Introduction à `Unix` et `bash`

Master-I parcours SSD

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

Unix / Linux ?

Wikipedia[1] : **Unix** [...] est une famille de systèmes d'exploitation multitâche et multi-utilisateur [...]. Il repose sur un interpréteur ou superviseur (le **shell**) et de nombreux petits utilitaires, accomplissant chacun une action spécifique, commutables entre eux (mécanisme de « redirection ») et appelés depuis la ligne de commande.

1. https://fr.wikipedia.org/wiki/Unix
2. https://fr.wikipedia.org/wiki/Linux

Unix / Linux ?

Wikipedia[1] : **Unix** [...] est une famille de systèmes d'exploitation multitâche et multi-utilisateur [...]. Il repose sur un interpréteur ou superviseur (le **shell**) et de nombreux petits utilitaires, accomplissant chacun une action spécifique, commutables entre eux (mécanisme de « redirection ») et appelés depuis la ligne de commande.

Wikipedia[2] : **Linux** [...] est une famille de systèmes d'exploitation open source de type Unix fondé sur le **noyau Linux**, créé en 1991 par Linus Torvalds. De nombreuses distributions Linux ont depuis vu le jour et constituent un important vecteur de popularisation du mouvement du logiciel libre.

1. https://fr.wikipedia.org/wiki/Unix
2. https://fr.wikipedia.org/wiki/Linux

Terminal, shell, et bash

Terminal[3] : fenêtre d'invite de commande contenant un shell Unix.

```
mahep@master1 g2pDataset (master) $ ls -l
total 240
-rwxrwx--- 1 mahep technor 398 May 10 07:44 DESCRIPTION
-rwxrwx--- 1 mahep technor 312 Mar 12  2021 g2pDataset.Rproj
drwxrwx--- 4 mahep technor  51 Oct  5 10:46 inst
drwxrwx--- 2 mahep technor 684 May 10 08:52 man
-rw-rw---- 1 mahep technor 538 May 18 08:48 NAMESPACE
drwxrwx--- 2 mahep technor 172 Oct  5 10:46 R
mahep@master1 g2pDataset (master) $
mahep@master1 g2pDataset (master) $
```

Un shell Unix est un interpréteur de commandes destiné aux systèmes d'exploitation Unix [...] qui permet d'accéder aux fonctionnalités internes du système d'exploitation.

bash (pour "Bourne-Again Shell") est le shell le plus répandu (alternatives : `sh`, `csh`, `ksh`, ...).

3. https://fr.wikipedia.org/wiki/Terminal_(informatique)

Unix et bash pour le data-scientist

Pourquoi s'intéresser à `Unix` et à `bash` :

- vecteur principal du logiciel libre !
- système d'exploitation privilégié des serveurs de calcul
- permet d'automatiser certains traitements de fichiers

Ce cours : un guide de survie pour le Data Scientist

Objectif : savoir réaliser quelques opérations base en ligne de commandes via un terminal.

Plan

1. Quelques principes de base
2. Utilitaires de manipulation de fichier
3. Edition de fichier : l'exemple de `vim`
4. Scripts shell

⇒ TP : prise en main de `unix / bash`

I - Quelques principes de base

Principes de base...

Quelques principes de base :

- système de fichiers
- utilisateurs, groupes et permissions
- variables d'environnement
- fichiers de configuration et alias

Système de fichiers

Les fichiers sont organisés dans une arborescence :

- racine : "`/`"
- "`/un_dossier/un_sous_dossier`"

Système de fichiers

Les fichiers sont organisés dans une arborescence :

- racine : "`/`"
- "`/un_dossier/un_sous_dossier`"

Plusieurs répertoires système :

- `/opt`, `/usr`, `/bin`, ...

Système de fichiers

Les fichiers sont organisés dans une arborescence :

- racine : "`/`"
- "`/un_dossier/un_sous_dossier`"

Plusieurs répertoires système :

- `/opt`, `/usr`, `/bin`, ...

Quelques répertoires clés :

- répertoire personnel : `/home/user` (raccourci "~")
- repertoire courant : "`./`"
- répertoire parent : "`../`"

Système de fichiers

Les fichiers sont organisés dans une arborescence :

- racine : "`/`"
- "`/un_dossier/un_sous_dossier`"

Plusieurs répertoires système :

- `/opt`, `/usr`, `/bin`, ...

