Introduction aux systèmes d'exploitation (IS1)

TP 5 – Entrées / sorties

1 Les redirections

Une commande UNIX est une "boîte noire" à laquelle on fournit des arguments sur un fichier logique nommé *entrée standard* et qui renvoie deux types de réponses éventuelles sur des fichiers logiques appelés respectivement *sortie standard* et *sortie erreur standard*, comme illustré à la figure 1.

FIG. 1 – commande UNIX

Quelques exemples :
- La commande `cat` écrit sur la sortie standard ce qu'elle reçoit sur l'entrée standard.
- La commande `date` ne prend rien sur le flot d'entrée et renvoie quelque chose sur le flot de sortie.
- La commande `cd` ne prend rien en entrée et ne renvoie rien en sortie.

Toutes ces commandes peuvent évidemment renvoyer quelques choses sur la sortie erreur standard pour signaler un problème quelconque (mauvaise utilisation de la commande, arguments inexistants, problèmes de droits, etc.).

Par commodité, il est souvent nécessaire de sauver les données fournies par un programme dans un fichier pour pouvoir ensuite les voir en totalité ou les traiter ou inversement de réutiliser des données qui sont déjà dans un fichier. Par défaut, le canal d'entrée est le clavier et les canaux de sortie et sortie erreur sont l'écran. Mais il est très facile de leur substituer un fichier.

Redirection de la sortie : les symboles `>` et `>>`

Exercice 1 – *Redirection de sortie*

- Tapez `ps` dans une fenêtre de terminal. Tapez ensuite `ps > fic`. Un nouveau fichier nommé `fic` a été créé. Regardez son contenu.
- Maintenant faites `date > fic`. Regardez à nouveau le contenu du fichier `fic`.

Concluez sur le rôle de `>`.

Exercice 2 – *Redirection de sortie*

- Faites `who >> fic`. Regardez à nouveau le contenu du fichier `fic`.

Concluez sur le rôle de `>>`.

Redirection de l' entrée : le symbole `<`

Le symbole < sert à rediriger le flot d'entrée.

Exercice 3 – *Commande `cat`*

La commande `cat` prend ce qui lui arrive dans le flux d'entrée et le réécrit en sortie.

Lancez `cat` sans argument. L'invite change de forme pour vous permettre de fournir des données.

Tapez quelques caractères puis frappez la touche entrée. Observez le résultat.

Tapez encore quelques lignes, puis pour indiquer au processus la fin des données à traiter, pressez la combinaison de touches `Ctrl-D` (aussi appelé EOF, ou *end of file*).

Que se passe-t-il si à la fin d'une ligne contenant des caractères, vous pressez `Ctrl-C` au lieu de `Ctrl-D` ? Expliquez.

Lancez maintenant la commande `cat < fic`. Comment expliquez-vous ce qui se passe ?

Exercice 4 – *Commande `wc`*

La commande `wc` sert à compter un nombre de caractères, de mots et/ou de lignes. On l'utilise soit en lui fournissant comme paramètre le nom d'un fichier, soit avec le flux de l'entrée standard. Appliquez-la au fichier `fic` précédemment créé en utilisant ces 2 méthodes.

Redirection de la sortie erreur : les symboles `2>` et `2>>`

Attention : ce symbole est valable en `bash` mais ne s'étend pas aux autres shells[1].

Exercice 5 – *Rediriger les messages d'erreur*

Tapez dans votre terminal la commande `dater` (qui n'existe pas) : vous obtenez un message d'erreur. Rediriger la sortie standard de cette commande vers un fichier. Regardez le contenu de ce fichier. Que constatez-vous ?

En fait, les messages d'erreur ne sortent pas sur le même canal que le résultat des commandes. Recommencez l'opération en redirigeant avec le symbole `2>`.

Faites des tests pour comprendre la signification du symbole `2>>`.

