Programmation système - Shell et Commandes UNIX

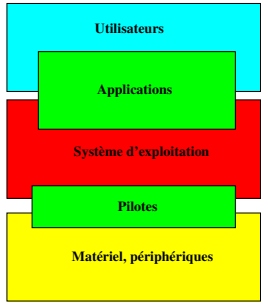
Système de fichiers et Gestion des processus

Tuyêt Trâm DANG NGOC
<dntt@u-cergy.fr>

Université de Cergy-Pontoise

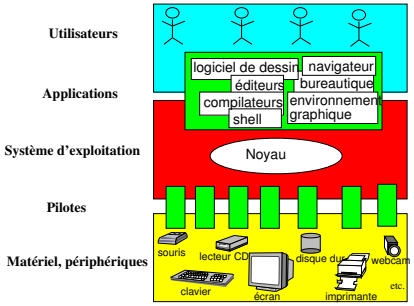
Système d'exploitation

Un Système d'exploitation *Operating System - OS)* assure la liaison entre le matériel et les applications.



Système d'exploitation

Un Système d'exploitation *Operating System - OS)* assure la liaison entre le matériel et les applications.



Système d'exploitation

Les caractéristiques et le fonctionnement d'un système d'exploitation seront vus en détail en L3 (S5).
Les systèmes d'exploitation récents gèrent :

- des fichiers : permettant de stocker l'information (souvent sur le disque dur). Ces fichiers sont organisés sous forme d'arborescence de répertoires et de fichiers.
- des processus : représentant les tâches (programmes) à exécuter
- des utilisateurs : identifiés et authentifiés qui possèdent des fichiers et lancent des processus.

Objectifs du shell

1. Fournir une interface pour la saisie de commande
2. Redirection des entrées/sorties standards
3. Analyser les commandes
   - substitution de noms de fichiers
   - substitution de variables
   - redirection d'entrées/sorties
4. Exécution de commandes
   - mode synchrone
   - mode asynchrone
5. Fournir un langage interprété

Utilisation du shell

Deux modes d'utilisation :

- interactif : en ligne de commande.
  1. Présente une invite (*prompt*) à l'utilisateur et attend que celui-ci tape une commande ;
  2. Exécute* la commande tapée par l'utilisateur
  3. Retour en 1.
- non-interactif : scripts shell, batch
  1. Lit une ligne du fichier
  2. Exécute* les instructions données dans la ligne du fichier
  3. Passe à la ligne suivante
  4. Retour en 1

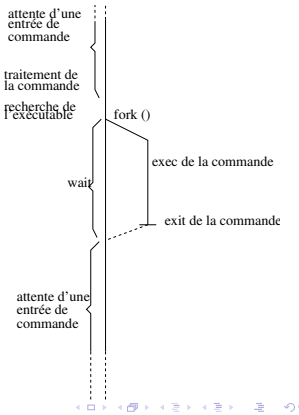
  Le programme s'arrête lorsqu'il n'y a plus de ligne à lire ou lorsqu'un instruction spéciale (**exit** ou **return**) est rencontrée.

Convention : l'invite est :

- $ pour l'utilisateur normal en sh, bsh, bash
- % pour l'utilisateur normal en csh, tcsh
- # pour root dans tous les shells

Exécution d'une commande

1. Attente d'une entrée de commande
2. traitement des caractères spéciaux de la commande
3. recherche de l'exécutable. Si non trouvé, afficher un message d'erreur et revenir en 1.
4. fork () + exec () de la commande à lancer
5. wait de la commande
6. Revenir en 1.



Manuel des commandes

La liste des commandes n'est pas exhaustive, seules les plus courantes et les plus adaptées à votre enseignement sont présentées.
De la même façon, seules les options les plus courantes seront indiquées pour chaque commande.

man *commande*

Pour plus d'information et de précision sur les commandes n'hésitez surtout pas à lire les pages de manuel associées en tapant la commande :
man *nom_de_la_commande*

Caractères spéciaux

Caractères	Description
tabulation, espace	Délimiteur de mot
retour chariot	Fin de la commande à exécuter
&	Lance une commande en tâche de fond
; ; ;	Séparateur de commande
∗?[][^]	Substitution de noms de fichiers
&& ||!	Opérateurs booléens
' " \	Caractères de quotation
<><<>> ‘ |<>< & > & << − >|	Opérateurs de redirection d'entrées sorties
$	Valeur d'une variable
#	Début de commentaires
(){}	Groupement de commande

Mots réservés

Mots réservés	Signification
case ... in esac	condition de test
for ... in ... do ... done	boucle itérative
if ... then ... elif ... else ... fi	condition de test
while ... do ... done	boucle avec condition
until ... do ...done	boucle avec condition
break, continue	sortie de boucle
return, exit	sortie

Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$ type ls
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$ type ls
ls is /bin/ls
$
```

Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$ type ls
ls is /bin/ls
$ type cat
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$ type ls
ls is /bin/ls
$ type cat
cat is /bin/cat
$
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$ type ls
ls is /bin/ls
$ type cat
cat is /bin/cat
$ type gcc
```


Commandes internes (builtins)

Les commandes internes sont réalisées de manière interne par le shell lui-même ; c'est-à-dire qu'il n'y a pas de création de processus pour exécuter la commande. Ces commandes ne possèdent pas d'exécutables associés puisqu'elles sont codées en interne au shell. Une méthode pour identifier les builtins est d'utiliser la commande interne **type**.

```
$ type cd
cd is a shell builtin
$ type echo
echo is a shell builtin
$ type ls
ls is /bin/ls
$ type cat
cat is /bin/cat
$ type gcc
gcc is /usr/bin/gcc
$
```


Commandes internes (builtins)

alias	bg	builtin
bind	cd	chdir
command	echo	eval
exec	exit	export
fc	fg	getopts
hash	jobid	jobs
pwd	read	readonly
set	setvar	shift
trap	type	ulimit
umask	unalias	unset
wait		.
	:	

Commandes (externes)

Les commandes qui ne sont pas internes sont des exécutables qui peuvent être trouvés dans la hiérarchie des répertoires :

- soit directement si le chemin complet est spécifié
- soit trouvé par le shell en explorant les répertoires spécifiés dans la variable d'environnement `PATH`.

