INF600A: Laboratoire #1
Pipelines avec `sed`, `find`, `xargs`, etc.

Jeudi 19 janvier 2017
13h30–15h30
PK-S1535

Le but de ce laboratoire est de vous familiariser avec l'utilisation des **pipelines** Unix et avec les commandes de base pour la recherche **de fichiers** et la recherche **dans des fichiers**, notamment, `grep`, `sed`, `diff`, `find` et `xargs`.

Pour ces exercices, divers fichiers vous sont fournis sous forme d'un dépôt `git`, que vous devez obtenir en exécutant la commande suivante :

```
$ git clone http://www.labunix.uqam.ca/~tremblay/git/LaboSedEtAl.git
```

Ce qui vous est fourni

Voici les principaux fichiers qui vous sont fournis :

- `*.sh` : Divers scripts `bash` que vous devez **compléter**, pour qu'ils réalisent les fonctionnalités présentées plus bas.
- `makefile` : Fichier qui permet d'automatiser l'exécution des scripts — au moins une cible par script (plusieurs cibles pour le dernier exercice).

  Les principales cibles associées à l'utilisation de ce `makefile` sont décrites dans chacun des exercices.

  Important : La première chose à faire lorsque vous aurez cloné le dépôt est d'aller dans le répertoire `LaboSedEtAl` puis exécuter la commande «`make`», ce qui modifiera les permissions des scripts pour les rendre exécutables.

  Suggestion : Ensuite, dans le fichier `makefile`, vous pouvez modifier la cible `default` pour qu'elle indique l'exercice sur lequel vous travaillez.
- `Fichiers/*.*` : Des fichiers de données utilisés par les scripts que vous devez définir.
- `Attendus/{blancs,enseignants,url}` : Les résultats attendus pour trois des scripts.

Quelle machine utiliser pour ce laboratoire?

Les solutions des exercices ont été testées sur `malt.labunix.uqam.ca` ainsi que sur mes deux machines personnelles — Linux/CentOS et MacBook/MacOS X. Si vous avez votre propre machine Linux, vous devriez pouvoir utiliser et définir ces scripts sans problème.

Si vous désirez plutôt utiliser `malt.labunix.uqam.ca` à partir du PK-S1535, voir à la fin du présent document (p. 10).

Suggestions/Indices

- Lorsque vous développez un script complexe, allez-y de façon incrémentale, donc une étape de pipeline à la fois.

  Et pour faciliter la lecture et la modification du script, indiquez une étape par ligne, en utilisant simplement `cat` comme dernière étape si nécessaire.

  Exemple :

  ```
  etape1 ... |
    etape2 ... |
    cat
  ```

- Pour un script qui doit modifier «en place» un fichier, commencez par développer une version qui génère les modifications demandées, *mais sans modifier le fichier initial.* Lorsque les modifications générées sont correctes, créez alors les fichiers modifiées — en créant des copies de sauvegarde au préalable.

---

1. Complétez le script `lister-meme-nom-extension.sh` qui doit effectuer la tâche suivante :

- Le script reçoit un argument, optionnel, qui doit être un nom de répertoire — si aucun argument est fourni, on utilise le répertoire courant «`.`».

  Note : Cette partie du script vous est fournie.
- Le script produit la liste des fichiers du répertoire indiqué **dont le nom et l'extension sont identiques**.

Le `makefile` qui vous est fourni définit la cible `lister` pour éxécuter ce script avec le répertoire «`Fichiers`». L'exécution devrait indiquer deux fichiers : `foo.foo` et `bar.bar`.

---

2. Complétez le script `supprimer-blancs.sh` qui doit effectuer les tâches suivantes :

- Le script reçoit en argument un nom de fichier.

  Note : Cette partie du script vous est fournie, qui vérifie qu'un argument est bien fourni et qui l'associe à la variable `FICH`.

- Le script génère un fichier de même nom que celui reçu en argument mais avec l'extension additionnelle «`.sans-blanc`». Dans ce fichier, **tous les blancs en fin de ligne ont été supprimés**.

