Utilisation des systèmes UNIX

Benjamin Negrevergne

PSL University – Paris Dauphine – Équipes *MILES*

Outline

Mode texte / mode graphique

L'utilisation d'un ordinateur sous Unix se fait

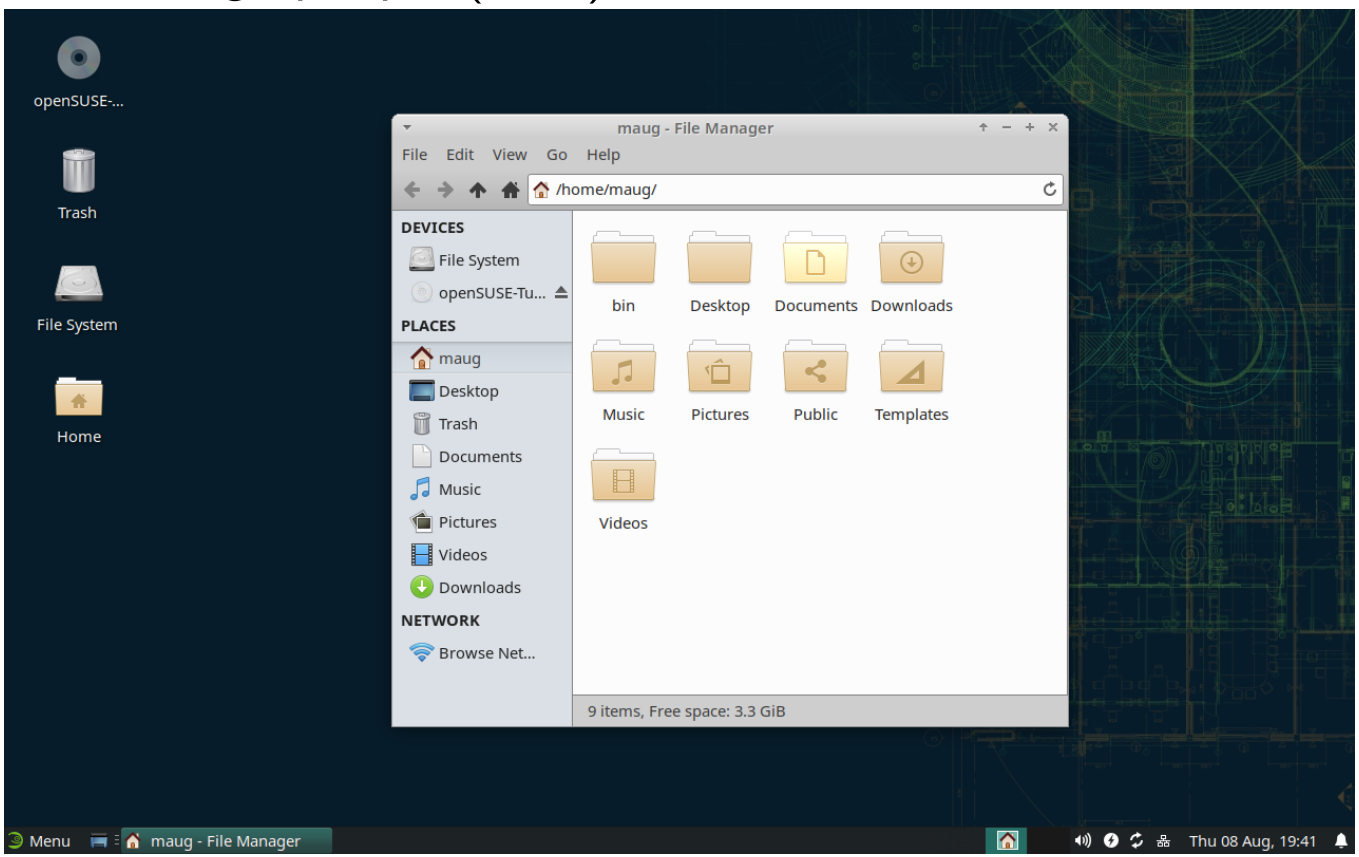
- soit en mode texte (commandes)
- soit en mode graphique (interface graphique).

Mode texte / mode graphique

L'utilisation d'un ordinateur sous Unix se fait

- soit en mode texte (commandes)
- soit en mode graphique (interface graphique).

Interface graphique (Xfce):

