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 »). Pour cela, un nouveau PCB (Process Control Block) est alloué et initialisé. Pour mieux comprendre ce concept, vous pouvez exécuter la commande 'ps auxwww' dans le shell ; PPID est l'identifiant du processus parent.

0.2.1 Héritage des attributs du parent par le processus enfant

Dans POSIX, le processus enfant hérite de la plupart des attributs du processus parent. Cela inclut l'UID, les fichiers ouverts (qui devraient être fermés si inutiles ; pourquoi ?), le répertoire de travail courant, etc.

0.2.2 Déplacement du PCB entre différentes files d'attente

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 d'un événement i (i=1,2,3…).

0.2.3 Processus zombie

Après qu'un processus meurt (soit par une sortie normale, soit par une interruption), il devient un zombie. Le processus parent utilise l'appel système wait* pour effacer le zombie du système. Pourquoi ? C'est une question à approfondir.

0.2.4 Attente du processus parent

Le processus parent peut soit attendre que son enfant termine, soit s'exécuter en parallèle. L'appel système wait*() bloquera à moins que WNOHANG ne soit donné dans 'options'.

0.3 Devoirs

Pour mieux comprendre ces concepts, vous pouvez lire la page de manuel 'man 2 wait'.

```

\begin{tcolorbox}[    colback=yellow!10,    colframe=yellow,    title={\
 fontfamily{lmr}\selectfont \faBookmark À retenir},    fonttitle=\bfseries,
 fontupper=\fontfamily{lmr}\selectfont,    boxrule=1pt,    sharp corners,
 ]  Un processus parent peut créer un processus enfant, qui hérite de la
 plupart des attributs du parent. Le PCB du processus se déplace entre
 différentes files d'attente pendant son exécution. Lorsqu'un processus meurt,
 il devient un zombie et doit être nettoyé par le processus parent à l'aide de
 l'appel système wait*.  \end{tcolorbox}

```

```
\section{Création d'un processus enfant avec fork()}
La fonction fork() est utilisée pour créer un nouveau processus, appelé processus enfant, qui fonctionne en parallèle avec le processus qui l'a créé, appelé processus parent.

\subsection{Initialisation d'un nouveau PCB avec fork()}
La fonction fork() initialise un nouveau Bloc de Contrôle de Processus (PCB) basé sur la valeur du processus parent. Le nouveau PCB est ensuite ajouté à la file d'exécution.

\subsection{Création de deux processus avec fork()}
Après l'appel à fork(), il y a maintenant deux processus qui se trouvent au même point d'exécution.

\subsection{Espace d'adresse du processus enfant}
L'espace d'adresse du processus enfant est une copie complète de l'espace d'adresse du processus parent, avec une seule différence...

\subsection{Retour de fork()}
La fonction fork() retourne deux fois : une fois dans le processus parent avec un pid>0 et une fois dans le processus enfant avec un pid=0.

\subsection{Ordre d'impression}
Quel est l'ordre d'impression ?

\subsection{La variable globale 'errno'}
'errno' est une variable globale qui contient le numéro d'erreur du dernier appel système.

\begin{lstlisting}

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;
}
```

À retenir

La fonction fork() est utilisée pour créer un nouveau processus, appelé processus enfant. Cette fonction initialise un nouveau Bloc de Contrôle de Processus (PCB) basé sur la valeur du processus parent et l'ajoute à la file d'exécution. Après l'appel à fork(), il y a maintenant deux processus qui se trouvent au même point d'exécution. L'espace d'adresse du processus enfant est une copie complète de celui du processus parent, avec une seule différence. La fonction fork() retourne deux fois : une fois dans le processus parent avec un pid>0 et une fois dans le processus enfant avec un pid=0.

Fiche Récapitulative

0.4 Création et terminaison de processus

Un processus, appelé le « parent », peut créer un autre processus, appelé le « enfant ». Pour cela, un nouveau Bloc de Contrôle de Processus (PCB) est alloué et initialisé. Pour mieux comprendre ce concept, vous pouvez exécuter la commande 'ps auxwww' dans le shell ; PPID est l'identifiant du processus parent.

0.4.1 Héritage des attributs du parent par le processus enfant

Dans POSIX, le processus enfant hérite de la plupart des attributs du processus parent. Cela inclut l'UID, les fichiers ouverts (qui devraient être fermés si inutiles ; pourquoi ?), le répertoire de travail courant, etc.

0.4.2 Déplacement du PCB entre différentes files d'attente

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 d'un événement i (i=1,2,3...).

0.4.3 Processus zombie

Après qu'un processus meurt (soit par une sortie normale, soit par une interruption), il devient un zombie. Le processus parent utilise l'appel système wait* pour effacer le zombie du système. Pourquoi ? C'est une question à approfondir.

0.4.4 Attente du processus parent

Le processus parent peut soit attendre que son enfant termine, soit s'exécuter en parallèle. L'appel système wait*() bloquera à moins que WNOHANG ne soit donné dans 'options'.

0.5 Devoirs

Pour mieux comprendre ces concepts, vous pouvez lire la page de manuel 'man 2 wait'.

0.6 Création d'un processus enfant avec fork()

La fonction fork() est utilisée pour créer un nouveau processus, appelé processus enfant, qui fonctionne en parallèle avec le processus qui l'a créé, appelé processus parent.

0.6.1 Initialisation d'un nouveau PCB avec fork()

La fonction fork() initialise un nouveau Bloc de Contrôle de Processus (PCB) basé sur la valeur du processus parent. Le nouveau PCB est ensuite ajouté à la file d'exécution.

0.6.2 Création de deux processus avec fork()

Après l'appel à fork(), il y a maintenant deux processus qui se trouvent au même point d'exécution.

0.6.3 Espace d'adresse du processus enfant

L'espace d'adresse du processus enfant est une copie complète de l'espace d'adresse du processus parent, avec une seule différence...

0.6.4 Retour de fork()

La fonction fork() retourne deux fois : une fois dans le processus parent avec un pid>0 et une fois dans le processus enfant avec un pid=0.

0.6.5 Ordre d'impression

Quel est l'ordre d'impression ?

0.6.6 La variable globale 'errno'

'errno' est une variable globale qui contient le numéro d'erreur du dernier appel système.