Basic Tools for System Management

Frederic.Mallet@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR

Objectives

- Learn how to manage your own computer
  - Using the shell
    - Linux / Windows
  - Version control and git
  - Installing a Virtual Machine
    - Linux Mint 19
  - Using bash

System

- User
  - Access rights: identify the actors that can use the system, what they can do, and what the system can do for them
- File
  - Data, programs, configuration parameters: Store the data, the programs that use the data, and the parameters that determine how the system and its services behave
- Process
  - Dynamic view: running programs, system services in execution and daemons doing things in the background for the users.

Root

- Virtual user that can do everything
  - He can decide what all the users can or cannot do
  - He can run process for all the users
  - You can often run programs as root
- Root files
  - Own system files and parameter files
- Virtual user
  - You can assume the role of root at some points

Responsibility

- Users are responsible for their files and processes
  - A file always belongs to one system user
  - A process always act on behalf of one system user
- Root
  - Can do any thing on any file and any process
- Non-Root
  - Can only interact with files or processes if the user (or root) has allowed it
  - Process have the same rights are their owner, they act on his behalf

Initialization

- Files and processes
  - “Executable” files are loaded in the memory before running as processes
  - Processes can modify/create files or launch other processes
- Operating System start
  - Launch the “init” process, that populates the memory (including login window)
  - “init” is owned by root

Users

- Users are uniquely Identified by an integer (UID)
  - Login + password allows for identification and gives the access rights associated with a UID
    - **echo $UID**
  - Root has UID 0
- Groups
  - Users are structured into groups and each user belongs to at least one group
  - Groups are identified by their unique group identifier (GID)
    - **newgrp**: join a group (if allowed by /etc/group)
    - **groups**: give the list of groups in which the user belongs
    - **id**: displays the user id, group(s) id(s)

Users

- Group Root (GID=0)
  - User root belongs to group root (GID=0)
  - Other virtual users can also belong to group root
    - They get the permissions of group root (different from the ones of user root)
    - Used to manage the system without being root (e.g., print spooler)
- Home
  - Each user has a space on the disk that is reserved for his login
    - **echo $HOME**: display your home directory
      - e.g. /home/fmallet
  - You need to decide what access rights you give to that directory: who is allowed to access

/etc/passwd

- user’s Init process
  - Each user has a init process
    - Usually the ‘shell’ that is called after login
  - /etc/passwd:
    - Login, User id, Group id, encrypted passwd, real name, home directory, init process
  - /etc/shadow:
    - Contains the encrypted passwords when connected

Virtual Users

- Not related to a specific person
  - Examples: root, ntp (net time protocol: update system hours), smtp (send mail transfer protocol to send emails), lpd or cups (line printer or common unix printing system, to handle printers), nfs (network file system, to deal with file systems through networks)
- May also use a Network Information Service
  - One computer in the network contains the encrypted password
  - May copy locally on the first login of a user

Files

- Persistent Memory
  - Contain persistent information about data or programs of the system
  - Physically stored into a hard-drive of high capacity with passive memory: hard drive, flash memory, usb drive, CDROM
  - Not like temporary memory storage: RAM, cache

File hierarchy

- Files are identified by their i-node number, the system does not use their name to locate it
- The File system handles a tree which root is ‘/’ and is owned by root
- A directory is a link to the files it “contains”, its children. These links have names (not the file)
- Files are (not uniquely) identified by an **absolute path** from ‘/’ to itself
  - files can have many names
  - Paths are filenames separated by ‘/’, absolute paths start from the root ‘/’, relative paths may start from any node
    - E.g., **/home/fmallet**, **/etc/passwd** …

File hierarchy

- When ‘moving’ a file, only the file system tree changes, the file keeps its i-node and does not physically move on the drive
  - This is not true if the physical support has to change: moving from the hard drive to a usb stick
  - **mv oldpath newpath**
- A directory contains at least two links
  - ‘.’ pointing to itself
  - ‘..’ denotes its (relative) parent
- Each process has a working directory
  - It can access files using **absolute paths** (i.e., starting with ‘/’) or **relative paths** (from the working directory)

What do you think happens to ‘/’ ?

Working directory

- The shell using displays its working directory
  - By default, this is the user home directory
  - When a process launches another one, it transmits its own working directory
    - **pwd**: print working directory
    - **cd path**: change directory
    - **ls [paths]**: list the files
- Most commands have options (starting with -)
  - Try the **-a** option on **ls**
  - How do you find the options of a command ?

