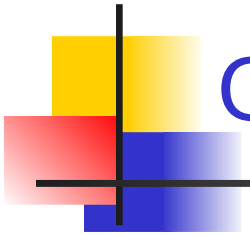
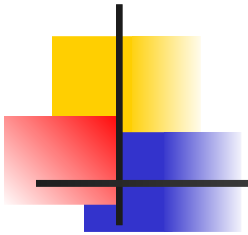


Luc Courtrai

CONCURRENCE

Université de Bretagne SUD
UFR SSI - Departement MIS



Plan

Notion de processus

Les appels système Unix

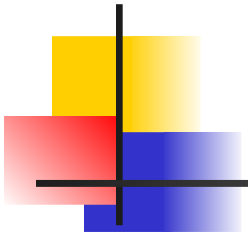
La synchronisation entre processus

La communication entre processus

Problème d'interblocage

Les processus legers : les threads JAVA

Les Posix threads



Concurrence

Plan

Notion de processus

Les appels système LINUX/UNIX

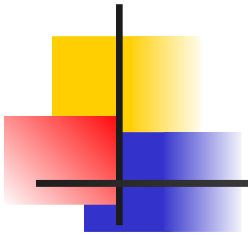
La synchronisation entre processus

La communication entre processus

Problème d'interblocage

Les processus légers : les threads JAVA

Les Posix threads

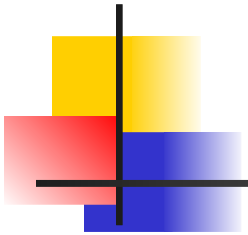


Notion de processus

Un processus représente l'activité résultante de l'exécution d'un programme.

Sous Unix un processus est un fichier binaire chargé et lancé

Un système d'exploitation doit permettre l'exécution de plusieurs tâches en même temps.



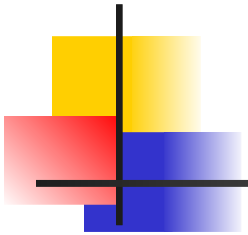
Notion de processus

Le parallélisme matériel:

Plusieurs processeurs,

cœurs,

architecture Hyper Threading



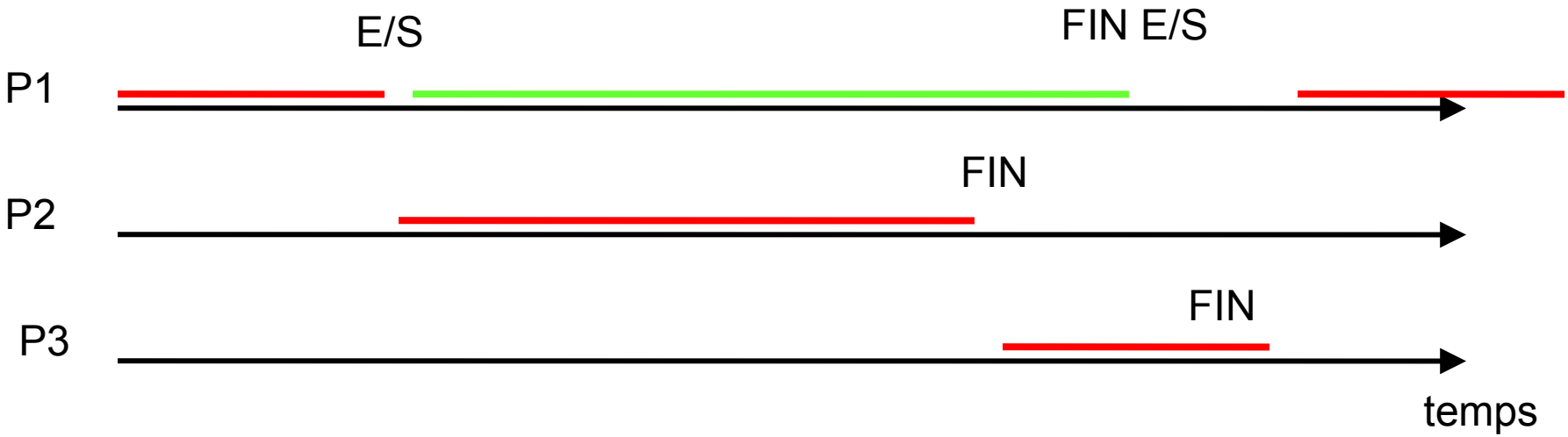
Notion de processus

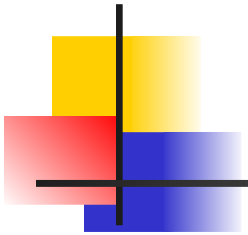
Le parallélisme matériel:

Les entrées-sorties sont gérées par des processeurs ou co-processeurs spécialisés (controleur disque, carte graphique..).

Pendant le temps de ces entrées-sorties le processeur ne fait rien. Il peut donc exécuter un autre processus.

Concurrence/processus





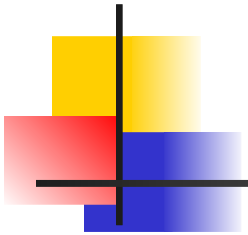
Concurrence/processus

inconvénients :

Le processus doit soit

- rendre explicitement la main
- effectuer une entrée sortie
- se terminer

pour qu'un autre processus puisse travailler.



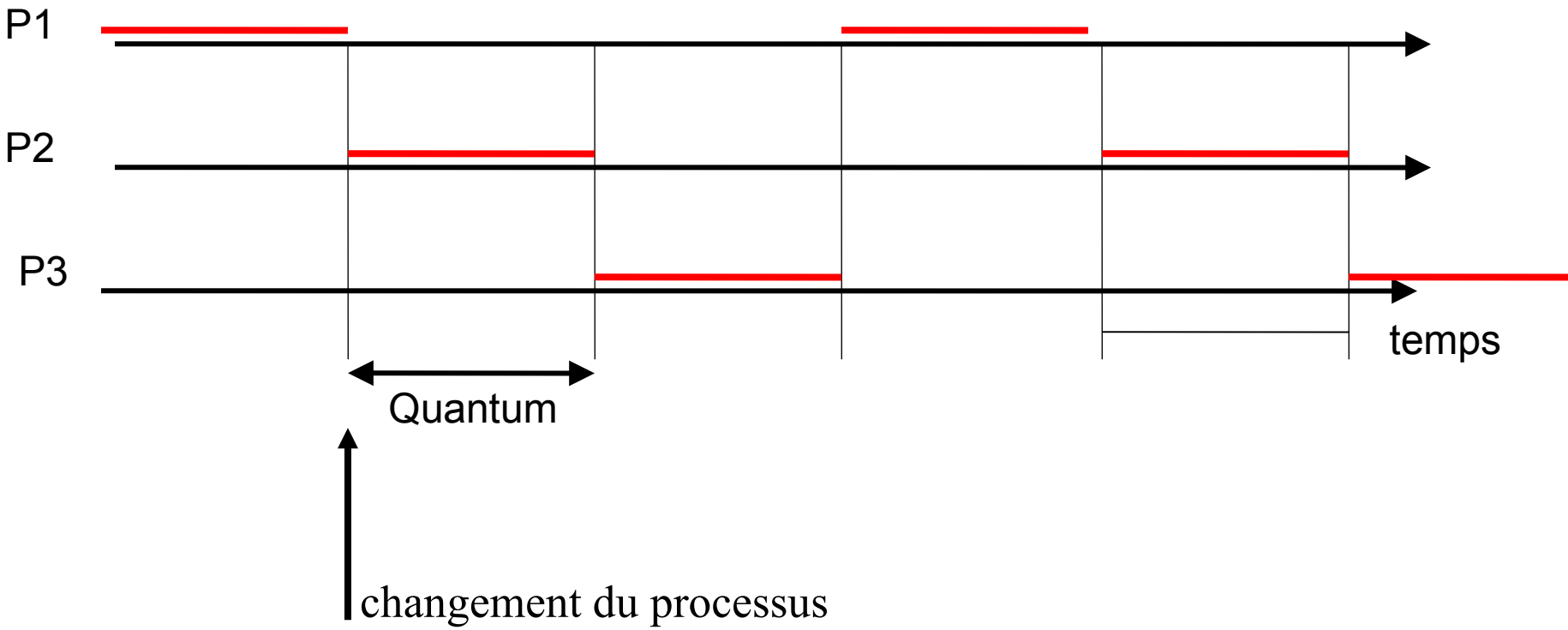
Concurrence/processus

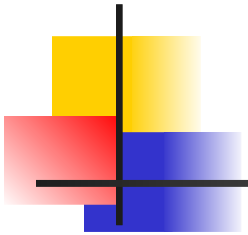
La multiprogrammation en temps partagé

“**time sharing**”

Le temps de travail du processeur est découpé en quantum (fraction de temps). Le processeur traite un processus pendant un quantum de temps puis passe à un autre processus pour le quantum suivant.

