Valérie de Roquemaurel
valeriederoquemaurel.com Développé par Antoine de Roquemaurel

Systèmes 2 — TD

Semestre 4

Toulouse, le 14 février 2013

Table des matières

Chapitre 1

Introduction

1.1 Exercice 1

```c
#include <stdio.h>

int main(int argc, char** argv) {
  int i;
  for(i=1; i < argc; ++i) {
    printf("%s\n", argv[i]);
  }

  return 0;
}
```

Listing 1.1 – Exercice 1 – Version portable

```c
#define _POSIF_C_SOURCE 1

#include <stdio.h>

int main(int argc, char** argv, char** envp) {
  int i;
  printf("Argument :\n");
  for(i=1; i < argc ; ++i) {
    printf("argv[%d]=%s\n", i, argv[i]);
  }
  i=0;
  while(envp[i] != NULL) {
    printf("%s\n", envp[i++]);
  }

  return 0;
}
```

Listing 1.2 – Exercice 1 – Version Unix

1.2 Exercice 2

```c
#define _POSIX_C_SOURCE 1

#include <stdio.h>
#include <string.h>

int main(int argc, char** argv) {
  if(argv != 1 || !(strlen(argv[1]))) {
    return 1;
  }

  printf("%s = %s\n", argv[1], getenv(argv[1]));
```

```
}
```

Listing 1.3 – Exercice 2

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
  char *valeur;

  if(argc != 2) {
    fprintf(stderr, "Usage : %s variable\n", argv[0]);
    return (1);
  }

  valeur = getenv(argv[1]);

  if(valeur == NULL) {
    fprintf(stderr, "Variable %s inconnue \n", argv[1]);
    return (2);
  }

  printf("%s=%s", argv[1], valeur);

  return 0;
}
```

Listing 1.4 – Exercice 2 – Correction

Chapitre 2

Processus

2.1 Exercice 3

```c
#define _POSIX_C_SOURCE 1
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(int argc, char** argv) {
  pid_t pid;
  switch(pid = fork()) {
    case -1:
      perror("fork");
      exit(1);
    case 0: //fils
      printf("Executé par le fils\n");
      printf("PID du père: %d\n", (int)getppid());
      printf("PID du fils %d\n\n", (int)getpid());
      break;
    default: //père
      printf("Executé par le père\n");
      printf("PID du père: %d\n", (int)getpid());
      break;
  }

  return 0;
}
```

Listing 2.1 – Exercice 3

2.2 Exercice 4

```c
#define _POSIX_C_SOURCE 1
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

#define NBPROC 4 //nombre de processus fils
#define N 5 // nombre d'affichage

void affichage(pid_t pid);

int main(int argc, char** argv) {
  pid_t fils, fils_termine;
  pid_t successeur;
  int i;
```

```
17    int circonstance;
18
19    successeur = getpid();
20
21    for(i = NBPROC; i > 0 ; --i) {
22      switch(fils = fork()) {
23        case -1:
24          perror("Erreur fork");
25          exit(EXIT_FAILURE);
26        case 0:
27          printf("Je suis le fils %d, je m'apelle %d mon successeur est %d\n",
28                 i, (int)getpid(), successeur);
29          affichage(getpid());
30          printf("[%d] fin\n", (int)getpid());
31          exit(i);
32        default:
33          successeur = fils;
34      }
35    }
36
37    printf("Je suis le père, je m'appelle %d mon successeur est %d",
38        (int)getpid(), (int)successeur);
39    while(fils_termine = wait(&circonstance) != -1) {
40      printf("[%d] : Mon fils %d est terminé avec le code %d\n",
41          (int)fils_termine, WEXITSTATUS(circonstance));
42    }
43    printf("Tous les fils sont terminés");
44
45    return EXIT_SUCCESS;
46  }
47
48  void affichage(pid_t pid) {
49    int k;
50    for (k=0 ; k < N ; ++k) {
51      printf("[%d] : c'est moi\n", (int) pid);
52      sleep(2); // processus pas de l'état élu à l'état bloqué
53    }
54  }
```

Listing 2.2 – Exercice 4

2.3 Exercice 5

```
execlp("date", "date", NULL);
```

```
Maintenant:
vendredi 8 février 2013, 15:13:33 (UTC+0100)
```

2.4 Exercice 6

Il va afficher deux fois le pid du programme courant.

2.5 Exercice 7

```
1  #define _POSQIX_C_SOURCE 1
```

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

int main(void) {
  pid_t fils;
  switch ( fils = fork() ) {
    case -1 :
      perror("Erreur fork");
      exit(EXIT_FAILURE);
    case 0 :
      execlp("ls", "ls", "-l", NULL);
      perror("Erreur execp");
      exit(EXIT_FAILURE);
    default :
      wait(NULL);
      break;
  }

  switch ( fils=fork() ) {
    case -1 :
      perror("Erreur fork");
      exit(EXIT_FAILURE);
    case 0 :
      execlp("pwd", "pwd", NULL);
      perror("Erreur execp");
      exit(EXIT_FAILURE);
    default :
      wait(NULL);
      break;
  }

  return EXIT_SUCCESS;
}
```

Listing 2.3 – Exercice 7

Chapitre 3

Gestion de la mémoire : mémoire virtuelle et allocation non contiguë

3.1 Exercice 8

3.1.1

- **Pages 2 et 3** $[5 \times 1024; 6 \times 1024 - 1] = [2048; 4095]$
- **Page 5** $[7 \times 1024; 8 \times 1024 - 1] = [5120; 6143]$
- **Page 7** $[2 \times 1024; 4 \times 1024 - 1] = [7168; 8191]$

3.1.2

R Numéro de cadre × 1024 + décalage

Adresse virtuelle	Page virtuelle	Page physique	Adresse physique
0	0	3	$3 \times 1024 = 3072$
3728	3	défaut de page	défaut de page
1023	0	3	$3 \times 1024 + 1023 = 4095$
1024	1	1	1024
1025	1	1	1025
7800	7	défaut de page	défaut de page
4096	4	2	2048

3.2 Exercice 9

d : taux de défaut de page.

3.2.1 Temps d'accès moyen

$$(1-d) \times 2 \times 0.5 + d \times (20 + 2 \times 0.5) = (1-d) + 21d = 1 + 20d$$

3.2.2 Taux maximal de défaut de page

$$\begin{aligned} 1 + 20d &< 1.2 \\ d &< \frac{0.2}{20} \\ d &< 0.01 \end{aligned}$$

3.3 Exercice 10

3.3.1

Chaque processus provoque 5 défauts de page correspondants aux changements initiaux des 5 pages.

Soit n le nombre de page référencancés par un processus. Le taux de défaut de page est $\frac{5}{n}$.

Chapitre 4

Structure interne du système de fichier d'Unix

Chapitre 5

Primitives Unix (POSIX.1) de manipulation des fichiers