TD1

Se familiariser avec les commandes linux

Objectif : Le but de cette partie est la prise en main des commandes de base de l'environnement Linux.

Enoncé :

1. Qu'est-ce que le shell ?

- interpreteur de cmds
- interface entre S.E. et user (utilisateur)

2. Identifier les répertoires suivants : (3 dossiers spécifiques)

. : répertoire (dossier) courant (déjà affiché)

.. : répertoire parent

~ : répertoire home

3. Compléter le tableau

dossier ≡ répertoire

Commande	A quoi sert	exemple
cd	Change le répertoire courant	cd .. cd ~
ls	afficher le contenu d'un rép.	ls (affiche contenu du rép. courant) ls ..
cp	Copier un ou plusieurs fichier	cp file1 file2 cp ../f1 ./rep1/f1
rm	Efface un ou plusieurs fichiers	rm file ./rep1/f1 rm -r ./rep1
mkdir	Créer un répertoire (vide)	mkdir rep1
rmdir	supprimer un rép vide	rmdir rep1
mv	Déplace ou renomme un fichier	mv f1 f2 mv ./f1 ./rep1/f1
cat	affiche contenu d'un fichier (afficher les données)	cat f1 cat f1 f2
man	(affiche help d'une cmd) créer un fichier vide	man ls man cat
touch	man : affiche help d'une cmd créer un fichier vide	touch ./f1 touch ./f2 ./rep/f3

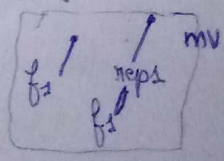
[ls ≡ ls .]

.. ← père

f1 (un fichier) rep1

rm f1 (supprimer tout les fichiers de rep1)

mkdir ; exp : mkdir rep1 rep2 rep3



4. Afficher le listing d'un repertoire en couleurs

ls --color

ls -a

Scanné avec CamScanner

5. Afficher aussi les fichiers cachés (commençant par un point) :

ls -a

6. Afficher le listing détaillé

ls -l

-: fichier ordinaire = fichier de données | l: liens | b: périph mode bloc
d: directory (rép/dossier) | c: périph mode caract

Pour afficher les fichiers d'un répertoire en triant sur la date de mise à jour des fichiers

rwx | rwx | rwx → other
user | droits d'accès | groupe

7. Afficher les fichiers les plus récents en premier

ls -lt // le plus récents en premier (last)

8. Afficher les fichiers les plus vieux en premier

ls -ltr (last reverse)

Dans les systèmes d'exploitation dérives d'Unix, le codage des droits se fait sur 9 bits groupes par 3 bits. Ces droits sont codes en un entier. Pour ce faire, on convient de la correspondance : r = 4; w = 2 et x = 1.

9. A quels droits correspondent les entiers 751; 521; 214 et 150 ?

751 : rwx r-x --x ; 521 : r-x -w- --x ; 214 : -w- --x r--

150 : --x r-x ---

751 : rwx r-x --x | 214 : -w- --x r--

521 : r-x -w- --x | 150 : --x r-x ---

10. Par quels entiers sont codes les droits rw-r- -r- - et rwxr-xr-x ?

(110)(100)(100) 644 / 755

La commande `ls -l toto.sh` affiche :

```
-r-xr--r-- 1   john    staff   128     18 mai 11:56 toto.sh
```

11. À qui appartient ce fichier ?

toto.sh appartient à john (user)