Concurrence/processus

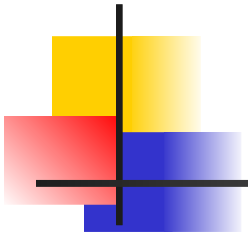




Concurrence/processus

A l'origine, les Os multi-taches en temps partagé ont été créés pour des ordinateurs multi-postes. Plusieurs terminaux (sans processeurs, vt100) sont reliés à un ordinateur "serveur de terminaux". Toutes les tâches sont effectuées par le processeur du serveur. Le terminaux ne gère que l'affichage et les saisies au clavier.

Si le quantum de temps est de l'ordre de 1/60 secondes. Si le système doit exécuter moins de 60 processus. Ils ont tous tourné (run) en moins d'une seconde. Les utilisateurs des terminaux ont donc l'impression d'avoir le processeur pour eux seuls.



Concurrence/processus

Le système UNIX gère de nombreux processus

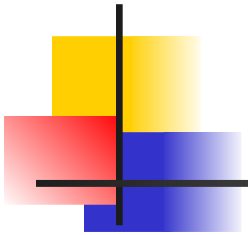
Les démons (connexion, disque, swapp, ttyinput …)

pour chaque commande (a.out ; ls | wc -l)

Le système en temps partagé permet d ’exécuter tous ces processus en pseudo-parallèle.

Pour un quantum de 1/60 seconde, un processeur 3Ghertz peut exécuter 50 10^6 cycles d ’instructions

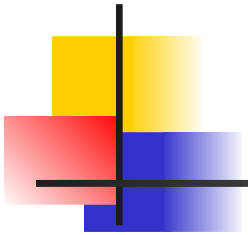
Un processus peut éventuellement passer plus souvent en fonction de sa priorité.



Concurrence/processus

Un processus est un programme en d'exécution. Lors de l'exécution chaque processus est associé à un contexte d'exécution. Celui-ci contient au moins :

- le compteur ordinal (le numéro de la case mémoire contenant la prochaine instruction à exécuter)
- un ensemble de registres dont l'accumulateur
- position dans la pile d'exécution
- adresse des espaces mémoires



Concurrence/processus

la commutation de contexte

Le processeur commute de processus en processus.

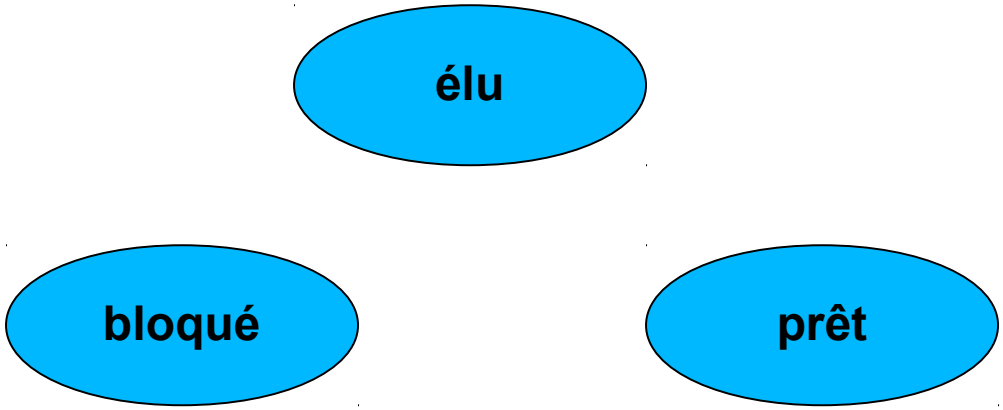
- sauvegarde le contexte du processus courant
- élection d'un nouveau processus
- chargement du contexte du processus
- exécution des instructions du processus à partir de son compteur ordinal.

La partie du code qui élit un processus est l'ordonnanceur.
"scheduler"

Concurrence/processus

L’état d’un processus

élu : en cours d ’exécution
bloqué :en attente de ressources
prêt : prêt à être élu



Concurrence/processus

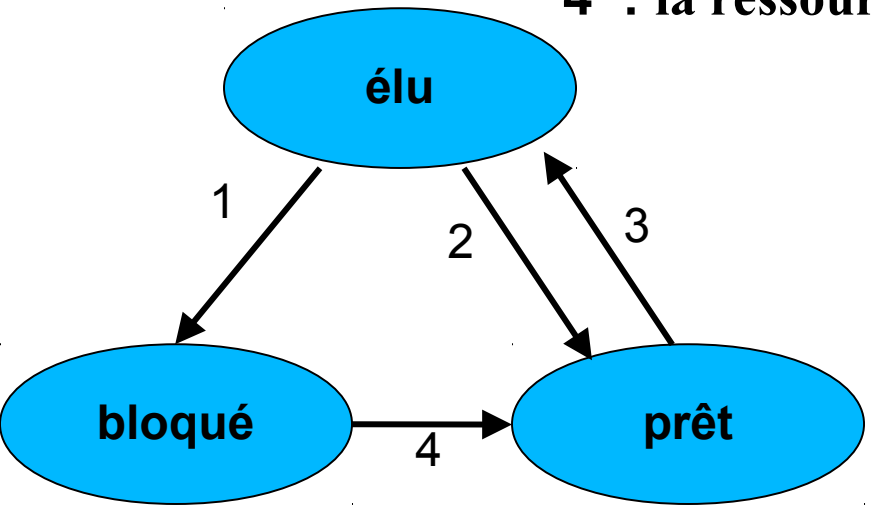
L'état d'un processus et les transitions.