/bin/cat	/bin/chmod	/bin/cp	/bin/date
/bin/kill	/bin/ln	/bin/ls	/bin/mkdir
/bin/mv	/bin/ps	/bin/pwd	/bin/rmdir
/bin/sleep	/usr/bin/awk	/usr/bin/basename	/usr/bin/bc
/usr/bin/bg	/usr/bin/chgrp	/usr/bin/cmp	/usr/bin/comm
/usr/bin/cut	/usr/bin/diff	/usr/bin/dirname	/usr/bin/find
/usr/bin/grep	/usr/bin/head	/usr/bin/join	/usr/bin/man
/usr/bin/more	/usr/bin/nohup	/usr/bin/paste	/usr/bin/sed
/usr/bin/sort	/usr/bin/tail	/usr/bin/time	/usr/bin/top
/usr/bin/touch	/usr/bin/uniq	/usr/bin/vi	/usr/bin/w
/usr/bin/wc	/usr/bin/xargs	/usr/sbin/chown	

Caractères de substitution

Car.	Commande
*	N'importe quelle séquence de caractères
?	N'importe quel caractère
[]	N'importe quel caractère choisi dans les caractères donnés entre crochets
[^]	n'importe quel caractère sauf ceux dans les caractères donnés entre crochets
[–]	n'importe quel caractère dans la plage de caractères donnés entre crochets

```
$
```


Caractères de substitution

Car.	Commande
*	N'importe quelle séquence de caractères
?	N'importe quel caractère
[]	N'importe quel caractère choisi dans les caractères donnés entre crochets
[^]	n'importe quel caractère sauf ceux dans les caractères donnés entre crochets
[–]	n'importe quel caractère dans la plage de caractères donnés entre crochets

```
$ ls *
```


Caractères de substitution

Car.	Commande
*	N'importe quelle séquence de caractères
?	N'importe quel caractère
[]	N'importe quel caractère choisi dans les caractères donnés entre crochets
[^]	n'importe quel caractère sauf ceux dans les caractères donnés entre crochets
[–]	n'importe quel caractère dans la plage de caractères donnés entre crochets

```
$ ls *
achat bateau chat chateau cheval chien gateau rateau
$
```

Caractères de substitution

Car.	Commande
*	N'importe quelle séquence de caractères
?	N'importe quel caractère
[]	N'importe quel caractère choisi dans les caractères donnés entre crochets
[^]	n'importe quel caractère sauf ceux dans les caractères donnés entre crochets
[–]	n'importe quel caractère dans la plage de caractères donnés entre crochets

```
$ ls *
achat bateau chat chateau cheval chien gateau rateau
$ ls ?ateau
bateau gateau rateau
$ ls *ateau
bateau chateau gateau rateau
$ ls [gr]ate*
gateau rateau
$ ls [^br]ateau
gateau
$ ls [a-c]*
achat bateau chat chateau cheval chien gateau rateau
$
```

Caractères de substitution

Car.	Commande
*	N'importe quelle séquence de caractères
?	N'importe quel caractère
[]	N'importe quel caractère choisi dans les caractères donnés entre crochets
[^]	n'importe quel caractère sauf ceux dans les caractères donnés entre crochets
[–]	n'importe quel caractère dans la plage de caractères donnés entre crochets

```
$ ls *
achat bateau chat chateau cheval chien gateau rateau
$ ls ?ateau
bateau gateau rateau
$ ls *ateau
bateau chateau gateau rateau
$ ls [gr]ate*
gateau rateau
$ ls [^br]ateau
gateau
$ ls [a-c]*
achat bateau chat chateau cheval chien
```


Caractères spéciaux

' " \ changent la façon dont le shell interprête les caractères spéciaux

Symbole	Signification
' (single-quote)	le shell ignore tout caractère spéciaux entre deux '
" (double-quote)	le shell ignore tout caractère spéciaux entre deux ", à l'exception de $ et \ et ‘
\ (antislash ou backslash)	le shell ignore le caractère spécial suivant le \
‘ (backquote ou antiquote)	le shell exécute ce qu'il y a entre deux ‘

Caractères spéciaux : exemples

```
$
```


Caractères spéciaux : exemples

```
$ ls
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$
```

Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$ echo ‘whoami‘ et le ${TITI} et les c*; ls
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$ echo ‘whoami‘ et le ${TITI} et les c*; ls
dntt et le raton-laveur et les chat chien
chat chien poisson
$
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$ echo ‘whoami‘ et le ${TITI} et les c*; ls
dntt et le raton-laveur et les chat chien
chat chien poisson
$ echo ’‘whoami‘ et le ${TITI} et les c*; ls’
```


Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$ echo ‘whoami‘ et le ${TITI} et les c*; ls
dntt et le raton-laveur et les chat chien
chat chien poisson
$ echo ’‘whoami‘ et le ${TITI} et les c*; ls’
‘whoami‘ et le ${TITI} et les *; ls
$
```

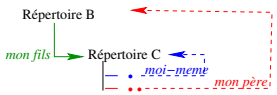

Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$ echo ‘whoami‘ et le ${TITI} et les c*; ls
dntt et le raton-laveur et les chat chien
chat chien poisson
$ echo ’‘whoami‘ et le ${TITI} et les c*; ls’
‘whoami‘ et le ${TITI} et les *; ls
$ echo "‘whoami‘ et le ${TITI} et les c*; ls"
```

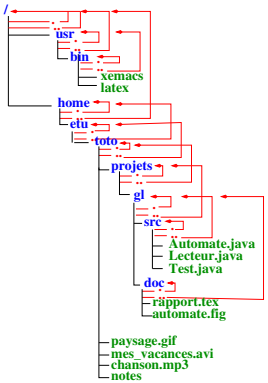

Caractères spéciaux : exemples

