Jusqu'à maintenant, nous avons principalement vu plusieurs commandes d'affichages. De nombreuses commandes produisent un affichage, mais nous garderons en tête principalement :
— `echo` qui répète le texte donné en argument,
— `cat` qui affiche le contenu d'un fichier régulier et
— `ls` qui affiche le contenu d'un dossier.

En pratique, lorsque l'on veut traiter des informations, il est fréquent de ne pas vouloir utiliser la totalité du retour de ces commandes d'affichage, mais seulement une partie. Pour cela il existe des commandes d'un type particulier : Les filtres. Ces commandes ont pour point commun de lire sur leur entrée standard, de traiter l'information lue puis de retourner sur leur sortie standard une version modifiée de ce qui a été lu.

Il existe de nombreux filtres avec des effets variés. En voici une liste des plus intéressants. L'explication complète des commandes se trouve bien sûr dans le `man`.

`cat`	retourne les lignes lues sans modification.
`cut`	ne retourne que certaines parties de chaque ligne lue.
`grep`	retourne uniquement les lignes lues qui correspondent à un modèle particulier ou qui contiennent un mot précis.
`head`	retourne les premières lignes lues.
`sort`	trie les lignes lues.
`tail`	retourne les dernières lignes lues.
`tee`	envoie les données lues sur la sortie standard **ET** dans un fichier passé en paramètre.
`tr`	remplace des caractères lus par d'autres.
`uniq`	supprime les lignes identiques.
`wc`	retourne le nombre de caractères, mots et lignes lus.
`sed`	édite le texte lu.

Pour utiliser les filtres, l'approche la plus simple est de récupérer la sortie standard d'une commande d'affichage et de l'envoyer sur l'entrée standard du filtre. Le tube (`|`) permettant précisément de remplir cette fonction, il est l'élément parfait.

Bien que d'autres usages soient possibles, pour des raisons pédagogiques, pendant les TPs de R1.04, on utilisera toujours les filtres de la manière suivante :

`<commande d'affichage> | <filtre>`

Si plusieurs filtres sont nécessaires à la suite, la ligne aura la forme suivante :

```
<commande d'affichage> | <filtre1> | <filtre2> | <filtre3> ...
```

Où `<commande d'affichage>` est remplacé par une commande `echo` ou une commande `cat` suivie de leurs arguments et `<filtre>` est remplacé par l'un des filtres listés ci-dessus. Par exemple on pourra utiliser le filtre `tr` de la manière suivante : `echo toto | tr o a`. Comme `tr "o" "a"` transforme tous les `o` en `a`, le retour sur la sortie standard sera `tata`.

1 Prise en main des filtres

Pour réaliser cet exercice, vous devrez copier le fichier `/home/public/baste/R1.04/TP04/tuyaux` dans votre dossier `~/R1.04/TP04` que vous aurez créé pour l'occasion.

Pour cet exercice, vous allez être amené a consulter le manuel des commandes à plusieures reprise. On rappel à toute fin utilse que :
— le SYNOPSIS donne la syntaxe d'une commande (et seule cette syntaxe est correcte).
— Lorsqu'un mot est entre crochet (`[]`), c'est qu'il est facultatif.
— Lorsqu'un mot est souligné, c'est qu'il doit être remplacé par autre chose et qu'il est expliqué dans le manuel par quoi il peut être remplacé.
  — Il est indispensable d'aller trouver comment il doit être remplacé !
— Pour effectuer une **recherche dans le manuel**, il faut tapper `/` suivi du mot recherché. On peut ensuite aller à l'occurence suivante avec `n` et la précédente avec `N`.

On donnera quelques aides de lectures :
— Un complément d'un ensemble de caractères consiste à prendre tous les caractères sauf ceux de l'ensemble.
— Un champ est une zone de texte située entre deux délimiteurs. On considérera que le début de la ligne et la fin de la ligne sont des délimiteurs.