1 : le processus se bloque en attente d 'une ressource

2 : l 'ordonnanceur retire le processus en exécution

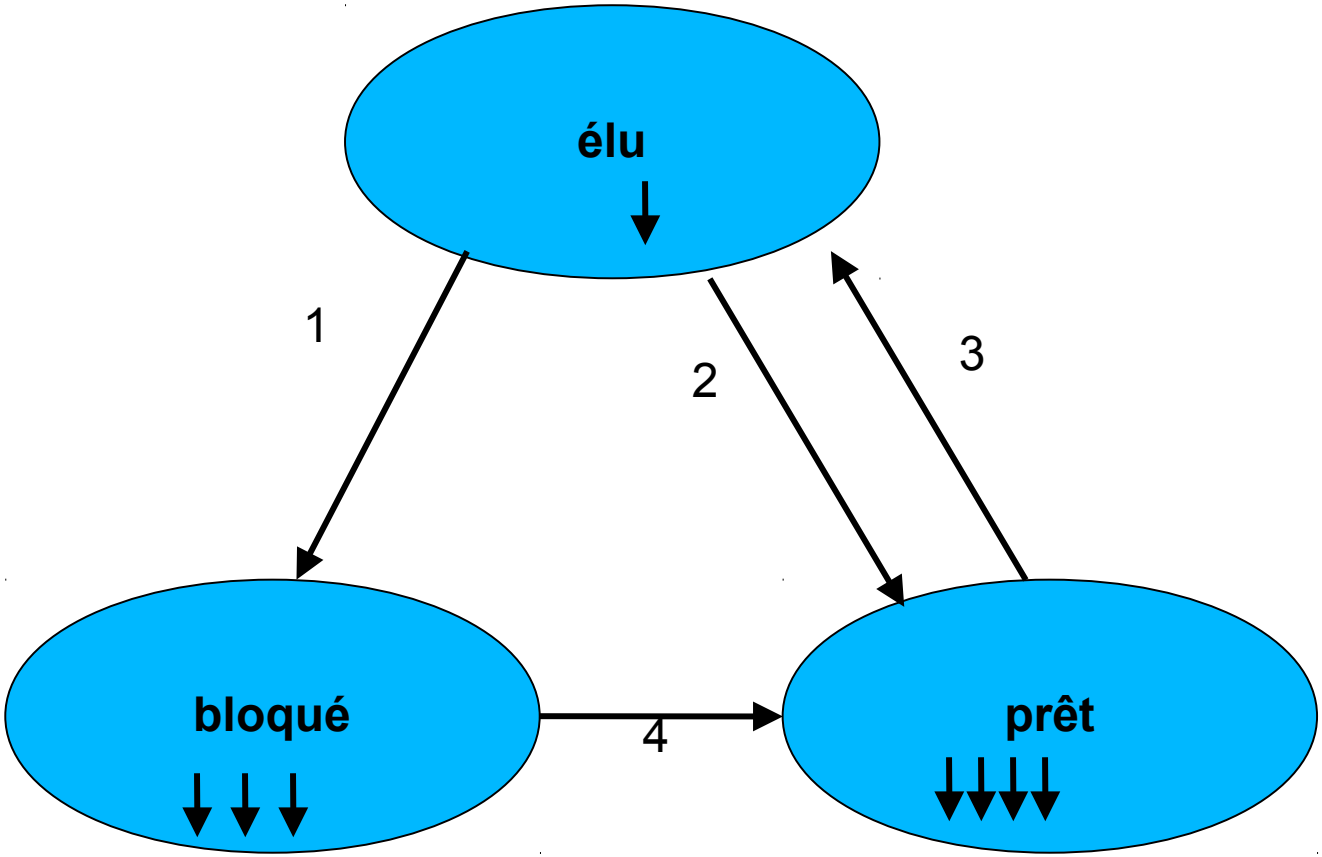
3 : l 'ordonnanceur élit le processus

4 : la ressource est disponible et affectée au processus



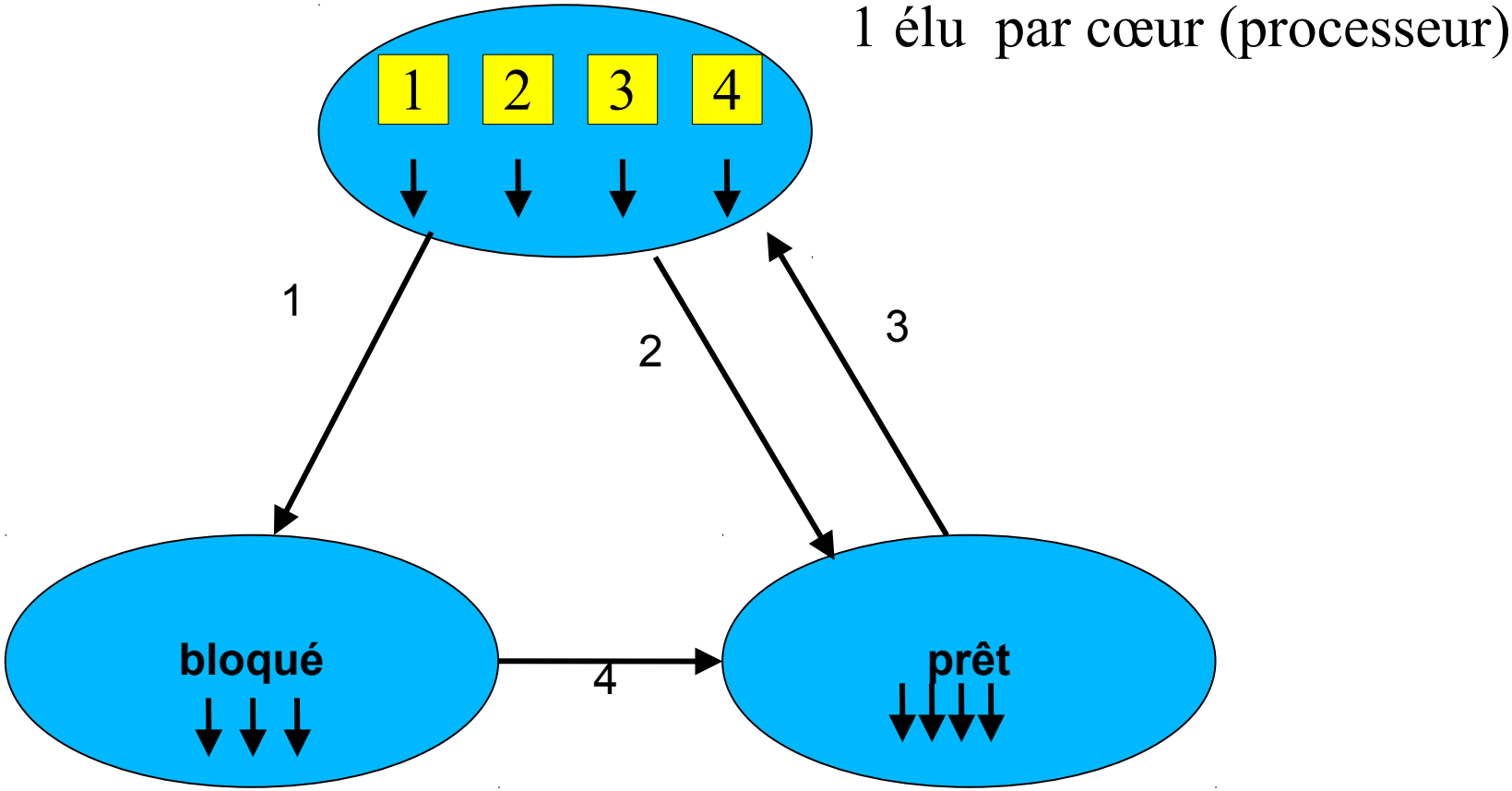
Concurrence/processus

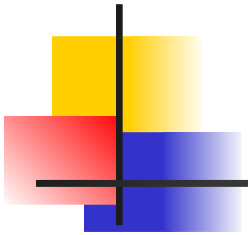
L'état d'un processus et les transitions.



Concurrence/processus

L'état d'un processus et les transitions.





Concurrence/processus

Le système gère une table des processus.

La table contient :

- l ’état du processus
- son contexte d’éxecution
- des stats (temps dans le cpu, age …)

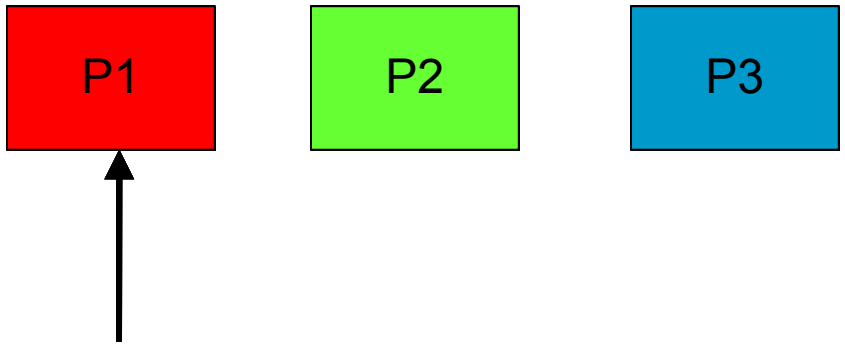
Ces infos servent à l’ordonnanceur pour sa politique d’élection.

Politique d'ordonnancement

Le Round-Robin (tourniquet)

Le système gère une file d'attente de processus prêts . Le premier de la liste est celui en execution.

A la fin de son quantum, l'ordonnanceur met le processus en fin de liste et charge le premier de la liste.

