L'arborescence du système de fichiers d'UNIX

- *Créer à l'aide de la commande « mkdir » un répertoire de nom « rep » dans votre répertoire de travail. Placez-vous dans ce nouveau répertoire (« cd ») et examiner le contenu de ce dernier, en utilisant successivement les commandes « ls » puis « ls -al » enfin « ls -ali ». Qu'observez-vous ? Ce répertoire contient-il des fichiers ? Si oui, quels sont-ils ?*

```
$ mkdir rep

$ cd rep

$ ls

$ ls -al

total 2

drwxr-xr-x 2 guest users 1024 Nov 25 21:36 .

drwx------ 3 guest users 1024 Nov 25 21:36 ..

$ ls -lai

total 2

38917 drwxr-xr-x 2 guest users 1024 Nov 25 21:36 .

77828 drwx------ 3 guest users 1024 Nov 25 21:36 ..
```

A sa création, un fichier répertoire contient deux fichiers répertoires cachés « . » et « .. ».

- *Pour vous aidez à répondre à ces questions, lancer les commandes suivantes : « $ cat . », « $ rm . » et « $ cd .. » ; puis pour chacune d'elles, expliquer le message renvoyé par l'interprète de commande (shell).*

```
$ cat .

cat: .: est un répertoire.

$ rm .

rm: Ne peut enlever `.' or `..'

$ pwd
```

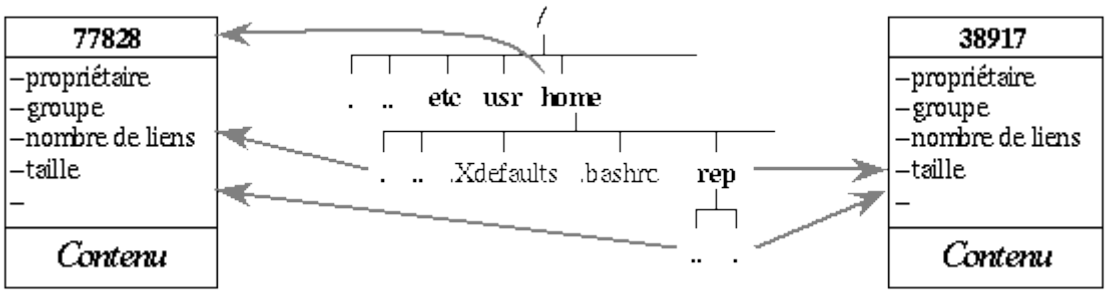
```
/home/guest/rep

$ cd .

$ pwd

/home/guest/rep

$ cd ..

$ pwd

/home/guest
```

Les fichiers « . » et « .. » sont des répertoires quon ne peut pas supprimer. Le fichier « . » correspond au répertoire courant et le fichier « .. » au répertoire père.

- *Retourner dans votre répertoire d'accueil (« home directory »), lancer la commande « ls -ali | more » (lire « pipe more »). En observant les deux premières lignes obtenues, pouvez-vous compléter la réponse à la question précédente (nature des fichiers observés) ?*

```
$ pwd

/home/guest

$ ls -ali | more

total 11

77828 drwx------ 3 guest users 1024 Nov 25 21:36 .

2 drwxr-xr-x 6 root root 1024 Nov 25 19:59 ..

77829 -rw-r--r-- 1 guest users 3768 Nov 25 19:59 .Xdefaults

77832 -rw-r--r-- 1 guest users 124 Nov 25 19:59 .bashrc

38917 drwxr-xr-x 2 guest users 1024 Nov 25 21:36 rep

&

$ ls lai rep/

total 2

38917 drwxr-xr-x 2 guest users 1024 Nov 25 21:36 .

77828 drwx------ 3 guest users 1024 Nov 25 21:36 ..
```

L'inode du fichier « rep/.. » est identique à l'inode du fichier « . ». Un fichier « .. » correspond donc au répertoire père du répertoire qui le contient. De la même façon, l'inode du fichier « rep/. » correspond à celui du répertoire « rep ». Un fichier répertoire « . » fait donc référence au répertoire qui le contient.

