MEMENTO DES COMMANDES ESSENTIELLES ADMINSTRATEUR UNIX

1 Préliminaires sur ce document

1. Tout ce qui est en **gras** dans ce document, correspond à des commandes, des options de commandes, des fichiers standards sous Unix Solaris, et à des mots clés importants et tous les mots réservés UNIX.
2. Dans la suite, nous remplacerons souvent l'appellation " Administrateur système " par l'appellation " **root** " (celui qui a tous les droits sous Unix).

2 Qu'est ce que Solaris ?

Solaris est un OS (Operating system / Système d'exploitation) + une interface graphique Open Windows (CDE) + des Progiciels réseau (ONC+).
Note : sur une machine donnée, peut coexister les 2 types d'environnement graphique :
CDE et Open Windows.

3 Qu'est ce qu'un OS ?

-> Contrôle des terminaux, lignes de communication, imprimantes, disques dur, I/O (entrées/sorties).
-> Noyau (**Kernel**) : partage des utilisateurs, applications => gère les ressources des utilisateurs,
-> partage des fichiers (UFS, VxFS ...)

4 Qu'est ce que le Shell ?

Le Shell est un langage de commande du système UNIX (Job control langage - JCL) :
➔ dialogue entre utilisateur et l'OS (le système)

5 Différents types de shell

Bourne : /bin/sh (prompt : $), C-shell : /bin/csh (prompt : %), Korn shell : /bin/ksh (prompt : $)

-> # forme de shell : **sh**, **ksh**, **csh** -> **perl** (sh, grep, sed, awk) -> utilisé majortairement pour application Internet,
-> **dtksh** (desktop Korn shell), pour environnement graphique Motif
(situé dans répertoire **/usr/det/bin/dtksh** , voir en fin de ce cours).

Attention ! sous Unix, on doit distinguer les majuscules et les minuscules, dans les commandes contrairement au MS-DOS (par ex. le mot avec " I " majuscule dans cet exemple " Init " n'est pas équivalent au mot avec " i " minuscule dans cet autre exemple " init ").

CDE :

CDE (Common Desktop Environment) est l'environnement graphique de Solaris (environnement X-Windows).

6 Informations de base sur le Noyau Solaris

2 répertoires importants:
1) /platform/`**uname - i**`/**kernel**
2) /kernel

7 Commandes pour avoir des informations sur le noyau

uname -m : avoir la configuration hardware.
uname - i : avoir le nom de la machine (est souvent aussi le nom réseau).

Note : **~** : répertoire d'accueil (celui dans lequel on arrive quand on se connecte). Exemple : **ls –l ~**

8 Fichiers spéciaux Unix (pour les Entrées / sorties …)

a) pour Disques Durs, CD : **/dev/dsk**/... et **/dev/rdsk/** ... (raw, brut ...).
b) écrans : **/dev/rdsk/** ...
c) console système : **/dev/console** et **/dev/term/** ...
d) disquette : **/dev/diskette**
e) bandes (DAT ...) : /**dev/rmt/** ....
f) imprimantes : **/dev/bpp0/** ...
g) clavier : **/dev/kbd/** ...
h) fichier null : **/dev/null**

9 Démarrage de la machine (démarrage système)

Langage Forth, open boot : "**OK boot**", "**b**" :
a) Pour booter sur le CD : **ok boot cdrom**
b) sur le disque : **ok boot disk1**
c) sur le réseau : **ok boot net**

10 Arrêt machine (arrêt système)

-g : délais de grâce, -i : "niveau d'init"

arrêt immédiat : **init 0**, **shutdown - i 0 g300** ou encore **halt** (on force l'arrêt de toutes les applications)
arrêt avec coupure de courant : **init 5**, **shutdown -i 5 -g300** ou encore **poweroff** (on attend l'arrêt de ces applications)
reboot : **init 6**, **shutdown -i 6 -g300** ou encore **reboot**
shutdown pour être en mono-utilisateur : **init S**, **shutdown**

La commande shutdown est dans /usr/sbin

Fin de cession : **Ctl-D** ou **exit** ou **logout** (pour le csh).

Doc Unix en ligne sur la machine (si elle est installée) :

Recherche d'une doc sur une commande : 1) **man - s** section nom_cde
2) recherche par mot clé :
a) il faut d'abord installer cette fonctionnalité par : a) **install catman -w**
b) la recherche par mot clé : **man -k** mot_clef

En résumé : Unix contient des programmes, des utilitaires, des applications, des outils de formatages de doc (**nroff**, **troff** ...),
des outils de communications entre utilisateurs, des éditeurs (**ed**, **vi**, voire "**emacs**" ...), des bases de donnée.
Mais à la base, il ne comprend pas un traitement de texte.

11 Commandes shell de bases

1) **ls** :
-a : tous les fichiers (y compris cachés ...)
-F : type du fichier
-R : liste tous les fichiers, les répertoires, avec leurs fichiers et sous répertoires (affichage récurcif),
-l: toutes les informations possibles sur les fichiers (type, permission, si le fichier est de type lien _ voir plus loin _,
son propriétaire _ owner _, son groupe _ voir plus loin _, sa taille en octect, sa date de création ou mise à jour,
et le nom du fichier.
-d : liste uniquement les répertoires

12 Les méta-caractères utilisés dans les commandes shell

* : tous les caractères quelconques sauf le "." (comme sous DOS),
? : un caractère (quelconque, mais un seul caractère)
[] : ex [a-z] : liste tous les caractères de la liste entre crochets droits, ici de a à z,
[!] : ex. : [!a-z] : on exclut une liste de caractères (ici on exclut les caractères de a à z).

méta-caractères utilisés pour l'exécution de commandes :

& : force l'exécution, en arrière-plan (background) de la commande (dont elle termine la syntaxe). ex. :
cmde & grep "toto" * > titi &

\ : neutralise la prise en compte de caractères spéciaux dans la commande.
`cde`: remplace par la sortie (résultat) de cette commande,
'...' : "littéralise" la chaîne sauf les 2 quotes " ' ".
" ... ": idem sauf pour les méta-caractères suivants : $, `, ', \, !

12.1 La commande find

find rep **– name** nom_fic **–exec** cmde \ ;

Exemple : recherché de tous les scripts shell (terminés par « sh ») contenant la chaîne de caractère « *orale* » (recherche commençant à partir du répertoire courant « . ») :

find . –name "*sh" –exec grep -l "*orale*" {} \;

stdin → sh → stdout (stdout : 1> 1>>)
└——→ stderr (stderr : 2> 2>>)

et 2> &1 →sortie des stdout et stderr dans le même fichier.

12.2 Nettoyer et/ou purger un file system

Il faut supprimer tous les fichiers inutiles (les fichiers dump « core », les fichiers dans le répertoire « lost+found », les fichiers toto …) et purger les fichiers trop gros (fichiers *.log, *.tmp …) :

cd /*répertoire_racine_de_la_recherche*

A) supprimer les fichiers inutiles :

find . \(-name core –o –name toto \) –exec rm –f {} \ ; -print
cd lost+found ; pwd ; rm –f *

A) commande « find » pour rechercher de gros fichiers :

1) on recherche tous les fichiers de plus de 10000000 octets :

find . -type f -size +10000000c

2) on recherche tous les fichiers de plus de 10000000 octets

find . -type f -size +10000000c

on recherche de tous les fichiers log de plus de 10000000 octets, et on les supprime :

find . -name "*log" -type f -exec rm {} \;

B) diminuer la taille d'un fichier log :

tail -1000 fic.log > /tmp/toto

compress fic.log

mv /tmp/toto fic.log

12.3 Commandes " pipe " :

|, ||, &&

commande1 **|** commande2 : le résultat de la commande1 est passé à la commande2.
commande1 **||** commande2 : même si la commande1 est en échec, on exécute la commande2.
commande1 **&&** commande2 : on ne peut exécuter la commande2 que si la commande2 est réussie.
(*code de contrôle* de la commande1 = **0** _ i.e. " **exit 0** ")).

Note : tous le composants d'un *pipe* doivent être des commandes exécutables.

tee : syntaxe : cmde **| tee** [**–a**] fic : **tee** permet à ce que la sortie **sysout** d'une commande aille à l'écran et en même temps dans le fichier « fic » (note : en ajout si l'option « **-a** » est précisée).

12.4 Autres commandes importantes

echo texte : affiche à l'écran le texte.
echo $Variable_shell : affiche à l'écran le contenu de la variable shell.

cd [rep] : changement de répertoire.
" **cd** " sans nom de répertoire précisé : retour au répertoire d'accueil.

cat fic : affiche contenu d'un fichier (texte).
more fic : idem
rm : suppression de fichier(s) (remove),
rm –i fic : on interroge avant chaque suppression.
rm –f fic : on force la suppression.
touch fic : création d'un fichier vide, ou mise à jour à la date système actuelle d'un fichier existant.
mv fic fic_cible : déplacement (ou renomage) de fichier.
cp fic [fic2 …] fic_cible : copie de fichier.
cp – p :
cp –r :
ln : ln fic fic_link ou ln –s fic fic_link

cmp fic fic2 : compare 2 fichiers (texte) ligne par ligne.
diff fic fic2 : affiche différences entre 2 fichiers (texte) ligne par ligne.
diff – c : affiche 3 lignes,
diff –e : pour executer sed ou ed
diff –n : affichage ordre inversé et n° de ligne ;

last : affiche les dernières connexion de l'écran.

compress fic : outil de compression de fichier
uncompress fic : commande inverse de " compress "
(**tar c** fic_tar fic : commande pour concaténer des fichiers dans fichier de type " tar ", voir plus loin).

