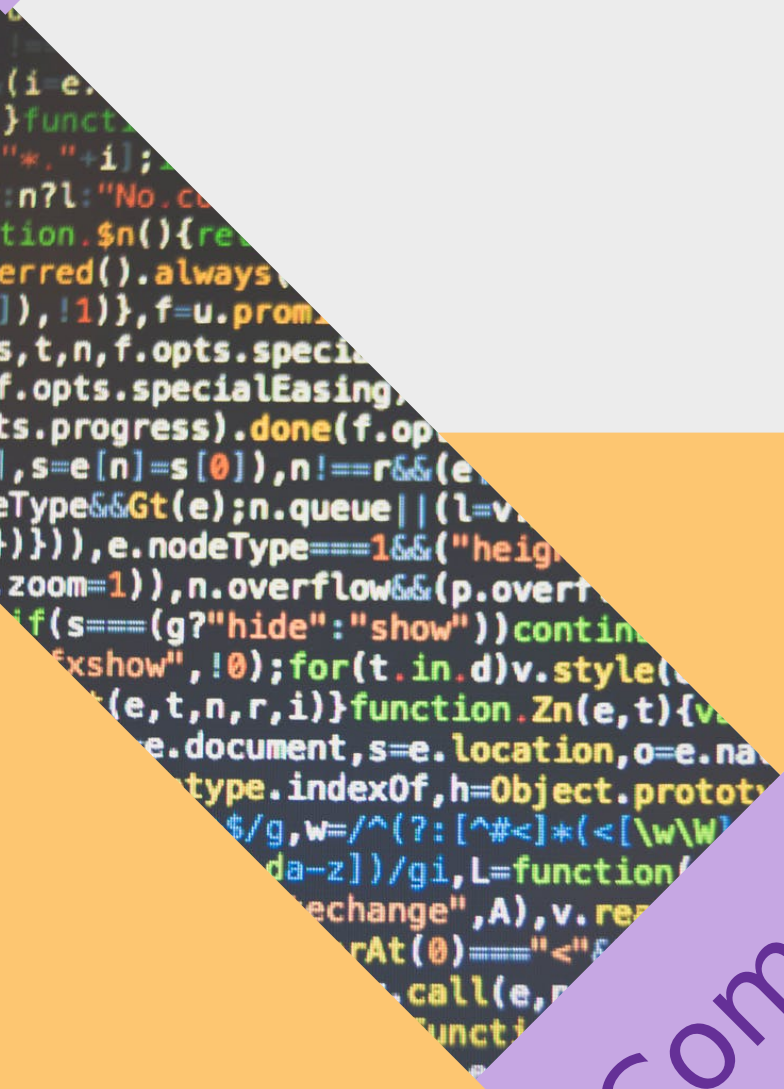
Curs 2020-2021



Computer Systems Processes Admin.

IES Jaume II El Just
Tavernes de la Valldigna

Contents

1 Processes **2**
1.1 Processes management 2

2 Managing services **7**

1 Processes

It is common knowledge that a process is a program which uses the CPU, or it is loaded in main memory. The difference between process and programs is that each time that the program is run, a new and different process is created. Moreover, different processes of the same program can be created at the same time. Besides, the context of each process is different. It depends on the user, who runs the process, permissions, (and) whether it has been called from another process, parent process, etc...

Processes operate in the same way that users, groups or files, are assigned by a number. That number is called Process ID (PID), it identifies each process and it is different for each process as well.

1.1 Processes management

To manage processes, the following commands can be used:

- **ps** ⇒ it informs about the process state. It has different options in order to display different columns. The following columns are the most featured:
  - USER ⇒ user with whom the process is run
  - PID ⇒ process identifier
  - PPID ⇒ Parent process identifier
  - UID ⇒ identifier of the owner of the process
  - TTY ⇒information about what terminal is running the process.
  - TIME ⇒accumulated CPU time
  - CMD ⇒ name of the program which starts the process
  - RSS⇒ (resident set size) Size of the resident part of the process. It is the amount of the main memory that the process has used (KiB)
  - NI ⇒ process priority.
  - %CPU⇒ percentage of the used CPU
  - STAT ⇒ process state. The states of the process can be:
    - R ⇒ running process or in ready queu
    - S ⇒ Blocked (waiting for other events taking place within the process.)
    - T ⇒ stopped, either by a job control
    - D ⇒ waiting for other events, some input/output task
    - Z ⇒ zombie, finished process but it needs to report some information to the parent process.