```
$ ls
chat chien poisson
$ whoami
dntt
$ TITI=raton-laveur
$ echo ‘whoami‘ et le ${TITI} et les c*; ls
dntt et le raton-laveur et les chat chien
chat chien poisson
$ echo ’‘whoami‘ et le ${TITI} et les c*; ls’
‘whoami‘ et le ${TITI} et les *; ls
$ echo "‘whoami‘ et le ${TITI} et les c*; ls"
dntt et le raton-laveur et les c*; ls
$
```


Arborescence UNIX

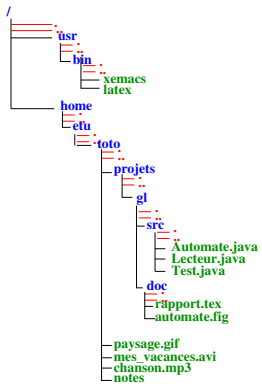


Exemple d'arborescence



Exemple d'arborescence

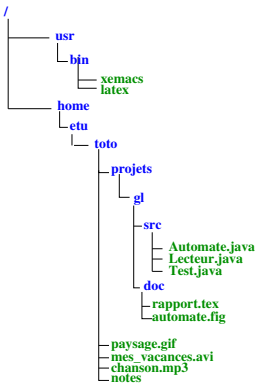
.



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple d'arborescence

Pour des raisons de lisibilité, on ne représente en général pas . et .. dans les dessins d'arborescence (implicite).



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$
```

Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
/var
$
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
/var
$ cd log
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
/var
$ cd log
$
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
/var
$ cd log
$ pwd
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
/var
$ cd log
$ pwd
/var/log
$
```


Changement et affichage du répertoire courant : cd, pwd

cd
La commande interne **cd** permet de changer de répertoire courant.

pwd
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
```

```
$ pwd
/var
$ cd log
$ pwd
/var/log
$ cd .
```

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
$ pwd
/var
$ cd log
$ pwd
/var/log
$ cd .
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
$ pwd
/var
$ cd log
$ pwd
/var/log
$ cd .
$ pwd
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Changement et affichage du répertoire courant : cd, pwd

cd

La commande interne **cd** permet de changer de répertoire courant.

pwd

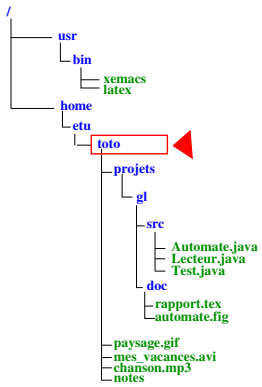
Afficher le nom du répertoire de travail en cours

```
$ pwd
/users/dntt
$ cd /var/log
$ pwd
/var/log
$ cd ..
$ pwd
/var/log
$ cd .
$ pwd
/var/log
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

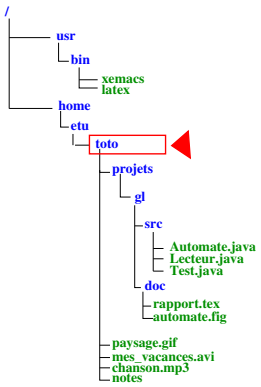
```
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

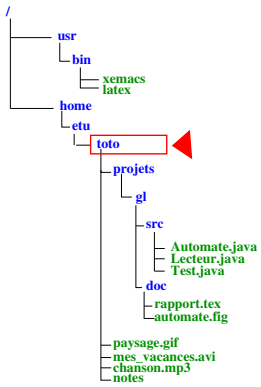
```
$ pwd
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

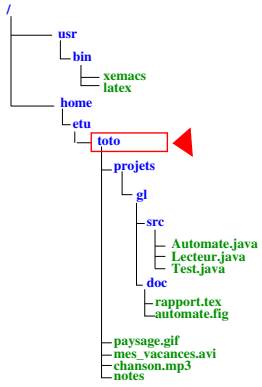
```
$ pwd
/home/etu/toto
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

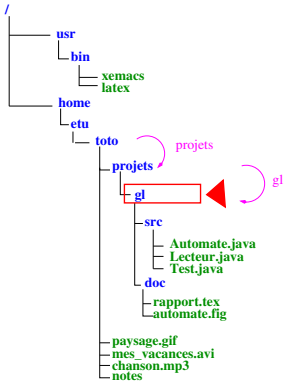
```
$ pwd
/home/etu/toto
$ cd projets/gl
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

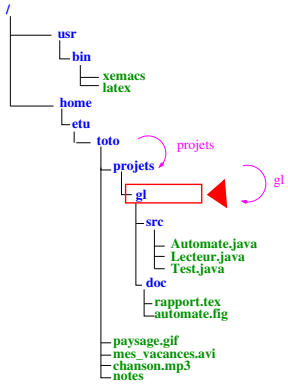
```
$ pwd
/home/etu/toto
$ cd projets/gl
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

