# DIU EIL BlOC3

## Architectures matérielles et robotique, systèmes et réseaux

# Processus

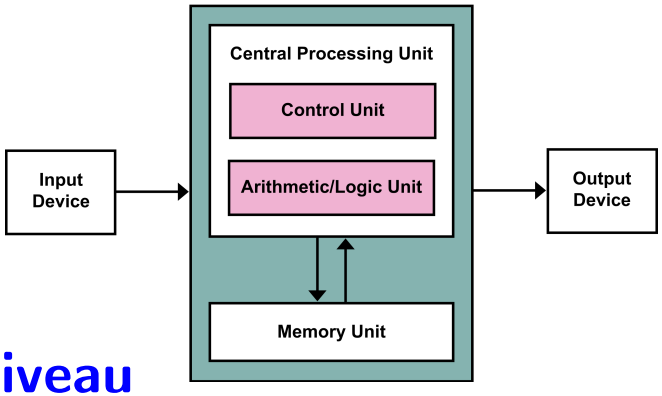
**Vania Marangozova-Martin**

**Maître de Conférences, UGA**

Vania.Marangozova-Martin@imag.fr

# Pour introduire les processus

- **Un processeur est l'unité matérielle en charge de l'exécution au sein d'un ordinateur**
  - CPU = Central Processing Unit




- **Un processeur exécute des instructions bas niveau**

En Python

```
a= b+c
```

En langage assembleur

```
MOV R0, #0x00       ; Adresse de la variable b
LDR R0, [R0]        ; Lire la variable b dans le registre R0
MOV R1, #0x01       ; Adresse de la variable c
LDR R1, [R1]        ; Lire la variable c dans le registre R1
ADD R2, R0, R1          ; R2 = R0 + R1
MOV R0, #0x02       ; Adresse de la variable c
STR R2, [R0]        ; Écrire le registre R2 dans la variable a
```

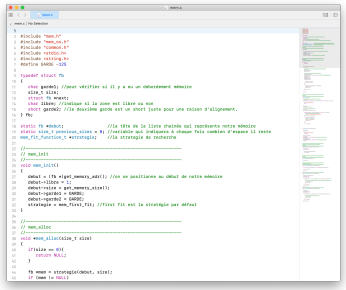
en vrai

```
00011101010101011100101000111100
10100010010100110101011011010101
10001001010110101011101110110110
...
```

- **Quelles instructions exécute-t-on sur un processeur?**
  - Les instructions des **programmes lancés** sur l'ordinateur

# Pour introduire les processus (2)

- **Un programme est une suite d'instructions**
- **Cette suite d'instructions est définie dans des fichiers**
  - Ce sont des **fichiers source** (fichiers contenant le code source)
  - Il peut y avoir un ou plusieurs fichiers source
  - Les fichiers source sont généralement écrits dans un langage haut niveau
  - Les fichiers sont des entités statiques, ce sont des données stockées sur disque
- **Comment passe-t-on des sources à l'exécution d'un programme?**



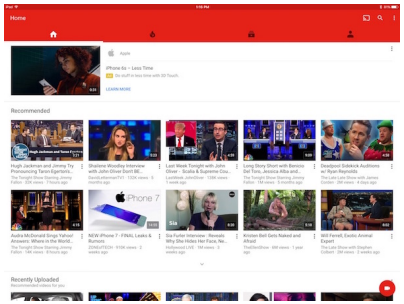
**code source**

↝

**code exécutable**

lancement

↝

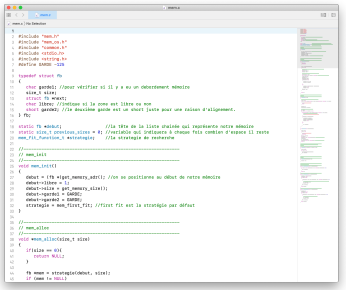


**programme en exécution = processus**

- **Un processus = exécution d'un programme**

# Pour introduire les processus (2)

- **Comment passe-t-on des sources à l'exécutable?**



**code source**

**code exécutable**

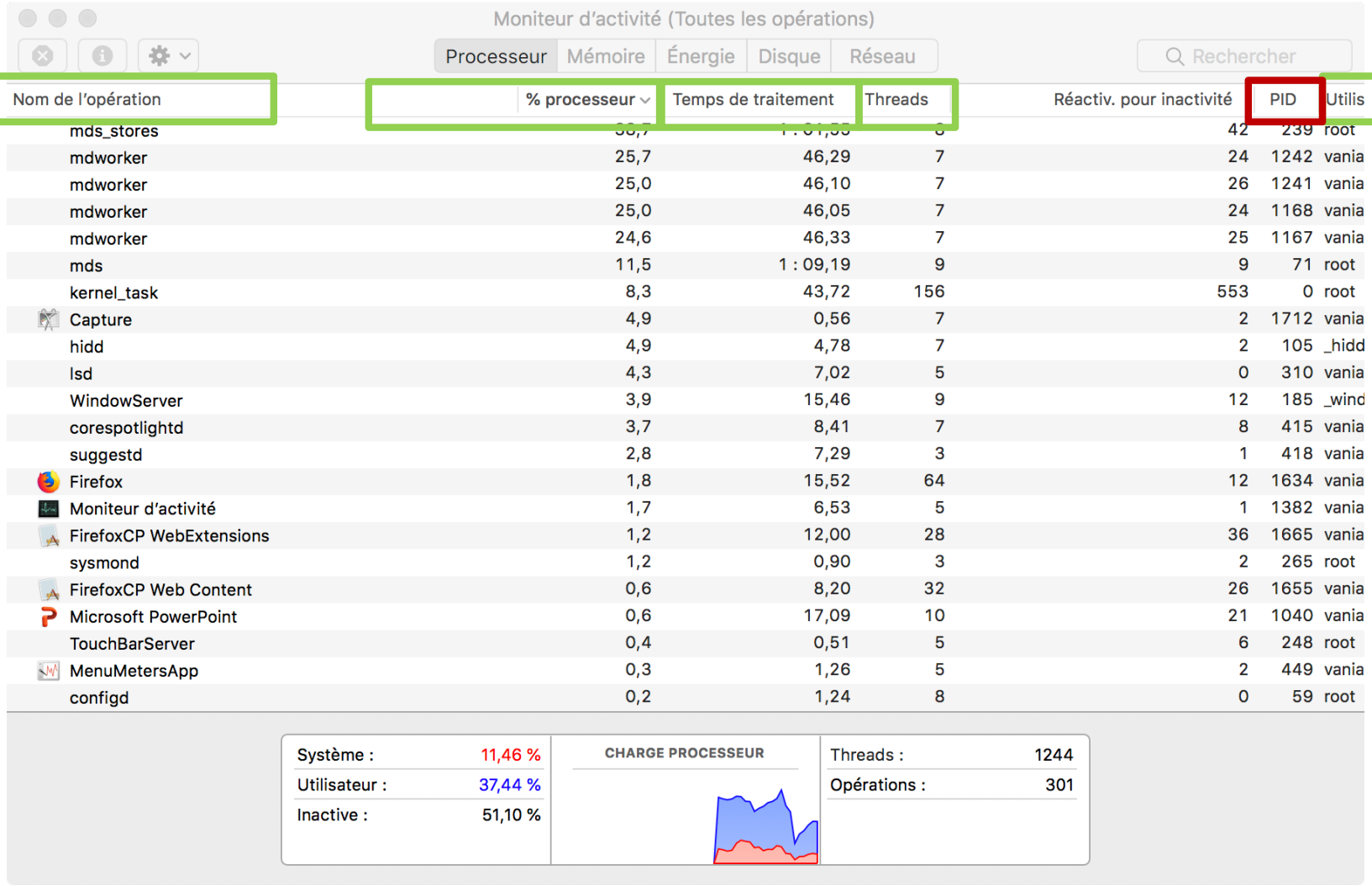
- **soit le programme est compilé**
  - langage C, C++, Pascal, Cobol, Fortran, LaTeX, ...
  - on obtient un fichier exécutable et on le lance
- **soit le programme est interprété**
  - Python, PHP, ...
  - il y a déjà un exécutable dont le rôle est d'interpréter le code source
  - c'est l'interpréteur qui est lancé

# Les processus

- **Un processus = exécution d'un programme**
- **Types de processus**
  - Système
    - Pilotes d'entrées/sorties, processus de sauvegarde, démons...
    - Usuellement lancés au démarrage
  - Utilisateur
    - Applications : Nescape, Facebook, mplayer...
    - Exécution de commandes : ls, make, gcc, python, ...
    - Lancés par action explicite de l'utilisateur
- **Le système gère les processus = il sait/peut tout**
  - sait combien de processus sont actuellement en exécution
  - sait ce qu'ils exécutent
  - sait combien de ressources ils utilisent
  - peut agir sur leur état de fonctionnement
  - décide si et quand ils s'exécutent

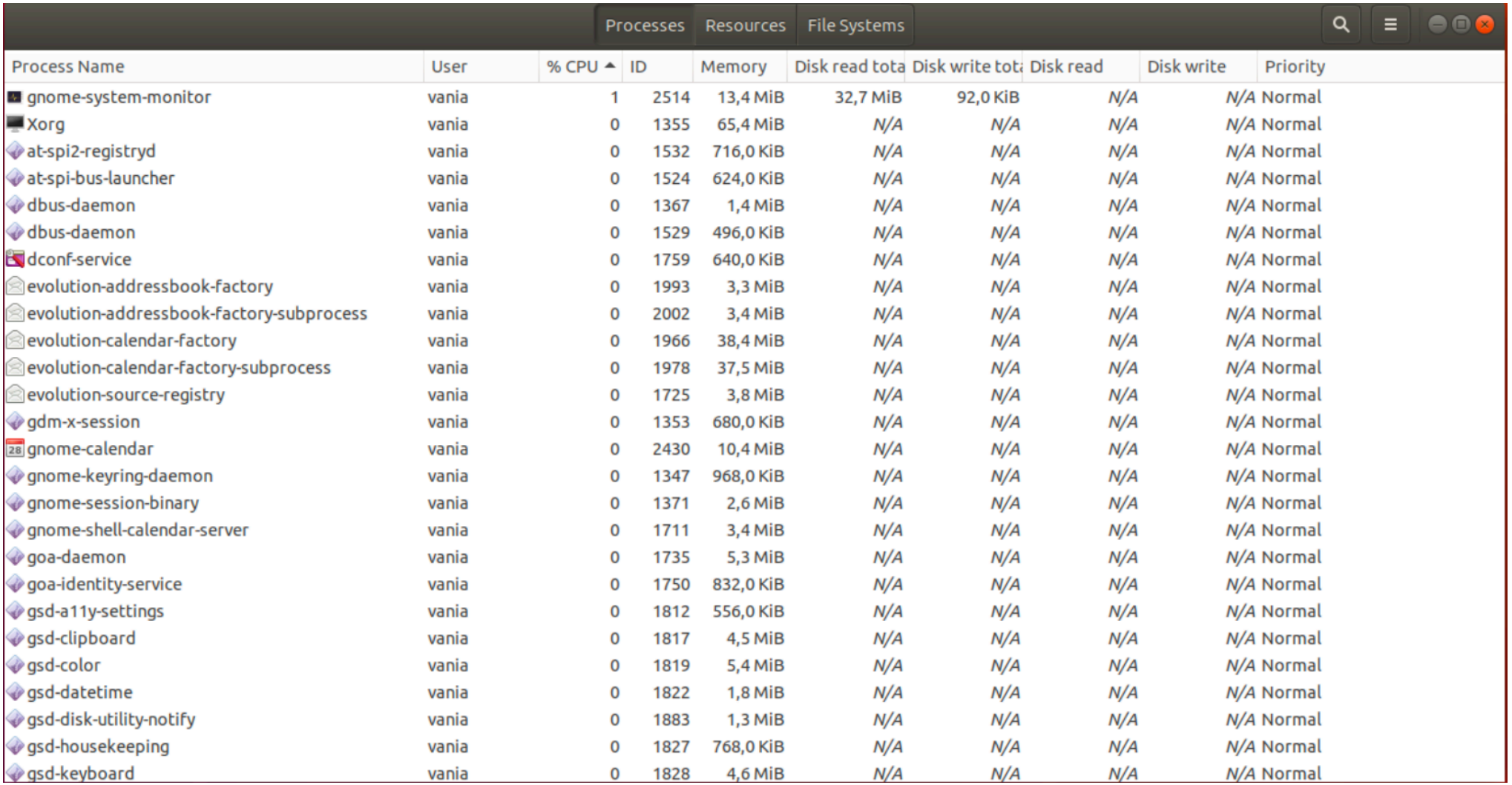
# Comment voir les processus?