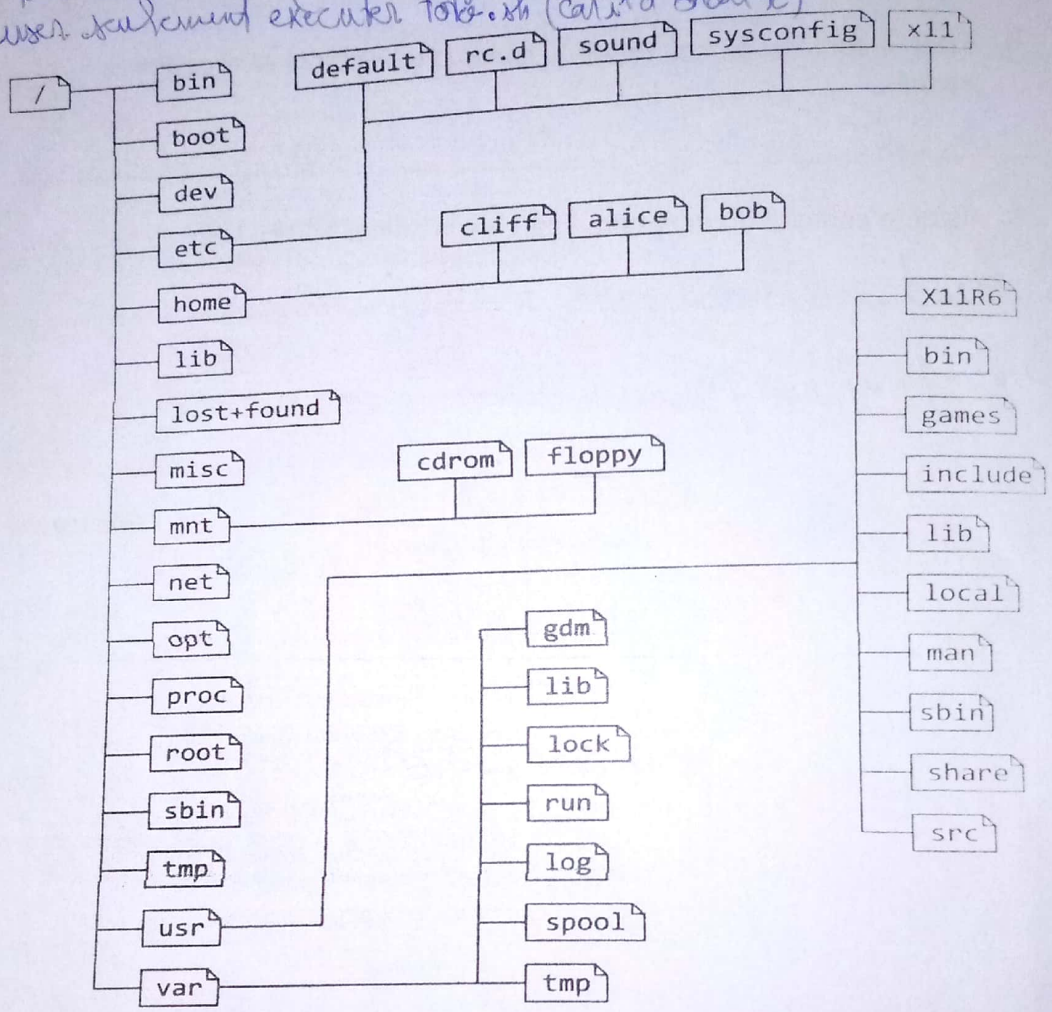
12. Qui peut écrire dans le fichier ?

personne (w manquent pr user, groupe et other)

Scanné avec CamScanner

13. Qui peut le lire ? Qui peut l'exécuter ?

tout le monde (user se groupe + ~~tous~~ others ont droit "r")
user seulement executer toto.sh (car il a droit x)



14. Proposer une commande qui permette de se déplacer du répertoire HOME de Alice à celui de Bob :

a. en utilisant un chemin relatif ;

chemin relatif : cd ../../bob
(../../ = revenir en arrière 2 fois répertoire courant)

b. en utilisant un chemin absolu.

chemin absolu : cd /home/bob

15. Alice est à la racine /. Proposer trois commandes qui peuvent lui permettre de se déplace dans sont répertoire d'accueil (HOME).

cd /home/alice
cd ./home/alice
cd ~ (tilde)

16. Bob est dans son HOME. Aidez-le à :

Scanné avec CamScanner

/bob/home/bob

a. lister le contenu de son HOME ;

ls

b. lister le contenu de son HOME y compris les fichiers et répertoires cachés.

ls -a

c. lister le contenu du répertoire share sans quitter son HOME.

