R. Namyst
LaBRI, bureau 322

le 25 avril 2006

Licence informatique — Programmation Système
Correction du Devoir Surveillé du 24 avril 2006

1 Gestion des processus (question d'échauffement)

Un *zombie* est un processus terminé dont la mort n'a pas encore été « récupérée » par son père (au moyen de `wait` ou `waitpid`). A ce stade, le processus ne peut plus s'exécuter mais son bloc de contrôle est conservé dans la table des processus, jusqu'à ce que le père (ou le processus `init`) récupère les diverses informations statistiques qu'il contient lors de l'opération `wait`...

Lorsque le nombre de *zombies* devient grand, on risque une saturation de la table des processus. Voici un code provoquant une telle situation :

```
while(fork() != 0) ;
```

2 Communication par tubes

Question 1

Voici le code remplaçant les lignes 14 et 15 :

```
lseek(fd, atoi(argv[2])*sizeof(int));
len = atoi(argv[3])*sizeof(int);
```

Question 2

```
void merge(int fd1, int fd2, int fdout)
{
  int n1, n2, i1, i2;

  n1 = read(fd1, &i1, sizeof(int));
  n2 = read(fd2, &i2, sizeof(int));

  while((n1 != 0) || (n2 != 0)) {
    if(n1 == 0) {
      write(fdout, &i2, sizeof(int));
      n2 = read(fd2, &i2, sizeof(int));
    } else if((n2 == 0) || (i1 < i2)) {
      write(fdout, &i1, sizeof(int));
      n1 = read(fd1, &i1, sizeof(int));
    } else {
      write(fdout, &i2, sizeof(int));
      n2 = read(fd2, &i2, sizeof(int));
    }
  }
}
```

Question 3

```
int main(int argc, char *argv[])
{
  int tube1[2], tube2[2];

  pipe(tube1);
  if(fork() == 0) {
    close(tube1[0]);
    // 1er fils
  } else {
    close(tube1[1]);
    pipe(tube2);
    if(fork() == 0) {
      close(tube2[0]); close(tube1[0]);
      // 2e fils
    } else {
      close(tube2[1]);
      // pere
    }
  }
  exit(0);
}
```

Question 4 Il suffit de

1. placer au début du programme

```
struct stat file_attributes;
int len, fd = open(argv[1], O_RDONLY);

fstat (fd, &file_attributes);
close (fd);
len = file_attributes.st_size / sizeof(int);
```

2. remplacer `// 1er fils` par

```
{
  char nb[10];

  sprintf(nb, "%d", len/2);
  dup2(tube1[1], STDOUT_FILENO);
  execlp("trier", "trier", argv[1], "0", nb, NULL);
}
```

3. remplacer `// 2e fils` par

```
{
  char min[10], nb[10];

  sprintf(min, "%d", len/2);
  sprintf(nb, "%d", len - len/2);
  dup2(tube2[1], STDOUT_FILENO);
  execlp("trier", "trier", argv[1], min, nb, NULL);
}
```

4. remplacer `// pere` par

```
merge(tube1[0], tube2[0], STDOUT_FILENO);
```

Question 5

Voici, de manière synthétique, ce qu'il faut modifier dans le programme `fusion` :

```
int main()
{
   // argv[1] = <fichier>
   // argv[2] = nb-etages
   // argv[3] = borne inf
   // argv[4] = nb
   if(argv[3] == "")
      nb = <taille du fichier calculee avec fstat>;

   if(nb-etage == 0)
      exec("trier", argv[1], argv[3], argv[4]);
   else {
      // placer ici le code de la version antérieure et modifier
      // 1er fils et 2e fils de manière à exécuter respectivement
      exec("fusion", argv[1], argv[2]-1, argv[3], argv[4]/2);
      // et
      exec("fusion", argv[1], argv[2]-1, argv[3]+argv[4]/2, argv[4]/2);
   }
}
```