


Introduction aux systèmes d'exploitation (IS1)

TP n° 6 : les entrées / sorties

Télécharger depuis Moodle le script `tp6.sh`, puis l'exécuter depuis `\home/Cours/2023/IS1`. Placez-vous dans le répertoire TP6 qui vient d'être créé et répondez aux questions marquées par  dans le fichier `reponses_TP6.txt`.





La variable PATH

Trouver le bon chemin La variable d'environnement PATH joue un rôle primordial : c'est elle qui contient la liste de tous les chemins vers les répertoires dans lesquels les fichiers exécutables doivent être recherchés. C'est grâce à elle qu'il est possible d'utiliser des noms de commandes simples tels que « `bash` », « `grep` » ou « `ls` » en lieu et place du chemin complet vers les exécutables concernés.

Le caractère `:` sert de séparateur entre les différents chemins contenus dans la valeur de PATH.

Exercice 1 – la variable d'environnement PATH

1. Afficher la valeur de la variable d'environnement PATH.
2. Créer dans votre répertoire `~/Cours/2023/IS1/TP6` un fichier nommé `fic.sh`, contenant la ligne suivante :

```
echo "ceci est le résultat de l'exécution du fichier fic.sh."
```
3.  Après lui avoir donné les droits nécessaires, essayez de l'exécuter depuis votre répertoire `~/Cours/2023/IS1` (en utilisant une référence valide, relative ou absolue). Recommencer depuis votre répertoire `~/Cours/2023/IS1/TP6`, d'abord par la référence (valide) « `./fic.sh` », puis par « `fic.sh` ». Que se passe-t-il dans chaque cas ? Pourquoi ?
4.  Modifier PATH pour que la commande « `fic.sh` » s'exécute sans erreur depuis le répertoire `~/Cours/2023/IS1/TP6` (et seulement depuis celui-ci).
5.  Créer un répertoire `~/mybin`, puis déplacer `fic.sh` dans ce répertoire. Essayer ensuite de l'exécuter par la simple ligne de commande « `fic.sh` », depuis le répertoire `~/mybin`, puis depuis un autre répertoire.  Comment faire en sorte que l'exécution s'effectue sans erreur dans tous les cas ?

Si vous le souhaitez, vous pouvez rendre pérenne(s) l'un et/ou l'autre de ces comportements en modifiant votre fichier `~/bashrc`.

Redirections

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 ci-dessous. On pourra aussi parler de *flots* (ou *flux*) d'entrée et de sortie.



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 inexistant, problèmes de droits, etc.).

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 trois flots standard correspondent au terminal depuis lequel la commande est lancée : l'entrée standard est saisie au clavier et les sorties (standard et erreur) s'affichent à l'écran. Cependant, on peut changer la destination de ces flots à l'aide du mécanisme de *redirection*.

À chaque flot d'entrée/sortie est associé un entier, appelé *descripteur* : 0 correspond à l'entrée standard, 1 à la sortie standard et 2 à la sortie erreur standard.

Redirection de la sortie standard : les opérateurs >, >| et >>

Lors des précédents TP, on a utilisé les opérateurs de redirection > et >> pour créer des fichiers avec la commande « echo » ; ce mécanisme de redirection peut en fait s'utiliser avec toutes les commandes.

Exercice 2 – rediriger la sortie standard

1. Exécuter la commande « date > fic », puis afficher le contenu de fic. Exécuter ensuite « cat > fic » et comparer le contenu de fic. ➤ Quel est le rôle de > ?
2. Refaire les manipulations précédentes en utilisant maintenant l'opérateur >> plutôt que >. ➤ Que fait l'opérateur >> ?

Exécuter maintenant la commande « `set -o noclobber` » : cela active l'option `noclobber` du shell, et en modifie le comportement.

3. Exécuter à nouveau la commande « `date > fic` ». 🚩 Que constatez-vous ?

Exécuter maintenant la commande « `date >| fic` ».

🚩 Quelle est la différence entre les opérateurs `>` et `>|` ?

L'option `noclobber` du shell peut à nouveau être désactivée avec la commande « `set +o noclobber` ». Si, au contraire, vous souhaitez rendre pérenne cette option, il faut ajouter la ligne « `set -o noclobber` » à votre fichier `~/ .bashrc`.

Redirection de la sortie erreur standard : les opérateurs `2>` et `2>>`

En shell POSIX (`sh`, `bash`, `zsh`...), les opérateurs `2>`, `2>|` et `2>>` redirigent la **sortie erreur standard**.

Exercice 3 – rediriger les messages d'erreur

Placez vous dans le répertoire `~/Cours/2023/IS1/TP6/Shadoks`.

1. Exécuter la commande « `cat Personnages/DevinPlombier` ». Ce fichier étant sans droit en lecture, vous devez obtenir un message d'erreur. Refaire la même opération en redirigeant la sortie standard dans un (nouveau) fichier `gabuzo`.

🚩 Que constatez-vous ?

2. Exécuter ensuite « `cat Personnages/DevinPlombier 2> zobuga` » et comparer. Recommencer en remplaçant `2>` par `2>>`. 🚩 Que constatez-vous ?

3. Exécuter la commande « `Personnages/ProfesseurShadoko` » (fichier sans droit en exécution) en redirigeant la sortie d'erreur vers un fichier `meuzobu`, puis afficher le contenu de ce fichier. La commande « `Personnages/ProfesseurShadoko` » échoue, pourtant le message d'erreur a été redirigé.