ls /user/share (absolu)

ls ./../user/share (relatif

Bon travail

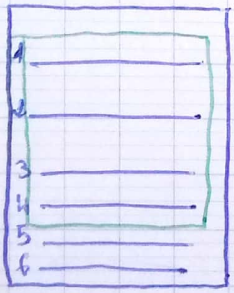
Scanné avec CamScanner

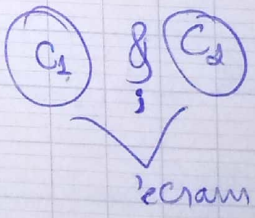
Les filtres : find, grep, tail, head, cut, sort

find → forme
grep → extraction des lignes selon motif
tail, head → lignes
head : à partir du début
tail : à partir de la fin
→ contenu

Syntaxe : head -numéro fichier
tail
numéro = nbre de ligne

Exp :
extraire les 5 premières lignes d'un fichier F
↳ head -5 F
extraire les 2 dernières lignes de F
↳ tail -2 F
extraire 3ème et 4ème ligne de F
↳ head -4 F | tail -2 F = head -4 F > test
tail -2





Extraire 2ème et 4ème ligne de F
head -4 F | tail -1 F ; head -2 F | tail -1 F

nom ␣ prénom ␣ poids ␣ tel
cut : extraction selon une colonne

Syntaxe :
cut : -c (caractère / numéro) → numéro de caractère en question (virtuelle)
-b (byte)

Scanné avec CamScanner

cut -c3 F : extraire 3ème caractère
cut -c1-3
↳ tiret 6
-c1,2,3 F
cut -c3-5 F : du 3ème au 5ème caract
cut -c3- F : du 3ème à la fin de ligne
cut -d' ' -f numéro fichier
délimiteur field

test

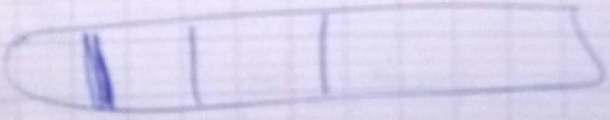
nom	Prenom	id	abc
Somaa	fdvmn	70	9993
Riahi	ali	21	4653
Rjab	Safa	33	1275
Sahab	med	22	6791

S cut -d' ' -f2 test
↳ Prenom
fdvmn
ali
Safa
med

cut '-d␣' -f2,4 F
↳ afficher 2ème et 4ème champs

cut -d' ' -f-4 F
↳ de 1ère jusqu'au 4ème champs

sort : permet de faire tri
sort test : trier contenu du test selon ordre alphabétique



T[0] = @T + 0*10 (ax0
T[1] = @T + déplacement

Scanné avec CamScanner

cut -c3 F : extraire 3ème caractère

cut -c1-3 ← taire 6

-c1,2,3 F

cut -c3-5 F : du 3ème au 5ème caract

cut -c3- F : du 3ème à la fin de ligne

cut -d' ' -f numero fichier.

déliminer field

test

nom	Prenom	id	dbc
Somaa	fatma	70	9999
Riahi	ali	21	4653
Rjab	Safa	33	1295
Salah	med	22	6791

$ cut -d' ' -f2 test

retourner

Prenom
fatma
ali
Safa
med

cut '-d ' -f2,4 F

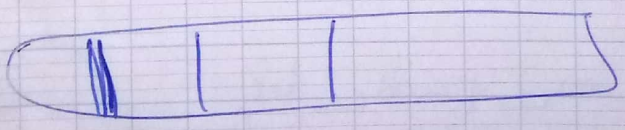
2ème et 4ème champs

cut -d' ' -f2-4 F

du 2ème au 4ème champs

sort : permet de faire tri

sort test : trier contenu du test selon ordre alphabetique



T[0] = @T + 0^10 case 0

T[1] = @T + deplacement

②

TD2 les filtres

Commande « grep»

Exercice 1

1. Nous devons créer dans notre répertoire courant le répertoire TD-grep.

mkdir TD-grep) création du répertoire TD-grep

2. Dans ce répertoire nous devons créer les fichiers suivants:

tomate poire pomme cerise Fraise fraise courgette POMME3 afraise

Méthode 1 :

