Operating Systems Lab (C+Unix)

Enrico Bini

University of Turin

Outline

1. Unix: introduction and basic usage
    - Shell
    - File system
    - Accounts

Operating System (OS)

- An operating system is the software interface between the user and the hardware of a system.
- We say that the operating system manages the available **resources**.
  - Whether your operating system is Unix-like (Linux), Android, Windows, or iOS, everything you do as a user or programmer interacts with the hardware in some way.
- the components that make up a Unix-like operating system are
  1. device drivers: make the hardware work properly (coded in C and assembly),
  2. the kernel: CPU scheduling, memory management, etc. (coded in C)
  3. the shell: allows the interaction with OS
  4. the file system: organizes all data present in the system
  5. applications: used by the user (coded in fancy languages: Java, python, or else)

Outline

1. Unix: introduction and basic usage
   - Shell
   - File system
   - Accounts

Shell

- shell (Italiano = "guscio"), versus kernel (Italiano = "nucleo")
- The shell is a command line interpreter that enables the user to access to the services offered by the kernel
- The shell is used almost exclusively via the command line, a text-based mechanism by which the user interacts with the system.
- Terminals (the "black window". Icon: >_) allows the user to enter shell commands
- When entering commands in a terminal, the button "TAB" helps to complete
- The real hacker uses the terminal only. The mouse and the graphic interfaces are for kids: is it more efficient to use 10 fingers over a keyboard? Or one finger over a strange device?
- Exercise: open a terminal and try

  `cat /etc/shells` (cat shows the content of a file)

  `echo $SHELL` (`SHELL` is an environment variable)

System calls

- system calls ("syscalls" for short) are the "access point" to the kernel: the way programs ask the kernel for any service
- Example of services asked to the kernel:
  - reading a file from the disk,
  - reading the keyboard,
  - printing over the screen,
  - reading from the network card
  - ...
- syscalls are identified by a unique number
- `strace <command>` shows all system calls happening when invoking `<command>`. Example:

  ```
  echo ciao
  strace echo ciao
  ```
- `strace -wC <command>` also shows a summary of the invoked system calls

Help on commands

- Unix manual pages (or `man` pages) are the best way to learn about any given command
- man pages are invoked by "`man <command>`"
  - Space to scroll down, `b` to scroll up, `q` to quit
- `man` pages are divided in sections

Sec.	Description
1	General commands
2	System calls
3	Library functions, covering in particular the C standard library
4	Special files (usually devices, those found in `/dev`) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons

- if same entry in more section, it is returned lower section
- try: `man printf`, `man 1 printf`, `man 3 printf`

Outline

1. Unix: introduction and basic usage
   - Shell
   - File system
   - Accounts

File system

- The file system enables the user to view, organize, store, and interact with all data available on the system
- Files have names: file extension does not imply anything about the content, it is just part of the name
- **Files** are arranged in a tree structure
- **Directories** are special files which may contain other files
- The root of the tree is “/”
- The **full pathname** of a file is the list of all directories from the root “/” until the directory of the file
- “.” is the current directory
- “. .” is the parent directory
- “~” is the home directory of the user
- Files may be **links** to other files: command `ln` to create links

File types

- Files are an abstraction of anything that can be viewed as a sequence of bytes: the disk is a (special) file
- More in general, there are 7 types of files:
  1. (marked by "-" in `ls -l`) regular file: contains data, are on disk
  2. (marked by "d" in `ls -l`) directories: contains names of other files
  3. (marked by "c" in `ls -l`) character special file: used to read/write devices byte by byte (`stat /dev/urandom`)
  4. (marked by "b" in `ls -l`) block special file: used to read/write to devices in block (disks). Try `stat /dev/nvme0n1`
  5. (marked by "p" in `ls -l`) FIFO: a special file used for interprocess communication (IPC)
  6. (marked by "s" in `ls -l`) socket: used for network communication
  7. (marked by "l" in `ls -l`) symbolic link: it just points to another file
- try `stat <some-file>`, `stat /dev/nvme0n1` to view status and type of any file
- the disk is a file: `cat /dev/nvme0n1` to show its content

Directory content

`/bin`	common programs, executables (often subdirectory of `/usr` or `/usr/local`)
`/boot`	The startup files and the kernel
`/etc`	contains configuration files
`/home`	parent of home directory of common users
`/tmp`	place for temporary files, writable by everybody, cleaned upon reboot
`/root`	home directory of the administrator
`/lib`	library files
`/proc`	information on processes and resources (only on some Unix-like machines)
`/dev`	contains references to special files (disks, terminals, etc.)

- The content of directories follows the "Filesystem Hierarchy Standard"
  `https://refspecs.linuxbase.org/fhs.shtml`
  so that programmers can expect to find something in the right place

Common commands

Most important key is TAB: it helps auto-complete

`cd`	Change directory: moves you to the directory identified
`cat`	Concatenate: displays a file
`cp`	Copy: copies one file/directory to specified location
`du`	Disk usage
`echo`	Display a line of text
`grep`	Print lines matching a pattern
`ls`	List: shows the contents of the directory specified
`mkdir`	Make directory: creates the specified directory
`more`	Browses through a file (has an advanced version: `less`)
`mv`	Move: moves the location of or renames a file/directory
`pwd`	shows the current directory the user is in
`rm`	Remove: removes a file
`sort`	Sort lines of text
`tail`	Shows the end of a file
`touch`	Creates a blank file or modifies an existing file's attributes

