TP/TD Unix&Co : Exercices variés, scripts etc.

Vous trouverez tout à l'URL suivante : http://www.math-info.univ-paris5.fr/~lomn/Cours/BC/ ou BI/

unix

http://www.shellunix.com/index.html

Récupérez sur le site *ftp* à l'aide d'une ligne de commande le fichier suivant *splice .tar.gz* que l'on peut trouver à l'URL suivante : ftp.cs.toronto.edu dans le répertoire *pub/neuron/delve.* Bien sûr vous pouvez le faire en mode graphique, mais faisons un peu de *'commandes type unix'* pour cela.

```
   #pour se connecter au serveur
   $ftp ftp.cs.toronto.edu #pour file transfer protocol #ou sftp
   # en général ces serveurs utilisent le USER LOGIN : anonymous et le
#PASSWORD : votre_email pour des raisons de sécurité
   >cd pub/neuron/delve
   >ls
   >cd data
   >ls
   >cd tarfiles
   >ls
   >get splice.tar.gz
   #si pb passer en mode passiv pour que la commande get fonctionne.
   >quit
   $ls
```

Bravo ! Vous commencez à bio-hacker sur un serveur au Canada.

```
   #si la commande mail est installée sinon ne rien faire
   $mail -s ''surprise'' votre_email < splice.tar.gz

   #pour decompresser un fichier
   $gunzip splice.tar.gz
   #pour dérouler l'arborescence
   $tar -xvf splice.tar

   #pour se déplacer
   $cd splice/Source
   $gunzip splice.data.gz

   #pour lire ce que contient le fichier
   $more splice.data
   $more splice.names
   # Que contient ce fichier ? Comment est-il organisé ?
   #usage du pipeline avec la commande grep ('$man grep' pour comprendre)
   $more splice.names | grep EI
   $more splice.names | grep [EI,IE]
   $more splice.data
   #que fait la commande cut ? $man cut par exemple pour répondre
   $cut -c 1 splice.data
      #Extraite la sous-séquence 23-80 de UDR_HUMAN vitamin D dans
      #P11473.fasta ('$man cut' etc)
   $cut -f 2-3 -d, splice.data
   #Décomposer ce pipeline en sous-commandes et utilise man pour parser et
       comprendre cette commande Unix
   $head -20 splice.data | tail -10 | cut -f 3 -d, > splice1.data
   $more splice1.data
```

Dans cet exemple ***head*** envoie les 20 premières lignes du fichier *splice.data* vers la sortie standard; ***tail*** récupère les 10 dernières lignes de la sortie de ***head***; et ***cut*** récupère la troisièmes colonnes et

stocke le résultat dans le fichier *splice1.data*.
Faites de même pour créer 4 fichiers de *splice1.data* à *splice4.data* de sorte que chaque fichier *spliceX.data* contienne les lignes 10*X à 10*(X+1) du fichier originel *splice.data*

```
#wc pour word count
$ls *.data | wc -l
#que fait la commande suivante? Utilité du symbole ? Versus  *
$paste splice?.data > spliceall.data
#que fait la commande suivante?
$cat splice?.data > spliceall.data
#créer un fichier splice5.data et l'ajouter sans écraser à spliceall.data
$cat splice5.data >> spliceall.data
```

grep simple

http://www.tuteurs.ens.fr/unix/exercices/solutions/grep-sol.html

```
   #sur un ensemble de plusieurs fichiers de séquences fasta
$head -1 *.fasta | grep '^>' | sort
   #fichier 1 et 2 sont des noms génériques :
   #remplacer par des fichiers  réels sur votre disque dur, et testez :
$grep -c '>' fichier1.fasta fichier2.fasta

$grep '>' fichier1.fasta fichier2.fasta

$grep -c > fichier1.fasta fichier2.fasta
```

