**Gestion des processus et des ressources**

# 1 Processus et ordonnancement

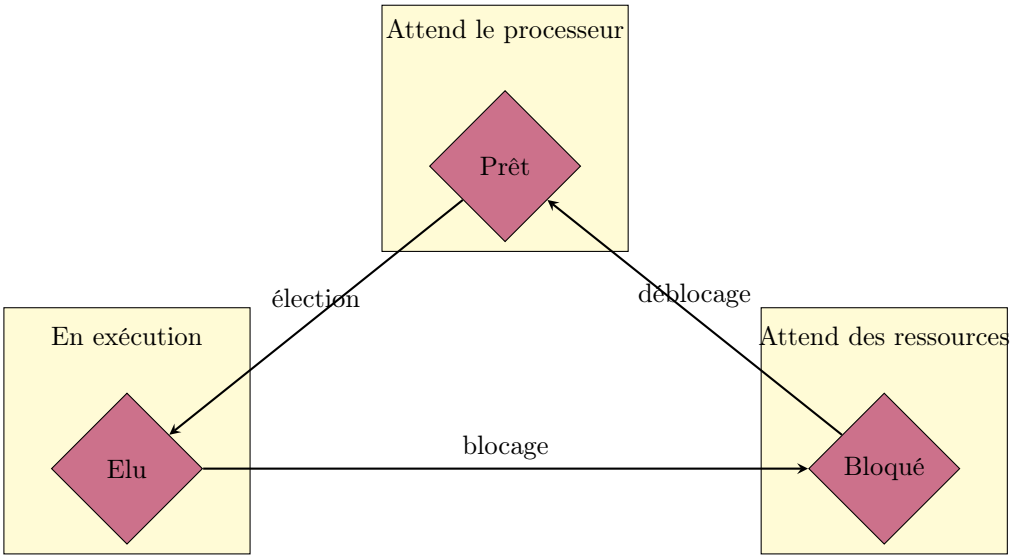
## 1.1 Notion de processus

L'année dernière vous avez appris les principes du modèle de von Neumann avec lequel un programme s'éxécute de manière séquentielle par le processeur jusqu'à sa terminaison. Pourtant lorsque vous utilisez votre ordinateur, vous pouvez en même temps avoir un navigateur web ouvert avec plusieurs onglets, écoutez de la musique et utiliser votre IDE Python préféré pour programmer votre prochain projet à rendre.
Cette exécution *concurrente* de *processus* est l'une des fonctionnalités des systèmes d'exploitation moderne. On parle de systèmes d'exploitation multitâches.

Définition

- Un **processus** est une instance d'exécution d'un programme.
- Deux processus s'exécutent de manière **concurrente** si les intervalles de temps entre le début et la fin de leur exécution ont une partie commune.

## 1.2 Les différents états d'un processus




# 2 Commandes Unix de gestion des processus

## 2.1 Les commandes `ps` et `top`

Dans les systèmes POSIX, la commande `ps` (*process status*) permet d'obtenir des informations sur les processus en cours d'exécution.

```
$ ps -a -u -x
```

Les options `-a`, `-u` et `-x` permettent respectivement d'afficher les processus de tous les utilisateurs (et pas uniquement les votre), le nom des utilisateurs et les processus lancés ailleur que depuis le terminal.
La commande affiche alors des informations sur les processus, comme par exemple :

```
USER       PID %CPU %MEM    VSZ   RSS TTY   STAT START   TIME COMMAND
root         1  0.0  0.0 225612  8920 ?     Ss   Dec05   0:03 /sbin/init splash
...
pascal    2525  4.5  4.3 4101096 706008 ?   Sl   Dec05 142:48 /usr/lib/firefox/firefox
pascal   20722  0.3  0.9 2969204 159756 ?   Sl   Dec06   6:51 texmaker
pascal   26945  0.0  0.0  40904  3632 pts/0 R+   18:11   0:00 ps -a -u -x
```

- `USER` indique le nom de l'utilisateur qui a lancé le processus.
- `PID` donne l'identifiant numérique du processus.
- `%CPU` et `%MEM` indiquent respectivement le taux d'occupation du processeur et de la mémoire pa rle processus.
- `STAT` indique l'état du processus, `S` pour *sleeping*, le processus est en attente et `R` pour *running*, le processus est dans l'état prêt ou élu.
- `COMMAND` indique la commande utilisé pour lancer le programme.
- `START` et `DATE` indiquent respectivement l'heure ou la date à laquelle le programme à été lancé et le temps cumulé d'exécution du processus.

La commande `top` permet de voir en temps réel des iformations similaire à celle de `ps`. Cette commande peut servir à déterminer quel processus occupent le plus le processeur ou utilise le plus de mémoire.

```
$ top
```

```
PID   USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM  TIME+    COMMAND
21044 pascal    20   0 3669456 768032 220892 S  43,0  4,7  35:34.35 Web Content
 2525 pascal    20   0 4212372 806024 312864 S  10,9  4,9 152:27.52 firefox
 1586 pascal     9 -11 2792428  21948  17244 S   7,0  0,1  32:47.90 pulseaudio
 2639 pascal    20   0 3131072 435844 205260 S   4,0  2,7  10:27.75 Web Content
 1159 root     -51   0       0      0      0 S   0,7  0,0  12:53.35 irq/137-nv+
 1746 pascal    20   0 3800000 310868  95660 S   0,7  1,9  24:45.39 cinnamon
27882 pascal    20   0   45456   4116   3396 R   0,7  0,0   0:00.31 top
 2118 pascal    20   0  487404  20796  17784 S   0,3  0,1  10:03.92 conky
    1 root      20   0  225612   8920   6816 S   0,0  0,1   0:03.68 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:00.04 kthreadd
```

### 2.2 La commande `kill`

La commande `kill` permet d'interrompre un processus dont on connait PID.

```
\$ kill 21044 2639
```

La commande précédente va mettre fin aux processus dont les PID sont 21044 et 2639.
L'option `-9` de la commande `kill` permet d'immédiattement terminé un processus sans que celui-ci n'exécute la moindre opération supplémentaire. Cette option est uniquemenent à utiliser lorsqu'une application se comporte de manière anarchique.

## 3 Programmation concurrente