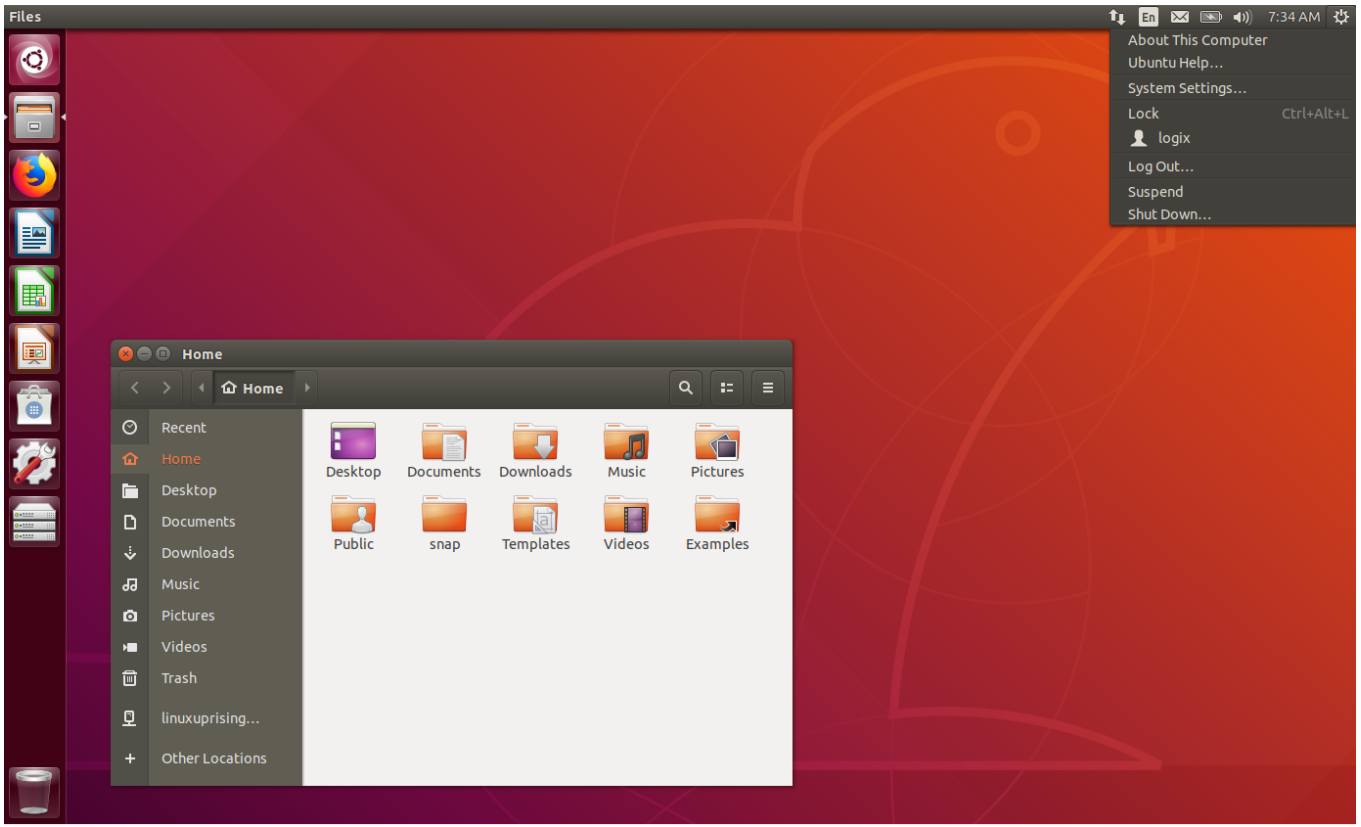


Mode texte / mode graphique

L'utilisation d'un ordinateur sous Unix se fait

- soit en mode texte (commandes)
- soit en mode graphique (interface graphique).

Interface graphique (Unity):

