Exercice 1 - My cat

- Utiliser l'appel système `open()` pour ouvrir un fichier qui sera spécifié sur la ligne de commande. Lire les 10 premiers octets du fichier avec l'appel système `read()`, puis les afficher sur la sortie standard en utilisant `write()`. Utiliser un buffer statique.
  Rappel : `unistd.h` definit `STDIN_FILENO`, `STDOUT_FILENO`, et `STDERR_FILENO` pour les descripteurs de `stdin`, `stdout`, et `stderr`.
- Toujours en utilisant un buffer statique, écrire un programme fonctionnant comme `cat`, c'est-à-dire ouvrant un par un les fichiers spécifiés sur la ligne de commande, les lisant, et les écrivant sur la sortie standard.

Exercice 2 - My cp

Réécrire un programme `cp` simple (copie d'un fichier vers un autre). Pour créer le fichier "cible", n'oubliez pas d'utiliser les flags `O_CREAT`, `O_TRUNC`, et le troisième paramètre de `open()`, pour positionner correctement les droits du fichier nouvellement créé (regardez ce qui se passe si vous ne mettez pas cet argument).

Exercice 3 – Redirection

Ecrire un programme qui redirige stdout dans un fichier dont le nom est passé en argument. Si le fichier n'existe pas, vous devrez le créer. S'il existait, vous devez le vider. Pour vérifier si cela fonctionne, faites quelques printf dans le programme, puis regardez le contenu du fichier.

Exercice 4 - Re-redirection

Ecrire un programme qui redirige stdin dans un fichier dont le nom est passé en argument. Si le fichier n'existe pas le programme s'arrête. Pour vérifier si cela fonctionne, faites quelques scanf, et pour chaque lecture xxxx réussie, afficher sur la sortie standard : "J'ai lu xxxx".

Exercice 5 – wc

Ecrire un programme qui fait la même chose que la commande shell `wc`. Utiliser les appels systèmes et gérer les options c, l et w

Exercice 5 - Bonus 1: Highlander

Écrire un programme qui "résiste" au signal `SIGINT` (rappel: c'est ce signal qui est envoyé lorsque vous tapez `Control-C` dans le terminal). Pour cela utiliser l'appel système `signal()`.

Exercice 6 - Bonus 2: speed-O-meter

Écrire un speed-o-meter, un programme qui mesure la vitesse à laquelle on lui envoie des données (par l'entrée standard), en octets par seconde.

Exercice 7 - Bonus 3: shaper

Écrire un shaper, un programme qui lit sur l'entrée standard et écrit sur la sortie standard, mais n'écrit pas plus vite qu'une certaine vitesse spécifiée en octets par seconde.