Attention : Pour les deux exercices qui suivent, vos lignes de commandes auront systématiquement la forme `<commande d'affichage> | <filtre>`, en particulier vous n'utiliserez qu'un seul filtre par ligne de commandes demandée.

Exercice 1.1 : Quelques filtres s'exécutant ligne par ligne

Q1. La commande `tr` permet de changer des caractères en d'autres caractères. Par exemple, `echo toto | tr "ot" "ap"` va retourner `papa`. En effet, les `o` sont remplacés en `a` et les `t` sont remplacés en `p`.
En vous servant du manuel si besoin, écrivez les lignes de commandes utilisant la commande `tr` permettant :

1. d'afficher le contenu du fichier `tuyaux` avec tous les caractères «`i`» remplacés par des caractères «`I`» ;
2. d'afficher le contenu du fichier `tuyaux` avec tous les caractères «`e`» remplacés par des caractères «`a`» et tous les caractères «`a`» remplacés par des caractères «`e`» ;
3. d'afficher le contenu du fichier `tuyaux` avec tous les caractères «`:`» remplacés par des caractères retour à la ligne «`\n`» ;
4. d'afficher le contenu du fichier `tuyaux` avec tous les caractères «`:`» remplacés par des caractères retour à la ligne «`\n`» et tous les caractères «`i`» remplacés par des caractères «`I`» ;
5. d'afficher le contenu du fichier `tuyaux` avec toutes les lettres remplacées par leur équivalent en majuscules ;
6. d'afficher le contenu du fichier `tuyaux` en supprimant toutes les répétitions de lettres majuscules (on ne garde qu'une fois les lettres majuscules répétées) ;
7. d'afficher le contenu du fichier `tuyaux` en supprimant tous les chiffres.

Q2. La commande `cut` vous permet de ne garder que certaines parties des lignes lues.
En vous servant du manuel, écrivez les lignes de commandes utilisant la commande `cut` permettant :

1. d'afficher le premier caractère de chaque ligne du fichier `tuyaux` ;
2. d'afficher le contenu du fichier `tuyaux` en supprimant le premier caractère de chaque ligne ;
3. d'afficher le contenu du fichier `tuyaux` en conservant, pour chaque ligne, uniquement le deuxième caractère et les caractères à partir du 6e ;
4. d'afficher le contenu du fichier `tuyaux` en supprimant le second **mot** de chaque ligne (un mot est une suite de caractères ne contenant pas d'espace. Un mot est donc un champ situé entre deux délimiteurs qui sont soit des espaces soit le début ou la fin de la ligne).
5. d'afficher pour chaque ligne du fichier `tuyaux` le texte écrit entre les deux premiers `c` présents sur la ligne.

Exercice 1.2 : Quelques filtres globaux

Q1. Les commandes `head` et `tail` permettent respectivement de n'afficher que les premières et les dernières lignes données dans leur entrée standard.
En vous servant du manuel, écrivez les lignes de commandes utilisant la commande `head` et/ou `tail` permettant :

1. d'afficher les deux premières lignes du fichier `tuyaux`.
2. d'afficher les deux dernières lignes du fichier `tuyaux`.

Q2. La commande `grep` permet de sélectionner les lignes qui contiennent un mot précis. C'est une commande très utilisée et dont la maitrise est indispensable.
La lecture du manuel de `grep` est assez longue. Pour des raisons pédagogiques, il vous est fortement conseillé d'utiliser la syntaxe donnée par la deuxième ligne du SYNOPSIS, et de lire les parties DESCRIPTION et Matching Control (dans OPTION). Le reste du manuel, aussi intéressant soit-il risquerait d'embrouiller plus qu'il ne vous servirait. Pour les plus avancés, on notera la section REGULAR EXPRESSIONS qui est complexes à prendre en main, mais très importante.
En vous servant du manuel, écrivez les lignes de commandes utilisant la commande `grep` permettant :

1. d'afficher uniquement les lignes du fichier `tuyaux` contenant le *mot* `commande` ;
2. d'afficher les lignes du fichier `tuyaux` contenant la chaîne de caractères «`-c`» ;
3. d'afficher les lignes du fichier `tuyaux` **ne contenant pas** la chaîne «`-c`» ;
4. d'afficher les lignes du fichier `tuyaux` contenant le mot `tee` quelque soit la hauteur des lettres (capitale ou minuscule, on parle aussi de case) qui le composent.

