Unix : les PROCESSUS

Laure Danthony

1 Introduction

Un processus est composé d'un programme (code machine) et un environnement de processus (ensemble de données d'identification pour chaque programme en cours).

2 Caractéristiques d'un processus

Un processus a un descripteur : une structure qui contient :

- un numéro spécifique, son PID (il est unique).
- un numéro de processus parent (PPID).
- un numéro d'utilisateur (UID).
- un numéro de groupe (GID).
- une durée de traitement (temps CPU) et une priorité.
- une référence au répertoire de travail courant.
- une table de référence des fichiers ouverts (indexée : 0 pour le canal d'entrée standard, 1 pour la sortie standard, 2 pour le retour standard d'erreur).

3 Gestion de mémoire

- On alloue généralement 8Mo max pour la **mémoire virtuelle** d'un processus.
- Les composantes du processus (code, données, pile, tas -malloc-) sont projetées sur des pages (4 ou 8 ko) indivisibles de la mémoire physique.

4 Création d'un processus

- Le primitive `Fork()` duplique complétement le processus en cours. Elle renvoie le PID du nouveau processus au père, et 0 au fils.
- Les pages sur lesquelles pointe le processus fils sont moins importantes. On n'alloue pas de la mémoire au fils, il pointe sur les mêmes pages que son père. Il a quand même des informations personnelles, mais qui prennent peu de place.

5 Les différents états d'un processus

- Après sa naissance, il passe à l'état **"prêt"** où il attend son élection par l'ordonnanceur vers un état **"actif noyau"**.
- Pour passer d'un état **"actif utilisateur"** à un état "actif noyau", il faut faire un appel système.
- Un processus peut passer à un état **"bloqué"** (`^Z`), mais après avoir été débloqué il repasse à l'état "prêt".
- Un état **"Zombie"** (sans retour) est atteint à partir de l'état "actif noyau" si l'on n'a pas récupéré l'information de mort du processus (avec `wait(int *status)`, `status` étant le code de retour d'un processus.

6 Ordonnancement et recouvrement d'un processus

- **Priorité d'un processus** : chaque processus est affecté d'une priorité de 0 à 20 qui permet de les ordonnancer: les priorités les fortes (numéros les plus bas) sont pour les appels disques ("priorités noyau"), les priorités utilisateur sont affectées par rapport au temps utilisateur (plus un processus prend de CPU, plus sa priorité se rapproche de 20).

  REMARQUE 1 Certains processus utilisateurs (ex `fwrite`) peuvent obtenir des priorités dites "noyau".

- **Recouvrement d'un processus** : la famille des primitives `exec` écrase l' espace d'adressage courant avec un programme chargé depuis le disque. Par exemple, `execv("bin/ls", char *argv)` permet de faire un `ls` lors d'un processus (c'est utile pour faire un shell).