INF7235: Laboratoire #1
Parallélisme de style *fork/join* (`pcall`/`future`)

23 janvier 2017
PK-4665

Le but de ce laboratoire est de vous familiariser avec l'utilisation de la bibliothèque `PRuby` et ses constructions pour le **parallélisme *fork/join***, plus spécifiquement pour la mise en oeuvre de **parallélisme récursif**.

---

0 Comment se connecter à `japet.labunix.uqam.ca`

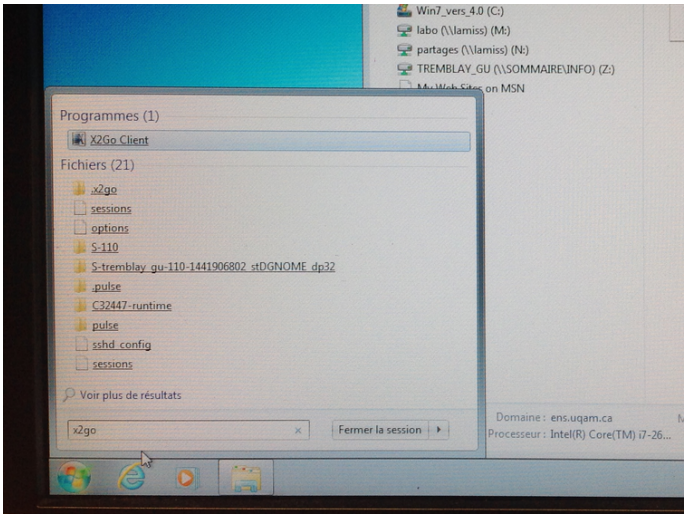
Remarque importante : S'il s'agit de votre premier trimestre à l'UQAM, il est important **qu'avant la première séance de laboratoire**, vous preniez quelques minutes pour essayer de vous connecter à `malt.labunix.uqam.ca` — avec une simple connexion `ssh` ou avec `Putty`. Le but est de vérifier **que votre compte est actif** et **que votre répertoire `home` a été créé**. Si vous ne réussissez pas à vous connecter ou que votre `home` n'est pas actif, contactez-moi et je ferai les démarches pour faire créer/activer votre compte.

Pour les premiers laboratoires de programmation parallèle en mémoire partagée, nous utiliserons la machine `japet.labunix.uqam.ca`, une machine multiprocesseurs à 64 coeurs/processeurs.

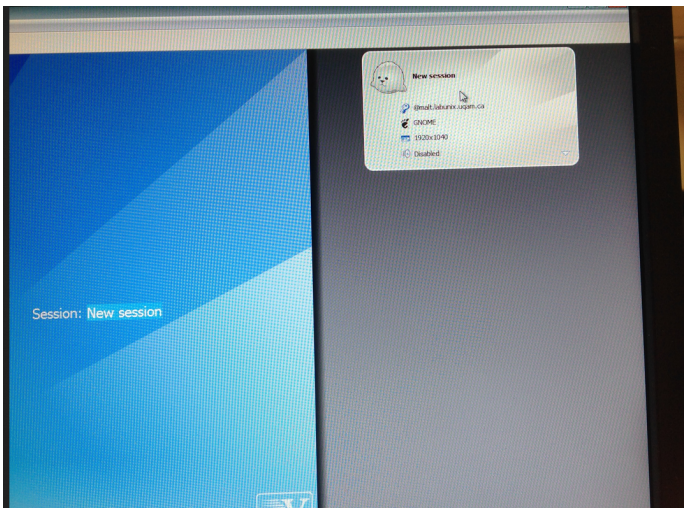
a. Connectez-vous tout d'abord à un poste Windows.

   Remarque importante : La **première fois** où vous vous loggerez 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 :

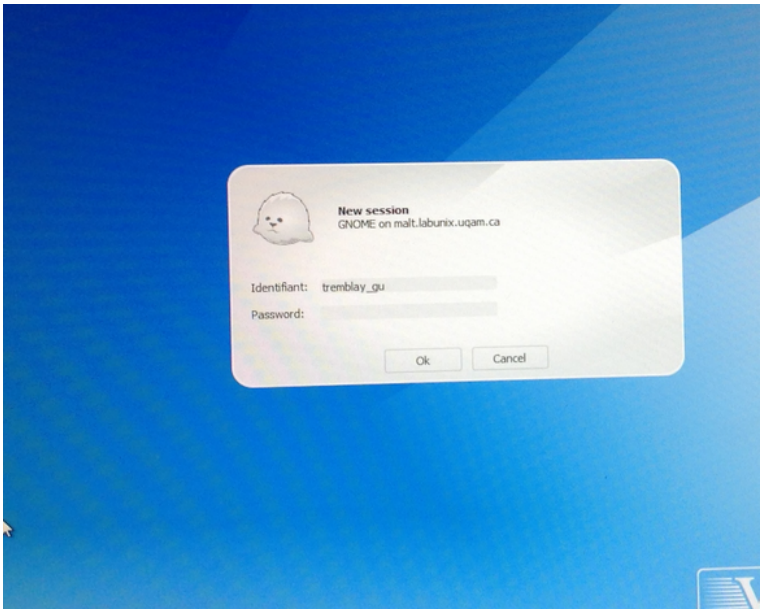


c. Dans l'application `x2go`, lancez une session sur `malt.labunix.uqam.ca` — voir en haut à droite :



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 :



e. Une fois sur `malt`, vous pouvez finalement vous connecter à `japet` — en mode ligne de commande. Ouvrez un terminal (avec la souris) puis lancez la commande suivante :

```
$ ssh japet.labunix.uqam.ca
```

0.1 Informations additionnelles sur `x2go`

- Lorsque vous êtes sur `japet`, 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 `japet` — 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. Toutefois, avant de vous connecter à `japet`, vous devez quand même vous connectez à `malt`, qui est le seul serveur acceptant les connexions `ssh` provenant de l'extérieur du réseau UQAM.