- **L'utilitaire système (Moniteur système)**
  - sous Mac



# Comment voir les processus?

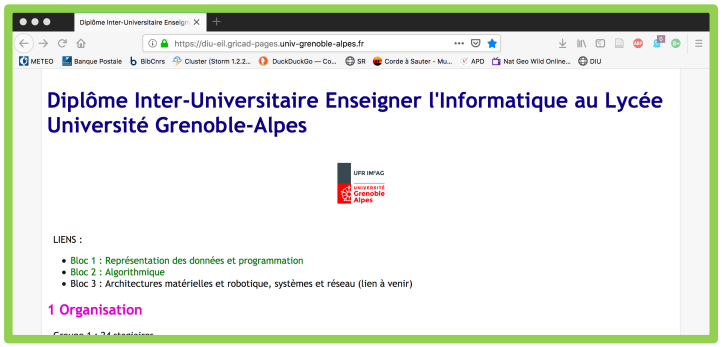
- **L'utilitaire système**
  - sous Linux

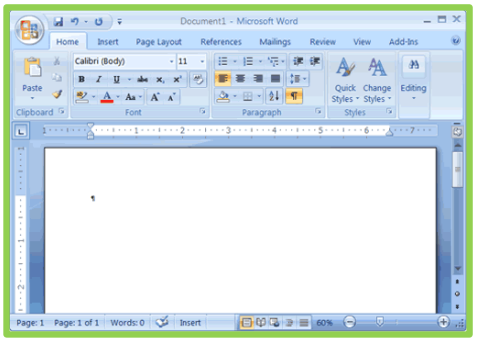


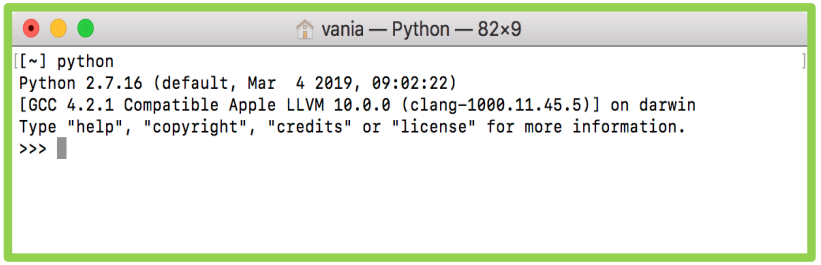

| Process Name | User | % CPU | ID | Memory | Disk read tota | Disk write tot | Disk read | Disk write | Priority |
|---|---|---|---|---|---|---|---|---|---|
| gnome-system-monitor | vania | 1 | 2514 | 13,4 MiB | 32,7 MiB | 92,0 KiB | N/A | N/A | Normal |
| Xorg | vania | 0 | 1355 | 65,4 MiB | N/A | N/A | N/A | N/A | Normal |
| at-spi2-registryd | vania | 0 | 1532 | 716,0 KiB | N/A | N/A | N/A | N/A | Normal |
| at-spi-bus-launcher | vania | 0 | 1524 | 624,0 KiB | N/A | N/A | N/A | N/A | Normal |
| dbus-daemon | vania | 0 | 1367 | 1,4 MiB | N/A | N/A | N/A | N/A | Normal |
| dbus-daemon | vania | 0 | 1529 | 496,0 KiB | N/A | N/A | N/A | N/A | Normal |
| dconf-service | vania | 0 | 1759 | 640,0 KiB | N/A | N/A | N/A | N/A | Normal |
| evolution-addressbook-factory | vania | 0 | 1993 | 3,3 MiB | N/A | N/A | N/A | N/A | Normal |
| evolution-addressbook-factory-subprocess | vania | 0 | 2002 | 3,4 MiB | N/A | N/A | N/A | N/A | Normal |
| evolution-calendar-factory | vania | 0 | 1966 | 38,4 MiB | N/A | N/A | N/A | N/A | Normal |
| evolution-calendar-factory-subprocess | vania | 0 | 1978 | 37,5 MiB | N/A | N/A | N/A | N/A | Normal |
| evolution-source-registry | vania | 0 | 1725 | 3,8 MiB | N/A | N/A | N/A | N/A | Normal |
| gdm-x-session | vania | 0 | 1353 | 680,0 KiB | N/A | N/A | N/A | N/A | Normal |
| gnome-calendar | vania | 0 | 2430 | 10,4 MiB | N/A | N/A | N/A | N/A | Normal |
| gnome-keyring-daemon | vania | 0 | 1347 | 968,0 KiB | N/A | N/A | N/A | N/A | Normal |
| gnome-session-binary | vania | 0 | 1371 | 2,6 MiB | N/A | N/A | N/A | N/A | Normal |
| gnome-shell-calendar-server | vania | 0 | 1711 | 3,4 MiB | N/A | N/A | N/A | N/A | Normal |
| goa-daemon | vania | 0 | 1735 | 5,3 MiB | N/A | N/A | N/A | N/A | Normal |
| goa-identity-service | vania | 0 | 1750 | 832,0 KiB | N/A | N/A | N/A | N/A | Normal |
| gsd-a11y-settings | vania | 0 | 1812 | 556,0 KiB | N/A | N/A | N/A | N/A | Normal |
| gsd-clipboard | vania | 0 | 1817 | 4,5 MiB | N/A | N/A | N/A | N/A | Normal |
| gsd-color | vania | 0 | 1819 | 5,4 MiB | N/A | N/A | N/A | N/A | Normal |
| gsd-datetime | vania | 0 | 1822 | 1,8 MiB | N/A | N/A | N/A | N/A | Normal |
| gsd-disk-utility-notify | vania | 0 | 1883 | 1,3 MiB | N/A | N/A | N/A | N/A | Normal |
| gsd-housekeeping | vania | 0 | 1827 | 768,0 KiB | N/A | N/A | N/A | N/A | Normal |
| gsd-keyboard | vania | 0 | 1828 | 4,6 MiB | N/A | N/A | N/A | N/A | Normal |

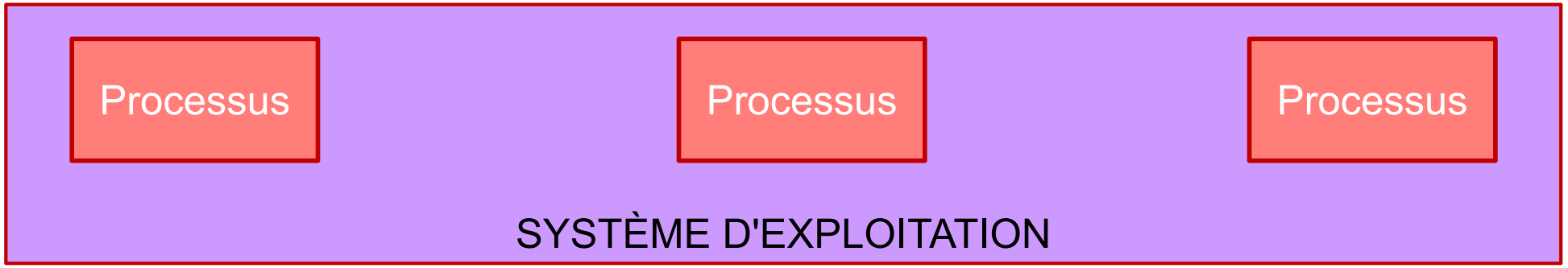
# Exemple de lancement de processus

- **Lançons Firefox, Microsoft Word et l'interpréteur Python**



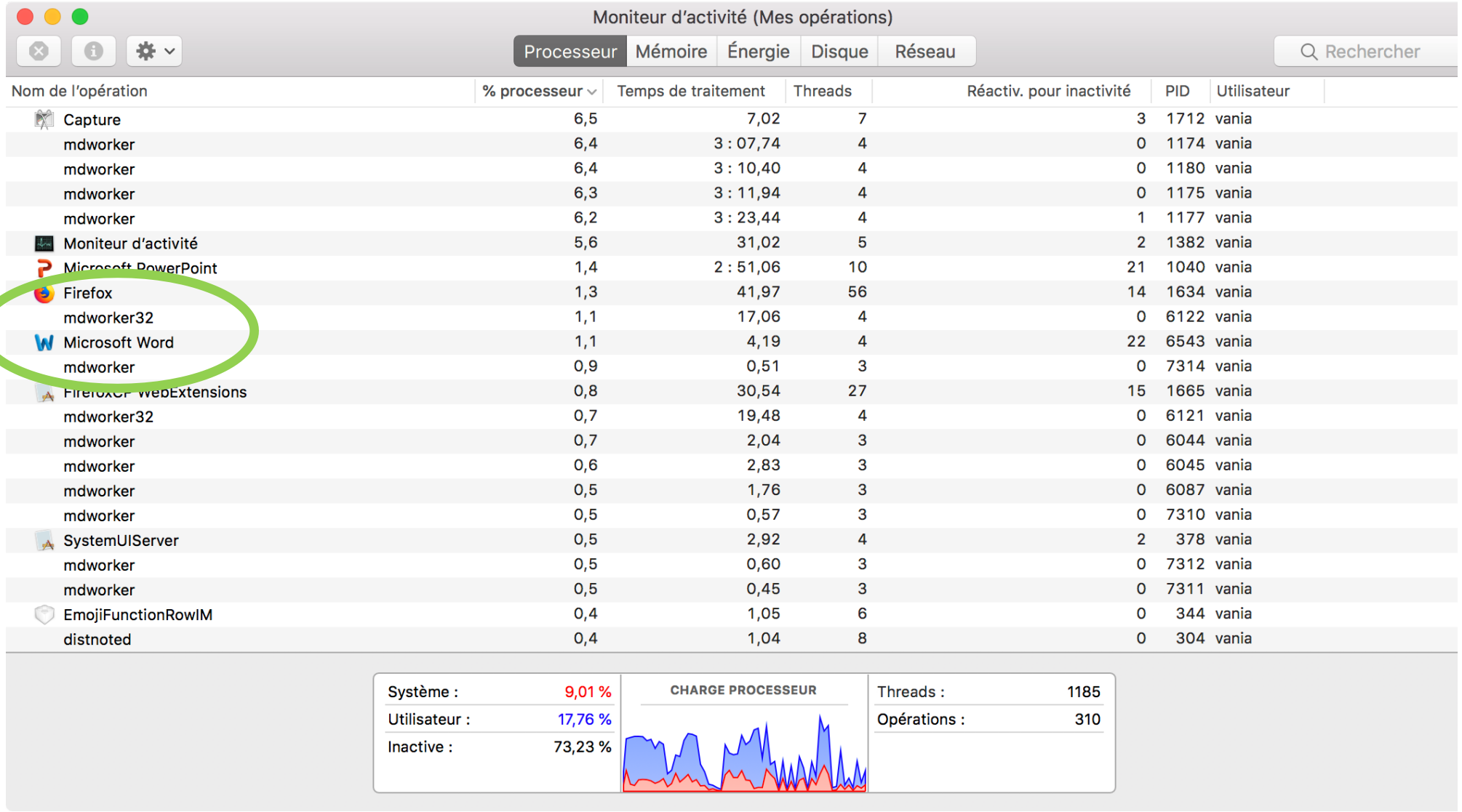





# Observation des processus lancés (interface graphique)

- **Vérifions...**




| Nom de l'opération | % processeur | Temps de traitement | Threads | Réactiv. pour inactivité | PID | Utilisateur |
|---|---|---|---|---|---|---|
| Capture | 6,5 | 7,02 | 7 | 3 | 1712 | vania |
| mdworker | 6,4 | 3 : 07,74 | 4 | 0 | 1174 | vania |
| mdworker | 6,4 | 3 : 10,40 | 4 | 0 | 1180 | vania |
| mdworker | 6,3 | 3 : 11,94 | 4 | 0 | 1175 | vania |
| mdworker | 6,2 | 3 : 23,44 | 4 | 1 | 1177 | vania |
| Moniteur d'activité | 5,6 | 31,02 | 5 | 2 | 1382 | vania |
| Microsoft PowerPoint | 1,4 | 2 : 51,06 | 10 | 21 | 1040 | vania |
| Firefox | 1,3 | 41,97 | 56 | 14 | 1634 | vania |
| mdworker32 | 1,1 | 17,06 | 4 | 0 | 6122 | vania |
| Microsoft Word | 1,1 | 4,19 | 4 | 22 | 6543 | vania |
| mdworker | 0,9 | 0,51 | 3 | 0 | 7314 | vania |
| FirefoxCP WebExtensions | 0,8 | 30,54 | 27 | 15 | 1665 | vania |
| mdworker32 | 0,7 | 19,48 | 4 | 0 | 6121 | vania |
| mdworker | 0,7 | 2,04 | 3 | 0 | 6044 | vania |
| mdworker | 0,6 | 2,83 | 3 | 0 | 6045 | vania |
| mdworker | 0,5 | 1,76 | 3 | 0 | 6087 | vania |
| mdworker | 0,5 | 0,57 | 3 | 0 | 7310 | vania |
| SystemUIServer | 0,5 | 2,92 | 4 | 2 | 378 | vania |
| mdworker | 0,5 | 0,60 | 3 | 0 | 7312 | vania |
| mdworker | 0,5 | 0,45 | 3 | 0 | 7311 | vania |
| EmojiFunctionRowIM | 0,4 | 1,05 | 6 | 0 | 344 | vania |
| distnoted | 0,4 | 1,04 | 8 | 0 | 304 | vania |

