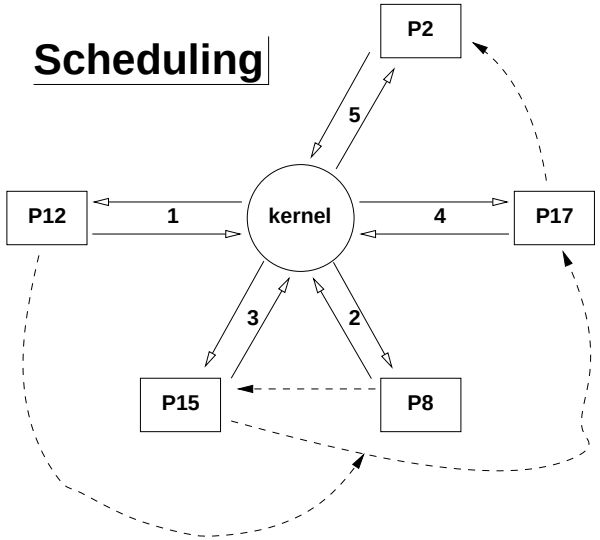
UNIX filesystem layout

3.1 the Kernel

- Can be monolithic (Linux, BSD) or micro-kernel (OS X/Darwin)
- preemptive multitasking
- takes care of:
  - I/O (disk, terminals, printers, tapes, ...)

- basic peripheral control (disk, serial, NICs...)
- memory management (allocation, recovery, Virtual Memory / swap)
- timesharing / scheduling of processes
- networking (IP, etc...)

The kernel is always running. It schedules running tasks one at a time (context switching) so that they get an equal share of the CPU time.



Scheduling of processes

3.2 Virtual memory

On UNIX, the total memory is composed of the physical memory + an on-disk area, commonly called "swap" -- this together represents virtual memory.

Programs can address the total amount of memory in the system (T = Physical + Virtual). The swap is usually 2 x the amount of physical RAM in the system.

3.3 Multi-user

Several users can be logged in to the system simultaneously, running each multiple processes (timesharing + scheduling).

Each user has shared access to the resources (disk, printers, memory, ...) according to his privileges.

3.4 the Filesystem

3.4.1 The UNIX philosophy: Everything is a file

Most of the resources are accessible through "special files" in the filesystem. For example:

- `/dev/da*, /dev/ad*` -- the hard disks and partitions
- `/dev/tty*, /dev/pty*` -- terminals and pseudo-terminals
- `/dev/lpt0` -- the printer
- ...

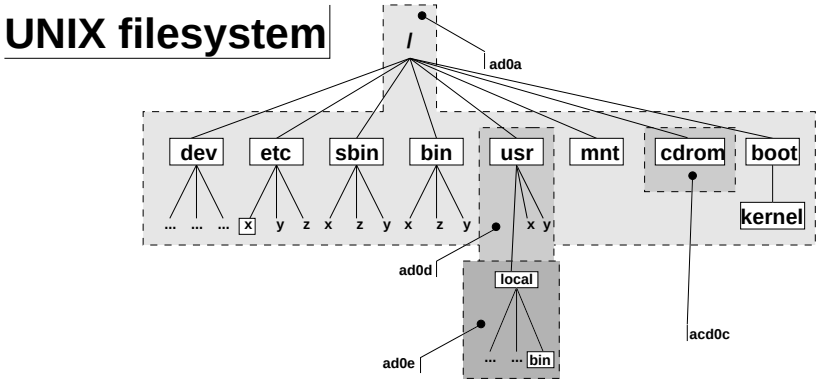
Other special files:

- named pipes (FIFO): allow communication between processes
- unix sockets: another form of interprocess communication
- links: pointers to other files/directories in the filesystem

The special files above can be manipulated like any other file: open, read, write, close.

Additionally, some special filesystems exist which contain "virtual" files - for example, /proc in Linux.

3.4.2 Hierarchy



UNIX filesystem layout

The filesystem tree has only one root: `/`. There are no multiple top-level roots like on DOS/Windows (`C:, D:, ...`).

Additional partitions, removal mediums (CD-ROMs), network filesystems are *mounted on top of* an existing directory (see figure).

