

# Systemes 2 — TD

---

Semestre 4



---

# Table des matières

---

---

# Introduction

---

## 1.1 Exercice 1

```
1 #include <stdio.h>
2
3 int main(int argc, char** argv) {
4     int i;
5     for(i=1; i < argc; ++i) {
6         printf("%s\n", argv[i]);
7     }
8
9     return 0;
10 }
```

Listing 1.1 – Exercice 1 – Version portable

```
1 #define _POSIX_C_SOURCE 1
2
3 #include <stdio.h>
4
5 int main(int argc, char** argv, char** envp) {
6     int i;
7     printf("Argument :\n");
8     for(i=1; i < argc ; ++i) {
9         printf("argv [%d]=%s\n", i, argv[i]);
10    }
11    i=0;
12    while(envp[i] != NULL) {
13        printf("%s\n", envp[i++]);
14    }
15
16    return 0;
17 }
```

Listing 1.2 – Exercice 1 – Version Unix

## 1.2 Exercice 2

```
1 #define _POSIX_C_SOURCE 1
2
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(int argc, char** argv) {
7     if(argv != 1 || !(strlen(argv[1]))) {
8         return 1;
9     }
10
11     printf("%s = %s\n", argv[1], getenv(argv[1]));
```

12 | }

## Listing 1.3 – Exercice 2

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char** argv) {
5      char *valeur;
6
7      if(argc != 2) {
8          fprintf(stderr, "Usage : %s variable\n", argv[0]);
9          return (1);
10     }
11
12     valeur = getenv(argv[1]);
13
14     if(valeur == NULL) {
15         fprintf(stderr, "Variable %s inconnue \n", argv[1]);
16         return (2);
17     }
18
19     printf("%s=%s", argv[1], valeur);
20
21     return 0;
22 }
```

## Listing 1.4 – Exercice 2 – Correction

# Processus

## 2.1 Exercice 3

```
1 #define _POSIX_C_SOURCE 1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6
7 int main(int argc, char** argv) {
8     pid_t pid;
9     switch(pid = fork()) {
10         case -1:
11             perror("fork");
12             exit(1);
13         case 0: //fils
14             printf("Exécuté par le fils\n");
15             printf("PID du père: %d\n", (int)getppid());
16             printf("PID du fils %d\n\n", (int)getpid());
17             break;
18         default: //père
19             printf("Exécuté par le père\n");
20             printf("PID du père: %d\n", (int)getpid());
21             break;
22     }
23
24     return 0;
25 }
```

Listing 2.1 – Exercice 3

## 2.2 Exercice 4

```
1 #define _POSIX_C_SOURCE 1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <sys/types.h>
5 #include <unistd.h>
6 #include <sys/wait.h>
7
8 #define NBPROC 4 //nombre de processus fils
9 #define N 5 // nombre d'affichage
10
11 void affichage(pid_t pid);
12
13 int main(int argc, char** argv) {
14     pid_t fils, fils_termine;
15     pid_t successeur;
16     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'appelle %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 _POSIX_C_SOURCE 1

```

```
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(void) {
9     pid_t fils;
10    switch ( fils = fork() ) {
11        case -1 :
12            perror("Erreur fork");
13            exit(EXIT_FAILURE);
14        case 0 :
15            execlp("ls", "ls", "-l", NULL);
16            perror("Erreur execlp");
17            exit(EXIT_FAILURE);
18        default :
19            wait(NULL);
20            break;
21    }
22
23    switch ( fils=fork() ) {
24        case -1 :
25            perror("Erreur fork");
26            exit(EXIT_FAILURE);
27        case 0 :
28            execlp("pwd", "pwd", NULL);
29            perror("Erreur execlp");
30            exit(EXIT_FAILURE);
31        default :
32            wait(NULL);
33            break;
34    }
35
36    return EXIT_SUCCESS;
37 }
```

Listing 2.3 – Exercice 7



# 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  $\times 1024 + \text{décalage}$

| 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  $\frac{5}{n}$ .

# Structure interne du système de fichier d'Unix

---

4

# Primitives Unix (POSIX.1) de manipulation des fichiers

## 5.1 Exercice 16

