Programmation système

M. Davy MOUSSAVOU- Consultant
Linux, Certifié Linux International

Gestion des processus

Objectifs du cours

Après cette session les apprenants doivent:

- Comprendre les caractéristiques d'un processus UNIX;
- Connaître le cycle de vie d'un processus
- Savoir utiliser les appels systèmes de gestion de processus

Agenda

- Séquence 1: Caractéristique d'un processus Unix
- Séquence 2: Cycle de vie d'un processus
- Séquence 3: Appels systèmes de gestion des processus

Séquence 1 : Caractéristiques d'un processus UNIX

Séquence 1 : Caractéristiques d'un processus

❑ Objectifs

- Comprendre le rôle d'un processus dans un système d'exploitation;
- Organisation des processus;

Séquence 1 : Caractéristiques d'un processus

❑ Rôle d'un processus

Définition

Sous Unix, toute tâche qui est en cours d'exécution est représentée par un ***processus***. Un processus est une entité comportant à la fois des donnée et du code. Il peut être considéré comme une unité élémentaire concernant l'activité du système.

Séquence 1 : Caractéristiques d'un processus

❑ Processus d'exécution d'un processus

Sur un ordinateur uni-processeur , un seul processus est exécuté à un instant donné. Le noyau assure une commutation régulière entre tous les processus, donnant à l'utilisateur l'impression que tout les programme s'exécutent parallèlement.

Séquence 1 : Organisation des disques

❑ Caractéristique d'un processus

Unix offre à chaque processus un environnement d'exécution contenant toutes les informations et services dont il a besoin telles que la limitation des la mémoire, temps d'exécution, etc.

C'est ce qu'on qualifie de contexte d'exécution. Ainsi chaque processus dispose de son propre contexte d'exécution.

Séquence 1 : Caractéristiques d'un processus

❑ Informations d'un processus

- ✓ Numéro unique du processus PID
- ✓ Numéro du processus parent PPID
- ✓ Numéro d'utilisateur UID ayant lancé le processus
- ✓ Numéro du groupe GID ayant lancé le processus
- ✓ Durée de traitement utilisé (temps cpu)
- ✓ Priorité du processus
- ✓ Référence au répertoire de travail courant du processus
- ✓ Table de référence des fichiers ouverts par le processus

Séquence 1 : Caractéristiques d'un processus

❑ Processus et la mémoire

- ✓ **Un processus est une entité comportant à la fois des donnée et du code.**
- ✓ **Il peut être considéré comme une unité élémentaire concernant l'activité du système.**
- ✓ **En conséquence, il consomme de la mémoire vive**

Séquence 1 : Caractéristiques d'un processus

❑ Charge du système

- Chaque processus contribue à la charge du système.
- Linux est un système à temps partagé.
- Sur un ordinateur uni-processeur , un seul processus est exécuté à un instant donné.
- Donc chaque processus occupe du temps processeurs à instant donné.
- Le noyau assure une commutation régulière entre tous les processus, donnant à l'utilisateur l'impression que tout les programme s'exécutent parallèlement.