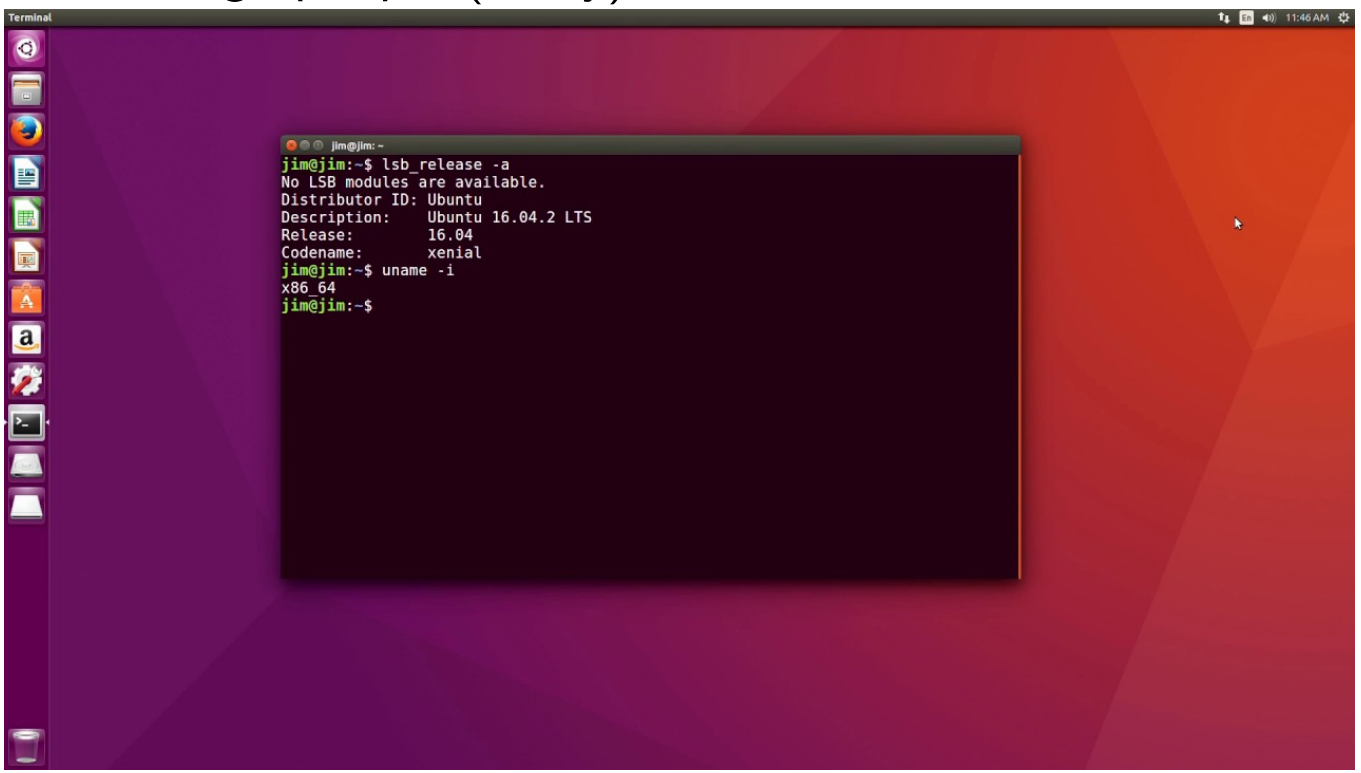


Mode texte / mode graphique

L'utilisation d'un ordinateur sous Unix se fait

- soit en mode texte (commandes)
- soit en mode graphique (interface graphique).

Interface graphique (Unity):



Le plus souvent : interface graphique + terminal

Session Unix

Toute session Unix (graphique ou texte):

- commence par une connexion (*login*)
  - ▶ permet d'authentifier l'utilisateur
- termine par une déconnexion (*logout*)

Au cours de la session

- Les fichiers crées
- Les programmes exécutés

appartiennent à l'utilisateur (et à un groupe)

Cela permet:

- D'assurer une distribution équitable des ressources
  Quotas d'espace disque ou de temps CPU
- De contrôler l'accès aux données de l'utilisateur
  Via un système de permissions sur les fichiers

Quelques commandes de base

Connaître son nom d'utilisateur

```
$ whoami
```

Connaître la liste des utilisateurs connectés

```
$ who
```

Obtenir des informations sur un ficher (e.g. son propriétaire)

```
$ ls -l test.c
-rw-r--r-- 1 bnegreve bnegreve 253 May 18 2020 test.c
```

Terminer une session

```
$ exit
```

Repertoire home

- Chaque utilisateur dispose d'un répertoire *home* dans lequel il se trouve au début de la session. (souvent `/home/utilisateur`)

- Le repertoire contient les fichiers de l'utilisateur

- Les utilisateurs sont libres d'organiser leur répertoire *home* comme ils le souhaitent

- Il est possible d'accéder aux fichier, via l'interface graphique, ou via des commandes tapées dans le terminal.
  ▶ Il s'agit de la même hiérarchie de fichers!

Parcourir son répertoire home

Lister le contenu d'un répertoire

```
$ ls # rien si le repertoire est vide
```

Créer un sous-répertoire

```
$ mkdir unix # mkdir = make directory
$ ls
unix
```

Se déplacer dans un sous répertoire et en revenir

```
$ pwd # pwd = print working directory, affiche le repertoire courant
/home/bnegreve/
$ cd unix # cd = change directory, se déplacer dans un sous-repertoire
$ pwd
/home/bnegreve/unix
$ cd .. # revenir dans le répertoire parent
$ pwd
/home/bnegreve/
```

Manipuler des fichiers

mkdir/rmdir/rm/mv/cp

```
$ mkdir dir # creer un repertoire dir
$ rmdir dir # supprimer le repertoire dir (vide)
$ rm file # supprimer le fichier file
$ rm -r dir # supprimer le rep dir repertoire et tout son contenu
$ mv file file2 # renomer file en file2
$ mv file dir/ # déplacer file dans dir
$ cp file file2 # copier le fichier file dans file2
```

Répertoire parent / repertoire courant

Le répertoire '.' fait toujours référence au répertoire courant

```
1 $ pwd
2 /home/bnegreve/unix
3 $ cd .
4 $ pwd
5 /home/bnegreve/unix
6 $ # on a pas bougé!
```

Le répertoire '..' fait toujours référence au repertoire parent

```
1 $ pwd
2 /home/bnegreve/unix
3 $ cd ..
4 $ pwd
5 /home/bnegreve
6 $ # on a est remonté d'un cran!
```

Racine

Racine

```
$ pwd
/home/bnegreve
$ cd ..
$ pwd
/home
$ cd ..
$ pwd
/
$ cd .. # ???
/
```

