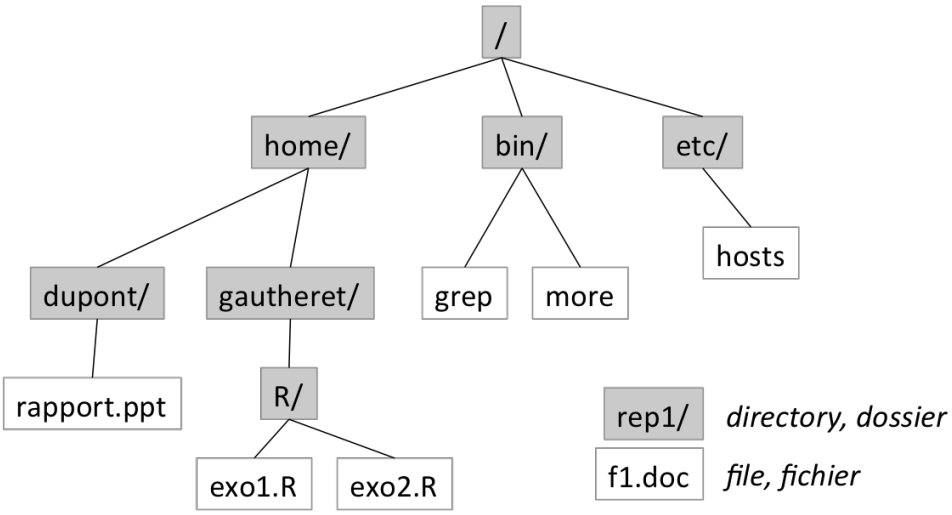
Mini Manuel Unix

Daniel Gautheret

2023.2

1. Dossiers et fichiers sous UNIX



Un exemple d'arborescence Unix

Chemin absolu (commence par /):

```
/home/gautheret/R/exo1.R
/bin
```

Chemin relatif (pour un utilisateur dans `gautheret`):

"." désigne le dossier courant, ".." le dossier de niveau supérieur

```
R/exo1.R
./R/exo1.R
../dupont/rapport.ppt
../../etc/hosts
```

Voir le contenu d'un dossier, avec la commande `ls` :

```
ls R            (tous les fichiers du dossier R)
ls ../dupont
ls /etc
ls -l /etc      (détaillé)
ls -lt /etc     (classé par date)
ls *.doc        (fichiers se terminant par .doc)
ls f*.R         (fichiers commençant par f, terminant par .R)
ls R/f*R        (idem, dans le dossier R)
```

Se déplacer dans les dossiers, avec la commande `cd` :

```
cd ..
cd /etc
cd ../dupont    (ça marche?)
cd              (cd seul: revient à votre dossier 'home')
```

Voir le contenu d'un fichier texte, avec les commande more ou cat:

```
more /etc/hosts
cat /etc/hosts
```

Copier un fichier avec la commande cp:

```
cp /etc/hosts hosts2.txt
```
-> copie /etc/hosts dans le dossier courant sous le nom "hosts2.txt"
```
cp /etc/hosts .
```
-> copie /etc/hosts dans le dossier courant, sans changer le nom
```
cp exo1.R exo3.R
```
-> copie exo1.R dans le dossier courant sous le nom "exo2.R"
```
cp /partage/gautheret/ASG/*.pdf .
```
-> copie tous les fichiers .pdf du dossier ASG vers le dossier courant ('.')

Déplacer ou renommer un fichier avec la commande mv:

```
mv exo2.R exo2b.R  (renomme)
mv exo2.R Dossier/  (déplace vers le rep "Dossier")
mv exo2.R ../exo2.R  (déplace vers le rep supérieur)
mv ../exo2.R exo2.R (déplace depuis le rep supérieur)
```

Créer un dossier avec les commandes mkdir:

```
mkdir ASG
```

Effacer un fichier ou un dossier avec les commandes rm, rmdir:

```
rm exo2b.R
rm *.R
rm *       (attention!)
rm ASG     (ne marche pas si ASG est un dossier)
rmdir ASG  (efface dossier ASG, si vide)
```

Informations cruciales sur les fichiers et dossiers Unix:

- Les majuscules et minuscules sont discriminantes
  - F1.doc ≠ f1.doc
- Les espaces sont autorisés mais sont très génants et à éviter. Exemple :

```
more mes vacances.doc
mes: Aucun fichier ou dossier de ce type
vacances.doc: Aucun fichier ou dossier de ce type
```

- Les accents sont autorisés mais ne sont pas gérés par de nombreux programmes. A éviter.
- Les extensions (.doc, .txt, .fasta, .gz, etc.) sont totalement libres et ne définissent pas le contenu du fichier. Rien n'empêche d'ajouter .doc, .ppt, .fasta à n'importe quel fichier. C'est à vous de faire attention.