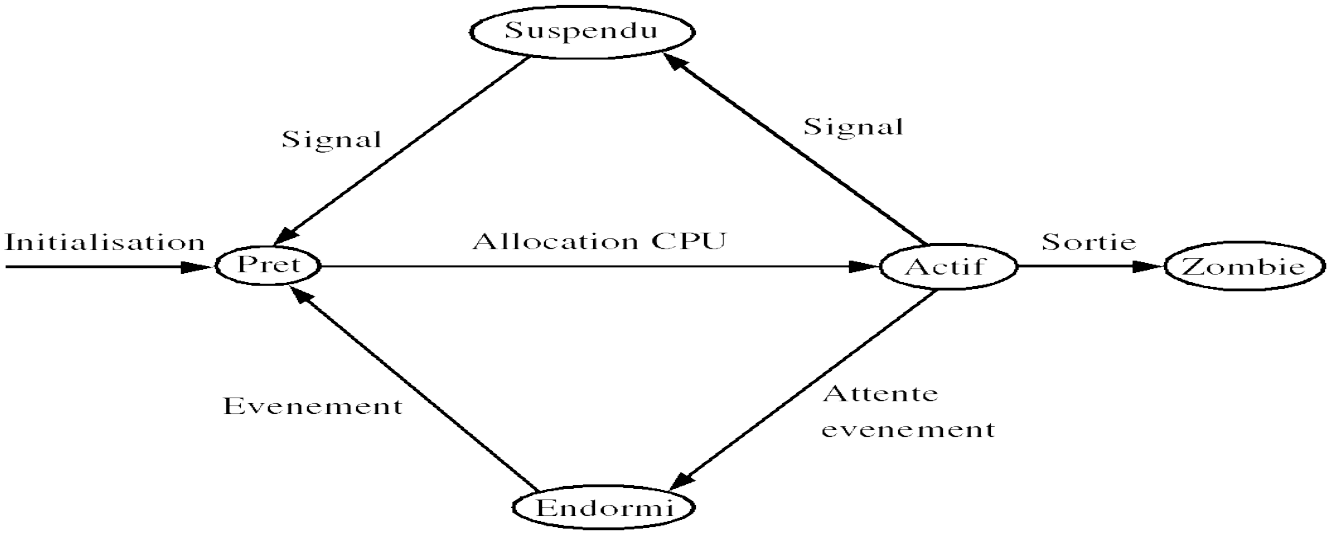
Séquence 2 : Cycle de vie d'un processus

❑ Objectifs