Hiérarchie classique UNIX

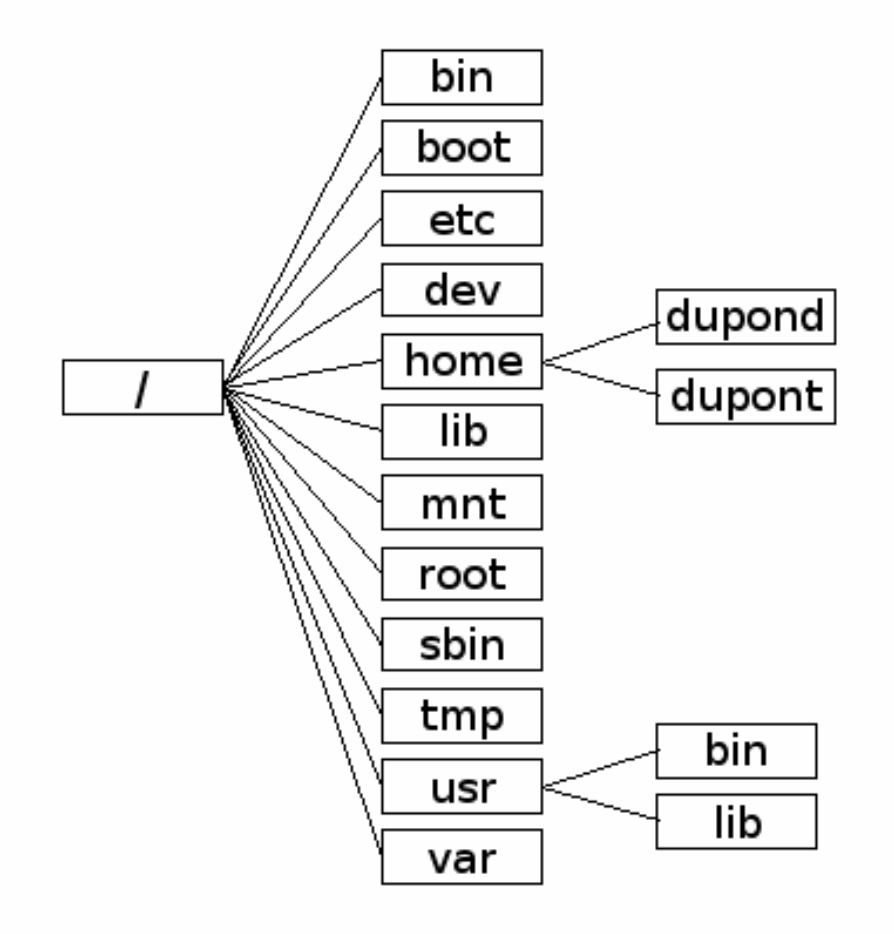


image: coursunix.files.wordpress.com

/bin: exécutables nécessaires

/lib: librairies nécessaires

/usr/bin: exécutables destinés aux utilisateurs

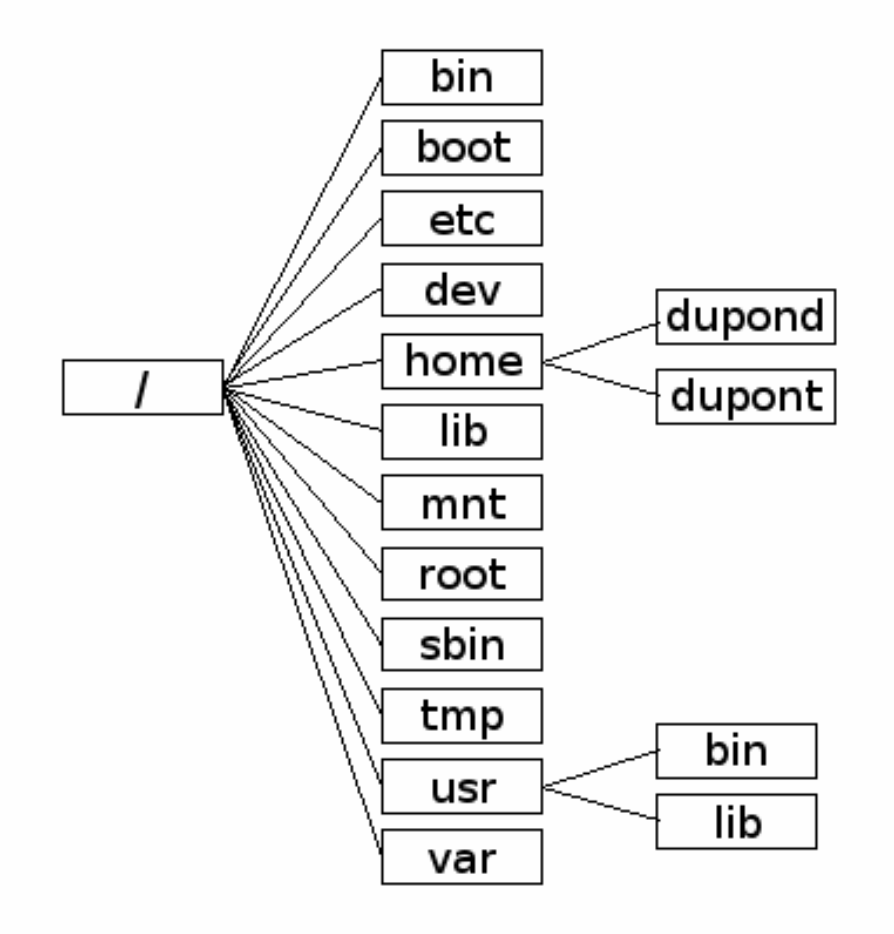
/etc: fichiers de configuration

/tmp: repertoire pour les fichiers temporaires (supprimé au reboot)

/home: contient les repertoires des utilisateurs

/home/dupont: repertoire de l'utilisateur dupont.