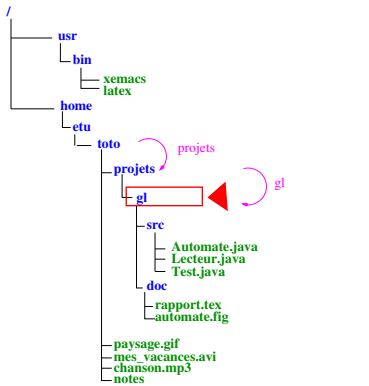
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
```



Exemple : cd et pwd

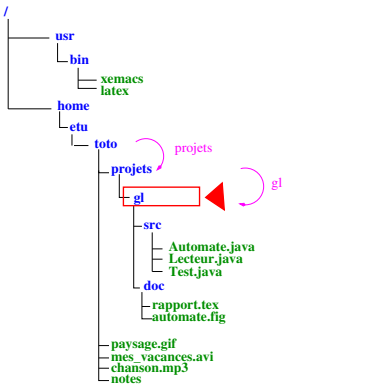
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

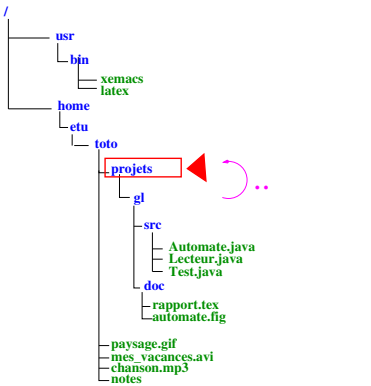
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

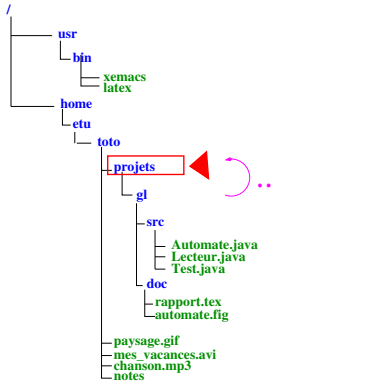
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

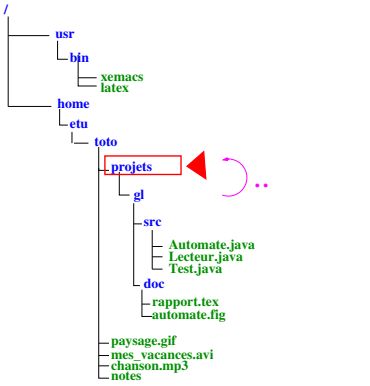
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

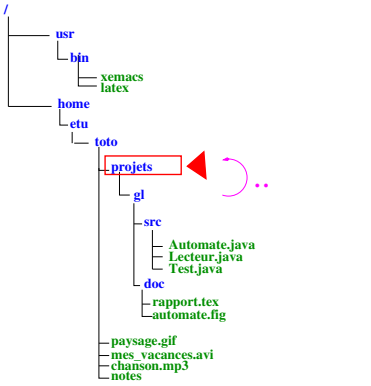
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

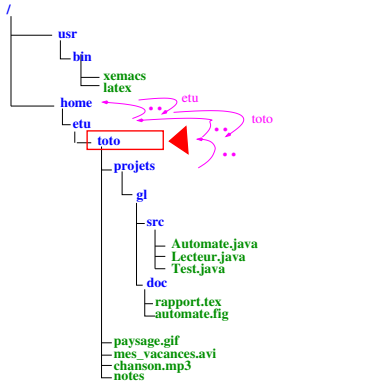
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

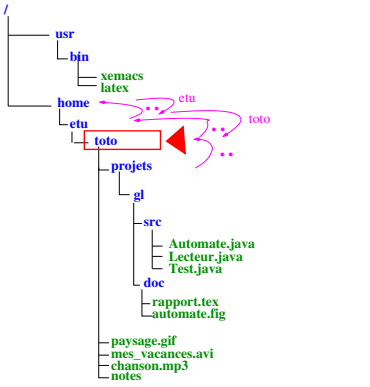
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Exemple : cd et pwd

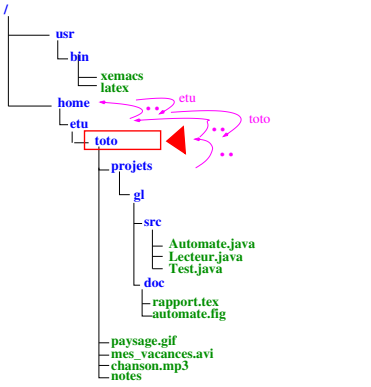
```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
```



Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

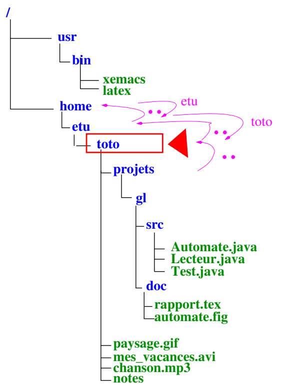
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$
```



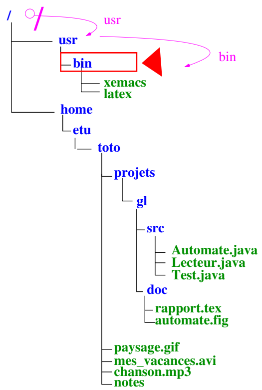
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
```



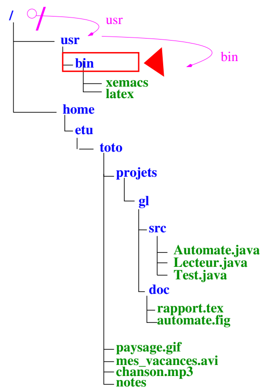
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$
```



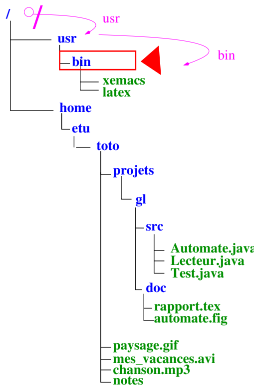
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
```



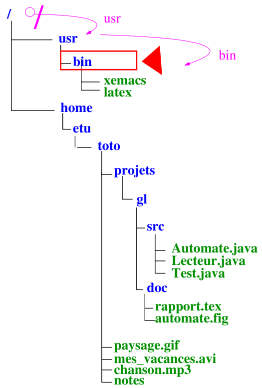
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$
```



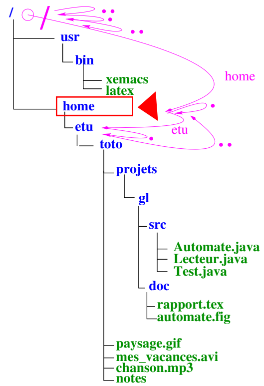
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
/../../../home/./etu/./..
```



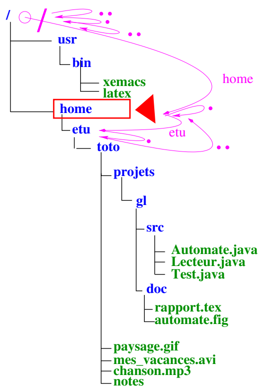
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
/../../../home/./etu/./..
$
```