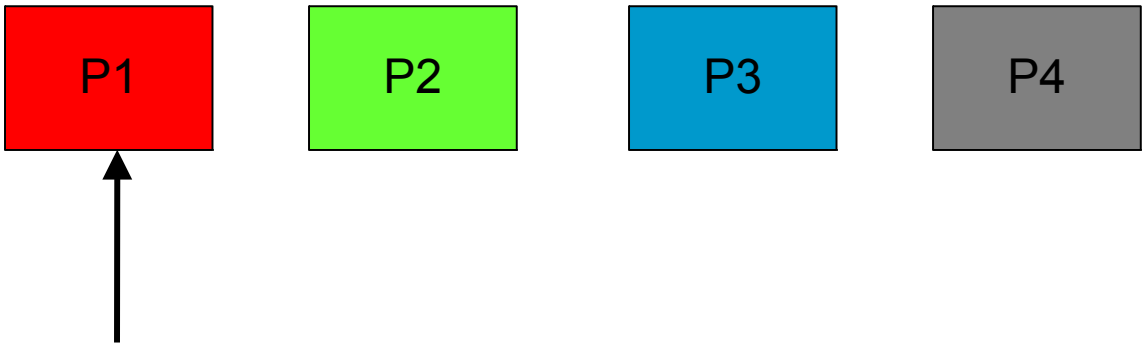


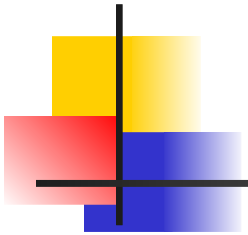
Concurrence/processus

Politique d ’ordonnancement

L’ajout de nouveau processus prêt s’effectue en fin de liste

Un processus en cours d ’exécution demandant une ressource sort de cette liste des prêts





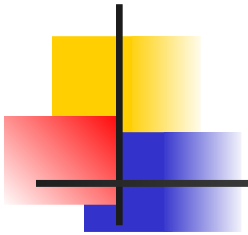
Concurrence/processus

Exemple UNIX/LINUX

Un processus UNIX est un fichier binaire en exécution.

Le format a.out (remplacé par elf)

- entête qui décrite la suite du fichier
- la taille de la section pour les variables non initialisées (tas)
- la section Texte (segment de texte) binaire en assembleur
- la section des variables initialisées (variables globales du C) segment data
- Eventuellement : la table des symboles pour le debug



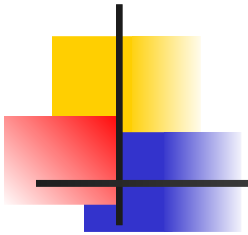
Concurrence/processus

Exemple UNIX/LINUX

Au chargement du fichier binaire

- alloue et charge par copie les segments de texte de data
- alloue les segments de pile et de données non initialisées
- alloue un descripteur de processus (dans la table)

A l'exécution le programme (texte) pourra étendre les segments de data non initialisées ainsi que le segment de pile.



Concurrence/processus

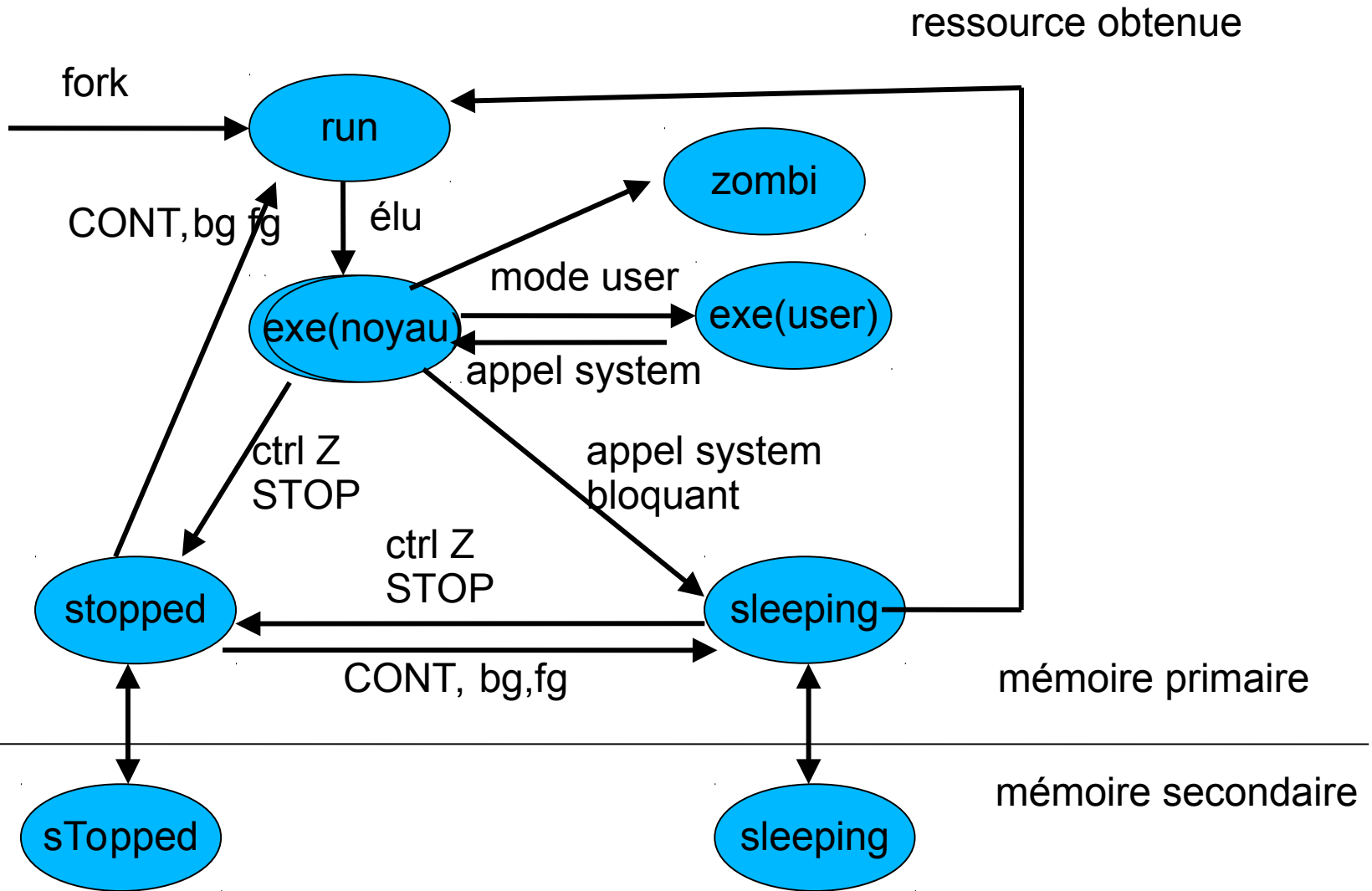
Exemple UNIX

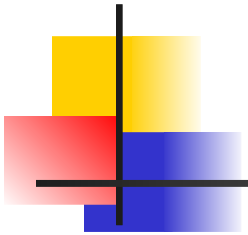
Le contexte des processus :

- état
- compteur ordinal
- valeur des registres actifs
- entrée dans la table des processus (numéro ou pointeur)
- sommet de pile
- adresse des 3, 4 segments (dans les registres)

Concurrence/processus

L ’état d’un processus





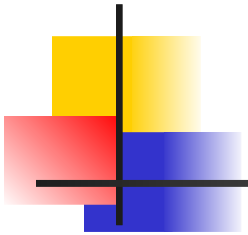
Concurrence/processus

L ’état d’un processus

Un processus en cours d’exécution passe en mode noyau pour effectuer les appels système.

Un processus est créé par un appel système fork effectué par un autre processus (le nouveau processus est dans l ’appel système fork et sortira de cet appel lorsqu’il sera élu.)

Un processus reste dans l ’état zombie jusqu’à ce que son père ait pris connaissance de la fin de ce processus fils.



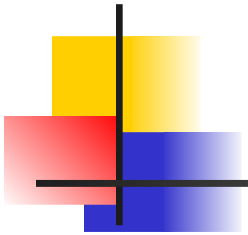
Concurrence/processus

L ’ordonnanceur : implémentation BSD

L ’ordonnancement des processus s’effectue en temps partagé avec différentes files.

