Process management

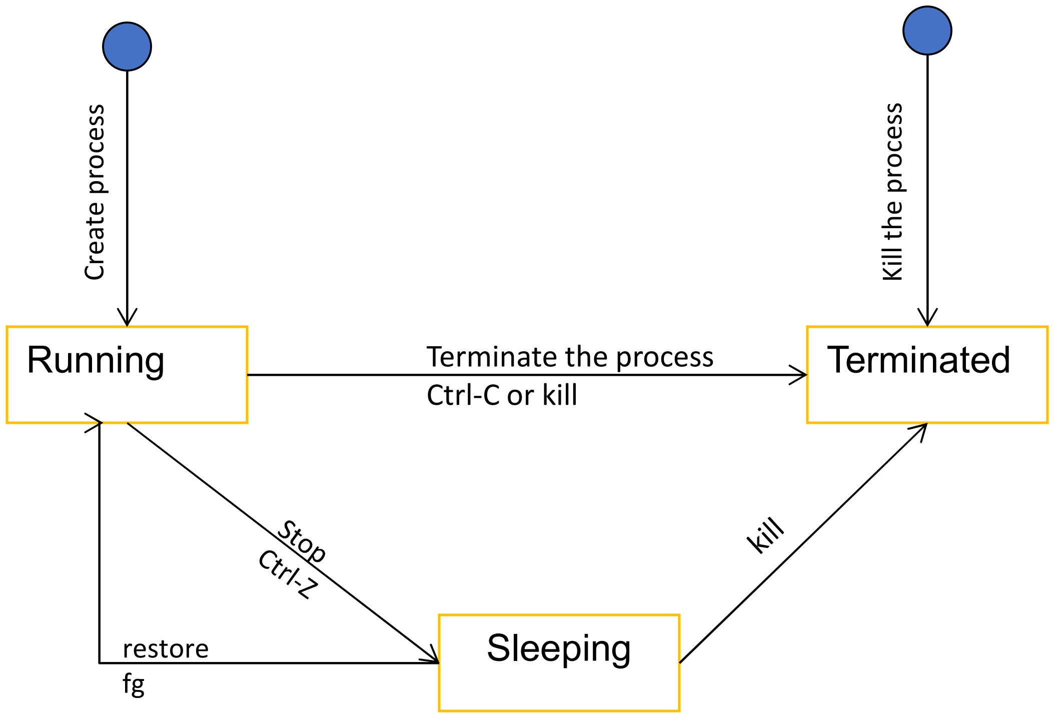
Ngoc Nguyen Tran

Introduction

- Process = a running program
- Each process has the following information:
  - Process id (pid)
  - Parent process id (ppid)
  - Owner (uid) and group (gid)
  - Command
  - Standard input (stdin), standard output (stdout), standard error (stderr)
  - CPU time and priority
  - Current working directory of the process
  - Reference table to used files
- Processes are organised to share CPU using

States of a process

- S: Sleeping
- R: Running
- T: Terminted
- Z: undefined



Process type (1)

- System process
  - Usually belongs to root
  - No interactive interface
  - Usually daemon one
  - Purpose: general tasks, providing for everyone
  - Example:
    - **lpsched**: manage printing service
    - **cron**: schedule command/ program.
    - **inetd**: manage networking service.

Process type(2)

- User process
  - Perform tasks of a specific user
    - Need to login before executing any tasks.
    - Is performed through a shell or GUI
  - Usullay being executed, managed by a terminal
  - Example:
    - cp
    - vi
    - man
    - …

Command ps

- Show the processes
  - By default, only show the process belongs to the current user of the terminal.
  - Use option aux to show all current running processes

```
$ ps
 PID TTY          TIME CMD
2803 pts/1    00:00:00 bash
2965 pts/1    00:00:00 ps
$ ps aux
USER   PID  %CPU  %MEM  VSZ   RSS  TTY   STAT START TIME COMMAND
root     1   0.1   0.1  1104  460   ?     S   15:26 0:03 init[3]
...
ttanh  951   0.0   0.3  1728  996  pts/0 S    16:09 0:00 bash
ttanh  953   0.0   1.9  6860 4916  pts/0 S    16:09 0:00 emacs
ttanh  966   0.0   0.3  2704 1000  pts/0 R    16:23 0:00 ps aux
...
```

Command kill

- Send a signal to a process (ID of the process is one of parameters).
  - By default, signal 15 will be sent (SIGTERM – terminate the process)
  - Option -9: send the signal 9 (SIGKILL – kill the process)
  - Option –l: list all available signals
- Command killall: use to kill all processes by providing the name of a command.
- Permission to terminate a process belongs to the owner of the process

Priority of a process

- All processes have a default priority of zero (**0)**
- Priority of a process ranges from -19 to +19
  - Only root (or users with root privilege) can reduce the value of process priority
  - Normal users can only increase the value of process priority (reduce the priority of a process)
- Command **nice** allows to change/modify the priority of a process in execution of a program/process.
  - $ nice [-n Value] [Command [Arguments ...]]
- Command **renice** allows to change the priority of a process **after** starting a process.

