TD Programmation système et réseaux
No 5

Avril 2003

Rappel

- Exécution d'une commande par le système : int system (const char * comm);
  - exécute la commande indiquée dans comm en appelant /bin/sh -c comm
  - rend -1 en cas d'erreur sinon ou le code de retour de la commande.
- Duplication de processus pid_t fork(void);
  - rend -1 en cas d'échec
  - rend 0 dans le fils et le numéro de processus crée dans le père
- Arrêt d'un processus void exit(int statut)
  - renvoi la valeur de statut vers son processus père (très souvent le shell)
- Mise en attente d'un processus int wait(int * valeur_statut)
  - le processus se bloque en attente de la fin d'un de ses fils
  - rend le pid du premier fils mort trouvé
  - met la valeur de statut lors de la mort du fils dans valeur_statut.
- Recouvrement du code d'un processus : int execve (char *filename, char * argv[], char *envp[]);
  - Remplace le code du processus par le code donné dans le fichier filename
  - argv est un tableau de pointeurs de chaînes de caractères contenant les arguments du nouveau processus :
    - argv[0] est le nom du programme.
    - le dernier indice du tableau doit contenir NULL
  - envp est un tableau de pointeurs de chaînes de caractères contenant les variables d'environnement construite sous la forme <nom de la varaible > = < valeur de la variable >
    - le dernier indice du tableau doit contenir NULL
- Recouvrement du code d'un processus (variantes) :
  - int execvl (char *filename, char * argv0, char * argv1, ...);
  - int execvle (char *filename, char * argv0, char * argv1, ..., char *envp[]);
  - int execvl (char *filename, char * argv[]);
  - int execvlp (char *cmdname, char * argv0, char * argv1, ...);
  - int execvp (char *cmdname, char * argv[]);
    - permettent de spécifier les arguments sous forme d'une liste d'arguments (argv0, ...) et/ou de recherche le programme par rapport à la variable d'environnement $PATH

Exercices

1. Soient les deux programmes

   Exec.c

   ```
   int main() {
     char * Argv[2];
     Argv[0] = "ps";
     Argv[1] = NULL;
     char * filename = "/bin/ps";
     execv(filename, Argv);
   }
   ```

   et System.c

   ```
   int main() {
     system ("/bin/ps");
   }
   ```

   Combien y a-t-il de processus créés lorsqu'on lance le programme Exec? le programme System?

2. Fork et pid

   Ecrire un programme C qui se duplique puis disparait. Le fils créé doit recommencer la même opération, ... Le petit-petit-petit-petit-petit fils du processus initial doit afficher les PID de ses ancêtres et disparaître.

   On utilisera uniquement les fonctions fork et getpid.

3. Fork et fichier

   (a) Ecrire un programme qui crée le fichier Res.txt puis crée deux fils. Ces deux fils doivent écrire 5 'A' et 5 'B' (par exemple "AAABBABBAB") dans le fichier Res.tex avec la contrainte que le premier fils ne peut qu'écrire des 'A' dans ce fichier et le deuxième fils que des 'B'. Enfin, le processus père doit afficher le fichier Res.txt.

   (b) idem, mais le fichier doit contenir exactement "ABABABABAB"

4. Soit le fichier de commande lsEx

   ```
   for P in `ls $1` ; do
    if [ -x $1/$P ]; then
     if [ ! -d $1/$P ]; then
       echo $1/$P;
     fi
    fi
   done
   ```

   Ecrire un programme C qui en utilisant lsEx, lance en "même temps" grâce à des execv tous les exécutables se trouvant sous un répertoire donné en paramètre du programme.

   Il est interdit de modifier lsEx. On suppose que les exécutables trouvés n'ont pas besoin de paramètres.