wc fic : compte de mots (word count)
head [**-***N*] fic : affiche les *N* 1^{ières} lignes d'un fichier (texte)
(Note : si l'option « **–***N »* n'est pas précisée => affiche les 10 1^{ières} lignes).
tail [**-***N*] fic : affiche les *N* dernières lignes d'un fichier (texte)
(Note : si l'option « **–***N »* n'est pas précisée => affiche les 10 dernières lignes).
cut –f*N* **–d**'*C*' fic (ou encore : cmd **| cut –f***N* **–d**'*C*') : sélectionne le *N*° champs selon le caractère séparateur '*C*'. Exemples :
lognam**=`id | cut -d'**(**' -f2 | cut -d'**)**' -f1`**
uustat | cut -d" " -f1
Note : Avec la syntaxe **–f***N***-** indique qu'on sélectionne tous les champs à partir du *N*° champ
Avec la syntaxe **–f-***N* indique qu'on sélectionne tous les champs jusqu'au *N*° champ.

od fic : affiche contenu d'un fichier, **od –o** : en octal, **od –c** : affichage des caractères affichables,
od –x : en hexadécimal,
split [-n] [-an] fic [fic_cible]
Note : si fic_cible est omis, créé des fichiers portant comme nom " xaa ", " xab " etc ..
Saucissonne un fichier en x fichiers de " n " ligne chacun.
paste fic1 fic2 : fusionne les fichiers fic1 et fic2, ligne par ligne.
tr [-n]|[-an] 'car1' 'car2' < fic_src > fic_cible : " traduit " (remplace) les occurrences d'un caractère donné par un autre dans un fichier
ex : tr '[a-z]' '[A-Z]' < fic_in > fic_out

sort [-r] [-o fic_out] [**-t** car] [**-k** num_champ[n]] fic : tri des lignes d'un fichier sur des critères de tri donné.
avec : . car : caractère séparateur des champs (à trier)
. num_champ : numéro du champ
exemple : **sort –k** 2 fic : tri sur le 2ième champ,
sort –n fic : tri sur champ numérique (?)
sort –u fic : ramène des lignes uniques (ramène une occurrence unique de chacune ligne trouvée dans le fichier. Voir aussi la commande " uniq ").
uniq [-udc] fic : tri sur un fichier ne ramenant qu'une occurrence unique d'une ligne donnée dans le fichier à trier.
uniq –u : ramène, en résultat du tri, des lignes unique (en cas de lignes identiques dans le fichier),
uniq –d : liste que les lignes dupliquées (et chacune en un seul exemplaire),
uniq –c : liste toutes les lignes + nombre d'occurrence(s) de la ligne affichée.

whereis cde : arborescence de la commande (affiche toutes les occurrences de la commande avec leur arborescence),
which cde : arborescence de la commande par défaut,
type cde : type de la commande,
file fic : type du fichier (type texte ascii, type binaire exécutable, type archive etc …).
which cmde : localise une commande, affiche le nom de son chemin (pathname) ou son alias
type fic : donne une description du type de la commande.

df –k|n rep|dev|fic : volume du répertoire
df : informations sur l'occupation du disque. Affiche le nombre de fichiers et de blocs disques libres.

du –ksa rep|fic : volume (taille) du fichier ou du répertoire du disque dur.

date [**+**Format] : 1) sans argument affiche la date système, 2) avec argument, modifie la date système. Exemple : **date +%T** **date +%Y%m%D**
sleep *n*[**s**] : attend *n* secondes. **sleep** *n***m** : attend *n* minutes. **sleep** *n***h** : attend *n* heures.
Exemples : **sleep** 2 : attend 2 secondes. **sleep** 3**h** : attend 3 heures. **sleep** 3**m** : attend 3 minutes.

12.5 Commande de traitement ou de recherches de chaînes de caractères

12.5.1 Commandes " grep, egrep, fgrep "

grep [**-viln**] chaîne|critère fic [fic2 …] : recherche de la chaîne de caractère " chaîne " dans le / les fichiers listés dans la commande. Exemple : grep '*' fic : recherche toutes les lignes du fichier " fic " contenant le caractère " * ". Avec options :
-v : sélectionne les lignes correspondant au critère inverse du critère indiqué dans la commande,
-i : ne différencie pas les majuscules et minuscules dans la commande,
-l : donne une liste de nom de fichiers contenant le critère au lieu des lignes elles-même.
-h : idem
-n : affiche les lignes sélectionnées et leur n° de ligne.
Note : dans une expression régulière du grep, '**^**' indique début de ligne, '**$**' indique fin de ligne.
'**.**' indique un caractère quelconque, '**.***' indique une groupe de caractère quelconque.

Exemples : **grep –v '**oui**'** fic **grep –n '**R**.***n**'** fic **ls | grep '**.jpg**$**' **cat** fic **| grep `pwd`**

Expressions régulières dans certaines commandes (grep …) ou mode " commande " de certains éditeurs

Expressions régulières	**Signification**
.	Un caractère quelconque (joker)
^chaîne	Ligne commençant par " chaîne "
Chaîne**$**	Ligne finissant par " chaîne "
[xYyYzZ**]**	Classe de caractères
[x-z**]**	Intervalle de caractères
[^x**]**	Tout caractère sauf " x "

*****	Une ou plusieurs occurrence du caractère précédent
\x	Ote la signification spéciale du caractère " x " (en général un caractère spécial : " \ ", " * ",` ? " …)
\<	Marque de début de mot
\>	Marque de fin de mot
\(…\)	Un groupe dans une expression régulière qui peut être répété.
\1 \n (associé avec **\(**…**\)**	Répétition d'une fois, de 2 fois … de l' expression régulière \(…\)

Exemples :

grep '\(chaîne**\).*\1'** fic : sélectionne les lignes avec au moins 2 occurrences de la chaîne "chaîne" etc …
grep '\<q.*me\>' fic **; ls –la | grep '^d' ; find . –name '*.h' –exec grep 'time' '{}' \ ;**
niscat passwd.org_dir | grep 'ksh$'** ; **ypcat passwd | grep 'ksh$'

Note : **ATTENTION** ! une " expression régulière " est différente de " méta-caractère " (voir plus haut).

Expressions régulières	**Métacaractère correspondant**
.	**?**
^chaîne	Chaîne*****
Chaîne**$**	*****Chaîne
[xYyYzZ**]**	Idem
[x-z**]**	idem
[^x**]**	**[** !x **]**
*****	/ (n'existe pas)
\x	Idem
\<	/ (n'existe pas)
\>	/ (n'existe pas)
\(…**\)**	/ (n'existe pas)
\1 \n (associé avec **\(**…**\)**	/ (n'existe pas)

egrep : commande **grep** reconnaissant plus de caractères spéciaux que **grep**

Expressions régulières du egrep	**Signification**
?	Zéro ou plusieurs occurrence de l'expression régulière précédente
+	Une ou plusieurs occurrence de l'expression régulière précédente
|	Choix dans une liste (couple) d'expressions régulières séparées dans le caractère " **|** "
(… **)**	Groupe de caractère dans une expression régulière
\(…**\)**	/ (n'existe pas). Non reconnu
\(…**\)**	/ (n'existe pas). Non reconnu
\<…**\>**	/ (n'existe pas). Non reconnu

12.5.2 Commande " sed "

sed : éditeur non intératif, n'effectuant pas de modification dans les fichiers (uniquement comme filtre sur un flot de données en sortie d'un fichier ou d'un pipe " | ").
sed [**-n**] [**-e** instruction] [**-f**] fichier : avec :
-e : utilisé lorsque plusieurs instructions sont données sur la ligne de la commande " **sed** " (chaque instruction à son " **-e** ").
-n : les résultats du " sed " ne sont pas sortis (affichés) sur la sortie standard.
-f fichier_inst : fichier_inst est une une liste d'instruction (de fonctions) " sed " à traiter sur le fichier " fichier ".

Exemple : **cat** agenda **| sed 's/**mencherini/Mencherini**/' >** agenda_corrige
sed 'd/→9./&*/' agenda (ici → représente le caractère tabulation. Dans l'expression, il faut impérativement mettre le caractère tabulation).
Note ici " **&** " insère le masque (la chaîne) recherchée dans la chaîne de remplacement.

sed '/^→.*$/d' agenda : cette commande recherche toute ligne vide (sans caractère, sans blanc, sans tabulation …) et l'efface.
sed '/===/r fic_a_inclure**'** agenda **>** agenda_modifie : à chaque occurrence de la chaîne " === " inclus le contenu du fichier " fic_a_inclure **".**
sed 's/:x :[^:]*:/:x::/' passwd : afficher le fichier **passwd** sans sa colonne " **UID** ".
sed 's/:x :[^:]*:/:x::/w fic_no_uid**' passwd** : idem, mais le résultat au lieu d'être affiché à l'écran est redirigé dans le fichier " fic_no_uid **".**

Exemple de script " **sed** " à utiliser avec l'option " **-f** " du " **sed** " :
s/Ph**.***re**/**Phil Rebiere**/**
1,3d
s/→|0-9] [0-9]/ →/

Exemples :

uustat | cut -d" " -f1 | sed -e "1,\$s/^/uustat -k**/" > /tmp/**kiluu**$$**

sid**=`echo $**dirora **| sed "**s:oracle_::**"`**

find $startdir **-type d -print 2> /dev/null | sort -f | **
sed -e "s,^$startdir**,," **
**-e "/^$/d" **
**-e "s,[^/]*/\([^/]*\)$,\`-----\1," **
-e "s,[^/]*/,| ,g"

(voir le script « **dtree** » dans le document « *outils et scripts d'administration UNIX »* de ce cours « *Unix pour futurs administrateurs* »).