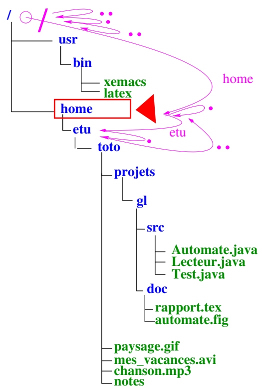
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
/../../../home/./etu/./..
$ pwd
```



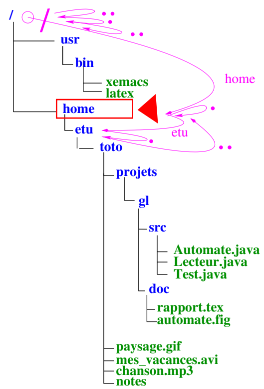
Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
/../../../home/./etu/./..
$ pwd
/home
$
```



Exemple : cd et pwd

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
/../../../home/./etu/./..
$ pwd
/home
$
```



Gestion des répertoires

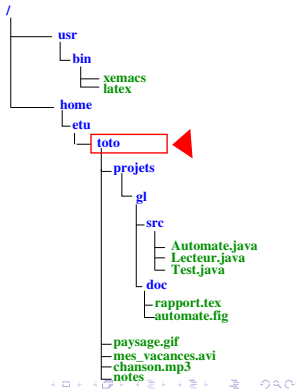
ls

Liste les fichiers d'un répertoire.
Utilisé simplement avec des arguments, suivant le type de l'argument :

- Si c'est un répertoire, liste les fichiers qui y sont contenus
- Affiche le nom du fichier si c'est un fichier
- Affiche une erreur sur l'erreur standard si le fichier n'existe pas.

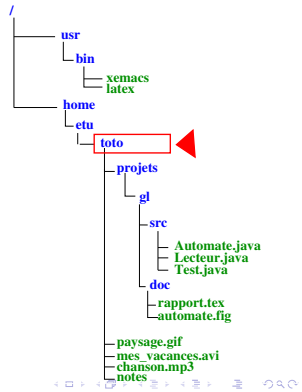
Exemple : ls

```
$
```



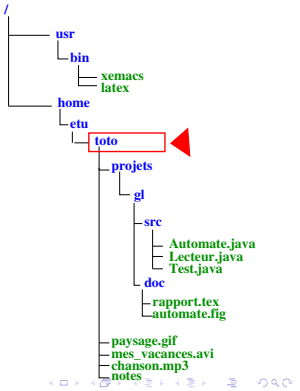
Exemple : ls

```
$ ls
```



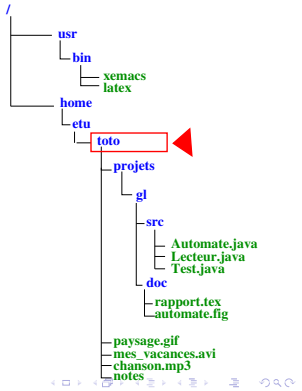
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$
```



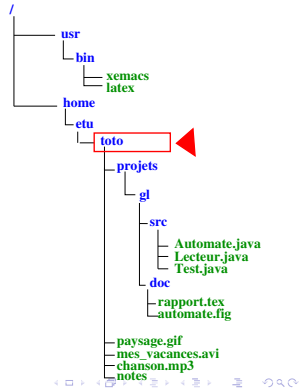
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
```



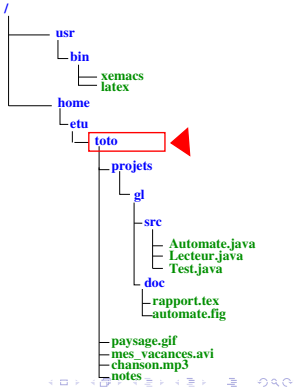
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$
```



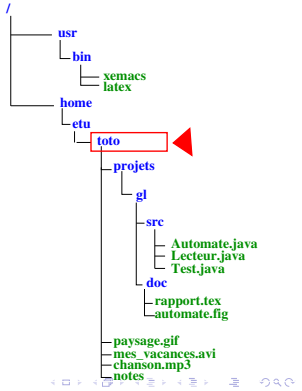
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$ ls ..
```



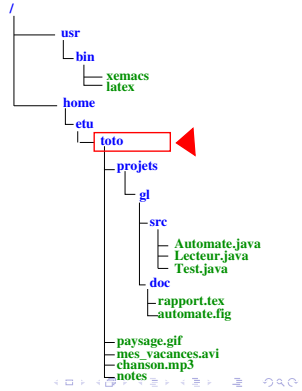
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$ ls ..
toto
$
```



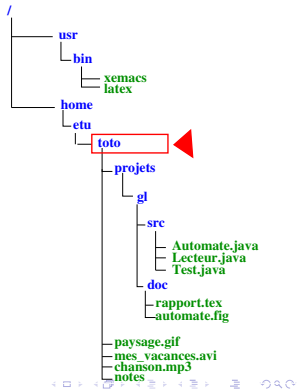
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$ ls ..
toto
$ ls /usr/bin paysage.gif projets/gl
```



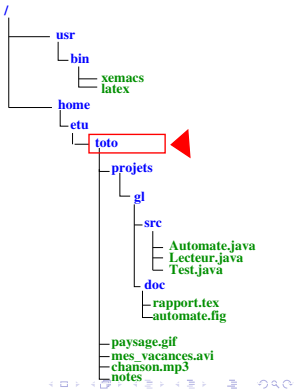
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$ ls ..
toto
$ ls /usr/bin paysage.gif projets/gl
 latex xemacs
paysage.gif src doc
$
```



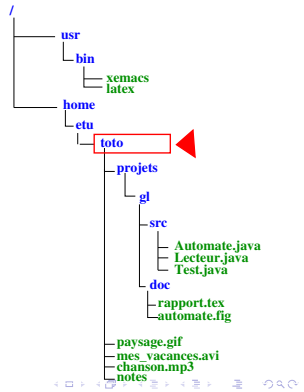
Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$ ls ..
toto
$ ls /usr/bin paysage.gif projets/gl
 latex xemacs
paysage.gif src doc
$ ls / toto ../toto / /home/etu/toto
```



Exemple : ls