M1: cd TD-grep

Me: touch poire pomme cerise fraise courgette POMME3 afraise

Méthode 2 :

touch TD-grep/poire TD-grep/pomme

find . type f -name "[fF]raise"

3. Comment peut-on afficher que les fichiers qui se trouvent dans TD-grep dont les critères sont :

Critère	réponse	
	Avec find	Avec grep
Le nom de fichier doit être Fraise ou fraise	find type f -name "*se" find . -type f -name "[fF]raise"	ls | grep -E '([fF])raise$'
« se » est en fin de nom	find type f -name "*se"	ls | grep -E '*se$'
« ai » est présent dans le nom	find . -type f -name "*ai*"	ls | grep 'ai' *ai* à essayer
Nom contenant un chiffre numérique	find . type f -name "*[0-9]*"	ls | grep '[0-9]'
Nom contenant la chaîne mm ou MM	find . type f -name "*MM*" -O -name "*mm*"	ls | grep -i 'mm'

Scanné avec CamScanner

mkdir : création d'un rép vide

Exercice 2

1. Créer un fichier resume

touch resume) création de fichier "resume" vide

2. Metttre dans ce fichier

Bonjour mes collegues GCR1

Nous allons tester la commande grep par plusieurs options

Ce test envisage la maitraise et la comprehension de cette commande

Esperant qu'il soit un test benefique pour plusieurs

~~A ce stade nous connaissons grep et find~~

Au revoir mes collegues (presque 55 etudiants)

au prochain TD

3. Comment faire apparaître le numéro de la ligne où figure le mot « test » dans fichier « resume » ?

grep -n "test" resume

grep -c "test" resume → le nbre de ligne contenant le motif "test"

4. Comment faire pour afficher le nombre d'occurrences du mot « test » ?

grep -c "test" resume : nbre de lignes contenant "test"

→ nbre d'occurence il faut faire un script

5. Chercher toutes les lignes commençant par «a» ou «A».

grep "^[aA]" resume

6. Chercher toutes les lignes finissant par «rs»

grep "rs$" resume

~~[rs]$~~

Scanné avec CamScanner

7. Chercher toutes les lignes contenant au moins un chiffre

grep [0-9] nexsume

8. Chercher toutes les lignes commençant par une majuscule

grep ^[A-Z] nexsume

9. Combien de lignes commençant par une majuscule

grep -c ^[A-Z] nexsume

Bon travail

afficher les lignes qui ne contient pas le mot test :

grep -v "test" nexsume

Scanné avec CamScanner

Scanné avec CamScanner

-1- cd : déplacement entre les rép
cat : affichage du contenu d'un fichier

sol 1

```
echo "ligne 1" >> resume
echo "ligne 2" >> resume
echo "dernière ligne" >> resume
```

```
echo "ligne 1
ligne 2
dernier ligne" >> resume
```

```
pour        echo -e "bjr \n bsr \n Hello"

bjr
 bsr
 Hello
```

éditeur de txt
- textuelle
  - éditeur de terminal
    - vi, nano] cmd interne à utiliser ajout text "i"
      - ↳ insert
      - echap
      - :wq) sauvegarder et quitter
- graphique

exp : vi nom_fichier
vi resume

affiche la ligne d'un motif

graphique
gedit nom_fichier

man grep → affiche ("help") utilise avec "grep"

_3_ grep -n "test" resume ↓

1: test
_4_ 4: plusieurs test
5: premier test
6: dernière test test & Test 3

grep -c "test" resume ↓

4

"^test $"
^ début ligne
$ fin

n'importe quelle caractère
"test " → se trouve
" test " mais il faut trouver
se trouve seulement le mot "test"

_5_ grep "^[aA]" resume
cherche tous les lignes commençant par "a" ou "A"

grep "[^aA]" resume
cherche tous les lignes qui ne commence pas par "a" ou "A"

[^A-Z] ← négation d'un caractère
ecrit tous les ligne qui ne commence pas par un caractère majuscule

test
test 1
test B
il n'y a pas de A ou a

Grep :