Command top

- Display and update the following information of current processes:
  - CPU usage
  - Memory usage including virtual memory
  - Other information such as PID, PR, USER, TIME,…
- $ top [–d] delay
  - Option –d allows to determine the delay time between screen updates (seconds).
- Command top also allows users to interact and manage processes (modify priority, send signals,…)

Foreground and background (1)

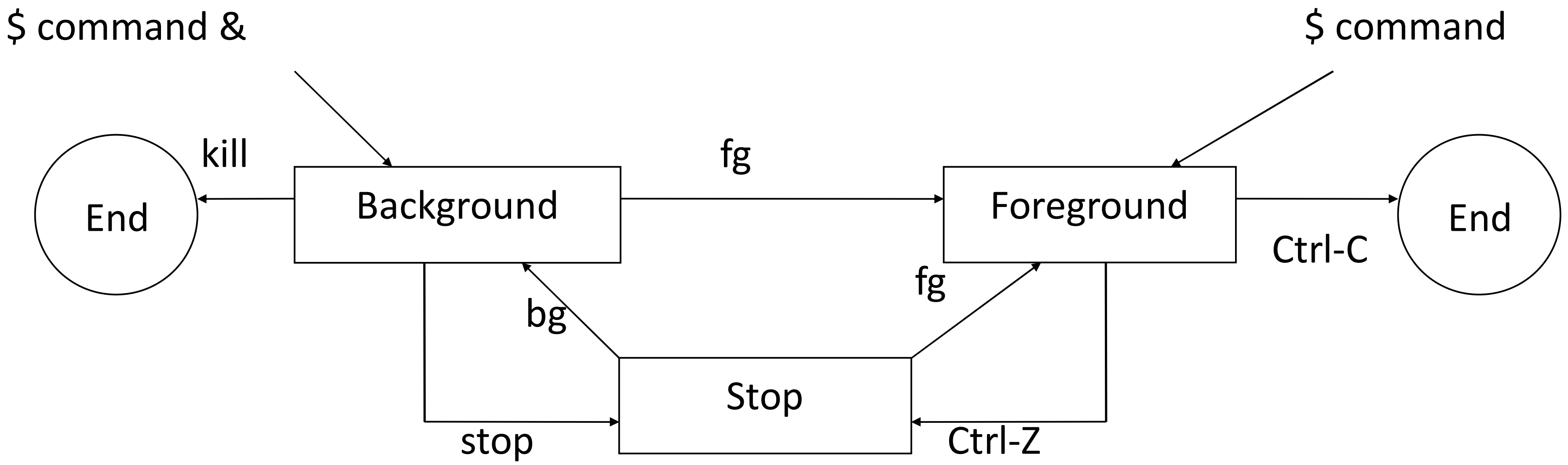
- Foreground type: a process will be started as followed:
  - « fork » is used to duplicate the parent process (it would be shell process if you enter a command)
  - « wait » is run to put the parent process to sleep state
  - « exec » is used to execute the child process.
  - After finishing the child process, a « wake-up » signal is sent to the parent process.
  - So, users cannot interact with the parent process while executing the child process.

Foreground and background (2)

- If you want to interact with the parent process while running the child process, the child process need to be run as background type.
- Example: $ emacs&
  - After entering this command, emacs will run as a background process. Users can use the terminal to enter other commands.