- Comprendre les états d'un processus

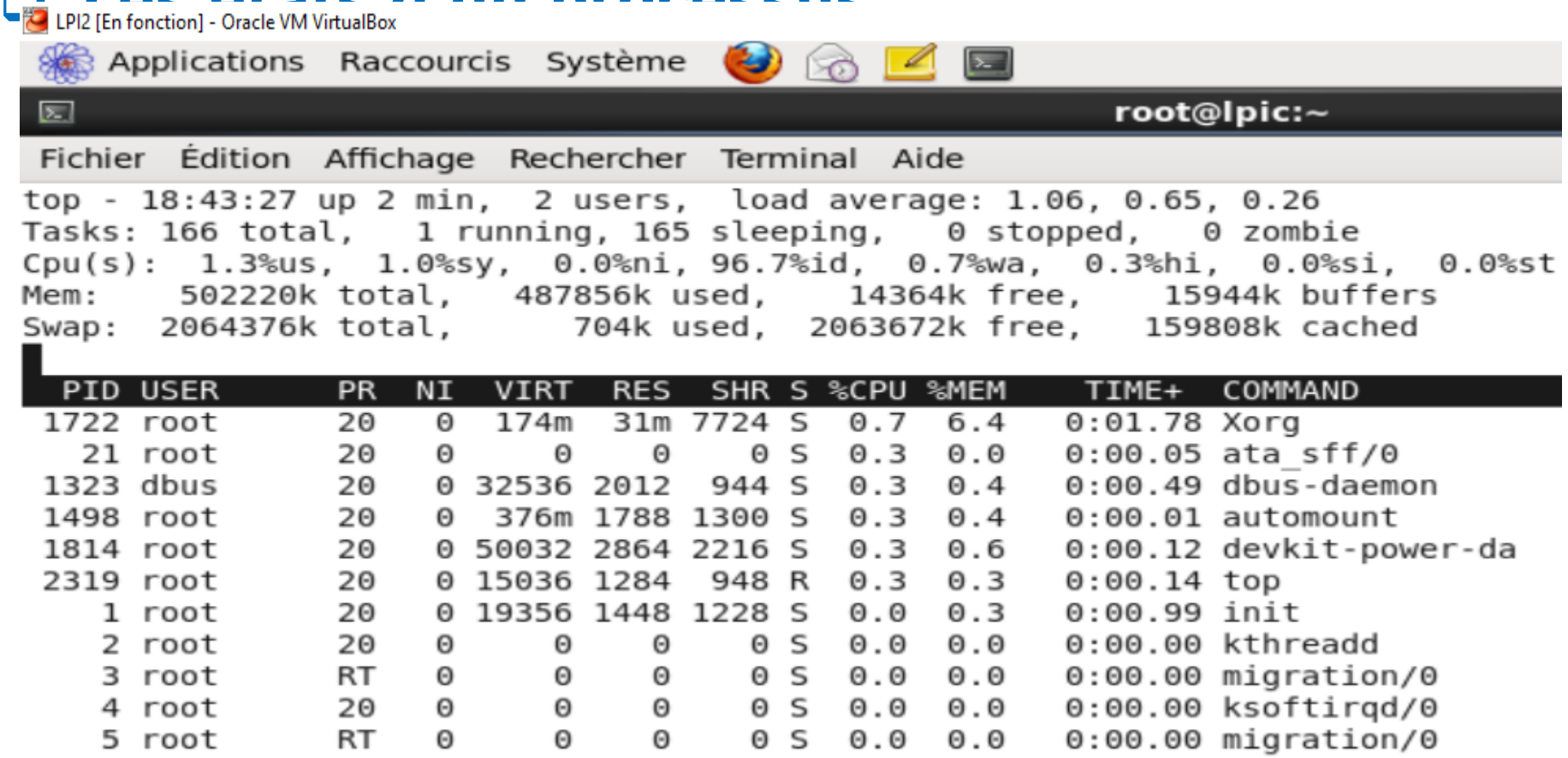
Séquence 2 : Cycle de vie d'un processus