Exercice 6 – *Fichier `/dev/null`*

[exécution lente] Le fichier `/dev/null` est un "puits sans fond" : on peut écrire dedans tant qu'on le veut et les données sont alors perdues. Il peut servir à jeter par exemple la sortie erreur quand on en n'a pas besoin.

Lancez la commande `ls -R /home`. Vous pouvez constater que vous n'avez pas accès à un certain nombre de répertoires et fichiers. Relancez cette commande en dirigeant la sortie erreur sur `/dev/null`.

Exercice 7 – *Enchaînement de commandes et redirections*

Comme on a déjà eu l'occasion de le voir, les commandes peuvent être enchaînées grâce aux opérateurs `;`, `&&` et `||`. En combinant ces derniers et des redirections, écrivez une ligne de commande pour exécuter chacune des opérations suivantes :
- Ecrire le contenu de votre répertoire de login dans le fichier `replog` et votre identifiant (`id`) dans le fichier `identifiant`.
- Si le fichier `~/toto` existe, afficher son contenu à l'ecran.
- Ecrire dans un fichier la liste des fichiers (au sens large) de votre répertoire de login et la liste des processus lancés sur votre machine (une seule redirection).

[1] Par exemple en `ksh` ou en `tcsh`, le symbole équivalent est `>&` qui a une toute autre signification en `bash`.

Combiner les redirections : `>&`

On peut bien entendu faire plusieurs redirections en même temps :

```
commande > fichier_out 2> fichier_err
commande > fichier_out < fichier_in
commande < fichier_in > fichier_out
```

etc.

On peut aussi vouloir rediriger un flux sur un autre. Les exemples les plus courants étant de vouloir écrire la sortie standard et la sortie erreur dans le même fichier ou encore de vouloir écrire sur la sortie erreur. On utilise alors le symbole `>&` suivi du descripteur concerné[2].

```
commande > fichier 2>&1
echo "message d'erreur" >&2
```

Exercice 8 – *Toutes les sorties dans le même fichier*

On veut lancer la commande `ls fic fictoto` (en supposant que `fictoto` n'existe pas) et faire en sorte que la sortie et la sortie erreur soient écrites dans un nouveau fichier `sorties`. Comment faire ? Donnez plusieurs solutions.

2 Les tubes et les filtres

On peut vouloir utiliser le résultat d'une commande (sortie standard) pour le réinjecter dans une autre commande (entrée standard) comme indiqué dans la figure 2, sans passer par un fichier régulier intermédiaire.

FIG. 2 – illustration du tube : <*commande 1*> | <*commande 2*>

Pour faire cela, on utilise le symbole `|` nommé *tube* (*pipe* en anglais) de la façon suivante : <*commande 1*> | <*commande 2*>.

Exercice 9 – *Utilisation de tube*

On veut compter le nombre de personnes connectées, grâce aux commandes `who` et `wc` (sachant qu'on peut éventuellement compter plusieurs fois un utilisateur si celui-ci utilise plusieurs terminaux). Comment faire ? Et si on veut écrire le résultat dans un fichier ?

Exercice 10 – *Défilement page par page*

On veut voir tous les processus en cours d'exécution. Si vous lancez la commande `ps -ax`, le résultat défile très vite et on rate le début. Comment faire ?

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* un tube et *après* un tube, comme illustré à la figure 3.

Voici quelques exemples typiques de filtres :

– `cat` : recopie sur la sortie ce qu'il reçoit sur l'entrée
– `sort` : trie les données par ordre alphabétique

[2]L'entrée standard correspondant au descripteur 0, la sortie standard au descripteur 1 et la sortie erreur standard au descripteur 2.

FIG. 3 – un filtre

– `less` : affiche sur la sortie ce qu'il reçoit sur l'entrée, page par page
– `grep` : recherche un motif dans les lignes d'un texte
– `tee` : copie de l'entrée standard sur la sortie standard et sur un fichier

Les filtres sont particulièrement utiles lorsqu'on les combine à l'aide de tubes. Par exemple, pour trier par ordre alphabétique les lignes d'un fichier `liste.txt` puis les afficher page par page, on utilisera :

```
cat liste.txt | sort | less
```

Exercice 11 – *Tri*

La commande `sort` trie ce qu'elle reçoit sur l'entrée standard. Triez le contenu de votre répertoire personnel par ordre alphabétique.