Manage jobs/tasks

- A job/task = exécution of a command. A job can relate to a group of process (one parent process and many child processes)
- Can not have more than 1 foreground job
- Can have multiple background tasks/jobs



Examples

```
$ emacs &
[1] 756
$ stop 756
# or $ stop %1
$ bg 756
# or $ bg %1
$ kill -9 756
# or $ kill %1
```

Run multiple commands

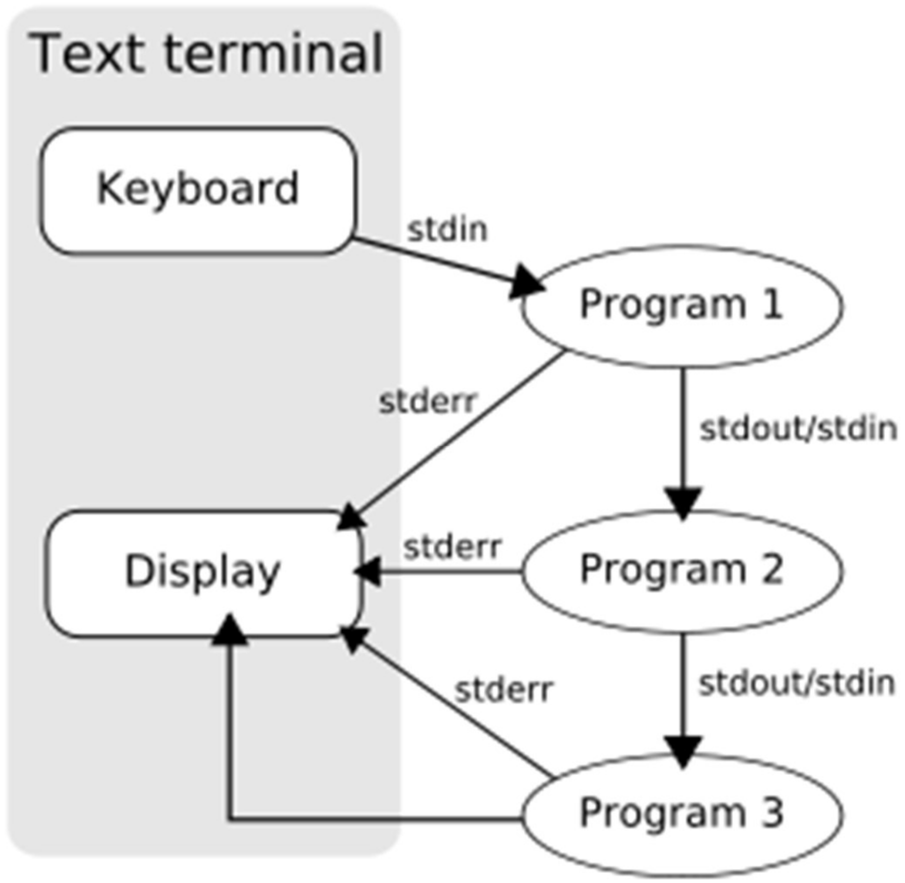
- cmd1;cmd2
- cmd1 && cmd2
- cmd1 | cmd2

Execution types

- Execute independent commands
  - Use the character ";« to execute many consecutive and independent commands.
  - `$cp public/* perso; rm -r public`
- Execute dependent commands
  - Use the character **&&** to execute many consecutive and independent commands. The next command can only be executed after the previous command is finished without any errors.
  - `$cp public/* perso && rm -r public`
  - Use the character **||** to execute many consecutive and independent commands. The next command can only be executed after the previous command is finished without any errors.
  - `$cp public/* perso || rm -r public`

Pipepline mechanism

- Pipeline allows the output of the first command becoming the input of the second one
- Pipeline can be established by using the character "|"
  - $ cmd1 | cmd2



Change the standard input/output/error

- Each process has :
  - A standard input (default one is keyboard)
  - A standard output (default one is terminal)
  - A standard error (default one is terminal)
- Change the standard input **(<)**

```
$ tee < test.txt
```

