Redirections et tubes

Thèmes

- Duplication de descripteurs et redirection
- Utilisation des tubes
- Pipelines

Ressources

Pour ce TP, comme pour les suivants, vous pourrez vous appuyer sur

- Le polycopié intitulé « Systèmes d'exploitation : Unix », qui fournit une référence généralement suffisante sur la sémantique et la syntaxe d'appel des différentes primitives de l'API Unix. Chaque section du sujet de TP indique la (ou les) section(s) du polycopié correspondant au contenu présenté.
- Les pages du manuel en ligne (commande `man`), et plus particulièrement les sections 2 et 3.

Déroulement de la séance :
L'ensemble des questions de ce TP devrait pouvoir être traité dans le temps imparti au TP : chaque question porte sur une fonctionnalité/observation de base de l'API sur les tubes et la duplication de descripteurs. Les questions sont donc à traiter en suivant l'ordre du texte.

Rendu (26/4) : code source répondant à la question 4. (Voir la page du dépôt Moodle)

0 Avant de commencer...

La réalisation du TP demande une connaissance de base de la syntaxe d'appel et de la sémantique des primitives de l'API tubes Unix. Ces notions sont présentées de manière progressive dans le tutoriel proposé en **préparation** du TP (*Attention* : ce tutoriel ne remplace en aucun cas le TP ; il devrait être suivi en dehors de la séance de TP, dans le cas où vous estimeriez utile d'avoir une présentation « en douceur » de l'API).

Le QCM accompagnant ce TP va vous permettre de vous situer par rapport à cette connaissance de base. Comptez une petite dizaine de minutes. Si votre score est inférieur à 75/100, vous auriez sans doute (eu) intérêt à jeter un coup d'œil au tutoriel...

1 Duplication de descripteurs et redirection (polycopié API Unix, section 3.1.4)

Opération essentielle

`dup2(d1,d2)` affecte le descripteur $d1$ au descripteur $d2$.

Attention : c'est bien $d2$ qui reçoit le contenu de $d1$, et non l'inverse.

Rappel

Si vous souhaitez avoir un aperçu de la table des descripteurs de fichier, par exemple pour observer l'effet d'une opération `dup2`, vous pouvez afficher le contenu du sous-répertoire `fd` du répertoire correspondant à chacun des processus dans le répertoire `/proc` (par exemple : `/proc/1234/fd` pour le processus de pid 1234)

Exercices

1. Ecrire un programme C listant dans un fichier fourni en argument les noms de fichiers du répertoire courant, par ordre chronologique de dernier accès.
   Indication : `man ls` devrait vous être utile pour résoudre simplement cette question.
2. Écrire un programme qui réalise la commande UNIX `cp` (`cp source destination`) en utilisant la commande `cat`. On souhaite de plus que le ficher destinataire de la copie soit uniquement accessible en lecture pour tous les utilisateurs (groupe et autres).
   Note : Cet exercice est l'exercice 1 du TD, et ne doit être traité que dans le cas où il n'aurait pas été traité en TD.

2 Tubes (polycopié API Unix, section 3.2)

Opération essentielle

`pipe(p)` crée un tube et affecte le descripteur de sortie du tube à p[0], et le descripteur d'entrée du tube à p[1]

Exercices

3. Ecrire une série de programmes permettant de tester l'effet obtenu dans les scénarios ci-dessous.

   Pour chaque scénario, prédire l'effet attendu **avant** de lancer le programme. Dans le cas où l'effet obtenu n'est pas l'effet attendu, proposer une explication (ou une correction de votre programme :)).

   - (*accès aux tubes*) un processus père crée un fils, puis crée un tube. Le père écrit un entier dans le tube, le fils lit un entier dans le tube et affiche l'entier lu.
   - (*accès aux tubes*) un processus père crée un tube, puis écrit un entier dans le tube. Il crée ensuite un fils qui lit un entier dans le tube et affiche l'entier lu.
   - (*couplage en lecture*) un processus père crée un tube, puis un fils.
     Le père écrit une série d'entiers dans le tube, puis attend (par appel à `pause`), puis se termine.
     Le fils effectue une boucle consistant à lire un entier du tube, afficher l'entier et la valeur retournée par `read(...)`, jusqu'à ce que cette valeur soit ≤ 0. Une fois sorti de la boucle, le fils affiche un message `"sortie de boucle"`, puis se termine.
   - même question, en remplaçant l'appel à `pause` par un appel à `sleep(10)` ;
   - (*couplage en écriture*) un processus père crée un un tableau de N octets, puis un tube, puis un fils.
     Le père effectue une boucle infinie consistant à écrire le contenu du tableau dans le tube, attendre 1 seconde, puis afficher la valeur retournée par `write(...)`.
     Le fils attend un signal (qu'il devra ignorer, et qui pourra être envoyé via le terminal), puis lit une série de 10*N octets, puis se termine. En profiter pour évaluer la taille d'un tampon.
4. Écrire un programme `wgw` prenant en argument (obligatoire) un nom d'utilisateur (*`nom_utilisateur`*) et utilisant le mécanisme de pipes pour réaliser la commande :

   `who | grep` *`nom_utilisateur`* `| wc -l`

   en vous attachant, pour chaque processus, à ne conserver ouverts que les descripteurs (entrée/sortie de tubes) **réellement utilisés**.
5. Modifier le programme précédent (en supprimant des appels à `close(...)`) pour mettre en évidence des anomalies dues à une mauvaise gestion des descripteurs ouverts/fermés, qui pourront se traduire par des des blocages résultant de l'impossibilité de détecter l'absence de producteurs potentiels pour un consommateur (ou l'absence de consommateurs potentiels pour un producteur).

3 Projet

Vous devriez maintenant être en mesure de réaliser les dernières étapes (9 à 11) du projet. *(remise le 25/5)*