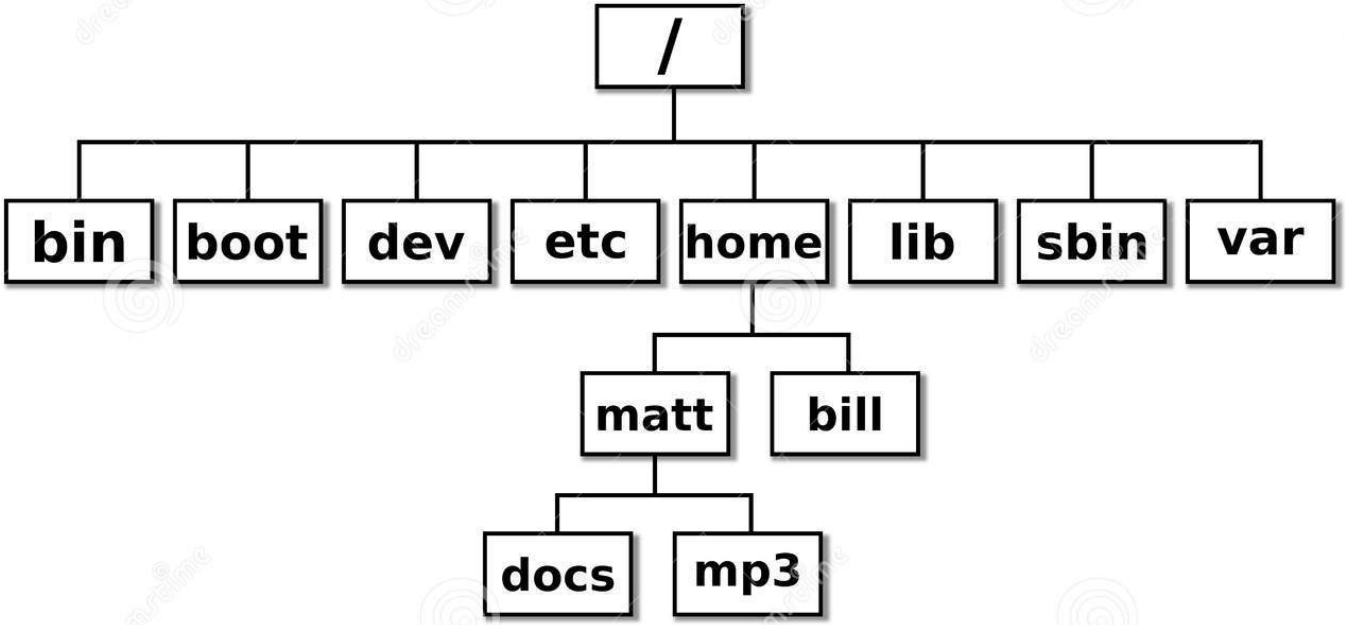
Quelques répertoires clés :

- répertoire personnel : `/home/user` (raccourci "~")
- repertoire courant : "`./`"
- répertoire parent : "`../`"

⇒ savoir où on est : `pwd`

Système de fichiers

Illustration :



Système de fichiers

Commandes de base :

- lister le contenu d'un répertoire : `ls directory`
  - répertoire courant : `ls ./` ou juste `ls`
  - répertoire parent : `ls ..`

Système de fichiers

Commandes de base :

- lister le contenu d'un répertoire : `ls directory`
  - répertoire courant : `ls ./` ou juste `ls`
  - répertoire parent : `ls ..`
- se déplacer dans un répertoire : `cd directory`
  - chemin absolu ou relatif
  - remonter d'un niveau : `cd ..`
  - revenir chez soi : `cd ~` ...ou tout simplement `cd`

Système de fichiers

Commandes de base :

- lister le contenu d'un répertoire : `ls directory`
  - répertoire courant : `ls ./` ou juste `ls`
  - répertoire parent : `ls ..`
- se déplacer dans un répertoire : `cd directory`
  - chemin absolu ou relatif
  - remonter d'un niveau : `cd ..`
  - revenir chez soi : `cd ~` ...ou tout simplement `cd`
- créer un répertoire : `mkdir`

Système de fichiers

Commandes de base :

- copier un fichier : `cp file1 file2`
- supprimer un fichier : `rm file`
- supprimer un répertoire : `rm -R directory`
  - "R" = récursif
- déplacer / renommer un fichier : `mv file1 file2`
- afficher le contenu d'un fichier : `cat file`

Principes de base...

Quelques principes de base :

- système de fichiers
- utilisateurs, groupes et permissions
- variables d'environnement
- fichiers de configuration et alias

Utilisateurs, groupes et permissions

- `Unix` : système d'exploitation multi-utilisateurs
- Les utilisateurs appartiennent à des groupes
- Les fichiers et répertoires ont 3 niveaux de permission :
  - utilisateur / groupe / le reste du monde...
  - ...en lecture / écriture / execution
- Le(s) "super-utilisateur(s)" a(ont) tous les droits
  - compte "root"

