Aide-mémoire Unix : Commandes de base

Table des matières de chapitre

Note : Unix comporte plus de 200 commandes ! Seules les commandes les plus importantes pour l'utilisateur courant sont décrites ici. De même, nous ne décrivons pour celles-ci que les options présentant le plus d'intérêt. Pour davantage de détails, il est vivement conseillé de consulter le "manuel" Unix (frapper : "`man` *`commande`*").

---

1. Liste des commandes et correspondances Unix - VMS - DOS

Commande	UNIX	VAX/VMS	MS-DOS
********	****	*******	******
Début et fin de session, mot de passe			
Fichier prologue utilisateur	.cshrc	login.com	autoexec.bat
Changer mot de passe	passwd	set password	-
Fin de session	exit, logout	logout	-
Outils d'aide et d'information			
Aide en ligne	man, apropos	help	help
Documentation complète	AnswerBook(Sun)	BookReader	<F1> dans DOS-Sh
News	xrn, tin	news	Trumpet
Rappel et édition des commandes, alias			
Rappel des commandes	history	recall/all	-
Alias (symbole)	alias nom cmd	nom :== cmd	-
Répertoires (directoires)			
Afficher contenu répertoire	ls	directory	dir, tree
Changer répertoire courant	cd	set default	cd, chdir

Nom répertoire courant	pwd	show default	
Créer répertoire	mkdir	create/directory	mkdir, md
Détruire répertoire	rmdir	delete	rmdir, rd

Visualisation de fichiers

Affichage contenu du fichier	cat	type	type
	more	type/page	more
Concaténation de fichiers	cat f1 f2 > f3	copy f1,f2 f3	
Affichage octal/hexadéc.	od	dump	

Manipulation de fichiers, occupation disque

Copie d'un fichier	cp	copy	copy, xcopy
Renommer/déplacer fichier	mv	rename	ren, rename
Détruire fichier	rm	delete	del, erase
Changer protection fichier	chmod	set file/prot	attrib
Protection par défaut	umask	set prot/default	-
Changer appartenance fichier	chown, chgrp	set file/owner	-
Recherche fichier	find	directory	-
Place disque utilisateur	du	dir/size=all	-
Espace disque	df	show dev	chkdsk

Comparaisons, recherches, tris

Comparaison de 2 fichiers	diff, cmp	difference	comp
Recherche chaîne dans fich.	grep, egrep	search	find
Tri contenu d'un fichier	sort	sort	sort

Impression, formatage

Impression	lpr, lp	print	print, lpr
Etat queue d'impression	lpq, lpstat	show queue	lpq
Avorter impression	lprm, cancel	delete/entry	lprm

Processus, jobs et traitement "batch"

Liste des processus	ps	show sys,sh proc	-
Destruction processus	kill	stop/id	-
Changement priorité process	nice, renice	set process/prio	-
Soumission différée job	at, bg	submit	-
Commande en arrière-plan	commande &	spawn/nowait	-
Liste jobs courants	jobs	show queue/batch	-
		show process/sub	-
Avorter job courant	<ctrl-C>	<ctrl-Y>	<ctrl-C>

Usage des périphériques de stockage (disquettes, cassettes, CD-ROM...)

Ecriture sur bande/cassette	tar, cpio, dump	backup, copy	backup, restore

Messagerie et communication inter-utilisateurs

Messagerie	mail, mailtool	mail	Eudora...
Communication interactive	talk, irc, write	phone	-
Affichage liste utilisateurs	rusers, finger	show user/node	-
Transfert de fichiers	ftp,ftptool, rcp	ftp, copy	ftp
Remote session	rlogin, telnet	set host, telnet	telnet
Sortie sur port série	tip	set host/dte	

