

Redirections et tubes

Thèmes traités

- redirections : principe, API Unix ;
- processus communiquant par tubes : architecture, protocole d'usage, API Unix ;
- exercice de synthèse : parallélisme, communication par tubes et signaux.

1 Déroulement

- duplication (dup, dup2), mise en œuvre des redirections
 - présentation : planche 17 du support d'accompagnement des TD (Fichiers)
 - exercice 1
- tubes
 - présentation, protocole d'usage : planche 18
 - exemple de réalisation du schéma producteur-consommateur : exercice 2
 - réalisation d'un pipeline sur des filtres standard : exercice 3 (architecture et algorithmique uniquement)
 - présentation (architecture) de l'exercice de synthèse, qui sera corrigé directement et rapidement en début de TD suivant.

2 Exercices

1. Écrire un programme qui réalise la commande UNIX `cp` (`cp source destination`) en utilisant la commande `cat`. On souhaite de plus que le fichier destinataire de la copie soit uniquement accessible en lecture pour tous les utilisateurs (groupe et autres).
2. Définir deux processus devant respectivement :
 - (a) écrire dans un tube les nombres 1 à N (*processus 1, producteur*) ;
 - (b) lire les nombres écrits dans ce tube, en faire la somme puis afficher sur la sortie standard le résultat (*processus 2, consommateur*). Le consommateur ne connaît pas la valeur de N.Une fois cette version implantée, on souhaite la compléter de la manière suivante :
le processus 2 écrit maintenant son résultat final dans le tube. On ajoute par ailleurs un processus qui lit ce résultat et l'affiche sur la sortie standard. Quel(s) problème(s) pose(nt) cette spécification ?
3. Écrire un programme qui utilise le mécanisme de pipes pour réaliser la commande :
`who | grep nom_utilisateur | wc -l`

3 Testez vous

Vous devriez maintenant être en mesure de répondre clairement aux questions suivantes :

- Est-il possible de transmettre des entiers via un tube ?
- Un processus crée un tube, puis lit des données sur ce tube. Que se passe-t-il ?
- Un processus crée un tube, ferme l'entrée du tube, puis lit des données sur ce tube. Que se passe-t-il ?
- Que se passe-t-il si un processus tente d'écrire dans un tube ayant atteint le maximum de sa capacité ?

4 Synthèse : processus communiquant par signaux et tubes

On considère le programme suivant :

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <signal.h>
5  #include <stdlib.h>
6
7  #define NBTESTS 5
8  static int iBoucle = 0; /* compteur de boucle incrémenté sans fin par le fils pid1
9                          * et remis à zéro par le déclenchement de H1 */
10 static int pariteIBoucle = 0; /* 0 si iBoucle est pair */
11 static int message1NonAffiche = 1; /* indique si H1(_) n'a jamais été déclenché */
12
13 static int p[2];
14
15 void H1 (int sig) {
16     if (message1NonAffiche==1) {
17         printf ("Message_1::P1=%d_a_recu_S1=%d\n", (int) getpid(), sig);
18         message1NonAffiche = 0;
19     } else {
20         if (sig==SIGQUIT) {
21             printf ("Message_2::P2=%d_a_recu_S2=%d\n", (int) getpid(), sig);
22             exit(3);
23         }
24     }
25     pariteIBoucle=iBoucle%2;
26     printf ("iBoucle=%d_pariteIBoucle=%d\n", iBoucle, pariteIBoucle);
27     write(p[1],&pariteIBoucle, sizeof(int));
28     iBoucle=0;
29 }
30
31 void H2 (int sig) {
32     printf ("Message_3::P3=%d_a_recu_S3=%d\n", (int) getpid(), sig);
33 }
34
35 int main (int argc, char *argv[]) {
36     int pid1, pid2, pid3, ret, i, n;
37     int q[2];
38     float nbttotal, nbpair;
39     float bilan, nbpairmoyen;
40     struct sigaction sa1,sa2;
41
42     printf ("Message_4::P4=%d_de_pere_PP4=%d\n", (int) getpid(), getppid());
43
44     pipe(p);
45     pipe(q);
46
47     sa1.sa_handler = H1;
48     sa1.sa_flags = 0;
49     sigemptyset(&sa1.sa_mask);
50     sigaction(SIGQUIT, &sa1, NULL);
51     sigaction(SIGUSR1, &sa1, NULL);
52
53     sa2.sa_handler = H2;
54     sa2.sa_flags = 0;
55     sigemptyset(&sa2.sa_mask);
56     sigaction(SIGUSR2, &sa2, NULL);
57
58     pid1=fork();
59     if (pid1 != 0) {
60         printf("....._pid1=%d\n", (int) pid1);
61         pid2=fork();
62         if (pid2 != 0) {
63             close(p[1]);
64             close(q[1]);

```

/* Ligne 63 */

/* Ligne 64 */