Tableau des commande “ **sed** ”

Résumé de la commande sed	
sed options 'instruction' fichier (s)	édition fichiers et résultat sur stdout
Syntaxe d’une adresse sed	
1,$ commande	opération sur tout le fichier
/chien/commande	opération sur lignes contenant "chien"
/chien/,$commande	opération de la ligne contenant "chien" à la fin de fichier
Description des fonctions et arguments de sed	
s/ancien/nouveau/[g]	substitue ancien par nouveau
s/ancien/&nouveau/[g]	ajoute nouveau à ancien
	annule les lignes indiquées (ex. 4,10d)
r fichier	charge fichier à la ligne indiquée
w fichier	écrit les lignes indiquées dans fichier
	imprime les lignes indiquées
a\ texte	place le texte après la ligne indiquée
i\ texte	place le texte avant le ligne indiquée
c\ texte	remplace lignes indiquées par texte
Description de quelques options sed	
-e instruction	utilisée lorsque plusieurs instructions sont données sur la ligne de commandes sed. Chaque fonction a son propre -e
-f fichier	fichier est une liste de fonctions sed à traiter sur le fichier
-n	les résultats issus d'une commande sed ne sont affichés sur la sortie standard.

12.5.3 Commandes “ nawk ”, “ awk ”

Notes : **nawk** est le **awk** de Solaris avec plus de fonctionnalités. Elle permet de traiter des champs. C’est un langage interprété, donc moins rapide d’exécution que des programmes compilées (comme le C …).

nawk [**-F** sép] [**-v** variable**=**valeur] **‘**pattern **{**action**}’** fichier
nawk [**-f** script_nawk] fichier
avec :
-F : indique quel est le séparateur de champs, dans le fichier “ fichier ” (par défaut, espace ou tabulation)
-v variable**=**valeur : permet d’initialiser des variables externe au programme “ nawk ”.
-f script_nawk : permet d’exécuter un programme **nawk** rédigé de manière externe.
pattern **{**action**}** : où “ pattern ” est encore appelée “ *expression régulière* (RE) ”, toujours entre “ */* … */* ” et “ action ” une ou plusieurs instruction du langage **nawk**.
3 cas :
pattern **{**action**}** : si “ pattern ” est vérifié, on exécute “ action ” (“ pattern ” est un filtre).
{action**}** : (sans présence de “ pattern ”), on exécute “ action ” sur toutes les lignes du fichiers.
pattern : on sort toutes les lignes correspondant / vérifiant “ pattern ”. On affiche la ligne entière. Cela équivaut à un **grep** .

Dans la partie “ action ”, voici les indications ou commandes **nawk** qu’on peut trouver :
print : affiche certains champs du fichier ou variables
$0 : variable indiquant la ligne entière.
$1 : variable indiquant le 1er champs du fichier
$2 : variable indiquant le 2ème champs du fichier
etc ..
Les champs, dans la ligne de commande **nawk** sont séparés par des “ **,** ”.

Exemple : **nawk ‘/9/{print $1, $4, $3 ‘’\t’’ $4}’** agenda : recherche lignes avec le chiffre “ 9 ” et reformate la sortie.
nawk ‘/→[0-9][3]/{print $1}’ agenda : imprime le champ 1 de lignes contenant une tabulation, suivi d’un chiffre entre 0 et 9, puis du chiffre “ 3 ”.
echo ‘’Il est**’’ `date | nawk ‘{print $4 }’ | nnawk –F : ‘{print $1}’ ` ’’**heure.**’’**
(affiche par ex. : il est 17 heure).

12.6 Chaînes ou caractères employés dans les nawks :

Symbole de la chaîne constante dans le nawk	**valeur**
\t ou **\011** ou **\042**	Tabulation
\n ou **\012**	Newline (saut à la ligne)
\045	%

12.7 Etiquettes " BEGIN " et " END " :

L'étiquette **BEGIN** indique qu'une {action} doit être exécutée **avant** que toutes les lignes soient lues. (souvent utilisée pour un titre, un entête d'un rapport),
L'étiquette **FIN** indique qu'une {action} doit être exécutée **après** que toutes les lignes soient lues. (souvent utilisée pour les sommaires et les totaux).
Note : dans les parties BEGIN et END, on ne peut utiliser les champs **$n** .
Exemple : **nawk '{**total **+= $6 } END {print** total **}'** agenda : on affiche la variable total qui est égale à la somme du 6° champ (de tous les enregistrement du fichier).

12.8 Opérateur d'affectation arthmétique :

Symbole de l'opération	Son action
+=	Ajouter à
-=	Soustraire à
/=	Diviser par
%=	Modulo (n)
^=	Puissance (de n)

12.9 Opérations dans le " print " :

Symbole de l'opération	Son action
+	Addition
-	Soustraction
*****	Multiplication
^	Exponentiation
++	Incrément
--	Décrément
%	Modulo

12.10 Liste des variables internes au " nawk "

Nom de la variable	Valeur par défaut	Description
OFS	Espace	Séparateur de champ en sortie
FS	Espace ou tab	Séparateur de champ en entrée
ORS	Newline	Séparateur d'enregistrement en sortie
RS	Newline	Séparateur d'enregistrement en entrée
NF		Nombre de champs dans l'enregistrement
NR		Numéro d'enregistrement depuis le début
FNR		Nombre d'enregistrements dans le fichier
ARGC		Nombre d'arguments sans la commande
ARGV		Tableau des arguments de la commande
OFMT	" %.6g "	Format de sortie pour les nombres
FILENAME		Nom du fichier d'entrée

Exemples :

nawk 'BEGIN {OFS=''/'' } {print $1,$2,$3}' agenda
nawk 'BEGIN {OFS=''\t'' } {print $2, $5*1000/$6}' agenda
nawk 'BEGIN {OFMT=''%.2g'' } {print $2, $5/$6}' agenda

Exemple de sortie, avec le dernier exemple précédent :
LISAN 0.69
LOTAR 2.4
LUTTEN 4
…

12.11 **Exemple de tableau avec** nawk **:**

List [0] = ‘’1° element’’
List [dupont] = ‘’jean-jacques’’ (“ dupont ” est un indice)
List [total] = ‘’55’’ (“ total ” est un indice)
List [1] = $0
Print list [0] list [dupont] list [total]

12.12 **L’instruction “** printf **” du** nawk

C’est une instruction **print** formatée du **nawk** (dont le formatage est inspiré du langage C).
Syntaxe : **{ printf (‘’** chaîne de caractères pouvant comporter des *masques***’’** [**,** valeurs] **) ; }**

Les valeurs sont utilisées pour remplir les *masques*. Voici la liste de ces masques :

Symbole du *masques*	**Format des données en sortie**
%d	Valeur entière
%f	Valeur point flottant
%c	Valeur caractère
%s	Valeur chaîne
%x	Valeur hexadécimale
%*n*d (idem pour : **%*n*f %*n*s %*n*c**)	La valeur est cadrée à droite dans la taille *n* du champ
%-*n*d (idem pour : **%-*n*f %-*n*s %-*n*c**)	La valeur est cadrée à gauche dans la taille *n* du champ
%*m.n*f **%.*n*f**	Avec les valeurs “ points flottants ”, il est possible de contrôler la taille du champ “ *m ”* et le nombre de décimales “ *n ”* après le point

Exemples : **printf(‘’**Le résultat est : **%10.2f %-10s %d\n’’,** num **/** 6 * 25 **, $3,** num **)**
printf(‘’%d, %7.2f, %20s\n’’, $1, $2, ‘’==➔’’)

12.13 **Expressions conditionnelles avec** nawk

nawk ‘{if (opérande opérateur opérande **)** action **} ‘** fichier

Symbole de l’opérateur relationnel	**Type**	**Rôle ou signification**
==	Test nombres ou chaînes	Egal à
!=	Idem	Différent de
>	Test nombres	Plus grand que
<	Idem	Plus petit que
>=	Idem	Plus grand ou égal à
<=	Idem	Plus petit ou égal à
~	Test chaînes	Contient /RE/ (contient expression régulière)
!~	Test chaînes	Ne contient /RE/ (ne contient expression régulière)
&&	Test logique	AND logique
||	Test logique	OR logique

Exemples :
nawk ‘{num = $5/$6 ; if (num>2) print $1, num }’ agenda
si le résultat de $5 divisé par $6 est supérieur à 2, on affiche $1 et le résultat de la division.
nawk ‘{if ($1 ~ /e/) print $1 }’ agenda
si l’on trouve des lignes dont le champ n° 1 contient un “ e ” on affiche la ligne.

12.14 **Fonctions de** nawk

tolower(chaîne**)** : convertit toute la chaîne “ chaîne ” en minuscule.
toupper(chaîne**)** : convertit toute la chaîne “ chaîne ” en majuscule.

Il en existe bien d’autre selon le type de awk (c’est à dire selon le constructeur) :
sort , **sqrt** , **exp** , **log** , …
Etc …

Exemple : **nawk ‘{if (tolower($4) == ‘’**paris**’’) print $0 > ‘’**fic_resultat_paris**’’ }’**

12.15 **Instructions de contrôle** next **et** exit

next : arrête le traitement sur la ligne courante, lit la ligne suivante du fichier d’entrée et redémarre au début du programme **nawk** .
Exemple : **nawk ‘{if ($5<**50**) next ; print $1}’** agenda
Si le résultat est vrai, la ligne est écartée, et la prochaine ligne est lue.

exit : termine le programme **nawk** . Elle transmet le code d’état (facultatif) indiqué dans le programme **nawk** au processus parent de **nawk** (sauf s’il existe un bloc END : dans ce cas, celui-ci est exécutée avant la fin du processus).
Exemple : **nawk ‘{if ($3 == ‘’**50**’’) {print ; exit 0}}’** agenda

12.16 Ecriture de sorties dans un fichier

nawk ‘{print $1, $2 > ‘’fic_resultat**’’}’** agenda
met le résultat dans le fichier “ fic_resultat **”.**

12.17 Commandes d’impression

lp : **lp -d** printer fic : impression du fichier (prendre plutôt un fichier texte),
lpstat : voir les jobs d’impression en file d’attente (queued jobs) et l’état de/des imprimante(s).
cancel n°_de_job : suppression du job d’impression ayant comme n° de job : “ n°_de_job ”.

