Examen de Système d'exploitation (OS 302)

28 mai 2019

Enseignants : O. Aktouf, J. Marconot, K. Ndiaye

Documents de CM, TD et TP autorisés

Partie machine

(durée estimée : 1 heure)

Consignes générales

Cet examen sur machine comporte 2 exercices.

Dans votre répertoire privé (celui dans lequel vous êtes positionné juste après votre connexion) vous devez créer un répertoire de nom **os302VOTRENOM** (« VOTRENOM » doit être remplacé par votre nom de famille en majuscules, sans accent, sans tiret, sans espace, ...).

Dans le répertoire **os302VOTRENOM**, créez un répertoire **EXO1** et un autre **EXO2**.

Le(s) source(s) de l'exercice 1 doivent se trouver dans le répertoire `EXO1` et ceux de l'exercice 2 doivent être dans le répertoire `EXO2`.

Exercice 1 : création de processus (4 points)

L'objectif de cet exercice est d'effectuer la somme de 2 matrices carrées 2×2. Le programme principal permet la création d'un processus fils par opération élémentaire. Le processus père se synchronise alors sur la terminaison de tous les processus fils afin de s'assurer que tous les calculs sont bien finis.

Les fonctions nécessaires à cet exercice sont décrites ci-dessous. Certaines sont fournies, d'autres sont à compléter.

Fichier funcl.h

```
#ifndef DM
// DM : dimensions matrice carrée
#define DM 2
// NF : nombre de processus fils
#define NF DM*DM

int **mat1, **mat2, **matsomme;
// mat1 et mat2 sont les 2 matrices lues au clavier. La matrice matsomme
correspond à la somme des 2 matrices mat1 et mat2.

/*** fonctions à compléter    ***/
void lecture_matrice(void);
//lecture des éléments des 2 matrices mat1 et mat2

int somme(int v1, int v2);
// somme de 2 entiers

void creation(pid_t tab[NF]);
// création de 4 processus fils dont chacun calculera une somme élémentaire

/*** fonctions fournies       ***/
int **allocate_matrice(int i, int j);
// allocation de la zone mémoire pour la création des matrices

void affiche_matrice(int **mat, int i, int j);
// affichage des éléments d'une matrice

void free_matrice(int** mat, int i);
// libération de la zone mémoire occupée par une matrice

void free_matrices(void);
//libération de la zone mémoire occupée par les 3 matrices de cet exercices
#endif
```

Fichier func1.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <wait.h>
#include "func1.h"

void lecture_matrice(void)
{ /* Lit 2 matrices d'éléments entiers et de dimensions 2□2 - à compléter */
}

int somme(int v1, int v2)
{ /* renvoie la somme des 2 entiers transmis en arguments - à completer */
}

void creation(pid_t tab[NF])
{/*
Permet de créer 4 processus fils dont les PID sont stockés dans le tableau
tab. Chaque processus fils effectue une some élémentaire :
-     Le fils 1 effectue la somme de mat1[0][0] et mat2[0][0]
-     Le fils 2 effectue la somme de mat1[0][1] et mat2[0][1]
-     Le fils 3 effectue la somme de mat1[1][0] et mat2[1][0]
-     Le fils 4 effectue la somme de mat1[1][1] et mat2[1][1]
Chaque processus fils transmets le résultat de son calcul au processus père au
moyen de la fonction exit().
*/
printf("Création processus \n");
}

int **allocate_matrice( int i, int j)
{ int k;

  int **mat;
  mat = malloc(i*sizeof(*mat));
  for(k=0; k<i; k++){
    mat[k] = malloc(j*sizeof(*mat[k]));
  }
  return mat;
}

void affiche_matrice(int** mat, int i, int j)
{int k, l;

  for (k=0; k<i; k++){
    for (l=0; l<j; l++){
      printf("%d \t", mat[k][l]);
    }
    printf("\n");
  }
}

void free_matrice(int** mat, int i)
{int k;