Hiérarchie classique UNIX



/bin: exécutables nécessaires

/lib: librairies nécessaires

/usr/bin: exécutables destinés aux utilisateurs

/etc: fichiers de configuration

/tmp: repertoire pour les fichiers temporaires (supprimé au reboot)

/home: contient les repertoires des utilisateurs

/home/dupont: repertoire de l'utilisateur dupont.

image: coursunix.files.wordpress.com

- Les logiciels sont installés automatiquement via des paquets (e.g. apt)

Hiérarchie abstraite & supports physiques

Le *montage* de supports physiques permet d'associer un noeud de la hiérarchie avec un support physique.

```
$ df -Tvh
Filesystem             Type     Size  Used Avail Use% Mounted on
/dev/mapper/neb--vg-root ext4    28G   18G  8.5G  68% /
/dev/sda2              ext2     237M  162M   63M  73% /boot
/dev/mapper/neb--vg-home ext4   425G  219G  185G  55% /home
...
```

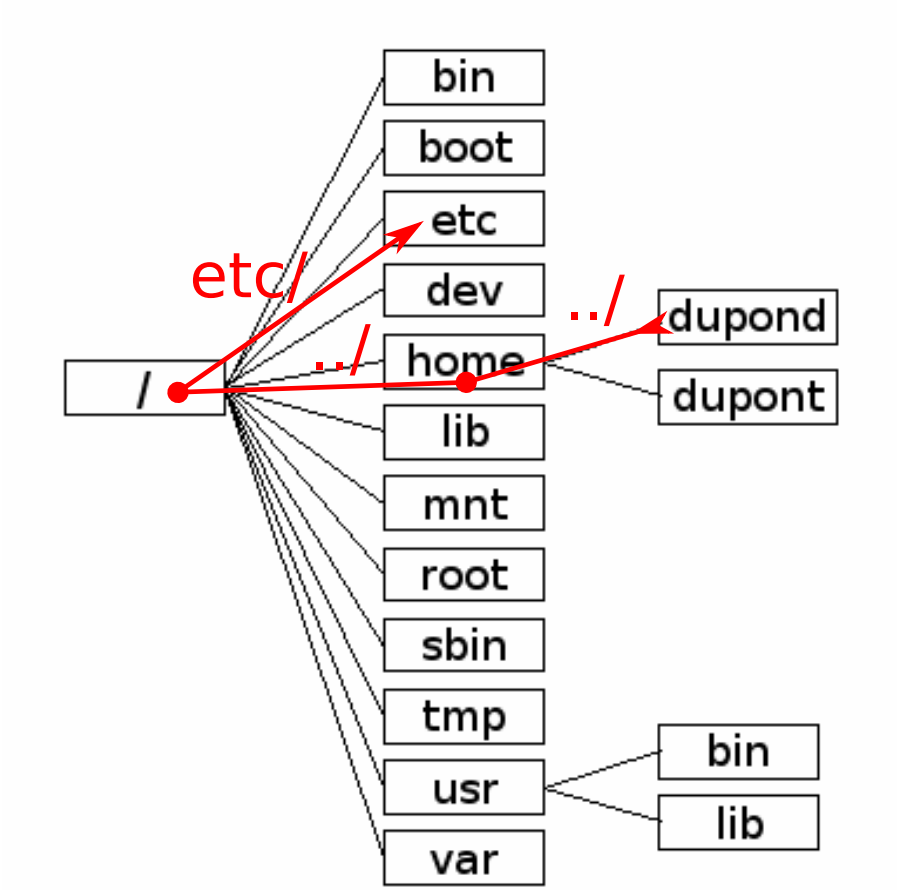
Hiérarchie abstraite & supports physiques

Le *montage* de supports physiques permet d'associer un noeud de la hiérarchie avec un support physique.

```
$ df -Tvh
Filesystem             Type     Size  Used Avail Use% Mounted on
/dev/mapper/neb--vg-root ext4    28G   18G  8.5G  68% /
/dev/sda2              ext2     237M  162M   63M  73% /boot
/dev/mapper/neb--vg-home ext4   425G  219G  185G  55% /home
...
```

- Le disque `/dev/mapper/neb--vg-root` est monté en tant que partition `root`
- Le disque `/dev/sda2` est monté en tant que partition `boot`
- Le disque `/dev/mapper/neb--vg-home` est monté en tant que partition `/home`

Chemin relatif, chemin absolu



'../../etc' : Chemin relatif

(au répertoire courant)

