R. Namyst
LaBRI, bureau 322

le 25 avril 2006

Licence informatique — Programmation Système
Devoir Surveillé

Durée : 1h30 — Sans document

1 Gestion des processus (question d'échauffement)

Qu'est-ce qu'un processus zombie ? Pourquoi a-t-on eu besoin d'introduire un tel "état" ? Que risque-t-on lorsque le nombre de zombies devient grand ? Donnez un petit exemple de programme C provoquant cette situation à coup sûr.

2 Communication par tubes

On dispose d'un programme `trier` capable de trier un fichier d'entiers et d'envoyer le résultat (dans le même format) sur sa sortie standard. Les entiers sont stockés dans le même format que le format binaire natif du processeur. Pour afficher le résultat sous forme lisible, on utilise donc un programme `afficher` qui imprime sous forme décimale les entiers lus sur l'entrée standard.

```
> cat fichier | afficher
7 2 6 8 1
> trier fichier | afficher
1 2 6 7 8
```

trier.c

```c
// La fonction suivante permet de trier un tableau d'entiers.
// On ne demande pas d'écrire son code.
extern void sort(int *array, unsigned nb_integers);

int main(int argc, char *argv[])
{
  int fd;
  unsigned len;
  int *buf;
  struct stat file_attributes;

  fd = open(argv[1], O_RDONLY);

  fstat (fd, &file_attributes);
  len = file_attributes.st_size;

  buf = malloc(len);
  read(fd, buf, len);
  sort(buf, len / sizeof(int));
  write(STDOUT_FILENO, buf, len);

  return 0;
}
```

Question 1

On souhaite modifier `trier` pour ne prendre en compte qu'une sous-partie du fichier. Pour cela, on ajoute deux arguments qui représentent respectivement *la position du premier entier* du segment à considérer et *le nombre d'entiers* à traiter. Voici deux exemples illustrant le nouveau fonctionnement de `trier` :

```
> trier fichier 0 3 | afficher
2 6 7
> trier fichier 3 2 | afficher
1 8
```

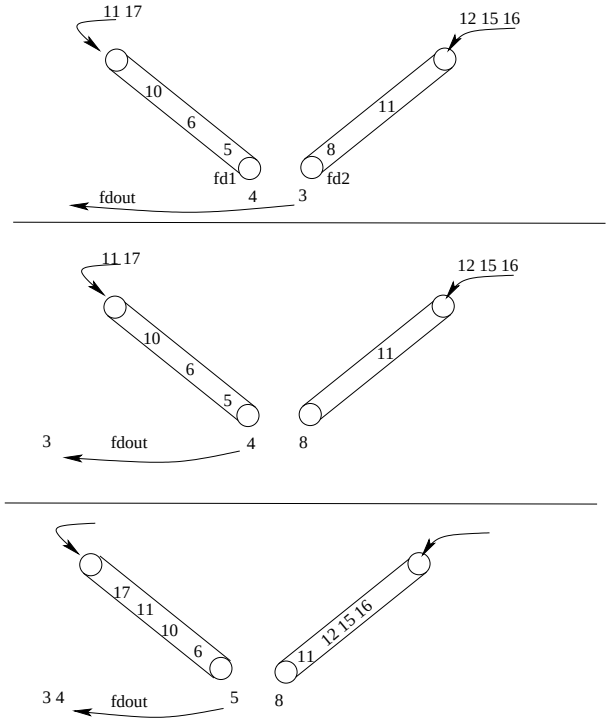

FIG. 1 – Principe du tri fusion avec 2 tubes

Donnez le code remplaçant les lignes 14 et 15 afin de prendre en compte les deux nouveaux paramètres. On pourra utiliser les fonctions bien connues `off_t lseek(int fd, off_t offset, int whence)` (où `whence` prendra une valeur dans `SEEK_SET`, `SEEK_CUR`, `SEEK_END`) et `int atoi(char *s)`.

Question 2 On peut remarquer que si l'on trie *en parallèle* la première et la seconde moitié du fichier, alors il n'y a plus qu'à effectuer une simple *fusion* des résultats pour obtenir la version triée de l'intégralité du fichier. On se propose d'utiliser ce principe pour accélérer le tri d'un fichier sur une machine multi-processeurs. On utilisera pour cela trois processus : deux effectueront chacun le tri d'une moitié du fichier, le troisième effectuera la fusion des résultats qu'il recevra via deux tubes depuis les processus trieurs. Ce fonctionnement est illustré en figure 1.

Écrivez le code d'une fonction `void merge(int fd1, int fd2, int fdout)` qui lit deux suites d'entiers triés depuis les descripteurs `fd1` et `fd2` (extrémités de tubes en lecture) et qui envoie, au fur et à mesure, le résultat de la fusion sur la sortie désignée par `fdout`. Attention à bien gérer le fait que l'extraction des données ne progressera pas forcément de manière uniforme entre les deux tubes.

Question 3 Ecrivez un squelette de programme `fusion` créant deux processus fils, chacun connectés par un tube distinct au processus père. Les fils doivent uniquement pouvoir écrire dans le tube, et le père ne doit pouvoir que lire. Toutes les extrémités de tubes non utiles doivent être fermées. Pour cette question, les processus ne font aucun traitement utile : ils appellent `exit` sitôt la mise en place des tubes terminée.

Question 4 Enrichissez le programme `fusion` pour que les fils exécutent le programme `trier`, chacun avec des paramètres différents de sorte que l'un trie la première moitié du fichier et l'autre la seconde. Le père, quant à lui, pourra utiliser la fonction `merge` pour réaliser le tri complet. Le résultat sera envoyé sur la sortie standard. On supposera que le nom du fichier à trier est passé en paramètre du programme.

Note : on pourra utiliser la fonction `int execvp(char *file, char *argv[])`.

Question 5 Sur une machine disposant de beaucoup de processeurs, on pourrait souhaiter utiliser davantage de processus `trier` en parallèle pour accélérer encore le tri. Une façon simple de procéder est de lancer un arbre binaire de processus dont les feuilles seraient des instances de `trier` et dont les nœuds intermédiaires seraient des instances de `fusion`. Pour simplifier, `fusion` pourrait prendre en second paramètre le nombre d'étages de l'arbre à déployer.

Indiquez ce qu'il faudrait changer au programme fusion pour qu'il puisse fonctionner de cette manière.