  Syntax:

  ```
  ps [options]
  ```

  Options:

  - -A ⇒ it displays the information columns: PID, TTY, TIME,CMD
  - -e ⇒ it displays information about all the process of the system (the same that -A)

- -l ⇒ it displays a lot of information about the processes (the result is different depending on the sign "-" before the option)
- -u pepe ⇒ it displays the information of the user pepe.
- -ely ⇒ that options are used together in order to display all the information as PID, PPID , TIME, CMD, NI, RSS, ...
- aux ⇒ (without sign "-") it displays the columns of USER, PID, RSS, TTY, STAT, TIME,

- **pstree** ⇒ shows the running processes as a tree.

  Syntax:

  ```
  pstree [options]
  ```

  Options:

  - -u ⇒it shows the process owner

Example

1. Show the information of the all the processes which are running.
2. Display the PID of some of the executing process
3. Show all the processes as a tree and show the owners. Page the output

Solution:

1. ps -aux
2. ps -A | grep -i process_name
3. pstree -Gu | more

- **nohup,&** ⇒ It is used for executing processes in the background. Whereby, terminal is available to run commands while the process in the background is running.

  Syntax:

  ```
  nohup command &
  command &
  ```

Sleep Command

- sleep ⇒ that command is used to delay a specified period of time. The unit by default is in seconds but it is possible to use it with minutes, hours and even days (m, h, d).

Example:
Sleep 5 ⇒ 5 seconds will be delayed.
Sleep 5m ⇒ 5 minutes will be delayed or temporized.

Example

1. Execute sleep command in order to temporize 10 seconds and try to run some other command as ls –l. Look at that the second command, it won't run after ten seconds.

2. Execute sleep command to delay it by 10 seconds again. But this time use the command in order to send the command at background (nohup or &).

   Solution:

1. sleep 10

2. sleep 10 & or
   nohup sleep 10 &

   Now, the prompt is available for executing new commands and it is possible to execute ls –l or some other command due to sleep command is running in the background.

- **renice** ⇒ that command is used for modifying the priority of a running process

  Syntax:

  ```
  renice [options] priority PID_process
  ```

  Options:

  – -p ⇒ it is used in case you want to apply the command to more than one process.
  – -u ⇒ it is used in case the objective is to apply the command to all the processes of some user.

Example

1. The desired objective for an user is to execute a scientific operation in a process, but while that process is executed, the user wants to run a game. Therefore, the user will want to change the priority of the scientific process since the priority is now running the game. If the scientific process is using the PID 785:

   renice +15 785
   where the most positive priority is the lower priority

2. Being the user administrator of the system, it becomes apparent that a user is running a lot of processes. So, it is possible to change all the processes of that user:

   renice +19 -u user_name

- **nice** ⇒ That command is used for establishing the priority of the process. The priority available for the process can be from -20 (max priority) to 19 (minimum priority). The priority can be seen at NI column (ps command). Later, the real priority is calculated, and it could be seen in the PRI column (ps command).

  Syntax:

  ```
  nice [options] command
  ```

Options:

- -n number ⇒ the command will be executed with the indicated priority.

Example

If the user wants to burn Ubuntu iso in the cd-Rom, but at the same time wants to execute Libreoffice in order to finish some task, the user can execute the command for making the Ubuntu iso with the established priority, since making an iso is a very demanding process in terms of used resources.

```
nice -n 19 dd if=/dev/cdrom of=~/ubuntu-19.10-desktop-amd64.iso
```

- **jobs** ⇒ it displays the status of jobs started in the current terminal window. Jobs are numbered starting from 1 for each session. The job ID numbers are used by some programs instead of PIDs. To suspend some process, (keys CTRL+Z must be pressed)

  Syntax:

  ```
  jobs
  ```

Example:

- **fg** ⇒ that command is used to send to foreground the process which are in the background or stopped.

  Syntax:

  ```
  fg %n
  ```

  where n is the number in the list showed by jobs command. If any process is specified, the process with the mark + in the jobs command list it is sent to foreground.