13 Commandes d’administration UNIX de base

su [-] : changement de compte utilisateur.
id : informations sur l’identifiant de l’utilisateur en cours.
Exemple d’info affichées par la commande ‘id’ :
uid : 2000 (sam), gid : 10 (staff)

whoami : nom de login de l’utilisateur en cours.
who am i : idem mais …

14 Permissions sur un fichier

Permissions : **r w x**
r : 4 : lecture
w : 2 : écriture
x : 1 :exécution
7 : rwx , **6** : rw , **0** : aucune permission

chmod [-R] droits fic|rep : modifie les droits sur fichier(s) ou répertoire(s).
chmod –R droits rep : modifie récursivement les droits sur répertoires et tout ce qui se trouve dessous.

Classes :
u : usager
g : groupe
o : others (autres)
a : all

Opération concernant les droits :
= : affecte une permission ;
- : suppression d'une permission
+ : ajouter / positionner un droit

permissions : **r w x**

exemple : **chmod – R g+x** rep

chown nom_utilisateur [fic | rep] : changement du propriétaire du ou des fichiers ou répertoires listés dans la commande " **chown** ".

umask [droits] : en général mis dans le scripts d'ouverture d'une cession (login) utilisateur, positionne des droits sur les actions de l'utilisateur durant sa cession.

Par défaut : droits pour utilisateur : 777 pour les répertoire, 666 : pour les fichiers,

umask 000 (ne fait rien)
umask 022 (rw-r—r--)
umask 027 (rw-r-----)

15 Autres type de permission

setuid : S : **SUID** , s : **SUID** + execute
setgid : S : **GUID**, s : **GUID** + execute
sticky : T : **STICKY** , t : **STICKY** + execute

exemple de droits **SUID** et **GUID** : -r-Sr-sr-x
exemple de commande **chmod** pour positionner le **GUID** : **chmod 2777** rep => drwxrwSrwx

Comment cela fonctionne ?
L'exécutant d'id = id1, prend l'id = id2, pour exécuter le fichier du owner d'id = id2.

16 Chmod – tableau récapitulatif

Notations symboliques					
Classe		Opérations		Permissions	
u	user (propriétaire)	=	assigne	r	lecture
g	group	-	retire	w	écriture
o	others	+	donne	x	exécution
a	all				

Notations octales		
Valeur	Permissions	Explication
7	rwx	Lecture, écriture, exécution
6	rw-	Ecriture, lecture
5	r-x	Lecture, exécution
4	r--	Lecture
3	-wx	Ecriture, exécution
2	-w-	Ecriture
1	--X	Exécution
0	---	Aucun accès

Permissions spéciales			
Valeur	Permissions	Chmod	Explication
4	setuid : -rwsr-x--x	4751	Prend id. user
2	setgid : -rwxr-s--x	2751	Prend id. groupe
1	sticky bit : drwxr-x--t	1751	Limite effacement

17 Fichiers importants pour les droits

/**etc/passwd** : fichier des utilisateur
/etc/shadow : (il est **root** et **sys**, mais un utilisateur par la commande " **passd** " peut modifier un mot de passe dedans, parce que la commande " **/usr/bin/passwd** " a les droits –r-Sr-sr-x)

Note : **netstat** : donne les statistiques réseau.

Un sticky bit :
1) sur un répertoire : indiquer que seul " **root** " et le owner peuvent supprimer les fichiers qui s'y trouvent.
(par exemple, pour éviter que les fichiers dans **/tmp** soient effacés par d'autres utilisateurs que " **root** ".
exemple de droits sur **/tmp** : drwxrwxrwt)
2) sur un fichier : la taille du fichier est fixe (ex. : zone de swap, diskless). Permet de réserver de la place
sur le disque.

Permissions	Valeurs octales	Valeur symbolique
Setuid (SUID)	**4**	**S**
Setgid (GUID)	**2**	**S**
Sticky bit	**1**	**t**

18 Permissions et fichiers ACL

ACL : définition d'un fichier de sécurité pour un fichier ou répertoire.
Owner, group, others, owners spéciaux, groupes spéciaux,

Exemple de commande de mise à jour des droits =>
setfacl –s user : :rw-, group : :---, other :---, mask :rw- fic
Autre exemple de mise à jour : **setfacl – m user :**matin**:rw, group:**market**:r** fic

getfactl : **getfacl –a** permission_courante_et_fichier_ACL
gatfacl –d :acl_par_défaut
Exemple de commande d'affichage des ACL :

19 Les éditeurs sous UNIX

19.1 Introduction des éditeurs courant sous Unix

ed : éditeur ligne
vi , **view** , **vedit** : éditeur plein écran
sed : éditeur sur un flot de données (en général sur le résultat d'un " pipe ").

Pour mention **dtpad** : éditeur pleine page graphique dans l'environnement graphique CDE X-Windows.

vi –r fic : récupération du fichier qui était en cours d'édition par " vi " (en fait, un " recovery "). On va le chercher dans " **/var/tmp** ".
vi +-n° fic : édite à partir de la ligne dont le numéro = n°
vi +/chaîne fic : éditer à partir de la 1ière occurrence de la chaîne dans le fichier fic.

Tableau des commandes “ vi **”**

Différent mode de vi	Description
Mode Insertion	
a	Ajout après le curseur
A	Ajout à la fin de la ligne
i	Insère avant le curseur
1	Insère au début de la ligne
0	Ouvre une ligne après le curseur
0	Ouvre une ligne avant le curseur
cw	Modifie un mot
r	Remplace un caractère
R	Remplace jusqu'au prochain <ESC>
Mode commande	
x	Efface un caractère
dw	Efface un mot
dd	Efface une ligne
yy	Sélectionne une ligne
p	Pose la sélection après le curseur
P	Pose la sélection avant le curseur
u	Défaire la dernière modification
.	Refaire la dernière modification
nG	Aller à la ligne n (ou à la dernière ligne si n est omis)
/chaîne	Recherche "chaîne" en avançant dans le fichier
?chaîne	Recherche "chaîne" en remontant dans le fichier
Mode ligne	
:w	Sauvegarde le fichier
:wq	Sauvegarde et quitte le fichier
:q	Quitte sans sauvegarder
:q!	Quitte sans rien sauvegarder du tout

19.2 Commandes d'insertion sous " vi "

Différents mode de vi	Description
Mode insertion	
a	Ajout après le curseur
A	Ajout à la fin de la ligne
i	Insère avant le curseur
1	Insère au début de la ligne
o	Ouvre une ligne après le curseur
O	Ouvre une ligne avant le curseur
cw	Modifie un mot
r	Remplace un caractère
R	Remplace jusqu'au prochain <ESC>

19.3 Commandes de fin sous " vi "

Fin de cession d'édition	
Mode commande	
ZZ	retourne sous le prompt de l'interpréteur.
Mode ligne (ou mode ex)	
Ecriture d'un fichier	
: w [nouveau_fichier]	Le nouveau fichier est écrit sur disque. Si le fichier existe déjà, un message apparaît à l'écran pour recommander l'utilisation de w! (réécrit sur un fichier existant).
: ligne début,ligne fin**w** [nouveau_fichier]	
:w! nomfich	
Fin de la session d'édition	
:q	quitte vi et retourne sous l'interpréteur
:q!	annule les modifications faites et récupère le fichier dans son état initial
:wq	sauvegarde le fichier et quitte vi
:x	sauvegarde le fichier et quitte vi (identique à :wq)

19.4 Les commandes de déplacement

Déplacement sur l'écran Mode commande	
Déplacement avec les touches de fonction	
→ ou 1	Déplacement d'un caractère à droite (id. <space>)
← ou h	Déplacement d'un caractère à gauche (id. <bs>)
↑ ou k	Déplacement d'une ligne vers le haut
↓ ou j	Déplacement d'une ligne vers le bas (id. <retour>)
Déplacement avec les touches du clavier	
+	Positionnement au début de la ligne suivante
-	Positionnement au début de la ligne précédente
Déplacement avec les caractères de contrôle	
^d	Descend d'un demi-écran
^u	Monte d'un demi-écran
^f	Déplacement d'une page vers la fin du fichier
^b	Déplacement d'une page vers le début du fichier
Shift-g	Fin de fichier

19.5 Les commandes de recherche sous “ vi ”

Déplacement recherche mode commande	
Recherche de texte	
/chaîne	recherche chaîne vers la fin du texte.
?chaîne	recherche chaîne vers le début du texte.
N, // ou ??	trouvent l'occurrence suivante (n pour next)
N	répète la dernière recherche en sens opposé

19.6 Les commandes de déplacement dans la ligne sous “ vi ”

Déplacement sur la ligne Mode Commande	
Déplacement vers l’avant de la ligne	
$	déplace à la fin de la ligne
w	déplace d’un mot sur la droite
#w	déplace du nombre spécifié de mots vers la droite; par exemple 5w déplace de 5 mots.
Déplacement vers l’arrière de la ligne	
(zéro) ou ^	déplace au début de la ligne
b	déplace d'un mot vers la gauche (backward)
#b	déplace du nombre spécifié de mots vers la gauche; par exemple 3b déplace vers la gauche de 3 mots.