Depending on the type of mount, the content of the directory *below* is visible or not.

3.5 Access-rights

Files and directories are protected with *access-rights*. They consist of two categories:

1. permissions (Read, Write, Execute)
2. ownership (User, Group, Others)

Below is a sample output of the `ls -l` command

```
- rwx r-x r-x   2   root   staff    [...]   filename
|  |   |   |    |    |       |
|  |   |   |    |    |       +-- belongs to group "staff" as well
|  |   |   |    |    |
|  |   |   |    |    +-- belongs to the user "root"
|  |   |   |    |
|  |   |   |    +-- link count (number of name references to this file)
|  |   |   |
|  |   |   +-- permissions for "others" (Read and Execute)
|  |   |
|  |   +-- permissions for the group (Read and Execute)
|  |
|  +-- permissions for the owner (user) (Read, Write and Execute)
|
+-- File type: normal  (d = directory, l = link, p = named pipe, ...)
```

Additionally, certain extra permissions exist, such as setuid and setgid, which are not explained here.

4 flags for each file:

1. file type:
   - `-` plain file
   - `d` directory
   - `p` named pipe
   - `s` socket
   - `l` link
   - `b` block device (disk, tape)
   - `c` character device (terminal, serial port, parallel port)
2. Read, Write and eXecute bits for the *U*ser (owner of the file)
3. Read, Write and eXecute bits for the *G*roup
4. Read, Write and eXecute bits for the *O*thers

To manipulate these access rights, two commands:

- `chown`: change ownership For example: `chown root:staff file` will change the owner and group of the file to respectively root and staff
- `chmod`: change mode. For example: `chmod o-rx file` or `chmod 755 file`

The 755 above represents the bit-values of the fields above, i.e.:


```
4   4   4
 2   2   2    <- values
  1   1   1
rwx r-x r-x   <- permissions

111 101 101   <- bit

 7   5   5    <- decimal
```

3.5.1 Various example combinations

`---x--x--x  1 bin   bin  65322 26 Jun 19:42 makelist`
: An execute-only program: the user can only run the program, but cannot read the content of the program file.

`d-wx-wx-wx  1 bin   bin      4 15 Jan 20:54 private`
: A directory than one can `cd` to (change directory), but cannot list files, or write files. *But* if the user knows the name of a "hidden" file, he can access it.

`d-w--w--w-  1 bin   bin      4 15 Jan 20:54 private`
: Special case: here it is not possible to even write files in the directory: for this to be possible, the "x" but should be present so that the shell can at least check to see if the file already exists.

3.5.2 Special bits

3.5.2.1 Setuid / Setgid

The 'x' (eXecute) bit for the *U*ser and the *G*roup can be made *setuid* (SET User ID):

```
-r-sr-xr-x  1 bob  users    12288 21 May 10:43 program
```

In the above example:

- Any user can run the program
- When the program runs, it will run with the privileges of user 'bob', since the access rights for *U*ser are `r-s` (setuid).

Other example:

```
-r-sr-x---  1 root  wheel    12288 21 May 10:43 program
```

Here:

- Only users of the group 'wheel' and root itself can launch the program
- When it runs, it does with the privileges of the 'root' user

This makes it possible to restrict commands to a certain group of users.

This above is also true with the *setgid* (SET Group ID) bit:

```
-r-xr-sr-x  1 bin  bin    12288 21 May 10:43 program
```

In this case:

- all users can run the program
- when the program runs, it does with the privileges of group 'bin'

3.5.2.2 The sticky bit

On a directory, it is possible to set what is called the 'sticky' bit:

```
drwxrwxrwt  1 bin  staff      4 21 May 10:43 files/
```

This means that all users can create files in this directory, but only the owner of these files can modify / delete them. Example: `/tmp`

4. Basic concepts

4.1 Sessions

To work with UNIX, one has to start a *session*. This is done by logging in to the system with a login and password. Once this is done, a command shell is started and the user can start to work. To close the session, use the commands "exit" or "logout".

During the time of the session, the user is logged in and is visible to other users (`who` and `w` commands). Other users can send messages or talk with the user (`write` and `talk`).

To see which processes are running, one can use the `ps` and `top` command (`top` is not always available).