```
$ ls
 projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$ ls projets/gl
src doc
$ ls ..
toto
$ ls /usr/bin paysage.gif projets/gl
 latex xemacs
paysage.gif src doc
$ ls / toto ../toto / /home/etu/toto
 usr home
toto : no such file or directory
projets paysage.gif
mes_vacances.avi chanson.mp3
notes projets paysage.gif
mes_vacances.avi chanson.mp3
notes
$
```



Affichage des fichiers et du contenu de répertoires : ls

ls OPTIONS fichiers...

La commande **ls** affiche tout d'abord l'ensemble de ses arguments fichiers autres que des répertoires. Puis **ls** affiche l'ensemble des fichiers contenus dans chaque répertoire indiqué.

- -R Afficher récursivement le contenu des sous-répertoires.
- -a Afficher tous les fichiers des répertoires, y compris les fichiers commençant par un '.'
- -i Afficher le numéro d'index (i-noeud) de chaque fichier à gauche de son nom.
- -l En plus du nom, afficher le type du fichier, les permissions d'accès, le nombre de liens physiques, le nom du propriétaire et du groupe, la taille en octets, et l'horodatage.

Création et suppression de répertoire : mkdir, rmdir

mkdir OPTIONS *repertoire...*

mkdir crée un répertoire correspondant à chacun des noms mentionnés

- -p Créer les répertoires parents s'ils manquent

rmdir OPTIONS *repertoire...*

rmdir supprime chaque répertoire vide indiqué pour supprimer récursivement des répertoires non-vides, utiliser **rm -r**

Création et suppression de répertoire : mkdir, rmdir

```
$
```


Création et suppression de répertoire : mkdir, rmdir

```
$ ls -al
```


Création et suppression de répertoire : mkdir, rmdir

```
$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$
```


Création et suppression de répertoire : mkdir, rmdir

```
$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
```

Création et suppression de répertoire : mkdir, rmdir

```
$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création et suppression de répertoire : mkdir, rmdir

```
$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14
$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$
```

Création et suppression de répertoire : mkdir, rmdir

```
$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14
```

```
$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$ ls -als
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création et suppression de répertoire : mkdir, rmdir


```
$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14
```


```
$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$ ls -als
total 14
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 .
6 drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
2 -rw-r--r-- 1 dntt users
64 25 jan 11 :42 fichier
2 -rw-r--r-- 1 dntt users
36 25 jan 11 :36 fichier2
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création et suppression de répertoire : mkdir, rmdir

$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14

$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$ ls -als
total 14
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 .
6 drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
2 -rw-r--r-- 1 dntt users
64 25 jan 11 :42 fichier
2 -rw-r--r-- 1 dntt users
36 25 jan 11 :36 fichier2

Tuyêt Trâm DANG NGOC | Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création et suppression de répertoire : mkdir, rmdir

$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14

$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$ ls -als
total 14
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 .
6 drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
2 -rw-r--r-- 1 dntt users
64 25 jan 11 :42 fichier
2 -rw-r--r-- 1 dntt users
36 25 jan 11 :36 fichier2

Tuyêt Trâm DANG NGOC | Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création et suppression de répertoire : mkdir, rmdir

$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14

$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$ ls -als
total 14
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 .
6 drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
2 -rw-r--r-- 1 dntt users
64 25 jan 11 :42 fichier
2 -rw-r--r-- 1 dntt users
36 25 jan 11 :36 fichier2

Tuyêt Trâm DANG NGOC | Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création et suppression de répertoire : mkdir, rmdir

$ ls -al
total 12
drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
-rw-r--r-- 1 dntt users 64
25 jan 11 :42 fichier
-rw-r--r-- 1 dntt users 36
25 jan 11 :36 fichier2
$ mkdir monrep
$ ls -al
total 14

$ ls -als
 total 4
2 drwxr-xr-x 2 dntt users
512 8 fév 13 :15 .
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 ..
$ cd ..
$ ls -als
total 14
2 drwxr-xr-x 3 dntt users
512 8 fév 13 :15 .
6 drwxr-xr-x 37 dntt users
5120 8 fév 13 :04 ..
2 -rw-r--r-- 1 dntt users
64 25 jan 11 :42 fichier
2 -rw-r--r-- 1 dntt users
36 25 jan 11 :36 fichier2

Tuyêt Trâm DANG NGOC | Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

ln OPTIONS *fic1 fic2*

Création de lien entre fichiers.

-s lien symbolique Lorsqu'on crée un lien de *fic1* vers *fic2*, *fic2* pointe vers le même inode que *fic1*. Si *fic1* est effacé, l'inode continue à exister et est encore accessible au moins par *fic2*. Lorsqu'on crée un lien symbolique de *fic1* vers *fic2*, *fic2* référence *fic1*. Si *fic1* est effacé, *fic2* référencera un fichier qui n'existe plus

Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$
```

Tuyêt Trâm DANG NGOC | Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion | Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
```

Tuyêt Trâm DANG NGOC | Programmation système - Shell et Commandes UNIX

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$
```

Tuyêt Trâm DANG NGOC Programmation système - Shell et Commandes UNIX

Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$ cat fichier3
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$ cat fichier3
abcdef
ghijkl
toto
$
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$ cat fichier3
abcdef
ghijkl
toto
$ rm fichier
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$ cat fichier3
abcdef
ghijkl
toto
$ rm fichier
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$ cat fichier3
abcdef
ghijkl
toto
$ rm fichier
```


Création de liens : ln

```
$ ls -il
 total 4
65329 -rw-r--r-- 1 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
$ cat fichier
abcdef
ghijkl
$ ln fichier fichier3
$ ls -il
 total 6
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 2 dntt users 14 8 fév 13 :22 fichier3
$ echo toto >> fichier
$ cat fichier
abcdef
ghijkl
toto
$ cat fichier3
abcdef
ghijkl
toto
$ rm fichier
```


Création de liens symbolique : ln -s

```
$
```

Création de liens symbolique : ln -s

```
$ ls -il
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
```

Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$ cat fichier4
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$ cat fichier4
abcdef
ghijkl
toto
titi
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$ cat fichier4
abcdef
ghijkl
toto
titi
$ rm fichier3
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$ cat fichier4
abcdef
ghijkl
toto
titi
$ rm fichier3
$
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$ cat fichier4
abcdef
ghijkl
toto
titi
$ rm fichier3
$ ls -il
```


Création de liens symbolique : ln -s

