Processus

Laurent Réveillère

Enseirb-Matmeca
Département Télécommunications

Laurent.Reveillere@ipb.fr
http://uuu.enseirb-matmeca.fr/~reveille/

Système UNIX

- Système mutli-tâches multi-utilisateurs
- Mécanisme préemptif
  - » Interruption autoritaire de la tâche en cours d'exécution pour passer la main à la suivante
  - ➔Évite des situations de blocage dans un programme utilisateur
- Droits d'accès
  - Spécifiques à chaque utilisateur
  - S'appliquent pour toutes les ressources (processus, fichiers, périphériques …)

Noyau UNIX

- Assure la gestion de la mémoire et le partage du processeur
  - Lancé au démarrage du système (*boot*)
    - → S'exécute jusqu'à la fin
  - Programme relativement petit
    - » Réside constamment en mémoire centrale
- Rôle principal
  - » Assurer une bonne répartition des ressources
  - » Requière un mode d'exécution privilégié (mode *noyau*)

Mode d'exécution

- Mode privilégié (mode noyau, *kernel mode*)
  - Accès à toutes les fonctionnalités du processeur
  - Uniquement pour le code du noyau
- Mode non privilégié (mode utilisateur, *user mode*)
  - Accès restreint au processeur et au matériel
  - Mode d'exécution de toutes les applications

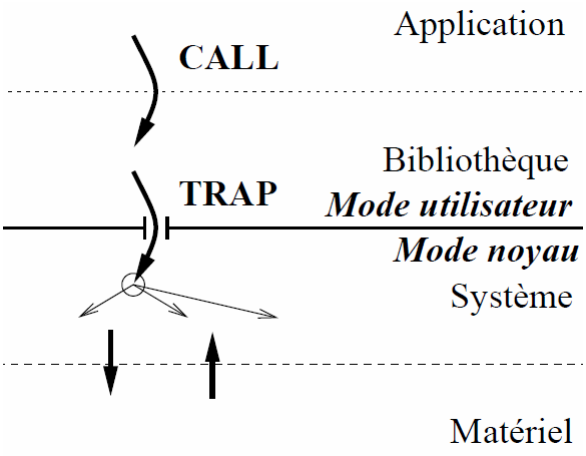
Gestion des modes

- Gestion du mode noyau et utilisateur au niveau du processeur
  - Presque tous les processeurs modernes
  - Différent de la notion de super-utilisateur qui existe dans certains SE
    - Géré au niveau logiciel (dans le noyau)
    - Un super-utilisateur passe le plus clair de son temps CPU en mode non privilégié (utilisateur) du CPU

Changement de mode

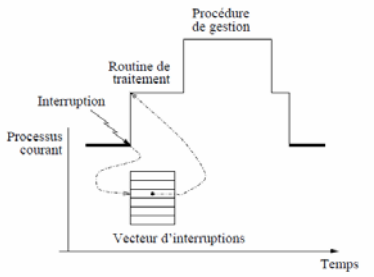
- Contexte
  - Application s'exécutant en mode utilisateur demande à accéder à une ressource géré par le noyau
- Mise en œuvre
  - Exécution d'un *appel système*
    - Passage du flot d'exécution au noyau
    - Vérification des privilèges
    - Exécution de la fonction correspondante par le code noyau

Appel système



Invocation d'un appel système

- Déroutement de l'exécution du programme pour exécuter du code du noyau
  - Adresse de branchement non connue à la compilation
    - Non souhaitable pour raison de maintenabilité, portabilité, …
  - Adresse de destination du *trap* stockée dans une portion de la mémoire physique
    - *Appelé vecteur d'interruptions*
    - *Accessible qu'en mode noyau*
    - *Initialisé au démarrage du SE*



Commandes

- Commande
  - Petit programme qui s'exécute en mode utilisateur
  - Application orientée caractère
  - Souvent associée à un appel système du noyau
    - Manipulation des programmes en cours d'exécutions, des fichiers, de la mémoire, des périphériques ….
- Commande de base
  - `ls` Liste les fichiers du répertoire courant
  - `rm` Supprime un fichier
  - `top` Listes des programmes en cours d'exécution

Shell UNIX

- Interprète de commandes d'un système Unix
  - Couche entre l'utilisateur et le système
  - Interface utilisateur orientée caractères
- Plusieurs Shells (beaucoup de points communs)
  - Bourne Shell (**sh**)
  - C Shell (**csh**)
  - Bourne Again Shell (**bash**)