Q3. La commande `sort` permet de trier les lignes lues. On notera que l'on peut trier par rapport à une clef (key) qui a un fonctionnement similaire à `cut`.

En vous servant du manuel, écrivez les lignes de commandes utilisant la commande `sort` permettant :

1. d'afficher les lignes du fichier `tuyaux` dans l'ordre alphabétique ;
2. d'afficher les lignes du fichier `tuyaux` dans l'ordre alphabétique donné par le 6e mot (celui après le chiffre entre parenthèses) ;

Q4. La commande `wc` permet d'obtenir des informations sur le nombre de caractères, mots et lignes lus.

En vous servant du manuel, écrivez les lignes de commandes utilisant la commande `wc` permettant :

1. d'afficher uniquement le nombre de caractères du fichier `tuyaux`.
2. d'afficher uniquement le nombre d'octets du fichier `tuyaux`.
3. d'afficher uniquement le nombre de lignes du fichier `tuyaux`.
4. d'afficher uniquement le nombre de mots du fichier `tuyaux`.

Q5. La commande `sed` permet d'éditer du texte lu. Cette commande est extrêmement puissante et permet une diversité d'utilisation très importante. Pour comprendre pleinement cette commande, le manuel de `sed` vous propose d'aller consulter une autre documentation (`info sed`). Pour cette question, nous nous intéresserons principalement à quelques-unes des fonctionnalités principales de `sed`. Exceptionnellement, pour cette exercice, je vous invite à vous contenter de l'explication de `sed` donnée ci-dessous. Elle simplifie énormément la compréhension de la commande et couvre 90% des usages que vous pourrez en avoir.

Comme les autres filtres, `sed` va lire des informations sur l'entrée standard et écrire sur la sortie standard. Dans un premier temps, on considérera que `sed` s'utilise de la manière suivante :

`sed [LIGNES]LETTRE[OPTION]`

où :

- `LIGNES` désigne le numéro de ou des lignes sur laquelle doit s'exécuter la commande. On distinguera plusieurs notations :
  - rien pour désigner toutes les lignes.
  - `N` pour désigner la ligne `N`.
  - `N,M` pour désigner les lignes de `N` à `M`.
  - `/MOT/` pour désigner les lignes contenant `MOT`.
- `LETTRE` désigne l'action qui va être demandée
  - `p` pour afficher les lignes demandées. À utiliser avec l'option `-n`.
    Usage : `sed -n [LIGNES]p`
  - `d` pour supprimer les lignes demandées.
    Usage : `sed [LIGNES]d`
  - `s` pour effectuer un remplacement.
    Usage : `sed [LIGNES]s[OPTION]`
- `OPTION` va s'utiliser (pour nous) uniquement avec la lettre `s` et s'écrit : `/MOT/REMPLACEMENT/g` afin que toutes les occurrences de `MOT` soient remplacées par `REMPLACEMENT`.

ATTENTION : Il n'y a pas d'espace entre LIGNE, LETTRE et OPTION !

En vous servant des indications précédentes, écrivez les lignes de commandes utilisant la commande `sed` permettant :

- d'afficher uniquement la ligne 3 du fichier `tuyaux` ;
- de supprimer la première ligne du fichier `tuyaux` ;
- de remplacer toutes les occurrences du mot `commande` par le mot `chouette` dans le fichier `tuyaux` ;
- de supprimer les lignes 3, 4 et 5 du fichier `tuyaux` ;
- de remplacer, dans les lignes 3 et 4, toutes les occurrences du mot `commande` par le mot `chouette` dans le fichier `tuyaux`.