Une partie des processus noyau sont non-intéruptibles. Il n ’y a pas de temps partagé pour eux. Ce sont eux qui rendent explicitement la main à l’ordonnanceur.

Si un événement apparaît qui met un de ces processus noyau à l’état prêt, le processus utilisateur courant est arrêté, même si il n’a pas terminé sur quantum de temps. L ’ordonnanceur donne alors la main au processus noyau. (Le processus utilisateur est dit **préempté**)



Concurrence/processus

L ’ordonnanceur : implémentation BSD

Le système gère plusieurs files d’attente correspondant à des niveau de priorités différentes.

Les niveaux les plus hauts sont réservés aux processus du noyau

Les processus utilisateurs sont mis dans des files de bas niveau.

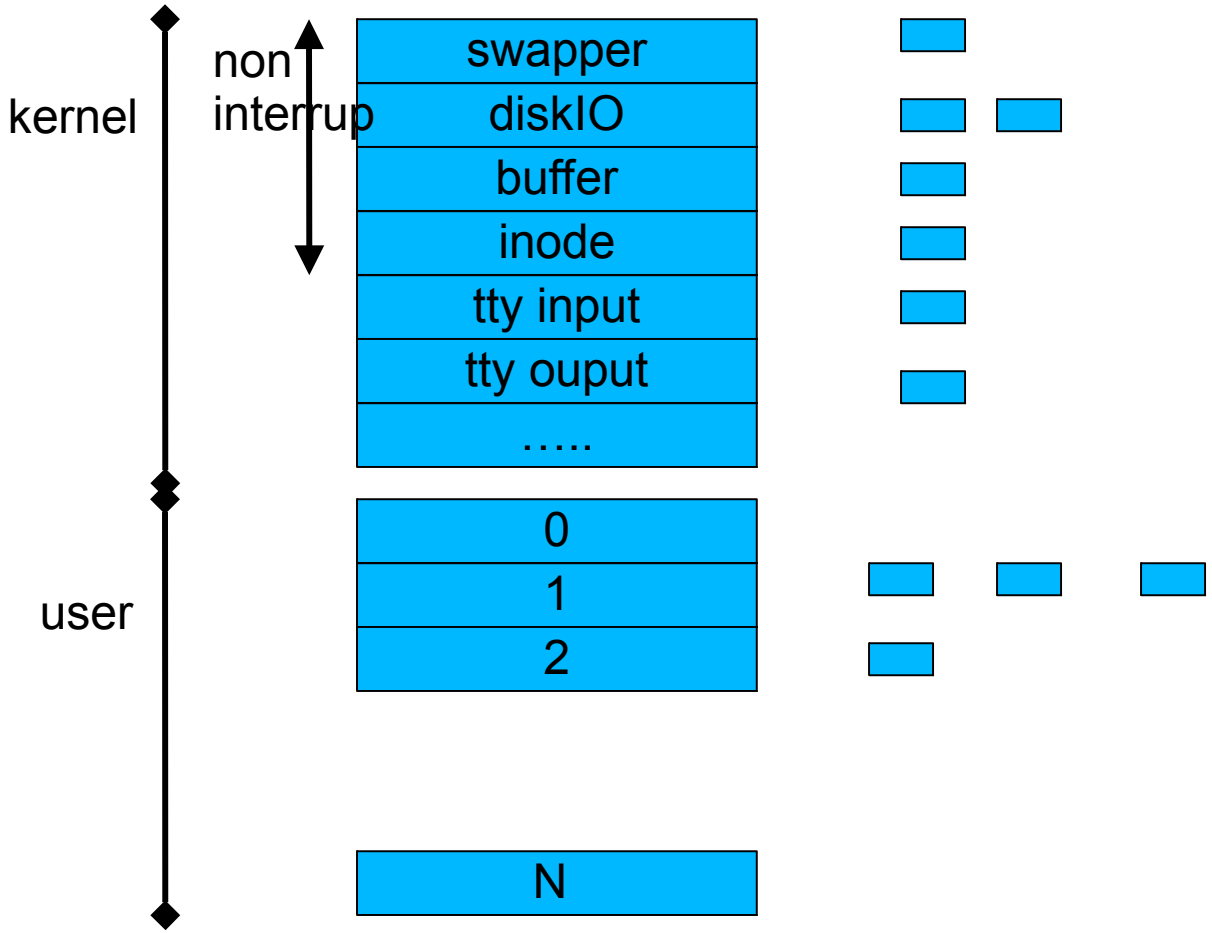
Un processus ne peut s’exécuter que si il n’a aucun processus de plus haut niveau dans les files.

Au sein d’une même file, le système utilise une politique de round robin.

Le système recalcule les priorités des processus pour faire remonter des vieux processus de plus faible prioritaire.

Concurrence/processus

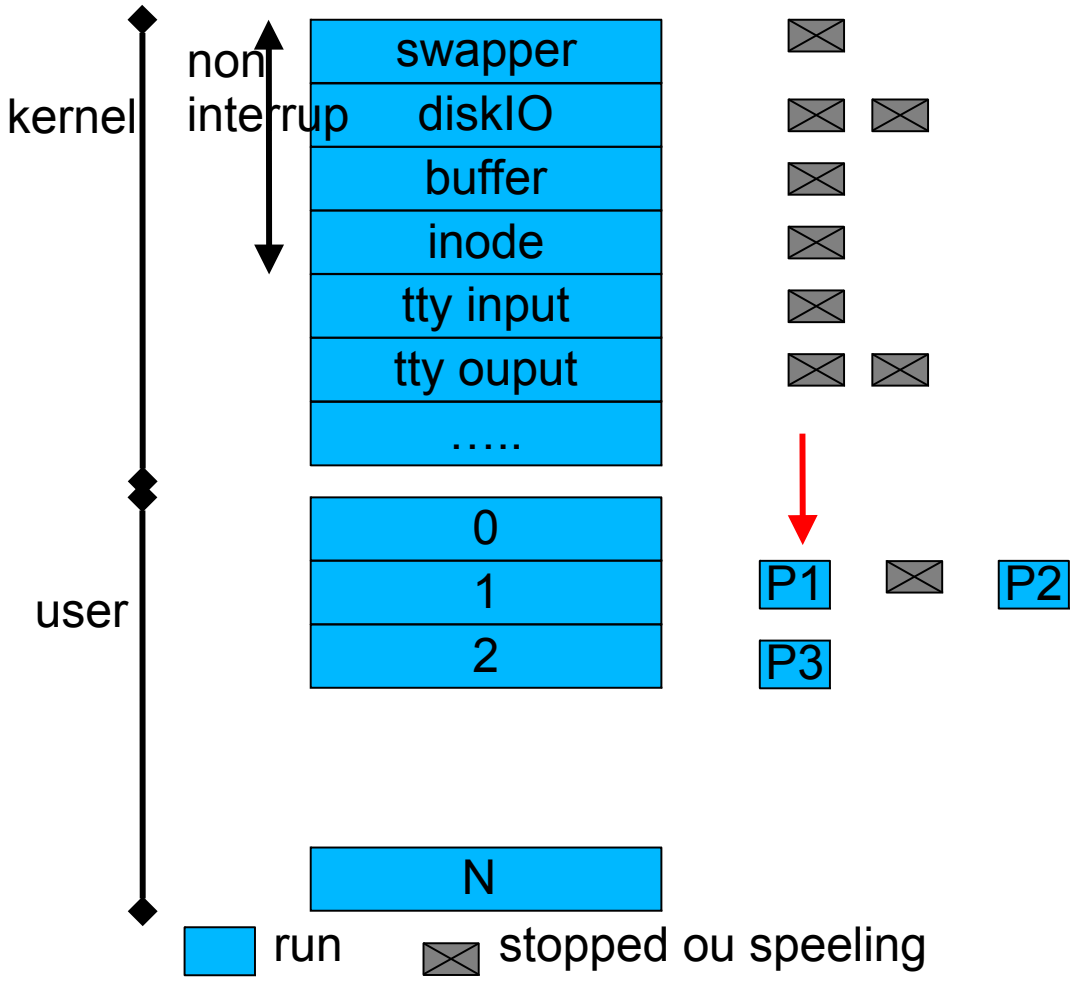
L ’ordonnanceur



Concurrence/processus

L ’ordonnanceur

L ’ordonnanceur choisit le processus éligible de la file la plus haute.