Commandes diverses

Affichage date et heure	date	show time	date, time
Affichage liste utilisateurs	who, w, users	show user	-
Affichage à l'écran	echo	write sys$output	echo
Effectue pause	sleep	wait	-
Settings terminal	stty, tset	set terminal	mode
Ressources autorisées	limit	show process/all	-
Usage mémoire	vmstat	show memory	mem
Usage réseau	netstat	show network	-

Redirection et tube

Redirection	>	/OUTPUT=	>
	<	/INPUT=	<
Tube	|	(fich. interm.)	|

Variables

Var. locales : définition	set var=val	var := val	set
" " : effacement	unset var	var = ""	
" " : liste	set	show symbol	
Var. d'environ. : déf.	setenv VAR val	define LOG val	-
" " : effac.	unsetenv VAR	deassign LOG	-
" " : liste	printenv	show logical	-
" " : déf.		:==	-

Shell-scripts

Exécution script (procédure)	source fichier	@fichier{.com}	fichier{.bat}
Commentaire	#comment	!comment	REM comment

Éditeurs

Éditeurs	vi, emacs	edt, lse	edlin
Créer fichier au vol	cat > fichier	create fichier	-

Compilateurs

Compilation Pascal	pc	pascal	...
" Fortran	f77	fortran	...
" C	cc, gcc	cc	...
Édition de lien	pc, f77, cc, ld	link	...
Gestion librairies	ar, ranlib	library	...
Exécution programme	program	run program{.exe}	program{.exe}
Debugger	dbx, dbxtool	debug	...

---

2. Répertoires (directoires)

`ls {`*`fichier(s)`*`}`
`ls {`*`path(s)`*`}`

Liste le contenu du répertoire courant {ou le(s) *fichier(s)* ou *répertoire(s)* spécifiés} [*list files*]
`-a` = tous les fichiers, y compris hidden-files (fichiers dont le nom commence par ".")
`-F` = identifie les fichiers en ajoutant "/" aux noms de dir., "*" aux exécutables, "@" aux liens symboliques

`-l` = listage long : droits d'accès (voir commande `chmod`), liens, propriétaire, taille [bytes], date et heure de modification
`-lu` = comme `-l` mais indique date/heure de dernier accès (au lieu de modification)
`-t` = listage trié en fonction de la date de modification
`-g` = indique le groupe auquel le fichier est rattaché
`-R` = liste en parcourant récursivement tous les sous-répertoires
`-1` = listage à raison d' 1 seul fichier par ligne
`-i` = indique No de i-node
`-b` = affiche les car. non imprimables de noms de fichier en octal
`-ld` *`répertoire(s)`* = affiche informations sur *répertoire(s)* spécifié(s) (et non pas sur leur contenu)

Exemple de sortie de la commande "`ls`" :

```
[123]dupond@dgrsun15:/USERS/dupond> ls -lgF
total 184
-rwxr-x--x    1   dupond  cy_1_2   188416    Jun  9 13:35  prog*
-rwxr-x--x    1   dupond  cy_1_2      436    Jun  9 13:29  prog.p
drwxr-xr-x    6   dupond  cy_1_2      512    Jun  5 10:24  projet1/
-rw-rw-r--    1   dupond  cy_1_2       61    Jun  6 17:54  tata
lrwxrwxrwx    1   dupond  cy_1_2        7    Jun  7 17:53  titi -> tata

<-------->   <->  <---->  <---->   <---->    <---------->  <-------->
Permissions   Nb.  User   Group    Taille     Date/heure    Nom de
(tuuugggooo)  liens                (bytes)    dern.modif.   fichier

t(ype)= -(fichier ordin.)  u(ser) = droit accès à l'égard utilisateur
      = d(irectoire)       g(roup)=   "     "   "    "    groupe
      = l(ien symbolique)  o(ther)=   "     "   "    "    des autres
```

`cd {`*`répertoire`*`}`
Change de répertoire courant [*change directory*]. Si on ne donne pas de paramètre, renvoie dans le répertoire principal de l'utilisateur (home)