```
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 19 8 fév 13 :23 fichier3
$ ln -s fichier3 fichier4
$ ls -il
total 4
65350 -rw-r--r-- 1 dntt users 36 25 jan 11 :36 fichier2
65329 -rw-r--r-- 1 dntt users 24 8 fév 13 :24 fichier3
65378 lrwxr-xr-x 1 dntt users 8 8 fév 13 :24 fichier4 -> fichier3
$ echo titi >> fichier3
$ cat fichier3
abcdef
ghijkl
toto
titi
$ cat fichier4
abcdef
ghijkl
toto
titi
$ rm fichier3
$ ls -il
total 2
```

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
65340 fichier2 65338 fichier3
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
65340 fichier2 65338 fichier3
$ mv fichier2 ..
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
65340 fichier2 65338 fichier3
$ mv fichier2 ..
$
```


Bases d'UNIX Gestion des fichiers et des répertoires Gestion Gestion des fichiers et des répertoires Gestion des droits

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
65340 fichier2 65338 fichier3
$ mv fichier2 ..
$ ls -i ../fichier2
```

Déplacement ou renommage des fichiers : **mv**

mv *fic1 fic2*

Déplacement ou renommage des fichiers En réalité, la commande **mv** n'effectue pas de copie de données physique mais modifie le nom et transfère les informations d'un répertoire à l'autre.

```
ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
65340 fichier2 65338 fichier3
$ mv fichier2 ..
$ ls -i ../fichier2
65340 ../fichier2
$
```


Copie de fichiers : **cp**

cp OPTIONS *fic1... fic2*

Duplique physiquement les données d'un fichier Si *fic1* est le fichier existant à copier

- si *fic2* n'existe pas, il est créé de façon identique à *fic1* ;
- si *fic2* existe et est un fichier, il est écrasé par le contenu de a ;
- si *fic2* est un répertoire, alors fic1 est dupliqué sous le même nom dans le répertoire *fic2*.

Si *fic1* est un répertoire existant à copier, il faut utiliser l'option -r

- si *fic2* n'existe pas, il est créé de façon identique à *fic1* (sous répertoires et sous-fichiers compris, récursivement) ;
- si *fic2* est un répertoire, alors *fic1* est dupliqué sous le même nom récursivement dans le répertoire *fic2*.

cp *liste-de-fichiers repertoire* permet de dupliquer les fichiers dans le répertoire.

Copie de fichiers : **cp**

```
ls -il
```


Copie de fichiers : **cp**

```
ls -il
total 0
65338 -rw-r--r-- 1 dntt users 0 20 fév 17 :58
fichier3
$
```


Copie de fichiers : **cp**

```
ls -il
total 0
65338 -rw-r--r-- 1 dntt users 0 20 fév 17 :58
fichier3
$ cp fichier3 fichier4
```


Copie de fichiers : **cp**

```
ls -il
total 0
65338 -rw-r--r-- 1 dntt users 0 20 fév 17 :58
fichier3
$ cp fichier3 fichier4
$
```


Copie de fichiers : **cp**

```
ls -il
total 0
65338 -rw-r--r-- 1 dntt users 0 20 fév 17 :58
fichier3
$ cp fichier3 fichier4
$ ls -il
```


Copie de fichiers : **cp**

```
ls -il
total 0
65338 -rw-r--r-- 1 dntt users 0 20 fév 17 :58
fichier3
$ cp fichier3 fichier4
$ ls -il
total 0
65338 -rw-r--r-- 1 dntt users 0 20 fév 17 :58
fichier3
65341 -rw-r--r-- 1 dntt users 0 20 fév 18 :01
fichier4
$
```


Recherche de fichiers : **find**

find *répertoire* OPTION

Rechercher des fichiers dans une hiérarchie de répertoires

-name *motif*	Fichier dont le nom de base (sans les répertoires du chemin d'accès), correspond au motif du shell
-perm *mode*	Fichier dont les autorisations d'accès sont fixées exactement au mode indiqué
-type *type*	Fichier du type donné (mode bloc b, caractère c, répertoire d, tube nommé p, régulier f, liens symbolique l, socket s)
-links *n*	Fichiers ayant n liens
-user *utilisateur*	fichier appartenant à l'utilisateur indiqué
-nouser	fichier n'appartenant à aucun utilisateur non numérique
-group *groupe*	fichier appartenant au groupe indiqué
-nogroup	fichier

-inum *n*	Fichier dont le numéro d'i-noeud est n
-newer *fichier*	Fichier modifié plus récemment que le fichier indiqué
-atime *n*	Dernier accès au fichier il y a n*24 heures
-mtime *n*	Fichier dont les données ont été modifiées il y a n*24 heures
-ctime *n*	Dernière modification du statut du fichier il y a n*24 heures
-print	Affiche le nom complet du fichier sur la sortie standard
-exec *cmd*	Exécute la commande jusqu'à ce qu'on rencontre un ';', La chaîne '' est remplacée par le nom du fichier en cours de traitement
-ok	comme -exec mais interroge d'abord l'utilisateur
-ls	affiche le nom du fichier en cours dans le format 'ls -dils

Recherche de fichiers : **find**

Exemple : Chercher dans l'arborescence `/usr` les répertoires dont le nom commence par un "chan" :

```
find /usr -name "local" -type d -print
```


Manipulation des noms de fichiers : **basename, dirname**

basename *nom_de_fichier*

La commande **basename** élimine les répertoires en tête du chemin d'accès du fichier.

```
$
```

dirname *nom_de_fichier*

La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

Manipulation des noms de fichiers : **basename, dirname**

basename *nom_de_fichier*

La commande **basename** élimine les répertoires en tête du chemin d'accès du fichier.

```
$ basename /usr/local/bin/lynx
```

dirname *nom_de_fichier*

La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

Manipulation des noms de fichiers : **basename, dirname**

basename *nom_de_fichier*

La commande **basename** élimine les répertoires en tête du chemin d'accès du fichier.

```
$ basename /usr/local/bin/lynx
lynx
$
```

dirname *nom_de_fichier*

La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

Manipulation des noms de fichiers : **basename, dirname**

basename *nom_de_fichier*

La commande **basename** élimine les répertoires en tête du chemin d'accès du fichier.