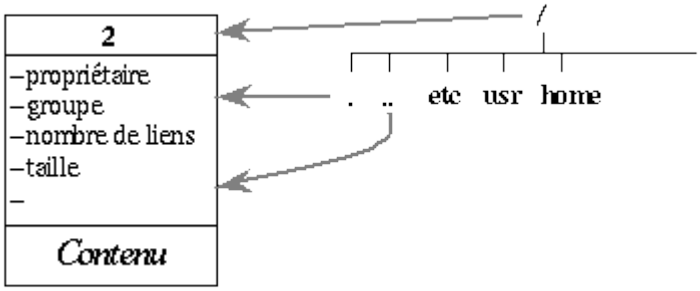


- *Comment cela se passe-t-il pour la racine ? Expliquer. La commande « $ cd /.. » est-elle correcte ?*

```
$ cd /

$ ls -lai | more

total 52

2 drwxr-xr-x 17 root root 1024 Jan 7 1999 .

2 drwxr-xr-x 17 root root 1024 Jan 7 1999 ..

22353 drwxr-xr-x 2 root root 2048 Jan 7 1999 bin

24385 drwxr-xr-x 2 root root 1024 Jul 5 22:04 boot

58994 drwxr-xr-x 2 root root 1024 Jan 7 1999 bru

&

$ pwd

/

$ cd .

$ pwd

/

$ cd ..
```

```
$ pwd

/
```

Cest un peu différent dans le répertoire racine. Le fichier « . » du répertoire racine fait là encore référence au répertoire racine. En revanche, puisque le répertoire racine n'a pas de répertoire père (par définition et par construction), le fichier « .. » qu'il contient ne peut pas référencer le répertoire père. Le fichier « .. » du répertoire racine correspond au répertoire racine lui-même (comme le répertoire « . »).



- *En utilisant cette notion de lien « hard », expliquer en général combien un répertoire contient de sous-répertoires. Pour cela vous prendrez le cas du répertoire « /etc » et vous vérifierez votre formule sur votre répertoire « rep » sous votre répertoire d'accueil. Vous pourrez vérifier votre réponse en utilisant la commande « $ ls -ld » sur ces deux exemples et en comptant à la main le nombre de sous-répertoires.*

```
$ pwd

/etc

$ ls -la | grep ^d

drwxr-xr-x 22 root root 2048 Nov 25 19:59 .

drwxr-xr-x 17 root root 1024 Jan 7 1999 ..

drwxr-xr-x 12 root root 1024 Jan 7 1999 X11

drwxr-xr-x 2 root root 1024 Jan 7 1999 cron.daily

drwxr-xr-x 2 root root 1024 Aug 26 1997 cron.hourly

drwxr-xr-x 2 root root 1024 Jul 25 1997 cron.monthly

drwxr-xr-x 2 root root 1024 Jan 7 1999 cron.weekly

drwxr-xr-x 2 root root 1024 Jan 7 1999 default
```

```
drwxr-xr-x 2 root root 1024 Jan 7 1999 logrotate.d

drwxr-xr-x 2 root root 1024 Jan 7 1999 mail

drwxr-xr-x 2 root root 1024 Jan 7 1999 nmh

drwxr-xr-x 2 root root 1024 Jan 7 1999 pam.d

drwxr-xr-x 2 root root 1024 Jan 7 1999 pcmcia

drwxr-xr-x 2 root root 1024 Jan 7 1999 ppp

drwxr-xr-x 2 root root 1024 Jan 7 1999 profile.d

drwxr-xr-x 10 root root 1024 Jan 7 1999 rc.d

drwxr-xr-x 2 root root 1024 Jan 7 1999 security

drwxr-xr-x 2 root root 1024 Jan 7 1999 skel

drwxr-xr-x 2 root root 1024 May 6 1998 smrsh

drwxr-xr-x 3 root root 1024 Jan 7 1999 sysconfig

drwxrwx--- 3 uucp uucp 1024 Jan 7 1999 uucp

drwxr-xr-x 2 root root 1024 Jan 7 1999 vga
```

Pour tout répertoire autre que le répertoire racine, le nombre de ses sous-répertoires peut se calculer à partir du nombre de ces liens hards ; si *l* est le nombre de ces liens hards, alors le répertoire contient *l-2* sous-répertoires.