`cd -`
Revient dans répertoire précédent (commande valable uniquement sous le T-shell)

`pwd`
Affiche sur sortie standard le chemin d'accès (path absolu) du répertoire courant [*path of working directory*]

`mkdir` *`répertoire(s)`*
Crée *répertoire(s)* de nom(s) spécifié(s) [*make directory*]
`-p` = crée les répertoires parents s'ils n'existent pas encore (ex: "`mkdir -p tata/titi`" crée sous-répertoire "tata" puis crée "titi" dans "tata")

`rmdir` *`répertoire(s)`*
Détruit le(s) *répertoire(s)* spécifié(s) (dont le contenu doit avoir préalablement été détruit) [*remove directory*]. Voir aussi la commande "`rm -r`" pour détruire répertoire non vide

---

3. Visualisation de fichiers

`more {`*`fichier(s)`*`}`

Affichage contrôlé de l'entrée standard ou du contenu du(des) *fichier(s)* texte spécifiés. (La commande System V plus ou moins équivalente est `pg`).
Avancement dans le fichier contrôlé par les **commandes** :
`<SPACE>` = page suivante
{*n*}`<CR>` = ligne suivante {ou n-ème ligne suivante}
{*n*}`b` = page précédente {ou n-ème page précédente}
`'` = revient au début du fichier
`/chaîne` = cherche la chaîne de caractère spécifiée (en fait "expression régulière")
`n` = cherche la prochaine occurence de cette chaîne
`=` = affiche No de ligne courante
`.` = répète la dernière commande
`v` = entre dans l'éditeur `vi` et se positionne sur la ligne courante
`:n` = passe au fichier suivant
`:p` = revient au fichier précédent
`!`*`commande`* = exécute *commande* Unix
`q` = termine affichage (sortie de more)
`?` ou `h` = aide en ligne sur l'utilitaire more

`cat` *`fichier(s)`*
Ecrit contenu de(s) *fichier(s)* sur sortie standard
`-n` = numérote les lignes
`-v` = remplace caractères de contrôle (non imprimables) par notation "`^x`" (voir aussi commande "`od -c`")

`cat` *`fichier1 fichier2 fichier3`* `>` *`fichierf`*
Concatène fichiers 1, 2 et 3 dans *fichierf* [*catenate*]

`cat >` *`fichier`*
Création d'un *fichier* par introduction de données au clavier. Terminer avec `<CR>` et `<ctrl-D>`

`head {-`*`n`*`}` *`fichier(s)`*
Ecrit les 10 {ou *n*} premières lignes de(s) *fichier(s)* sur sortie standard

`tail {-`*`n`*`}` *`fichier(s)`*
Ecrit les 10 {ou *n*} dernières lignes de(s) *fichier(s)* sur sortie standard
`+`*`n`* = à partir de la *n*-ème ligne jusqu'à la fin du fichier

Exemple : "`head -10` *`fichier`* `| tail -4`" : affiche *fichier* de la 7e à la 10e ligne

`od {`*`fichier`*`} {`*`offset`*`}`
Affiche sur sortie standard *fichier* binaire à partir du début {ou depuis *offset* spécifié} [*octal dump*]. Voir aussi commande "`cat -v`".
Quelques type d'affichage possibles (on peut les combiner) :
`-b` = affiche bytes en octal
`-c` = affiche bytes en ASCII
`-o` = affiche mots de 16 bits en octal (défaut)
`-x` = affiche mots de 16 bits en hexa.
`-O` = affiche mots de 32 bits en octal
`-X` = affiche mots de 32 bits en hexa.
Syntaxe de l'*offset* :
`offset` = en octal
`offset``.` = en décimal
`offset``b` = en blocs 512 bytes

`strings {`*`fichier`*`}`
Recherche dans *fichier* objet ou binaire les chaînes de caractères imprimables