Exercice 12

A l'aide des commandes `cut` et `uniq`, affichez tous les types de droits présents dans votre répertoire de login.

3 Tubes nommés.

Les tubes créés par l'opérateur `|` sont ce que l'on appelle des tubes anonymes. Ils sont créés directement par le shell pour une commande donnée. Il est possible de créer manuellement des tubes en leur donnant un nom. Cela permet ensuite de les utiliser dans plusieurs commandes. Ces tubes sont ce que l'on appelle des tubes nommés.

La commande `mkfifo` permet de créer un tube nommé. La syntaxe est la suivante :

```
mkfifo nom
```

où `nom` est le nom du tube que vous voulez créer.

Exercice 13 – *Création de tubes.*

1. Créez un fichier `fic`. Est-il possible, en utilisant 2 terminaux de lire instantanément dans le premier ce qu'on écrit dans `fic` à partir du deuxième ? Pourquoi la commande `cat` se termine-t-elle après avoir imprimé le contenu du fichier ?
2. Créez un fichier nommé `tube1` avec la commande `mkfifo`. Regardez ses caractéristiques (droits, taille, type de fichier, etc.), que constatez-vous ?
3. Essayez d'ouvrir ce fichier dans emacs puis kwrite, que constatez-vous ? Dans la suite, toutes les lectures et écritures dans les différents fichiers se feront à l'aide de la commande `cat`.
4. Écrivez dans ce fichier sans terminer la commande.
5. Ouvrez un deuxième terminal et lisez le fichier `tube1`. Que constatez-vous ? Essayez d'écrire autre chose dans la commande de la question précédente. Que se passe-t-il ?
6. Que constatez-vous lorsque vous terminez la commande qui écrit dans le tube ?
   Que pouvez-vous en déduire pour la question 1 ?

7. On appelle généralement `écrivain` le programme chargé d'écrire dans le tube et `lecteur` celui qui lit son contenu. Peut-on avoir plusieurs écrivains pour un lecteur ? Essayez avec au moins deux écrivains.
8. Que se passe-t-il si on a plusieurs lecteurs ? Essayez avec un seul écrivain et au moins deux lecteurs. Qu'en déduisez-vous sur le fonctionnement des tubes nommés ?

Exercice 14 – *Messagerie instantanée.*

Le but de cet exercice est de reproduire à l'aide de tubes nommés un équivalent à la commande `talk`.

1. Que fait la commande `talk` ?
2. Sans la commande talk, essayez de communiquer avec un autre terminal à l'aide de tubes. Combien vous faut-il de tubes pour avoir une conversation correcte dans les deux sens ?
3. Demandez à votre voisin de se logger sur votre machine en utilisant `ctrl+alt+F1` pour ouvrir un terminal en mode texte. En donnant les droits nécessaires pour accéder au tube, essayez de communiquer avec lui. Pour revenir au terminal contenant KDE, il faut utiliser `crtl+alt+F9`.
4. Comme beaucoup de messageries instantanées modernes, on voudrait conserver un fichier journal de la conversation. Utilisez la commande `tee` pour le faire.

Exercice 15 – *Lancer des commandes dans un autre terminal.*

Le but de cet exercice est d'exécuter des commandes dans un autre terminal. Pour que l'exercice soit plus amusant, vous pouvez vous logger dans un terminal en mode texte.

1. À partir d'un terminal, essayez d'exécuter des commandes dans un autre terminal en utilisant ce que vous avez vu dans l'exercice précédent.
2. Essayez de voir le résultat des commandes dans le premier terminal. Faites en sorte de pouvoir voir aussi les messages d'erreur.