- Le script vérifie ensuite a) qu'il y a bien des différences entre le fichier initial et le fichier généré lorsqu'on utilise `diff` sans aucune option (comparaison stricte), mais b) **qu'il n'y a aucune différence lorsqu'on utilise `diff` avec des options appropriées pour ignorer les différences au niveau des blancs.**

Le `makefile` qui vous est fourni définit la cible `blancs` pour éxécuter votre script avec le fichier «`Fichiers/avec-blancs.txt`».

Votre script doit donc génèrer le résultat dans «`Fichiers/avec-blancs.txt.sans-blanc`».

3. Complétez le script `trouver-usagers.sh` qui doit effectuer la tâche suivante :

- Le script analyse le fichier `/etc/passwd` pour produire une liste ordonnée (ordre alphabétique) des **noms d'usager**, et rien d'autre.

Le `makefile` qui vous est fourni définit la cible `usagers` pour éxécuter ce script.

Chaque ligne du fichier `/etc/passwd` donne les informations pour un (1) usager : `nom d'usager`, indicateur de mot de passe encrypté, UID, GID, etc. Tous ces champs sont séparés par des «`:`».

Pour plus de détails, voir l'URL suivant :
`http://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/`

Remarque : La mise en oeuvre de votre script doit utiliser `grep` et/ou `sed`. Essayez de compléter les deux mises en oeuvre et vérifiez que vos deux solutions donnent le même résultat.

---

4. Le fichier `Fichiers/journal-operations.txt` contient un extrait du journal (*log file*) de l'outil d'aide à la correction Oto.

Chaque ligne du journal indique une utilisation de l'outil Oto. Par exemple, voici deux lignes extraites de ce journal :

```
08/01/16 | 14:56:52 | tremblay_gu | decrire_boite bh
01/01/16 | 10:03:07 | cb491128 | lister_boites nkambou_r
```

Les champs de chaque ligne, séparés par le caractère «`|`», sont donc les suivants :

- Date
- Heure
- Nom d'usager
- Commande Oto exécutée par cet usager, à la date indiquée

Complétez le script `trouver-enseignants.sh` qui doit effectuer la tâche suivante :

- Le script analyse le fichier `Fichiers/journal-operations.txt` pour produire une liste alphabétique **des noms des enseignants** qui ont utilisé Oto.
- On considère qu'un **nom d'enseignant** est un nom d'usager **qui n'est pas** un nom d'usager étudiant.
- Un **nom d'usager étudiant** est identifié par deux lettres suivies de six chiffres.

Évidemment, chaque nom ne devrait apparaitre qu'une seule fois dans la liste.

Le `makefile` qui vous est fourni définit la cible `enseignants` pour éxécuter ce script.

5. Les fichiers `Fichiers/*.tex` sont quelques fichiers LaTeX pour des notes de cours (cours MGL7460). À divers endroits dans ces fichiers, on trouve des appels à la macro LaTeX `url`, par exemple :

```
\url{http://www.labunix.uqam.ca/~tremblay}

\url{https://www.ruby-lang.org/en/}

\url{geek-and-poke.com}
```

Complétez le script `trouver-urls.sh` qui doit effectuer la tâche suivante :

- Le script analyse les fichiers `Fichiers/*.tex` pour identifier les URLs qui n'ont pas le préfixe indiquant le protocole, **donc qui ne débutent pas** par une chaine de la forme *protocole*`://`, où *protocole* est composé de lettres ou chiffres (e.g., `http`, `https`, `ftp`, etc.).
- Plus exactement, le script **doit identifier les URLs sans protocole en indiquant aussi dans quel fichier cet URL se trouve**. Un extrait du résultat attendu est donc le suivant :

  ```
  $ make urls
  ./Fichiers/build.tex geek-and-poke.com
  ...
  ```

Le `makefile` qui vous est fourni définit la cible `urls` pour éxécuter ce script.