# Comment voir les processus en ligne de commande?

- **Commande `ps (process status)`**

vania — less ◂ man ps — 83×21

```
PS(1)                    BSD General Commands Manual                    PS(1)

NAME
     ps -- process status

SYNOPSIS
     ps [-AaCcEefhjlMmrSTvwXx] [-O fmt | -o fmt] [-G gid[,gid...]]
        [-g grp[,grp...]] [-u uid[,uid...]] [-p pid[,pid...]]
        [-t tty[,tty...]] [-U user[,user...]]
     ps [-L]

DESCRIPTION
     The ps utility displays a header line, followed by lines containing
     information about all of your processes that have controlling terminals.

     A different set of processes can be selected for display by using any
     combination of the -a, -G, -g, -p, -T, -t, -U, and -u options.  If more
     than one of these options are given, then ps will select all processes
     which are matched by at least one of the given options.
:
```

# Un aparté : la commande `man`



**Dans un terminal, taper `man man`**

```
vania@vania-ubuntu: ~
File  Edit  View  Search  Terminal  Help
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-C  file]  [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path]
       [-S list] [-e extension] [-i|-I]  [--regex|--wildcard]  [--names-only]  [-a]  [-u]  [--no-subpages]  [-P
       pager]  [-r  prompt]  [-7]  [-E  encoding]  [--no-hyphenation]  [--no-justification]  [-p  string]  [-t]
       [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
       man -k [apropos options] regexp ...
       man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
       man -f [whatis options] page ...
       man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7]
       [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
       man -w|-W [-C file] [-d] [-D] page ...
       man -c [-C file] [-d] [-D] page ...
       man [-?V]

DESCRIPTION
       man  is  the  system's manual pager.  Each page argument given to man is normally the name of a program,
       utility or function.  The manual page associated with each of these arguments is  then  found  and  dis-
       played.   A  section,  if  provided,  will  direct  man to look only in that section of the manual.  The
       default action is to search in all of the available sections following a pre-defined order ("1 n l 8 3 2
       3posix  3pm  3perl  3am  5  4 9 6 7" by default, unless overridden by the SECTION directive in /etc/man-
       path.config), and to show only the first page found, even if page exists in several sections.

       The table below shows the section numbers of the manual followed by the types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions eg /etc/passwd
       6   Games
       7   Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

       A manual page consists of several sections.
```

# Utilisation de ps sur l'exemple précédent

- **Lister tous les processus de l'utilisateur vania**
  - sans surprise, il y en a bcp trop (les mêmes que dans le moniteur...)

vania — -bash — 82×36

```
Last login: Tue May  7 17:57:46 on ttys001
[~] ps -u vania
  UID   PID TTY           TIME CMD
  501   303 ??         0:03.36 /usr/libexec/secd
  501   304 ??         0:01.37 /usr/sbin/distnoted agent
  501   305 ??         0:02.49 /usr/sbin/cfprefsd agent
  501   306 ??         0:01.97 /usr/libexec/UserEventAgent (Aqua)
  501   308 ??         0:00.40 /usr/sbin/universalaccessd launchd -s
  501   309 ??         0:00.74 /System/Library/Frameworks/CoreTelephony.framewo
  501   310 ??         0:11.31 /usr/libexec/lsd
  501   311 ??         0:02.33 /usr/libexec/trustd --agent
  501   315 ??         0:01.06 /System/Library/PrivateFrameworks/CloudKitDaemon
  501   316 ??         0:08.14 /System/Library/Frameworks/ApplicationServices.f
  501   317 ??         0:00.46 /System/Library/PrivateFrameworks/TCC.framework/
  501   318 ??         0:00.40 /usr/libexec/nsurlsessiond
  501   320 ??         0:03.64 /System/Library/Frameworks/Accounts.framework/Ve
  501   321 ??         0:00.22 /System/Library/CoreServices/iconservicesagent
  501   322 ??         0:00.89 /System/Library/PrivateFrameworks/TelephonyUtili
  501   324 ??         0:00.03 /System/Library/PrivateFrameworks/CoreCDP.framew
  501   325 ??         0:01.03 /System/Library/PrivateFrameworks/IDS.framework/
  501   326 ??         0:00.31 /System/Library/PrivateFrameworks/IMCore.framewo
  501   327 ??         0:00.09 /System/Library/Frameworks/Security.framework/Ve
  501   328 ??         0:00.20 /System/Library/PrivateFrameworks/PassKitCore.fr
  501   329 ??         0:00.33 /System/Library/Frameworks/AddressBook.framework
  501   330 ??         0:00.36 /System/Library/Frameworks/InputMethodKit.framew
  501   331 ??         0:00.22 /usr/libexec/secinitd
```

# Utilisation de ps sur l'exemple précédent (2)

- **Filtrons...**

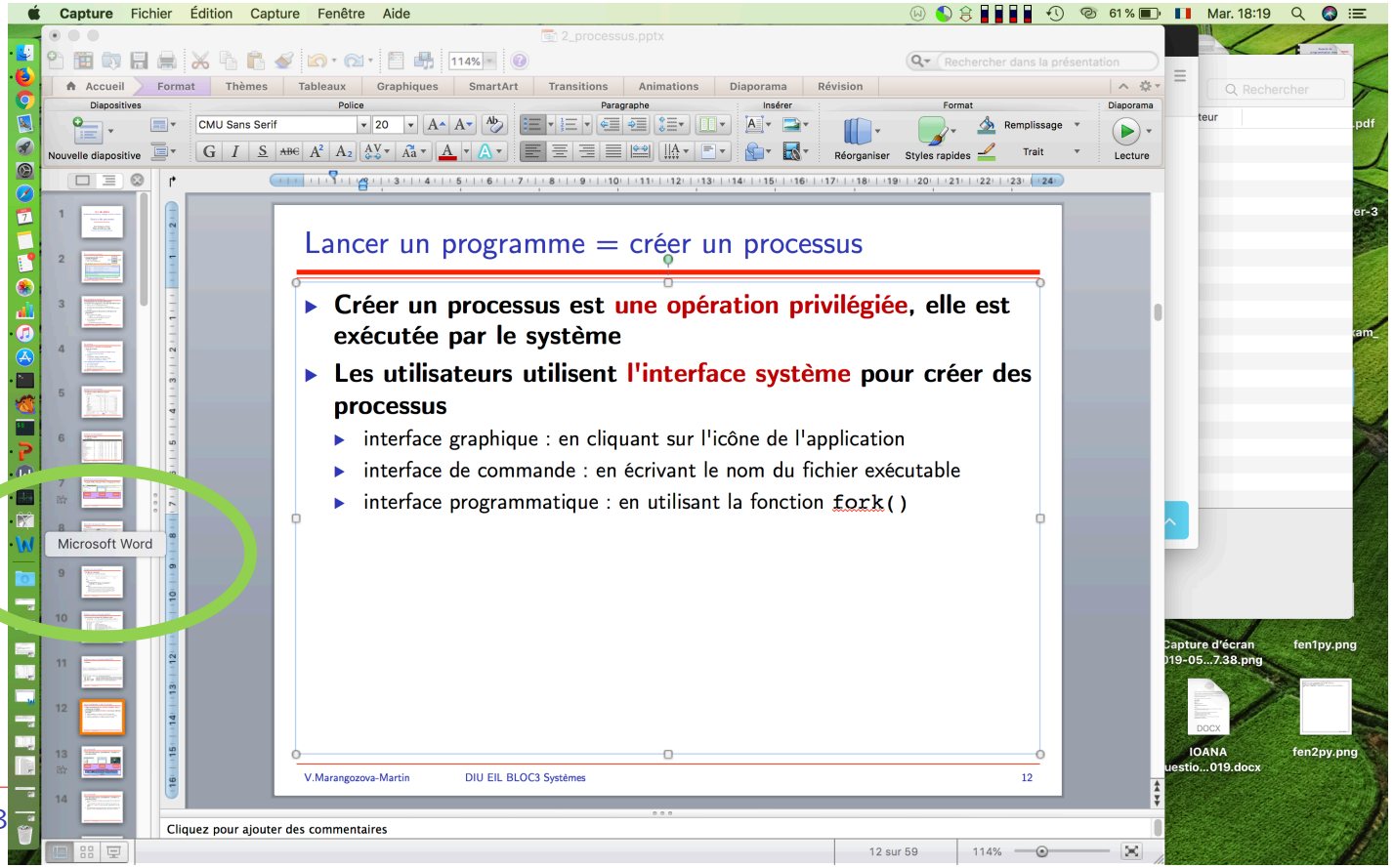
vania — -bash — 163×7

```
[~] ps -u vania | grep -E "(firefox|Word|Python)"
  501  1634 ??         1:13.55 /Applications/Firefox.app/Contents/MacOS/firefox -foreground
  501  6543 ??         0:06.38 /Applications/Microsoft Office 2011/Microsoft Word.app/Contents/MacOS/Microsoft Word
  501  5990 ttys000    0:00.03 /usr/local/Cellar/python@2/2.7.16/Frameworks/Python.framework/Versions/2.7/Resources/Python.app/Contents/MacOS/Python
  501 10206 ttys001    0:00.00 grep -E (firefox|Word|Python)
[~]
```


```
[~] ps -u vania | grep -E "(firefox|Word|Python)"
  501  1634 ??         1:13.55 /Applications/Firefox.app/Contents/MacO
  501  6543 ??         0:06.38 /Applications/Microsoft Office 2011/Mic
  501  5990 ttys000    0:00.03 /usr/local/Cellar/python@2/2.7.16/Frame
  501 10206 ttys001    0:00.00 grep -E (firefox|Word|Python)
[~]
```

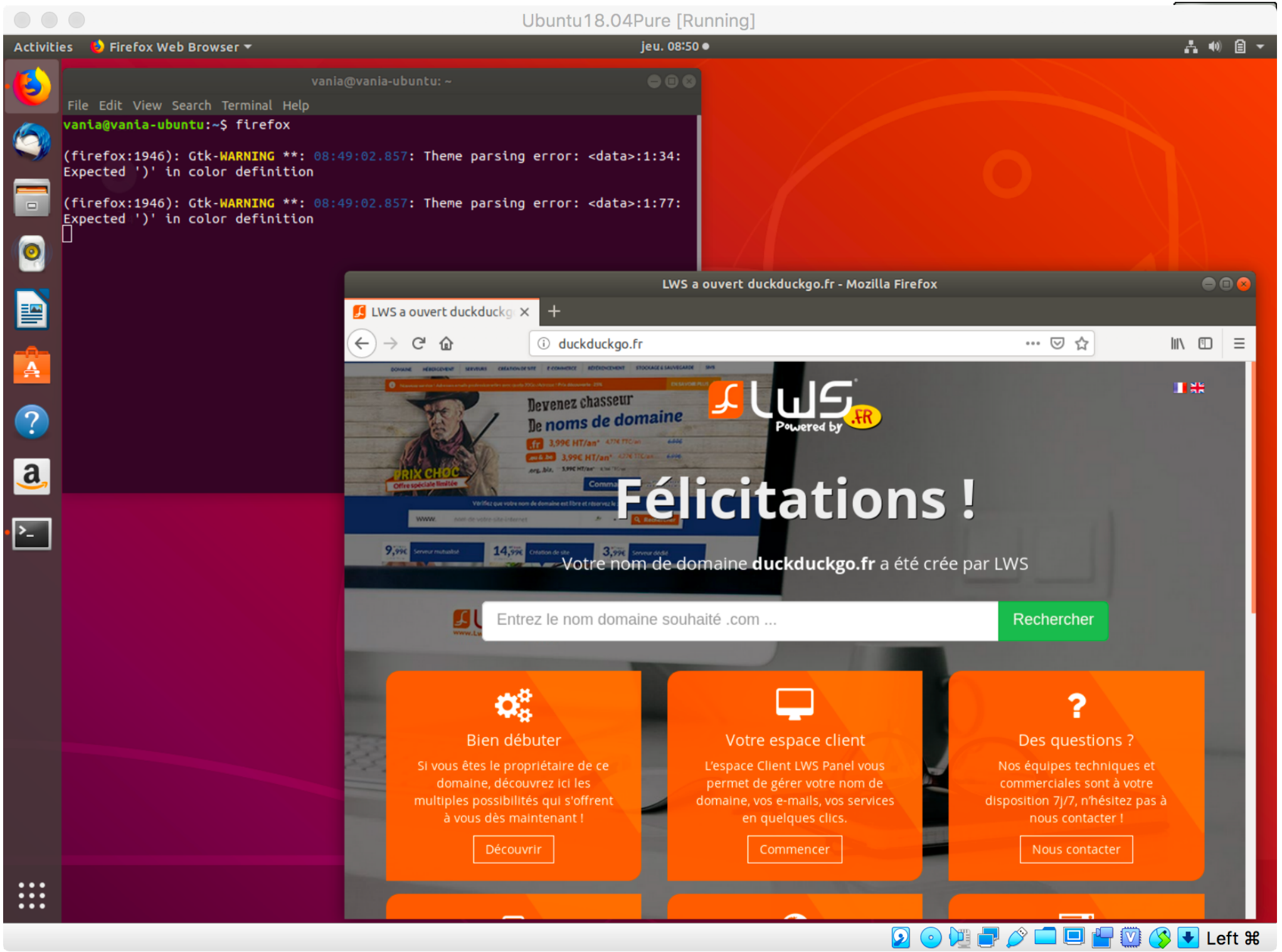
# Lancer un programme = créer un processus

