Mini-projet de travaux pratiques
μsh, le micro shell

Le projet est à effectuer en binôme (ou exceptionnellement, en monôme). Le rapport complet **imprimé** est à déposer dans la boîte aux lettres de l'enseignant encadrant la séance de travaux pratiques. Le code ainsi que le rapport de projet au format PDF sont à déposer *impérativement* sur Madoc sous la forme d'une archive [1] dont le nom sera composé des noms de chacun des étudiants du binôme — ex. `tartempion-dupont.tar.gz`. La décompression de cette archive devra créer un répertoire `tartempion-dupont/` contenant les fichiers. On re-précisera en commentaire dans les fichiers de code source les noms et prénoms de chacun des étudiants.

Un projet rendu après la **date limite fixée au lundi 29 avril 2019, 23h55**, sera affecté d'un malus proportionnel au nombre de jours de retard, dans la limite de deux jours.

Travail à effectuer

On souhaite développer un micro-shell à l'image de Bash. Le programme doit présenter en boucle une invite (« *prompt* ») pour entrer une ou plusieurs commandes et les exécuter. S'il est appelé avec un nom de fichier en paramètre, il doit exécuter les commandes se trouvant dans le fichier (exécution « *batch* ») puis rendre la main.
À l'invite, on peut rentrer une ou plusieurs commandes. Les commandes peuvent être séparées les unes des autres par un point-virgule, auquel cas l'exécution se fait séquentiellement, ou par les caractères « `//` », auquel cas l'exécution est faite en parallèle.

Example.

```
goualard@port-goualard> ./mush
mush> ls *.cpp; ls *.hpp // ls *.o; echo pomme
common.cpp  main.cpp  mush.cpp  parsing.cpp
common.hpp  mush.hpp  parsing.hpp
common.o  mush.o  parsing.o
pomme
mush> quit
goualard@port-goualard>
```

Dans l'exemple précédent, la commande « `ls *.cpp` » est d'abord exécutée, puis les commandes « `ls *.hpp` » et « `ls *.o` » sont exécutées en parallèle, puis la commande « `echo pomme` » est exécutée quand les commandes précédentes ont fini leur exécution.
On notera que les expressions de globbing doivent être supportées. Le shell doit posséder la commande interne « `quit` » qui doit arrêter le programme et revenir au shell précédent.
Une attention particulière devra être apportée au rapport de projet. On trouvera sur madoc une description succincte des éléments qu'il doit contenir.

Aide

On trouvera sur madoc une archive de code offrant une fonction `parse_line()` de découpage d'une ligne d'instructions ainsi qu'un squelette de `Makefile`.
On pensera à utiliser les fonctions `fork()`, `execvp()` et `glob()`.
Le design du code devra être modulaire, en séparant dans des fichiers différents les différentes parties du programme et en écrivant systématiquement des fichiers d'en-tête pour les APIs des différentes unités de compilation.

1. Manipulation d'archives :
   — Création de l'archive `titi.tar.gz` à partir du répertoire `titi` : `tar -vzcf titi.tar.gz titi/`
   — Récupération du contenu de l'archive `titi.tar.gz` : `tar -vzxf titi.tar.gz`

Extensions possibles

On peut envisager d'étendre le micro-shell de façon à supporter plus de commandes internes, des structures de contrôle et des variables. Ce travail est hors barème et n'apportera pas de points si le travail minimal demandé n'est pas fait correctement.