Grep est une commande particulièrement utile pour « parser » et analyser des documents textuels. Notamment l'usage des expressions régulières est très rapide.
Récupérer le texte pg17489.txt
wget -q http://www.gutenberg.org/cache/epub/17489/pg17489.txt;
Puis essayez :
grep -E '([a-zA-Z]){18}' pg17489.txt --color=always

```
grep -E '([a-zA-Z]){18}' pg17489.txt |wc -l;

(Tout ceci peut être pérenisé dans le .bashrc ou par EXPORT color always)

Faire un petit script en shell (bash en l'occurence) pour
```

wget -q liberation.fr ; grep Femme index.html | wc -l

En python, on peut faire appel aux modulex *os* puis *re* en présentant un peu plus les expression régulières

```
#! /usr/bin/python2
import os
import commands

os.mkdir('Test')
cmd="pwd"
ch=commands.getoutput(cmd)
ch=commands.getoutput("head -n 2 SNP.py")
```

Text Mining avec Python

```
import os
import sys
import re

#os.system("wget http://www.gutenberg.org/cache/epub/20/pg20.txt")
#Si nécessaire ou
file = open(sys.argv[1] + ".txt")
#Ouvre le fichier et  l'appelle file. sys.argv[i]permet de mettre un
#argument après le nom du code lors de son éxécution pour permettre le
#même parsing avec différents textes. Ex:la commande "python3
#ParsingPython.py pg20" va ouvrir pg20.txt

for line in file: #parcourt les lignes du fichier pg20
    if re.match('people', line):
#si le mot "people" est présent dans une ligne.
#On utilise les Regular Expressions (re)
        print(line) #affiche la ligne

os.system("more " + sys.argv[1] + ".txt | grep -o -E [a-zA-Z]{15} | wc
-w")
#autre méthode utilisant les expressions régulières, cette fois avec
#grep, permettant d'afficher le nombre d'occurences de mots de 15
#lettres. Dans ce texte on en a 9.
findall = re.findall("kiki", "Le kiki de mon kiki est mon kiki")
#fonction de la librairie re, qui affiche l'occurence d'un mot (ou
#séquence de mots), ici "kiki"; dans une phrase
search = re.search("kiki", "Le kiki de mon kiki est mon kiki")
#fonction légèrement différente de findall dans son output
print(findall)
print(search)
file.close()
```

Vérifiez la loi de Zipf sur le texte téléchargé.

gawk

http://www.shellunix.com/awk.html

Récupérer le tableau de mesures *2D.txt* décrivant un nuage de points 2D. Le nuage n'est pas bien formaté. Il faut le formater différemment. Par exemple, certaines lignes comportent une troisième coordonnée égale à 0. Il faut l'enlever. On va utiliser la commande ***gawk*** ou ***awk*** d'Unix :

- Créez un programme ***reformate.awk*** à l'aide de ***gedit*** ou ***gvim*** ou ***emacs*** qui contient les deux lignes de code suivante :

```
{print $1,$2}
END{print NR}
```

- Appliquez-le au fichier *2D.txt* en lançant :

```
$awk -f reformate.awk 2D.txt
```

- puis

```
$awk  '{print $1,$2}' 2D.txt >2Df.txt
```

latex

Premiers pas avec Latex

Générez des pdf à partir d'un fichier Latex
Récupérer FirstStepLatex.zip et essayez. (pdftex xxx.tex, latex xxx.tex, pdflatex xxx.tex)

Comment créer un document 3D **pour un rapport.**
Avec ***Fiji ou imageJ***, et le Menu ***Plugins/3D Viewer*** et le fichier sample cochlea de *z-stacks (Bat Cochlea Volume)*, visualiser la scène 3D comme une surface rouge. Exportez la scène en tant que fichier de type u3D (universal 3D format) : *cochlea.u3d* . Sauvegardez le texte échantillon au format latex *cochlea.tex* (utilisez éventuellement un éditeur latex pour communication scientifique ou un simple éditeur de texte).
A présent utilisez la commande ***unix*** *$pdflatex* pour générer un fichier *cochlea.pdf* avec une vue 3D intégrée de la scène (commande ***unix*** *$acroread cochlea.pdf*). On aura éventuellement besoin d'une version récente d'acrobat reader pour pouvoir bénéficier des modules de visualisation 3D.

```
$pdflatex cochlea.tex
$acroread cochlea.pdf
```