Utilisateurs, groupes et permissions

Voir les permissions : `ls -l`

```
mahep@master1 g2pDataset (master) $ ls -l
total 240
-rwxrwx--- 1 mahep technor 398 May 10 07:44 DESCRIPTION
-rwxrwx--- 1 mahep technor 312 Mar 12  2021 g2pDataset.Rproj
drwxrwx--- 4 mahep technor  51 Oct  5 10:46 inst
drwxrwx--- 2 mahep technor 684 May 10 08:52 man
-rw-rw---- 1 mahep technor 538 May 18 08:48 NAMESPACE
drwxrwx--- 2 mahep technor 172 Oct  5 10:46 R
mahep@master1 g2pDataset (master) $
mahep@master1 g2pDataset (master) $
```

Utilisateurs, groupes et permissions

Voir les permissions : `ls -l`

```
mahep@master1 g2pDataset (master) $ ls -l
total 240
-rwxrwx--- 1 mahep technor 398 May 10 07:44 DESCRIPTION
-rwxrwx--- 1 mahep technor 312 Mar 12  2021 g2pDataset.Rproj
drwxrwx--- 4 mahep technor  51 Oct  5 10:46 inst
drwxrwx--- 2 mahep technor 684 May 10 08:52 man
-rw-rw---- 1 mahep technor 538 May 18 08:48 NAMESPACE
drwxrwx--- 2 mahep technor 172 Oct  5 10:46 R
mahep@master1 g2pDataset (master) $
mahep@master1 g2pDataset (master) $
```

Changer les permissions :

- d'un fichier `chmod XXX file`
- de tout un répertoire : `chmod -R XXX directory`

où XXX donne les permissions aux niveaux user/group/all :

- e.g., 7 = `rwx` ; 5 = `r-x` ; 1 = `- -x`

Utilisateurs, groupes et permissions

Voir les permissions : `ls -l`

```
mahep@master1 g2pDataset (master) $ ls -l
total 240
-rwxrwx--- 1 mahep technor 398 May 10 07:44 DESCRIPTION
-rwxrwx--- 1 mahep technor 312 Mar 12  2021 g2pDataset.Rproj
drwxrwx--- 4 mahep technor  51 Oct  5 10:46 inst
drwxrwx--- 2 mahep technor 684 May 10 08:52 man
-rw-rw---- 1 mahep technor 538 May 18 08:48 NAMESPACE
drwxrwx--- 2 mahep technor 172 Oct  5 10:46 R
mahep@master1 g2pDataset (master) $
mahep@master1 g2pDataset (master) $
```

Changer les permissions :

- d'un fichier `chmod XXX file`
- de tout un répertoire : `chmod -R XXX directory`

où XXX donne les permissions aux niveaux user/group/all :

- e.g., 7 = `rwx` ; 5 = `r-x` ; 1 = `- -x`

Changer l'utilisateur / le groupe d'un fichier : `chown` et `chgrp`

Principes de base...

Quelques principes de base :

- système de fichiers
- utilisateurs, groupes et permissions
- variables d'environnement
- fichiers de configuration et alias

Variables d'environnement

Variables d'environnement : variables permettant de paramétrer le fonctionnement du système.

Variables d'environnement

Variables d'environnement : variables permettant de paramétrer le fonctionnement du système.

Quelques variables importantes :

- `SHELL` interpréteur de commande utilisé
- `HOME` chemin du répertoire de l'utilisateur
- `PATH` chemin des exécutables

Variables d'environnement

Variables d'environnement : variables permettant de paramétrer le fonctionnement du système.

Quelques variables importantes :

- `SHELL` interpréteur de commande utilisé
- `HOME` chemin du répertoire de l'utilisateur
- `PATH` chemin des exécutables

⇒ Afficher une variable d'environnement : `echo $HOME`

Variables d'environnement

Variables d'environnement : variables permettant de paramétrer le fonctionnement du système.

Quelques variables importantes :

- `SHELL` interpréteur de commande utilisé
- `HOME` chemin du répertoire de l'utilisateur
- `PATH` chemin des exécutables

⇒ Afficher une variable d'environnement : `echo $HOME`

⇒ Définir une variable d'environnement :
`export PATH=$PATH:/home/mahep/mon_programme`

Variables d'environnement

Variables d'environnement : variables permettant de paramétrer le fonctionnement du système.

Quelques variables importantes :

- `SHELL` interpréteur de commande utilisé
- `HOME` chemin du répertoire de l'utilisateur
- `PATH` chemin des exécutables

⇒ Afficher une variable d'environnement : `echo $HOME`

⇒ Définir une variable d'environnement :
`export PATH=$PATH:/home/mahep/mon_programme`

⇒ De nombreuses variables d'environnement sont définies dans le(s) fichier(s) de configuration !