4. Manipulation de fichiers, occupation disque

Copie, déplacement, destruction

`cp {-ip}` *`fich_source fich_dest`*
Copie *fichier source* sur *fichier destination* [*copy*]
`-i` = demande confirmation avant d'écraser destination (option activée par défaut au DGR)
`-p` = préserve date de modification et protection sur *fichier destination*

`cp {-ip}` *`fichier(s) répertoire`*
Copie *fichier(s)* dans *répertoire*

`cp {-rip}` *`rép_source rép_dest`*
Copie *répertoire source* dans *répertoire destination*
`-r` = copie également récursivement tous les sous-répertoires éventuels

`mv {-i}` *`fich_source fich_dest`*
Renomme fichier
`-i` = demande confirmation avant d'écraser destination (option activée par défaut au DGR)

`mv {-i}` *`fichier(s) répertoire`*
Déplace *fichier(s)* dans *répertoire* [*move*]

`mv {-i}` *`rép_source rép_dest`*
Renomme répertoire ou le déplace dans autre répertoire

`rm {-i}` *`fichier(s)`*
Détruit *fichier(s)* [*remove*]. Avec `-i`, demande confirmation à l'utilisateur (option activée par défaut au DGR).
Note : "`rm *`" détruira tous les fichiers ne commençant pas par "`.`". Pour encore détruire tous les "hidden-files" du dir. courant, faire "`rm .??*`" ou "`rm .[!.]*`" , mais pas "`rm .*`"

`rm -r{i}` *`répertoire`*
Détruit *répertoire* et, récursivement, tout son contenu (fichiers et sous-répertoires)

Pour détruire un fichier dont le nom commence par le caractère "`-`" sans que ce caractère soit interprété comme le caractère introduisant les options de la commandes, procéder de la façon suivante : p.ex. pour détruire le fichier "`-toto`" faire "`rm ./-toto`"

Protections

`chmod` *`mode fichier(s)`*
`chmod {-R}` *`mode répertoire`*
Change les droits d'accès à *fichier(s)* ou *répertoire* [*change mode*]
Avec `-R` : récursivement dans toute l'arborescence sous *répertoire*
Visualisation des droits d'accès avec commande "`ls -l`"

On a 2 façons d'exprimer le *mode* :

A) La notation **symbolique** de *mode* est : *`qui=permission`* (affecter droit) ou *`qui+permission`* (ajouter droit) ou *`qui-permision`* (enlever droit), où :

- *qui* = `{u}{g}{o}{a}` où :
  u=user (propriétaire), g=groupe, o=others (les autres), a=all (tous)
- *permission* = `{r}{w}{x}` où :

- pour un fichier : r=read (voir contenu), w=write (modifier ou détruire), x=execute (exécuter script ou programme)
- pour un répertoire : r=voir contenu, w=créer ou détruire fichiers, x="traverser" le répertoire (accéder à ce qui s'y trouve)

Exemples de *modes* : `u+x` (pour rendre exécutable shell script), `go-rw` (pour cacher accès aux autres) ou encore `u=rw,g=r`

B) La notation **octale** de *mode* est : *ugo* où :
u, g, o sont des valeurs octales de 0 à 7 définissant droits d'accès à l'égard de l'**u**tilisateur, du **g**roupe et des autres (**o**thers) selon la règle :

- 0 = aucun droit
- 1 = exécution (x)
- 2 = écriture (w)
- 4 = lecture (r)
- *addition* = combinaison

Exemple : mode `751 = "rwxr-x--x"` car :
"`rwx`" = 4 + 2 + 1 = **7**, "`r-x`" = 4 + 1 = **5**, "`--x`" = 1 = **1**

Autres manipulations de fichiers

`touch` *`fichier(s)`*
: Met à la date/heure courante la date/heure de modification du(des) *fichier(s)*. Crée *fichier(s)* vide(s) s'il(s) n'existe(nt) pas

`file` *`fichier(s)`*
: Identifie le type des *fichier(s)* en fonction du contenu. Cette commande se base sur le "magic file" (`/etc/magic`) définissant les différents types possibles. Reconnait notamment au DGR : ascii texte, données, répertoire, shell-script, fichier PostScript, fichier TIFF, fichier SunRaster, fichier vide, lien symbolique...