Exemples : le nombre de liens hards du répertoire « /etc » (et donc du répertoire « /etc/. ») est 22. Ce répertoire contient donc 20 sous-répertoires. Le répertoire « /etc/vga » ne contient aucun sous-répertoire (le nombre de ses liens hards étant 2). Le répertoire « /etc/rc.d » contient 8 sous-répertoire. Etc.

Cette formule se vérifie avec n'importe quel autre répertoire autre que la racine &

```
$ cd

$ pwd

/home/guest

$ ls -la

total 11
```

```
drwx------ 3 guest users 1024 Nov 25 21:36 .

drwxr-xr-x 6 root root 1024 Nov 25 19:59 ..

-rw-r--r-- 1 guest users 3768 Nov 25 19:59 .Xdefaults

-rw-r--r-- 1 guest users 187 Nov 25 20:22 .bash_history

-rw-r--r-- 1 guest users 24 Nov 25 19:59 .bash_logout

-rw-r--r-- 1 guest users 220 Nov 25 19:59 .bash_profile

-rw-r--r-- 1 guest users 124 Nov 25 19:59 .bashrc

drwxr-xr-x 2 guest users 1024 Nov 25 21:36 rep

$ cd rep/

$ ls -la

total 2

drwxr-xr-x 2 guest users 1024 Nov 25 21:36 .

drwx------ 3 guest users 1024 Nov 25 21:36 ..
```

- *Placez-vous dans le répertoire « rep » et créer le fichier de nom « .fich1 ». Remonter d'un niveau dans l'arborescence (« $ cd .. ») puis effectuer les deux commandes « rmdir » et « ls » qui produiront les résultats suivants :*

```
$ rmdir rep

rmdir : rep not empty

$ ls -l rep

total 0
```

- *Expliquer en utilisant la commande « ls » et son option « -a ».*

```
$ ls -la rep

total 2

drwxr-xr-x 2 guest users 1024 Nov 25 21:53 .
```

```
drwx------ 3 guest users 1024 Nov 25 21:36 ..

-rw-r--r-- 1 guest users 0 Nov 25 21:53 .fich1
```

La commande « rmdir » permet de supprimer des répertoires vides. La commande « ls » (sans option) permet de lister les fichiers non cachés (ceux dont le nom ne commence pas par un « . ») du répertoire « rep ». L'option « -a » permet de lister tous les fichiers (y compris les fichiers cachés). Elle permet donc de s'apercevoir que le répertoire nest pas vide et qu'il ne peut donc pas être supprimé avec la commande « rmdir ».

Remarque : La commande « rm r » permet de supprimer un répertoire et tout ce quil contient (y compris les sous-répertoires). Il convient de l'utiliser avec prudence !!!

- *Nom absolu, nom relatif & Expliquer ces deux notions que vous illustrerez en utilisant la commande « cd » en donnant deux chemins différents qui modifient votre répertoire courant pour devenir celui qui correspond au répertoire d'accueil d'un autre étudiant.*

```
$ pwd

/home/guest

$ cd ../meric/ Nommage relatif

$ pwd

/home/meric

$ cd

$ cd /home/meric Nommage absolu

$ pwd

/home/meric
```

Le nommage relatif consiste à spécifier le chemin d'accès à un fichier à partir du répertoire courant. Le nommage absolu consiste à spécifier le chemin d'accès à un fichier à partir de la racine.

Remarque : Ces deux méthodes de nommage s'appliquent à nimporte quel répertoire. D'autres méthodes permettent de se déplacer vers un répertoire de travail (« home directory »).

Supposons par exemple que vous soyez connecté sous le login « guest » (votre répertoire de travail étant « /home/guest »).