Variables d'environnement

La commande `env` permet de lister l'ensemble de ses variables d'environnement :

```
mahep@master1 ~ $ env
MANPATH=/vol/vol_jobsched/sge-8.1.9/man:/usr/share/man:/usr/local/share/man
XDG_SESSION_ID=293225
PICARD_HOME=/Softs/bioinfo/picard-tools-1.77
HOSTNAME=master1
_LMFILES__modshare=/Softs/etc/modules/python/3.7.2:1
taxoDB=/Master_Data/RAW_DATA_MIRRORS/EXTERNAL/taxonomy/NCBI/2018-04-19/ncbi-taxo.db
MODULEPATH_modshare=/Softs/etc/modules:1
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=10.250.14.143 64747 22
PERL5LIB=/home/mahep/workspace_SVN/mixgenomix-cluster/trunk/library/perl:/Softs/bioinfo/grinder/0.5.3/blib/lib/:/Softs/System/perl/5.26.1/lib
SGE_CELL=default
SGE_ARCH=lx-amd64
SNPEFF_HOME=/Softs/bioinfo/snpEff/4.0
QTDIR=/usr/lib64/qt-3.3
QTINC=/usr/lib64/qt-3.3/include
SSH_TTY=/dev/pts/4
QT_GRAPHICSSYSTEM_CHECKED=1
USER=mahep
LD_LIBRARY_PATH=/Softs/System/python/3.7.2/lib:/Softs/System/GCC/4.9/rtf/lib64/::/Softs/bioinfo/lib/:/Softs/bioinfo/GDL/lib/:/Softs/System/go
ib:/home/mahep/INSTALL/cppunit-1.12.1/INSTALLED/lib:/Softs/openmpi-1.6.5/lib/:/Softs/bioinfo/lib/
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=
=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z
.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:
=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.
=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:
=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.axa=
ENV=/Softs/System/modules/current//init/profile.sh
MAIL=pierre.mahe@biomerieux.com
PATH=/home/mahep/SVN-MINES/bmx/large-scale-metagenomics-2.0_TEST/tools/ext:/home/mahep/SVN-MINES/bmx/large-scale-metagenomics-2.0_TEST/tools:
arch_Program/B2720-Information_Content_Services/WP2-Genotype_to_Phenotype/Tools/mash/mash-Linux64-v2.0:/home/mahep/workspace_GIT/dbgwas/build
orkspace_SVN/genepi/src:/Softs/bioinfo/bwa/0.7.8:/home/mahep/workspace_SVN/mixgenomix-cluster/trunk/seq:/home/mahep/workspace_SVN/mixgenomix-
runk/svm_taxo:/home/mahep/workspace_SVN/mixgenomix-cluster/trunk/svm_multiclass:/home/mahep/workspace_SVN/mixgenomix-cluster/trunk/liblinear:
ahep/workspace_SVN/mixgenomix-cluster/trunk/scripts/R:/home/mahep/workspace_SVN/mixgenomix-cluster/trunk/scripts/perl:/home/mahep/workspace_S
VN/mixgenomix-cluster/trunk/library/shell:/Softs/System/perl/5.26.1/bin/:/vol/vol_jobsched/sge-8.1.9/bin:/vol/vol_jobsched/sge-8.1.9/bin/lx-a
r/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/ibutils/bin:/opt/dell/srvadmin/bin:/Softs/bin:/Softs/bioinfo/FastQC:/Softs/bioinfo/statge
fo/blast/2.2.28+/bin:/Softs/bioinfo/blast/2.2.17/bin/:/Softs/bioinfo/MUMmer/3.22:/Softs/bioinfo/velvet/1.2.10/contrib/VelvetOptimiser-2.2.4:/
nfo/glimmer3.02/bin:/Softs/bioinfo/interproscan/5.7-48.0/bin:/Softs/bioinfo/ratt:/Softs/bioinfo/abyss/1.1.2/bin:/Softs/bioinfo/infernal-1.1rc
oinfo/mira/3.4.1.1/bin:/Softs/bioinfo/phylip-3.69/exe:/Softs/bioinfo/Proteinortho:/Softs/bioinfo/snpEff/4.0/scripts:/Softs/bioinfo/SOAPdenovo
bin://home/mahep/bin:/home/mahep/INSTALL/gsl-1.16/INSTALLED/bin:/home/mahep/INSTALL/gsl-1.16/INSTALLED/include:/home/mahep/INSTALL/gsl-1.16/I
/bin:/home/mahep/INSTALL/cppunit-1.12.1/INSTALLED/include:/Softs/openmpi-1.6.5/bin:/var/applis/bioinfo/vowpal_wabbit-7.5/vowpalwabbit/:/Softs
PWD=/home/mahep
_LMFILES_=/Softs/etc/modules/python/3.7.2
LANG=en_US.UTF-8
MODULEPATH=/Softs/etc/modules
SGE_ROOT=/vol/vol_jobsched/sge-8.1.9
```

Principes de base...