2 Combinaison de filtres

Maintenant que vous avez pris en main les filtres, vous pouvez observer que la combinaison `<commande d'affichage> | <filtre>` vous produit elle-même un affichage. De ce fait vous pouvez tout à fait enchainer cette combinaison avec un autre filtre. C'est d'ailleurs l'un des intérêts des filtres. En pratique lorsque l'on enchaine plusieurs filtres on aura donc une ligne de commandes de la forme :

`<commande d'affichage> | <filtre_1> | <filtre_2> | <filtre_3> ...`

Exercice 2.1 : Quelques combinaisons simples

Dans cet exercice, vous devez répondre aux questions en utilisant des lignes de commandes de la forme mentionnée juste au-dessus. À vous de définir de combien de filtres vous aurez besoin.

Q1. Affichez uniquement la 3e ligne du fichier `tuyaux`.

Q2. Affichez dans l'ordre lexicographique le second caractère de chaque ligne du fichier `tuyaux` ;

Q3. Affichez uniquement la 5e ligne du fichier `tuyaux` en remplaçant toutes les lettres par leur équivalent en minuscules ;

Q4. Affichez le nombre de lignes du fichier `tuyaux` contenant **exactement** le mot *pour* ;

Q5. Parmi les lignes du fichier `tuyaux` contenant le mot `tee`, affichez les deux premières.

Q6. Affichez le nombre de mots de la dernière ligne du fichier `tuyaux`.

Q7. Affichez uniquement le premier mot (`:XXX:`) des lignes du fichier `tuyaux` contenant le mot `pipe`.

Q8. Affichez le nombre de lignes du fichier `tuyaux` contenant le mot *fichier* dans le terminal courant et les lignes contenant le mot *fichier* dans un second terminal que vous aurez ouvert pour l'occasion (la compréhension du filtre `tee` pourrait vous être utile) ;

Q9. Affichez le **nombre** de lignes du fichier `tuyaux` contenant le mot *commande* dans le terminal courant et les **lignes** contenant le mot *commande* dans un second terminal. Les lignes apparaissant sur le second terminal devront être triées sur le cinquième mot de chaque ligne.

Exercice 2.2 : Au dela du fichier `tuyaux`

Une autre fonction d'affichage est `getent`. Elle permet d'obtenir des informations contenues dans certains fichiers du système. Dans cet exercice on utilisera `getent group` et `getent passwd`. Pour comprendre comment est structuré l'affichage produit par ces commandes d'affichage, il faut aller regarder le manuel de `group` et de `passwd`.

Q1. En combinant des filtres via des tubes, et en utilisant `getent passwd` comme fonction d'affichage, affichez le nombre d'étudiants (finissant par `.etu`) ayant un compte sur les machines du département.

Q2. En combinant des filtres via des tubes, et en utilisant `getent group` comme commande d'affichage, affichez l'ensemble des groupes du département info (contenant le mot clef `info`) à raison d'un groupe par ligne, trié dans l'ordre alphabétique.

Q3. En combinant des filtres via des tubes, et en utilisant `getent group` comme commande d'affichage, affichez les membres de votre groupe à raison d'un membre par ligne. (Attention aux identifiants contenant un numéro.)

3 Utilisation des filtres sur de gros fichiers

L'administration française rend disponible publiquement de plus en plus de données via le site web data.gouv.fr. Vous allez travailler sur certains de ces jeux de données pour manipuler les filtres shells via les tubes.

Exercice 3.1 : Recherche de données dans l'annuaire de l'éducation

Le jeu de données utilisé dans cet exercice est l'annuaire de l'éducation.

Le fichier `annuaire.csv` contient ces données. C'est un fichier dont chaque ligne contient un *enregistrement* : les données pour un établissement d'éducation. Chaque ligne est structurée de manière identique en plusieurs colonnes. La première ligne du fichier donne le nom de chaque colonne. Les colonnes sont séparées les unes des autres par le caractère "`;`".