- **Créer un processus est une opération privilégiée, elle est exécutée par le système**
- **Les utilisateurs utilisent l'interface système pour créer des processus**
  - interface graphique : en cliquant sur l'icône de l'application



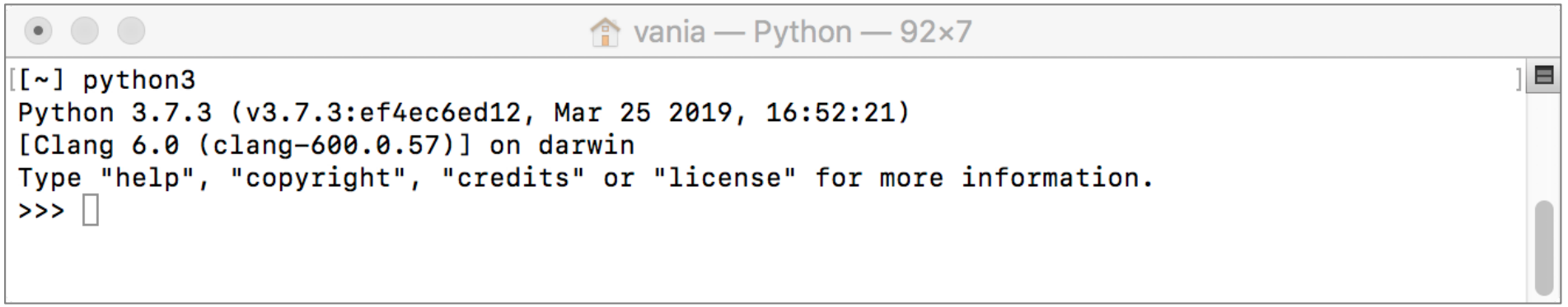
# Lancer un programme = créer un processus

- **Créer un processus est une opération privilégiée, elle est exécutée par le système**
- **Les utilisateurs utilisent l'interface système pour créer des processus**
  - interface graphique : en cliquant sur l'icône de l'application
  - interface de commande : en écrivant le nom du fichier exécutable
    - Lancer le navigateur Mozilla Firefox

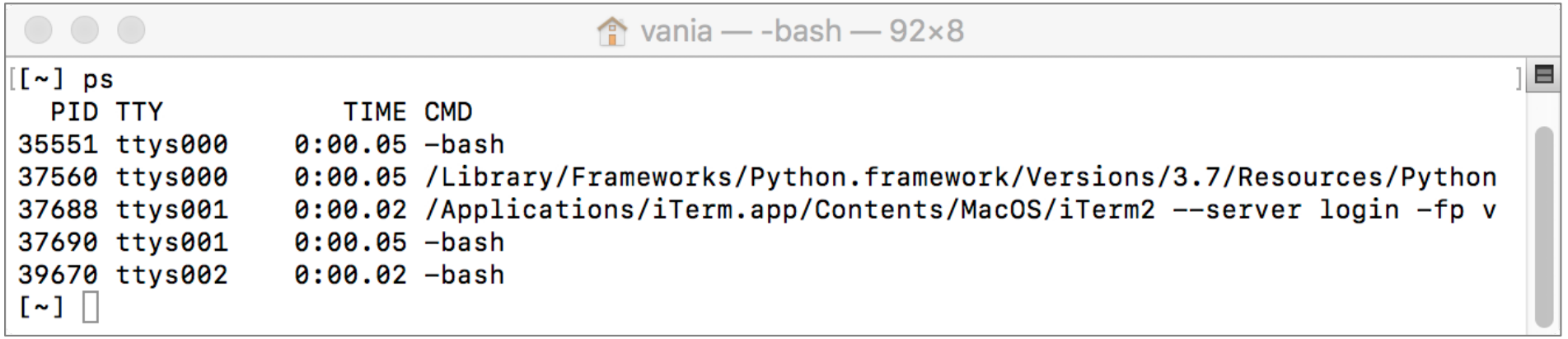

Ubuntu18.04Pure [Running]
Activities
Firefox Web Browser
jeu. 08:50
vania@vania-ubuntu: ~
File Edit View Search Terminal Help
vania@vania-ubuntu:~$ firefox
(firefox:1946): Gtk-WARNING **: 08:49:02.857: Theme parsing error: <data>:1:34:
Expected ')' in color definition
(firefox:1946): Gtk-WARNING **: 08:49:02.857: Theme parsing error: <data>:1:77:
Expected ')' in color definition
LWS a ouvert duckduckgo.fr - Mozilla Firefox
LWS a ouvert duckduckg
duckduckgo.fr
Devenez chasseur
De noms de domaine
Powered by .FR
PRIX CHOC
Offre spéciale limitée
Félicitations !
Votre nom de domaine duckduckgo.fr a été crée par LWS
Entrez le nom domaine souhaité .com ...
Rechercher
Bien débuter
Si vous êtes le propriétaire de ce domaine, découvrez ici les multiples possibilités qui s'offrent à vous dès maintenant !
Découvrir
Votre espace client
L'espace Client LWS Panel vous permet de gérer votre nom de domaine, vos e-mails, vos services en quelques clics.
Commencer
Des questions ?
Nos équipes techniques et commerciales sont à votre disposition 7j/7, n'hésitez pas à nous contacter !
Nous contacter
Left ⌘

# Lancer un programme/créer un processus (cont.)

- **Un autre exemple**
  - Lancer l'interpréteur Python




  - Si on vérifie, on verra le processus python dans la liste des processus



# Lancer un programme/créer un processus (cont.)

- **Un autre exemple**
  - Lancer un programme Python

```
1. bash
[~/Enseignement/DIU_EIL/2_processus/exos] ./hello.py
Hello DIU!
[~/Enseignement/DIU_EIL/2_processus/exos]
```

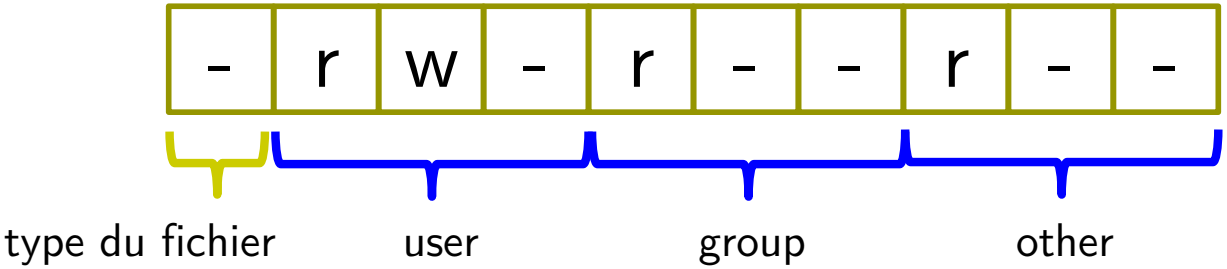
Fichier hello.py

```
#!/usr/bin/env python3

def main():
    print('Hello DIU!')

main()
```

  - Explication
    - Le *shell* a pour rôle d'interpréter la commande
    - Il voit que ce n'est pas une commande intégrée i.e. ce n'est pas lui (le *shell*) qui fournit le fonctionnement/l'implémentation
      - Exemples de commandes intégrées : `chmod`, `passwd`, `bg`, `fg`, ...
    - Il vérifie que c'est un fichier exécutable (voir transparent suivant)
    - Il commence à interpréter et tombe sur la 1ère ligne qui dit que ce qui suit doit être interprété par python3
    - Il lance l'interpréteur python3

# Un aparté sur les droits d'exécution (UNIX)

- **On ne peut exécuter un exécutable que si on a les droits**
- **Les droits/permissions sont encodés sur des bits et concernent**
  - le propriétaire (**u**ser)
  - le groupe (**g**roup)
  - les autres (**o**ther)
  - 3 bits : **r**ead, **w**rite, e**x**ecute



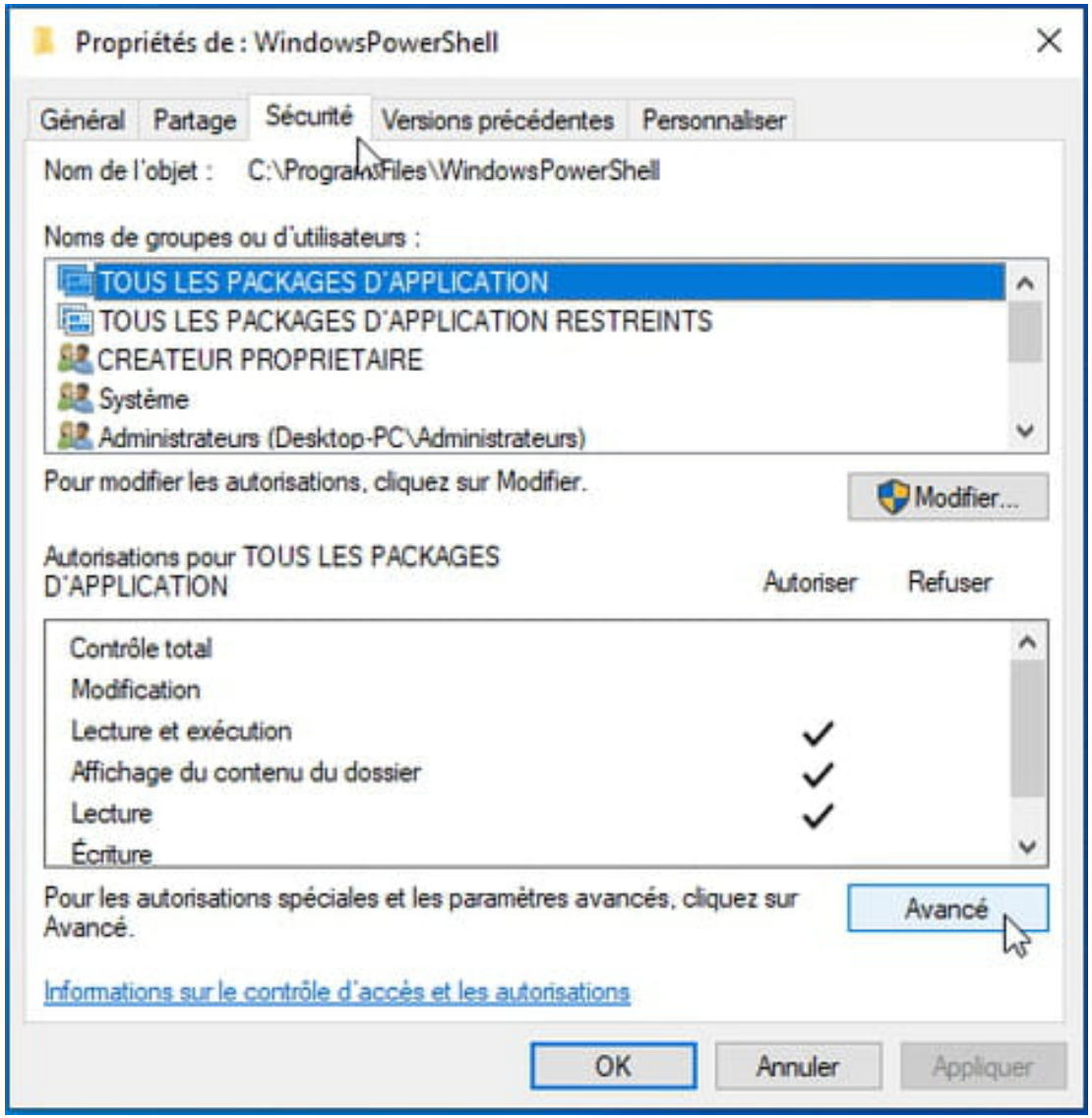

```
[~/Enseignement/DIU_EIL/2_processus/exos] ls -l hello.py
-rwxr-xr-x@ 1 vania  staff  102  9 mai 09:03 hello.py
```

permissions — propriétaire — groupe — taille — date modif — date chemin

- **Changer les bits de permission : chmod, chgrp, chown**

```
[~/Enseignement/DIU_EIL/2_processus/exos] chmod a-x hello.py
[~/Enseignement/DIU_EIL/2_processus/exos] ./hello.py
-bash: ./hello.py: Permission denied
[~/Enseignement/DIU_EIL/2_processus/exos]
```