Les Propriétés des fichiers

Résultat de la commande ls -l:

```
drwxr-xr-x   2 root     root         4096 Oct  9 16:44 docs
drwxr-xr-x   2 root     root         4096 Oct  9 16:44 vms
drwxr-xr-x   6 root     root         4096 Oct  9 16:44 widgets-mesa
drwxr-xr-x   2 root     root         4096 Oct  9 16:44 widgets-sgi
-rwxr-xr-x   1 2001     2001       139227 Jul 20 17:25 configure
-rw-r--r--   1 2001     2001        12685 Jul 20 17:25 Makefile.in
-rw-r--r--   1 2001     2001        18838 Jul 20 17:25 aclocal.m4
-rw-r--r--   1 2001     2001        13920 Jul 20 17:24 configure.in
-rw-r--r--   1 2001     2001        25326 Jul 18 15:36 Makefile.X11
-rwxr-xr-x   1 2001     2001         5598 Jul 18 02:19 install-sh
-rwxr-xr-x   1 2001     2001         6283 Jul 18 02:19 missing
-rwxr-xr-x   1 2001     2001          722 Jul 18 02:19 mkinstalldirs
-rw-r--r--   1 2001     2001            0 Jul 18 02:19 stamp-h.in
-rw-r--r--   1 2001     2001         1060 Jul 18 02:19 conf.h.in
```

2. Les Principales commandes UNIX

Nom	Description	Options	Arguments
Gestion fichiers et répertoires			
cd	se positionne sur le répertoire désigné		chemin d'accès à un répertoire
ls	liste le contenu d'un répertoire	-**a** : prise en compte des fichiers cachés -**F** : renseigne sur le type de fichier (*, /, @) -**i** : précision du numéro d'inode des fichiers -**R** : liste récursivement le contenu du répertoire -**l** : informations détaillées -**g** : ajout du nom du groupe -**d** : renseigne sur le répertoire lui-même -**t** : liste par date de modification -**u** : liste par date d'accès -**r** : ordre inverse	nom de répertoire
pwd	retourne la référence absolue du répertoire courant		
cp	copie du fichier source vers la destination	-**i** : demande confirmation -**p** : conservation de date et droits -**r** : recopie récursive d'un répertoire	source - destination
mv	déplace fichiers et répertoires	-**i** : demande confirmation	source - destination

rm	supprime la référence du fichier dans le répertoire	-**f** : force la commande sans s'occuper des droits -**i** : demande confirmation -**r** : destruction récursive	nom de fichier ou de répertoire
mkdir	création d'un répertoire		nom de répertoire
rmdir	suppression d'un répertoire vide		nom de répertoire
cat, **zcat**	imprime le contenu du fichier mentionné (zcat: pour fichier compressé)		nom de fichier
echo	permet d'écrire à l'écran ou dans un fichier		'texte' > nom de fichier
more	liste le contenu d'un fichier page par page		nom de fichier
Divers			
man	retourne le mode d'emploi de la commande s'il existe		nom de commande
gzip	compresse un fichier (voir compress) par défaut, la destination est la sortie standard qu'on redirige vers un fichier si on veut obtenir un fichier compressé	-**c** : résultat sans modification du fichier source -**f** : écrasement d'un fichier compressé préexistant -**v** : taux de compression -**d** : force une décompression -**9** : niveau maximum de compression	nom de fichier
gunzip	décompresse un fichier	-**c** : le résultat est produit sans modification du fichier d'origine -**v** : taux de compression	nom de fichier
du	« disk usage » : espace utilisé par chaque répertoire sous le répertoire courant	**-s** : ne montre que l'espace total occupé par un répertoire et ses sous répertoire (faire : –s *)	
ps	liste des processus BSD	-**a** : processus détenus par les autres utilisateurs -**u** : nom du propriétaire du processus -**x** : processus sans terminal associé -**l** : description complète -**t** : terminal particulier	
kill	envoi d'un signal à un processus	-**l** : liste des signaux disponibles -**n** : numéro du signal à envoyer **2** : INT (ctrl-**c** :) **3** : QUIT (ctrl-\) **9** : KILL	numéro du process
sort	tri des lignes du fichier suivant l'ordre ASCII	-**r** : tri inverse -**u** : élimine les lignes identiques	fichier