```
1 #define POSIX_C_SOURCE 1
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8
9 #define TAILLE_BLOC 512
10
11 /* code de retour */
12 #define TOUT_VA_BIEN 0
13 #define ERR_NB_PARAM 1
14 #define ERR_ACCES_FICHIER 2
15 #define ERR_CREATION_FICHIER 3
16
17 int main(int argc, char** argv) {
18     int src, dest;
19     int nbLus;
20     char bloc[TAILLE_BLOC];
21
22     if(argc != 3) {
23         fprintf(stderr, "Usage: %s fichiersource fichierdest", argv3);
24         exit(ERR_NB_PARAM);
25     }
26
27     src = open(argv[1], O_RDONLY);
28     if(src == -1) {
29         perror(argv[1]);
30         exit(ERR_ACCES_FICHIER);
31     }
32
33     dest = open(argv[2], O_WRONLY|O_TRUNC|O_CREAT, S_IRUSR|S_IWUSR|S_IRGRP);
34     if(dest == -1) {
35         perror(argv[2]);
36         close(src);
37         exit(ERR_CREATION_FICHIER);
38     }
39
40     while((nbLus = read(src, bloc, TAILLE_BLOC)) > 0) {
41         write(dest, bloc, nbLus);
42     }
43     close(src);
44     close(dest);
45
46     return (TOUT_VA_BIEN);
```

47 | }

Listing 5.1 – Exercice 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 – Exercice 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 descripteur 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");

```

```
20     }
21
22 }
23
24
25 /*CORRECTION DU 2*/
26 #define _POSIX_C_SOURCE 1
27 #include <stdio.h>
28 #include <stdlib.h>
29 #include <unistd.h>
30 #include <fcntl.h>
31 #include <sys/stat.h>
32 #include <sys/wait.h>
33
34 int main(int argc, char * argv[])
35 {
36     int Entree, Sortie;
37
38     if(argc!=4)
39     {
40         fprintf(stderr, "usage:%scommande fich.entree fich.sortie", argv[0]);
41         exit(1);
42     }
43     Entree=open(argv[2], O_RDONLY);
44
45     if(Entree == -1)
46     {
47         perror("Erreur ouverture lecture");
48         exit(2);
49     }
50
51     Sortie=open(argv[3], O_WRONLY|O_CREAT|O_APPEND, S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
52
53     if(Sortie==-1)
54     {
55         perror("Erreur ouverture en écriture");
56         close(Entree);
57         exit(3);
58     }
59     if(dup2(Entree, STDOUT_FILENO)==-1){
60
61         perror("Pb lors de la redirection de stdout");
62         close(Entree);
63         close(Sortie);
64         exit(4);
65
66     }
67     if(dup2(Sortie, STDOUT_FILENO)==-1)
68     {
69         perror("pb lors de la redirection de stdout");
70         close(Entree);
71         close(Sortie);
72         exit(5);
73     }
74     switch(fork()){
75     case -1: perror("Erreur fich");
76             exit(6);
77
78     case 0:
79         execlp(argv[1], argv[1], NULL);
80         perror("Execlp");
81         exit(7);
```

```

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

```

Listing 5.5 – Exercice 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 – Exercice 21

# Fonctions de la bibliothèque standard pour la manipulation des fichiers

```

1 #include <stdio.h>
2
3 int main(void) {
4     FILE* f;
5     char c1,c2,ch[81];
6     int n;
7     f = fopen("fich.txt", "r+");
8     c1 = fgetc(f);
9     c2 = fgetc(f);
10    fcanf(f, "%d", &n);
11    fclose(f);
12
13    fprintf("%c%c %s %d\n", c1, c2, ch, &n);
14
15    return 0;
16 }
```

Listing 6.1 – Exemple 1

```

1 #include <stdio.h>
2
3 int main(void) {
4     FILE* f;
5     float tab[100];
6     int i, n;
7
8     f = fopen("fich1.data", "rb");
9     n = fread(tab, sizeof(float), 100, f);
10    for( i=0 ; i < n ; ++i ) {
11        printf("%f\n", tab[i]);
12    }
13    fclose(f);
14    return 0;
15 }
16
17 /* Autre possibilité */
18 int main(void) {
19     FILE* f;
20     float tab[100];
21     int i, n;
22
23     f = fopen("fich1.data", "rb");
24     while((fread(&tab[i], sizeof(float), n, f) != 0)) {
25         ++i;
26         n = i;
27     }
28     for( i=0 ; i < n ; ++i ) {
29         printf("%f\n", tab[i]);
30     }
31     fclose(f);
32     return 0;
33 }
```



## Listing 6.2 – Exemple 2

```
1 #include <stdio.h>
2 int main(void) {
3     f = fopen("fich2.txt", "wt");
4     fputc('z', f);
5     fput('\n', f);
6     fprintf(f, "Entier %d\n Floatant %f\n", 12, 3.14);
7     fprintf("Chaine\n", f);
8     fclose(f);
9 }
```

## Listing 6.3 – Exemple 3