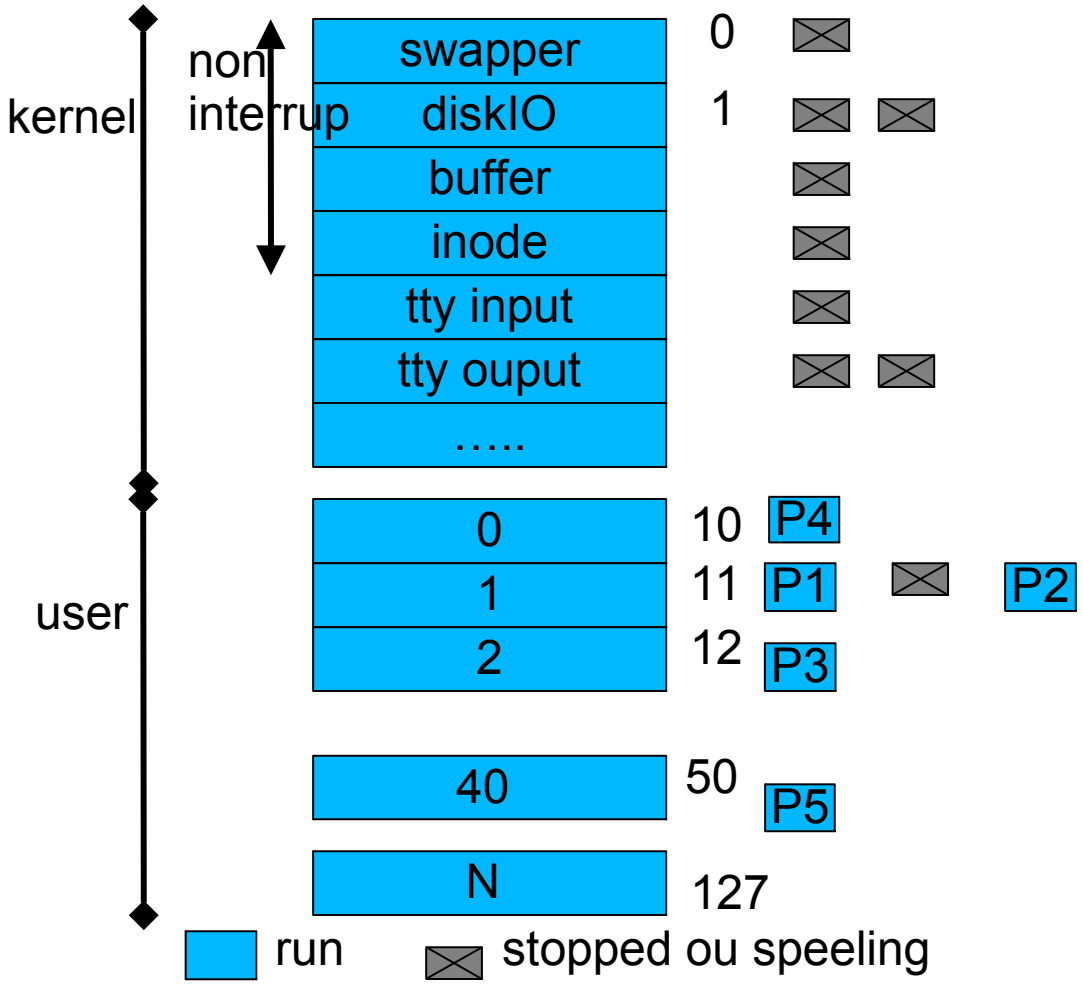
Concurrence/processus

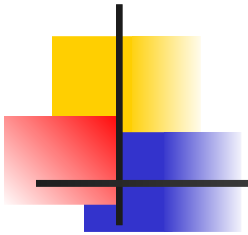
L ’ordonnanceur

Par départ un processus est mise dans une file:
l = PCLASSE + 2 * nice
PCLASSE : swapp 0 , inode 10; user 50,
nice priorité de départ (-20 19) défaut 0 (seul root nice < 0).

P4 est nouveau USER root avec une priorité de -20
l = 50 + 2*-20 -> 10

P5 est USER normal priorité 0
l = 50 + 2*0 -> 50





Concurrence/processus

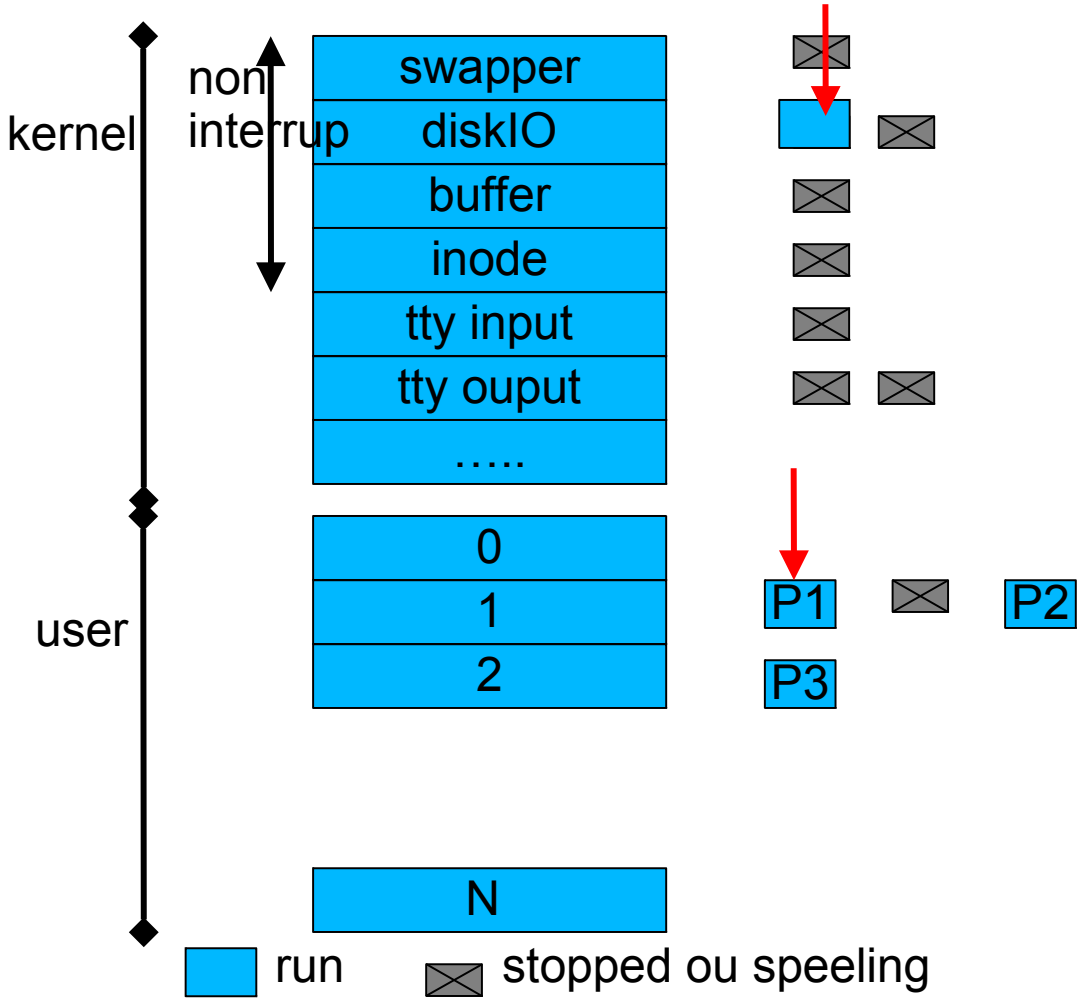
L ’ordonnanceur

```
>   ps -l
  F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY
    TIME CMD
000 S   261  5137  5134  0  69   0    -   620 rt_sig pts/1    00:00:00 tcsh
000 T   261  5238  5137  0  68   0    -   268 do_sig pts/1    00:00:00 a.out
000 R   261  5239  5137  0  73   0    -   683 -      pts/1    00:00:00 ps
```