19.7 Les commandes de changement et de recherche de texte sous vi

Changement et remplacement de texte Mode insertion	
Changement de texte	
cw	change un simple mot
#cw	change le nombre spécifié de mots
C	change le texte jusqu'à la fin de la ligne
$	marque la fin de la suite de caractères à substituer
Remplacement de texte	
R	remplace le texte jusqu'à ce que la touche <ESC> soit entrée
r	remplace le caractère présent sous le curseur avec le nouveau caractère saisi au clavier

19.8 Les commandes de suppression sous " vi "

Suppression de texte mode commande	
Suppression de texte	
x	supprime un caractère
4x	supprime 4 caractères
dw	supprime 1 mot
6dw	supprime 6 mots
dd	supprime 1 ligne
3dd	supprime 3 lignes
dO (zéro)	supprime vers le début de la ligne
D ou d$	supprime vers la fin de la ligne
J	concatène deux lignes (suppression du retour chariot

• Les caractères supprimés sont conservés dans un buffer.
• On peut les faire réapparaître à l'endroit choisi en se positionnant et en utilisant P ou p (Put) pour poser le contenu du buffer.
• C'est un moyen pour déplacer des morceaux de texte.

19.9 Les commandes de substitution sous " vi "

Substitution de texte Mode ligne	
Entrez en mode ligne par " : "	
:début,fin**$**/ancienne/nouvelle/	Cela substitue de la ligne *début* à la ligne *fin* toutes les premières occurrences sur chaque ligne de *ancienne* par *nouvelle*.
:début,fin**$**/ancienne/nouvelle/**g**	Pour substituer toutes les occurrences, il faut le préciser par *g* en fin de commande (g pour *global*).

•Le caractère"$"fait référence à la dernière ligne du texte.
•Le caractère"."fait référence à la ligne courante.
•Le caractère""remplace à lui seul 1,$

19.10 La commande globale sous “ vi ”

Commande globale Mode ligne	
Entrez en mode ligne par “ : ”	
:début,fin**g**/chaîne/commande	La commande est exécutée lorsque entre la ligne *début* et la ligne *fin*, *chaîne* est détectée. Il s’agit d’un filtre spécial de recherche.
:début,fin**g**/chaîne/**S**/ancienne/nouvelle/**g**	La substitution est faite pour toutes les lignes entre *début* et *fin*, contenant *chaîne*. (Premier g pour le *filtre* et le second g pour *global*).
:début,fin**g**/chaîne/**d**	Toutes les lignes contenant *chaîne*, dans l’intervalle indiqué, seront détruites.

19.11 Les commandes de déplacement de texte sous vi

Déplacement de texte Mode ligne	
Entrez en mode ligne par “ : ”	
:début,fin**m**no_de_ligne	Déplacement du texte après le n° de ligne indiqué en destination de *no_de_ligne* et renumérotation des lignes à l’écran.

19.12 La commande d’interaction avec le shell sous “ vi ”

Les accès au système Unix sous “ vi ” en Mode ligne	
Entrez en mode ligne par “ : ”	
:r fichier	Insertion du contenu de *fichier* à la ligne courante
:! commande_Unix	Exécution de la commande et retour sous “ vi ”
:r! commande_Unix	Exécution de la commande et insertion de son résultat après la ligne courante

19.13 Les commandes de personnalisation sous vi

Personnalisation de la session vi Mode ligne	
Entrez en mode ligne " : "	
:set	Affiche tous les paramètres actuellement positionnés
:set ail	Affiche tous les paramètres possibles sous vi et leur valeur actuelle
: set var [=valeur]	Initialisation
Quelques variables de l'éditeur	
ai	Permet l'auto-indentation
IC	Dans recherches/substitutions, pas de distinction entre minuscules et majuscules
Nu	Affiche les numéros de ligne
List	Visualise des fabulations (A!) et des fins de lignes ($)
Ws	Boucle sur une recherche (wrapscan)
Smd	Affiche en bas à droite le type d'insertion en cours ("OPEN, INSERT, APPEND MODE")
: set novar	Supprime une initialisation

19.14 Exemple d'abréviations sous " vi "

Abrége une chaîne de caractère en une chaîne restreinte.

:ab chaîne chaîne_longue
:ab rep repertoire
:ab => Donne la liste des abréviations valides.
:unab chaîne => supprime une abréviation.

19.15 Exemple de macros sous " vi "

Permet de simplifier les commandes et programmer les touches de fonctions.

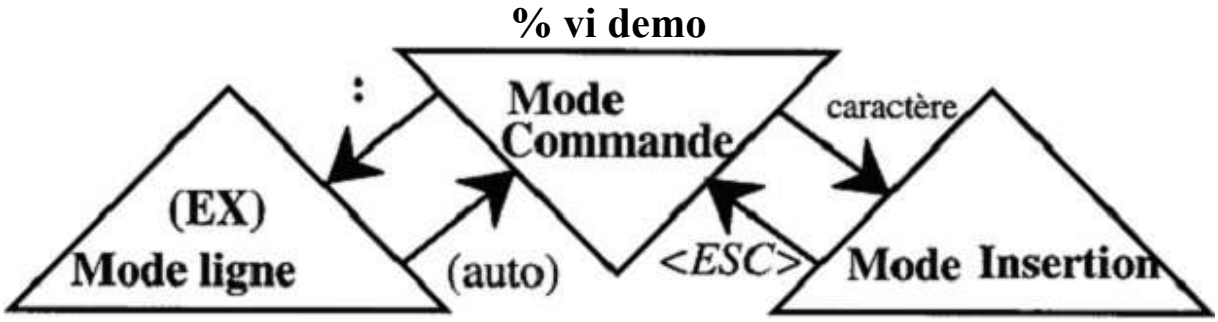
:map caractères commande
:map X 10dd => Le caractère X permettra la suppression de 10 lignes.
:map => permet de visualiser les macros définies.
:unmap caractères => supprime une macro.
:map ^[[20~ :set list^M => ^[[20~ est la touche de fonction F9 et s'obtient par la séquence clavier suivante : ctrl-v + touche de fonction F9.

19.16 Exemple de contenu du fichier " .exrc "

map ^[[11~ :set list^M
map ^[[12~ :set nu^M
set ai
ab sys systeme de fichiers
ab rep repertoire

Note : insertion d'un raccourci clavier dans le fichier (fic) sous " **vi** " :
Sous " **vi** ", **Ctl-V** + frappe en même temps de la touche de fonction choisie (par ex. F9).
Ce qui donne quand on frappe F9, dans le fichier (fic) : ^[[20~

19.17 Tableau détaillé et récapitulatif de toutes les commandes " vi "



Déplacement et insertion de texte :

:3,8d Annule lignes 3 à 8
:4,9m 12 Déplace 4 à 9 à la ligne 12
:2,5t 13 Copie 2 à 5 à la ligne 13
:5,9w file Ecrit lignes 5 à 9 dans file

Sauvegarde de fichiers et sortie :

:w Ecrire buffer sur disque
:w newfile Ecrire buffer dans newfile
:w! file Ecrire impérativement
^q Ecrire buffer et sortir
:q Quitter l'éditeur
:q! Quitter impérativement
:e! Ré-éditer en oubliant les modifs
:wq Ecrire buffer et sortir

Contrôle de l'édition :

:set nu Afficher les numéros de lignes
:set nonu Oter une option
set ail Montrer les options
set list Afficher les car. invisibles
set wm=5 Garder 5 espaces de marge de droite

Déplacement d'écran/de ligne :

j, k, h, 1 et touches fléchées
0 Au début de la ligne
$ A la fin de la ligne
% A la parenthèse / accolade correspondante
G A la dernière ligne
3G A la ligne 3

Déplacement de mots :

w En avant d'un mot
3w En avant de 3 mots
b En arrière d'un mot
3b En arrière de 3 mots

Fonctions de recherche :

n Répéter recherche précédente
N Inverser recherche précédente

Effacement de texte :

X Efface un caractère
dw Efface un mot
dd Efface une ligne
D Efface jusqu'à la fin de la ligne
d0 Efface jusqu' au début ligne
dG Efface jusqu'à la fin de fichier
4dd Efface 4 lignes

Annulation de fonction d'édition :

u Défaire dernière modification
. Faire la dernière modification à nouveau

Copie et insertion de texte :

Y Sélectionne une ligne
5Y Sélectionne 5 lignes
p Dépose la sélection après curseur
P Dépose la sélection avant curseur

Fonctions de traitement de texte :

J Union de la ligne suivante avec la courante
4J Union de 4 lignes avec la courante
xp Transpose 2 caractères

Fonctions de recherche :

/exp En avant pour exp
?exp En arrière pour exp

Ajout de texte :

a Ajout après le curseur
A Ajout à la fin de la ligne
i Insère avant le curseur
5i Insère le texte 5 fois
I Insère au début de la ligne

Ajout de nouvelles lignes :

o Ouvre une ligne après curseur
0 Ouvre une ligne avant curseur

Modification de texte :

cw Modifie un mot
3cw Modifie 3 mots
C Modifie une ligne
r Remplace un caractère
R Remplace/ retape une ligne

20 Commandes de sauvegarde et restauration : tar, cpio, jar, pax

Les commandes les plus portables et employées dans le monde Unix : **tar** et **cpio**

Rappel : nom du répertoire des fichiers de type *device* bande : **/dev/rmt**
Dans ce dernier répertoire, on y trouve des noms de device bande :
0cbn : avec
0 : n° d'instance du lecteur de bande
c : indique que la bande est compressée
b : format permettant une compatibilité BSD (Unix Berkeley)
n : la bande n'est pas rembobinée en fin de lecture (de cassette)

0[hm]**bn** : avec " hm "
u : ultra
h : high
m : medium
l : low density

Voir en annexe : le tableau des Noms des fichiers spéciaux liés à des unités de bandes.

mt : **mt -f** device : définit quel fichier spécial (device) utilisé pour la commande " mt "
mt [option] : avec option
fsf n : saute n enregistrement => vers l'avant.
bsf n : saute n enregistrement => en arrière.
rewind : rembobinage de la bande
eom : on va en fin de la bande.
status : affiche l'état de la bande (le lecteur est actif, en panne, quelle bande est montée …)
erase : effacement de la bande

20.1 Commande de sauvegarde restauration " tar " :

- " **tar** " n'est pas multi-volumes.
- La commande " **tar** " sert à concaténer des fichiers sur le disque ou sur bande.
- Elle sert aussi à visualiser le contenu d'un fichier archive de type " **tar** " : **tar tvf** fic

tar *option fichier_ou_ensemble_de_fichiers fic_archive_tar* avec option :
c : création d'un / des enregistrement(s) ou fichier(s)
x : lecture / extraction d'un / des enregistrement(s) ou fichier(s)
v : affichage de plus d'information (mode " verbeux " _ " verbose "),
f : précise le device,
t : liste les fichiers lus ou copiés,
p : lorsqu'on extrait le fichier, on conserve ses permissions (owner, mode …) (important)

Exemples :

tar cvf /dev/dsk/floppy0 ./sauvegarde
cd /tmp ; **tar xvf** /dev/dsk/floppy0
tar cvf sauve.tar ./sauvegarde
tar cvf /dev/rmt/0n ./sauvegarde
cd ; tar cvf /dev/rmt/0n .

