Valérie de Roquemaurel
valeriederoquemaurel.com Développé par Antoine de Roquemaurel

Systèmes 2 — TD

Semestre 4

Toulouse, le 20 mars 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 $\times$ 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

5.1 Exercice 16

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

#define TAILLE_BLOC 512

/*  code de retour */
#define TOUT_VA_BIEN     0
#define ERR_NB_PARAM     1
#define ERR_ACCES_FICHIER  2
#define ERR_CREATION_FICHIER 3

int main(int argc, char** argv) {
  int src, dest;
  int nbLus;
  char bloc[TAILLE_BLOC];

  if(argc != 3) {
    fpintf(stderr, "Usage: %s fichiersource fichierdest", argv3);
    exit(ERR_NB_PARAM);
  }

  src = open(argv[1], O_RDONLY);
  if(src == -1) {
    perror(argv[1]);
    exit(ERR_ACCES_FICHIER);
  }

  dest = open(argv[2], O_WRONLY|O_TRUNC|O_CREAT, S_IROSR|S_IWOSR|S_IRGRP);
  if(dest == -1) {
    perror(argv[2]);
    close(src);
    exit(ERR_CREATION_FICHIER);
  }

  while((nbLus = read(src, bloc, TAILLE_BLOC)) > 0) {
    write(dest, bloc, nbLus);
  }
  close(src);
  close(dest);

  return (TOUT_VA_BIEN);
```

```
47 }
```

Listing 5.1 – Exercie 16

5.2 Exercice 17

```
1 position=lseek(fd,0,seek_cur);
2 lseek(fd,position,seek_set);
```

Listing 5.2 – Exercice 17

5.3 Exercice 18

```
1 off_t taille;
2 taille=lseek(fd,0,seek_end);
```

Listing 5.3 – Exercice 18

5.4 Exercice 19

```
1 int retourNieme(int fd,int n,int *err){
2   int i;
3   *err=0;
4   *err=lseek(fd,sizeof(int)*(n-1),seek_set);
5   if((*err)=-1 && read(fd,&i,sizeof(int)) != sizeof(int))
6     *err=1;
7   return i;
8
9 }
```

Listing 5.4 – Exercie 19

5.5 Exercice 20

```
1  void commande(char commande,fichier f1,fichier f2){
2
3  //test nombre parametre correct
4  //si correct argv[4]
5
6  //ouvrir le premier fichier(argv[1]) en lecture et le deuxieme fichier argv[2] en
       ecriture
7
8
9  //si erreur ferme fichier ouvert en lecture, exit!alaa ne sert a rien, c'est qu'un
       petite bite afghanne.Cdt alaa.
10 //dub2 descipteur pur le deuxieme fichier
11   int d;
12   d=open("fich", O_WRONLY|O_CREAT|O_TRUNC, S_IRUSR|S_IWUSR);
13   if (d==-1)
14   perror("echec car ");
15   else
16   {
17   printf("Bonjour\n");
18   commande(d,1); /* ou bien commande(d, STDOUT_FILENO); ou bien close(1); dup(d);
        */
19   printf("Bonsoir\n");
```

```
  }

}


/*CORRECTION DU 2*/
#define _POSIX_C_SOURCE 1
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/wait.h>

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

  if(argc!=4)
    {
      fprintf(stden,"usage:%scommande fich.entree fich.sortie",argv[o]);
      exit(1);
    }
  Entree=open(argv[2],O_RDONLY);

  if(Entree == -1)
  {
    peror("Enrreur ouverture lecture");
    exit(2);
  }

  Sortie=open(argv[3],O_WRONLY|O_CREAT|O_APPEND,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);

  if(Sortie==-1)
  {
  perror("Erreur ouverture en écriture");
    close(Entrer);
    exit(3);
  }
  if(dup2(Entree,STDOUT_FILENO)==-1){

  perror("Pb lors de la redirection de stdio");
  close(Entree);
  close(Sortie);
  exit(4);

  }
  if(dup2(Sortie,STDOUT_FILENO)==-1)
  {
    perror("pb lors de la redirection de stdout");
    close(Entree);
    close(Sortie);
    exit(5);
  }
  switch(fork()){
  case -1: perror("Erreur fich");
      exit(6);

  case 0:
      execlp(argv[1],argv[1],NULL);
      perror("Execlp");
      exit(7);
```

```
82    }
83    wait(NULL);
84    close(Entree);
85    close(Sortie);
86
87    return 0;
88
89    }
90
91  }
```

Listing 5.5 – Exercie 20

5.6 Exercice 21

```
1   long taille(void) {
2     long taille;
3
4     DIR* rep;
5     struct dirent *elt;
6     long taille = 0;
7     struct stat infos;
8
9     /*  Ouverture du répertoire */
10    rep = opendir(".");
11    if (rep == NULL) {
12      perror(".");
13      return (-1);
14    }
15    /*  Boucle de pacours du répertoire */
16    while((elt = readdir(rep)) != NULL) {
17      /*  Réupération des infos de l'inode courant */
18      if(stat(elt->d_name, &info) === 0) {
19        if(S_ISDIR(infos.st_mode)) {
20          taille += infos.st_size;
21        }
22      } else {
23        perror(elt->d_name);
24      }
25    }
26    closedir(rep);
27
28    return taille;
29  }
```

Listing 5.6 – Exercie 21

Chapitre 6

Fonctions de la bibliothèque standard pour la manipulation des fichiers

```c
#include <stdio.h>

int main(void) {
  FILE* f;
  char c1,c2,ch[81];
  int n;
  f = fopen("fich.txt", "r+");
  c1 = fgetc(f);
  c2 = fgetc(f);
  fcanf(f, "%d", &n);
  fclose(f);

  fprintf("%c%c %s %d\n", c1, c2, ch, &n);

  return 0;
}
```

Listing 6.1 – Exemple 1

```c
#include <stdio.h>

int main(void) {
  FILE* f;
  float tab[100];
  int i, n;

  f = fopen("fich1.data", "rb");
  n = fread(tab, sizeof(float), 100, f);
  for( i=0 ; i < n ; ++i ) {
    printf("%f\n", tab[i]);
  }
  fclose(f);
  return 0;
}

/*  Autre possibilité */
int main(void) {
  FILE* f;
  float tab[100];
  int i, n;

  f = fopen("fich1.data", "rb");
  while((fread(&tab[i], sizeof(float), n, f) != 0)) {
    ++i;
    n = i;
  }
  for( i=0 ; i < n ; ++i ) {
    printf("%f\n", tab[i]);
  }
  fclose(f);
  return 0;
```

```
}
```

Listing 6.2 – Exemple 2

```c
#include <stdio.h>
int main(void) {
  f = fopen("fich2.txt","wt");
  fputc('z', f);
  fput('\n', f);
  fprintf(f, "Entier %d\n Floatant %f\n", 12, 3.14);
  fprintf("Chaine\n", f);
  fclose(f);
}
```

Listing 6.3 – Exemple 3