🚩 Quel est le programme dont le message d'erreur a été redirigé ?

4. Exécuter la commande « `nimportequoi` » (qui a priori n'existe pas) en redirigeant la sortie d'erreur vers un fichier `zogabu`, puis afficher le contenu de ce fichier. La commande « `nimportequoi` » n'existe pas, pourtant le message d'erreur a été redirigé.

🚩 À nouveau, quel est le programme dont le message d'erreur a été redirigé ?

Redirection de l'entrée standard

L'opérateur `<` 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).

« `tee` » sans argument, permet de **dupliquer** le flot de sortie, qui continue d'être affiché dans le terminal, mais est également recopié dans un ou plusieurs fichiers dont les références sont passées en paramètres.

Exercice 4 – copies multiples simultanées

Depuis le répertoire `BlancheNeige`, exécuter la commande « `tee Charmant < Prince` ». Comparer les deux fichiers « `Prince` » et « `Charmant` ». Que remarque-t-on ?

À l'aide d'un *seul* appel à la commande « `tee` », créer simultanément plusieurs copies du fichier `Nain`, respectivement appelées `Atchoum`, `Dormeur`, `Grincheux`, `Joyeux`, `Prof`, `Simplet` et `Timide`.

Combiner les redirections : `&>`

On peut bien entendu faire plusieurs redirections pour la même commande, par exemple : « *commande > sortie 2> erreur* ». Mais on également rediriger simultanément les deux flots de sortie (standard et erreur) sur le même fichier à l'aide de l'opérateur `&>`.

Exercice 5 – toutes les sorties dans le même fichier

À partir du répertoire `~/Cours/2023/IS1/TP6/Shadoks`, afficher le contenu de tous les fichiers du sous-répertoire `Personnages` en faisant les redirections suivantes :

1. rediriger les deux sorties séparément vers un (nouveau) fichier `gagabubu` *en mode écrasement forcé* ;
2. rediriger les deux sorties séparément vers un (nouveau) fichier `bubuzozo` *en mode écrasement prudent* (donc en activant l'option `noclobber`) ;
3. rediriger les deux sorties séparément vers un (nouveau) fichier `zozomeumeu` *en mode ajout* ;
4. rediriger les deux sorties *simultanément* vers un (nouveau) fichier `meumeugaga` *en mode écrasement prudent*.

Fichiers spéciaux

`/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 6 – Le fichier `/dev/null`

1. Afficher le contenu de tous les fichiers appartenant à un sous-répertoire du répertoire `~/Cours/2023/IS1/TP6/LesCowboysFringants`, et dont le nom contient au moins une majuscule.
Faire en sorte que les messages d'erreur ne s'affichent pas pour obtenir un affichage lisible.

2. ➤ Modifiez cette ligne de commande afin de créer un fichier `marine_marchande` contenant le texte lisible. On ne veut toujours pas que les messages d'erreurs s'affichent sur le terminal.
3. ➤ 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).

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 » 'c' (terminaux etc.) ou « bloc » 'b' (disques etc.). Ces termes désignent la manière de traiter les lectures et les écritures : soit caractère par caractère, soit par blocs.

Exercice 7 – Le terminal, un fichier très spécial...

1. La commande « `tty` » renvoie 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. ➤ Dans un autre terminal, exécuter une ligne de commande listant l'arborescence de racine `LesCowboysFringants` et affichant dans le premier terminal le résultat (erreurs comprises).

Quelques commandes manipulant l'entrée standard

Les 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 8 – la commande « `cat` »

1. Lancer « `cat` » sans argument, mais avec l'option « `-n` » ; la ligne reste vide : « `cat` » attend que vous lui fournissiez des données. Taper « Arrête de répéter tout ce que je dis ! » suivi 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 la fin des données à traiter.
2. Chercher dans la page man ce que fait l'option « `-s` » de « `cat` », et tester son effet ; pour mieux le visualiser, il pourra être utile de la combiner avec l'option « `-n` », ou de rediriger la sortie standard sur un autre terminal.
➤ Créer ensuite un fichier `Reine`, lisible sans devoir utiliser la barre de défilement, à partir du fichier `~/Cours/2023/IS1/TP6/BlancheNeige/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 9 – les commandes « head » et « tail »

1. Dans ~/Cours/2023/IS1/TP6/Brassens, tester les commandes « head » et « tail » en redirigeant leur entrée standard sur le fichier « BancsPublics ».
 ➤ Afficher ensuite seulement ses 6 premières lignes.
2. Lancer la commande « tail -n 3 » (ou « tail -3 ») en redirigeant la sortie sur un autre terminal. 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, terminées par ctrl-D. ➤ Qu'affiche votre commande ? Quand ? Pourquoi ?
3. Faire de même avec la commande « head -3 ».
 ➤ Expliquer la différence de comportement.
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 « BancsPublics ». Qu'obtient-on ?
6. Exécuter la commande « head » en lui donnant les deux paramètres « BancsPublics » et « Fernande ». Que peut-on remarquer ? ➤ Déterminer quelle option de « head » (qui fonctionne également avec « tail ») permet de supprimer l'affichage des noms.
7. ➤ Depuis le répertoire ~/Cours/2023/IS1/TP6, créer un fichier LeGorille à partir des 10 premières lignes de chaque fichier contenu dans le répertoire « Brassens ». De façon similaire, créer un fichier « LesPassantes » en utilisant les 6 dernières lignes de chaque fichier.