---

6. Un usager — Joe Bidon — possède de nombreux dépôts `git` et il a récemment migré ses dépôts de la machine `malt` vers `github`. Il doit maintenant mettre à jour, dans les copies locales de ses dépôts, les informations sur les dépôts distants. Pour ce faire, il veut définir un script `changer-remote.sh` :

- Le script analyse les répertoires accessibles du répertoire courant pour identifier ceux qui sont des dépôts `git`. Un tel répertoire est identifié par la présence d'un fichier «`.git/config`».
- Un attribut possible d'un dépôt `git` est la définition d'un **dépôt distant** — un dépôt *remote*. Ceci est indiqué par la présence dans le fichier `.git/config` d'une définition telle que la suivante :

  ```
  [remote "origin"]
          url = https://malt.labunix.uqam.ca/bidon_joe/GIT/Projet1
          fetch = +refs/heads/*:refs/remotes/origin/*
  ```

  Lorsqu'une ligne `url` est présente, le script va alors modifier l'`url`, et ce en modifiant directement le fichier `.git/config`. Par exemple, la modification suivante serait effectuée :

  `malt.labunix.uqam.ca/bidon_joe/GIT`
  ⇒
  `github.com/bidon-joe`

- De plus, le script va aussi conserver l'ancien fichier `.git/config` dans le fichier `.git/config.bak`.

Le `makefile` qui vous est fourni définit les cibles suivantes :

- `generer` : Crée trois dépots `git`, utilisés comme données pour le script.
- `remove` : Lance l'exécution du script.
- `reset` : Réinitialise les fichiers `.git/config`, donc retourne à un état équivalent à l'état initial obtenu avec `make generer`.
- `clean-all` : Supprime les dépôts `git`.

Donc, pour développer et tester votre script, vous devez procéder comme suit :

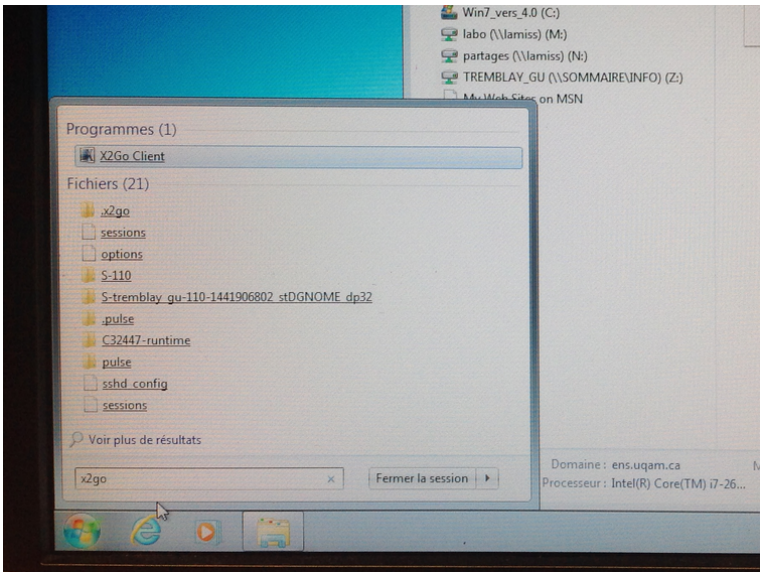
```
$ make generer
$ ... écrire le script ...
$ make remove
$ ... vérifier les résultats ...
$ make reset # Si les r\sultats n'etaient pas corrects
             # et que vous voulez modifier votre script
             # puis l'executer a nouveau avec des donnees 'fraiches'
```

Comment se connecter à `malt.labunix.uqam.ca` à partir d'un poste de travail dans le PK-S1535

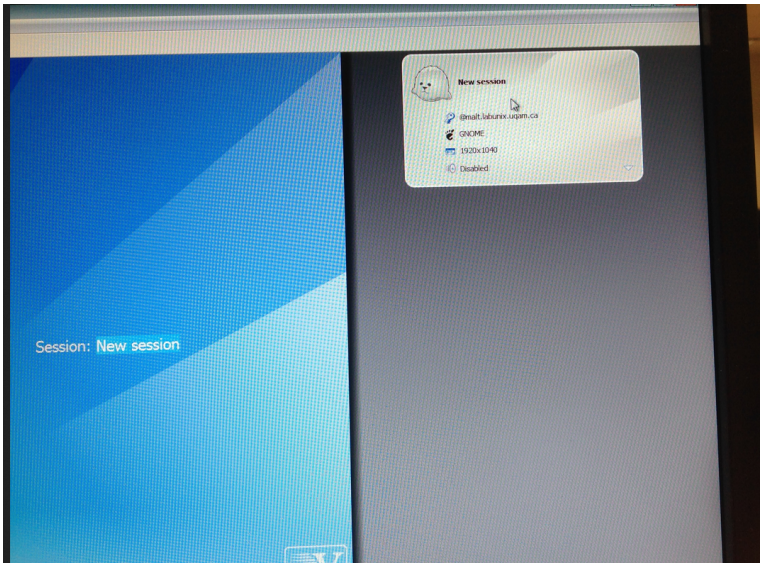
a. Connectez-vous tout d'abord à un poste Windows.