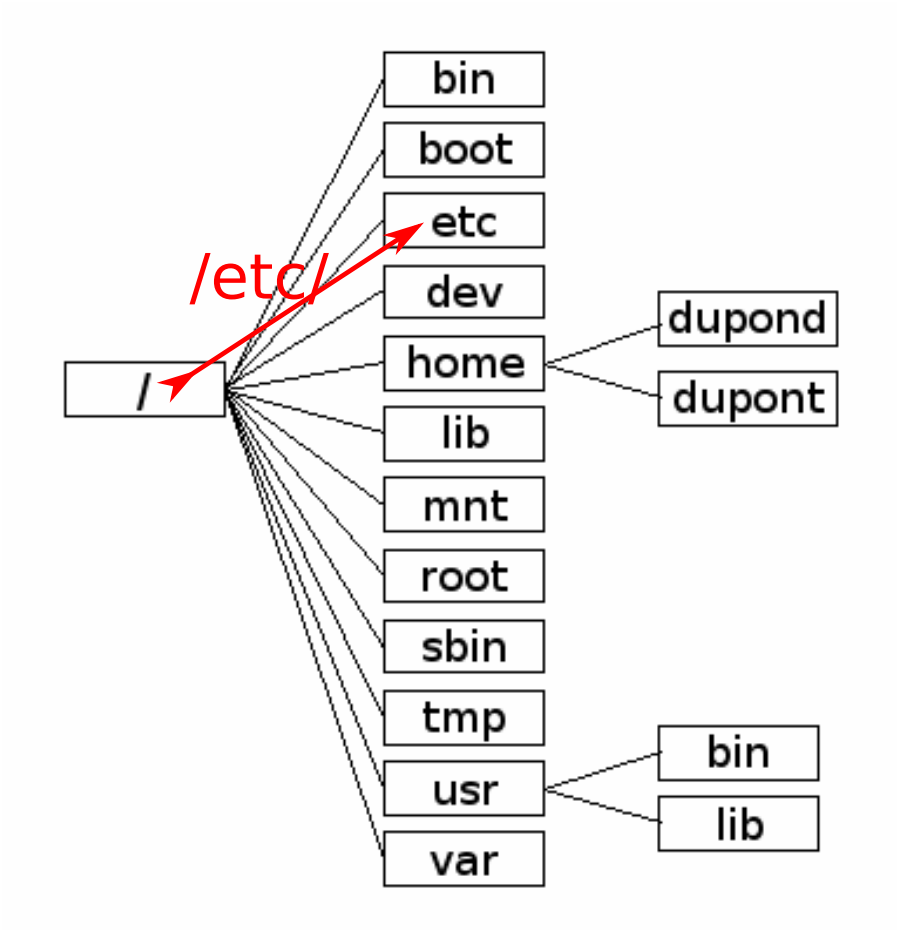
```
1 $ pwd
2 /home/dupond
3 $ cd ../../etc
4 $pwd
5 /etc
```

'/etc': Chemin absolu

(relatif à la racine)

```
1 $ pwd
2 /home/dupond
3 $ /etc
4 $ pwd
5 /etc
```

Chemin relatif, chemin absolu



'../../etc' : Chemin relatif

(au répertoire courant)

```
1 $ pwd
2 /home/dupond
3 $ cd ../../etc
4 $pwd
5 /etc
```

'/etc': Chemin absolu

(relatif à la racine)

```
1 $ pwd
2 /home/dupond
3 $ /etc
4 $ pwd
5 /etc
```

Chemin relatif, chemin absolu (2)

Chemin relatif (par rapport au repertoire courant)

```
$ pwd
/home/bnegreve
$ ls test.c
$ ls ../bnegreve/test.c
$ ls ../stamby/unix/solution_projet.c
```

Chemin absolu (par rapport à la racine '/')

```
/home/bnegreve
$ ls /home/bnegreve/test.c
$ ls /home/stamby/unix/solution_projet.c
```

```
$ pwd
/home/bnegreve
cd ../../etc/../home/bnegreve/cours_unix/tds/../
$ pwd ????
$ cd # ramène au repertoire home
$ pwd
/home/bnegreve
```

Fichiers

Un fichier

- existe dans le système de fichiers
- peut être ouvert, lu, écrit et refermé

Fichiers

Un fichier

- existe dans le système de fichiers
- peut être ouvert, lu, écrit et refermé

Types de fichiers

- Fichiers classiques (regular files).
  - Stockés sur le disque
  - Support d'information persistant ($\neq$ *RAM*)
  - Destiné aux utilisateurs, ou aux applications
  - Exemple: fichier jpeg (image), fichier texte, fichier doc etc
  - Moyen d'échange entre les applications.

Fichiers

Un fichier

- existe dans le système de fichiers
- peut être ouvert, lu, écrit et refermé

Types de fichiers

- Fichiers classiques (regular files).
  - Stockés sur le disque
  - Support d'information persistant ($\neq$ *RAM*)
  - Destiné aux utilisateurs, ou aux applications
  - Exemple: fichier jpeg (image), fichier texte, fichier doc etc
  - Moyen d'échange entre les applications.