Le fichier `annuaire.csv.gz` est une version comprimée du fichier `annuaire.csv`. Pour afficher son contenu sur la sortie standard, il suffit d'utiliser la commande `zcat annuaire.csv.gz`. Cette dernière commande nous servira donc de commande d'affichage.

Q1. Copiez le fichier `/home/public/baste/R1.04/TP04/annuaire.csv.gz` dans votre dossier `TP04`. (Pour les personnes utilisant leur ordinateur personnel, le fichier peut aussi être trouvé sur Moodle.)

Q2. À l'aide d'une ligne de commande utilisant des tubes, filtres et redirection, créez un fichier `annuaire.colonnes` contenant une ligne numérotée pour chaque entête de colonne du fichier `annuaire.csv`. (Aide : Il est intéressant de regarder la première ligne retournée par `zcat annuaire.csv.gz` pour comprendre ce que vous devez faire.) Votre fichier `annuaire.colonnes` doit avoir la forme suivante :

```
1  identifiant_de_l_etablissement
2  nom_etablissement
3  type_etablissement
[...]
```

Q3. Créez un fichier `annuaire` contenant uniquement la commande `zcat annuaire.csv.gz | sed 1d`. En utilisant la commande `chmod`, rendez le fichier annuaire exécutable par l'utilisateur.

Le fichier `annuaire` peut désormais être traité comme une commande shell. Vous pouvez l'appeler en utilisant `./annuaire`. Il affiche toutes les lignes du fichier `annuaire.csv` sauf la première. Il servira de commande d'affichage pour la suite de l'exercice.

Q4. En appelant la commande shell `annuaire`, affichez l'ensemble des lignes du fichier `annuaire.csv` sauf la première. Pourquoi vous ne pouvez pas simplement écrire `annuaire` pour l'exécuter ?

Q5. Utilisez cette commande connectée à d'autres filtres via des tubes pour effectuer les actions suivantes :

1. afficher le **nombre d'établissements** ;
2. afficher les valeurs utilisées pour la colonne `Statut_public_prive` ;

   Les valeurs à trouver sont :
   - une valeur vide
   - -
   - Public
   - Privé
3. afficher le **nombre d'établissements publics** ;
4. afficher le **nombre de *Lycée*** ;
5. afficher le **nombre de lycées publics** ;
6. afficher le **nombre de noms différents** parmi tous les établissements ;
7. afficher le **nombre d'établissement** portant le nom de BRASSENS ;
8. afficher l'**identifiant du lycée dans lequel vous avez préparé votre terminale** ;
9. afficher le **nombre de lycée publics dans le nord** (département de code 59) ;
10. afficher le nombre de lycée publics dans le nord (département de code 59) et leur nom dans un second terminal.

Exercice 3.2 : Les prénoms en France

Le jeu de données utilisé dans cet exercice est le fichier des prénoms de 1900 à 2019.

Le fichier `prenoms` est un fichier qui contient le nombre de naissances déclaré par prénom et département pour toutes les années entre 1900 et 2019. Chaque ligne est structurée de manière identique en plusieurs colonnes. La première ligne du fichier donne le nom de chaque colonne. Les colonnes sont séparées les unes des autres par le caractère "`:`". Pour la colonne du sexe, 1 correspond aux garçons et 2 aux filles.

Le fichier `prenoms.gz` est une version comprimée du fichier `prenoms`. Pour afficher son contenu sur la sortie standard, il suffit d'utiliser la commande `zcat prenoms.gz`. Comme pour l'exercice précédent, cette dernière commande nous servira temporairement de commande d'affichage.

Q1. Copiez le fichier `/home/public/baste/R1.04/TP04/prenoms.gz` dans votre dossier `TP04`. (Pour les personnes utilisant leur ordinateur personnel, le fichier peut aussi être trouvé sur Moodle.)