```
$ basename /usr/local/bin/lynx
lynx
$
```

dirname *nom_de_fichier*

La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

```
$
```


Manipulation des noms de fichiers : **basename, dirname**

basename *nom_de_fichier*

La commande **basename** élimine les répertoires en tête du chemin d'accès du fichier.

```
$ basename /usr/local/bin/lynx
lynx
$
```

dirname *nom_de_fichier*

La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

```
$ dirname /usr/local/bin/lynx
```


Manipulation des noms de fichiers : **basename, dirname**

basename *nom_de_fichier*

La commande **basename** élimine les répertoires en tête du chemin d'accès du fichier.

```
$ basename /usr/local/bin/lynx
lynx
$
```

dirname *nom_de_fichier*

La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

```
$ dirname /usr/local/bin/lynx
/usr/local/bin
$
```


Attributs d'un fichier

Résultat de **ls -ali**

```
718024 drwxr-xr-x   3  dntt  ens  4096  2006-01-24 21:13 .
2769   drwxr-xr-x  33  dntt  ens  4096  2006-01-24 20:29 ..
718024 -rwxr-xr-x   2  dntt  ens   231  2006-01-24 21:25 toto
---+--- ----+----- -+-  -+- --+--  --+- -----+--------  --+-
   |       |       |    |    |      |       |           |
   |    Les droits  |    | Le groupe |    La date de     |
   |                |    | auquel    |    derniere       |
   |                |    | appartient|   modification    |
   |                |    | le fichier|   du fichier      |
numéro              | La personne    |                   |
d'inode             |a qui appartient|                   |
                    | le fichier   la taille         Le nom
                    |              du fichier    du fichier
              nombre de liens
              sur le fichier
```

Droits d'un fichier

```
+--------------> Type de fichier, marque du repertoire
| +------------> Utilisateur  = Seul le proprietaire
| |              y a acces
| |   +--------> Groupe = les personnes du meme
| |   |          groupe y ont acces
| |   |   +----> Autres = tout le monde peut y acceder
| |   |   |
- --- --- ---
d rwx r-x --x
| |||
| ||+----------> si c'est 'x' on a le droit d'executer
| |+-----------> si c'est 'w' on a le droit d'ecrire
| +------------> si c'est 'r' on a le droit de lire
+--------------> type du fichier (-, d, l, s...)
```


Changement des droits d'un fichier : chmod

chmod *mode fichiers...*

Modifier les autorisations d'accès à un fichier. Le *mode* s'écrit en octal (777) ou en chaine (ugoa(+-=)rwxstX).

-R modifie récursivement les droits sur tout une arborescence

s : set-user-ID (suid)

t : sticky-bit

X : x seulement si répertoire.

Changement des attributs d'un fichier : chown, chgrp, touch

chown OPTIONS *user* :*groupe* fichiers...

Modifier le propriétaire et le groupe d'un fichier

chgrp OPTIONS *groupe* fichiers...

Modifier le groupe d'un fichier

touch *fichier*

Modifier l'horodatage d'un fichier

Liste des états d'un processus : ps

ps OPTIONS

Liste les états des processus

-A tous les processus

-l affichage des propriétés complète des processus

```
$
```


Liste des états d'un processus : ps

ps OPTIONS

Liste les états des processus

-A tous les processus

-l affichage des propriétés complète des processus

```
$ ps -A -l
```


Liste des états d'un processus : ps

ps OPTIONS

Liste les états des processus

-A tous les processus

-l affichage des propriétés complète des processus

```
$ ps -A -l
 UID PID PPID C SZ RSS PSR STIME TTY TIME CMD
dntt 3146 3145 0 412 540 0 23 :33 pts/4 00 :00 :00 -pdksh
dntt 3198 3146 0 622 868 0 23 :36 pts/4 00 :00 :00 ps -F
root 1839 1 0 592 932 0 19 :22? 00 :00 :00 /sbin/rpc.statd
root 1875 1 0 437 724 0 19 :22? 00 :00 :00 /usr/sbin/cron
root 1888 1 0 371 476 0 19 :22 tty1 00 :00 :00 /sbin/getty
38400
...
$
```


Liste des états d'un processus : top

top

Affiche les processus en cours et réalise l'actualisation au fur et à mesure

Lancer en arrière plan : &, bg

commande **&**

Lancer une commande en arrière plan

Suspendre un processus : (CTRL+Z)

bg

Met en arrière-plan le dernier processus suspendu dans ce shell

fg

Met en avant-plan le dernier processus suspendu dans ce shell

: nohup

nohup *commande*

Lance la commande avec ses arguments en ignorant le signal HANGUP (1) et redirige la sortie et l'erreur standard dans le fichier `nohup.out`
Le programme continue ainsi de tourner même si l'utilisateur se déconnecte. il est associé

```
$
```


: nohup

nohup *commande*

Lance la commande avec ses arguments en ignorant le signal HANGUP (1) et redirige la sortie et l'erreur standard dans le fichier `nohup.out`
Le programme continue ainsi de tourner même si l'utilisateur se déconnecte. il est associé

```
$ nohup ./long_programme &
```


: nohup

nohup *commande*

Lance la commande avec ses arguments en ignorant le signal HANGUP (1) et redirige la sortie et l'erreur standard dans le fichier `nohup.out`
Le programme continue ainsi de tourner même si l'utilisateur se déconnecte. il est associé

```
$ nohup ./long_programme &
appending output to nohup.out
$
```


Suspension d'exécution : **sleep**

sleep *secondes*

Suspend l'exécution durant un certain intervalle de temps exprimé en secondes

Exécution de commandes sans duplication de processus : **exec**

exec *commande*

Il n'y a pas de création de processus pour exécuter la commande.

Envoi de signal : **kill**

kill *-signal pid*

Envoyer un signal à un processus de pid donné

Les signaux sont générés par des évènements lancés par l'utilisateur lors de l'exécution du shell, par exemple :

- 1 : coupure de ligne
- 2 : arrêt (CTRL+C)
- 9 : destruction (NON INTERCEPTABLE)
- 15 : fin de process