Systèmes d'exploitation centralisés

Formation par apprentissage, première session
(Durée 1h45, les documents distribués en cours et TD sont autorisés)
La question 3 de shell sera rendue sur une feuille séparée

Juin 2018

1 Mise en œuvre Unix : Processus et fichiers (16 points)

On considère le programme suivant[1] :

```
1  void Affichage (int sig)
2  {
3       if (sig==SIGUSR1)  /* SIGUSR1=10*/
4       { printf ("Message 1, X1=%d a recu Y1=%d\n", (int) getpid(), sig); }
5       else if (sig==SIGUSR2) /*SIGUSR2=12*/
6       { printf ("Message 2, X2=%d a recu Y2=%d\n", (int) getpid(), sig); }
7       else
8       { printf ("Message 3, X3=%d a recu Y3=%d\n", (int) getpid(), sig); }
9  }
10 int main(int argc ,char *argv[])
11 {
12   int p[2], ret;
13   pid_t pid0, pid1, pid2, pid3, pid4, pid5;
14   int tab[2], nb1, nb2, i, j;
15
16   pipe(p);
17   pid0=getpid();
18   printf (" ... pid0= %d\n", (int) pid0);
19
20   signal(SIGUSR1,Affichage);
21   signal(SIGUSR2,Affichage);
22
23   if ( (pid1=fork()) == 0 )
24   {
25      int n_in, k, n_out;
26      pid5 = getppid();
27      printf ("Message 4, X4=%d, Y4=%d\n", (int) getpid(), pid5);
28      pid2 = fork();
29
30      k    = 3;
31      if (pid2!=0)
32       { k=k-1; printf (" ... pid2= %d, Y5=%d\n", (int) pid2,k);}
33      else
34       { printf (" ... X6= %d, Y6=%d\n", (int) getpid(),k);}
35
36      read(p[0], &n_in, sizeof(int));
37      close(p[0]);
38      n_out = k*n_in;
39      printf(" ... pid %d n_in=%d X7=%d\n", (int) getpid(), n_in, n_out);
40      write(p[1], &n_out, sizeof(int));
41      if (pid2!=0)
42      {
43        close(p[1]);
44        pid3=wait(&ret);
45        if (WIFSIGNALED(ret))
46        { printf("Message 5, X8=%d et Y8=%d \n", pid3, WTERMSIG(ret)); }
47        else
48        { printf("Message 6, X9=%d et Y9=%d \n", pid3, WEXITSTATUS(ret)); }
49        exit(2);
50      }
51      else
52      {
53        kill(pid2,SIGUSR2);
54        close(p[1]);
55        exit(3);
56      }
57   } else
58     {
59      printf (" ... pid1= %d\n", (int) pid1);
60      kill(pid1,SIGUSR1);
61      nb1=30;
62      nb2=50;
63      write(p[1], &nb1, sizeof(int));
64      write(p[1], &nb2, sizeof(int));
65      close(p[1]);
66      i = 0;
67      while ( read(p[0], &tab[i], sizeof(int)) > 0 ) /* Ligne 77*/
68      { printf(" ... MESSAGE Valeur TAB: i=%d, tab[i]=%d\n", i,tab[i]);
69        i++; }
70      close(p[0]);
71      pid4=wait(&ret);
72      if (WIFSIGNALED(ret))
73      { printf("Message 7, X10=%d et Y10=%d \n", pid4, WTERMSIG(ret)); }
74      else
75      { printf("Message 8, X11=%d et Y11=%d \n", pid4, WEXITSTATUS(ret)); }
76      /* Ligne 76 */
77    }
78    return(0);
79  }
```

1. Les directives d'inclusion ne sont pas précisées mais supposées présentes.

Questions (une réponse non argumentée est sans valeur)

1. (1,5pts) Précisez les processus et le schéma de communication par pipe engendrés par ce programme lors de son exécution. Faire un schéma précisant les canaux de communication utilisés.
2. (2pts) Lors d'une exécution du programme, on obtient la sortie à compléter suivante. Compléter dans les messages affichés, les valeurs de X et Y.

```
... pid0= 31253
... pid1= 31254
Message 1, X1=??? a recu Y1=???
Message 4, X4=???, Y4=???
... pid2= 31255, Y5=???
... pid 31254 n_in=30 X7=???
... X6= ???, Y6=???
... pid 31255 n_in=50 X7=???
Message 2, X2=??? a recu Y2=???
... MESSAGE Valeur TAB: i=0, tab[i]=60
... MESSAGE Valeur TAB: i=1, tab[i]=150
Message 6, X9=??? et Y9=???
Message 8, X11=??? et Y11=???
```

3. (2pts) Décrire l'ordre d'exécution des instructions ayant permis d'obtenir

```
... MESSAGE Valeur TAB: i=0, tab[i]=60
... MESSAGE Valeur TAB: i=1, tab[i]=150
```

4. (1,5pts) Expliquer pourquoi "MESSAGE Valeur TAB" est toujours écrit deux fois quelque soit l'ordre d'exécution des processus
5. (2pts) Lors d'une exécution de ce programme, le processus pid1 lit sur le tube la valeur 150.
   (a) Décrire l'ordre d'exécution des instructions ayant permis d'obtenir ce résultat.
   (b) Décrire dans ce cas le nombre et les valeurs imprimées lors de "MESSAGE Valeurs TAB"
6. (1pts) Que se passe-t-il si l'instruction close(p[1]) en ligne 65 est supprimée ?
7. (1,5pts) Une exécution de ce programme peut entraîner le blocage du père. Expliquer.
8. (2pts) Proposer une modification du code permettant d'éviter l'interblocage
9. (1,5pts) Montrer que "Message 2" est toujours affiché et toujours avant les deux messages "MESSAGE Valeurs TAB" et ce quelque soit l'ordre d'exécution des processus
10. (1pts) On ajoute la ligne pid4=wait(&ret); à la place de la ligne 76. Quelle est la valeur de retour pid4 (justifier votre réponse) ?

2 Question de cours (2 points)

Dans la section du cours concernant la programmation d'horloges, trois valeurs possibles de l'horloge ont été présentées. Ces horloges sont sensibles de façon différente à la charge de la machine. Expliquer et illustrer cette assertion.

3 Questions de shell (4 points)

(La question 3 de shell sera rendue sur une feuille séparée)

1. Compter le nombre de fichiers java présents dans la sous-arborescence du répertoire courant dont le nom ne contient pas la chaîne "`itf`".
   Indication : l'option `-v` de `grep` inverse la recherche, pour sélectionner les lignes ne correspondant pas au motif.
2. Quel est l'objectif du script suivant ? En particulier, détailler le comportement selon les invocations du script : nombre de paramètres et nature de ces paramètres (nombre, chaîne de caractères non numériques...).
   Indication : `grep --silent` empêche tout affichage et ne fait que retourner un code de retour égal à 0 (= ok) si une correspondance est trouvée, 1 (= erreur) sinon. `tr` avec l'option `-d` détruit les caractères, et `-c` prend le complément de l'ensemble précisé. `pactl set-sink-volume` contrôle le volume du haut-parleur via un pourcentage de la puissance.

```
#!/bin/sh

if [ "$1" = "" ]; then
   vol='100%'
else
   vol="$1";
fi

if ! echo "$vol" | grep --silent '%'; then
   vol="$vol"%
fi

vol=`echo "$vol" | tr -d -c '0-9%'`

pactl set-sink-volume 0 "$vol"
```