0.2 Informations additionnelles sur `Unix/Linux`

- **Éditeurs de texte** : Divers éditeurs de texte sont disponibles sur `japet` :
  - `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 étudiantes utilisent ce dernier éditeur, avec interface graphique simple.

- **Utilisation d'Unix** : Pour un bref rappel/aperçu des notions de base d'Unix/Linux — notamment, les commandes de base, **l'historique des commandes** et **l'édition de commandes** (très utiles pour sauver du temps lorsqu'on travaille en mode ligne de commande) — voir les diapositives suivantes :

  `http://www.labunix.uqam.ca/~tremblay/INF600A/Materiel/bash.pdf`

  Pour d'autres commandes Unix plus avancées (`grep`, `sed`, `find`, `xargs`, etc.) :

  `http://www.labunix.uqam.ca/~tremblay/INF600A/Materiel/commandes-unix.pdf`

- **Utilisation de `git`** : Pour un bref rappel/aperçu des notions de base de `git`, voir le chapitre 23 des notes de cours :

  `http://www.labunix.uqam.ca/~tremblay/INF7235/Materiel/7235.pdf`

- **Utilisation de `make`** : Pour un bref rappel/aperçu des notions de base de `make`, voir les diapositives suivantes :

  `http://www.labunix.uqam.ca/~tremblay/INF600A/Materiel/build.pdf`

1 Parallélisme récursif séquentiel et avec parallélisme *fork/join*

1.1 Obtenir le code à compléter

Obtenez une copie des fichiers pour le labo en exécutant la commande suivante sur `japet` :

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

1.2 Ce que vous devez faire

Pour ce laboratoire, vous devez **compléter** diverses versions d'une méthode, définie dans une extension de la classe `Array`, permettant de trouver le nombre d'**inversions**.

Plus spécifiquement, il s'agit de compter combien **d'éléments adjacents** du tableau *sont dans le mauvais ordre*. Voici quelques exemples :

```
[10, 20].nb_inversions.must_equal 0

[20, 10].nb_inversions.must_equal 1

[10, 10, 30, 30].nb_inversions.must_equal 0

[10, 10, 3, 3].nb_inversions.must_equal 1

[2, 1, 3, 4, 6, 5, 7, 9, 8, 10].nb_inversions.must_equal 3
```

Pour des exemples additionnels, voir le fichier de test `nb_inversions_spec.rb`.

Vous devez définir, puis comparer, trois méthodes :

1. `nb_inversions_rec` : Méthode récursive séquentielle fondée sur une approche diviser-pour-régner dichotomique avec seuil — troncation de la récursion lorsque la taille du problème devient inférieure à un certain seuil.
2. `nb_inversions_rec_pcall` : Méthode récursive parallèle — dichotomique et avec seuil — avec utilisation de `PRuby.pcall`.
3. `nb_inversions_rec_future` : Méthode récursive parallèle — dichotomique et avec seuil — avec utilisation de `PRuby.future`.

Dans les trois cas, vous pouvez réutiliser les méthodes séquentielles déjà définies qui vous sont fournies.

Note : Si vous avez le temps, vous pouvez aussi essayer définir une méthode `nb_inversions_rec3` avec **parallélisme récursif trichotomique** — donc qui décompose un problème en trois (3) sous-problèmes puis qui résout récursivement les sous-problèmes, de façon séquentielle ou parallèle selon la taille du problème et la valeur du seuil.

1.3 Informations additionnelles

- Divers fichiers vous sont fournis :

  - `nb_inversions.rb` : Fichier avec squelette de mise en oeuvre, qui inclut un «répartiteur» (*dispatcher*) — pour appeller la bonne version, selon les paramètres spécifiés par l'intermédiaire des variables d'environnement Unix — de même qu'une version séquentielle non récursive `nb_inversions_seq`.
  - `nb_inversions_spec.rb` : Programme de tests. Ce programme teste **toutes les versions** et ce avec divers seuils. Vous pouvez toutefois désactiver temporairement les tests pour une ou plusieurs versions simplement en mettant la ligne appropriée en commentaire.

    Remarque : Si vous désirez rendre temporairement *inactifs* certains cas de tests bien spécifiques, deux possibilités :

    * Remplacer «`it`» par «`_it`».
    * Ajouter une instruction «`skip "Raison ..."`» à la première ligne du cas de test.
  - `nb_inversions_bm.rb` : Programme de mesures des performances — temps et accélération. Lui aussi s'exécute pour un ensemble de combinaisons de versions et de seuils. Même remarque que pour le fichier précédent.
  - `makefile` : Fichier pour automatiser le lancement des tests et autres tâches :
    * `$ make` : Exécute le programme de tests.
      Par défaut : `DEFAUT=$(TESTS)`.
    * `$ make tests` : Lance le programme de tests.
    * `$ make bm` : Lance le programme de mesures des performances.