Quelques principes de base :

- système de fichiers
- utilisateurs, groupes et permissions
- variables d'environnement
- fichiers de configuration et alias

Fichiers de configuration et alias

Deux fichiers de configuration avec le shell `bash` :

- `.bash_profile` : exécuté à l'ouverture d'une session
- `.bashrc` : exécuté à l'ouverture d'un terminal

⇒ fichiers "." = fichiers cachés (`ls -a` pour les afficher)

Fichiers de configuration et alias

Deux fichiers de configuration avec le shell `bash` :

- `.bash_profile` : exécuté à l'ouverture d'une session
- `.bashrc` : exécuté à l'ouverture d'un terminal

⇒ fichiers "." = fichiers cachés (`ls -a` pour les afficher)

Ils permettent de paramétrer l'environnement de travail :

- définir des variables d'environnement
- définir des alias (~ des raccourcis)
- "customiser" son terminal
- ...

⇒ à utiliser sans modération !

Fichiers de configuration et alias

Définir une variable d'environnement :

```
export PATH=$PATH:/home/mahep/mon_programme
```

Fichiers de configuration et alias

Définir une variable d'environnement :

```
export PATH=$PATH:/home/mahep/mon_programme
```

Définir un alias :

```
alias cdpython='cd /home/mahep/projets/python'
alias ll='ls -lh'
```

Fichiers de configuration et alias

Définir une variable d'environnement :

```
export PATH=$PATH:/home/mahep/mon_programme
```

Définir un alias :

```
alias cdpython='cd /home/mahep/projets/python'
alias ll='ls -lh'
```

Customiser son invite de commande :

- par défaut : `user@hostname work_dir $`
  - e.g., `mahep@master ~ $`
- defini par la var. d'environnement `PS1="\u@\h \w $ "`
- ajouter date et heure : `PS1="[\d \t] \u@\h \w $ "`
  - e.g., `[Wed Nov 03 20:54:15] mahep@master ~ $`

⇒ de nombreuses options disponibles (infos, couleurs, ...)

⇒ à définir et exporter dans son `.bash_profile` ou `.bashrc`

II - Utilitaires de manipulation de fichiers

Utilitaires Unix

Unix inclut plusieurs utilitaires de manipulation de fichiers :

- cat
- head / tail
- wc
- grep
- sort
- cut
- sed
- ...

⇒ combinables entre eux par des "redirections" (ou "pipes")

⇒ permettent de réaliser de nombreux traitements sans faire appels à des langages plus évolués (e.g., python).

Utilitaires Unix

`cat` : concaténer des fichiers

- `cat file1 file2` : concatène le contenu des deux fichiers et affiche le contenu à l'écran.
- `cat file1` : affiche le contenu du fichier à l'écran.

Utilitaires Unix

`cat` : concaténer des fichiers

- `cat file1 file2` : concatène le contenu des deux fichiers et affiche le contenu à l'écran.
- `cat file1` : affiche le contenu du fichier à l'écran.

Mécanismes de redirection :

- `cat file1 file2 > file3` : concatène le contenu de `file1` et `file2` et écrit le résultat dans `file3`.
  - écrase le contenu de `file3` s'il existe déja !
- `cat file1 file2 » file3` : concatène le contenu de `file1` et `file2` et écrit le résultat *à la fin de* `file3`.
  - crée `file3` s'il n'existe pas

⇒ valables pour tous les utilitaires à venir !

Utilitaires Unix

`head / tail` : afficher les premières / dernières lignes d'un fichier

- `head -n 10 file` : affiche les 10 premières lignes du fichier à l'écran.
- `tail -n 10 file` : affiche les 10 dernières lignes du fichier à l'écran.

(là aussi : ">" ou "»" pour rediriger la sortie dans un fichier)

Utilitaires Unix

`head / tail` : afficher les premières / dernières lignes d'un fichier

- `head -n 10 file` : affiche les 10 premières lignes du fichier à l'écran.
- `tail -n 10 file` : affiche les 10 dernières lignes du fichier à l'écran.

(là aussi : ">" ou "»" pour rediriger la sortie dans un fichier)

`wc` : compte le nombre de lignes, de mots et d'octets d'un fichier

- `wc file` : affiche les 3 valeurs à l'écran
- `wc file -l / -w / -c` : affiche uniquement le nombre de lignes / de mots / d'octets

Utilitaires Unix

`grep` : rechercher un motif dans un fichier

- `grep pattern file` : affiche à l'écran les lignes du fichier contenant le motif `pattern`
- `grep pattern file -v` : affiche à l'écran les lignes du fichier *ne contenant pas* le motif `pattern`
- `grep pattern file -i` : affiche à l'écran les lignes du fichier contenant le motif `pattern`, sans prendre en compte les majuscules ou minuscules.