Recherche dans un contenu

egrep : grep expression reguliere

fgrep : grep sur les fichiers

grep opt* motif [argument]+

$[^[*a*][a-z]^+]^*$

nomfichier > f1

ls > f1

grep '[Ff]raise' f1

fraise

fraise

fraise

grep '^[Ff]raise'

touch fraiseaa

ls > f1

grep '^[Ff]raise$' f1

frais

fraise

fraise fraise

fraiseaa

grep '^Fraise'

Scanné avec CamScanner

TD les filtres

Commande « find »

1. Cherchez un fichier dans le répertoire racine qui contient le terme "net".

find / -type f -name "*net*"

2. Cherchez tous les fichiers dont les noms commencent par une majuscule suivie d'une autre lettre, et se terminent par un chiffre entre 3 et 6

-name "[Aa]*[3-6]"

find / -type f -name "a*" -o -name A* -a -name "*[3-6]"

3. Chercher les fichiers qui ont été modifiés il y a plus de 30 jours ? Il y a 30 jours ? Il y a moins de 30 jours?

find / -type f -mtime +30) Il y'a plus de 30 j

find / -type f -mtime 30

find / -type f -mtime -30) Il y'a au plus de 30 j

4. Chercher le fichier a été modifié plus récemment qu'un autre fichier donné ?

find / -type f -newer "f1.txt" fichier donné

5. Cherchons les fichiers modifiés plus récemment que bofichier.tex, mais il y a plus de 5 jours

find / -type f -newer "bofichier.txt" -mtime +5

6. On recherche un fichier modifié il y a moins de 12 jours et dont la taille soit supérieure à 30 K

find / -type f -mtime -12 -size +30K

7. Chercher dans /usr les fichiers dont la taille dépasse 1Mo et dont les droits sont fixés à 755

find /usr -type f -size +1M +1024KB -perm 755

8. Cherchez dans toute l'arborescence les fichiers dont le nom se termine par .c, redirigez les erreurs vers le fichier poubelle /dev/null

find / -type f -name "*.c" 2> /dev/null

9. Trouvez tous les fichiers modifiés entre 10 et 50 jours auparavant.

find / -type f -mtime [10-50]

-mtime -50 -mtime +10

10. Trouver des fichiers qui ont une taille > 10 Mo et < 40 Mo.

find / -type f -size +10*1024*1024+1 (40*1024*1024)-1

Bon travail

Scanné avec CamScanner

find rep expression

critères action

/user : les elts, dont le nom commence par 'a' et se termine par 'a'

int1 int2 int3
-a

find /usr -name "a*" -name "*a"
find /usr -name "-x" -a -name "*a"
-name "a*a"

les elts qui commencent par a ou se terminent par a.

find /usr -name "a*" -o -name "*a"

-o ou
-a et

ou
a*a

a/A
des caract
chiffre 3-6

[3-6]

-name "[a-A]*[3-6]

[aA]
[a,A]
[A,A]

[file]

-name "[Aa]*[3-6]"

find / -type f -mmin "f1.txt"

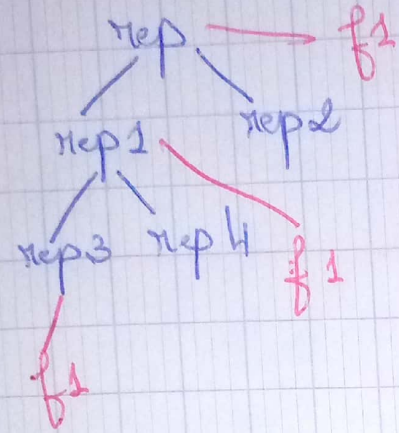
fichier donnée

Scanné avec CamScanner

CHAPITRE 2



tube
tube nommé



$ find rep -name "f1"

rep/f1
rep/rep1/f1
rep/rep1/rep3/f1

find rep + ← un ou plusieurs expression

options
critères

action*
- print : affichage
- exec cmd + exécution

exp: supprimer les fichiers ayant le nom "f1"
find rep -name "f1" -exec rm {} ;

cmd1/cmd2 f+ cmd1 cmd2

tube anonyme entre 2 commandes

* : 0 ou plusieurs instance

\+ : 1 ou " "