20.2 Commande de sauvegarde restauration " cpio " :

- " **cpio** " est multi-volumes.
- " **cpio** " n'est pas compatible avec " tar " (on ne peut pas lire une bande " tar " avec " cpio " et réciproquement).

cpio –i -[vcBd …] < fic_special ou **cpio –i -[vcBd …] <<** fic_special
cpio –o –[vcB …] > fic_special ou **cpio –o –[vcB …] >>** fic_special
avec option :
-t : liste contenu d'un enregistrement,
-v : verbeux
-c : écrit un entête ASCII pour la portabilité (d'un Unix à l'autre)
-B : transfert par bloc de 5120 bytes (octets),
-d : création des sous-répertoires si nécessaire (important).

Exemples :

ls | cpio –ovcB > /dev/dsk/floppy0

20.3 Commande de sauvegarde restauration " pax " :

- " pax " est multi-volume :
- " pax " peut aussi lire des fichiers au format " tar " (USTAR) qu'au format " cpio "
- (" pax " sert de " pont " entre la commande " cpio " et la commande " tar ").
- Les fichiers sauvegardés peuvent être enregistrés en absolu ou en relatif.
- La commande " pax " n'est pas disponible sur toutes les plateformes mais est disponible sur Solaris, AIX, Ultrix (Digital Unix), FreeBSD, …

pax [-v] [-s chaîne] -f fichier_spécial
pax [-v] [-x format] [-s chaîne] **-f** fichier_spécial
pax –r [-vi] [-x format] **[-s** chaîne] **-f** fichier_spécial

avec comme options pour la commande **pax** :

aucune option : liste le contenu d'une archive (liste le contenu de la table des symboles)
-w : écrit (sauvegarde)
-r : restauration
-f : indiquer le device choisi c'est à dire le fichier de sortie (par exemple /dev/rmt/0hn)
-s : substitution de caractères (dans un nom de fichier)
-i : mode interactif
-v : verbeux (verbose)
-x : format de la sauvegarde : " tar " (USTAR) ou " cpio ".
Par défaut, le format est de type " tar " (USTAR).

Exemples :

pax –w –v –f fic.pax .
pax –v –f fic.pax
pax –r –v –f fic.pax

pax –r –v –s '/.$/&_1/' –f fic.pax fic_cible : ici on remplace dans tous les noms de fichiers
" le_nom_du "fichier par le 'nom_du_fichier_1'.

pax –r –v –i –f fic.pax fic_cible

20.4 Commande de sauvegarde restauration " jar " :

- " jar " génère un fichier au format compressé " ZIP " portable.
- " jar " crée une archive sauvegardée uniquement en relatif.
- La syntaxe de la commande " jar " est la même que la commande " tar "
- Elle est supportée sur un bon nombre de plateformes Unix.

jar [options] fic.jar [fic | rep] avec options :
c : créée nouvel enregistrement

x : extrait un enregistrement à partir de l'archive
t : indique le device utilisé
t : liste le contenu d'un enregistrement

exemples :

jar cvf fic.jar *
jar xvf fic.jar
jar xvf fic.jar fic_a_extraire : on extrait de l'archive " fic.jar " que le fichier à extraire, ici " fic_a_extraire ".
jar tvf fic.jar : visualisation du contenu du fichier " fic.jar " sous la forme d'une liste de fichier.

21 Gestion mémoire

21.1 Swaping :

Si risque saturation mémoire RAM centrale, pageout prend le processus le plus endormi et le stocke en zone de swap sur disque.
Le noyau n'est jamais swapé.
Limite absolue mémoire : taille mémoire RAM + taille de la zone de swap.

L'utilitaire **swap** permet d'ajouter, de supprimer et de superviser la zone de swap.
Syntaxe : **/usr/sbin/swap -a** swapname [swaplow] [swaplen] : ajout de mémoire à la swap.
/usr/sbin/swap -d swapname [swaplow] : suppression de la swap ayant pour nom «swapname ».
/usr/sbin/swap -l : affichage de la swap.
/usr/sbin/swap -s : idem

21.2 Informations sur les processus : la commande « ps »

La commande " ps " permet de retrouver toutes les informations utiles sur un processus.

Syntaxe : **ps** [options] avec options :
-e : indique le statut de tous les processus (exec, wait, defunct … user id différents ?)
-f : tous les informations possibles sur les processus actifs
-t : affiche les processus actifs liés à un terminal
-u : affiche les processus actifs liés à un utilisateur.

Informations ramenés par / résultat d'un " ps " :

Nom de la Colonne	Description
UID	Uid (n° d'identifiant) de l'utilisateur
PID	N° d'identification du processus
PIDP	N° d'identification du processus père
C	Utilisation du processeur pour le sheduler
STIME	Instant de démarrage du processus (en heures, minutes et secondes). Si le processus a plus de 24 h, il est donné en mois et jours
TTY	Terminal de contrôle du processus
TIME	Temps CPU utilisé en secondes
COMD	Ligne de commande

D'autres commandes peuvent ramener des informations sur les processus et la charge machine :
top : ramène la liste des processus classés par ceux les plus " gourmands " en ressources.
sar :
wmstat : statistiques sur l'occupation de la mémoire et mémoire virtuelle.
Ce genre de statistiques existent aussi en sortie graphique (outils payants SUN Solstice, …).

21.3 Commande pour terminer un processus : kill

Envoi un signal à un processus ou à un ensemble de processus (identifiés par leur PID).

Kill [-n°_de_signal] pid [pid2 …] avec comme n°_de_signal :

N° de signal	SYMBOLE	Description
15 ou sans n° de signal	**SIGTERM**	demande au processus de se terminer (“ **kill** ” “ doux ”).
9	**SIGKILL**	Force la fin du processus (“ **kill** ” “ brutal ”).
Etc …		

- Certains processus ne peuvent être tués et certaines renaissent quand on cherche à les supprimer.
- Pour obtenir la liste des “ n° de signaux ” possible, 3 solutions :

- A) **man kill**
- B) **man –s5 signal**
- C) **kill –l** n°_de_signal

- Conseil : tuer le processus d’abord par “ **kill –15** no_processus ” avant, si vous n’arrivez pas à le tuer, de le faire par “ **kill –9** no_processus ” (avec no_processus : n° du processus à tuer).

Exemple d’une commande pour tuer un processus d’un nom donné :
Kill –9 `ps –ef | grep process |grep –v grep| nawk ‘{print $2}’`

Voir aussi chapitre suivant.

21.4 Commande de gestion de processus :

Commande	Action sur le processus
&	Lance le processus en arrière-plan (en background). “ & ” à mettre à la fin de la commande
fg	Ramène le job courant de l’arrière-plan à l’avant-plan
bg	Relance un job stoppé et le place en arrière-plan
Ctl-c	Termine le processus en/au premier plan
Ctl-z	stoppe l’exécution du processus en cours en/au premier plan
jobs	Liste les jobs stoppés ou/et passés en arrière-plan
kill %no_job	Terminer un job en arrière-plan en indiquant son n° de job : no_job
kill %+	Terminer le job courant
kill %-	Terminer le job précédent

Exemple :

sort –r fic > fic_trie **&**
ls –lR / > liste **&**
fg %2 : fg place le job **%2** en premier plan. On peut ensuite le stopper par **Ctl-z** ou le tuer par **Ctl-c**
bg %2
jobs

22 Notions de rédaction de scripts shell en Korn shell :

22.1 Configuration de l'environnement en Korn shell et Bourn shell :

Information concernée	Bourne shell	Korn shell
Fichier " profile " pour tous les utilisateurs	**/etc/profile**	**/etc/profile**
Fichier " .profile " de l'utilisateur : caractéristique du terminal et variables d'environnement (lues une seule fois)	**$HOME/.profile**	**$HOME/.profile**
Fichier " **.kshrc** " des variables locales du Korn shell et **alias** du Korn shell		**$HOME/.kshrc**
Variables d'environnement pré-définie	**$HOME**	**$HOME** **ENV=$HOME/.kshrc**

22.2 Commandes sur les variables shell en Bourne shell et Korn shell :

. **.**fic : execution d'un fichier caché (en premier plan).

Commande	Signification
VARIABLE**=**valeur	déclare la valeur
export VARIABLE	export une valeur (pour les shell fils crées par un script shell ou un utilisateur)
echo $VARIABLE	affiche contenu de la variable
env	affiche toutes les variables dans le shell appelant (shell père) (variables d'environnement)
set	affiche toutes les variables dans le shell appelé (shell fils) (variables locales)
unset VARIABLE	supprime une variable
set -x	Met en le shell en mode débuggage (équivant à **sh –x** commande).

