UNIX - TD n^o 3 : Pipe

1 Redirection simple

Ecrire un programme C qui :
- crée un processus fils par fork(),
- et dont la sortie standard est redirigée vers un fichier nommé sortie. (voir open(), close(), dup(), dup2())

2 Communication

Ecrivez un programme C qui :
- met en œuvre un tube : pipe(tube),
- met en œuvre fork() pour créer un fils.

Montrez qu'un tube permet de faire communiquer (transmettre des caractères) et de synchroniser (bloque si une instruction ne peut être réalisée immédiatement) deux processus.

Dessinez l'état de la table des descripteurs de fichiers pour chacun des processus, au cours de l'exécution de ce programme.

3 Redirection sur tube

À partir de ces expériences, écrire un programme C qui crée 2 processus partageant l'accès à un tube non-nommé (pipe()) :
- dont le premier a sa sortie standard redirigée sur l'entrée du tube,
- et dont le second a son entrée standard redirigée sur la sortie du tube.

Ces deux processus se synchronisent par l'envoi et la réception de messages : read() et write().

Dessinez l'état de la table des descripteurs de fichiers pour chacun des processus (toujours dans le temps) en mettant en valeur ce qui permet aux deux processus de partager l'accès aux tubes.

4 Deux redirections

Ecrire un programme C qui :
- crée deux tubes non-nommés,
- crée un processus fils,
- le processus fils peut écrire dans le tube 1 par sa sortie standard et lire des données du tube 2 (issues du père) sur son entrée standard,
- le processus père peut lire sur le tube 1 par son entrée standard, et écrit sur le tube 2 sur sa sortie standard.

5 Accès concurents à un unique tube

Un tube peut-il être utilisé par plus de deux processus ? Pour répondre à cette question, écrivez un programme qui crée deux processus fils, chacun d'eux envoyant des données au processus père à travers un unique tube.

6 Accès concurents à deux tubes

Écrivez un programme qui reçoit des données de deux processus fils (un tube différent par processus). Les informations doivent être affichées immédiatement. Pour cela, l'accès aux tubes ne doit pas être bloquant (voir la fonction fcntl()).

7 Tube nommé

Utilisez un tube nommé pour réaliser une communication entre deux programmes/processus séparés. La création d'un tube nommé peut être faite avec la commande mkfifo ou la fonction du même nom. Montrez qu'il est ainsi possible de faire communiquer deux programmes distincts. Et si vos processus tournent sur des systèmes différents ? Quel est l'apport des sockets ?