Licence Informatique — mai 2019

UNIVERSITÉ DE LORRAINE

Epreuve - Système 1

Durée : 2h — **Aucun document ni équipement électronique n'est autorisé**

Le barème n'est qu'indicatif. Les exercices peuvent être faits dans n'importe quel ordre.

Exercice 1 - Pipe'n fork ..... 6 points

Un étudiant de Licence a réalisé un petit programme.

```
01. int main(int argc, char *argv[]) {
02.  int tube[2];
03.  pid_t pid1, pid2;
04.  char param1[10]=''truc'';
05.
06.  /* Rappel : tube[0] est utilisé pour la lecture et tube[1] pour l'écriture*/
07.  if (pipe(tube) != 0) {
08.    perror("création du tube anonyme");
09.    exit(1);
10.  }
11.  switch(pid1 = fork()) {
12.     case -1:
13.       perror("erreur fork 1");
14.       exit(-1);
15.     case 0:
16.       close(STDOUT_FILENO);
17.       dup2(tube[1], STDOUT_FILENO);
18.       close(tube[0]);
19.       close(tube[1]); // <---- question d)
20.       execl("./commande1", "commande1", param1, NULL);
21.       exit(2);
22.     default:
23.      break ;
24.  }
25.  switch(pid2 = fork()) {
26.     case -1:
27.       perror("erreur fork 2");
28.       exit(-1);
29.     case 0:
30.       close(STDIN_FILENO);
31.       dup2(tube[0],STDIN_FILENO );
32.       close(tube[0]);
33.       close(tube[1]);  // <---- question d)
34.       execl("./commande2", "commande2", NULL);
35.       exit(3);
36.     default:
37.      break ;
38.  }
39.  close(tube[0]);
40.  close(tube[1]); // <---- question d)
41.  wait(NULL);
42.  wait(NULL);
43.  return(0);
44. }
```

a) Plusieurs processus semblent avoir été créés.

1. Combien en tout ? (dessinez-les sous forme d'un graphe père/fils)
2. A quel moment l'instruction "exit(2)" peut-elle être exécutée ?

b) Le manuel nous indique que la fonction *dup2(int descr1, int descr2)* duplique le descripteur *descr1* dans le descripteur *descr2*, *descr2* pointe alors vers le même fichier ouvert que *descr1*. STDOUT_FILENO et STDIN_FILENO sont les descripteurs associés à la sortie et à l'entrée standard.
Expliquez clairement (avec un(des) schéma(s) si besoin) ce qui est réalisé avec les deux lignes 16 et 17 (`close(STDOUT_FILENO);dup2(tube[1], STDOUT_FILENO);`)
Même question pour les lignes 30. et 31.

c) Quel est le but de ce programme (= il sert à quoi) ? Donnez la ligne de commande en Shell qui est équivalente.

d) L'étudiant a bien pris soin de mettre "`close(tube[1]);`" à plusieurs endroits (indiqués par : "`// <-- question d)`"). Pourquoi est-ce si important et que se passerait-il s'il ne le faisait pas ? (vous pouvez illustrer votre propos avec un schéma).

Exercice 2 - Gestion de fichiers ..... 6.5 points

a) Les systèmes de fichiers utilisent la notion de *bloc* (ou cluster) pour gérer l'espace d'un disque. Donnez un avantage du choix de blocs de petite taille (contrairement à un bloc de grand taille). Même question pour le choix de blocs de grande taille.

b) Rappelez l'utilité d'un *Buffer cache* (ou tampon) pour le gestion de l'écriture/lecture de fichiers.

c) Les systèmes de gestion de fichiers utilisés par les systèmes de type Linux utilisent la notion d'*i-node* (ou i-noeud). Soit un système Linux gérant des blocs disque de 16 Ko, une adresse de bloc est codée sur 4 octets. Combien de place (en octets) prendra réellement un fichier de 32 Mo de données ? Expliquez comment vous obtenez ce nombre d'octets sans oublier d'expliquer ce qu'est un i-node (vous pouvez utiliser un schéma).

d) L'ouverture du fichier `/home/etud/kev23/tmp/essai` nécessite plusieurs accès disque pour lire des blocs d'i-nodes et de répertoires. Calculez le nombre d'accès disque pour récupérer l'i-node du fichier `essai` en supposant que l'i-node de la racine est toujours en mémoire et que les répertoires ont une taille de 1 bloc. Vous rappellerez la structure d'un répertoire sous Linux.

Exercice 3 - Allocation mémoire ..... 5.5 points

Lorsque l'on charge un processus, il faut lui allouer une place mémoire.

a) Premier type d'allocation : Les listes de portions d'espace libre. On considère à l'instant initial les portions de mémoire libre suivantes (en Mo) 100, 500, 200, 300 et 600. Les processus sont chargés successivement et sont de tailles respectives (en Mo) de 212, 417, 112 et 426.

1. Donnez l'attribution de la mémoire selon les 3 algorithmes first-fit, best-fit et worst-fit.
2. Quel est le meilleur algorithme ici ? (justifiez votre réponse)

b) Deuxième type d'allocation : le principe de blocs compagnons (*Buddy system*). Sur un système doté de 1 Go de mémoire et qui utilise ce principe, les processus font successivement les requêtes suivantes (en Mo) : 150, 70, 260, 40, 80, 180, 50, 120, 60. Rappel : Un bloc peut se couper en 2 blocs égaux et un morceau de mémoire ne peut pas être sur deux blocs ni compléter un bloc déjà partiellement utilisé. La mémoire démarre avec le gros bloc de 1Go.

1. Quelle est la première requête à échouer dans la chaine des requêtes, en raison d'un manque de mémoire disponible ? Faites un schéma.
2. Au moment de l'échec de la requête, combien reste-t-il de mémoire libre et combien de mémoire est gaspillée ?

Exercice 4 - Principe de localité ..... 2 points

Dans un système, on parle souvent de *principe de localité*. Donnez un exemple de prise en compte du principe

- de localité temporelle dans la gestion de la mémoire.
- de localité physique dans la gestion des fichiers.

Dans les deux exemples, vous expliquerez en quoi consiste le principe de localité et comment il est mis en oeuvre (c-a-d quel algo est utilisé et en quoi il met en oeuvre ce principe).