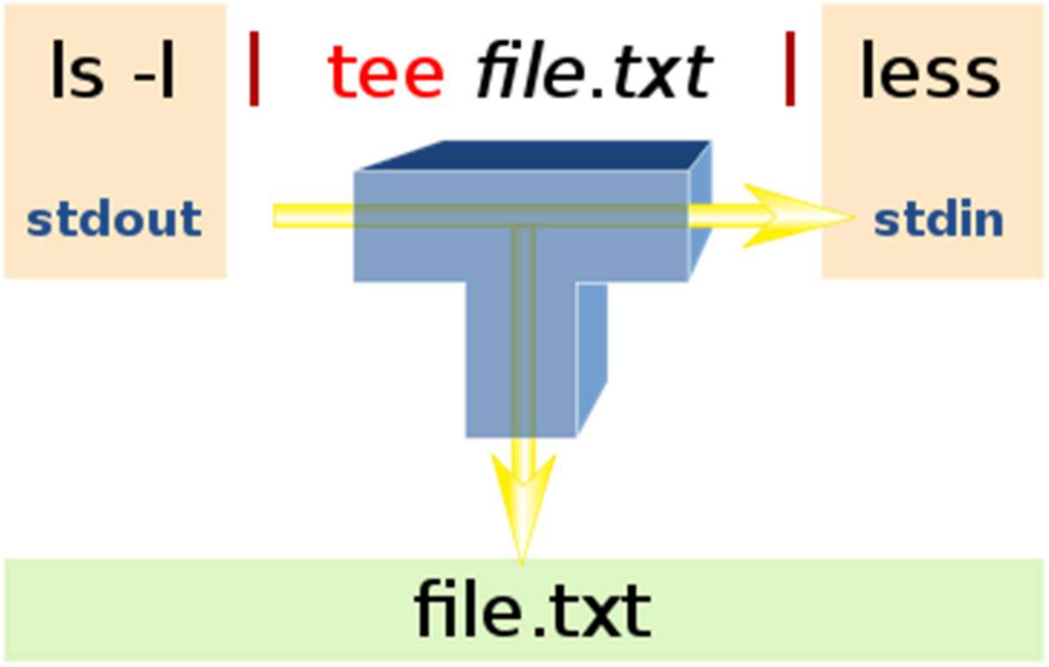
- Change the standard output **(>, >>)**

```
$ ls > /dev/lp
$ ls >> test.txt
```

- Change the standard error

```
$ rm prog.c 2> /dev/null
$ gcc prog.c 2>> erreur.txt
```

tee command



Show file contents

- $cat file_name […]
- $head -n file_name
- $tail -n file_name
- $wc file_name

grep : find within lines

$grep [-options] expreg [files]

- Find the line satisfied the conditions of expressions.

- Options
  - -c : only show the total number of lines satisfied conditions
  - -l : ony show file name
  - -v : only show unsatisfied lines
  - -i : don’t care capitalised or not
  - -n : only show line number

Some special characters

- grep can use some special characters :
  - . Represent any character
  - * Repeat the previous character
  - ^ Beginning of a line
  - $ End of a line
  - [...] list or range of finding characters
  - [^..] list or range of non-finding characters
  - Note: to avoid confusion, we should place characters inside double quotation marks.

Examples

- $grep “^t" /etc/passwd
  - Find inside the file /etc/passwd all lines beginning with “t” character.
- $grep [^t] /etc/passwd
  - Find all lines not beginning with “t” character
- $grep “tuananh" /etc/passwd
  - Find all lines containing “tuananh”
- $ls -l /etc | grep "^d"
  - Show all child directories of /etc

cut : determine the columns

$cut -options [files]

- Options
  - -c<no_of_character>
  - -f <field_number>
  - -d<splitting_character>
- Ví dụ
  - $cut -c5 file #show the fifth character
  - $cut -c5-10 file #show the character 5th to 10th
  - $cut -d: -f1 /etc/passwd #show all usernames

Change the file content

- split
  - Cut a file to many smaller files
  - Example:
    - split -10 /etc/passwd smallpasswd
- tr
  - Replace a string by another string with the same length
  - Example:
    - $cat /etc/passwd | tr ":" "#"

sort: sort the content

- $sort -options file_name
- Options
  - **-b:** skip the space at the beginning of each field
  - **-d** : only sort using characters in alphabet and numbers
  - **-r :** reverse the sorting order
  - **-f** : no distinguishing capitalised/non-capitalised
  - **-t x** : use character x as the demiliter between fields
  - **-u** : remove duplicate rows
  - **-n** sort using numbers
  - -k *x* sorting follow the field number x

Examples

- carnet.txt

```
maurice:29:0298334432:Crozon
marcel:13:0466342233:Marseille
robert:75:0144234452:Paris
yvonne:92:0133444335:Palaiseau
```

- $sort -n -t : -k2 carnet.txt
  - Thực hiện quá trình sắp xếp theo trường thứ 2

Compare two files

- $cmp file1 file2
  - Compare file1 and file2
- $diff file1 file2
  - Find the difference between file1 and file2 (text files)
  - Show results as lines

Command tar – to save and backup files/directories

Example of using ‘tar’

```
(1)# tar cvf file1.tar ./homework1
(2) # tar x file1.tar
(a)# tar cvfz backup.tar.gz file1 file2 file3

(c)# tar xvfz backup.tar.gz
```

Command gzip to compress and decompress

- tar to save the whole directory as a single file
- gzip to compress/decompress that file
- Use gzip:
  - gzip [options] [file]
  - gzip –d: decompress