Utilitaires Unix

`grep` : rechercher un motif dans un fichier

- `grep pattern file` : affiche à l'écran les lignes du fichier contenant le motif `pattern`
- `grep pattern file -v` : affiche à l'écran les lignes du fichier *ne contenant pas* le motif `pattern`
- `grep pattern file -i` : affiche à l'écran les lignes du fichier contenant le motif `pattern`, sans prendre en compte les majuscules ou minuscules.

`sort` : ordonne les lignes d'un fichier

- `sort file` : trie les lignes par ordre alphabétique croissant et affiche le résultat à l'écran
- `sort file -r` : trie les lignes par ordre alphabétique décroissant et affiche le résultat à l'écran
- `sort -u` : trie les lignes et retourne les valeurs uniques

Utilitaires Unix

`cut` : extrait des colonnes dans un fichier tabulé

- `cut -f 3 file` : extrait du fichier la 3ème des colonnes séparées par des tabulations
- `cut -f 3 -d ' ' file` : extrait du fichier la 3ème des colonnes séparées par des espaces
- `cut -f 2-5 file` : extrait les colonnes 2 à 5
- `cut -f 2-5,8 file` : extrait les colonnes 2 à 5 et 8
- `cut -f 2-5 -complement file` : supprime les colonnes 2 à 5

Utilitaires Unix

`sed` : rechercher / remplacer le contenu d'un fichier

- `sed 's/old/new/g' file` : remplace les occurences de `old` par `new` dans `file` et affiche le résultat à l'écran
  - sans le `g` : ne remplace que la 1ère occurence par ligne
- `sed -i 's/old/new/g'` : remplace les occurences de `old` par `new` dans le fichier (et l'écrase donc)
- `sed -i.save 's/old/new/g'` : remplace les occurences de `old` par `new` crée une sauvergarde du fichier original (dans `file.save`)

⇒ de nombreuses options pour contrôler les lignes sur lesquelles faire ces substitutions

⇒ on peut également utiliser des expressions régulières

Utilitaires Unix et redirections

Rappel : ">" ou "»" pour rediriger la sortie dans un fichier

Utilitaires Unix et redirections

Rappel : "$>$" ou "$\gg$" pour rediriger la sortie dans un fichier

Autre mécanismes de redirection : les tuyaux (ou "pipes")

- `head -n 10 file1 | sort > file2` : extrait les 10 premières lignes de `file1`, les trie et écrit le résultat dans `file2`
- `head -n 10 file1 | cut -f 7 | sort > file2` : extrait le 7ème champ des 10 premières lignes de `file1`, les trie et écrit le résultat dans `file2`
- `cut -f 7 file1 | head -n 10 | sort > file2` : fait la même chose !

$\Rightarrow$ permet de réaliser des traitements relativement complexes en combinant les utilitaires de base : à suivre en TP !

III - Edition de fichiers : l'exemple de vim

L'éditeur de texte `vim`

Plusieurs programmes pour éditer un fichier via le terminal :

- `vi / vim` (`vim` = `vi` improved)
- `emacs`
- `nano`
- ...

Aucun n'est meilleur que les autres, chacun utilise son favori...

Ici : une brève introduction à `vi / vim`

vim - le B.A.-BA (1/2)

Ouvrir l'éditeur : `vim` ou `vim my_file`

- `vim` ouvre l'éditeur "vide"
- `vim my_file` ouvre un fichier (nouveau ou existant)

vim - le B.A.-BA (1/2)

Ouvrir l'éditeur : `vim` ou `vim my_file`

- `vim` ouvre l'éditeur "vide"
- `vim my_file` ouvre un fichier (nouveau ou existant)

Editer le fichier :

1. entrer en mode "insertion" en tapant sur la touche `i`
2. taper son texte
3. quitter le mode "insertion" en tapant sur `echap`

`vim` - le B.A.-BA (2/2)

Sauvegarder le fichier :

1. entrer en mode "commande" en tapant sur la touche `:`
2. taper sur la touche `w` (write) pour dire de sauvegarder
3. appliquer la commande en tapant sur `entrée`

vim - le B.A.-BA (2/2)

Sauvegarder le fichier :

1. entrer en mode "commande" en tapant sur la touche :
2. taper sur la touche `w` (write) pour dire de sauvegarder
3. appliquer la commande en tapant sur `entrée`

Fermer l'éditeur :

1. entrer en mode "commande" en tapant sur la touche :
2. taper sur la touche `q` (quit) pour dire de quitter
3. appliquer la commande en tapant sur `entrée`

⇒ pour sauvegarder et fermer l'éditeur : taper `:wq` (+ entrée)

vim - quelques commandes utiles

Effacer une ligne :

1. se placer sur la ligne à effacer
2. taper sur `dd`

vim - quelques commandes utiles

Effacer une ligne :

1. se placer sur la ligne à effacer
2. taper sur `dd`

Annuler une commande : taper sur `u` (pour "undo")

