Responsable pédagogique	AF	AM	PB	**PM**
Période	Sem1	Sem2	**Sem 3**	Sem4
Volume horaire	Cours/TD		TP	
			6-8	

PSYST

TP 3 myshell&pipe

Indicateur temporel (hors rédaction du compte-rendu) :

questions	1h	2h	3h	4h	5h	6h-8h
1						
2						
3						
4						

Documents à rendre : Compte-rendu contenant à minima des explications concises des commandes développées ; les sources (avec entête standard) et les résultats obtenus.

Le but est de créer un programme qui se comporte un peu comme le shell, mais qui soit capable de gérer des redirections entrées/sorties de commandes par pipe.

Ex ls | wc qui compte les mots sortis sur le commande ls

Vous partirez du source fourni *myshell3_suj.c*.

Principe de la redirection des flux et des communications par tubes

L'ipc par tube (appel *pipe()* pour créer un tube anonyme) est un moyen puissant pour faire communiquer deux processus " lourds " entre eux.

Dès lors que l'on associe cela, à une redirection ; il devient facile de faire transiter des données en sortie d'une commande du shell, vers l'entrée d'une autre et ainsi de suite.

La redirection consiste à " dérouter " un flux vers un autre ; que ce soit en entrée ou en sortie.

L'idée ici, est de rediriger ce qui sort du premier processus (commande ls par ex.) vers l'entrée en lecture du 2eme processus (par ex . wc).

Plusieurs techniques existent pour faire cela : usage de *freopen()* ; duplication par *dup2()* ;

1. Codage de la création des tubes

Codez la création des 2 tubes : l'un de fils1 vers fils2 et l'autre de fils2 vers fils1.

Utilisez l'appel système *pipe()* ; en passant le bon nom de tableau de descripteurs .

Vous affecterez les descripteurs de fd12 à la communication fils1 vers fils2.

Ainsi, vous devrez après dans le TP, toujours lire sur fd12[0], et écrire sur fd12[1] ; ; en ce qui concerne la communication fils 1 vers 2.

Vous affecterez les descripteurs de fd21 à la communication fils2 vers fils1.

Ainsi, vous devrez après dans le TP, toujours lire sur fd21[0], et écrire sur fd21[1] ; en ce qui concerne la communication fils 2 vers 1. (Cf figure 1).

2. Duplication des processus

Effectuer l'enchainement des 2 fork() (Cf figure 1) ; afin de créer les 2 fils.

Respecter l'algorithme donné en Annexe 2.

Vous ferez en sorte d'appeler la fonction fils1(resp. Fils2) dans la portion de code associée au processus fils1 (resp. fils2).

Faites des essais de communication entre les 2 processus fils.

Envoyer par exemple 'A' de fils1 vers fils2 ; qui en retour doit renvoyer 'B'.

3. Redirection des flux stdin,stdout

Utilisez les appels à *close* suivi de *dup()* ,pour faire en sorte que le flux de sortie standard du 1er processus fils1 aille sur l'entrée du processus fils2.

Modifier aussi l'entrée standard du 2eme, pour que celui lise non plus sur le clavier, mais sur l'entrée du tube fd12[0] !

4. Codage des fonctions 1er fils et 2eme fils

Les prototypes sont void fils1(char *cmd) ; et void fils2(char *cmd) ;

Vous lancerez la commande filtrée list_arg[0] (resp. list_arg[2])(Cf Tpmyshell) en argument de cette fonction du processus fils1.(resp. Fils2).

Vous aurez recours à l'appel système `execl("/bin/sh","sh","-c",cmd,0);` où cmd contiendra bien sûr la commande isolée de votre shell maison : soit list_arg[0] pour la 1ère commande par exemple.

Vous devrez arriver à un fonctionnement similaire de celui figure 2 Annexe 3.

Annexe 1 : communication par tube

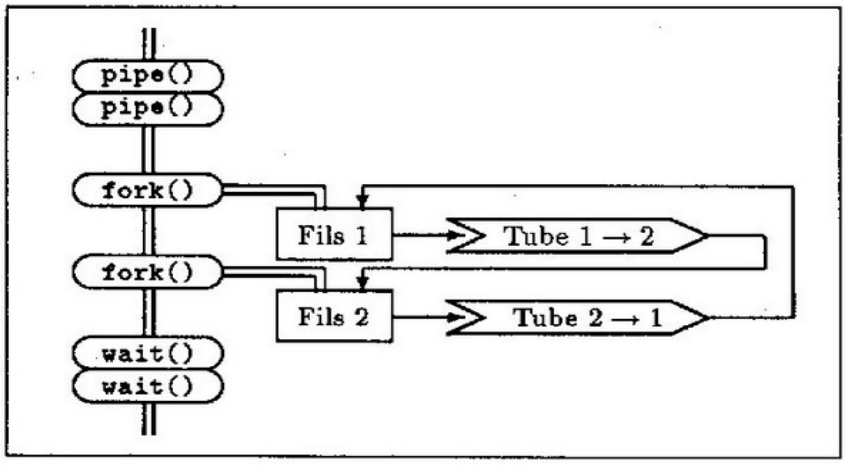

Figure 1 : synoptique de communication par tube entre 2 processus fils

Annexe 2 : algorithme de création de 3 processus

```
SI (pid1 ← fork() > 0 ) ALORS     {  portion code père qui crée le 2eme fils}
        SI ( pid2 ← fork() > 0 )  ALORS { portion code père attend la fin des fils}
                fermeture des tubes inutiles (dupliqués) ;
                        {boucle d'attente de fin des 2 fils }
                        {declarer 2 variables : nb (compteur) et pidexit(code de retour)}
                nb ← 2 ;   { nb de fils }
                TANT QUE  nb FAIRE
                        pidexit ← wait(0) ;
                        SI pidexit = pid1 ALORS
                                nb ← nb – 1 ;
                        SINON SI pidexit = pid2 ALORS
                                nb ← nb – 1 ;
                        FINSI
                FINTQ
        SINON     { code du fils 2}
                appel de fils2(cmd) ;
        FINSI

SINON  { code du fils 1}
        appel de fils1(cmd) ;
FINSI
```

Annexe 3 : Exemple d'exécution

```
patrickMAC:~/Documents/TS2/PSYST/TP/correc$ ./myshell2fork
ls -l | wc
>ls -l | wc<
arg[0] --> ls
arg[1] --> -l
arg[2] --> |
arg[3] --> wc
je suis le pere de pid ---> 12828
j'attends la fin des 2 fils
je suis le fils 1 de pid ---> 12829, lit sur fd5 et ecrit sur fd4
je suis le fils 2 de pid ---> 12830, lit sur fd3 et ecrit sur fd6
     100     893    6903
```

Résultats de la commande wc appliquée sur la sortie du ls -l

Figure 2 : Le shell maison avec pipe