230 Vie et états des processus

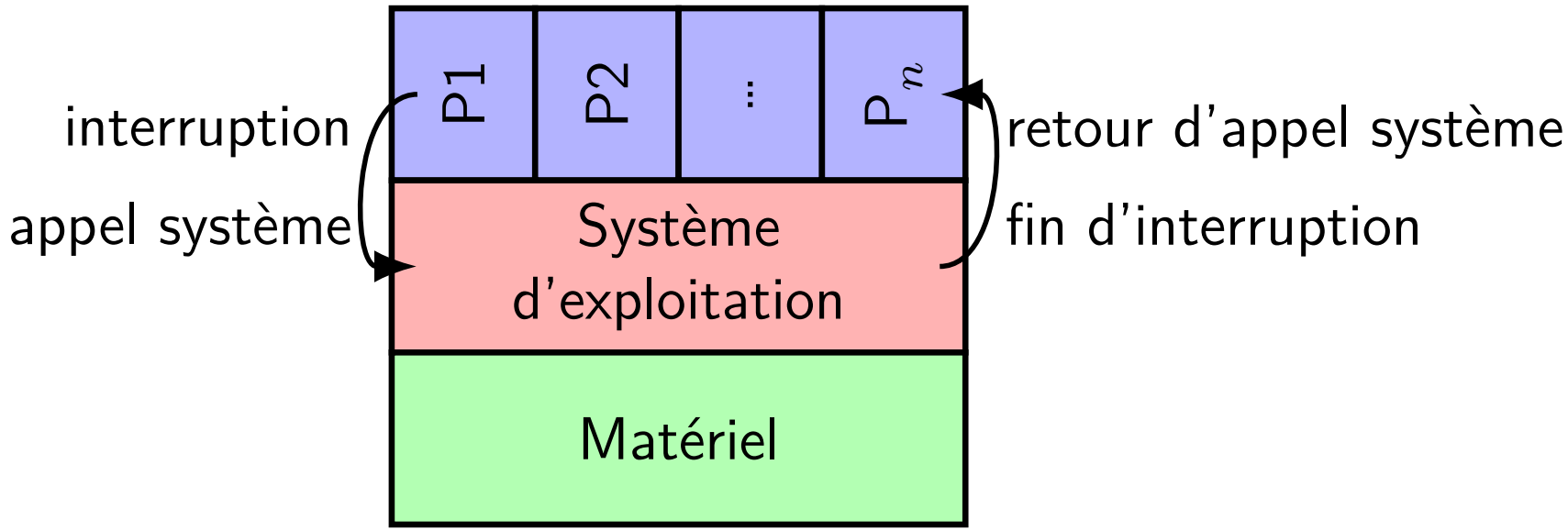
INF3173
Principes des systèmes d'exploitation

Jean Privat

Université du Québec à Montréal

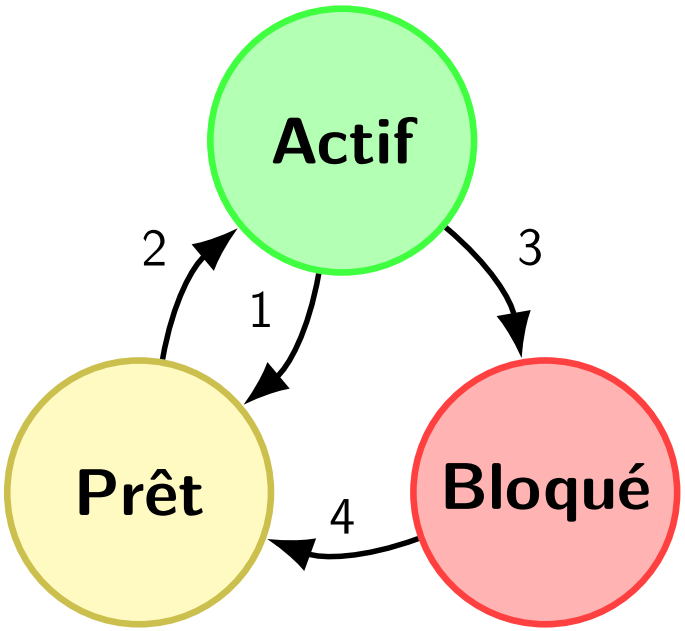
Hiver 2021

Processus et SE (Rappel)



- Un processus est actif sur le processeur
- Il fait un appel système (volontaire)
  OU une interruption matérielle survient (involontaire)
- Le processeur est passé au système (mode noyau)
- Le système traite l'appel système ou l'interruption
- Puis rend le processeur à un processus (mode utilisateur)

États d'un processus



- **Actif**: tient la ressource processeur
- **Prêt**: ne manque que le processeur
- **Bloqué**: manque une autre ressource

- 1 Le SE prend la main à un processus
- 2 Le SE donne la main à un processus
- 3 Le processus demande une ressource
- 4 La ressource demandée devient disponible

Questions

- Citer un exemple pour chaque changement
- Un processus peut-il passer de **prêt** à **bloqué** ?
- De **bloqué** à **actif** ?