Q2. Observez la première ligne affichée par `zcat prenoms.gz` pour comprendre à quoi correspond chaque colonne.

Q3. Créez un fichier `prenoms` contenant uniquement la commande `zcat prenoms.gz | sed 1d`. En utilisant la commande `chmod`, rendez ce fichier exécutable par l'utilisateur.

Le fichier `prenoms` peut désormais être traité comme une commande shell. Vous pouvez l'appeler en utilisant `./prenoms`. Il affiche toutes les lignes du fichier `prenoms` sauf la première. Il servira de commande d'affichage pour la suite de l'exercice.

Q4. En appelant la commande shell `prenoms`, affichez l'ensemble des lignes du fichier `prenoms` sauf la première. Pourquoi vous ne pouvez pas simplement écrire `prenoms` pour l'exécuter ?

Q5. À l'aide d'une ligne de commandes utilisant des tubes, filtres et redirection, créez un fichier `prenom.colonnes` contenant une ligne numérotée pour chaque entête de colonne du fichier `prenoms`.

Q6. Combien le fichier `prenoms` contient-il de lignes (sans compter la première) ?

Q7. Connectez des filtres via des tubes pour effectuer les actions suivantes :

1. afficher le nombre de prénom de filles différents ;
2. afficher le nombre de prénoms de garçons différents en 2000 ;
3. afficher la liste des prénoms de filles différents en 2000 dans le département du nord (59) et du pas de Calais (62) ;
4. afficher le prénom utilisé dans le plus de départements pour les garçons en 1900.

4 (avancé) `sed` : un éditeur de texte

L'utilisation de `sed` mentionnée dans la première section est l'utilisation la plus courante de `sed`. Ce filtre permet en réalité bien plus que le fragment qui a été présenté.

Exercice 4.1 : Documentation de `sed`

L'objectif de cet exercice est de vous informer du fait qu'il n'existe pas qu'un seul manuel pour les commandes.

Q1. Lire la section SEE ALSO du manuel de `sed`

Q2. Lire `info sed`

Exercice 4.2 : Mise en pratique

Q1. Trouver comment, en utilisant un seul filtre `sed` vous pouvez supprimer les lignes 1, 3 et 6 du fichier `tuyaux`.

Q2. Trouver comment, en utilisant un seul filtre `sed` vous pouvez supprimer toutes les lignes paires du fichier `tuyaux`. Votre approche doit pouvoir fonctionner si le fichier fait 10 000 lignes.

Q3. Trouver comment ajouter du texte grâce à la commande `sed`. Avant la première ligne, après la dernière ligne ?

Q4. Trouver comment remplacer la deuxième ligne du fichier par le texte `toto`.

Exercice 4.3 : Encore plus loin

L'utilisation d'expression régulière revient régulièrement en informatique. Il y en avait par exemple lorsque vous avez voulu configurer `nano`. Ici avec `sed`, vous retrouvez cette notion. Si vous êtes arrivé à ce stade du TP, prenez le temps de comprendre ce qu'est une expression régulière et comment vous en servir. Attention, lorsque vous utilisez des expressions régulières avec `sed`, vous devez mettre le block `[LIGNES]LETTRE[OPTION]` entre guillemets.

Q1. Faites en sorte qu'avec un seul appelle au filtre sed que les occurrences des mots `le` et `la` soient remplacées par `lu`.

Q2. Assurez-vous que pour la question précédente, les occurrences de `les` ne soient PAS remplacées par `lus`.

Q3. Faites en sorte qu'avec un seul appelle au filtre sed que pour chaque occurrence des mots `le` et `la`, le `l` passe en majuscule.

Q4. Faites en sorte qu'avec un seul appelle au filtre sed, tous les mots commencent par une majuscule.

Q5. Faites en sorte qu'avec un seul appelle au filtre sed, le passage qui à la forme `:XXX:` ait désormais la forme `:x:X:x:`.