# Les droits sous Windows



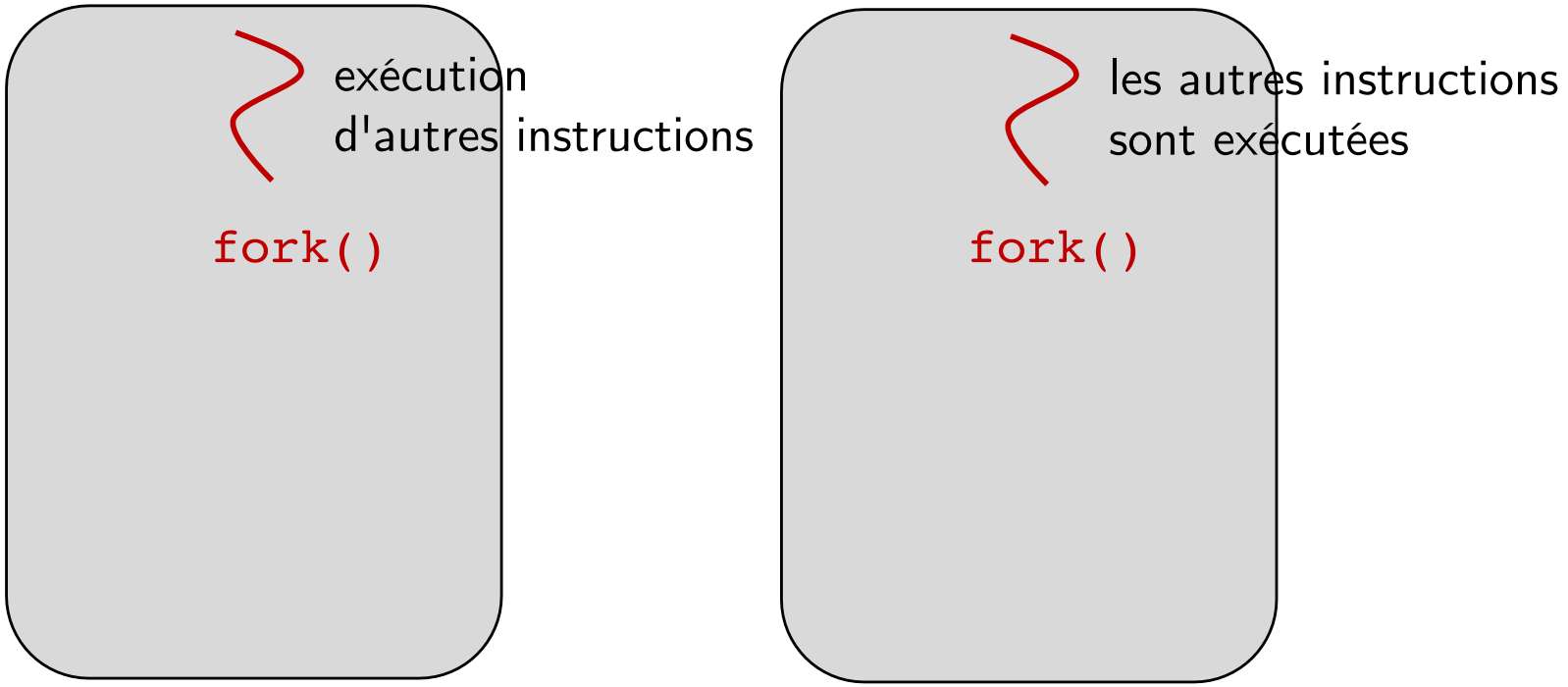
# Lancer un programme = créer un processus

- **Créer un processus est une opération privilégiée, elle est exécutée par le système**
- **Les utilisateurs utilisent l'interface système pour créer des processus**
  - interface graphique : en cliquant sur l'icône de l'application
  - interface de commande : en écrivant le nom du fichier exécutable
  - interface programmatique : en utilisant la fonction `fork()`
    - `man fork` (interface langage C)

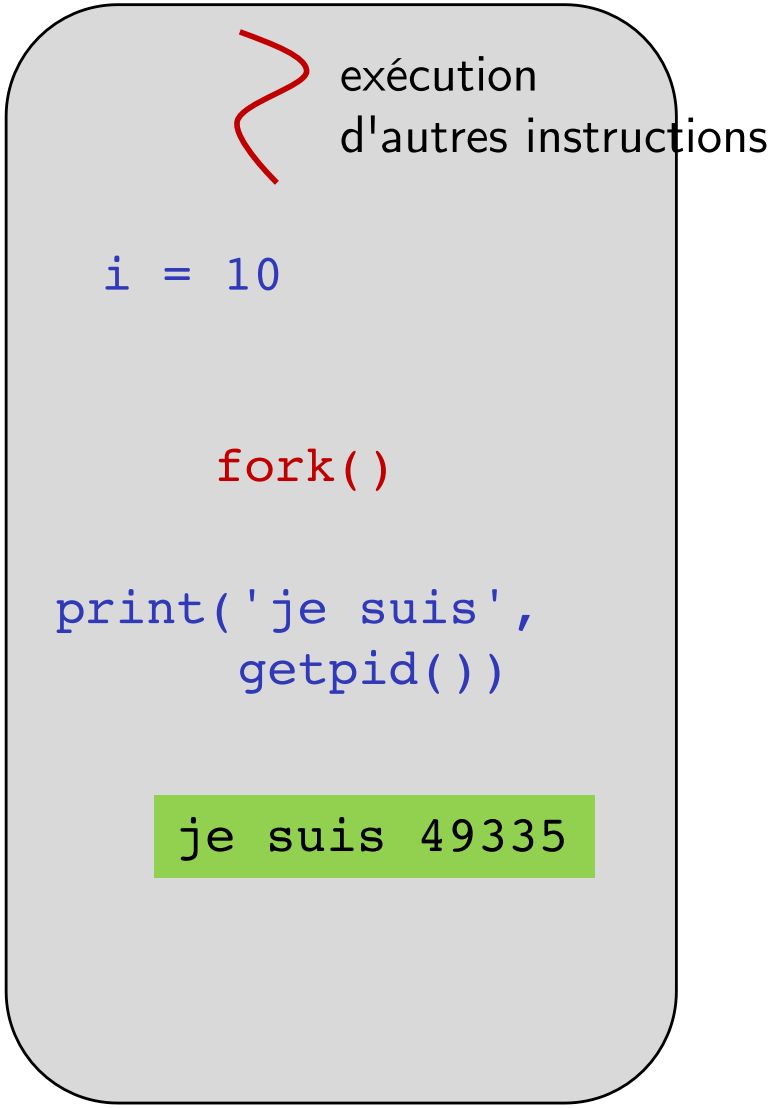
```
FORK(2)                         BSD System Calls Manual

NAME
     fork -- create a new process

SYNOPSIS
     #include <unistd.h>

     pid_t
     fork(void);
```

# Créer un processus avec la fonction `fork()`

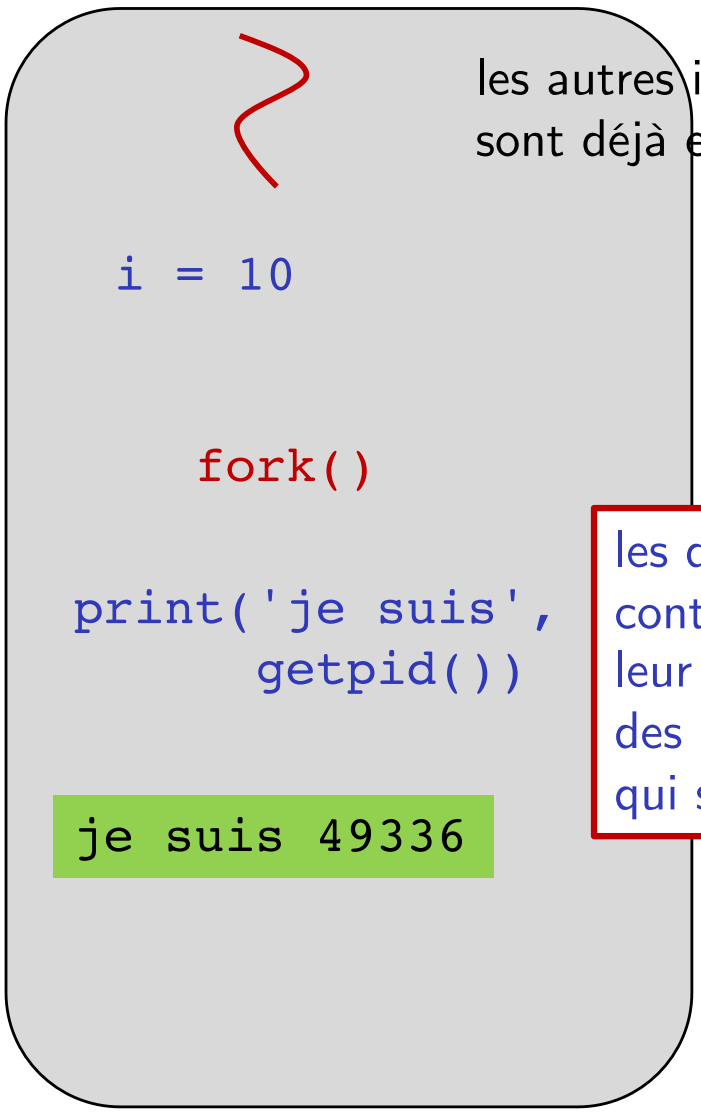
- **Quand un processus exécute la fonction fork()**
  (si pas d'erreur)
  - il crée **une copie conforme** de soi
  - le processus qui crée un autre processus est **le père**
  - le processus créé est **le fils**



# Créer un processus avec la fonction `fork()` (cont.)




processus père avec PID 49335




processus fils avec PID 49336

ce sont deux copies
de la variable i

les deux processus
continuent en parallèle
leur exécution,
des instructions
qui suivent l'appel à fork()

# Le programme Python correspondant

```python
#!/usr/bin/env python3

import os

os.fork()
print('je suis', os.getpid())
```

```python
#!/usr/bin/env python3

import os

def main():
    os.fork()
    print('je suis', os.getpid())

main()
```

1. bash

```
[~/Enseignement/DIU_EIL/2_processus/exos] chmod a+x 1_fork_illustration.py
[~/Enseignement/DIU_EIL/2_processus/exos] ./1_fork_illustration.py
je suis 52479
je suis 52480
[~/Enseignement/DIU_EIL/2_processus/exos]
```

# L'exemple en un peu plus informatif

- **Chaque processus imprime son PID et le PID de son père**

```python
#!/usr/bin/env python3

import os

def main():
    os.fork()
    print('je suis', os.getpid(), ' et mon père est ', os.getppid() )

main()
```

- **Le père du père est le shell**

```
exos — -bash — 80×7
[~/Enseignement/DIU_EIL/2_processus/exos] ./1_fork_illustration.py
je suis 56112  et mon père est  55879
je suis 56113  et mon père est  56112
[~/Enseignement/DIU_EIL/2_processus/exos] ps
  PID TTY           TIME CMD
55879 ttys000    0:00.04 -bash
[~/Enseignement/DIU_EIL/2_processus/exos]
```

# Revenons à `fork()`

- **Fonction surprenante : un appel, deux retours d'appel**
  - Pour le père, **`fork()`** retourne l'identifiant (*PID*) du fils
  - Pour le fils, **`fork()`** retourne 0

```python
#!/usr/bin/env python3

import os

def main():
    print('[père] mon pid est ', os.getpid())
    newpid = os.fork()
    if newpid == 0:
        print('[fils] mon pid est ', os.getpid())
        print('[fils] mon père a comme pid ', os.getppid())
        os._exit(2)
    else:
        print('[père] mon fils a comme pid ', newpid)
        res = os.waitpid(newpid, 0)
        status = res[1]
        os._exit(0)

main()
```

```
[~/Enseignement/DIU_EIL/2_processus/exos] ./2_fork_illustration.py
[père] mon pid est  64791
[père] mon fils a comme pid  64792
[fils] mon pid est  64792
[fils] mon père a comme pid  64791
```