- Répertoires
  - Fichier contenant une liste de fichiers
  - Tous les répertoires ont un repertoire parent, noté `..`
  - L'ensemble des fichiers forme un arbre
  - Racine de l'arbre est notée `/`

Fichiers

Un fichier

- existe dans le système de fichiers
- peut être ouvert, lu, écrit et refermé

Types de fichiers

- Fichiers classiques (regular files).
  - Stockés sur le disque
  - Support d'information persistant ($\neq$ *RAM*)
  - Destiné aux utilisateurs, ou aux applications
  - Exemple: fichier jpeg (image), fichier texte, fichier doc etc
  - Moyen d'échange entre les applications.

- Répertoires
  - Fichier contenant une liste de fichiers
  - Tous les répertoires ont un repertoire parent, noté `..`
  - L'ensemble des fichiers forme un arbre
  - Racine de l'arbre est notée `/`

- Liens physiques/symboliques
  - Fichier qui référence un autre fichier

Fichiers

Un fichier

- existe dans le système de fichiers
- peut être ouvert, lu, écrit et refermé

Types de fichiers

- Fichiers classiques (regular files).
  - Stockés sur le disque
  - Support d'information persistant ($\neq$ *RAM*)
  - Destiné aux utilisateurs, ou aux applications
  - Exemple: fichier jpeg (image), fichier texte, fichier doc etc
  - Moyen d'échange entre les applications.

- Répertoires
  - Fichier contenant une liste de fichiers
  - Tous les répertoires ont un repertoire parent, noté `..`
  - L'ensemble des fichiers forme un arbre
  - Racine de l'arbre est notée `/`

- Liens physiques/symboliques
  - Fichier qui référence un autre fichier

Fichiers spéciaux

Les fichiers UNIX: interface généralisées pour faire des E/S.

- Fichiers périphériques "/dev/"
  - Fichier sans existence physiques sur le disque
  - Permet de communiquer avec un périphérique de manière unifiée (via un logiciel pilote)

```
1  cat /dev/input/mouse1 # coordonnées de la souris
2  echo "salut" > /dev/printer # imprime "salut" sur l'imprimante par
       defaut
```

- Fichiers virtuels "/proc" "/sys"
  - Fichier sans existence physiques sur le disque
  - Leur contenu reflète le contenu de la mémoire d'un composant du noyau
  - Permet de piloter ou d'obtenir des données depuis un composant du noyau

```
1  cat /proc/cpuinfo # recupère les infos sur les CPUs
2  cat /proc/mem # recupère les infos sur l'état de la RAM
```

Nom des fichiers

Règles strictes:

- taille entre 1 et 255 caractères

Recommandation

- Utilisez lettres, chiffres, - (tiret) ou _ (souligné)
- Éviter les caractères spéciaux pour le shell:
  `<,>,|,$,?,&,[,],*,!,',",‘,(,),@,~,<espace>`
- En général, le . (point) est utilisé pour séparer le nom de l'extension

```
fichier.c # source en c
fichier.jpeg # image
fichier.sh # script shell
fichier.c.old # ..
fichier # souvent un fichier texte
```

- le . en début de ligne rend le fichier caché

```
$ echo "salut" > test
$ ls
test
echo salut > .test
$ ls
$ ls -a
.test
```

Caractères spéciaux

Pour désigner plusieurs fichiers à la fois, on peut utiliser des caractères spéciaux

Exemples:

- le caractère '*' désigne n'importe quel nom de fichier
- `*.pdf` permet de designer n'importe quel fichier qui termine par '.pdf'

```
$ ls # tout
a.jpeg   b.pdf   c.jpeg    d.jpeg
$ ls * # pareil
a.jpeg   b.pdf   c.jpeg    d.jpeg
$ ls *.pdf # que les noms de fichiers qui terminent par .pdf
b.pdf
$ ls *.jpeg # que les noms de fichiers qui terminent par .jpeg
a.jpeg   c.jpeg   d.jpeg
$ ls *d* # que les noms de fichiers qui contiennent 'd'
b.pdf   d.jpeg
$ ls b* # que les noms de fichiers qui commencent par 'b'
b.pdf
$ ls *a*j* # que les noms de fichiers qui contiennent un 'a' puis un 'j'
a.jpeg
```

Autres caractères spéciaux

- '?' désigne un caractère quelconque

```
$ ls
file.c   file.h    file.jpeg
$ ls file.? # tous les fichiers avec un seul caractère après le .
file.c   file.h
```

- '[...]' n'importe quel caractères parmi l'ensemble entre les crochets

```
$ ls file.[ch]
file.c file.h
$ ls *.[ch] # tous les fichiers .c ou .h
...
$ ls rapport_201[0-3].pdf # tous les rapport de 2010 a 2013
rapport_2010.pdf  rapport_2011.pdf  rapport_2012.pdf  rapport_2013.pdf
```

Attributs des fichiers

Les fichiers sont associés a des persmissions

```
1 $ ls -l test.sh
2 -rwxr-xr-x 1 bnegrevergne lamsade 31 Mar 22 2019 test.sh
```