tar	archivage/désarchivage de fichiers	**c** : création de l'archive **x** : restauration de l'archive **t** : listage du contenu de l'archive **v** : mode verbeux **p** : préserve dates et droits à la restauration **f** : le nom du fichier spécifié sera celui de l'archive	archive - répertoire ou fichiers
ssh	établir une connexion avec une machine distante(sortie : exit)		nom / IP de machine
wget	Télécharge le contenu de pages http ou ftp via Internet (équivalent MacOS : curl)	`wget [url]` **-O** `[fichier sortie]`	URL de la page à télécharger
which	retourne le chemin complet d'accès à une commande		nom de la commande
who (am i)	retourne le nom des utilisateurs qui ont ouvert une session		
chmod	change les permissions en lecture, écriture, exécution	-**R** change les droits récursivement à partir du noeud (mode = ugo)	mode - nom de fichier ou de répertoire

3. Jouer avec les entrée/sorties des taches

La redirection des sorties ">"

Envoie le résultat d'une commande dans un fichier (au lieu de la sortie standard : l'écran)

```
ls > fichier
blast seq1.fasta seq2.fasta > blastout
cat titi > toto
cat titi >> toto (>> n'écrase pas le fichier de sortie)
```

La redirection des entrées "<"

```
Fasta < inputfile
```

Lit les entrées dans un fichier (au lieu de l'entrée standard : le clavier)
(inputfile contient toutes les réponses aux questions posées par le programme.)

Le pipeline "|"

Utilise le résultat d'une commande comme argument d'une autre commande

```
ps -au | more
zcat RRECB345.fastq.gz | more
```

Le contrôle des tâches

"&" permet de récupérer le shell après le lancement d'un programme

```
firefox &
gedit &
nano &
```

Sans "&", le shell reste bloqué tant que le programme tourne. On peut toujours suspendre

l'exécution d'un programme avec CTRL-Z, puis le redémarrer en arrière-plan avec "`bg`", ou en premier plan avec "`fg`".
La commande "`jobs`" permet de voir la liste des taches en cours d'exécution (taches nous appartenant)
La commande "`kill [no-de-tache]`" tue une tache en cours.
La commande "`top`" permet de visualiser toutes les taches en cours d'exécution, y compris celles des autres utilisateurs.

4. La commande WGET (CURL sous MacOS)

Une commande magique pour aller directement interroger les API (Interfaces programmatiques) de nombreux services en ligne. L'EBI (ENSEMBL) et le NCBI offrent de tels services. L'équivalent MacOS est `curl` (les paramètres de curl et wget diffèrent un peu).

Exemples :

Sur le site du NCBI :
(ici CP000962 est à remplacer par un identifiant refseq NCBI)

Au format fasta :

```
wget
'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&
id=CP000962&rettype=fasta&retmode=text' -O myseq
```

(attention -O majuscule avec wget!)

```
curl
'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&
id=CP000962&rettype=fasta&retmode=text' -o myseq
```

(attention -o minuscule avec curl!)

Au format Genbank:

```
wget
'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&
id=CP000962&rettype=gb&retmode=text' -O myseq2
```

Sur le site de l'EBI/ENSEMBL/UNIPROT :

```
wget
'https://rest.ensembl.org/sequence/region/human/X:1000000..1000100:-
1?content-type=text/x-fasta' -O myseq3

wget
'https://rest.uniprot.org/uniprotkb/stream?compressed=true&format=fast
a&query=%28organism_id%3A%20XXX%29' -O myseq4.gz
```

où **XXX** est le numéro de taxon, ou TAXID.
Notez que le fichier est récupéré en format .gz (donc binaire) avec l'option compressed=true.