- e.g. suppression de ligne, modification du texte, ...
- appuyer plusieurs fois sur `u` pour annuler plusieurs commandes successives
- appuyer sur `CTRL+r` pour refaire une commande annulée

vim - quelques commandes utiles

Effacer une ligne :

1. se placer sur la ligne à effacer
2. taper sur `dd`

Annuler une commande : taper sur `u` (pour "undo")

- e.g. suppression de ligne, modification du texte, ...
- appuyer plusieurs fois sur `u` pour annuler plusieurs commandes successives
- appuyer sur `CTRL+r` pour refaire une commande annulée

Atteindre une ligne donnée :

1. entrer en mode "commande" en tapant sur la touche `:`
2. taper le numéro de la ligne à atteindre + `entrée`

vim - quelques commandes utiles

Rechercher une chaîne de caractères :

1. taper sur la touche /
2. taper la chaîne de caractères (peut contenir des espaces)

⇒ taper sur n ou N pour atteindre l'occurence suivante ou précédente

vim - quelques commandes utiles

Rechercher une chaîne de caractères :

1. taper sur la touche /
2. taper la chaîne de caractères (peut contenir des espaces)

⇒ taper sur n ou N pour atteindre l'occurence suivante ou précédente

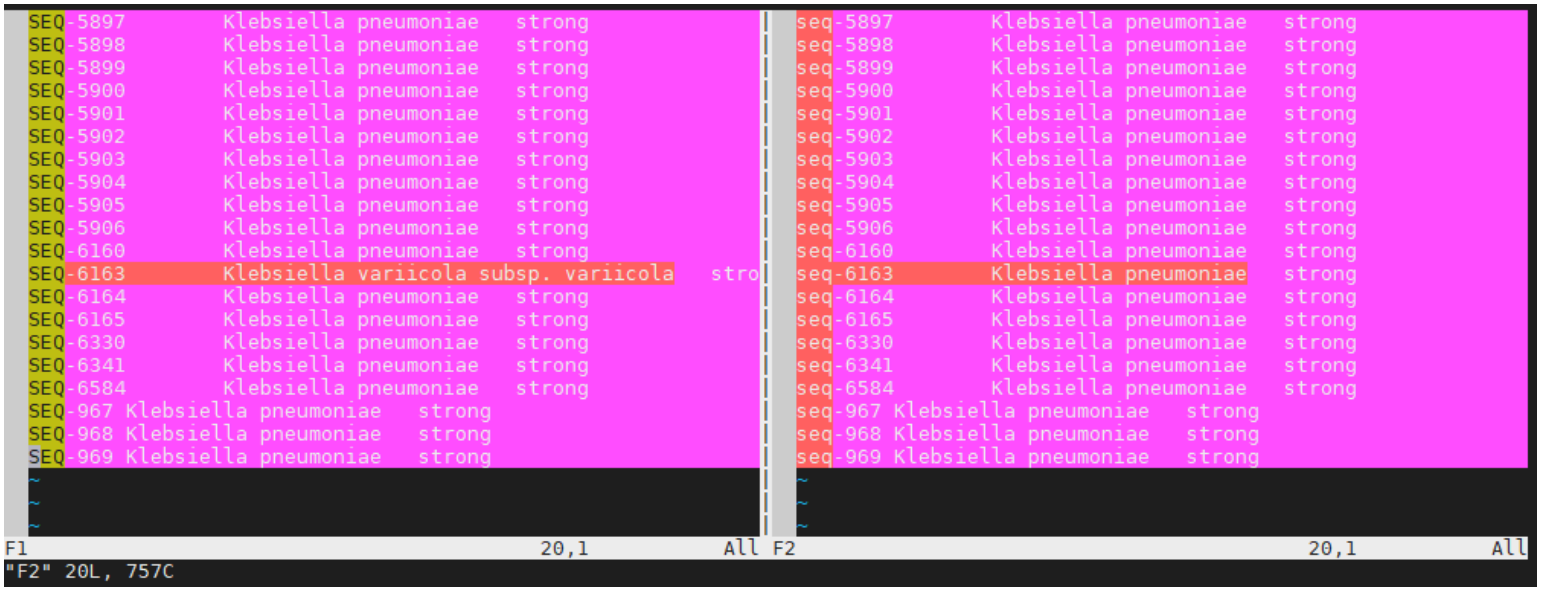
Rechercher et remplacer une chaîne de caractéres :

1. entrer en mode "commande" en tapant sur la touche :
2. sur la ligne courante : `s/old/new/g`
3. dans tout le fichier : `%s/old/new/g`

⇒ demander confirmation avant : `(%)s/old/new/gc`
⇒ ignorer minuscules / majuscules : `(%)s/old/new/gi`

vim - quelques commandes utiles

Comparer deux fichiers : `vim -d file1 file2`



IV - Scripts shell

Scripts shell

Script shell : un fichier texte contenant plusieurs commandes qui sont exécutées séquentiellement par le shell.