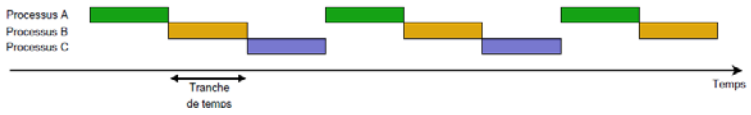
Shell et lancement de commande

- Décodage de la ligne de commande
  - Premier mot : nom de la commande
  - Reste : arguments
- Substitutions des variables et métacaractères
  - Voir plus loin …
- Recherche de la commande à exécuter
  - Fonctions internes du système « *builtins* »
  - Puis dans chacun des répertoires indiqués par la variable d'environnement **PATH**
- Exemples

`$ echo $PATH`	Affiche la valeur de la variable
`$ which ls`	Indique la commande utilisée par le shell

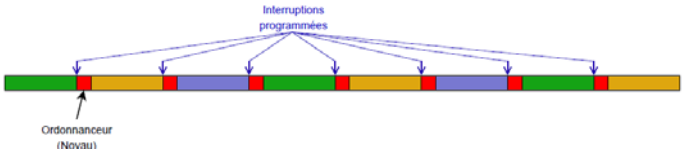
Processus

- Processus
  - Instance d'un programme en cours d'exécution
  - Un seul processus s'exécute à un instant donné (sur un processeur)
- Multiplexage
  - Simuler le parallélisme des programmes
  - Système multitâches : partage du temps



Notion d'ordonnanceur

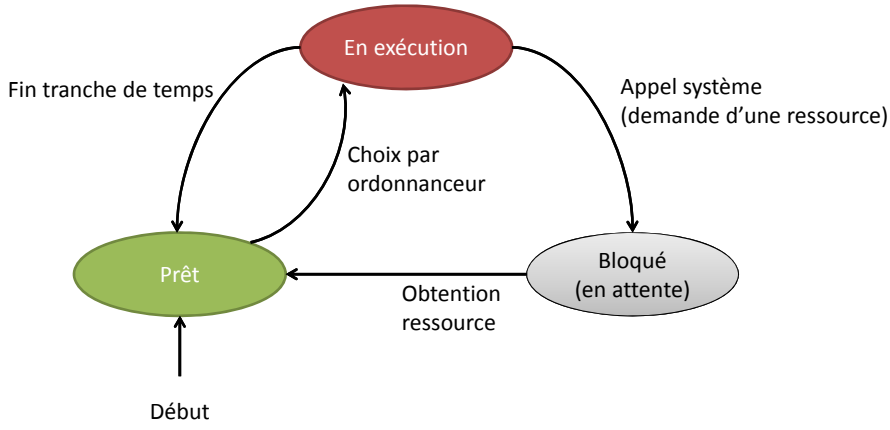
- Ordonnanceur
  - Programme du noyau
  - Gère l'ordre d'exécution des processus
- *Timer* (programmeur d'interrutions)
  - Interrompre l'exécution de chaque processus à la fin de sa tranche de temps (système préemptif)



Besoin d'ordonnanceur

- Interblocage de processus
  - Deux ou plusieurs processus se retrouvent dans une situations dont ils ne peuvent s'échapper
- Exemple avec deux processus P1 et P2
  - P1 et P2 ont besoins des ressources R1 et R2
    - (disque, DVD, imprimante, …)
  - P1 demande et obtient R1
  - P2 demande et obtient R2
  - P1 demande R2 ➔ refusé car déjà accordé à P2
  - P2 demande R1 ➔ refusé car déjà accordé à P1
  - Blocage des deux processus

États d'un processus



Attributs d'un processus UNIX

- Le système attribut à chaque processus :
  - De la mémoire (pile, tas, données statiques, registres, …)
  - Du temps processeur
- Quelques attributs principaux
  - PID — identifiant unique
  - PPID — identifiant du processus parent
  - UID — utilisateur propriétaire
  - Priorité — nombre entre -20 (très prioritaire) et +20 (non prioritaire)
  - Temps d'exécution (réel, processeur)
  - Terminal de contrôle (tty)
  - Répertoire courant

Threads

- Aussi appelé processus léger ou tâche
- Processus qui s'exécute au sein d'un processus
- Partage des ressources du processus hôte entre tous les threads qui le composent
- Un processus possède au moins un thread
- Partagent la même zone mémoire
  - ➔ Facilite la communication entre threads
  - ➔ Pose des problèmes d'accès concurrents
- Chaque thread possède son propre env. d'exécution (reg. du processeur) et sa pile (variables locales)