On peut visualiser les fichiers obtenus avec la commande « more » (après décompression le cas échéant)

5. La commande GREP

"grep" est un *filtre*, c'est à dire un programme qui selectionne automatiquement les lignes d'un fichier possédant telle ou telle propriété. Le filtre "grep" permet de sélectionner les lignes contenant une certaine expression régulière, c'est à dire un motif flexible décrivant un ensemble de chaines de caractères.

Par exemple, l'expression "...di" décrit les chaines "lundi" et "mardi" (et toute autre chaine de 3 caractères se terminant par "di"). L'expression "[Pp]hosphorylase" décrit les chaines "Phosphorylase"et "phosphorylase".

La commande "grep" a la forme:

```
grep [expression regulière] [fichier]
```

(si l'on veut afficher toutes les lignes répondant à l'expression), ou bien :

```
grep -c [expression regulière] [fichier]
```

Exemple :

```
grep "^le" fichier
grep "^l[ea]" fichier
```

(si l'on veut juste compter toutes les occurences).
L'option -i permet d'ignorer les majuscules/minuscules. Exemple :

```
grep -i "^l[ea]" fichier
```

La combinaison `.*` est très utile pour spécifier un nombre inconnu de caractères quelconques :

```
grep -i "gaule.*rome" fichier
```

L'option -E permet d'utiliser des expressions régulières étendues contenant des caractères `\w`, `\D` etc et d'éviter d'avoir à entrer des caractères d'échappement devant certains caractères spéciaux (`\|` pour `|`) :

```
grep -E "[expression régulière]" fichier
```

exemples :

```
grep -E -i "^(adn)|(arn)" fichier
grep -iE "^l(e|a)" fichier
```

Caractères spéciaux dans `grep` et `grep -E`:

.	Tout caractère (sauf newline)
^	Le début d'une ligne
$	La fin d'une ligne
|	Choix. A|B: A ou B
()	Groupement de caractères
[]	Classe de caracteres. [AGUC]: A,G,U ou C
Les modificateurs suivants sont à placer après le caractère concerné	
*	0 fois ou plus
+	une fois ou plus

?	une fois ou zero
{n}	exactement n fois
{n,}	au moins n fois
{n,m}	de n a m fois
grep -E	Voir ci-dessous

Exemples tirés de https://twitter.com/s0md3v/status/1171394406512910336:

Expression	Reconnaît:
`cat`	`cat`
`ca+t`	`caaaaaaaaaaaat but not ct`
`ca*t`	`caaaaaaaaaaaat and also ct`
`ca{2,4}`	`caat, caaat and caaaat`
`c(at)+`	`catatatatatat`
`c(at|orn)`	`cat and corn`
`c[ea]t`	`cat and cet`
`c[ea]+t`	`caaaat and ceeet`
`c[A-C0-9]t`	`cAt, cBt, cCt, c8t etc.`
`c.t`	`cat, c&t, c2t (any char between c and t)`
`c.+t`	`c3%x4t (any number of any chars)`
`c.*t`	`c3%x4t and as well as ct`
`^`	`denotes start of a string`
`$`	`denotes end of a string`
`^a+cat`	`aaacat in aaacat but not in bbaaacat`
`cat$`	`cat in aaacat but not in aaacats`
`^cat$`	`only and only this string i.e. cat`
`\d`	`Digits`
`\D`	`Non digits`
`\w`	`alphanumeric chars`
`\W`	`Non alphanumeric chars`
`\s`	`white space chars & line breaks`
`\S`	`Non white space chars`
`\t`	`Tabulations`
`\r`	`Carriage return`
`\n`	`Newline`
`c\d+t`	`c2784t`
`c\s+`	`c  t`
`c\D+`	`cxxxt ca2t`

Voir aussi cet excellent guide sur les expressions régulières: https://amitness.com/regex/

6. La commande AWK

Un programme AWK est une suite de couples de la forme:
`'/motif/ {action}'`

Awk est utilisé principalement pour rechercher des lignes contenant un certain motif de caractères et effectuer une action sur ces lignes (par ex. afficher une certaine partie du texte).
Le motif est une expression régulière. Les variables $1, $2, etc. font références aux colonnes 1,2, etc. dans le fichier.