`split {-`*`n`*`} {`*`fich_entrée`*`} {`*`fich_sortie`*`}`
: Découpe l'entrée standard {ou *fich_entrée*} en segments de 1000 {ou *n*} lignes en créant fichiers de noms *`fich_sortie`*`aa`, *`fich_sortie`*`ab`, etc... Voir aussi commande `csplit` où l'on peut fixer critère de découpage basé sur "pattern" !

`cut`
`paste`
: Plutôt que de présenter ces commandes de manipulation de champs, nous décrirons plus loin la commande `awk` (beaucoup plus puissante)

`ln -s` *`fichier_ou_répertoire lien`*
: Crée un *lien* symbolique sur le *fichier* ou *répertoire* spécifié [*link*]. Sans `-s` (déconseillé), créerait un lien physique (hard link) au lieu de symbolique. Le nom du fichier pointé par le lien apparaît lorsque l'on passe la commande "`ls -l`".
"`rm` *`lien`*" détruit le lien symbolique (mais pas le fichier pointé par le lien)

`wc {`*`fichier`*`}`
: Compte le nombre de lignes, de mots et de caractères du *fichier* spécifié (ou de l'entrée standard) et l'écrit sur la sortie standard [*word count*]
`-l` = n'affiche que le nombre de lignes

`-w` = n'affiche que le nombre de mots
`-c` = n'affiche que le nombre de caractères

`find` *`répertoire(s) condition(s)`*
Recherche dans toute l'arborescence de chaque <<I>répertoire spécifié les fichiers satisfaisant au(x) *condition(s)* donnée(s). En faisant précéder la condition de `\!` on recherche les fichiers ne satisfaisant **pas** la condition donnée.
Quelques **conditions** possibles (plusieurs critères peuvent être combinés) :
`-name` *`'fichiers'`* = recherche *fichiers* de noms spécifiés
`-user` *`username`* = fichiers appartenant à utilisateur *username*
`-type` *`type`* = fichiers de *type* donné : `d`= directory, `f`= fichier, `l`= lien symbolique...
`-mtime` *`n`* = fichiers modifiés depuis *n* jours
`-perm` *`nnn`* = fichiers dont la protection est *nnn* (mode en notation octale)
`-print` = retourne noms des fichiers et des répertoires satisfaisant condition(s) avec leur path complet
`-exec` *`commande`* `{} \;` = applique *commande* sur tous les fichiers satisfaisant aux critères de recherche
`-ok` *`commande`* `{} \;` = idem sauf qu'il y a demande de confirmation à l'utilisateur pour chaque fichier avant l'exécution de *commande*

Exemple: "`find . -name '*.txt' -print`" : écrit sur sortie standard noms des fichiers ayant l'extension `.txt` dans toute l'arborescence courante

Utilisation de l'espace disque

`du` {*`répertoire(s)`*}
Affiche sur sortie standard l'espace-disque utilisé par toute l'arborescence courante {ou par celle(s) spécifiée(s)} [*directory usage*]
`-a` = affiche non seulement les répertoires mais aussi les fichiers
`-s` = n'affiche que le grand total pour tous les répertoires spécifiés
`-k` = cette option sous Solaris 2 affiche les tailles en Kbyte (exprimées sinon en blocs de 512 bytes)
Passée dans votre home, la commande "`du -sk`" vous indique donc la place disque totale que vous occupez

`quota -v`
Affiche l'usage de l'espace-disque par l'utilisateur (maximum autorisé et espace utilisé) sur chaque file-system

`df -k` {*`file_system ou répertoire`*}
Affiche sur sortie standard l'espace-disque libre et utilisé par tous les utilisateurs sur tous les file systems montés {ou sur celui spécifié}, qu'ils soient locaux ou distants (NFS) [*disk free*].
L'option `-k` sous Solaris 2 affiche les tailles en Kbyte (exprimées sinon en blocs de 512 bytes).