Concurrence/processus

L ’ordonnanceur

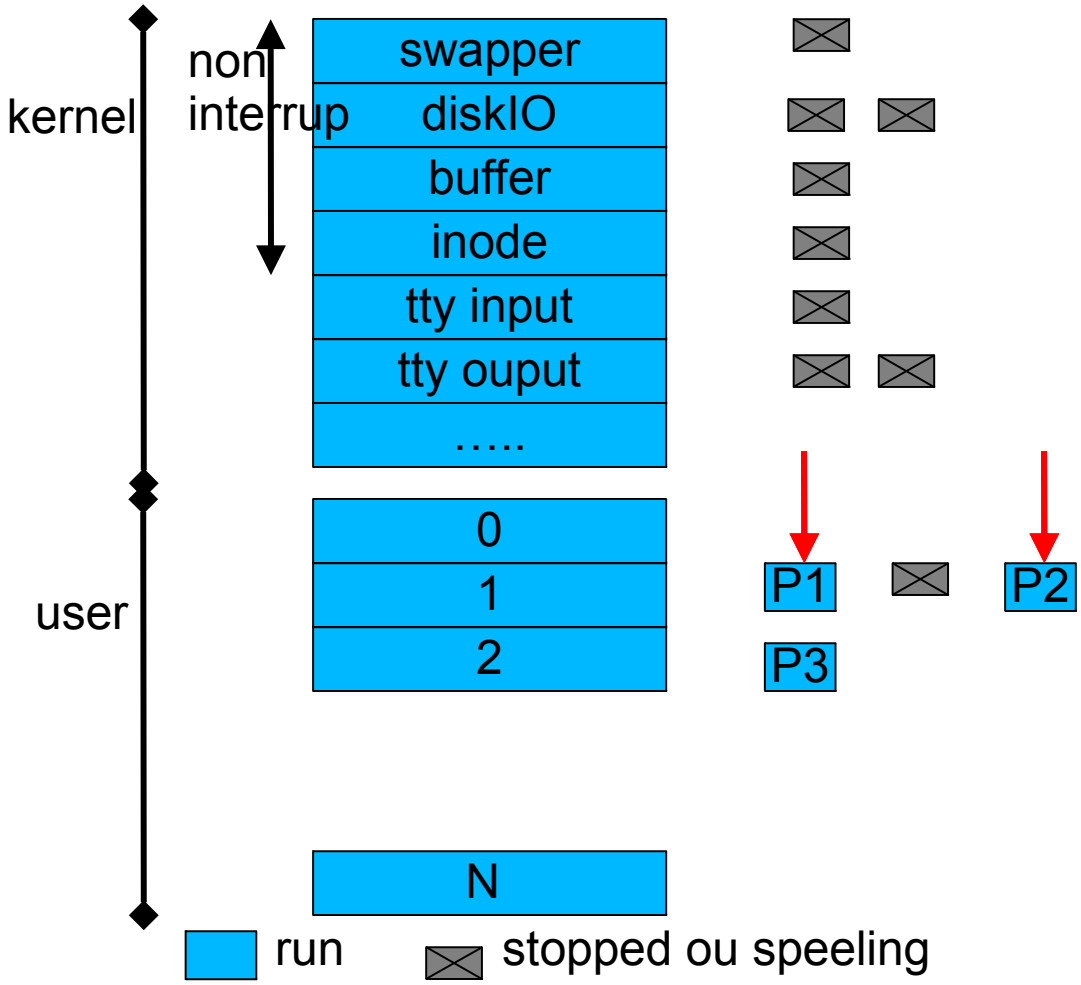
Si un processus de haut niveau est réveillé. Le processus courant est préempté.

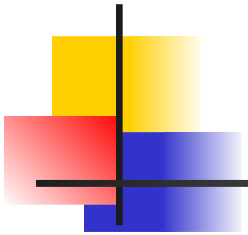


Concurrence/processus

L ’ordonnanceur

Au sein d ’une même file, on utilise du round-robin





Concurrence/processus

L ’ordonnanceur

Problème de famine.

Pour ne pas pénaliser les processus moins prioritaires,

le système recalcule le numéro de file d’un processus régulièrement.

num = PCLASSE + (Pcpu / 4) + 2 * nice

- Pcpu consommation CPU du processus (top horloge)

- PCLASSE swapp 0 , inode 10; user 50,
- nice priorité de départ (-20 20) défaut 0 (seul root nice < 0).

Concurrence/processus

L ’ordonnanceur

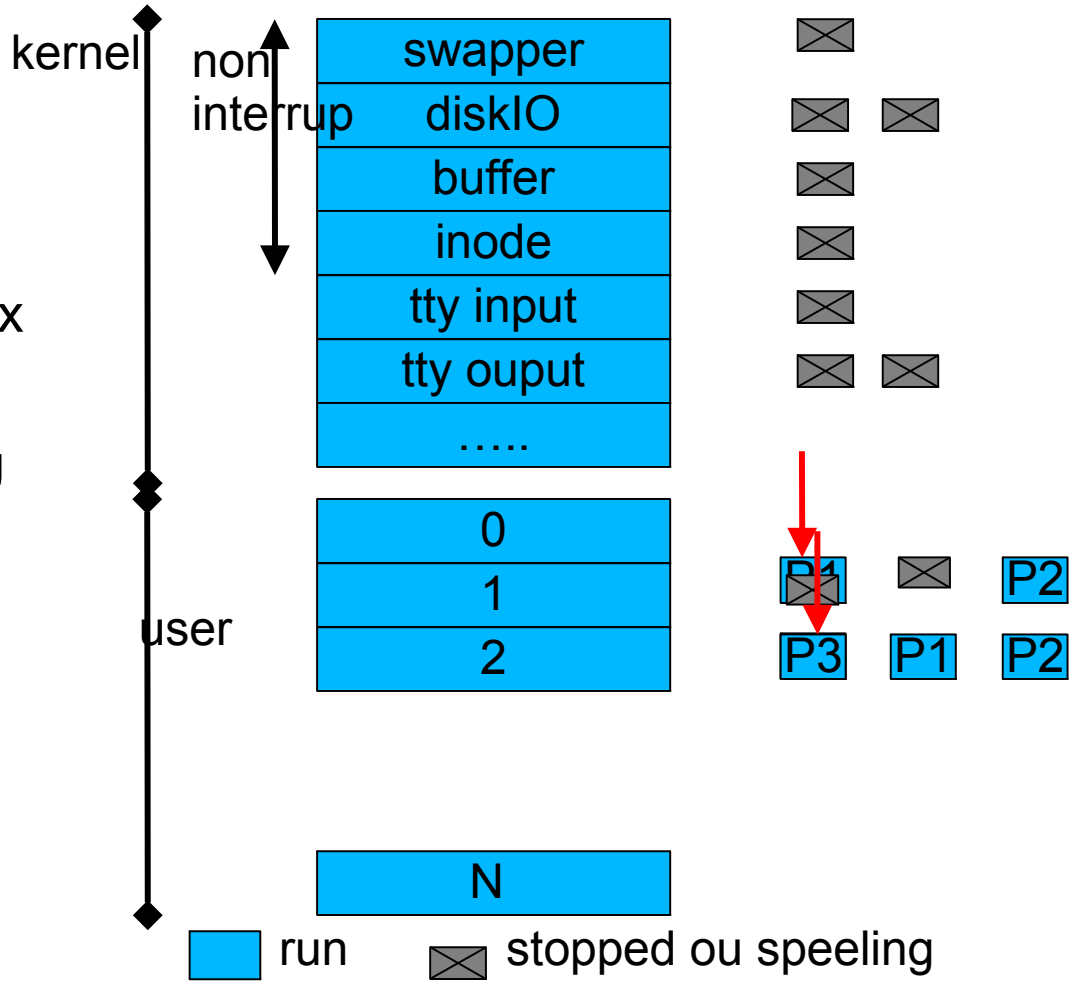
L= PCLASSE + (Pcpu / 4) + 2 * nice

Au départ P1 et P2 sont nouveaux
L = 50 + 0/4 + 2*0 -> 50

P1 et P2 sont restés dans la CPU donc leur Pcpu a augmenté chacun 4 clicks d ’horloge

