Travail pratique #1

IFT-2245

January 10, 2017

¡¡ Dû le 30 janvier à 23h59 !!

1 Survol

Ce TP vise à vous familiariser avec la programmation système dans un système d'exploitation de style POSIX. Les étapes de ce travail sont les suivantes:

1. Parfaire sa connaissance de Python et POSIX.
2. Lire et comprendre cette donnée. Cela prendra probablement une partie importante du temps total.
3. Lire, trouver, et comprendre les parties importantes du code fourni.
4. Compléter le code fourni.
5. Écrire un rapport. Il doit décrire **votre** expérience pendant les points précédents: problèmes rencontrés, surprises, choix que vous avez dû faire, options que vous avez sciemment rejetées, etc... Le rapport ne doit pas excéder 5 pages.

Ce travail est à faire en groupes de 2 étudiants. Le rapport, au format LaTeX exclusivement (compilable sur `frontal.iro`) et le code sont à remettre par remise électronique avant la date indiquée. Aucun retard ne sera accepté. Indiquez clairement votre nom au début de chaque fichier.

Si un étudiant préfère travailler seul, libre à lui, mais l'évaluation de son travail n'en tiendra pas compte. Si un étudiant ne trouve pas de partenaire, il doit me contacter au plus vite. Des groupes de 3 ou plus sont **exclus**.

2 CH: un shell pour les hélvètes

Vous allez devoir implanter une ligne de commande, similaire à `/bin/sh`, qui sait:

1. Démarrer des processus externes: être capable d'exécuter des commandes comme "`cat Makefile`".
2. Expansion d'arguments: être capable d'exécuter des commandes comme "`echo *`".
3. Rediriger les entrées et sorties de ces processus: être capable d'exécuter des commandes comme "`cat <Makefile >foo`".
4. Connecter ces processus via des *pipes* pour faire des *pipelines*: être capable d'exécuter des commandes comme "`find -name Makefile | xargs grep ch`".

Tout cela bien sûr sans réutiliser un autre shell mais en utilisant les primitives du système d'exploitation, telles que `os.exec`, `os.fork`, `os.dup2`, `os.pipe`, `os.listdir`, `os.waitpid`, ... (et pas `os.system` ou `os.popen`, et pas un module comme `subprocess` non plus).

Le programme doit être exécutable sur `frontal.iro`. Cela ne vous empêche pas bien sûr de le développer sur un système différent, e.g. sous Windows avec Cygwin, mais assurez-vous que le résultat fonctionne *aussi* sur `frontal.iro`.

3 Cadeaux

Vous recevez en cadeau de bienvenue les fichiers `Makefile`, `rapport.tex`, et `ch.py` qui contiennent le squelette (vide) du code et du rapport que vous devez rendre.

3.1 Remise

Pour la remise, vous devez remettre deux fichiers (`ch.py` et `rapport.tex`) par la page Moodle (aussi nommé StudiUM) du cours. Assurez-vous que tout fonctionne correctement sur `frontal.iro`.

4 Détails

- La note sera divisée comme suit: 20% pour chacune des 4 fonctionalités, et 20% pour le rapport.
- Tout usage de matériel (code ou texte) emprunté à quelqu'un d'autre (trouvé sur le web, ...) doit être dûment mentionné, sans quoi cela sera considéré comme du plagiat.
- Le code ne doit en aucun cas dépasser 80 colonnes.

- Vérifiez la page web du cours, pour d'éventuels errata, et d'autres indications supplémentaires.
- La note sera basée d'une part sur des tests automatiques, d'autre part sur la lecture du code, ainsi que sur le rapport. Le critère le plus important, et que votre code doit se comporter de manière correcte. Ensuite, vient la qualité du code: plus c'est simple, mieux c'est. S'il y a beaucoup de commentaires, c'est généralement un symptôme que le code n'est pas clair; mais bien sûr, sans commentaires le code (même simple) est souvent incompréhensible. L'efficacité de votre code est sans importance, sauf s'il utilise un algorithme vraiment particulièrement ridiculement inefficace.