Université Paul Sabatier – Toulouse III
IUT A - Toulouse Rangueil

Gestion des processus

28 décembre 2011

Semestre 3

TP nº 1 — Les processus

1 Exercice 1

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

void traitFils1();
void traitFils2();

int main(int argv, char** argc) {
  pid_t idProc ;
  int rapport, numSignal, statut;


  /* processus pere */
  /* creation du fils1 */
  idProc = fork();
  switch (idProc) {
     case -1 :
       perror("echec fork");
       exit(1);
     case 0 :/*  appel du traitement fils1 */
       traitFils1();
       exit(1);
  }

  /* creation du fils2 */
  idProc = fork() ;
  switch (idProc)
  {
     case -1 :
       perror("echec fork");
       exit(1);
     case 0 : /* appel du traitement du fils2 */
       traitFils2();
       exit(1) ;
  }

  /* suite du processus pere */
  /* attente de la terminaison des fils */
  idProc = wait( &rapport ) ;
  while (idProc != -1) {
```

```
42      printf("\nTerminaison du fils de PID= %d\n", idProc);
43      numSignal = rapport & 0x7F ;
44      switch (numSignal) {
45       case 0 :
46         statut = (rapport >>8)& 0xFF ;
47         printf("Fin normale, statut= %d\n", statut);
48       break ;
49       default :
50         printf("Fin anormale, numSignal= %d\n", numSignal);
51      }
52      idProc = wait(&rapport) ;
53    }
54  }
55  /* fin du processus pere */
56  /* traitement du fils1 */
57  void traitFils1() {
58    printf("\n***fils1 --> PID= %d\n", getpid());
59    exit(3);
60  }
61  /* traitement du fils2 */
62  void traitFils2() {
63    int v=10, int* ptr = NULL;
64    printf("\n***fils2 --> PID= %d\n", getpid());
65
66    *ptr = 42;
67  }
```

Listing 1 – Exercice 1 – proc1.c

2 Exercice 2

```
1   #include <sys/types.h>
2   #include <unistd.h>
3   #include <stdlib.h>
4   #include <stdio.h>
5
6   int main(int argc, char** argv) {
7     pid_t idProc ;
8     int err,rapport, numSignal, statut;
9     char * tabParam[3]={"p2fils2", "123456", NULL};
10
11    /* processus pere */
12    /* creation du fils1 */
13    idProc = fork();
14    switch (idProc) {
15       case -1 :
16         perror("echec fork");
17         exit(1);
18       case 0 : /* execution du traitement du fils1 */
19        err=execl("./p2fils1", "p2fils1", "coucou", NULL);
20        if (err == -1) {
21          perror("echec execl");
22          exit(2);
23        }
24    }
25
26    /* creation du fils2 */
```

```
  idProc = fork();
  switch (idProc) {
  case -1 :
    perror("echec fork");
    exit(1);
  case 0 : /* execution du traitement du fils2 */
    err=execvp("p2fils2", tabParam);
    if (err == -1) {
      perror("echec execvp");
      exit(3);
    }
  }

  /* suite du processus pere */
  /* attente de la terminaison des fils */
  idProc = wait( &rapport );
  while ( idProc != -1 ) {
     printf("\nTerminaison du fils de PID= %d\n", idProc);
     numSignal = rapport & 0x7F ;
     switch (numSignal) {
      case 0 :
        statut = ( rapport >> 8 ) & 0xFF ;
        printf("Fin normale, statut= %d\n", statut);
      break ;
      default :
        printf("Fin anormale, numSignal= %d\n", numSignal);
     }
     idProc = wait( &rapport ) ;
   }
}
/* fin du processus pere */
```

Listing 2 – Exercice 2 – Le père

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

/* traitement du fils1 */
int main(int nbParam, char * tabParam[]) {
  char chaine[50+1];
  switch (nbParam) {
    case 2 : /* recuperation du parametre 1 */
      strncpy(chaine, tabParam[1],50);
      chaine[50]='\0';
    break;
    default :
      printf("***fils1 --> nombre de parametres incorrect!!!");
      exit(1);
    }

  printf("\n***fils1 --> PID= %d\n", getpid());
  printf("***fils1 -->tabParam[1]= %s\n", tabParam[1]);
  exit(3);
}
```

Listing 3 – Exercice 2 – Traitement du fils 1

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

int main(int nbParam, char** tabParam) {
  int nbre;
  int* ptr = NULL;

  switch (nbParam) {
    case 2 : /* conversion du parametre 1 */
      sscanf(tabParam[1], "%d",&nbre);
    break;
    default :
      printf("***fils2 --> nombre de parametres incorrect!!!");
      exit(1);
  }

  printf("\n***fils2 --> PID= %d\n", getpid());
  printf("***fils2 --> tabParam[1]= %d\n", nbre);

  *ptr = 42; //woops, une seg fault
}
```

Listing 4 – Exercice 2 – Traitement du fils 2