ex = 50 + 4/4 + 2*0 -> 51

P3 a une chance de s’exécuter



Concurrence/processus

L ’ordonnanceur

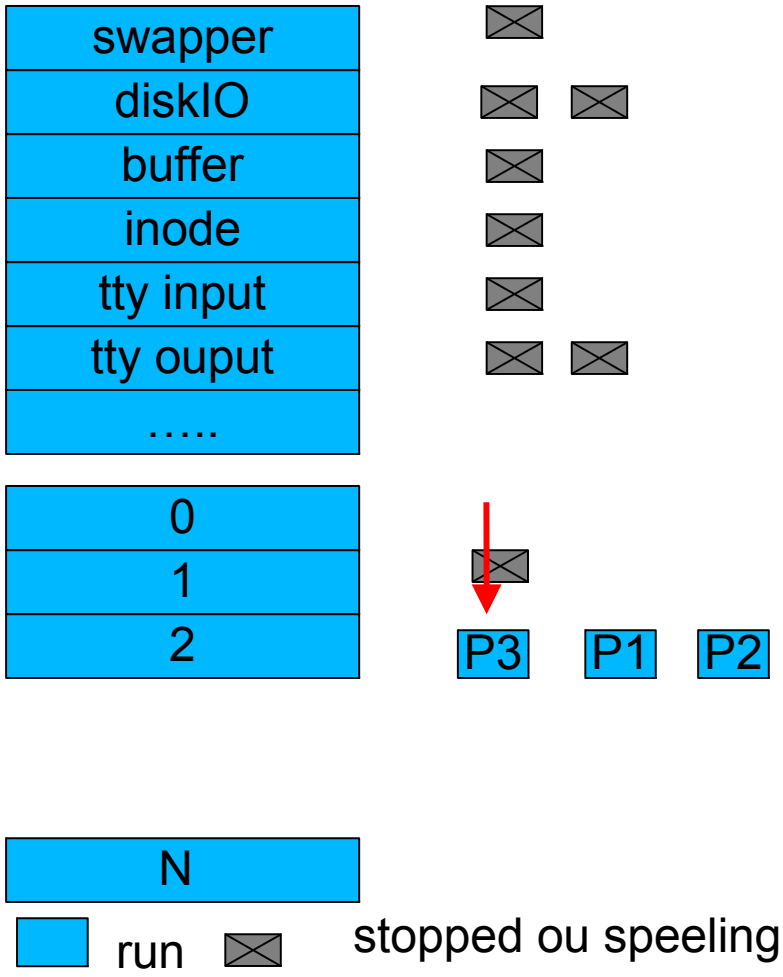
Pcpu est recalculé toutes les minutes en fonction du nombre de processus actifs dans la minute pour atténuer les Pcpu importants des vieux processus

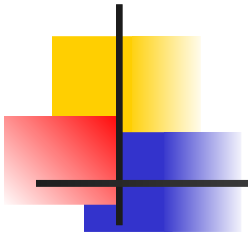
Pcpu = (2*load) / (2*load +1) *Pcpu +Pnice

avec load nombre moyen de processus actif dans la minute.

Si il y a peu de processus actifs ex 2
coeff de 2/3 -> 0.666
(les vieux processus remontent)

A l’inverse en cas de forte charge ex 60
coeff de 120/121 -> 0.999999
(les vieux processus restent dans leur file)

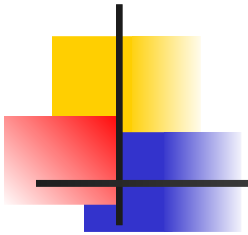




Concurrence/processus

L ’ordonnanceur

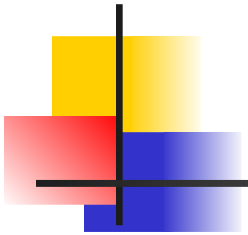
```
>    ps -l
  F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY
    TIME CMD
000 S   261  5137  5134  0  69   0    -   620 rt_sig pts/1    00:00:00 tcsh
000 T   261  5238  5137  0  68   0    -   268 do_sig pts/1    00:00:00 a.out
000 R   261  5239  5137  0  73   0    -   683 -      pts/1    00:00:00 ps
>    ps -l
F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY
    TIME CMD
000 S   261  5137  5134  0  76   0    -   620 rt_sig pts/1    00:00:00 tcsh
000 T   261  5238  5137  0  63   0    -   268 do_sig pts/1    00:00:00 a.out
000 R   261  5241  5137  0  70   0    -   683 -      pts/1    00:00:00 ps
```



Concurrence/processus

L ’ordonnanceur Linux multicoeur

- une "run queue" par coeur
- réequilibrage des coeurs toutes les 200ms
  - calcul de la charge de chaque coeur
  - déplacement des processus vers le coeur le moins chargé
    (PB: coût, cache ...)

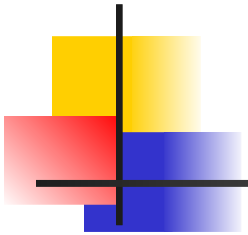


Concurrence/processus

L ’ordonnanceur

> ps x -o pid,pcpu,user,args,psr

```
  PID %CPU USER     COMMAND                   PSR
 2143  0.0 courtrai /lib/systemd/systemd --user    2
 2144  0.0 courtrai (sd-pam)                      1
 2422  0.0 courtrai /usr/bin/pulseaudio --start    3
 2426  0.0 courtrai /usr/lib/pulseaudio/pulse/g    3
11746  0.0 courtrai /usr/bin/gnome-keyring-daem    0
11748  0.0 courtrai /sbin/upstart --user           3
11831  0.0 courtrai upstart-udev-bridge --daemo    3
```

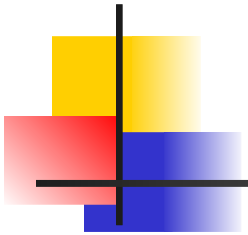


Concurrence/processus

La table des processus :

chaque entrée contient :

- état
- adresses (4 segments …. maps)
- UID
- PID,PPID
- un descripteur d'événement lorsque le processus est endormi
- Priorité
- vecteur des interruptions (ensemble de signaux reçus mais pas encore traités par le processus
- divers : compteur CPU p-cpu
- la zone u (utilisée lorsque le processus est dans la CPU)
-

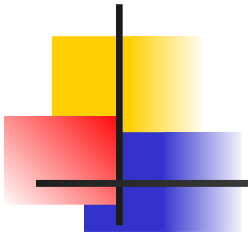


Concurrence/processus

La table des processus :

...la zone u (utilisée lorque le processus est dans la CPU)

- uid et gid effectif
- compteur de temps (user et system)
- terminal associé
- erreur (dernier code d'erreur dans les appels système)
- retour (code de retour du dernier appel système)
- E/S adresses de buffer
- . et /
- limites de taille de fichier et mémoire (ulimit CSH ou SH)
- umask

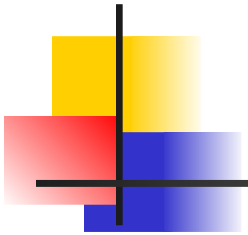


Concurrence/processus

Les interruptions

Une interruption est un événement produit par :

- le matériel (E/S, tty , horloge)
- un déroutement d'erreur du processeur (débordement calcul, division par zéro, violation de segment (produit un fichier core), image du processus en mémoire)
- un appel système demande E/S bloquante



Concurrence/processus

Les interruptions

L'interruption produit un signal qui est détecté par le système.

Il y a en général un changement de contexte par le système. Le processus courant est préempté, le système prend en compte l'interuption, éventuellement change l'état d'un processus et l'ordonnanceur élit un nouveau processus.

Il y a différents niveaux d'interruption (0 horloge,1 disque, 2 console,3 autre périphérique, 4 appel système, 5 autre).