. : 1

[ ] : intervalle / [a-z] _name f[a-z]

[a-dx-z] [acbf]

[a-zA-Z]

{ }

جاهز يزي

commence par

^[a-zA-Z][0-9]*

^[a-zA-Z[0-9]]

a19

a1

^[a-zA-Z[0-9]*]+

plusieurs >< 1

^[a-zA-Z[0-9]*]*

"f*"

fite

f

f

f1

"f+"

fik

f1

fe

f

car f+ = f

"f?" ← seulement 1 caract

CHAPITRE 2

LES FILTRES

Docteur Ingénieur : Fatma SOMAA

AU: 2022-2023

RECHERCHE D'UN FICHIER: FIND

find [répertoire]+ [expression]+

- **find** descend récursivement des sous-hiérarchies de répertoires données par leur racine, en cherchant les fichiers répndant à certaines critères (nom,droit,type, taille..).
- **Répertoire:** la liste des racines des sous-hiérarchies à explorer
- **Expression:** suite de *primitives* exprimant à la fois les critères de sélection des fichiers et les actions à leur appliquer

Scanné avec CamScanner

EXPRESSION

-name <fic> recherche le fichier **fic**

-user <nom> recherche un fichier appartenat à l'utilisateur **nom**;

-atime <n> recherche un fichier utilisé dans les n derniers jours;

-ntime <n> recherche un fichier modifié dans les **n** derniers jours;

-newer <fic> recherche un fichier modifié plus récemment que **fic**;

-type <x> recherche un fichier de type <x> avec :
- **b** pour fichier spécial bloc,
- **c** pour fichier spécial caractère,
- **d** pour fichier répertoire,
- **f** pour fichier ordinaire,
- **p** pour tube nommé (System III/V);

-perm <octal> recherche un fichier ayant les permissions données;

- links <n> recherche un fichier ayant **n** liens ;

-group <nom> recherche sur un groupe **nom**;

EXPRESSION

- **- size <nc|n>** recherche sur une taille en blocs ou en caractères;
- **- exec commande** vrai si la commande restitue une valeur de sortie nulle. la fin de la commande doit être ponctuée par **\;** et le nom du fichier examiné est représenté par **{}**;
- **- print** le nom de fichier doit être imprimé;
- **- inum <n>**recherche un fichier dont le i- noeud **n**;
- **-\(expression\)** a pour valeur booléenne celle de l'expression parenthésée.
- Toutes ces primitives peuvent être composées avec: **!**, **-o** , et **\(\)**.

Scanné avec CamScanner

EXEMPLES

```
find /usr/ens \( -name *.p -o -name *.f \) -mtime
  -30 -exec ls -i {} \;
2135 /usr/ens/mhs/exam.p
2149 /usr/ens/mhs/scarable.f
2149 /usr/ens/rim/exp.p
2149 /usr/ens/rim/examen.p
find $HOME -newer .datesauv -print
affiche les fichiers de l'arborescence definie
  par  le répertoire personnel de l'usage
  modifiés  après le fichier .datesauv
```


RECHERCHE D'UNE CHAÎNE DANS UN FICHIER: GREP

- **grep**, **egrep** et **fgrep** lisent « ligne par ligne » et affichent les lignes de fichiers donnés en paramètre et qui contiennent un motif donné.

grep [option] <motif> [<fichier> ...]

- **motif** est une expression régulière dans laquelle certains caractères ont une signification particulière.
- Sauf option contraire, chaque ligne ayant cette propriété est écrite sur la sortie standard.