22.3 Variables Bourne shell toujours présentes dans l'environnement de l'utilisateur

Variable	Usage
HOME	Contient chemin du répertoire d'accueil de l'utilisateur
LOGNAME	Nom (login) de l'utilisateur
PATH	Chemins vers les répertoires par défaut (ceux pour lesquels si l'on trouve la commande dans ce chemin, on n'a pas à ressaisir toute le chemin de la commande pour exécuter la commande). Exemple : **PATH=/usr/bin:/usr/sbin**
SHELL	Type de shell initialisé pour l'utilisateur (exemple : **SHELL=/bin/sh**)
TERM	Type de teminal (utilisé par " vi ", par " dtpad " etc …) (exemple : **TERM=ddtterm** ou **TERM=vt220**)
PWD	Représente le chemin au répertoire courant (celui sur lequel on est à l'instant)
PS1	Prompt de 1er niveau (par défaut : $. Sauf pour l'utilisateur " **root** " : " # ").
PS2	Prompt de 2ème niveau (par défaut : >).

22.3.1 Alias

Un " alias " est nom symbolique donné à une commande UNIX, un script, un programme, par l'utilisateur (uniquement en Korn shell

alias : sans argument, donne la liste de tous les alias défini dans le shell courant.
alias nom_alias**='**définition**' :** créer un alias ayant comme texte 'définition'
Exemple : **alias** lpmoz='**lp –d** MOZART'
unalias nom_alias : supprime l'alias " nom_alias "

22.3.2 Alias " **history** " et fichier historique des commandes passées : " **.sh_history** " :

- L'alias " **history** " affiche les 16 dernières commandes passées, chaque commande passée étant affichée précédée d'un numéro " num_ordre_historique ".
- Ces commandes passées sont stockées dans le fichier " **.sh_history** " passé dans le répertoire de l'utilisateur d'accueil "HOME ",
- La variable **HISTFILE** placé dans un des fichiers d'initialisation de l'environnement d'accueil de l'utilisateur permet de changer le nom de ce fichier historique,
- La variable **HISTSIZE** placé dans un des fichiers d'initialisation de l'environnement d'accueil de l'utilisateur permet de changer le nombre de commandes visibles dans l'historique,
- On peut rappeler la commande déjà passé, par la commande " **r** num_ordre_historique " (voir ci-dessus).
- La commande " fc " permet d'éditer une commande passé, puis d'exécuter cette commande modifiée :
  - **fc** [**- e** éditeur] num_ordre_history exemple : **fc –e vi 14**
- Un solution plus simple est de lancer le Korn shell en mode " **vi** " (par exemple : " **ksh –o vi** " ou bien **" ksh ; set –o vi** ").
- La commande " **set [+-] [**drapeau**]** " permet d'activer ou de désactiver un " drapeau " reconnu par le Korn shell.
- Exemple de " drapeaux " :
  - " **vi** " : Initialise le mode insertion " **vi** " pour la ligne de commande, tant que la touche **Esc** n'est pas enfoncée.
  - " **noclobber** " : protège les fichiers existant contre les redirections " > ". Pour contourner ce verrou, il faut utiliser le mécanisme de redirection " **>|** ". Exemple : **set –o noclobber ;** commande **>|** fic
  - " **ignoreeof** " : inhibe la possibilité de quitter son shell avec **Ctl-D**
- " **set –o** " sans argument liste les " drapeaux " positionnés.

23 Rédaction de scripts (programmes) Bourne shell

Exemple de script :

```
# !/bin/sh
# Note :cette première ligne permet de ''setter'' l'environnement shell en cas d'exécution de ce script
# par le processus " cron " (qui lui par défaut, ne ''sette'' aucun environnement shell à l'exécution d'un script shell)
# Note : attention, les accentués ne marchent pas toujours dans les scripts shell
# Cela dépend du type de terminal (i.e. du type de vt ).

# exemple d'initialisation de variable en début du script
var1=A
var2=B
var3=C

# on peut créer une nouvelles= variable, concaténant le résultat des variables précédentes :
vartot=$var1$var2$var3

# on peut aussi concaténer le contenu d'une variable avec une chaîne de caractère :
vartot2=${var1}tion
vartot2=${var1}''tion : ''
```

23.1.1 Substitution de paramètres en shell

Syntaxe	**traitement**
${var **:**=valeur_par_defaut**}**	Si la variable est déjà initialisée, on substitue son contenu, sinon on substitue par la valeur par défaut.
${var **:-**valeur_par_defaut**}**	Si la variable est déjà initialisée **et non nulle**, on substitue son contenu, sinon on substitue par la valeur par défaut.
${var **: ?**valeur_par_defaut**}**	Si la variable existe **et non nulle**, on substitue son contenu, sinon on substitue par le nom de la variable suivi du message. Si le message est vide, on obtient le message " parameter null or not set " par défaut. Exemple : ${var **: ?**''texte message''**}**

23.1.2 La commande " **set** "

Elle permet de mettre chaque chaîne résultat d'une commande dans les variables standard du Bourne shell : **$1 $2 $3 …**
Exemple : **set `date`** par exemple : **$1** contient **Wed** **$2** contient **Jun** etc. ..

23.1.3 la commande " **shift** "

Elle déplace à gauche tous les contenus des variables standards shell **$1 $2 $3** …
La 1ère valeur de **$1** est éliminé, et le contenu de **$2** transféré à **$1** et devient le contenu de **$1** et ainsi de suite.
(même mécanisme pour **$2** et **$3** …).
Le compteur du nombre d'argument de **$#** est décrémenté de 1.

23.1.4 Variables standards du shell

Syntaxe	**Signification**
$*	Contient tout les paramètres passés au script shell
$@	Donne la liste de tous les paramètres passés à un script shell
$#	Nombre de paramètres passés à un script shell
$?	Code retour de l'exécution du script shell ou de la commande que l'on vient juste d'exécuter (**0** : exécution correcte , **> 0** : défaillance, erreur …)
$0	Nom du script shell invoqué (celui en cours d'exécution)
$1	1er paramètre passé au script shell
…	…
$9	9° paramètre passé au script shell

Attention, il ne peut y avoir plus de 9 paramètres passés à un script shell
Astuce : passer un paramètre, sous la forme d'une chaîne, contenant elle-même plusieurs valeurs séparées par des blancs et entourées par des doubles cotes.

Exemple :

test_param A1 B2 C3 D4 E5 F6 G7 H8 "I9 J10 K11"

23.1.5 L'opérateur " **test** "

3 syntaxes :
1) **test** expression_condition commande

2) **if [**expression_condition **]**
then
commande ; commande …
[**else**
commande ; commande …]
fi

c) **if test** expression_condition
then
commande ; commande …
[**else**
commande ; commande …]
fi

3) avec " **elif** "
if condition
then
commande ; commande …
[**elif** condition
then
commande ; commande …]
[**else**
commande ; commande …]
fi

Note : il est préférable d'utilisateur l'instruction " **case** " :

23.1.6 L'instruction " **case** " :

case variable **in**
valeur1 **)** commande11 [; commande12 …] ; ;
valeur2 **)** commande21 [; commande22 …] ; ;
…
valeur*N* **)** *commandeN1* [; *commandeN2* …] ; ;
***)** commandex1 [; commandex2 …] ; ; # tous les valeurs non citées dans la liste de valeurs ci-avant
esac

23.1.7 La boucle " **for** "

for variable **in** liste # pour la liste de valeurs, on exécute les commandes ci-dessous
do
commandes ;
done

23.1.8 La boucle " **while** "

while condition # tant que la condition est vérifiée, on exécute les commandes ci-dessous
do
commandes ;
done

23.1.9 Opérateur de test :

Opérateur de test	**Type pour**	**Signification** (retourne **0** si la condition ci-dessous est vérifiée)
Str1 = Str2	**Chaînes**	égalité entre Str1 et Str2
!=	**Chaînes**	inégalité entre Str1 et Str2
Str1	**Chaînes**	si la chaîne Str1 n'est pas la chaîne vide
-n Str1	**Chaînes**	si chaîne Str1 n'est pas nulle (sa taille n'est pas égale à zéro)
-z Str1	**Chaînes**	si chaîne Str1 est nulle (sa taille est égale à zéro)
Int1 **-eq** Int2	**Entiers**	Si Int1 est égal à Int2
Int1 **-ne** Int2	**Entiers**	Si Int1 n'est pas égal à Int2
Int1 **-ge** Int2	**Entiers**	Si Int1 est supérieur ou égal à Int2
Int1 –**gt** Int2	**Entiers**	Si Int1 est supérieur à Int2
Int1 –**le** Int2	**Entiers**	Si Int1 est inférieur ou égal à Int2
Int1 –**lt** Int2	**Entiers**	Si Int1 est inférieur à Int2
-d fichier	**Fichier**	Si fichier est un répertoire
-f fichier	**Fichier**	Si fichier est un fichier ordinaire
-r fichier	**Fichier**	Si fichier est un fichier qui peut être lu (read), par le processus
-w fichier	**Fichier**	Si fichier est un fichier qui peut être écrit (write), par le processus
-s fichier	**Fichier**	Si fichier est un fichier de longueur non nulle
-x fichier	**Fichier**	Si fichier est un fichier exécutable
! expression	**Opérateur unaire**	Si l'expression est FALSE
Expression1 –a Expression2	**Opérateur unaire**	Les 2 expressions sont TRUE
Expression1 –o Expression2	**Opérateur unaire**	L'une des 2 expressions (ou les 2) est TRUE

23.1.10 Calcul d'expression avec la commande " **expr** "

Syntaxe : **expr** var1 operateur_expr var2

Opérations possible avec la commande " **expr** "

Opérateur pour la commande " expr "	**Son action**
+	Addition
-	Soustraction
*****	Multiplication
/	Division
%	Modulo

Exemple : a=0 ; **echo `expr $a** + 100`

Notes :
a) **Attention !** , toutes les variables doivent être initialisées dans la commande " **expr** ".
b) Il faut toujours séparer opérandes et opérateurs par un espace.