Exemples:
```
awk '{print $2}' toto
```
(pas de motif spécifié, affiche toutes les colonnes 2 du fichier toto)

```
awk '/JOURNAL/ {print $2}' mgen.gbk
```
(affiche la colonne 2 du fichier mgen.gbk si la ligne contient JOURNAL)

On peut spécifier le caractère qui sépare les colonnes (FS), par exemple ici le caractère « : »
```
awk '{FS = ":" ; print $2 }' myfile
```

7. Le Shell et le fichier .bashrc

Le shell ou terminal est l'environnement qui interprète les commandes Unix. Il existe plusieurs types de shell, notamment : csh, tcsh et bash (bash est souvent le shell par défaut sous Linux et MacOS).

Selon le shell utilisé, on dispose de plus ou moins de facilité pour entrer des commandes. Par exemple, tcsh et bash permettent de revenir aux commandes précédentes avec les flèche-haut ou les touches CTRL-R + une chaine de caractères à rechercher.

Les variables d'environnement

Le fonctionnement du shell est contrôlé par des variables d'environnement, qui stockent des informations comme :

- l'endroit où se trouvent les programmes exécutables (variable $PATH)
- Votre login ($USER), votre dossier racine ($HOME)…
- l'endroit où se trouvent certaines bases de données,
- etc.

On peut afficher ces variables avec la commande echo :
```
echo $HOME
echo $PATH
```

Le contenu de ces variables est défini dans un fichier qui est lu automatiquement lors de l'ouverture d'un nouveau shell/terminal. Pour le shell bash, il s'agit du fichier `.bashrc`, qui peut contenir par exemple:
```
export PATH=.:/usr/X11R6/bin:~/Bin/:$PATH
export PLPLOT_LIB=/usr/local/share/EMBOSS
```

(Lorsqu'on définit des variables, on n'utilise pas le signe `$` avant leur nom)

Si l'on modifie le fichier `.bashrc`, la modification sera prise en compte lors de l'ouverture du prochain terminal, ou en entrant la commande :
```
source .bashrc
```

Attention : le fichier `.bashrc` se trouve dans votre dossier racine. Si vous n'êtes pas dans ce dossier il faut le préciser :
```
source [chemin-racine]/.bashrc
```

8. Les scripts Shell

Pour réaliser automatiquement une suite de commandes Unix, on peut entrer ces commandes dans un script shell. Les scripts shell sont des fichiers texte. Pour exécuter un script shell `test.sh` on peut entrer:

```
bash test.sh
```
ou
```
source test.sh
```

On peut aussi demander au shell de traiter directement le script comme s'il s'agissait d'un programme exécutable. Pour cela il faut ajouter une ligne d'en-tête dans le script qui signale au système quel interpréteur doit être utilisé pour l'exécution. Cette ligne est :

```
#!/bin/bash
```

Puis il faut modifier le mode du fichier script pour qu'il devienne exécutable:
```
chmod +x test.sh
```

Le script peut maintenant être exécuté directement ainsi:
```
./test.sh
```

Pour ne pas avoir à taper `./` il faut indiquer dans le fichier `.bashrc` que le dossier courant fait partie des dossiers où se trouvent les fichiers exécutables. Ainsi il faut ajouter à la fin du fichier `.bashrc` :

```
export PATH=$PATH:.
```

Attention : le fichier `.bashrc` se trouve dans votre dossier racine. La modification sera prise en compte lors de l'ouverture du prochain terminal, ou en entrant la commande :
```
source [chemin-racine]/.bashrc
```

Le script peut maintenant être exécuté directement ainsi:
```
test.sh
```

9. Variables, boucles et tests dans un script shell

Variables et opérations

Les variables sont déclarées simplement avec le signe =, sans espace autour du signe =, ainsi :

```
myvar=”Hello”
```
ou
```
i=5
```

ATTENTION : Les noms de variables n'ont pas d'espaces, pas d'accent, et les majuscules/minuscules sont significatives. Lorsque la variable est utilisée, elle doit toujours être précédée du signe `$` (mais pas lors de sa déclaration).