- ~ un programme qui n'est pas compilé

Scripts shell

Script shell : un fichier texte contenant plusieurs commandes qui sont exécutées séquentiellement par le shell.

- ~ un programme qui n'est pas compilé

Intérêt :

- permet d'automatiser une série d'opérations
- accès "direct" aux utilitaires `unix`
- relativement simple à mettre en oeuvre
- (tourne sans `python`, `perl` ou autres)

⇒ ce cours : quelques rudiments de scripts shell.

Scripts shell

Structure type d'un script shell (`mon-script.sh`) :

- spécifier le chemin vers l'interpréteur :
  - `#!/usr/bin/bash`
- récupérer les éventuels arguments du script
  - `inputFile=$1`
- réaliser les commandes
  - `echo "fichier d'entrée :" $inputFile`

Scripts shell

Structure type d'un script shell (`mon-script.sh`) :

- spécifier le chemin vers l'interpréteur :
  - `#!/usr/bin/bash`
- récupérer les éventuels arguments du script
  - `inputFile=$1`
- réaliser les commandes
  - `echo "fichier d'entrée :" $inputFile`

Exécuter le script : deux (voire trois) possibilités

1. `$ bash mon-script.sh`
2. `$ ./mon-script.sh` si le script a été rendu exécutable
   - via la commande `$ chmod +x mon-script.sh`
3. `$ mon-script.sh` si il est exécutable et dans le `$PATH`

Commandes utiles

Affecter le résultat d'une commande à une variable :

```
nLines=‘cat $inputFile | wc -l‘
echo "number of lines =" $nLines
```

Commandes utiles

Affecter le résultat d'une commande à une variable :

```
nLines=‘cat $inputFile | wc -l‘
echo "number of lines =" $nLines
```

Tester une condition :

```
if [[ $var == "" ]]
then
    echo "string is empty"
fi
```

- négation : `if [[ $var != "" ]]`
- existence d'un fichier : `if [[ -f "$inputFile" ]]`

Commandes utiles

⇒ Premier exemple de script (`example-1.sh`) :

```
#!/usr/bin/bash

inputFile=$1

echo "processing input file :" $inputFile

if [[ -f "$inputFile" ]]
then
    nLines=‘cat $inputFile | wc -l‘
    echo "--> number of lines =" $nLines
else
    echo "--> file doesn’t exist !"
fi
```

- à exécuter avec un nom de fichier en argument
- on peut ajouter des conditions dans le `else` avec `elif`

Commandes utiles

Boucles for :

```
for variable in valeurs
do
    instructions
done
```

Commandes utiles

Boucles for :

```
for variable in valeurs
do
    instructions
done
```

⇒ par exemple :

- `for var in 1 2 3 4 5 6 7 8 9`
- `for var in toto tata tutu`
- `for (( i=$min; i<=$max; i++ ))`
- `for var in *.txt`
- ...

Commandes utiles

⇒ Deuxième exemple de script (`example-2.sh`) :

```
#!/usr/bin/bash

start=12
end=18

for (( i=$start; i<=$end; i=i++ ))
do
    echo "current index is" $i
    if [[ $((i%2)) -eq 0 ]]
    then
        echo -e "\tindex is even"
    else
        echo -e "\tindex is odd"
    fi
done
```

Commandes utiles

Parcourir un fichier ligne à ligne :

```
while read line
do
    echo $line
done < $inputFile
```

- lit ligne à ligne le fichier `inputFile`
- stocke le contenu de la ligne dans la variable `line`
- (ces noms sont arbitraires - `"while read toto"` aurait marché tout aussi bien !)

Commandes utiles

⇒ Dernier exemple de script (example-3.sh) :

```
#!/usr/bin/bash

inputFile=ard.csv
echo "processing input file :" $inputFile

i=0
while read line
do
    i=$((i+1))
    gene=‘echo $line | cut -f 2 -d ’;’‘
    echo -e "reading line no" $i ": gene =" $gene
done < $inputFile
```

- à exécuter dans le répertoire tp-unix pour avoir accès au fichier ard.csv

Conclusion

Conclusion

Ce cours : une **introduction** à `unix` et `bash` !

De nombreuses fonctionalités pour aller plus loin :

- d'autres utilitaires (e.g., `awk`)
- fonctionalités pour paramétrer son environnement
- des opérations plus complexes avec `vim`
  - et d'autres éditeurs de texte !
- les expressions régulières
- le langage `shell`
  - `https://doc.ubuntu-fr.org/tutoriel/script_shell`
- ...

⇒ se référer aux pages d'aide en cas de problème : `man ls`

⇒ de nombreuses ressources disponibles sur internet !