❑ Les états d'un processus



Séquence 2 : Cycle de vie d'un processus

- Les états d'un processus



Séquence 3 : Appel système de gestion de processus

❑ Objectifs

- Afficher les informations d'un processus
- Créer un processus
- Attente d'un processus
- Exécuter des programmes

Séquence 3 : Appels systèmes de gestion de processus

Information sur un processus

- **getpid(2), getppid (2)**

```
NOM
       getpid, getppid - Obtenir l'identifiant d'un processus

SYNOPSIS
       #include <sys/types.h>
       #include <unistd.h>

       pid_t getpid(void);
       pid_t getppid(void);

DESCRIPTION
       getpid() renvoie l'identifiant du processus appelant. Ceci est sou-
       vent utilisé par des routines qui créent des  fichiers  temporaires
       uniques.

       getppid() renvoie le PID du processus père de l'appelant.
```

Séquence 3 : Appels systèmes de gestion de processus

Information sur un processus

- **getpid(2), getppid (2)**

Exemple :

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
int pid;
//appel de la fonction
pid =getpid()
printf("pid:%d\n", pid);
return 0;
}
```

```
$cc my_pid –o my_pid
$./my_pid
```

Séquence 3 : Appels systèmes de gestion de processus

Création d'un processus

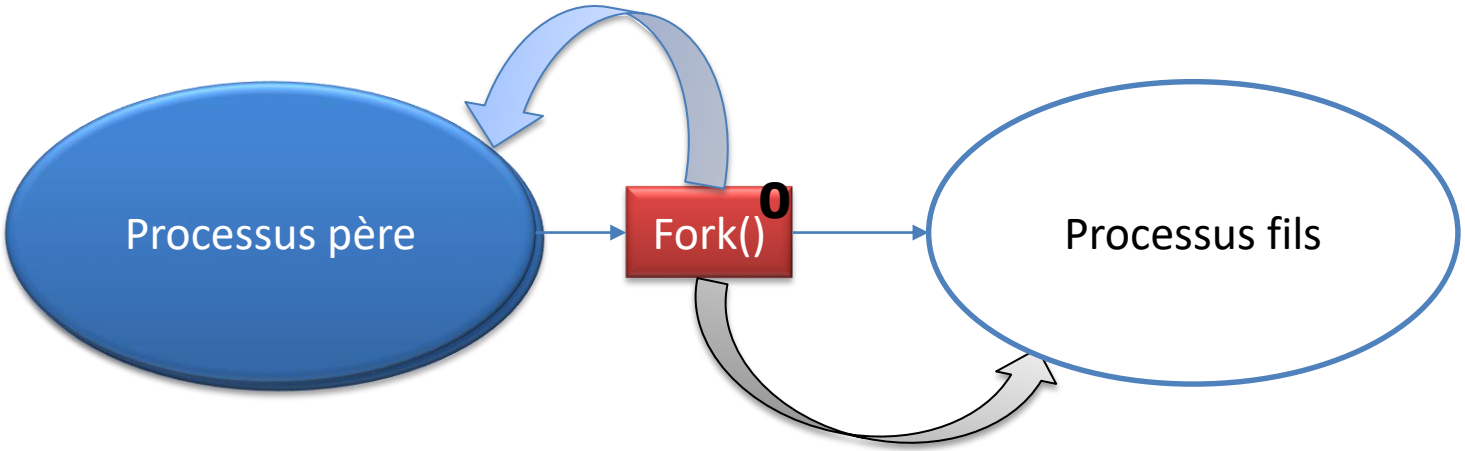
- **fork(2)**

```
root@lpic:~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
FORK(2)                    Manuel du programmeur Linux                    FORK(2)

NOM
       fork - Créer un processus fils

SYNOPSIS
       #include <unistd.h>

       pid_t fork(void);

DESCRIPTION
       fork()  crée un nouveau processus en copiant le processus appelant.
       Le nouveau processus, qu'on appelle fils (« child »), est une copie
       exacte  du  processus  appelant, qu'on appelle père ou parent, avec
       les exceptions suivantes :
```

Séquence 3 : Appels systèmes de gestion de processus

Création d'un processus

- **fork(2)**



Séquence 3 : Appels systèmes de gestion de processus

Attente d'un processus

- **wait(2)**

```
NOM
       wait, waitpid, waitid - Attendre la fin d'un processus

SYNOPSIS
       #include <sys/types.h>
       #include <sys/wait.h>

       pid_t wait(int *status);

       pid_t waitpid(pid_t pid, int *status, int options);

       int   waitid(idtype_t   idtype,  id_t  id,  siginfo_t  *infop,  int
       options);

   Exigences de macros de test de fonctionnalités pour la glibc (voir fea-
   ture_test_macros(7)) :

       waitid() : _SVID_SOURCE || _XOPEN_SOURCE

DESCRIPTION
       Tous  ces  appels  système  attendent  qu'un  des fils du processus
       appelant change d'état, et permettent  d'obtenir  des  informations
       sur le fils en question. Un processus est considéré comme changeant
       d'état s'il termine, s'il est stoppé par un  signal,  ou  s'il  est
       relancé  par  un  signal.  Dans  le  cas  d'un fils qui se termine,
:
```

Séquence 3 : Appels systèmes de gestion de processus

Exécution d'un programme

- **exec*()**

```
NOM
       execl, execlp, execle, execv, execvp - Exécuter un fichier

SYNOPSIS
       #include <unistd.h>

       extern char **environ;

       int execl(const char *path, const char *arg, ...);
       int execlp(const char *file, const char *arg, ...);
       int execle(const char *path, const char *arg,
        ..., char * const envp[]);
       int execv(const char *path, char *const argv[]);
       int execvp(const char *file, char *const argv[]);

DESCRIPTION
       La  famille  des  fonctions exec() remplace l'image du processus en
       cours par une nouvelle image du processus. Les  fonctions  décrites
       dans  cette page sont en réalité des frontaux pour execve(2) (voyez
       la page de manuel de execve(2) pour plus de  détails  sur  le  rem-
       placement de l'image du processus en cours).

       L'argument initial de toutes ces fonctions est le chemin d'accès du
       fichier à exécuter.
```

Atelier