Les chaînes de caractères peuvent être écrites sans guillements ou entre guillements simples ou doubles. Les guillemets sont nécessaires pour lever les ambiguités (par ex. en cas d'espace dans la chaine). Lorsque le shell rencontre des simples guillemets, il conserve littéralement tout ce qui est inclus, alors qu'avec les double guillemets il interpréte ce qui est inclus. Par exemple :

Code	résultat
`name=coucou` `echo '$name'`	$name
`name=coucou` `echo "$name"`	coucou

Pour concaténer des chaînes, on peut procéder de différentes façons :

Code	résultat
`name=test` `echo $name'.sh'`	test.sh
`name=test` `echo "$name".sh`	test.sh

Pour utiliser le résultat d'une commande comme une variable, mettre la commande dans $():

Code	résultat
`echo nous sommes le date` `echo nous sommes le $(date)`	`nous sommes le date` `nous sommes le Lun 1 mar 2021 16:55:15`

Pour extraire la racine d'un nom de fichier (partie avant l'extension), on utilise la commande `basename`, ainsi :

Code	Résultat

`monfichier="ecoli.fasta"`
`racinefichier=$(basename $monfichier ".fasta")`
`echo $racinefichier`
	ecoli

Pour traiter des listes de fichiers, on utilise souvent des listes de chaînes de caractères. Par exemple, on crée ici une liste de trois noms de génomes :
`my_genomes=(E_coli B_subtilis S_aureus)`

On peut aussi générer une liste automatiquement à partir de noms de fichiers:
`my_genomes=*.fasta`

Pour acccéder à un élément d'une liste, on utilise la syntaxe suivante (le premier indice de la liste est 0) :
`echo ${my_genomes[0]}`

Les `{}` servent à éviter que le shell interprète d'abord `$my_genome` et ensuite applique l'indice `[0]` sur le résultat, ce qui donnerait avec la liste ci-dessus `E_coli[0]` et donc une erreur.

Pour accéder à toute la liste, on utilise l'indice `[@]` :
`echo ${my_genomes[@]}`

Enfin il existe des variables prédéfinies, comme les variables d'environnement ($HOME, etc. voir plus haut) et les variables :

`$1 $2 …` : les arguments utilisés lors de l'appel du script. Par exemple si l'appel est `myscript.sh coli.fasta`, la variable $1 contient `'coli.fasta'`

`$@` : la liste de tous les arguments utilisés lors de l'appel du script

`$#` : le nombre d'arguments utilisés lors de l'appel du script

Boucles

Pour parcourir une liste, on utilise une boucle `for`. Par exemple :

```
my_genomes=(E_coli B_subtilis S_aureus)
for i in ${my_genomes[@]}
do
  echo $i'.fasta'
done

# autre exemple:
ANA='analysis'
SAMPLES=(normal tumor)
for N in ${SAMPLES[@]}
do
  cp $N'.fastq' $ANA/.
done
```

Une boucle parcourant tous les fichiers ".fasta" d'un dossier "genomes":

```
for fichier in genomes/*.fasta
do
     echo $fichier
done
```

Tests logiques (if)

Comparaisons de variables numériques (symboles : `-lt`, `-eq` , `-gt`, `-ne`, …)

```
if [ $# -lt 1 ]
then
  echo "$0 : il faut au moins un argument"
  exit 1
fi
```

Comparaisons de chaines de caractères (symboles: `==`, `!=`, …)

```
if [ $x1 != $x2 ]
then
     blastp -query $x1 -db $x2
fi
```

10. Trucs utiles sous WSL

Ouvrir un explorateur Windows dans le dossier courant depuis le terminal:

```
explorer.exe .
```

Voir et copier des fichiers Windows depuis le terminal WSL

```
ls /mnt/c/Users/timeo/Documents/
ls /mnt/c/Users/timeo/Download/
mv /mnt/c/Users/timeo/Download/toto.fa toto.fa
```

11. Trucs utiles sous macOS

Ouvrir le finder dans le dossier courant depuis le terminal:

```
open .
```

Voir et déplacer des fichiers téléchargés depuis le terminal :

```
ls ~/Downloads/
mv ~/Downloads/toto.txt toto.txt
```

Ou, dans Safari afficher le menu déroulant des téléchargements, puis faire glisser le fichier voulu vers le terminal.