---

5. Comparaisons, recherches, tris

Comparaisons

`diff` *`fichier1 fichier2`*
Compare ligne par ligne le contenu des fichiers texte *fichier1* et *fichier2*, et écrit les différences sur la sortie standard. Préfixe par "<" les lignes de *fichier1* et par ">" celles de

fichier2
`-i` = ne distingue pas majuscules et minuscules
`-w` = ignore différences d'espacement (n espaces ou <TAB> sont équivalents à 1 espace)
`-e` = produit script de commandes permettant de recréer avec `et` *fichier2* à partir de *fichier1* ("patch")

`diff {-r} répertoire1 répertoire2`
Compare le contenu de 2 répertoires et le contenu des fichiers portant le même nom {récursivement dans tous les sous-répertoires}

`cdiff fichier1 fichier2`
"Word context diff" (version plus perfectionnée de diff, spécifique au DGR). Utiliser options :
`-B` = data char-by-char context
`-v` = force usage effets vidéo pour mise en évidence différences

Voir aussi commandes `diff3` (pour comparaison de 3 fichiers), `comm` (pour comparaison avec affichage multi-colonne), `cmp` (pour comparaison de fichiers binaires), `dircmp`, `uniq`

Recherche de chaînes

`grep 'motif' {fichier(s)}`
Ecrit sur sortie standard les lignes de l'entrée standard {ou de *fichier(s)*} contenant chaîne de caractères satisfaisant à expression régulière *motif* [*global regular expression print*]. Voir aussi commande `egrep` plus riche [*extended grep*] et `fgrep`
`-i` = ne distingue pas majuscules et minuscules
`-v` = affiche toutes les lignes ne satisfaisant pas la condition
`-n` = affiche aussi No de ligne
`-l` = n'affiche que le nom des fichiers où chaîne a été trouvée
`-c` = affiche uniquement nombre de lignes contenant chaîne
`-h` = n'affiche pas le nom de fichier, normalement placé devant chaque ligne

Exemple : "`egrep '(Jules|Jim)' fichier`" : affiche toutes les lignes de *fichier* contenant "Jules" ou/et "Jim"

Voir la commande `strings` pour la recherche de chaînes ASCII dans fichiers binaires.

Vérification orthographique

`look chaîne {fichier_trié}`
Affiche tous les mots du dictionnaire anglais `/usr/dict/words` {ou du *fichier_trié*} contenant *chaîne*

`spell {fichier}`
Vérification orthographique en anglais : affiche les mots de l'entrée standard {ou de *fichier*} ne se trouvant pas dans dictionnaire. Nombreuses options (voir man-page). Il existe aussi `ispell` (version internationale et interactive de `spell`)

Tris

`sort {fichier_entree} {> fichier_sortie}`
Trie alphabétiquement (selon table ASCII) ligne par ligne l'entrée standard {ou le contenu de *fichier_entree*} et envoie résultat sur sortie standard {ou sur le *fichier_sortie*}
`-n` = trie numériquement plutôt qu'alphabétiquement ("10" viendra après "2")
`-r` = trie dans ordre inverse
`-f` = ne distingue pas majuscules et minuscules
`+n {-m}` = comme clé de tri, ignore les *n* premiers champs (les champs sont délimités par 1 ou

plusieurs espaces) {et trie jusqu'au *m*-ème champ plutôt que jusqu'à la fin de la ligne}
`-t`*x* = considère le caractère "*x*" comme séparateur de champs

---

6. Impression, formatage

Cliquer ici pour voir la liste des imprimantes accessibles au DGR. La plupart acceptent des fichiers de type : texte, PostScript, SunRaster et TIFF (détection/filtrage automatique du type et conversion en PostScript par l'architecture d'impression NeWSprint mise en oeuvre par le SI-DGR).