Les commandes suivantes permettent de revenir dans votre répertoire de travail quelque soit le répertoire courant.

```
$ cd

$ pwd

/home/guest
```

```
$ cd ~

$ pwd

/home/guest
```

```
$ cd ~guest

$ pwd

/home/guest
```

La commande suivante permet d'aller dans le répertoire de travail de l'utilisateur « meric ».

```
$ cd ~meric

$ pwd

/home/meric
```

Noms génériques et principe de fonctionnement de la commande « ls »

- *Construisez la sous-arborescence suivante sous « rep » :*

1. Dans « rep » vous créez le répertoire « rep1 » et les fichiers « .c », « a » et « b ».

2. Dans « rep1 » vous créez le répertoire « rep2 » et les fichiers « a1 », « b1 » et « .c1 ».

3. Dans « rep2 » vous créez le répertoire « rep3 » et les fichiers « a2 », « b2 » et « .c2 ».

4. Enfin dans « rep3 » vous créez le fichier « a3 ».

```
$ pwd
/home/guest/rep
$ ls -la
total 2
drwxr-xr-x 2 guest users 1024 Nov 25 22:01 .
drwx------ 3 guest users 1024 Nov 25 21:36 ..
$ mkdir rep1
$ touch .c a b
$ cd rep1
$ mkdir rep2
$ touch a1 b1 .c1
$ cd rep2/
$ mkdir rep3
$ touch a2 b2 .c2
$ cd
.c2 a2 b2 rep3
$ cd rep3/
$ touch a3
```

- *Donner les résultats de : « $ ls ; ls . ; ls .. ; ls -a ; ls -a . ; ls -a ..*

```
$ pwd

/home/guest/rep/rep1/rep2/rep3

$ ls

a3

$ ls .

a3

$ ls ..

a2 b2 rep3

$ ls -a

. .. a3

$ ls -a .

. .. a3

$ ls -a ..

. .. .c2 a2 b2 rep3
```

- *Pour chacun des noms génériques ci-dessous, expliquez, en utilisant les commandes « echo », « ls » et « ls -a », comment le shell réalise l'expansion des noms (passage d'un nom générique à une liste de noms) de fichiers correspondant : * ; .* ; */* ; */.* ; .*/* ; .*/.**

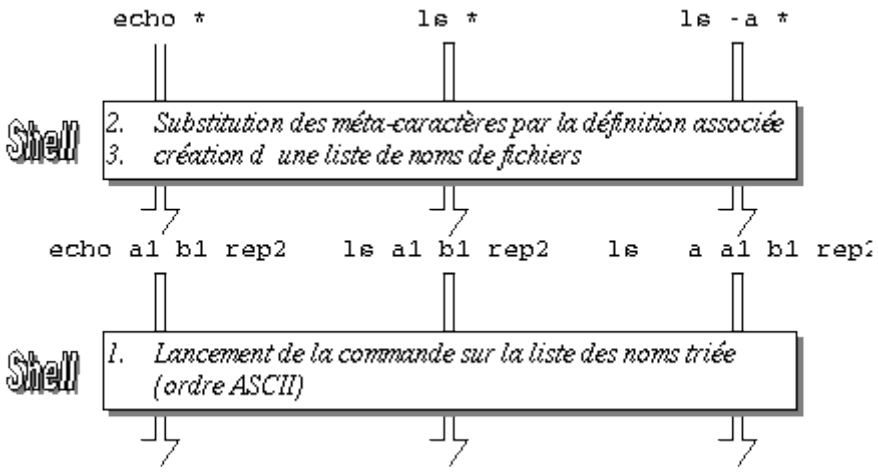
	Dans le répertoire rep3	**Dans le répertoire rep1**`echo *a3a1 b1 rep2echo .*. ... .. .c1echo */**/*rep2/a2 rep2/b2 rep2/rep3echo */.**/.*rep2/. rep2/.. rep2/.c2echo .*/*../a2 ../b2 ../rep3 ./a3../a ../b ../rep1 ./a1 ./b1 ./rep2echo .*/.*../. ../.. ../.c2 ./. ./..../. ../.. ../.c ./. ./.. ./.c1`

