osj

9 avril 2025

Table des matières

0.1 Introduction

0.2 Création et terminaison de processus

Un processus (le « parent ») peut créer un autre processus (le « enfant »). Un nouveau PCB est alloué et initialisé. Dans POSIX, le processus enfant hérite de la plupart des attributs du parent, tels que l'UID, les fichiers ouverts (qui devraient être fermés si inutiles ; pourquoi ?), le répertoire de travail courant, etc.

0.2.1 Mouvement du PCB

Lors de l'exécution, le PCB se déplace entre différentes files d'attente, en fonction du graphique de changement d'état. Ces files d'attente peuvent être : exécutable, sommeil/attente pour l'événement i (i=1,2,3…).

0.2.2 Processus zombie

Après qu'un processus meurt (exit()s / interrompu), il devient un zombie. Le parent utilise le syscall wait* pour effacer le zombie du système (pourquoi ?). La famille de syscall wait comprend : wait, waitpid, waitid, wait3, wait4. Par exemple :

```

pid_t wait4(pid_t, int *wstatus, int options, struct rusage *rusage);
```

Le parent peut dormir/attendre que son enfant termine ou s'exécute en parallèle. wait*() bloquera à moins que WNOHANG ne soit donné dans 'options'.

0.3 OS (234123) - processus et signaux

C'est le septième cours de la série sur les systèmes d'exploitation et les signaux.

```

pid_t wait4(pid_t, int *wstatus, int options, struct rusage *rusage);
```

0.4 Fork - Création d'un processus enfant

La fonction `fork()` est utilisée pour initialiser un nouveau PCB (Process Control Block) basé sur la valeur du processus parent. Une fois que le PCB est ajouté à la file d'exécution, il y a maintenant deux processus au même point d'exécution.

L'espace d'adresse du processus enfant est une copie complète de l'espace du processus parent, avec une différence : la fonction `fork()` retourne deux fois. Chez le parent, avec `pid > 0` et chez l'enfant, avec `pid = 0`.

La variable globale `'errno'` contient le numéro d'erreur du dernier appel système.