

Introduction aux systèmes d'exploitation (IS1)

TP n° 8 : les tubes et les filtres

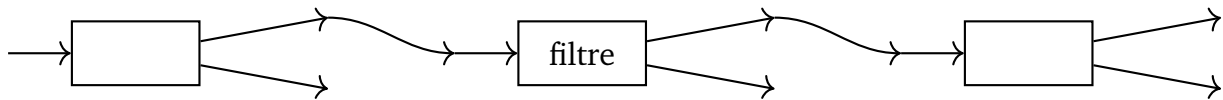
Lors du précédent TP, nous avons étudié les redirections des sorties et de l'entrée standard vers des fichiers ordinaires. En UNIX existe également une redirection qui, étant donné deux commandes « cmd1 » et « cmd2 », réinjecte la sortie standard de « cmd1 » sur l'entrée standard de « cmd2 » sans passer par l'intermédiaire d'un fichier ordinaire. La syntaxe de cette redirection est

« cmd1 | cmd2 »

où « | » est un connecteur marquant l'utilisation d'un fichier spécial pour la redirection appelé *tube* (*anonyme*). Les flots de « cmd1 » et « cmd2 » sont liés par un tube comme indiqué ci-dessous :



Les programmes qui traitent des données provenant de l'entrée standard et produisent un résultat sur la sortie standard sont communément appelés *filtres*. On peut donc utiliser un filtre *avant* et *après* un tube, et les enchaîner comme illustré ci-dessous :



Quelques exemples de filtres :

- « cat » : recopie sur la sortie ce qu'il reçoit sur l'entrée (utile essentiellement pour certaines de ses options)
- « head » : recopie seulement les premières lignes
- « tail » : recopie seulement les dernières lignes
- « less » : affiche l'entrée page par page
- « grep » : recherche un motif dans les lignes d'un texte
- « tee » (« T » en anglais) : dédouble la sortie – sur la sortie standard et sur un (ou plusieurs) fichier(s)
- « tr » : recopie l'entrée en remplaçant certains caractères par d'autres
- « wc » : compte les lignes, mots et caractères
- « yes » : écrit une ligne donnée à l'infini.

Exercice 1 – première utilisation d'un tube

1. L'option « ax » de « ps » affiche la liste de tous les processus en cours d'exécution, mais le résultat défile très vite. ➤ Comment obtenir un défilement page par page ?

2. ➤ Comment compter les processus attachés au terminal courant (à l'aide des commandes « `tail` » et « `wc` ») ?

« `tr` » (**transform**) avec en paramètres deux chaînes de caractères de même longueur, recopie sur la sortie standard le contenu de l'entrée standard tout en remplaçant chaque caractère de la première chaîne par le caractère correspondant de la deuxième. Par exemple « `echo ambassade | tr 'abc' 'ABC'` » écrit sur la sortie standard « `AmBAssAde` ».

Exercice 2 – la commande « `tr` »




La méthode de cryptage dite *de César* consiste à remplacer chaque lettre d'un texte par une autre selon un décalage circulaire des lettres : par exemple, avec un décalage d'une lettre, chaque lettre est remplacée par celle qui la suit dans l'ordre alphabétique, sauf `z` qui est remplacé par `a`.

1. Déterminer les paramètres adéquats pour que le filtre « `tr param1 param2` » crypte son entrée standard selon la méthode de César.
2. ➤ À l'aide de la commande « `echo` » et d'une redirection, créer un fichier `crypte` contenant le filtre précédent. Après avoir rendu le fichier `crypte` exécutable, vérifier son bon fonctionnement.
3. ➤ Créer de même un filtre `decrypte` permettant de décrypter l'entrée standard.
4. ➤ Écrire une ligne de commande permettant de confirmer que `decrypte` effectue bien la modification inverse de `crypte`.
5. ➤ À l'aide de la commande « `tee` », modifier la ligne précédente pour qu'elle stocke le résultat du cryptage dans un fichier `message_secret`, en affichant toujours le message décrypté sur la sortie standard.