- **bg** ⇒ This command is used to restart a stopped background process.

  Syntax:

  ```
  bg %n
  ```

  where n is the number in the list showed by the jobs command.

- **kill** ⇒ to kill the indicated process. It is used for sending some other signals to the process.

  Syntax:

  ```
  kill [options] PID
  ```

  Options:

  - -9 ⇒ -SIGKILL⇒ kill the process whose PID is sent as an argument.
  - -l ⇒ show all the signals which can be sent to the process.

Try it

1. Execute sleep command with 60 seconds in background
2. Displays all the processes in background
3. Make sure that sleep command is running
4. Stop the sleep process with the command kill (check with kill –l what would it be the proper signal)
5. Displays all the processes in background
6. Pay attention that sleep command is stopped
7. Terminate the sleep process
8. Displays all the processes in background

- **killall** ⇒ kill all the processes related with the indicated process. This time, the name of the process should be specified instead of the process number.

  Syntax:

  ```
  killall [options] process_name
  ```

  Options:

  – -kill ⇒ kill the process.
  – -i ⇒ ask whether the process is really wanted to be killed.
  – -l ⇒ show all the signals which can be sent to the process.

- **time** ⇒ it is run with a command in order to know how long it is spent for running the command, used resources, etc...

  Syntax:

  ```
  time command
  ```

- **top** ⇒ It displays information about the system and the executed processes. The information is updated each 3 seconds.

  Syntax:

  ```
  top [options]
  ```

  Options:

  – -d ⇒ in order to change the delay (in seconds) between top's updates.

Example

1. Execute nano file4.txt.
2. Stop it and send it to background.
3. Check it out.
4. Execute nano file5.txt in background.
5. Send the first process to foreground. That means the process is running again.
6. Close nano
7. Kill the second process.
   Solution:

1. nano fichero4.txt
2. CTRL+Z
3. jobs
4. nano fichero5.txt &
5. jobs
   fg %1
6. CTRL+X
7. kill %2

2 Managing services

The services, also called daemons, are programs which are executed in background. They try to offer some functions without user awareness. They are services to check the updates, to manage the system clock along with the network connection, and even to warn whether something doesn't work properly. Some system services are:

- Crond ⇒ execute the programmed task of the system
- Network ⇒ run the network interfaces of the system.
- Rsyslog ⇒ save the log of the system.

But there are others which are not related to the performance of the system. However, they do offer some functionalities:

- Dhcpd ⇒ DHCP server
- Smb ⇒ sharing some resources with Windows.
- Squid ⇒ proxy server.

In order to manage the service, Ubuntu uses Systemd. Basically, it is a mechanism of managing services which interacts with the kernel.

To control the administrated services by Systemd, systemctl command is used.

Normally, when command systemctl is used, the service name is added to the suffix ".service".

systemd-networkd

systemd-networkd is a system daemon that manages network configurations. It detects and configures network devices as they appear; it can also create virtual network devices.

With that command, it is possible to do the following operations about services:

- Check the service state :

  systemctl status systemd-networkd.service

  In this example, check the state of the network configuration.

```
exam@exam-VirtualBox:~$ systemctl status systemd-networkd.service
● systemd-networkd.service - Network Service
   Loaded: loaded (/lib/systemd/system/systemd-networkd.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:systemd-networkd.service(8)
```

- It is possible to check a specified state:
  - systemctl is-active: if the service is running.
  - systemctl is-enabled: it lets us know if the service is enabled.
  - sytemctl is-failed: it informs whether some problem was happening while starting the service.
- To start and stop services some privileges are needed to be acquired:
  - sudo systemctl start systemd-networkd.service
  - sudo systemctl stop systemd-networkd.service
- To enable or disable services when the system is booted, it is possible to use options:
  - enable: with this option the service will be enabled for booting the next time that the system is booted.
    Example:

    sudo systemctl enable systemd-networkd.service
  - disable: with this option the service will be disabled the next time that the system is booted.
    Example:

    sudo systemctl disable systemd-networkd.service
- Sometimes, when some features are configured in some services, it is necessary to restart the services in order for the new features to be applied. To do this:
  - Restart:

    sudo systemctl restart systemd-networkd.service