   Remarque importante : La **première fois** où vous allez vous logger sur un poste, cela pourrait prendre plusieurs (!) minutes avant que vous soyez connecté — la nouvelle configuration des postes doit être chargée et c'est un peu long ☹ Donc, soyez patient!

b. Une fois la session Windows ouverte, lancez l'application `x2go` — voir en bas à gauche :

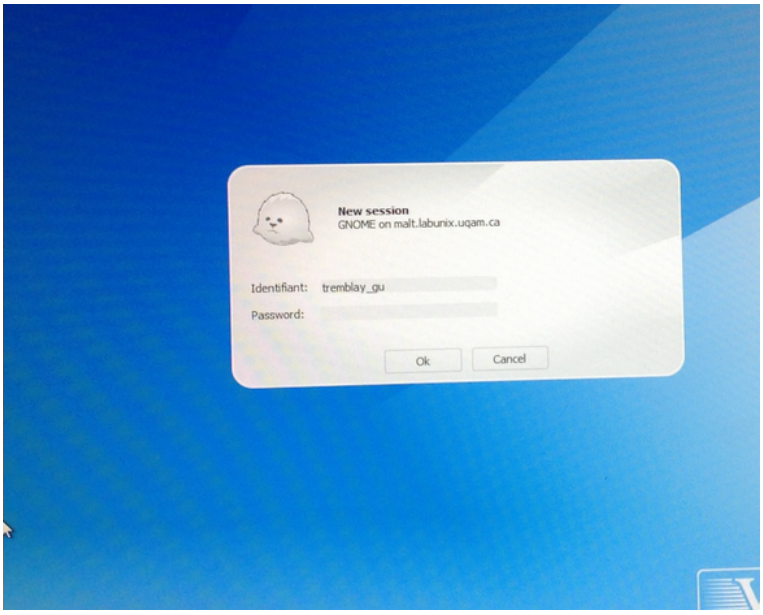


c. Dans l'application `x2go`, lancez une session sur `malt.labunix.uqam.ca` — voir en haut à droite :



Attention : Pour le type de session, par défaut, le type indiqué est «`KDE`» ; **il faut plutôt aller dans le menu et sélectionner `Gnome`.**

d. Connectez-vous au serveur `malt.labunix.uqam.ca` en utilisant votre nom usager et mot de passe habituels — les mêmes que pour la machine Windows :



Informations additionnelles sur x2go

- Lorsque vous êtes sur `malt`, votre répertoire de fichiers est exactement le même que sur `malt` ou sur n'importe quelle autre machine Linux du réseau `labunix`.
- Si vous ouvrez une fenêtre sur `malt` — peu importe avec quelle application — la fenêtre s'ouvrira sur votre poste de travail.
- **Important :** Pour fermer la connexion à `malt` créée avec `x2go`, il faut fermer **en cliquant sur le petit X de fermeture dans le coin droit en haut!**
- Si vous préférez travailler à partir de votre ordinateur portable via une connexion `ssh`, vous pouvez le faire, puisque `malt` accepte les connexions `ssh` provenant de l'extérieur du réseau UQAM.

Éditeurs de texte sur Unix/Linux

Divers éditeurs de texte sont disponibles sur `malt` :

- `vi` (`vim`)
- `emacs`
- `nano` :

  `http://mintaka.sdsu.edu/reu/nano.html`
- `gedit` :

  `https://en.wikipedia.org/wiki/Gedit`

  `https://help.gnome.org/users/gedit/stable`,

Plusieurs étudiants utilisent ce dernier éditeur, avec une interface graphique simple.