	Dans le répertoire rep3	**Dans le répertoire rep1**`ls *`a3a1 b1 **rep2:** `a2 b2 rep3`**`ls .*.:`** a3 **..:** a2 b2 rep3.c1 **.:** a1 b1 rep2 **..:** `a b rep1`**`ls */*`**`ls: */*: Aucun fichier ou répertoire de ce type.`rep2/a2 rep2/b2 **rep2/rep3:** `a3`**`ls */.*`**`ls: */.*: Aucun fichier ou répertoire de ce type.`rep2/.c2 **rep2/.:** a2 b2 rep3 **rep2/..:** `a1 b1 rep2`**`ls .*/*`**../a2 ../b2 ./a3 **../rep3:** `a3`../a ../b ./a1 ./b1 **../rep1:** a1 b1 rep2 **./rep2:** `a2 b2 rep3`**`ls .*/.*`**../.c2 **../.:** a2 b2 rep3 **../..:** a1 b1 rep2

		./.: a3 **./..:** a2 b2 rep3../.c ./.c1 **../.:** a b rep1 **../..:** rep **./.:** a1 b1 rep2 **./..:** a b rep1

	Dans le répertoire rep3	**Dans le répertoire rep1**ls a *a3a1 b1 **rep2:** . .. .c2 a2 b2 rep3**ls a .***.: . .. a3 ..: . .. .c2 a2 b2 rep3.c1 **.:** . .. .c1 a1 b1 rep2 **..:** . .. .c a b rep1**ls a */***ls: */*: Aucun fichier ou répertoire de ce type.rep2/a2 rep2/b2 **rep2/rep3:** . .. a3**ls a */.***ls: */.*: Aucun fichier ou répertoire de ce type.rep2/.c2

		rep2/.: . .. .c2 a2 b2 rep3 **rep2/..:** . .. .c1 a1 b1 rep2**ls a .*/***../a2 ../b2 ./a3 **../rep3:** . .. a3../a ../b ./a1 ./b1 **../rep1:** . .. .c1 a1 b1 rep2 **./rep2:** . .. .c2 a2 b2 rep3**ls a .*/.***../.c2 **../.:** . .. .c2 a2 b2 rep3 **../..:** . .. .c1 a1 b1 rep2 **./.:** . .. a3 **./..:** . .. .c2 a2 b2 rep3../.c ./.c1 **../.:** . .. .c a b rep1 **../..:** . .Xdefaults .bash_logout .bashrc .. .bash_history .bash_profile rep **./.:** . .. .c1 a1 b1 rep2 **./..:** . .. .c a b rep1



- *Placez-vous sous « rep » et créer à l'aide de la commande « touch » les quatre fichiers suivants : « 21 », « 27 », « 2a » et « 39 ». En utilisant la commande « ls » et les noms génériques de fichiers, afficher tous les fichiers de votre répertoire courant (« rep ») dont les noms sont un nombre compris entre 20 et 40.*

```
$ cd ..

$ pwd

/home/guest/rep

$ touch "21" "27" "2a" "39"

$ ls

21 27 2a 39 a b rep1

$ ls [23][0-9]

21 27 39
```

- *Sous le même répertoire courant, créer les fichiers « b.truc », « chose », « s.f » et « sous ». Trouver une commande qui permet d'afficher tous les fichiers dont le nom ne commence pas par « s. ».*

```
$ touch b.truc chose s.f sous

$ ls
```

```
21 27 2a 39 a b

b.truc chose rep1 s.f sous
```

Le fichier « s.f » est le seul nom de fichier de le dernier caractère est « f ». On peut donc afficher tous les nom de fichier dont le dernier caractère nest pas un « f » &

```
$ ls -d *[^f]

21 27 2a 39 a b b.truc chose rep1 sous
```

Cette commande ne répond plus au problème si le répertoire contient le fichier « sauf ». Dans ce cas, « s.f » est le seul fichier dont le nom dont l'avant-dernier caractère est « . ». On peut donc afficher les noms d'au moins deux caractères dont l'avant-dernier caractère nest pas « . » (« *[^.]? ») et les noms composés dun seul caractère (« ? »).

```
$ touch sauf

$ ls -d *[^f]

21 27 2a 39 a b b.truc chose rep1 sous

$ ls -d *[^.]? ?

21 2a a b.truc rep1 sous

27 39 b chose sauf
```