Exemple :

```
# !/bin/sh
a=1
while [ $a != 10 ]
do
    echo $a'' \c''
    a=`expr $a + 1`
```

done

23.2 La commande " read " :

read var1 : Elle lit ce que vous avez saisi sur votre clavier, dans la variable " var1 ".

24 Les fonctions dans le shell (sous-programmes dans le shell)

Exemple de script :

```
# !/bin/sh

lcd ()
{   cd $1
    PS1="`hostname` : `pwd` $ " ; export $PS1
}
lcd $2
```

l) la commande " **exec** "

exec fd> fic
exec fd< fic
exec < fic
exec > fic
exec > /dev/tty
exec 2> /dev/tty

24.1 Messagerie électronique : commandes « write », « wall » et « mesg » :

Elles permettent de communiquer simplement entre utilisateurs.

write nom_utilisateur : commande de messagerie électronique, permet d'envoyer un message à un utilisateur connecté de nom « nom_utilisateur ». Le message est envoyé ligne par ligne jusqu'à un caractère de fin de fichier et apparaît sur le terminal de l'utilisateur. Exemple :
write belgique
le texte du message
Ctrl-D
wall : le super-utilisateur « **root** » a la possibilité d'envoyer le même message à tous les utilisateurs connectés. Pour cela , il utilise **wall** à la place de **write**.
mesg [**-n**] [**-y**] : Il est possible d'ôter la possibilité de recevoir des messages envoyés par **write** en utilisant la commande **mesg**. Par défaut la permission est accordée.

24.2 Courrier électronique : commandes « mail » et « mailx »

mail [nom_utilisateur] : Envoi d'un message (un mail) vers la boîte aux lettres d'un ou plusieurs utilisateurs. Les messages sont conservés dans les boîtes (contrairement aux messages par **write** qui sont éphémères).
Le mail va dans la boîte de l'usager _ un fichier ayant en général comme nom : **/usr/spool/mail/**nom_utilisateur
L'utilisateur peut envoyer, consulter, détruire des messages.

Syntaxe : **mail** nom_utilisateur
le texte du message
Ctl-D

mail permet de lire sa boîte à lettres. Cet utilitaire affiche un prompt et attend les directives listées ci-après :

RC, +, n	message suivant
-n	message précédent
p	réaffiche le message courant
d	détruit le message courant
dn	détruit le message n

n	affiche le message numéro **n**
s [nom_fichier]	sauve le message dans le fichier de nom "nom_fichier" s'il est fourni , sinon le message est sauvé dans la mail box
w [nom_fichier]	sauve le message dans le fichier de nom "nom_fichier" s'il est fourni , sinon le message est sauvé dans la mail box
q, Ctl-D	sort de mail
?	liste des directives

Cours réalisé par Benjamin LISAN à Paris, janvier 2005.

25 Ajout d'utilisateurs

exemple :

useradd -g 202 -u 2965 -c "Utilisateur Oracle group 202 uid 2965 " -m -s /bin/ksh oracle

26 Gestion des systèmes de fichiers (FS ou File systems)

Montage d'un file système : mount /*repertoire*
Démontage d'un file système : umount /*repertoire*
Ce répertoire est encore appelé « point de montage ».

Exemple :

Montage d'un file système : mount /database
Démontage d'un file système : umount /database

Quand on lance un « umount » et qu'on a comme message d'erreur « *filesystem* busy »
(Par exemple :
[root@polygalee:/] umount /varsoft/xir2
UX:vxfs umount: ERROR: V-3-21705: /varsoft/xir2 cannot unmount : Device busy
)

On doit lancer la commande "fuser / *filesystem* " (avec *filesystem* le système de fichier utilisé par un processus).
Par exemple :

[root@polygalee:/] fuser -c /varsoft/xir2
/varsoft/xir2: 3225o 3077o 24418o 24200o 24166o 24100o 24004o 24000o 23960o 23934o 23844o
[root@polygalee:/] ps -ef|grep 3225 |grep -v grep
xir2 3225 3217 15 17:03:54 ? 161:12 /product/xir2/bobje/enterprise115/solaris_sparc/WIReportServer -name polygalee.

Ensuite, on évalue si l'on doit tuer ou non les processus utilisant le filesystem et empêchant de le démonter.

27 Annexe : Ajout de disque et ajout ou maj d'un file-system (sous volume manager)

1) Ajout d'un volume dans un groupe de disque :

Action	commande	?
Vérification de la machine sur laquelle on est (car on ne sait jamais …)	*hostname* *uname -a*	
1) Affichage de la liste des disques connus	vxdisk list	
2) si les nouveaux disques ne sont pas présents (Faire que les disques nouveaux soient reconnus)	devfsadm [devfsadm -Cv]	
2bis) Il est parfois nécessaire d'arrêter et de redémarrer le "daemon" de VxVM	vxdctl disable vxdctl enable	
3) Labeliser le nouveau disque : remplacer " ?" par les chiffres correspondants ((c)ontrôleur, (t)arget ID et (d)isque)	format c?t?d? Exemple : format c5t9d182	
5) re affichage de la liste des disques connus : le nouveau disque est présent, mais il est "offline"	vxdisk list	
6) affichage de liste des dg	vxdg list	
7) Vérif place libre dans le groupe de disque "nom_dg" (la taille est indiquée entre parenthèses en Mo)	vxassist -g "nom_dg" maxsize	
8) Mettre le disque sous le contrôle de VXvM	/usr/lib/vxvm/bin/vxdisksetup -i c?t?d?	
9) re affichage de la liste des disques connus : le nouveau disque est présent, mais il est "offline"	vxdisk list	
10) ajout du nouveau disque dans le groupe de disque	vxdg -g "nom_dg" adddisk "NOM_VOLUME"=c?t?d? ou vxdg -g "nom_dg" adddisk DIN_"nom_dg"_nn=c?t?d? Exemple : vxdg -g datadg1 adddisk DIN datadg1 04=c5t9d181	
10bis) Vérification	vxdisk disk "nom_dg"	

2) Création d'un système de fichier (FS) :

Action	commande	?
1) création du FS à la taille "NN" Go	vxassist -g "nom_dg" make "point_montage" "NN"G layout=nostripe	
2) déclaration du FS dans /etc/vfstab	vi /etc/vfstab ajout de la ligne : /dev/vx/dsk/c?t?d? /dev/vx/rdsk/c?t?d? "point_montage" ufs 2 yes - Exemple de ligne ajoutée : /dev/vx/dsk/datadg1/varsoft_ora_RCA_arch /dev/vx/rdsk/datadg1/varsoft_ora_RCA_arch /varsoft/oracle/RCA/arch vxfs 3 yes -	
3) création du point de montage	mkdir [-p] "point_montage"	
4) Initialisation droits sur ce point de montage	chown "own:grp" "point_montage" chmod "droits" "point_montage"	
5) construction (création) du FS (newfs pour type ufs, mkfs pour type vxfs)	a) Ufs : newfs /dev/vx/***rdsk***/"point_montage" b) Vxfs : mkfs -F vxfs /dev/vx/***rdsk***/"point_montage" exemple : mkfs **-F vxfs** /dev/vx/***rdsk***/datadg1/varsoft_ora_RCA_arch	
6) Si la taille du système de fichier est > à 2 Go et que le type est vxfs	/opt/VRTSvxfs/sbin/fsadm -o largefiles "point_montage"	
7) Montage du FS	mount **-F vxfs** /dev/vx/dsk/"nom_dg"/"Point_de_montage" (ou bien mount "point_montage")	
7bis) Vérification du montage du FS	df -k "point_montage" cd "Point_de_montage" > toto ls rm toto	
7ter) Autre vérification présence "Point_de_montage"	vxprint -Ath | grep "Point_de_montage"	

3) **Agrandissement ou réduction (VXFS) d'un système de fichiers** :

Action	commande	?
1)	`vxresize -g NOM_DG "Point_de_montage" [+-]taille[mg]` `Exemple : vxresize -g varsoft +10g`	
2) vérification	`df -k "Point_de_montage"`	

A vérifier	`vxdmpadm listctlr c?t?d???`
	`Exemple : vxdmpadm listctlr c5t9d183`
	`vxdmpadm listctlr all`	

A vérifier	`vxassist -g "nom_dg" make "Point_de_montage"  NNNg`
	`(avec NNN = taille en Go)`
	`Exemple : vxassist -g datadg1 make varsoft_ora_RCA_arch 30g`	

	`mountall`	

4) **Suppression d'un système de fichiers** :

1) vérification non utilisation du FS	`du -k "Point_de_montage"`	
2) Démontage du système de fichiers (FS)	`umount "Point_de_montage"`	
3) Suppression du système de fichiers	`vxedit -g NOM_DG -rf rm "Point_de_montage"`	
4) Suppression du point de montage	`rm -r "Point_de_montage"`	

4) **Autres commandes** :

vxprint -g NOM_DG
vxprint -Ath NOM_VOLUME
vxdg deport NOM_DG
vxdg import NOM_DG
vxvol -g NOM_DG startall
vxassist -g NOM_DG maxsize : permet d'obtenir la taille de l'espace encore libre dans le DG
vxdiskadd c#t#d#
vxdg init NOM_DG NOM_VOLUME=c#t#d#s2
vxtask list
vxdg [-g NOM_DG] spare
/usr/lib/vxvm/bin/vxdiskunsetup -C c1t1d0

vxlicrep

/usr/lib/vxvm/bin/vxdisksetup -i c1t0d0 format=sliced