Université de Strasbourg
Licence 2 Informatique

Programmation système
2023/2024 – printemps

TP noté

Durée : 1h30
Ce TP noté est individuel, toute communication avec un tiers est strictement interdite.
L'utilisation d'assistants de code (Copilot, ChatGPT, ...) est formellement prohibée.
Vous avez le **droit** de consulter vos anciens TP.
La navigation sur internet est **interdite** sauf pour consulter votre dépôt git ou pour accéder à la page Moodle du cours de Programmation Système.

Tout manquement constaté à ces consignes fera l'objet d'une procédure disciplinaire.

Dans ce TP noté vous devrez écrire le programme `revrot` qui s'utilise avec 0 ou 2 arguments :

```
Usage: revrot  [entree sortie]
```

où `[entree sortie]` signifie que le couple de paramètres `entree sortie` est optionnel.

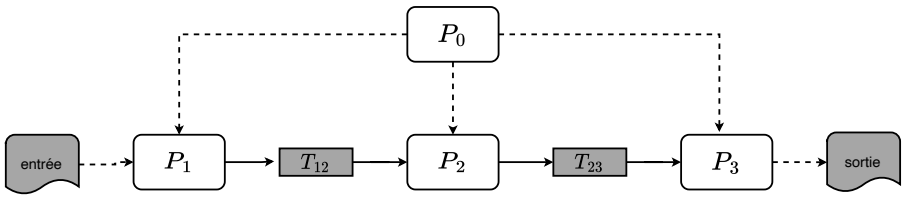
Dans le cas où les arguments `[entree sortie]` ne sont pas spécifiés, le programme attend les données sur l'entrée standard et écrit le résultat sur la sortie standard.

Les syntaxes valides sont donc, par exemple, les suivantes :

```
./revrot fichier1 fichier2
./revrot
./revrot < fichier1 > fichier2
```

Dans le cas où `entree` et `sortie` sont spécifiés en paramètres, `entree` doit désigner un fichier existant et `sortie` doit désigner un fichier qui sera produit au cours du traitement. Si le fichier `entree` n'existe pas ou que le fichier `sortie` ne peut pas être créé, le processus s'arrête et retourne le code d'erreur 1 : il ne lance pas les trois processus fils.

Le traitement effectué par `revrot` est réalisé à l'aide de trois processus fils P_1, P_2, et P_3 et de deux tubes T_{12} et T_{23}. Le schéma du système doit être le suivant :



Description des processus

- Le processus initial P_0 vérifie les arguments passés en ligne de commande, puis le cas échéant lance trois processus fils P_1, P_2, P_3. Il attend ensuite ses trois fils et récupère leurs trois codes de retour. Il retourne ensuite lui-même un code de retour en fonction des valeurs des codes de retour de ses fils (voir détail plus bas).
- Le processus P_1 redirige sa sortie standard vers l'entrée du tube T_{12} puis exécute à l'aide de la primitive `execlp` la commande 'rev' [1] :
  - sans argument si `[entree sortie]` n'a pas été spécifié en paramètres (la commande lira donc ses données sur l'entrée standard) ;
  - avec l'argument `entree` sinon (la commande s'exécutera donc sur ce fichier).
- Le processus P_2 lit les caractères, tant qu'il y en a, sur la sortie du tube T_{12}, effectue un traitement (chiffrage ROT13) sur ces caractères et écrit le résultat du traitement sur l'entrée du tube T_{23}. Le détail du traitement effectué est détaillé plus bas.

1. La commande 'rev' inverse l'ordre des caractères d'un fichier, ligne par ligne, et écrit le résultat sur la sortie standard.

— Le processus P_3 redirige la sortie du tube T_{23} vers son entrée standard puis exécute à l'aide de la primitive `execlp` la commande 'cat' :
  — si `[entree sortie]` n'a pas été spécifié en paramètres, 'cat' écrira, par défaut sur la sortie standard ;
  — si `[entree sortie]` a été spécifié en paramètres, P_3 redirige sa sortie standard vers le fichier `sortie` avant l'exécution de 'cat', de manière à ce que 'cat' écrive dans ce fichier.

Détail du processus P_2 Le processus P_2 chiffre les caractères lus sur la sortie du tube T_{12} et les écrit sur l'entrée du tube T_{23}. Le chiffrage s'appuie sur l'algorithme ROT13, qui consiste à décaler de 13 caractères chaque lettre du texte à chiffrer. Si le caractère est un chiffre, on applique le ROT5 en le décalant de 5 caractères. Vous devrez donc écrire l'algorithme de chiffrage suivant :

— si le caractère lu est une minuscule (fonction `islower` de la librairie C), on le décale de 13 caractères : on ajoute 13 à son code ASCII si le résultat est inférieur ou égal au code ASCII de 'z' ; sinon on soustrait 13 ;
— si le caractère lu est une majuscule (fonction `isupper` de la librairie C), on le décale de 13 caractères : on ajoute 13 à son code ASCII si le résultat est inférieur ou égal au code ASCII de 'Z' ; sinon on soustrait 13 ;
— si le caractère lu est un chiffre (fonction `isdigit` de la librairie C), on le décale de 5 caractères : on ajoute 5 à son code ASCII si le résultat est inférieur ou égal au code ASCII de '9' ; sinon on soustrait 5.
— si le caractère lu n'est ni une lettre ni un chiffre, le caractère est inchangé

A titre de débuggage, l'algorithme que vous devez écrire doit produire le même résultat que la commande suivante :

```
tr 'A-Za-z0-9' 'N-ZA-Mn-za-m5-90-4'
```

Cependant il vous est demandé de coder cet algorithme vous même, et de ne pas vous appuyer sur la commande précédente.

Code de retour du programme Le processus principal P_0 retourne le code suivant :

— 0 si et seulement si les trois processus fils se terminent avec succès ;
— 2 si l'un au moins des processus fils est arrêté par la réception d'un signal
— 1 dans tous les autres cas (aucun processus arrêté par un signal, au moins un échoue)

Tests Le script `test.sh` permet de tester votre programme :

— `./test.sh small` : un test simple, n'allez pas plus loin tant que ce test échoue
— `./test.sh large` : un test plus long (compter 10 secondes)
— `./test.sh errs` : pour tester la robustesse de votre programme aux erreurs (une partie de ces tests ne fonctionne que sous Linux)

Pour avoir plus détails sur les erreurs, vous pouvez activer le mode verbose avec `export VERB=0`.

Consignes Votre programme doit :

1. vérifier que le nombre d'arguments est valide ;
2. vérifier la validité d'accès en lecture de `input` et la validité d'accès en écriture de `output` si ces arguments ont été spécifiés ;
3. vérifier le code de retour de toutes les primitives système.

Vous écrirez le programme `revrot` en langage C en n'utilisant que des primitives systèmes ; les seules fonctions de bibliothèque autorisées sont `perror` et `exit`. Vous pouvez également utiliser la fonction `raler` qui vous est fournie dans le fichier `revrot.c` ou la macro `CHK`.

Vous devrez rendre sur Moodle un unique fichier `revrot.c` Votre programme doit compiler sur **turing** avec `cc -Wall -Wextra -Werror -Wvla`. Les programmes qui ne compilent pas avec cette commande ne seront pas examinés.