osj

April 9, 2025

Contents

0.1 Introduction

0.2 Process Creation and Termination

In an operating system, one process, referred to as the 'parent', can create another process, known as the 'child'. This creation involves the allocation and initialization of a new Process Control Block (PCB). A practical example of this can be seen by running the command 'ps auxwww' in the shell, where PPID represents the parent's PID.

0.2.1 Inheritance in POSIX

In POSIX, a child process inherits most of the parent's attributes. These attributes include the User ID (UID), open files, current working directory (cwd), and others. It is important to note that open files should be closed if they are not needed.

0.2.2 Process Control Block (PCB) Execution

While a process is executing, its PCB moves between different queues according to the state change graph. These queues include the runnable queue and the sleep/wait for event i queue (where i can be any integer).

0.2.3 Process Termination

After a process terminates (either through an exit or interruption), it becomes a 'zombie'. The parent process uses the wait* system call to clear the zombie from the system. The wait system call family includes wait, waitpid, waitid, wait3, and wait4. An example of this is the wait4 system call: pid$_t wait4(pid_t, int * wstatus, intoptions, structrusage *$
$rusage).Theparentprocesscaneithersleep/waitforitschildprocesstofinishorruninparallel.Thewait*$
$()systemcallwillblockunlessWNOHANGisgivenin'options'.Forfurtherunderstanding, itisrecommende$

0.3 Création d'un processus enfant avec fork()

La fonction fork() est utilisée pour créer un nouveau processus, appelé processus enfant, qui est une copie du processus qui a appelé fork(), appelé processus parent. Lorsqu'un processus appelle fork(), un nouveau Bloc de Contrôle de Processus (PCB) est initialisé basé sur la valeur du processus parent et ajouté à la file d'attente exécutable. À ce stade, deux processus existent au même point d'exécution.

0.3.1 Espace d'adresse du processus enfant

L'espace d'adresse du nouveau processus enfant est une copie complète de l'espace d'adresse du processus parent, avec une différence : fork() renvoie deux fois. Dans le processus parent, fork() renvoie un pid>0, tandis que dans le processus enfant, fork() renvoie pid=0.

0.3.2 Ordre d'impression

L'ordre d'impression n'est pas déterminé et dépend de l'ordonnanceur du système d'exploitation.

0.3.3 Gestion des erreurs

'errno' est une variable globale qui contient le numéro d'erreur du dernier appel système.
Si fork() échoue, 'errno' contiendra le numéro d'erreur correspondant.

```
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;
}
```