1 Consignes

1. Vous devez **impérativement** utiliser les options de compilation `-Wall -Werror` ;
2. lisez **attentivement** la totalité de l'énoncé avant de foncer tête baissée ;
3. Vous rendrez **un seul fichier** nommé `prenom_nom.c` (en utilisant évidemment vos nom et prénom) ;
4. Vos NOM, PRÉNOM GROUPE et NOM DE MACHINE doivent figurer en commentaire sur la première ligne de votre fichier ;
5. Vous penserez à traiter au maximum les erreurs possibles !
6. Commentez votre code de façon pertinente.

2 Travail demandé

Pour ce contrôle TP, vous allez écrire un programme qui permet d'effectuer des traitements sur des fichiers. Les possibilités que devra offrir votre programme sont les suivantes

— afficher un fichier ;
— afficher les 10 premières lignes d'un fichier ;
— afficher les 10 dernières lignes d'un fichier ;
— afficher le nombre de lignes, de mots et de caractères d'un fichier.

Le but de ce contrôle n'étant pas de réinventer la roue, vous allez utiliser les commandes déjà existantes afin de réaliser toutes ces fonctionnalités :

— `cat` pour afficher un fichier ;
— `head` pour afficher les 10 premières lignes ;
— `tail` pour afficher les 10 dernières lignes ;
— `wc` pour afficher le nombre de lignes de mots et de caractères.

Ces quatre commandes peuvent effectuer le traitement sur leur entrée standard. C'est pourquoi votre programme lancera la commande correspondant au traitement à effectuer en envoyant le contenu du fichier à traiter sur son entrée standard à l'aide d'un tube. L'appel à la commande se fera sans paramètre. Le programme final choisira donc, en fonction des paramètres passés, la bonne commande à exécuter pour effectuer le traitement.

Q 1. Écrivez une fonction *(3 points)*

```
int envoyer_donnees(int in, int out);
```

qui lit les données provenant du descripteur `in` à l'aide de la fonction `read()` et les envoient sur le descripteur `out` à l'aide de la fonction `write`. La fonction doit retourner -1 en cas d'erreur et 0 quand la fin du fichier d'entrée est atteinte.

Q 2. Écrivez une fonction *(3 points)*

```
int lance_commande(int in, const char *commande);
```

qui lance la commande `commande` sans paramètre et redirige l'entrée standard sur le descripteur `in` (supposé déjà ouvert). Cette fonction devra au préalable créer un processus fils qui exécutera la commande désirée. Elle devra retourner -1 en cas d'erreur et le `pid` du processus créé en cas de succès.

Q 3.Écrivez une fonction *(3 points)*

```
int lance_traitement(int in, const char *commande);
```

qui effectuera le traitement sur le descripteur `in` (supposé déjà ouvert) à l'aide de la commande `commande`. Cette fonction devra utiliser les fonctions des deux questions précédentes. Elle créera un premier processus chargé d'exécuter la fonction `envoyer_donnees()` ainsi qu'un tube permettant à celle ci d'envoyer les données contenues dans `in` à la commande `commande`. La commande sera lancée à l'aide de la fonction `lance_commande` de façon à ce qu'elle lise les données depuis la sortie du tube.

Cette fonction devra attendre la fin du traitement et retourner le code de retour de la commande `commande` (ou -1 si une erreur se produit).

Q 4. *(3 points)*

Vous allez maintenant écrire le `main` du programme qui devra appeler la fonction `lance_traitement()` avec les bons paramètres en fonction des arguments passés à votre programme. Ces arguments seront stricts afin de simplifier le sujet et sont les suivants

- le premier paramètre doit être une option qui spécifie le traitement (et donc la commande) à utiliser. Il peut prendre les valeurs suivantes :
  - `-c` : afficher le fichier (`cat`) ;
  - `-h` : afficher le début du fichier (`head`) ;
  - `-t` : afficher la fin du fichier (`tail`) ;
  - `-w` : afficher le nombre de lignes, de mots et de caractères du fichier (`wc`).
- le deuxième paramètre contient le nom du fichier à traiter. Si ce paramètre n'est pas présent, le fichier à traiter est l'entrée standard.

Le programme devra donc ouvrir le fichier à traiter (si nécessaire) et passer son descripteur à la fonction `lance_traitement()`. Il est rappelé que l'ouverture d'un fichier se fait par la primitive `open()` et que la fermeture d'un descripteur (fichier, tube,...) se fait par la primitive `close()`.

Ne sortez pas de la salle avant de vous être assuré auprès de l'enseignant que le TP a été correctement rendu