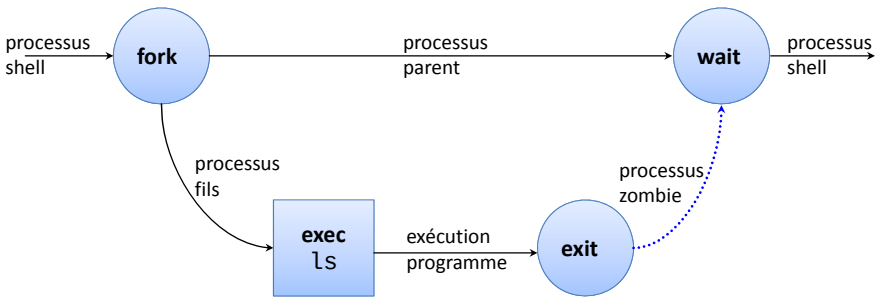
Processus lourds

- Processus *Init*
  - Premier processus créé par le noyau après l'amorçage
  - PID = 1
  - Denier processus avant l'arrêt de la machine
- Création d'un nouveau processus
  - Duplication (*fork*)
    - » Copie des segments de données et de pile
    - » Partage du segment texte (code)
    - » PID du fils différent de celui du père
    - » Fils hérite de l'environnement du père
  - Seule et unique manière de créer un processus (sauf Init)

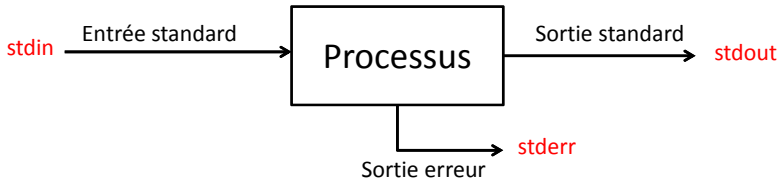
Hiérarchie de processus

- Relation *père-fils*
- Mort d'un processus
  - Libération des ressources du processus
  - Envoie d'une notification de décès au père
- Processus *zombie*
  - Cas du père qui ignore la mort de son fils
  - Processus mort reste visible
    - Ne consomme pas de ressources
    - Numéro PID toujours valide ➔ peux poser des problèmes
    - Processus impossible à tuer car déjà mort … d'où le nom!

Duplication de processus



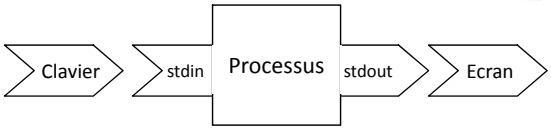
Processus et entrée/sorties



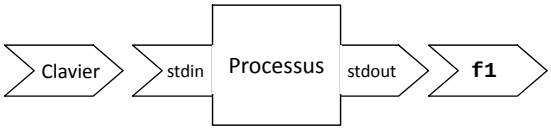
- Traitement orienté flux
  - Flot de données en entrée (entrée standard, clavier)
  - Traitement spécifique au processus
  - Flot de données en sortie (sortie standard, écran)

Redirection des entrées/sorties

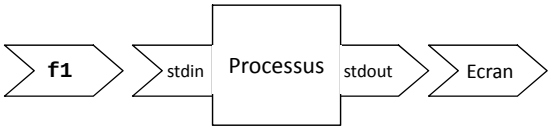
`travail`



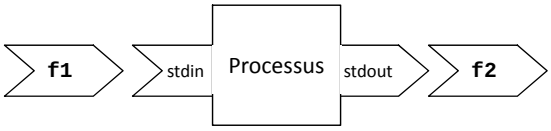
`travail > f1`



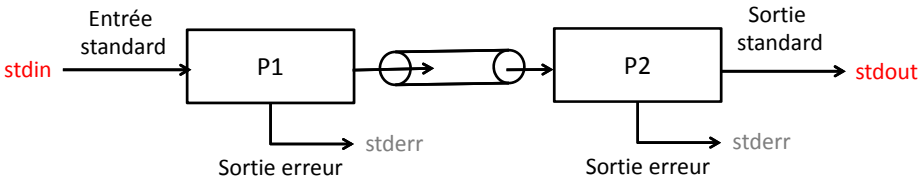
`travail < f1`



`travail < f1 > f2`



Communication par tube



- Échange de flux de données entre processus
  - Communication unidirectionnelle
  - Rediriger la sortie d'un processus sur l'entrée d'un autre
  - Exemple : `ls | wc -l`

Communication inter-processus

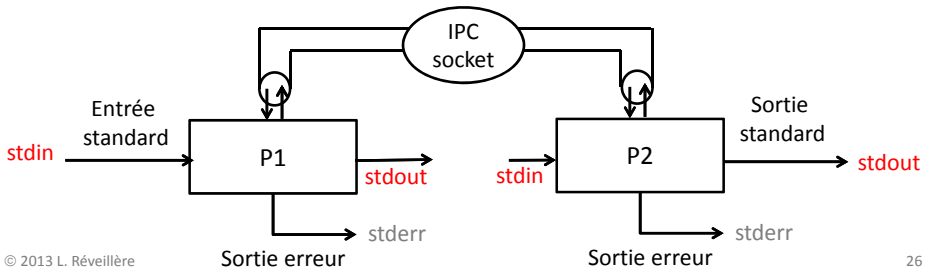
- Signaux
  - Envoie de message entre processus
  - Chaque signal identifié par un numéro de 1 à 31
  - Certains signaux associés à des actions prédéfinies
    - Exemple : tuer le processus
  - Un processus peux ignorer un signal ou réagir de manière spécifique
  - Le *shell* associe certaines combinaisons de touches à des envoies de signaux (ex. CTRL-c)