# Revenons à `fork()` (cont.)

- **Fonction surprenante : un appel, deux retours d'appel**
  - Pour le père, **`fork()`** retourne l'identifiant (*PID*) du fils
  - Pour le fils, **`fork()`** retourne 0

```
#!/usr/bin/env python3

import os

def main():
    print('[père] mon pid est ', os.getpid())
    newpid = os.fork()
    if newpid == 0:
        print('[fils] mon pid est ', os.getpid())
        print('[fils] mon père a comme pid ', os.getppid())
        os._exit(2)
    else:
        print('[père] mon fils a comme pid ', newpid)
        res = os.waitpid(newpid, 0)
        status = res[1]
        os._exit(0)

main()
```

`exit()` marque la terminaison d'un processus, en argument le code de retour

Le père attend la terminaison du fils et récupère son code de retour. Si on ne le fait pas, le fils peut se transformer on zombi

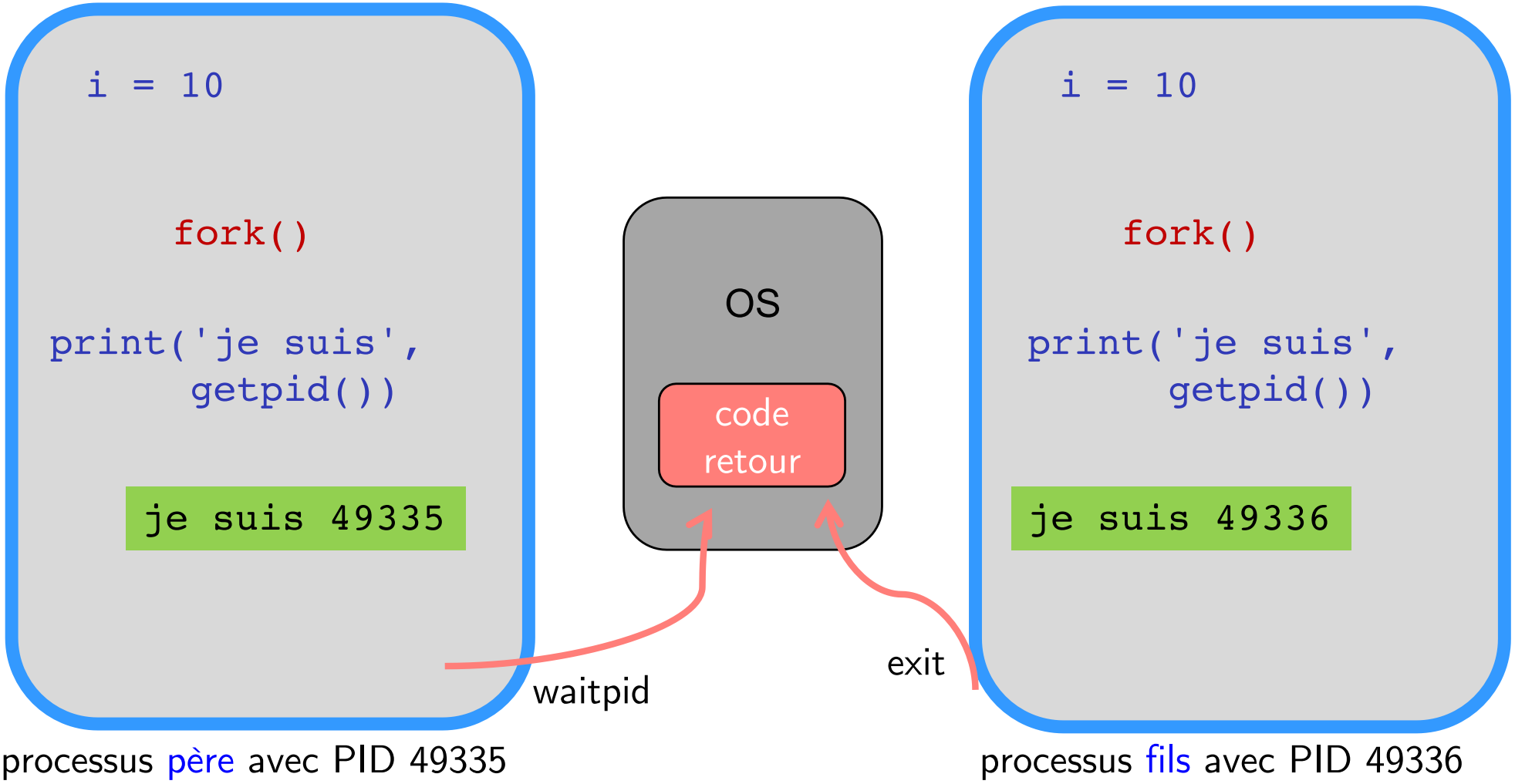
# Si le père n'attend pas le fils

```python
#!/usr/bin/env python3

import os

def main():
    print('[père] mon pid est ', os.getpid())
    newpid = os.fork()
    if newpid == 0:
        print('[fils] mon pid est ', os.getpid())
        print('[fils] mon père a comme pid ', os.getppid())
    else:
        print('[père] mon fils a comme pid ', newpid)
    os._exit(0)

main()
```

```
./3_fork_illustration.py
[père] mon pid est  67007
[père] mon fils a comme pid  67008
[fils] mon pid est  67008
[fils] mon père a comme pid  1
[~/Enseignement/DIU_EIL/2_processus/exos] ps -A | grep /sbin/launchd
    1 ??         4:07.98 /sbin/launchd
```

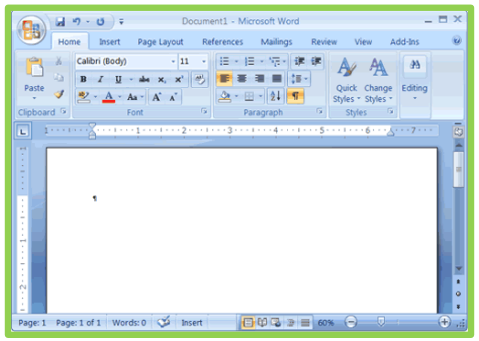
# Communication entre processus

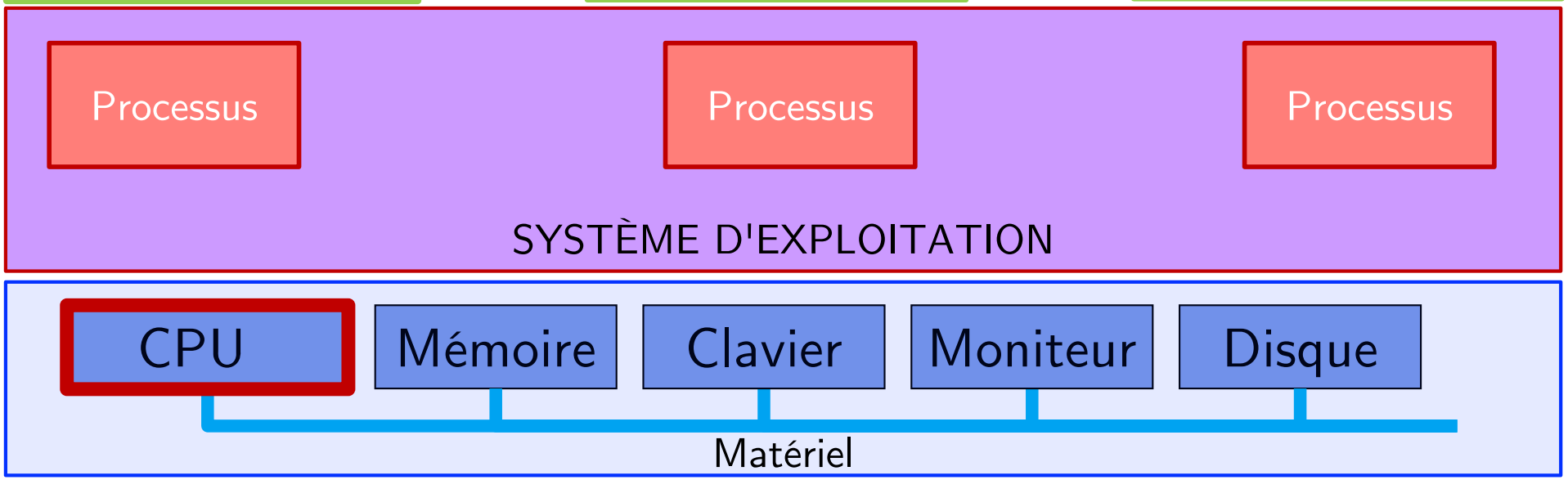
- **Les processus sont isolés et ne partagent pas de données**



# Exécution des processus

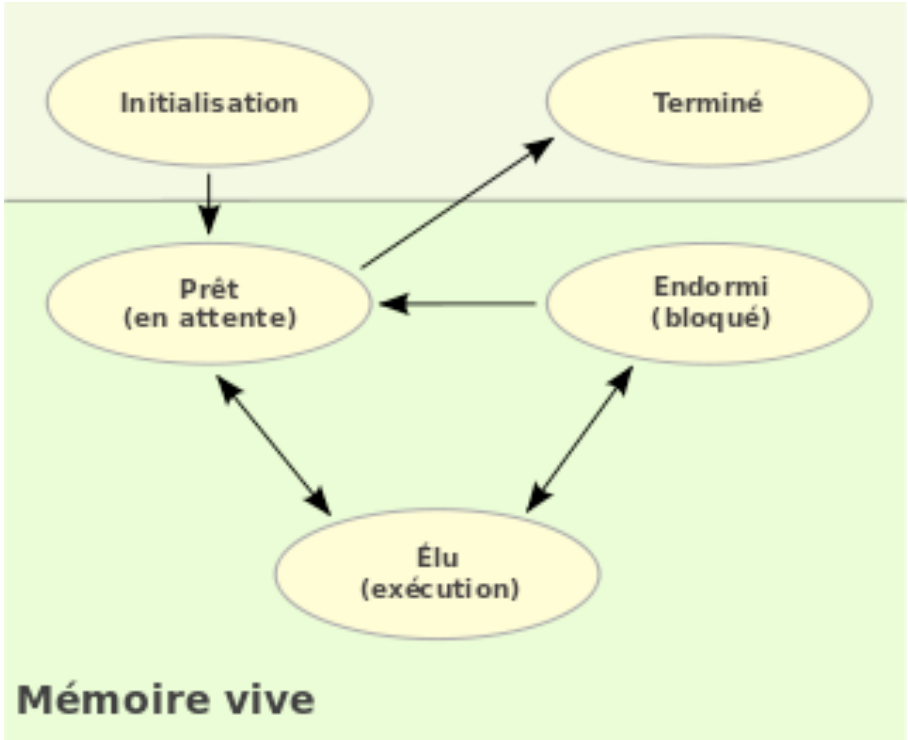
- **Les processus s'exécutent sur le processeur**
  - Un seul processeur (cœur) ne peut exécuter qu'une chose à la fois
  - Si plusieurs processus : un s'exécute, les autres attendent

# Exécution des processus sur le processeur

- **Pour exécuter un processus sur le processeur, le processus doit être prêt**
  - En effet, un processus peut ne pas être en état de s'exécuter puisque en attente
    - d'entrée clavier
    - de réception de message réseau
    - en attente d'un autre processus (comme pour `waitpid`)
    - ...
- **Un processus peut avoir plusieurs états**




*source : Wikipedia*

# Exécution des processus sur le processeur (cont.)

- **Observer les états d'un processus (dans `man ps`)**

vania@vania-ubuntu: ~/python-exos

File Edit View Search Terminal Help

**PROCESS STATE CODES**

Here are the different values that the **s**, **stat** and **state** output specifiers (header "STAT" or "S") will display to describe the state of a process:

| | |
|---|---|
| D | uninterruptible sleep (usually IO) |
| R | running or runnable (on run queue) |
| S | interruptible sleep (waiting for an event to complete) |
| T | stopped by job control signal |
| t | stopped by debugger during the tracing |
| W | paging (not valid since the 2.6.xx kernel) |
| X | dead (should never be seen) |
| Z | defunct ("zombie") process, terminated but not reaped by its parent |

For BSD formats and when the **stat** keyword is used, additional characters may be displayed:

| | |
|---|---|
| < | high-priority (not nice to other users) |
| N | low-priority (nice to other users) |
| L | has pages locked into memory (for real-time and custom IO) |
| s | is a session leader |
| l | is multi-threaded (using CLONE_THREAD, like NPTL pthreads do) |
| + | is in the foreground process group |

