osj

9 avril 2025

Table des matières

0.1 Introduction

0.2 Création et terminaison des processus

Un processus, appelé le « parent », peut en créer un autre, appelé le « enfant ». Un nouveau PCB (Process Control Block) est alloué et initialisé. Le processus enfant hérite de la plupart des attributs du processus parent dans POSIX, tels que l'UID, les fichiers ouverts (qui doivent être fermés s'ils ne sont pas nécessaires), le répertoire de travail courant, etc.

0.2.1 Exécution du processus

Lors de son 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 Terminaison du processus

Après qu'un processus meurt (soit par une sortie, soit par une interruption), il devient un zombie. Le parent utilise l'appel système wait* pour effacer le zombie du système.

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

0.3 OS (234123) - processus et signaux

Cette section traite des processus et des signaux dans le système d'exploitation.

```

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

0.4 Création d'un processus enfant avec fork()

La fonction fork() est utilisée pour créer un nouveau processus, appelé processus enfant, à partir du processus actuel, appelé processus parent. Cette fonction initialise un nouveau Bloc de Contrôle de Processus (PCB) basé sur la valeur du processus parent. Le PCB est ensuite ajouté à la file d'exécution.

Il y a maintenant deux processus en exécution au même point d'exécution. L'espace d'adressage du processus enfant est une copie complète de l'espace d'adressage du processus parent, à une différence près : la fonction fork() renvoie deux valeurs.

Dans le processus parent, fork() renvoie un pid > 0, tandis que dans le processus enfant, fork() renvoie un pid $= 0$.

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

L'ordre d'impression n'est pas déterminé et peut varier.

```

int main(int argc, char *argv[])
{
  int pid = fork();
  if( pid==0 ) {
   //
   // child
      //
      printf("parent=%d son=%d\n",
             getppid(), getpid());
  }
  else if( pid > 0 ) {
      //
      // parent
      //
      printf("parent=%d son=%d\n",
             getpid(), pid);
  }
  else { // print string associated
         // with errno
      perror("fork() failed");
  }
  return 0;
}
```

Fiche Récapitulative

0.5 Création et terminaison des processus

Un processus, appelé le « parent », peut en créer un autre, appelé le « enfant ». Un nouveau Bloc de Contrôle de Processus (PCB) est alloué et initialisé. Le processus enfant hérite de la plupart des attributs du processus parent dans POSIX, tels que l'UID, les fichiers ouverts (qui doivent être fermés s'ils ne sont pas nécessaires), le répertoire de travail courant, etc.

0.5.1 Exécution du processus

Lors de son 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...).

0.5.2 Terminaison du processus

Après qu'un processus meurt (soit par une sortie, soit par une interruption), il devient un zombie. Le parent utilise l'appel système wait* pour effacer le zombie du système.

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

0.6 OS (234123) - processus et signaux

Cette section traite des processus et des signaux dans le système d'exploitation.

0.7 Création d'un processus enfant avec fork()

La fonction fork() est utilisée pour créer un nouveau processus, appelé processus enfant, à partir du processus actuel, appelé processus parent. Cette fonction initialise un nouveau Bloc de Contrôle de Processus (PCB) basé sur la valeur du processus parent. Le PCB est ensuite ajouté à la file d'exécution.

Il y a maintenant deux processus en exécution au même point d'exécution. L'espace d'adressage du processus enfant est une copie complète de l'espace d'adressage du processus parent, à une différence près : la fonction fork() renvoie deux valeurs.

Dans le processus parent, fork() renvoie un pid > 0, tandis que dans le processus enfant, fork() renvoie un pid $= 0$.

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

L'ordre d'impression n'est pas déterminé et peut varier.