

## Introduction aux systèmes d'exploitation (IS1)

### TP n° 7 : les entrées / sorties

Une commande UNIX est une “boîte noire” à laquelle on peut fournir, en plus de ses arguments, 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 dans la Figure 1. On pourra aussi parler de *flots* (ou *flux*) d'entrée et de sortie.



FIGURE 1 – Flot d'entrée et de sortie d'une commande UNIX.

Quelques exemples :


- « cat » écrit sur la sortie standard ce qu'elle reçoit sur l'entrée standard.
- « date » ne prend rien sur le flot d'entrée et écrit sur le flot de sortie.
- « cd » ne prend rien en entrée et ne renvoie rien en sortie.

Toutes ces commandes peuvent renvoyer des informations 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 en entrée ou en sortie d'un processus dans des fichiers afin de pouvoir ensuite les réutiliser, les voir en totalité ou les traiter. Par défaut, les flots standard correspondent au terminal : l'entrée standard est donc saisie au clavier et les sorties (standard et erreur) s'affichent à l'écran. Mais le mécanisme de *redirection* permet de leur substituer un fichier.

À chaque flot d'entrée/sortie est associé un entier, appelé *descripteur* :

- 0 correspond à l'entrée standard,
- 1 correspond à la sortie standard et
- 2 correspond à la sortie erreur standard.

**Avant de passer aux exercices,** télécharger depuis Moodle le script `tp7.sh`. Son exécution crée une arborescence de racine `~/Cours/2019/IS1/TP7`. Placez-vous dans ce répertoire, et répondez aux questions marquées par  dans le fichier `reponses_TP7.txt`.

## Redirection de la sortie standard : les symboles > et >>

### Exercice 1 – les symboles > et >>

1. Le mécanisme de redirection utilisé avec la commande « echo » lors des précédents TP pour créer des fichiers s'applique à toutes les commandes. Exécuter par exemple `date > fic` puis afficher le contenu du fichier `fic`. Exécuter ensuite `cal > fic` et afficher de nouveau le contenu du fichier `fic`. ➤ Qu'en concluez-vous sur le rôle de « > » ?
2. Refaire les manipulations précédentes en utilisant à chaque fois >> au lieu de >. ➤ Qu'en concluez-vous sur le rôle de >> ?

Sous UNIX, les périphériques sont considérés comme des fichiers spéciaux, accessibles par des liens situés dans la sous-arborescence `/dev` du système de fichiers. Il existe deux types de tels fichiers spéciaux : “caractère” (terminaux etc.) ou “bloc” (disques etc.). Ces termes désignent la manière de traiter les lectures et écritures : soit caractère par caractère, soit par blocs.

### Exercice 2 – le terminal, un fichier très spécial...

1. La commande « `tty` » retourne la référence absolue du fichier spécial correspondant au terminal dans lequel elle est exécutée. Afficher les caractéristiques (droits, etc.) de ce fichier. ➤ Quel est son type : bloc ou caractère ?
2. ➤ Dans un autre terminal, exécuter la commande `date` en redirigeant sa sortie standard vers le fichier spécial correspondant au premier terminal. Que se passe-t-il ?
3. Se connecter par `ssh` à la machine de son (sa) voisin(e) et lui demander la référence du fichier spécial correspondant à un de ses terminaux. À l'aide de la commande « echo » essayer d'écrire le message « Salut mon cher ami » (ou « Salut ma chère amie », à vous de choisir) dans ce terminal. ➤ Que se passe-t-il ? Demander à son (sa) voisin(e) de modifier les droits en écriture sur ce fichier, puis retester.

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

En bash, la sortie erreur est redirigée grâce aux symboles `2>` et `2>>`.

### Exercice 3 – rediriger les messages d'erreur

1. Dans `~/Cours/2019/IS1/TP7/Shadocks`, exécuter la commande « `cat gabuzomeu` » (nom qui n'est pas censé exister). Vous devez obtenir un message d'erreur. Refaire la même opération en redirigeant la sortie standard dans un (nouveau) fichier `bugabu`. Que constatez-vous ?

2. Exécuter ensuite `man gabuzomeu 2> bugabu` et comparer. Recommencer en remplaçant `2>` par `2>>`. Que constatez-vous ?
3. Exécuter la commande « `meuzobuga` » (qui n'existe probablement pas plus que le fichier précédent) en redirigeant la sortie d'erreur vers un fichier `meuzobu`, puis afficher le contenu de ce fichier : la commande « `meuzobuga` » n'existe pas, pourtant le message d'erreur a été redirigé. 🚩 Expliquer pourquoi.

`/dev/null` est un fichier spécial qui se comporte comme un “puits sans fond” : on peut écrire dedans tant qu'on le veut et les données sont alors perdues. Il peut par exemple servir à jeter la sortie erreur quand on n'en a pas besoin.