Manual page ps(1) line 431 (press h for help or q to quit)

# Exécution des processus sur le processeur (cont.)

- **Observer les états d'un processus**

```
[~] ps -j
USER    PID  PPID  PGID   SESS JOBC STAT   TT       TIME COMMAND
vania 64535 37686 64535      0    0 Ss   s000    0:00.04 /Applications/iTerm.app/Contents/MacOS/iTerm2 --server login -fp van
vania 64537 64536 64537      0    1 S+   s000    0:00.06 -bash
vania 79811 79810 79811      0    1 S    s001    0:00.07 -bash
vania 81643 79811 81643      0    1 R    s001    0:03.29 /Applications/Firefox.app/Contents/MacOS/firefox
vania 81649 81643 81643      0    1 R    s001    0:00.82 /Applications/Firefox.app/Contents/MacOS/plugin-container.app/Conten
vania 81655 81643 81643      0    1 R    s001    0:00.81 /Applications/Firefox.app/Contents/MacOS/plugin-container.app/Conten
vania 81661 81643 81643      0    1 R    s001    0:00.15 /Applications/Firefox.app/Contents/MacOS/plugin-container.app/Conten
vania 81212 81211 81212      0    1 S    s002    0:00.04 -bash
vania 81561 81212 81561      0    1 R+   s002    0:37.75 ./a.out
```

- **Observer l'arborescence des processus (`ps f` ou `pstree`)**

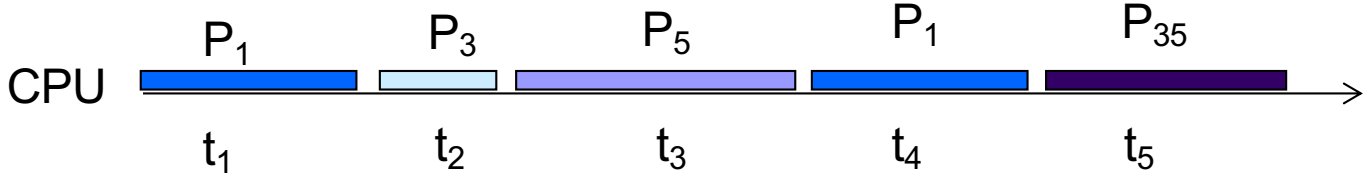
```
  PID TTY      STAT   TIME COMMAND
21916 pts/1    Ss     0:00 bash
21947 pts/1    R+     0:00  \_ ps f
 1925 pts/0    Ss     0:00 bash
Ubuntu Software
21907 pts/0    S+     0:00      \_ pager
 1226 tty2     Ssl+   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSI
 1228 tty2     Rl+    0:14  \_ /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/g
 1238 tty2     Sl+    0:00  \_ /usr/lib/gnome-session/gnome-session-binary --session=ubun
 1400 tty2     Sl+    0:56      \_ /usr/bin/gnome-shell
 1431 tty2     Rl     0:00      |   \_ ibus-daemon --xim --panel disable
 1435 tty2     Sl     0:00      |       \_ /usr/lib/ibus/ibus-dconf
 1809 tty2     Sl     0:00      |       \_ /usr/lib/ibus/ibus-engine-simple
 1638 tty2     Sl+    0:00      \_ /usr/lib/gnome-settings-daemon/gsd-power
 1639 tty2     Sl+    0:00      \_ /usr/lib/gnome-settings-daemon/gsd-print-notifications
 1641 tty2     Sl+    0:00      \_ /usr/lib/gnome-settings-daemon/gsd-rfkill
 1643 tty2     Sl+    0:00      \_ /usr/lib/gnome-settings-daemon/gsd-screensaver-proxy
```

# Combien de processus sont prêts à être exécutés?

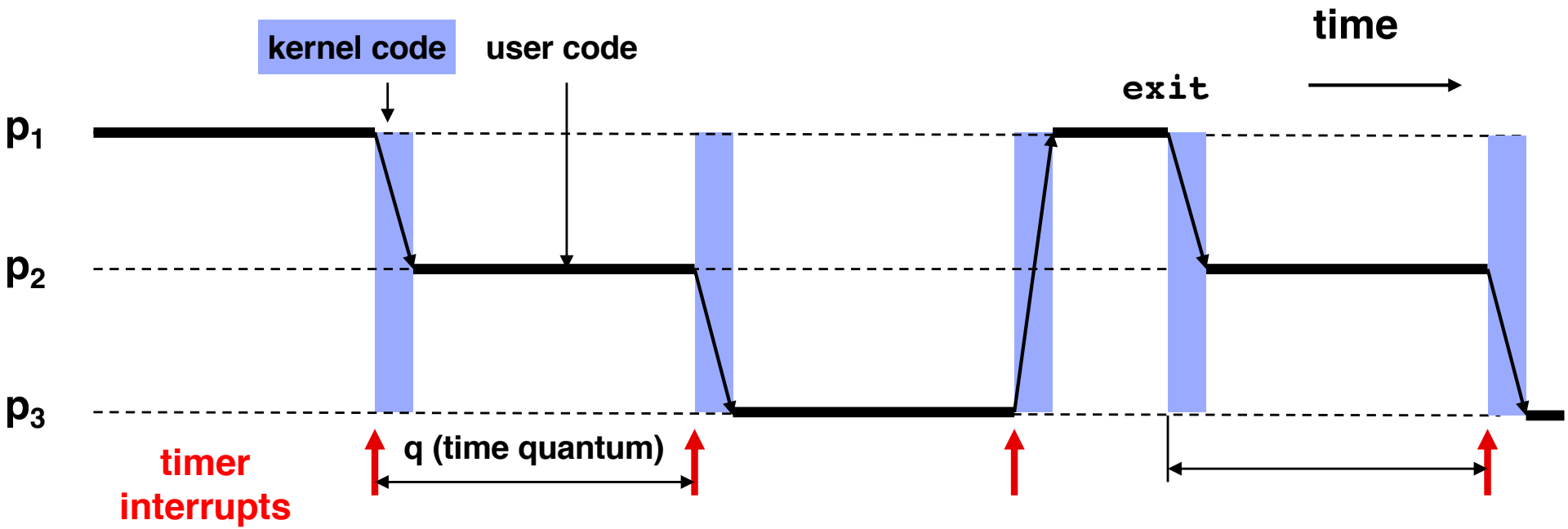
- **Au début de l'informatique : un seul processus était chargé en mémoire et pouvait s'exécuter (*batch processing*)**
  - Le temps de récupérer les résultats et de préparer le prochain processus, le CPU est inutilisé

# Combien de processus sont prêts à être exécutés?

- **Aujourd'hui : multi-programmation**
  - Il est possible de charger plusieurs processus en mémoire
  - Il peut y donc avoir **plusieurs processus prêts**
  - Le choix du processus à exécuter parmi tous ceux qui sont prêts est effectué par un module dédié du noyau système : **ordonnanceur**
  - Le processus s'appelle l'ordonnancement (*scheduling*)
- **Allocation du processeur aux processus**
  - Quand on choisit un processus à exécuter, pour quelle durée lui donner l'accès au processeur?
    - Une approche simple : donner le processeur au processus tant qu'il en a besoin. Il peut y avoir un problème de monopole.
    - L'approche pratiquée : **temps partagé.** Les processus ont accès au CPU pendant une tranche finie

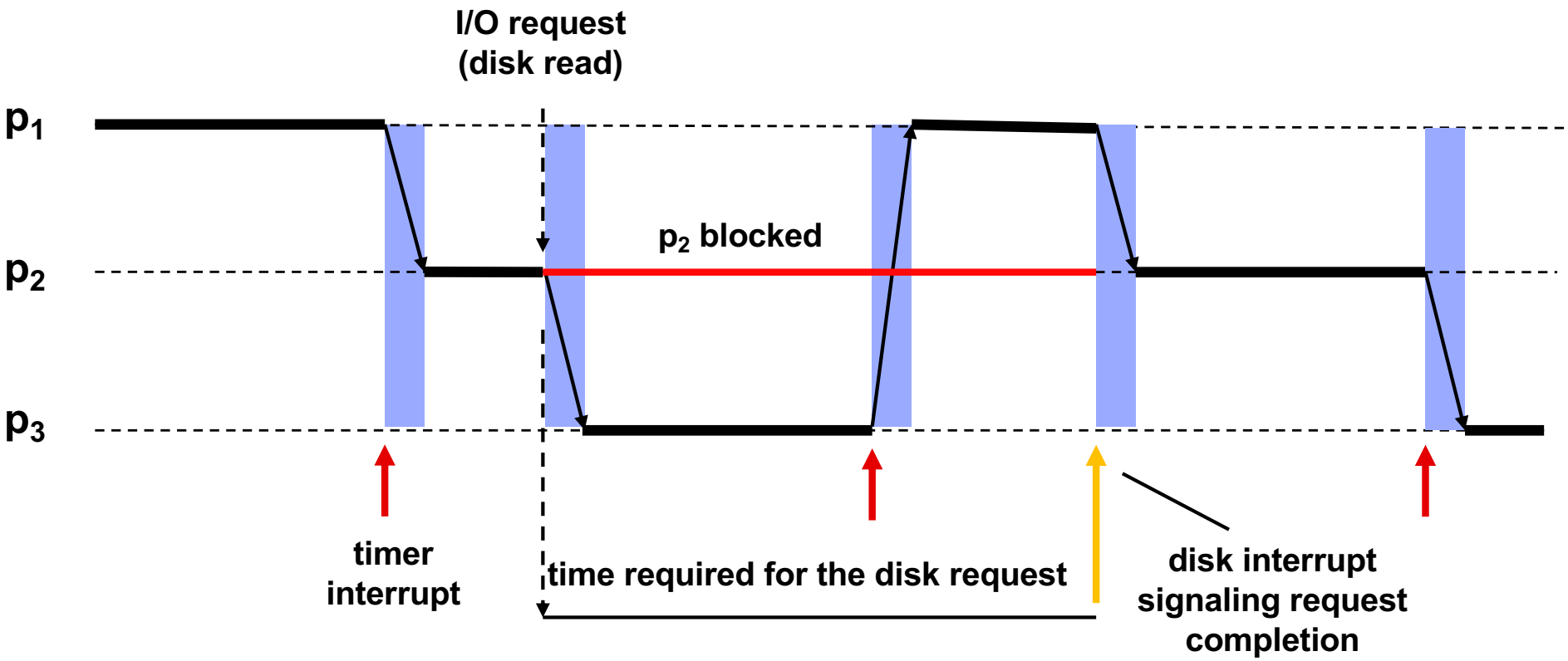


# Exemple d'ordonnancement




Images des slides de R.Lachaize, S.Krakowiak

# Exemples d'ordonnancement (2/2)




Images des slides de R.Lachaize, S.Krakowiak

# Objectifs de l'ordonnancement

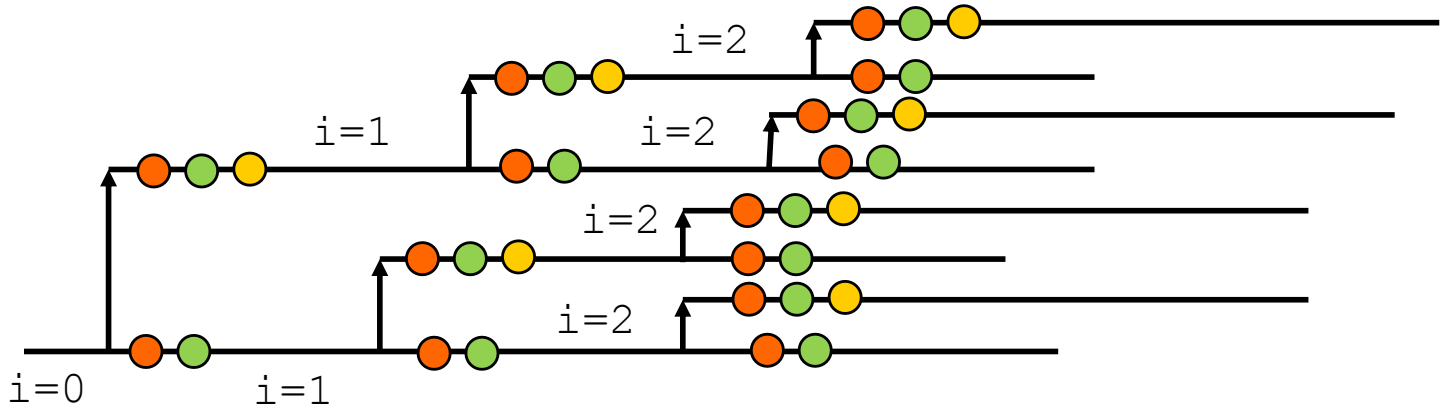
- **Équité**
  - Répartition équitable du temps CPU aux processus
- **Prévention de famine**
  - Les processus ont tous accès au processeur au bout d'un temps fini
- **Performances**
  - Temps de réponse (*turnaround time*)
  - Rendement (*throughput*) : Nombre de tâches complétées / unité de temps
  - Utilisation CPU
  - Temps d'attente
  - ...
- **Il n'y a pas d'algorithme d'ordonnancement universel**
  - Un problème d'optimisation compliqué
  - Un domaine de recherche très actif