Scanné avec CamScanner

OPTIONS

- **-v** seules les lignes ne contenant pas le motif sont affichées;
- **-v** seules les lignes contenant pas le motif sont affichées;
- **-c** seul le nombre de lignes satisfaisantes est affiché;
- **-i** ignore la distinction minuscule/majuscule (**grep**);
- **-l** seules les noms de fichiers contenant le motif sont affichés;
- **-n** numérote les lignes affichées;
- **-e motif** lorsque le motif commence par un tiret;
- **-f fichier** le motif est dans le fichier (**egrep** et **fgrep**)

```
$ grep -n variable *.ch  cherche variable dans les source
$ grep From $MAIL affiche les messages contenant From
$ grep From $MAIL| grep -v marie    ... autre que marie
$ grep -c marie $HOME/lib/annuaire trouve le n° de marie
$ who | grep  marie  est-elle connectée?
$ ls | grep -v temp  tous les fichiers sauf temp
```


EXPRESSIONS RÉGULIÈRES DE GREP

- c tout caractère non spécial
- \c ignore la signification spéciale de c
- ^ début de ligne
- **$** fin de ligne
- . n'importe quel caractère unique
- **[...]** n'importe quel caractère unique parmi ....;
- **[^...]** n'importe quel caractère unique ne figurant pas parmi ....
- **\n** ce qui a été identifié par la nième sous expression \(...\)
- **r*** zéro ou plus occurrences de r
- **r+** une ou plus occurrences de r
- **r?** zéro ou une occurrence de r
- **r1r2** r1 suivi de r2
- **r1| r2** r1 ou r2 (**egrep** seulement)
- **\(r \)** sous expression r repérée (**grep** seulement); voir \n;
- **(r)** expression régulière r (**egrep** seulement); peut être imbriqué

Scanné avec CamScanner

EXEMPLES

```
$ grep '^From' $MAIL
$ ls -l | grep '^d'
$ ls -l | grep '^.......rw'
$ grep '^[^:]*::' /etc/passwd
```

- **fgrep** recherche plusieurs expressions simultanément
- **egrep** traite des expressions régulières avec '**ou**' et des () pour grouper les expressions.

TRI

sort [options] [clé [option]]* [<fichier1>]*

- trie les lignes et écrit les résultats sur la sortie standard (sauf -o).

options

- -c (*check*) vérifie si le fichier est dèjà trié;
- -u (*unique*) conserve un seul exemplaires des lignes identiques;
- -o <fichier> (*output*) envoie la sortie sur fichier
- -d (*dictionary*) ordre lexicographique (lettres, les chiffres, blancs)
- -f (*fold*) met les majuscules en minuscules;
- -n clé numérique (signée éventuellement);
- -i les caractères dont le code ASCII 040-0176;
- -rtri en ordre inversé;
- -b ignorer les blancs en début de champ;
- -t<x> le délimiteur de champ dans les lignes est le caractère x.

Scanné avec CamScanner

NOTION DE CLÉ

- **clé** de la forme **+debut - fin**.

debut premier champ servant à faire la clé (numérotation à partir de zéro)

fin n° du premier champ qui n'appartient pas à la clé.

```
sort +1 -2 f        trie f sur une clé formaté
  par le                    deuxième champ de
  chaque ligne
```

- Une clé peut être préciser des portions de champs.

debut et **fin** sont de la forme **m.n** (éventuellement l'une des options **b d f i n** ou **r**) où:

m nombre de champs à sauter depuis le début de ligne

n nombre de caractères à sauter en début de champ .

EXEMPLES

```
ls | sort  trie les noms de fichiers en ordre
  alphanumérique
$ ls -s | sort -nr   trie par ordre de taille
  décroissante
$ ls -l | sort +4nr -5n  trie par taille  décroissant;
  après avoir sauter les 3 premiers champs
$ who  | sort +4n -5ntrie les plus anciennes connexions
$ sort +2.3 -5.0 f   trie f sur une clé qui commence au
  troisième champ (4ème  caractère), et qui se termine
  à  la fin du cinquième champ.
Soit f.t un fichier d'étudiants qui la format suivant:
nom, prénom, jj/mm/aa, codeBac, codeDiplôme
$ sort -t, +4bf -5bf +0f -1f f.t >diplome.t
tri sur diplôme, puis alphabétiquement, sortie   sur
  diplome.t
$ sort -t, +2.6b -2.8b +2.3b -2.5b +2.0b -2.2b f.t
  >age.t
tri par âge, la date étant la date de naissance, sortie
```


Scanné avec CamScanner