```

65     close(p[0]);
66     printf(".....pid2=%d\n", (int) pid2);
67     for (i=0; i<NBTESTS; i++) {
68         sleep(1);
69         kill(pid1, SIGUSR1);
70     }
71     kill(pid2, SIGUSR2);                /* Ligne 71 */
72     ret=wait(&n);                       /* Ligne 72 */
73     if (WIFSIGNALED(n)) {
74         printf("Message_5: P5=%d et R5=%d\n", ret, WTERMSIG(n));
75     } else {
76         printf("Message_6: P6=%d et R6=%d\n", ret, WEXITSTATUS(n));
77     }
78     kill(pid1, SIGINT);                 /* Ligne 78 */
79     ret=wait(&n);                       /* Ligne 79 */
80     if (WIFSIGNALED(n)) {
81         printf("Message_7: P7=%d et R7=%d\n", ret, WTERMSIG(n));
82     } else {
83         printf("Message_8: P8=%d et R8=%d\n", ret, WEXITSTATUS(n));
84     }
85     bilan=-999;
86     ret=read(q[0], &bilan, sizeof(float));
87     if (ret>0) {
88         bilan = bilan * 2;
89     }
90     printf ( "bilan=%f\n", bilan);
91 } else { /* -----> pid2 == 0 */
92     close(p[1]);                        /* Ligne 92 */
93     close(q[1]);
94     printf ("Message_9: P9=%d de pere PP9=%d\n", (int) getpid(), getppid());
95     pause();
96     execlp("ps", "ps", NULL);
97     ret=wait(&n);                      /* Ligne 97 */
98 }
99 } else { /* -----> pid1 == 0 */
100     pid3=fork();
101     if (pid3 != 0) {
102         close(q[1]);                  /* Ligne 102 */
103         printf(".....pid3=%d\n", (int) pid3);
104         printf ("Message_10: P10=%d de pere PP10=%d\n", (int) getpid(), getppid());
105         for (;;) {                    /* boucle infinie, Ligne 105 */
106             iBoucle ++;
107         }
108         /* Ligne 108 */
109     } else { /* -----> pid3 == 0 */
110         close(p[1]);
111         close(q[0]);
112         nbpair = 0;
113         nbtotal=0;
114         nbpairmoyen = -1;
115         nbtotal=0;
116         while ( read (p[0], &i, sizeof(int)) > 0 ) { /* Ligne 116 */
117             nbtotal = nbtotal + 1;
118             if (i==0) nbpair++;
119         }
120         if (nbtotal!=0) nbpairmoyen = nbpair/nbtotal;
121         printf("nbpair=%f\n", nbpair);
122         printf("nbpairmoyen=%f\n", nbpairmoyen);
123         write (q[1], &nbpairmoyen, sizeof(float));
124     }
125 }
126 return 0;
127 }

```

Questions

Une exécution du code exam avec NBTESTS=5 donne la sortie suivante (notez que certaines valeurs ont volontairement été remplacées par des ????)

```
>exam
Message 4, P4=???? de pere PP4=????
..... pid1 = 2504
..... pid2 = 2505
..... pid3 = 2506
  Message 10, P10=???? de pere PP10=????
  Message 9, P9=???? de pere PP9=????
Message 1, P1=???? a reçu S1=30
iBoucle=241534614 pariteIBoucle=0
iBoucle=242681100 pariteIBoucle=0
iBoucle=485825588 pariteIBoucle=0
iBoucle=242802438 pariteIBoucle=0
Message 3, P3=???? a reçu S3=????
iBoucle=485986617 pariteIBoucle=1
  PID TTY          TIME CMD
 2130 pts/0        00:00:00 bash
 2503 pts/0        00:00:00 exam
 2504 pts/0        00:00:04 exam
 2505 pts/0        00:00:00 ps
 2506 pts/0        00:00:00 exam
Message 6, P6=???? et R6=????
nbpair=4.000000,
nbpaarmoyen=0.800000
Message 7, P7=???? et R7=????
bilan=1.600000
```

1. Décrire l'architecture de communication entre processus et indiquer/représenter les descripteurs ouverts par processus.
2. Quels sont les descripteurs connectés au processus qui effectue la boucle infinie (Ligne 105). Comment ce processus sort-il de cette boucle ?
3. Compléter et expliquer l'affichage des valeurs P_i et (selon les cas) S_i ou R_i ou PP_i , pour $i=1,10$. Expliquer en particulier pourquoi les messages 2, 5 et 8 ne sont pas affichés.
4. Expliquer pourquoi dans la sortie
 - (a) on trouve 3 lignes dont le motif est
 pts/0 exam ?
 - (b) Quel que soit l'ordre de séquençement des processus par le système trouvera-t-on toujours toujours exactement 3 lignes avec
 pts/0 exam ?
5. Expliquer comment le processus qui effectue le while en Ligne 116 sort de cette boucle.
6. Si on supprime le `close(p[1])` en ligne 92, que se passe-t-il ?
7. Si on supprime le `close(q[1])` en ligne 102, que se passe-t-il ?
8. Que se passe t-il si on supprime le `close(p[1])` en Ligne 63 ?
9. Que se passe t-il si on déplace `close(p[1])` en Ligne 63 après le `wait` en Ligne 79 ?
10. Si on supprime l'instruction `kill(pid2, SIGUSR2)` en Ligne 71, que se passe-t-il ?
11. Dans quel cas atteint-on la ligne 97 ?
12. Si on ajoute l'instruction `signal(SIGUSR1, SIG_IGN)` avant la boucle infinie ligne 105 que se passe-t-il ? Indiquer la valeur de bilan.
13. Que se passe-t-il si on supprime la ligne 78 (`kill(pid1, SIG_INT)`) ?
14. Que se passe-t-il si on remplace `kill(pid1, SIGINT)` en ligne 78 par `kill(pid1, SIGQUIT)` ? Indiquer notamment l'effet sur l'affichage du message 7.
15. Pour de grandes valeurs de NBTESTS, vers quelle valeur doit tendre la variable bilan ?