Exercice 3 – créer un gros fichier

1. Le fichier `/dev/zero` est un fichier spécial qui renvoie un flot de caractères nuls (ASCII NUL, `0x00`) lors d'une lecture. ➤ Comment l'utiliser pour créer un fichier `null` contenant 10000 caractères (nuls) ?
2. ➤ En utilisant de plus la commande « `tr` », donner un enchaînement de commandes pour créer un fichier `grosfichier` contenant 200 caractères 'a'.
3. ➤ À l'aide de la commande « `yes` », concaténer 1000 fois au fichier `grosfichier` la ligne « Répétez après moi - "Après moi" ».

Exercice 4 – retour sur « head » et « tail »

1.  Afficher (uniquement) les 3 dernières lignes du fichier grosfichier, numérotée à l'aide de « cat » pour pouvoir vérifier que vous ne vous êtes pas trompé.
2.  Afficher la septième ligne du fichier, toujours précédée de son numéro de ligne.
3.  Afficher le 579^e mot de grosfichier. Pour cela, « tr » pourra être utile...

La commande « grep »





« grep » (**g**lobal **r**egular **e**xpression **p**rint) sans option, permet de sélectionner les lignes d'un fichier qui contiennent un motif passé en paramètre. Le fichier lu par défaut est l'entrée standard.



Le motif passé en paramètre à « grep » est une chaîne de caractères pouvant contenir des *jokers* ; si certains des caractères utilisés comme jokers sont les mêmes que ceux utilisés par le shell pour manipuler des ensembles de noms de fichiers, leur signification n'est pas la même, et leur pouvoir expressif est bien plus grand.

Quelques caractères spéciaux de « grep »

- « ^ » et « \$ » représentent respectivement le début et la fin de ligne ;
- « . » représente un caractère quelconque ;
- une chaîne délimitée par les caractères « [» et «] » représente un caractère quelconque dans l'ensemble représenté (par liste exhaustive, intervalle(s) ou complément) ;
- les caractères « ? », « * » et « + » sont des **opérateurs unaires postfixes** permettant de représenter des chaînes formées à partir du caractère qui les précède. Par exemple,
 - « a? » représente *au plus un* caractère a,
 - « [a-z]* » représente une chaîne formée uniquement de minuscules,
 - « .+ » représente une chaîne d'*au moins un* caractère.

Exercice 5 – « grep » et la recherche de motifs

1.  Lister les processus qui exécutent une commande se trouvant dans /usr/sbin.
2.  Lister tous les fichiers de votre *arborescence personnelle* ayant l'extension . java.
3.  Chercher quelle option de « grep » permet de compter les lignes sélectionnées, puis compter les fichiers ordinaires de votre arborescence personnelle.
 Obtenir le même résultat *sans* utiliser l'option précédente.


4.  Chercher quelle option de « grep » permet d'inverser la sélection des lignes, puis éliminer de la sortie de `ls -R` les lignes contenant des `/`.
5.  Compter les fichiers de la sous-arborescence `/lib...`
 - a. dont le nom contient (au moins) deux `s` séparés par exactement deux caractères ;
 - b. dont le nom contient (au moins) deux `s` ;
 - c. dont le nom contient *exactement* deux `s` ;
 - d. dont le nom ne contient pas de chiffre.

Exercice 6 – un peu de hasard

Le fichier `/dev/random` est un fichier (très) spécial correspondant à un générateur aléatoire de bits.

1. À l'aide de ce fichier, créer un fichier `alea` contenant un octet aléatoire.

En général, un octet ne représente pas un caractère *imprimable*. La commande « `cat` » est donc inappropriée pour consulter le contenu d'`alea`. Nous allons plutôt utiliser la commande « `od` », qui, comme « `cat` », recopie son entrée sur la sortie, mais en permettant de spécifier le format d'affichage.

2. Déterminer l'option de « `od` » permettant d'afficher la valeur décimale de l'octet contenu dans `alea`.
3.  À l'aide de filtres déjà utilisés durant la séance, écrire une ligne de commande permettant de tirer un entier aléatoire entre 0 et 256.