#### Exercice 4 – le fichier `/dev/null`

1. Afficher le contenu de tous les fichiers appartenant à un sous-répertoire du répertoire `LesCowboysFringants`, et dont le nom contient au moins une majuscule. 🚩 Faire en sorte d'éliminer les messages d'erreur pour obtenir un affichage lisible.
2. 🚩 Donner une ligne de commande qui utilise les messages d'erreur de « `ls` » pour déterminer la liste des répertoires non accessibles dans l'arborescence de racine `LesCowboysFringants` (sans donc afficher les fichiers et les répertoires accessibles).

### Redirection de l'entrée

Le symbole `<` sert à rediriger le flot d'entrée. En exécutant `cmd < fic` on passe à la commande « `cmd` » le contenu du fichier « `fic` » (et pas le fichier brut).

#### Exercice 5 – le symbole « `<` »

Dans le répertoire `~/Cours/2019/IS1/TP7/SnowWhite` exécuter les commandes « `cat BlancheNeige` » et « `cat < BlancheNeige` ». Faire de même avec la commande « `wc` ». 🚩 Expliquer la différence de comportement entre « `cat` » et « `wc` ».

### Quelques commandes pouvant manipuler l'entrée standard

La plupart des commandes qui manipulent l'entrée standard admettent en général un argument facultatif qui est un nom de fichier sur le contenu duquel elles peuvent s'exécuter. Il peut néanmoins être intéressant de les faire travailler directement sur l'entrée standard.

« `cat` » sans argument, recopie sur la sortie ce qui arrive dans le flot d'entrée.

**Exercice 6 – la commande « cat »**

Retourner dans le répertoire ~/Cours/2019/IS1/TP7/SnowWhite.

1. Lancer « cat » sans argument. La ligne reste vide, le shell attend que vous lui fournissiez des données. Taper « Arrête de répéter tout ce que je dis! » suivis de la touche entrée et observer le résultat.  
Taper encore quelques lignes, puis presser la combinaison de touches *ctrl-D* au début d'une ligne vide. Cela indique au processus la fin des données à traiter.
2. Relancer « cat », taper la même phrase qu'en haut (ou des caractères de votre choix) puis, sans aller à la ligne, appuyer sur *ctrl-D*. Que se passe-t-il ?  
Taper une nouvelle ligne et, au lieu d'appuyer sur la touche *Entrée* ou *ctrl-D*, taper maintenant *ctrl-C*. ➤ Expliquer la différence de comportement.
3. Essayer l'option « -n » de cat, d'abord sans argument (pour que la commande lise son entrée au clavier) et après en lui passant comme argument « Sorciere ». ➤ Que fait l'option « -n » ?
4. Chercher dans la page man ce que fait l'option « -s ». Pour la tester, il pourra être utile de la combiner avec l'option « -n ». ➤ Créer ensuite un fichier *Reine*, lisible sans devoir utiliser la barre de défilement, à partir du fichier *Sorciere*.

« head » et « tail » lisent sur leur entrée standard et conservent les premières lignes (pour « head ») et dernières (pour « tail »). Le nombre de lignes conservées est 10 par défaut, ou l'entier passé en argument après l'option « -n ».

**Exercice 7 – les commandes « head » et « tail »**

1. ➤ Tester les commandes « head » et « tail » en leur passant comme paramètre le fichier « *Reine* ». Afficher ensuite seulement les 6 premières lignes du même fichier.
2. Lancer la commande « tail -n 3 ». Comme dans l'exercice précédent, le shell attend que vous fournissiez des données, que la commande traitera ligne par ligne. Taper cinq lignes de texte suivies par *ctrl-D*. ➤ Qu'affiche votre commande ? Pourquoi ?
3. Comparer avec la commande head -n 3. ➤ Que se passe-t-il ? Expliquer la différence.
4. ➤ À quoi sert l'option « -c » de ces commandes ? La tester.
5. Pour la commande « tail », le nombre passé avec les options « -n » ou « -c » peut être précédé d'un signe « + », pour indiquer qu'il est compté à partir du début et non à partir de la fin. Tester tail -n +2 et taper cinq lignes sur le flot d'entrée, pour vérifier que vous avez bien compris. ➤ Faire de même en lui passant comme paramètre le fichier « *Reine* ». Qu'obtient-on ?

6. Lancer la commande « head » en lui donnant les deux paramètres « BlancheNeige » et « Prince ». Que peut-on remarquer? ➤ Déterminer quelle option de « tail » permet de supprimer l’affichage des noms.
7. ➤ À partir du répertoire « /Cours/2019/IS1/TP7/ » créer un fichier LeGorille à partir des 10 premières lignes de chaque fichier contenu dans le répertoire « Brassens ». De façon similaire, créer un fichier « LesPassants » en utilisant les 6 dernières lignes de chaque fichier.