États Unix

`ps(1)` sous Linux décrit des états un peu différents

- R s'exécute ou peut s'exécuter → Regroupe **prêt** et **actif**
- S/D en sommeil interruptible/ininterruptible
  → **Bloqué** sur un appel système.
  Si l'appel système a un code d'erreur `EINTR` l'appel est interruptible par un signal (`kill(1)`).
  Sinon, l'appel système fait des entrée-sorties mais termine vite (on voit rarement D apparaitre)
- T/t arrêté par le signal de contrôle de tâche/le débogueur
  → **Bloqué** par un signal spécial (`^Z`) ou par un débogueur
  Le processus peut continuer si on le débloque
- Z processus zombie (`defunct`) → On y reviendra...
- I fil inactif du noyau
  → Linux *utilise* l'infrastructure des processus pour gérer ses tâches noyaux internes. I n'a pas de sens autrement.

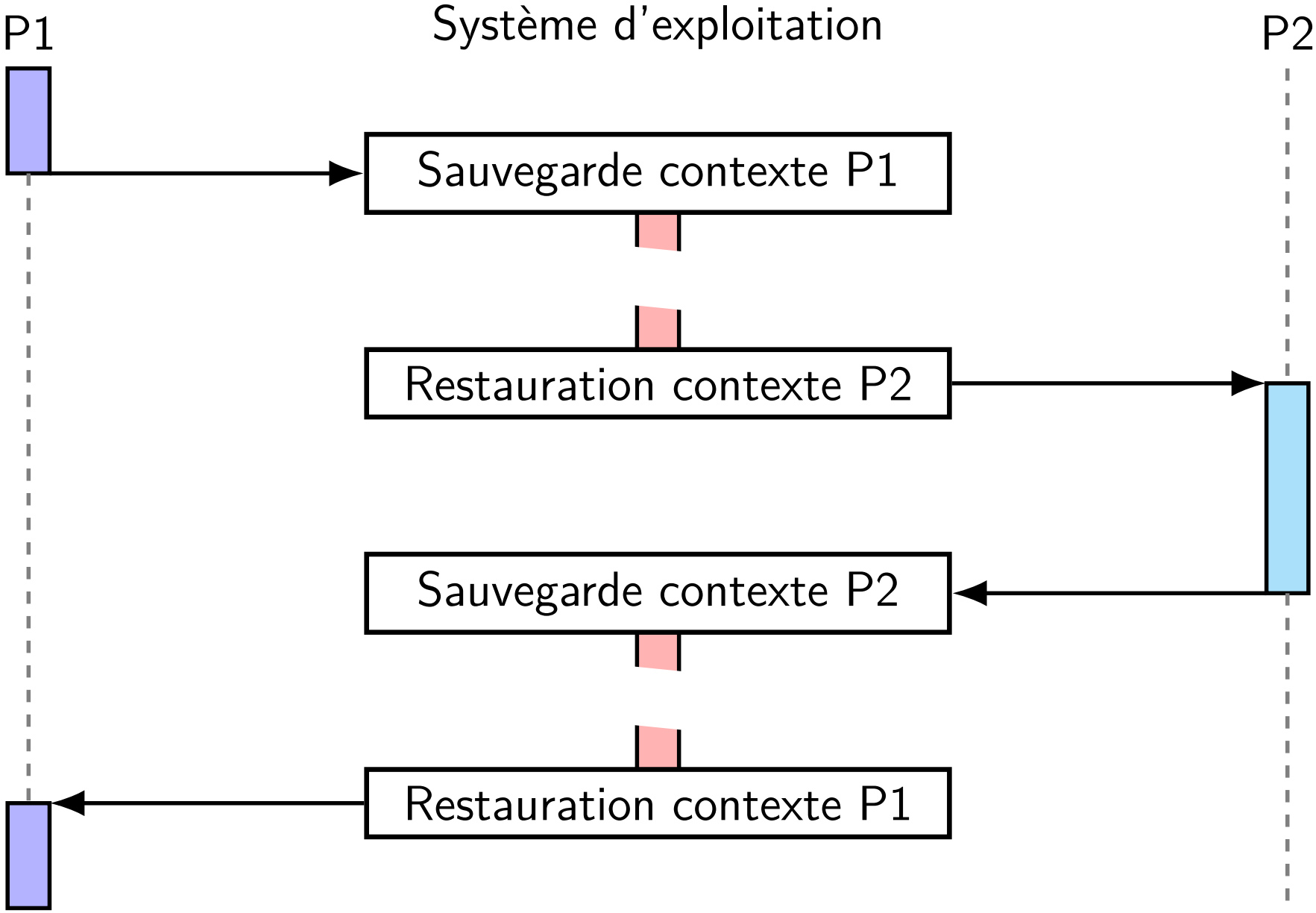
Changement de contexte

Un **changement de contexte** intervient quand le processeur est donné à un autre processus