Principaux signaux

Numéro	Nom	Signification
1	HUP	fin de session
2	INT	interruption clavier (ctrl-c)
8	FPE	exception calcul virgule flottante
9	KILL	fin du processus (non modifiable)
10	USR1	définissable par l'utilisateur
11	SEGV	référence mémoire invalide
12	USR2	définissable par l'utilisateur
15	TERM	signal de fin
17	CHLD	terminaison d'un fils
18	CONT	reprise d'un processus stoppé
19	STOP	stoppe (non modifiable)
20	TSTP	stoppe (depuis clavier (ctrl-z))
29	WINCH	redimensionnement de fenêtre

Communication par socket

- Communication bi-directionnelle
- Communication entre processus quelconque
- Similaire à des tubes nommés
- Objet spécifique dans le système de fichier
  - Lecture et écriture comme dans un fichier standard



Manipuler les processus

- Commande ps
  - Liste tous les processus attachés au termnial
  - Très nombreuses options pour filtrer la liste et les informations à afficher
- Exemple

```
$ ps aux
```

- Commande UNIX à ne pas oublier

```
$ man ps
```

Envoyer des signaux

- Commande kill

```
$ kill [ -sinal ] pid …
```

- Envoie d'un signal à un ou plusieurs processus
  - Spécifier numéro du signal et PID du processus cible
- Exemple

`kill -9 2509` Envoie le signal 9 (KILL) au processus 2509

Shell UNIX

- Interprète de commandes d'un système Unix
  - Couche entre l'utilisateur et le système
  - Interface utilisateur orientée caractères
- Plusieurs Shells (beaucoup de points communs)
  - Bourne Shell (**sh**)
  - C Shell (**csh**)
  - Bourne Again Shell (**bash**)

Shell et lancement de commande

- Décodage de la ligne de commande
  - Premier mot : nom de la commande
  - Reste : arguments
- Substitutions des variables et métacaractères
  - Voir plus loin …
- Recherche de la commande à exécuter
  - Fonctions internes du système « *builtins* »
  - Puis dans chacun des répertoires indiqués par la variable d'environnement **PATH**
- Exemples

`$ echo $PATH`	Affiche la valeur de la variable
`$ which ls`	Indique la commande utilisée par le shell

Variables d'environnement

- Ensemble de couples nom/valeur (chaîne de caractères) associé à un processus
- Modifiable, **hérité** d'un processus père à un processus fils lors de la duplication
- Chaque processus manipule sa propre copie des variables
- Manipulation

```
$ export VAR=foo
$ echo $VAR
$ printenv
```


Gestion des processus interactifs

- Lancement d'une commande depuis le shell
  - Exécution en avant-plan (foreground)
  - Exécution séquentielle
  - Utilisation de CTRL-c et CTRL-z pour envoyer des signaux au processus en avant-plan

```
$ travail_A
$ travail_B
```


Gestion des processus interactifs

- Lancement de plusieurs commandes à la fois
  - Exécution en arrière-plan (background)
  - Exécution en parallèle
  - Liste des processus lancé par le shell : travaux (*jobs*)
  - Chaque job est identifié par un numéro (1, 2, 3 …)
  - Commande `jobs` affiche la liste des jobs
  - Manipulation des jobs avec les commandes `bg, fg, kill`

```
$ travail_A &
 [1] 207
$ travail_B &
 [2] 288
$ jobs -l
 [1]- 207 Running travail_A &
 [2]+ 288 Running travail_B &
```


Avant-plan / arrière-plan

- Passage d'un job en avant-plan

```
$ %1
$ fg %1
```

- Passage d'un job en arrière-plan

```
$ travail_A
$ ^Z
$ jobs
 [1]+ 488 Stopped travail_A
$ bg %1
```


Traitement différé

- Chaque commande lancé par le shell est associé à son terminal
  - Commande tué lorsque l'utilisateur quitte le terminal
  - Détacher une commande du terminal avec la commande nohup
- Lancement d'un commande à une heure précise
  - Commande `at`
- Lancement d'une commande à intervalle de temps périodique : `crontab`

Démons Unix

- Daemons (Disk and Execution Monitor)
  - Processus s'exécutant en arrière-plan
  - Souvent lancés au démarrage
  - Répondent à des requêtes

- Exemples
  - `inetd, httpd, nfsd, sshd, named, …`
- Daemons UNIX correspondent aux *services* de Microsoft Windows