It is possible for one user to be logged in several times on the same system, for example using multiple virtual terminals. The user will appear as many time as he is logged in.

4.2 The shell and environment

The shell (also known as "command-line interpreter" or CLI) is a special application which interacts with the user and the system, thus creating an interface between the two.

The shell takes care of executing processes, managing the terminal, and interacting with users.

As long as the user is logged in, at least one shell is running.

To know what the current shell, use the command `echo $SHELL`.

$SHELL is an environment variable. The list of environment variables can be listed with `env` or `printenv` (bash, sh, zsh) or `setenv` (csh, tcsh).

Environment variables control the behavior of programs and shells.

4.3 Terminals

There are two kinds of terminals under UNIX:

- Physical terminals connected with serial lines: `/dev/ttyd*` or `/dev/ttyS*`
- Virtual terminals associated to network connections or virtual consoles : `/dev/ttyp*` or `/dev/ttyv*`

To know the type of the current terminal, use the `tty` command.

4.4 The manuals

Every UNIX system has online a complete set of documentation available with the command `man`. Type `man man` for more help on man :-) There are 9 manual sections:

1. general user commands and utilities
2. system calls
3. C functions and libraries
4. devices and drivers
5. file formats
6. games
7. misc.
8. system maintenance and operation commands
9. kernel interfaces

The standard notation is `command(sec#)`, as in `ls(1)`, which means "command ls, in section 1".

4.5 Users and Groups

To identify the users on a system, UNIX a combination of users and groups.

User are known by their *user id* or *uid*, which is a unique number < 65534, recorded in the password file (`/etc/passwd`). The group id, or *gid* allows user to belong to different groups and thus share resources/files with other members.

Only the super-user can add or remove users to/from a group.

4.6 Processes, signals and job control

Everytime a new application, shell or daemon is started, it gets a *PID* or process id. The list of running processes on a UNIX system can be obtained with the command `ps`. For example, to get the list of all running processes on the system, one can use `ps -ax` or `ps -e`.

Each of these running processes can be controlled with *signals*:

- To get the list of available signals on a system, see `signal(3)`.
- To send a signal to a command from the shell, use the `kill(1)` command.

Processes running on the user's terminal can be *controlled*:

- to suspend the process temporaroly, use the sequence CTRL-Z
- to stop a running process, use the sequence CTRL-C
- processes in the *running* in background can be brought to the "foreground" with the command `fg`.
- a suspended process can be told to continue in the background with the command `bg`.

Several processes can be running in the background at the same time. They can be stopped or brought forward individually. To get the list of currently running jobs on the present terminal, use the `jobs` command.

Usually, when a shell is terminated, all the jobs (children) that were started by it and are still running are also terminated.

5. Filesystems

Some filesystems:

- UFS, FFS, ext2fs : original UNIX disk-based filesystem
- NFS, AFS, CODA : network-filesystems
- CD9660 : CD-ROM ISO-9660
- UnionFS, NullFS, Portal : special filesystems

The original FFS was limited to 14 characters per filenane. UFS and ext2fs moved the limit to 255. Also, many speed improvements.

A typical file:

```
-rw-r--r--  1 regnauld  staff     6153 22 Oct 11:36 file.txt
```

5.1 Directory / file hierarchy relations

Example:

A directory "files" contains the file "text", with the following rights:

```
drwxr-xr-x  1 root  bin      4 21 May 10:43 files/
     |
     +-->  -rwxr-xr-x  1 bob   users  240 26 Jun 03:47 text
```

User "bob" can *modify* the file "text" in the "files" directory, but he *can not* delete it: file creation and deletion are submitted to the access right of the directory *the file is created / deleted in*.

6. Links

Two types of links:

- hard links (`ln`)
- symbolic links (`ln -s`)

Hard links are additional directory entries (with a different name) pointing to the *same* file. Since a directory table is local to a partition, hard links can only be made in the same partition (i.e.: it is not possible to do `ln /usr/local/bin/bash /bin/bash` if `/usr/local` and `/` are two different partitions. *Hard links cannot be made on directories*.

Symbolic links are files in the filesystem which point to another file or directory anywhere in the filesystem.

CONTENTS