- Passage du processus en mode noyau (syscall ou interruption)
- Sauvegarde du contexte du processus
- Exécution de l'appel ou gestion de l'interruption
- Modification éventuelle de l'état de processus
- Appel de l'**ordonnanceur** (*scheduler*) qui élit un processus
- Restauration du contexte de l'élu
- Modification de son état à **actif**
- Fin de l'appel ayant précédemment provoqué la suspension de l'élu
- Passage de l'élu en mode utilisateur et suite de son exécution

$\rightarrow$ C'est un procédé complexe et relativement coûteux

Changement de contexte



Coût du changement de contexte

Un changement de contexte coûte cher

- Rappel : le travail du SE n'est pas un travail utile en soi

Cause du coût

- Sauvegarde du contexte du processus
- Algorithme d'ordonnancement (détails plus tard)
- Restauration du contexte du processus
- Deux fois mise en défaut des caches et sections

Scénario (en quatre actes)

Personnages

- 3 processus (P1 **actif**, P2 et P3 **prêts**)
- Un système d'exploitation (accompagné de son ordonnanceur)

Acte 1

- P1 fait un `read(2)` avant l'épuisement de son quantum
- Passage en mode noyau et exécution de l'appel système `read`
- P1 passe à **bloqué**
- Sauvegarde du contexte de P1
- Appel de l'ordonnanceur

Questions

- Pourquoi `read` bloque le processus ?
- Est-ce que `read` bloque toujours le processus appelant ?

Scénario (suite)

Acte 2

- P2 est élu
- Le contexte de P2 est restauré
- P2 passe à **actif**
- L'horloge est programmée pour un nouveau quantum
- Fin de l'appel système/de l'interruption qui avait suspendu P2
- Et suite de l'exécution de P2 (en mode utilisateur)

Questions

- Pourquoi pas P1 ?
- Pourquoi P2 et pas P3 ?

Scénario (suite)

Acte 3

- P2 accapare le processeur
  - Ne fait pas d'entrée-sortie
  - Consomme entièrement son quantum de temps
- Interruption de l'horloge programmable
- Exécution du gestionnaire d'interruption (mode noyau)
- P2 passe à **prêt**, sauvegarde du contexte
- Appel de l'ordonnanceur

$\rightarrow$ Il y a eu **préemption**

Questions

- Est-ce que P2 pourrait être élu à nouveau ?

Scénario (suite)

Acte 4

- P3 est élu
  - Passe à **actif**
  - Son contexte est restauré
  - Suite de son exécution (en mode utilisateur)
- La donnée demandée par P1 arrive
  - Interruption du contrôleur de disque
- Exécution du gestionnaire d'interruption (mode noyau)
  - La donnée est placée en mémoire, accessible à P1
- P1 passe à **prêt** (on parle de réveil)
- P3 passe aussi à **prêt**, sauvegarde du contexte
- Appel de l'ordonnanceur

Question

- Pourquoi P3 passe à **prêt** plutôt que de continuer ?

Utilisation de la ressource processeur

Commandes et appels système

- `time(1)` décompte le total de ressources
  Utilisez `/usr/bin/time` pas la commande shell pour plus d'options
- `getrusage(2)` pour l'information en temps réel
  Le processus qui demande pour lui-même
- `ps(1)` et `top(1)` peuvent aussi présenter de l'information
- L'information est aussi dans `/proc/PID/stat` et `/proc/PID/status`
  Voir `proc(5)` pour les détails

Ressources CPU (de la commande time)

- `%E` Temps réel mis par le processus
  Heure de fin moins heure de début (en vraies secondes)
- `%U` Temps processus utilisateur utilisé
  Somme (pour chaque thread) du temps passé à l'état **actif**
- `%S` Temps processus système utilisé
  Somme du temps passé à l'état **actif** mais en mode noyau
  C'est à dire le travail fait par le SE au bénéfice du processus
- `%P` Pourcentage du processeur utilisé
  C'est juste (U+S)/E
- `%w` Nombre de changements de contextes volontaires
  Passages de **actif** à **bloqué**
- `%c` Nombre de changements de contextes involontaires
  Passages de **actif** à **prêt**

Question et exercices

- Où le noyau conserve le décompte de l'utilisation des ressources des processus ?
- Est-ce que %U peut être plus grand que %E ?
- Est-ce que %S peut être plus grand que %U ?
- Quelle est la valeur maximale de %P sur un système ?
- Comment avoir une grande ou une petite valeur de chacun des indicateurs (toutes choses étant égales par ailleurs) ?
- Cherchez l'équivalent des ressources dans `getrusage(2)`, `ps(1)`, `top(1)`, et `proc(5)`

Utilité des caches

La plupart des appels système d'entrée-sortie peuvent retourner sans bloquer le processus si l'information est disponible en cache.
Pour forcer le vidage de cache sous Linux:

```
$ sync
$ echo 3 | sudo tee /proc/sys/vm/drop_caches
```