# Ordonnancement et **non déterminisme**

- **L'ordre d'exécution des processus dépend**
  - de l'algorithme de l'ordonnancement (l'user ne sait pas)
  - de la charge de la machine i.e. des processus qui sont lancés
  - du comportement de ces processus et du déroulement de leurs interactions avec le monde extérieur
- **Cet ordre ne peut être prédit à l'avance**
  - sauf cas très spécifiques
- **Conséquence : un programme peut se comporter différemment d'une exécution à une autre**
  - prendre plus ou moins de temps
  - si plusieurs processus sont lancés, l'ordre de leur exécution n'est pas garanti (pas déterministe)
    - plus il y a des processus, plus ils ont une durée longue, plus l'entrelacement est important

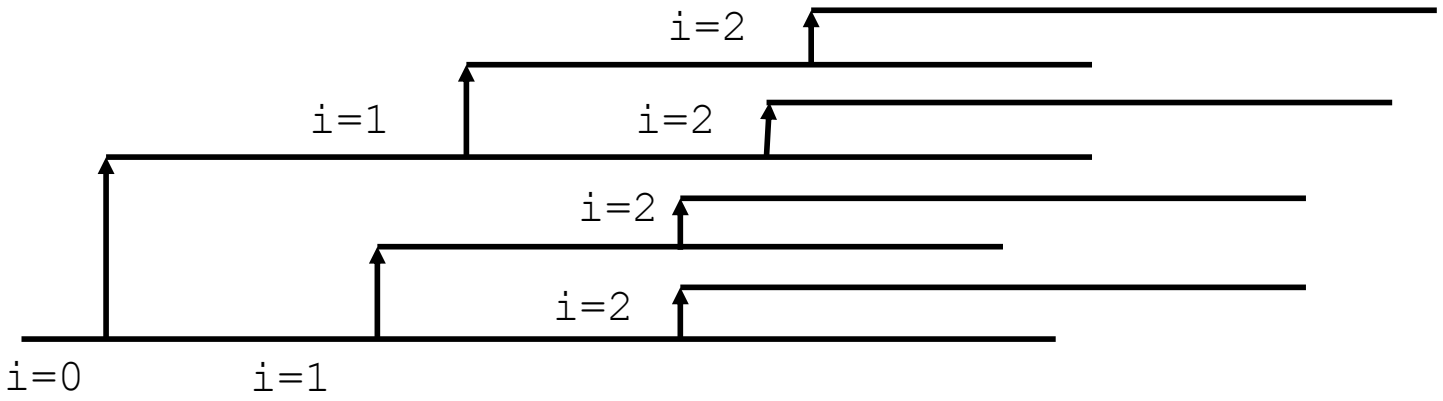
# Observons le non déterminisme

```
#!/usr/bin/env python3

import os
import time

def main():
    print('[parent process] my pid is', os.getpid())
    for i in range(0,3):
        newpid = os.fork()
        print('[process', os.getpid(), '] running')
        print('[process', os.getpid(), ']', i)
        if newpid == 0:
            print('[process', os.getpid(), '] my parent pid', os.getppid())
    time.sleep(10)
    os._exit(0)

main()
```



# Observons le non déterminisme (cont.)

```
\-+= 18731 vania /Library/Frameworks/Python.framework
  |-+- 18732 vania /Library/Frameworks/Python.framewo
  | |-+- 18735 vania /Library/Frameworks/Python.frame
  | | \--- 18738 vania /Library/Frameworks/Python.fra
  | \--- 18737 vania /Library/Frameworks/Python.frame
  |-+- 18733 vania /Library/Frameworks/Python.framewo
  | \--- 18736 vania /Library/Frameworks/Python.frame
  \--- 18734 vania /Library/Frameworks/Python.framewo
```



# Observons le non déterminisme

```
[parent process] my pid is 18731
[process 18731 ] running
[process 18731 ] 0
[process 18731 ] running
[process 18731 ] 1
[process 18732 ] running
[process 18732 ] 0
[process 18732 ] my parent pid 18731
[process 18733 ] running
[process 18733 ] 1
[process 18731 ] running
[process 18731 ] 2
[process 18733 ] my parent pid 18731
[process 18732 ] running
[process 18732 ] 1
[process 18734 ] running
[process 18734 ] 2
[process 18734 ] my parent pid 18731
[process 18733 ] running
[process 18733 ] 2
[process 18735 ] running
[process 18735 ] 1
[process 18735 ] my parent pid 18732
[process 18732 ] running
[process 18732 ] 2
[process 18736 ] running
[process 18736 ] 2
[process 18736 ] my parent pid 18733
[process 18735 ] running
[process 18735 ] 2
[process 18737 ] running
[process 18737 ] 2
[process 18737 ] my parent pid 18732
[process 18738 ] running
[process 18738 ] 2
[process 18738 ] my parent pid 18735
```

```
\-+= 18731 vania /Library/Frameworks
  |-+- 18732 vania /Library/Framewor
  | |-+- 18735 vania /Library/Framew
  | | \--- 18738 vania /Library/Fram
  | \--- 18737 vania /Library/Framew
  |-+- 18733 vania /Library/Framewor
  | \--- 18736 vania /Library/Framew
  \--- 18734 vania /Library/Framewor
```

# Une autre exécution

```
[parent process] my pid is 25104
[process 25104 ] running
[process 25104 ] 0
[process 25104 ] running
[process 25104 ] 1
[process 25105 ] running
[process 25105 ] 0
[process 25105 ] my parent pid 25104
[process 25106 ] running
[process 25106 ] 1
[process 25106 ] my parent pid 25104
[process 25104 ] running
[process 25104 ] 2
[process 25107 ] running
[process 25105 ] running
[process 25105 ] 1
[process 25107 ] 2
[process 25107 ] my parent pid 25104
[process 25108 ] running
[process 25108 ] 1
[process 25108 ] my parent pid 25105
[process 25106 ] running
[process 25106 ] 2
[process 25105 ] running
[process 25105 ] 2
[process 25109 ] running
[process 25109 ] 2
[process 25109 ] my parent pid 25106
[process 25108 ] running
[process 25110 ] running
[process 25108 ] 2
[process 25110 ] 2
[process 25110 ] my parent pid 25105
[process 25111 ] running
[process 25111 ] 2
[process 25111 ] my parent pid 25108
```

- **Les PID sont différents**

```
\-+= 25104 vania /Library/
  |-+- 25105 vania /Librar
  | |-+- 25108 vania /Libr
  | | \--- 25111 vania /Li
  | \--- 25110 vania /Libr
  |-+- 25106 vania /Librar
  | \--- 25109 vania /Libr
  \--- 25107 vania /Librar
```

```
[parent process] my pid is 18731
[process 18731 ] running
[process 18731 ] 0
[process 18731 ] running
[process 18731 ] 1
[process 18732 ] running
[process 18732 ] 0
[process 18732 ] my parent pid 18731
[process 18733 ] running
[process 18733 ] 1
[process 18731 ] running
[process 18731 ] 2
[process 18733 ] my parent pid 18731
[process 18732 ] running
[process 18732 ] 1
[process 18734 ] running
[process 18734 ] 2
[process 18734 ] my parent pid 18731
[process 18733 ] running
[process 18733 ] 2
[process 18735 ] running
[process 18735 ] 1
[process 18735 ] my parent pid 18732
[process 18732 ] running
[process 18732 ] 2
[process 18736 ] running
[process 18736 ] 2
[process 18736 ] my parent pid 18733
[process 18735 ] running
[process 18735 ] 2
[process 18737 ] running
[process 18737 ] 2
[process 18737 ] my parent pid 18732
[process 18738 ] running
[process 18738 ] 2
[process 18738 ] my parent pid 18735
```

```
[parent process] my pid is 25104
[process 25104 ] running
[process 25104 ] 0
[process 25104 ] running
[process 25104 ] 1
[process 25105 ] running
[process 25105 ] 0
[process 25105 ] my parent pid 25104
[process 25106 ] running
[process 25106 ] 1
[process 25106 ] my parent pid 25104
[process 25104 ] running
[process 25104 ] 2
[process 25107 ] running
[process 25105 ] running
[process 25105 ] 1
[process 25107 ] 2
[process 25107 ] my parent pid 25104
[process 25108 ] running
[process 25108 ] 1
[process 25108 ] my parent pid 25105
[process 25106 ] running
[process 25106 ] 2
[process 25105 ] running
[process 25105 ] 2
[process 25109 ] running
[process 25109 ] 2
[process 25109 ] my parent pid 25106
[process 25108 ] running
[process 25110 ] running
[process 25108 ] 2
[process 25110 ] 2
[process 25110 ] my parent pid 25105
[process 25111 ] running
[process 25111 ] 2
[process 25111 ] my parent pid 25108
```

Comparons

# Résumé

- **Un processus représente l'exécution d'un programme**
- **Il s'exécute sur le processeur**
  - pour lequel il est en compétition avec les autres processus
  - on ne peut prévoir exactement quand il est vraiment en exécution
- **Un processus a un état**
  - élu (en exécution), prêt (en attente du CPU), bloqué
- **Les processus sont gérés par le système d'exploitation**
  - qui sait tout sur eux
  - qui peut créer de nouveaux processus, de changer l'état, de tuer
- **Le lancement des processus**
  - passe via l'interface système

# Compléments

- **Lancer un processus en ligne de commande**
  - en premier plan (*foreground*) : sans & à la fin

```
[~] /Applications/Firefox.app/Contents/MacOS/firefox
```

le shell attend la terminaison
de ce programme et donc n'interagit pas
(pas possible de taper des commandes)

  - en arrière plan (en tâche de fond, *background*) : avec & à la fin

```
[~] /Applications/Firefox.app/Contents/MacOS/firefox &
[2] 30142
[~] ps
  PID TTY           TIME CMD
64535 ttys000    0:00.04 /Applications/iTerm.app/Contents/MacOS/iTerm2 --
64537 ttys000    0:00.06 -bash
79811 ttys001    0:00.19 -bash
81212 ttys002    0:00.04 -bash
17367 ttys003    0:00.07 -bash
29203 ttys003    0:00.03 /usr/local/Cellar/python@2/2.7.16/Frameworks/Py
30142 ttys003    0:04.08 /Applications/Firefox.app/Contents/MacOS/firefo
30148 ttys003    0:01.02 /Applications/Firefox.app/Contents/MacOS/plugin-
30149 ttys003    0:03.11 /Applications/Firefox.app/Contents/MacOS/plugin-
30155 ttys003    0:00.19 /Applications/Firefox.app/Contents/MacOS/plugin-
18378 ttys004    0:00.04 -bash
```

# Compléments lancement processus

- **Lancer des processus en séquence : avec ;**

```
[~/Enseignement/DIU_EIL] ls;ls;ls
1_intro     2_processus pres
1_intro     2_processus pres
1_intro     2_processus pres
```

- **Lancer des processus en parallèle : avec |**

```
ls | grep 1
```

```
[~/Enseignement/DIU_EIL] ls | grep 1
1_intro
```

  - Création de tubes qui sont des canaux de communication entre processus. Redirection des entrées/sorties.

# Redirection d'entrées/sorties des processus

- **Rediriger la sortie d'un processus vers un fichier** >

```
[~/Enseignement/DIU_EIL] ls > ls.txt
[~/Enseignement/DIU_EIL] cat ls.txt
1_intro
2_processus
ls.txt
pres
```

- **Rediriger l'entrée d'un processus depuis un fichier** <

```
[~/Enseignement/DIU_EIL] grep 1 < ls.txt
1_intro
```

# Changer l'état des processus

```
[~/Enseignement/DIU_EIL] firefox
^Z
[2]+  Stopped                 firefox
[~/Enseignement/DIU_EIL] ps j
…
vania 32603 17367 32603      0    1 T    s003    0:04.50 firefox
[~/Enseignement/DIU_EIL] fg
firefox
```

Ctrl-Z interromp le processus

on peut le relancer en premier (`fg`)
ou arrière plan (`bg`)

```
^C
[GFX1-]: Receive IPC close with reason=AbnormalShutdown
[~/Enseignement/DIU_EIL] firefox &
[2] 33689
[~/Enseignement/DIU_EIL] kill 33689
```

Ctrl-C tue le processus

On peut tuer un processus
en utilisant la fonction système `kill`