File types

- Files: store texts, images, music, videos, may contain data or programs
- Directories: folders that “contains” other files => no physical containment, only pointers to a list of files
- Symbolic links: point to one other files (may have a different name)
  - **ln -s path link_name**
- Special files: “gates” to and from other devices: printers, distance devices
- Command file
  - **file /etc/passwd** ASCII text
  - **file /home** directory
  - **file /dev/stdin** symbolic link
  - **file /dev/tty** character special

Access rights

What does it mean for directories ? For symbolic links ?

- All Files can be:
  - read: ls, read the content
  - written: modify the content
  - Executed: execute (the process belongs to the user who calls it, unless sticky bit !)
- Access rights are different depending on whether you are
  - The user (owner of the file)
  - The group of the file (not necessarily the group of the owner)
  - Another user: neither owner nor group
- Modify rights: if you are owner or root (check what **ls -l** does)
  - **chmod 644 path** or **chmod [-R] ugoa[+-=]rwxs path**
  - **chown login[.group] path** (only root can do that)

user			group			other		
r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1

Distribution

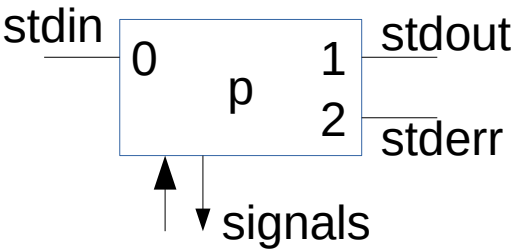
- Propose an default organization for system files and basic default programs
  - text editors, file system managers...
  - E.g., BSD, Mageia, Fedora, RedHat, Debian, Ubuntu…
- Standard commands
  - cd, ls [-al], rm [-rf]
  - mkdir, rmdir
  - cp, mv
  - man

Processes

- A process is identified by its PID, while running
  - init has number 1
- A process can start other “child” processes
  - Hierarchy of processes, the root of which is ‘init’
  - Shells may have a lot of child processes
- A process acts on behalf of a user with its access rights: be careful, especially when you are root
  - The only way to be sure is to read the source code when “open source” (does not mean free)
- Killing processes
  - **ps -aux** list all executing processes
  - **kill -9 process_number** kill a process

Standard streams

- All process can access to at least three standard streams (including shell)
  - /dev/stdin: standard input (usually keyboard)
  - /dev/stdout: standard output (usually terminal)
  - /dev/stderr: standard error (usually terminal)



- You can redirect them with < > >> 2> 2>>
  - **command < input.txt > output.txt 2> err.txt**
  - Using >> appends at the end of the file rather than creating a new one and overwriting it
- You can ‘pipe’ the commands using | ; || &&
  - **command1 | command2**
    - This redirects the standard output of command1 to the standard input of command2
  - **command1 ; command2** sequence
  - **command1 || command2** command2 only if command1 fails
  - **command1 && command2** command2 only if command1 succeeds

Controlling processes

- You can list the processes using **ps**
  - Check out the option **‘-aux’**
  - Pipe it with **less** or **more** or **head**
- You can check for resource consuming processes with **top**
- You can kill a process smoothly with **kill**
  - Child processes are attached to the parent processes
  - Or Violently using -9: it makes zombie processes

backup

- Needs to save/restore work
  - Archive old work
  - Retrieve work in case of loss or destruction or mishandling of the machine
- Using **rsync**
  - **sudo apt install rsync**
  - **rsync [options] source destination**
    - Can be relative or absolute local path
    - Can be remote path: login@srv-kairos.inria.fr:/users/… or cloud drive
    - Can be a local drive or usb drive: /media/login/...
    - Beware path and path/ have different semantics !

Lab

- Download virtualbox (http://www.virtualbox.org)
- Download linux mint ISO 19.2 (https://linuxmint.com/)
- Check the checksum of both files before using them
  - PowerShell (Windows)
    - Get-Content path
    - Foreach { echo $_ }
    - Test-Path path
    - CertUtil

    SubString
    Split
    equals
    -replace xx,yy

    if (…) { … }
    echo $x
    $t[n]
  - Bash (Linux)
    - sha256sum -b *.iso
    - sed