# INF600A: Laboratoire #2
# Utilisation de `awk`

Jeudi 20 septembre 2018
13h30–15h30
PK-S1565

Le but de ce laboratoire est de vous familiariser avec l'utilisation de l'outil **awk**. Vous devez donc réaliser les scripts des différents exercices en utilisant principalement `awk`.

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_gu/git/LaboAwk.git
```

## Ce qui vous est fourni

Voici les principaux fichiers qui vous sont fournis :

- `*.sh` : Scripts que vous devez **compléter**.
- `makefile` : Fichier qui permet d'automatiser l'exécution d'un petit test sur chacun des scripts. Les principales cibles sont décrites dans chacun des exercices.

  Important : Première chose à faire lorsque vous aurez cloné le dépôt = exécuter la commande «`make`» pour rendre exécutables les divers scripts.

  Suggestion : Ensuite, dans le fichier `makefile`, vous pouvez modifier la cible `default` pour qu'elle indique l'exercice sur lequel vous travaillez.
- `Fichiers/*.*` : Fichiers de données utilisés par les scripts que vous devez définir.
- `Attendus/*.*` : Résultats attendus pour l'exécution des scripts sur des données de tests.

### 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.

---

**1.** Le fichier `Fichiers/journal-operations.txt` contient un extrait du journal (*log file*) de l'outil d'aide à la correction Oto, où chaque ligne indique une utilisation, par exemple :

```
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 exécutée par l'usager, à la date indiquée

Complétez le script `anonymes.sh` qui doit effectuer la tâche suivante :

- Le script analyse le journal dont le nom de fichier est reçu en argument.
- Il émet sur `stdout` une version anonymisée du journal, où la colonne indiquant le nom d'usager a été supprimée.

Le `makefile` qui vous est fourni définit la cible `anonymes` pour éxécuter ce script. Voir le `makefile` pour un exemple/test.

Indices :

- On peut spécifier le séparateur de champ avec l'option `-F` ou avec la variable `FS`. Ici, c'est plus facile avec l'option `-F`. Mais attention : le caractère «`|`» a une signification spéciale au niveau du *shell*!

---

**2.** Complétez le script `uniq.sh` qui effectue la même tâche que la commande `uniq`. Pour simplifier, votre script peut ne traiter que le cas où c'est `stdin` qui est traité.

Le `makefile` qui vous est fourni définit la cible `uniq` pour éxécuter ce script. Voir le `makefile` pour un exemple/test.

Indices :

- Un script `awk` peut indiquer plusieurs actions, qui sont exécutées dans l'ordre.
- Une variable n'a pas besoin d'être déclarée. Elle est initialisée avec la chaine vide lors de sa première utilisation.

**Note :** Si vous voulez être plus général et permettre de traiter le cas avec un nom de fichier passé en argument, une cible appropriée est définie : `uniq_avec_fichier`.

---

**3.** Complétez le script `wc.sh` qui effectue la même tâche que la commande `wc`. Pour simplifier, votre script peut ne traiter que le cas où c'est `stdin` qui est traité.

Le `makefile` qui vous est fourni définit la cible `wc` pour éxécuter ce script. Voir le `makefile` pour un exemple/test.

Indices :

- Un script peut indiquer plusieurs actions, qui sont exécutées dans l'ordre.
- Une variable n'a pas besoin d'être déclarée. Elle est initialisée avec la chaine vide lors de sa première utilisation. Toutefois, une telle chaine vide est interprétée comme le nombre 0 dans un contexte numérique.
- Fonctions et variables utiles pour ce script :
  - `length(ch)` : Longueur de la chaine `ch`.
  - `NF` = *Number of fields* = Nombre de champ de l'enregistrement courant
  - `NR` = *Number of records* = Numéro de l'enregistrement courant (le premier enregistrement = 1).

**Note :** Si vous voulez être plus général et permettre de traiter le cas avec un ou plusieurs noms de fichier passés en argument, une cible appropriée est définie : `wc_avec_fichiers`.

---

**4.** Complétez le script `utilisations.sh` qui doit effectuer la tâche suivante :

- Le script analyse le journal des opérations Oto dont le nom de fichier est reçu comme premier argument pour déterminer **combien de fois** un usager — indiqué par le deuxième argument — a utilisé Oto.
- Il émet sur `stdout` le nom de l'usager et le nombre d'utilisations.

Le `makefile` qui vous est fourni définit la cible `utilisations` pour éxécuter ce script. Voir le `makefile` pour un exemple/test.

Indices :

- Voir plus haut l'indice sur la spécification du séparateur de champ.
- Voir plus haut l'indice sur la valeur initiale par défaut des variables numériques.
- Voir l'exemple à la fin du document sur `awk` pour l'utilisation d'un script défini avec des guillemets doubles, dans le but d'utiliser la bonne valeur d'une variable dans le script.