Input/Output redirection

- To work properly, every command uses a source of input and a destination for output. Unless specified differently
  - the input is read from the keyboard
  - the output is written to the terminal
- Unix allows the **redirection** of the input, output, or both
  - redirection of the input from a file (with “<”)
  - redirection of the output to a file (with “>”)
  - redirection of the output of command A as input to command B (“pipe” with “|”)
- Examples:
  - `ls > my_list`
  - `wc < my_list`
  - `ls -latr | less`
  - `du -a | sort -n`

Metacharacters

- **wildcards** are special characters that can be used to match multiple files at the same time
  - `?` matches any one character
  - `*` matches any character or characters in a filename
  - `[ ]` matches one of the characters included inside the `[ ]` symbols.
- Examples
  - `ls *.tex`
  - `ls *.[tl]*`
  - `ls *t*`
  - `ls ?t*`

Outline

1. Unix: introduction and basic usage
   - Shell
   - File system
   - Accounts

Accounts

- Unix is a multi-user systems: more than one user can use "simultaneously" the available resources (computing capacity, memory, etc.)
  - Once upon a time there were single-user operating systems such as MS-DOS
  - In applications where the resources must be used by a single application, multi-user is not needed (example: embedded systems)
- **accounts** are used to distinguish between different type of usage of resources
- There are three primary types of accounts on a Unix system:
  - the root user (or superuser) account,
  - system accounts, and
  - user accounts.

All accounts

- `cat /etc/passwd` to see all accounts. Seven colon-separated “:” fields:
  1. login name
  2. crypted password (today passwords are in `/etc/shadow`, accessible only with `root` privileges)
  3. numeric user ID
  4. numeric group ID
  5. a comment field (used to store the name of the user or the name of the service associated a system account)
  6. the home directory of the account
  7. the default shell
- Command `usermod [OPTIONS] <username>` to change any among the fields above and more
- `usermod -c "New Name" bini` to change the comment field into “`New Name`”

Root accounts

- The root account's user has complete control of the system: he can run commands to completely destroy the software system as well as some hardware component
- The root user (also called root) can do absolutely anything on the system, with no restrictions on files that can be accessed, removed, and modified.
- The Unix methodology assumes that root users know what they want to do, so if they issue a command that will completely destroy the system, Unix allows it.
- People generally use root for only the most important tasks, and then use it only for the time required and very cautiously.

"With great power comes great responsibility"

- command `sudo` allows running a command as another user (even `root` if allowed). Example: packages are installed by

  ```
  sudo apt install <package-name>
  ```
- command `su` allows becoming another user (even `root` if allowed)

System accounts

- System accounts are specialized accounts dedicated to specific functions

  ```
  cat /etc/passwd
  ```

  - the "`mail`" account is used to manage email
  - the "`sshd`" account handles the SSH server
  - web servers run as dedicated account
  - ...
- they assist in running services or programs that the users require
- they are needed because often running some services (mail, SSH, ...) requires **some** root privilege. Hence:
  - running these services with user privilege is not possible
  - running these services with root privileges is too risky
  - that's why system accounts are useful
- main access to hackers: accessible to user, but with some root privileges
- services running with system accounts **must** be super safe!

User accounts

- user accounts are needed to allow users to run applications system resources and are “protected” by passwords

User accounts

- user accounts are needed to allow users to run applications system resources and are “protected” by passwords
- most common passwords

  123456 qwerty password 987654321 mynoob
  666666 18atcskd2w 1q2w3e4r zaq1zaq1 zxcvbn
- Some users may be fully trusted and the OS would like to give them the possibility to do anything
- Some others may be authorized to do only a subset of the possible actions
- How are privileges managed?

Groups

- users with similar privileges are assigned to the same **group**
- the administrator (`root`) can then manage all the users belonging to the group by simply assigning privileges to the group
- an account may belong to more than one group, if needed
- `cat /etc/group` to view the list of group. Each row has:
  1. group name
  2. group password (very rarely used. From `man gpasswd`: “Group passwords are an inherent security problem since more than one person is permitted to know the password.”)
  3. group ID
  4. list of users belonging to the group
- Example: `cat /etc/group | grep sudo` shows all users belonging to the sudo group (who can launch `sudo <command>`)
- `groups bini` shows the groups a user belongs to

File ownership, permission

- Each “file” (which may be the disk and the terminal and other strange things) has
  - an `owner` and
  - a `group`
- Permissions are divided in three subsets:
  - `u` permissions of the user (owner)
  - `g` permissions of the users in the group
  - `o` permissions to all others
- Permissions are of three types:
  - **read** (`r`) if the file can be read
  - **write** (`w`) if the file can be written
  - **execute** (`x`) if the file can be executes (“search” permission id directory)
- `chown` to change the owner of a file
- `chgrp` to change the group of a file
- `chmod` to change the permissions of a file
- Example: `chmod u+rw <filename>` adds read/write for the owner
- Example: `chmod o-r <filename>` remove write for the others

File permission, octal representation

- File permissions are often represented in octal (base 8)

user			group			other			octal
r	w	x	r	w	x	r	w	x	
1	1	1	1	1	0	1	0	0	=764

- Equivalent commands
  - `chmod u=rwx,g=rw,o=r <filename>`
  - `chmod 764 <filename>`
- Examples:
  - `ls -l` to view permission (try it is `/dev/`)
  - `chmod` to change permissions of a file
  - `chown` to change owner and group of a file