  for (k=0; k<i; k++)
  {
    free(mat[k]);
```

```
  }
  free(mat);
}
void free_matrices(){
  free_matrice(mat1,DM);
  free_matrice(mat2,DM);
  free_matrice(matsomme,DM);
}
```

Fichier exo1.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <wait.h>
#include "func1.h"

/** ne doit pas être modifié **/

int main (void)
{int i;
  pid_t mes_fils[NF];
  int status;

  mat1=allocate_matrice(DM,DM);
  mat2=allocate_matrice(DM,DM);
  matsomme=allocate_matrice(DM,DM);

  lecture_matrice();
  creation(mes_fils);

  for (i=0; i<NF; i++){
    waitpid(mes_fils[i], &status, 0);
    printf("status fils %d %04x\n",(i+1), status);
    matsomme[i/DM][i%DM]=status >> 8;
  }
  affiche_matrice(matsomme,DM,DM);

return 0;
}
```

Fichier Makefile

```
all: exo1 clean

exo1: func1.o main.o
      gcc -o exo1 func1.o main.o

func1.o: func1.c func1.h
      gcc -o func1.o -c func1.c -g -Wall

main.o: exo1.c func1.h
      gcc -o main.o -c exo1.c -g -Wall

clean:
      rm -rf *.o
```

Exercice 2 : communication par tubes nommés (8 points)

Dans ce problème, un processus père communique avec un processus fils de manière bidirectionnelle à l'aide de deux tubes nommés. L'objectif est de réaliser un décompte de 1 à 9 alternativement (le père écrit « 1 » que le fils lit et affiche, puis le fils écrit « 2 » que le père lit et affiche, ainsi de suite).

Le « main » du programme est fourni dans « exo2.c » et on vous invite à ne pas le modifier. Votre travail consistera à implémenter les fonctions de lecture et de lecture d'**un seul** caractère dans le tube, en plus d'une fonction d'initialisation (fichier func2.c).

Voici le résultat attendu de l'exécution :

Sortie terminal

```
$ ./exo2
Début comptage
Fils        : 1
Parent      : 2
Fils        : 3
Parent      : 4
Fils        : 5
Parent      : 6
Fils        : 7
Parent      : 8
Fils        : 9
Fin comptage
```

Fichier func2.h

```
#ifndef DNI
#define DNI

// Création de deux tubes nommés à partir des deux chemins en paramètres
// Attention à verifier que le tube n'existe pas déja (pensez à "unlink)

void init_pipes(char *f1, char *f2);

// lit et retourne un seul caractère à partir du tube en paramètre

char readNextInt(char *f);

// écrit l'entier passé en paramètre dans le tube
// Attention: pensez à convertir l'entier de type "int" en sont équivalent
"char"

void writeNextInt(char *f, int n);



#endif
```

Fichier func2.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


void init_pipes(char *f1, char *f2){

}


char readNextInt(char *f){

}

void writeNextInt(char *f, int n){

}
```

Fichier exo2.c

```
/*************************************
 *******    A NE PAS MODIFIER       *******
*************************************/
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include "func2.h"


int main(int argc, char *argv[]){

      char f1[]="f1",f2[]="f2";

      init_pipes(f1,f2);

      int c;
      pid_t p;
      switch(p= fork()){
            case -1 : perror("fork fail");
            case 0 : {
                  printf("Début comptage\n");
                  writeNextInt(f1,1);
                  do{
                        c = readNextInt(f2) - '0';
                        printf("Parent\t: %d \n", c);
                        c++;
                        writeNextInt(f1,c);
```

```
                }while(c<9);
                exit(0);
        } break;
        default: {
                while(1){
                        c = readNextInt(f1) - '0';
                        printf("Fils  \t: %d \n", c);
                        c++;
                        if(c==10) break;
                        writeNextInt(f2,c);
                }
                kill(p,9);
                printf("Fin comptage\n");
                exit(0);

        }break;
    }

    return 0;
}
```

Fichier Makefile

```
all: exo2 clean

exo2: func2.o main.o
    gcc -o exo2 func2.o main.o

func2.o: func2.c
    gcc -o func2.o -c func2.c -g -Wall

main.o: exo2.c func2.h
    gcc -o main.o -c exo2.c -g -Wall

clean:
    rm -rf *.o
```