- `-rwxr-xr-x` droits d'accès au fichiers
- `1` nombres de liens pointant vers ce fichier
- `bnegrevergne` propriétaire du fichier (owner)
- `lamsade` groupe propriétaire du fichier (group)
- `31 Mar 22 2019` date de création
- `test.sh` nom du fichier

Permissions

Droits possibles:

- Read (r): permission de lire le fichier
- Write (w): permission d'écrire ou modifier le fichier
- Execute (x): permission d'exécuter le fichier

Les droits sont définis pour 3 catégories d'utilisateurs

- Le propriétaire du fichier
- Les membres du groupes propriétaire du fichier
- Les autres

`-rwxr-xr--` droits d'accès au fichiers à séparer en trois parties: `rwx`, `r-x` et `r--`

- `rwx` : droits du propriétaire: lecture, écriture, exécution
- `r-x` : droits du groupe: lecture, exécution
- `r--` : droits des autres: lecture seulement

Manipuler le mode d'un fichier (chmod)

`r` (read) = 4, `w` (write) = 2, `x` (execute) = 1

- Mode = 7: Read + Write + Execute
- Mode = 5: Read + Execute
- Mode = 4: Read
- Mode = 1: Execute

```
1  bnegreve@neb:/tmp$ chmod 777 file
2  bnegreve@neb:/tmp$ ls -la file
3  -rwxrwxrwx 1 bnegreve bnegreve 0 Feb 10 14:19 file
4  bnegreve@neb:/tmp$ chmod 722 file
5  bnegreve@neb:/tmp$ ls -la file
6  -rwx-w--w- 1 bnegreve bnegreve 0 Feb 10 14:19 file
7  bnegreve@neb:/tmp$ chmod 744 file
8  bnegreve@neb:/tmp$ ls -la file
9  -rwxr--r-- 1 bnegreve bnegreve 0 Feb 10 14:19 file
10 bnegreve@neb:/tmp$ chmod 755 file
11 bnegreve@neb:/tmp$ ls -la file
12 -rwxr-xr-x 1 bnegreve bnegreve 0 Feb 10 14:19 file
```

Manipuler le mode d'un fichier (2)

```
1  $ chmod g+x file # ajoute la permission x aux utilisateur du groupe
2  $ chmod ug+x file # ajoute la permission x au propriétaire du fichier, et aux
       membres du groupe propriétaire.
3  $ chmod o-rwx # supprime toutes les permission pour les utilisateurs qui n'
       appartiennent pas au groupe propriétaire.
4  $ chmod o-rwx # supprime toutes les permission pour les utilisateurs qui n'
       appartiennent pas au groupe propriétaire.
5  $ chmod a-x # supprime la permission 'x' àtout le monde.
6  $ chmod og-x # supprime la permission 'x' tout le monde, sauf au propriétaire du
        fichier
```

Changer le propriétaire d'un fichier

Changer le propriétaire d'un fichier

```
1 chown stamby test.sh
```

Changer le groupe propriétaire d'un fichier

```
1 chgrp ceremade test.sh
```

find/locate

- `locate` : trouve un fichier à partir d'un *pattern* grâce à un index

```
1 $ locate "*.jpeg" # tous les fichiers .jpeg
2 ...
```

  - rapide
  - updatedb pour mettre l'index à jour

- `find` : cherche un fichier a partir d'un *pattern* sans index
  - plus lent, mais toujours à jour

```
1 $ find . -name "*.jpeg" # tous les fichiers .jpeg dans le répertoire courant
2 ...
```

grep

Print lines that match patterns

man grep

```
1 $ grep 1862 file # toutes les lignes qui contiennent le 'mot' 1862
2 $ ls | grep .jpeg # tous les fichier dont le nom se termine par jpeg
3 $ grep poulet lesmiserables.txt # toutes les lignes qui contiennent le mot poulet
4 $ grep -o '18[0-9][0-9]' lesmiserables.txt | sort | head -n 1 # ??
```

Utile pour

- Extraire toutes les dates d'un fichier
- Extraire toutes les adresses emails
- Extraire les mots qui commencent par a et qui terminent par z
- ...

Expression rationnelles

Langage des motifs (patterns)

- ab (operateur implicite): concaténation de a et de b
- [ab]: a ou b
- a*: 0 ou plus répétitions de a
- ^a : motif qui commence par a
- a$: motif qui termine par a

```
1 $ grep ab # tous les mots qui contiennent un a directement suivit d'un b
2 $ grep a*b # tous les mots qui contiennent zero, un ou plusieurs a suivi d'un b
3 $ grep a[a-z]*z # tous les mots qui commencent par a et qui finissent par z
4 $ grep a[a-z]*k[a-z]*z # tous les mots qui commencent par a et qui finissent par z
      et qui contienennt un k
5 $ grep [12][1-9][1-9][1-9] # toutes les dates entre 1000 et 2999
```

sed

sed - stream editor for filtering and transforming text

man sed

18xx $\rightarrow$ 19xx

```
1 $ cat lesmiserables.txt | sed 's/18\([0-9][0-9]\)/19\1/g' > lesmiserables2000.txt
2 $ cat lesmiserables2000.txt | grep '19[0-9][0-9]' # enjoy!
```