Cette commande ne répond plus au problème si le répertoire contient le fichier « a.b ». Dans ce cas, il faut afficher :

- tous les noms de fichier d'au moins deux caractères dont le premier caractère nest pas « s » et dont le second caractère est « . » (« [^s].* ») ;
- tous les noms de fichier d'au moins deux caractères dont le premier caractère quelconque et dont le second caractère nest pas « . » (« ?[^.]* ») ;
- tous les noms de fichier composés d'un seul caractère (« ? »).

```
$ touch a.b

$ ls

21 27 2a 39 a

a.b b b.truc chose rep1

s.f sauf sous

$ ls -d [^s].* ?[^.]* ?

21 2a a b chose sauf

27 39 a.b b.truc rep1 sous
```

Remarque : La commande suivante est également satisfaisante. Toutefois, on notera une certaine redondance due au fait que certains noms de fichier « matchent » avec les deux noms génériques.

```
$ ls -d [^s]* ?[^.]*

21 27 2a 39 a b chose rep1 sauf

21 27 2a 39 a.b b.truc chose rep1 sous
```

Complément :

```
$ ls

10 20 33 50 Bb Zzz bb cc zzz

12 30 42 Aaa Cccc aa bbb ccc
```

Afficher les noms de fichier commençant par 2, 4, 5 ou 6 &

```
$ ls [24-6]*

20 42 50
```

Afficher les noms de fichier commençant par un caractère entre 0 et 9 ou entre a et z &

```
$ ls [0-9a-z]*

10 12 20 30 33 42 50 aa bb bbb cc ccc zzz
```

Afficher les noms de fichier commençant par A, C ou un caractère entre a et z &

```
$ ls [ACa-z]*

Aaa Cccc aa bb bbb cc ccc zzz
```

Afficher les noms de fichier commençant par un caractère entre 4 et 6 ou par a &

```
$ ls [4-6a]*

42 50 aa

$ ls [a6-4]* Attention à lordre des caractères séparés par -

aa (ordre ASCII)

$ ls [a4-6]*

42 50 aa
```

Afficher les noms de fichier commençant par un caractère entre A et z (i.e. une lettre majuscule ou minuscule) &

```
$ ls [A-z]*

Aaa Bb Cccc Zzz aa bb bbb cc ccc zzz
```

Attention à lordre des caractères séparés par (ordre ASCII).

```
$ ls [z-A]*

ls: [z-A]*: Aucun fichier ou répertoire de ce type.
```

Afficher les noms de fichier commençant par un caractère entre 1 et 9, un caractère entre A et Z ou un caractère entre a et z (i.e. une lettre majuscule ou minuscule) &

```
$ ls [1-z]*

10 20 33 50 Bb Zzz bb cc zzz

12 30 42 Aaa Cccc aa bbb ccc
```

- *En utilisant l'éditeur « vi » construire un fichier dont le contenu contient sur plusieurs lignes, plusieurs occurrences de la chaîne de caractères « le système unix », écrites sous les formes suivantes : « le système unix », « Le système unix », « Le Système Unix » ou encore « le système Unix ». Utiliser les commandes « grep », « | » et « wc » pour compter le nombre d'occurrences de cette chaîne, toute orthographe considérée (majuscule/minuscule).*

```
$ cat fich

le systeme unix

Le systeme unix

Le Systeme unix

Le Systeme Unix

le Systeme Unix
```

```
le systeme Unix

le Systeme unix

Le systeme Unix

$ wc -l fich

8 fich

$ grep "[lL]e [sS]ysteme [uU]nix" fich

le systeme unix

Le systeme unix

Le Systeme unix

Le Systeme Unix

le Systeme Unix

le systeme Unix

le Systeme unix

Le systeme Unix

$ grep "[lL]e [sS]ysteme unix" fich

le systeme unix

Le systeme unix

Le Systeme unix

le Systeme unix

$ grep "[lL]e [sS]ysteme [uU]nix" fich | wc -l

8
```