Sans spécifier d'imprimante dans la salle de station du DGR, les stations impaires impriment sur l'imprimante laser "salle1" et les stations paires sur "salle2".

Impression sous Unix System 5 (Solaris...)

`lp {-d` *imprimante*`} {`*fichier(s)*`}`
Insère l'entrée standard ou le(s) *fichier(s)* sur la file d'attente de l'*imprimante* spécifiée [*line printer*]. Si aucune *imprimante* n'est désignée, utilise celle définie par l'utilisateur par la variable d'environnement `LPDEST` (ou l'imprimante par défaut définie par l'administrateur)
`lpstat {`*imprimante*`}`
Indique *Id* (sous la forme *imprimante-No*) et statut des requêtes d'impression en attente sur toutes les imprimantes (ou l'imprimante spécifiée) [*line printer status*]
`cancel` *Id(s)*
`cancel -u` *username*
Avorte la(les) job(s) d'impression spécifié(s) par *Id(s)*,
ou avorte tous les jobs d'impression soumis par l'utilisateur *username*

Impression sous Unix BSD (SunOS...)

`lpr {-P`*imprimante*`} {`*fichier(s)*`}`
Insère l'entrée standard ou le(s) *fichier(s)* sur la file d'attente de l'*imprimante* spécifiée [*line printer*]. Si aucune *imprimante* n'est désignée, utilise celle définie par l'utilisateur par la variable d'environnement `PRINTER`
`lpq {-P`*imprimante*`}`
Indique *No* et statut des requêtes d'impression en attente [*line printer queues*]
`-l` = fourni informations étendues
`lprm {-P`*imprimante*`} {`*No*`}`
Avorte impression *No* [*line printer remove*]. Si *No* n'est pas spécifié, avorte la dernière impression soumise par l'utilisateur
`-` = le signe "-", à la place de *No*, avorte toutes les impressions en cours de l'utilisateur
`lpstat -o`
Affiche statuts de toutes les queues d'impression

Voir aussi la commande `lpc` [*line printer control*]

Formatage

`pr {-options} {fichier}`
Formate *fichier* selon *options* en vue d'une impression [*prepare*]. Sans options, découpe le fichier en pages et ajoute date, nom de fichier et No de page en en-tête de chaque page.
`-`*n* = formatage en *n* colonnes
`-h "`*en_tête*`"` = remplace nom du fichier par texte *en_tête*

`-l`*n* = crée des pages de *n* lignes
Exemple d'utilisation en combinaison avec lpr : "`pr fichier | lpr`"

Autres formateurs : voir commandes `nroff` et `troff`

---

7. Compression et autres transformation de fichiers

Compression, respectivement décompression

Pour être le plus universel, compresser avec `compress` et décompresser avec `gunzip`. Attention : les fichiers compressés sont des fichiers binaires qu'il faut encore encoder avec `uuencode` si l'on veut les envoyer par le canal de la messagerie électronique !

`compress {`*`fichier`*`}`
Compresse *fichier* (algorithme Lempel-Ziv, réduction de taille de l'ordre de 50%). Le résultat est un fichier de nom *`fichier`*`.Z`, et le *fichier* original disparaît du répertoire. Sans paramètre, travaille sur entrée standard et sortie standard.
`-v` = affiche pourcentage de compression
`compress -c` *`fichier`* `>` *`fichier`*`.Z`
Idem, sauf que le *fichier* original est préservé
`uncompress` *`fichier`*`{.Z}`
Décompresse fichier *fichier*.Z compressé avec `compress`. Le résultat est un fichier de nom *fichier*, et le *fichier*.Z disparaît du répertoire
`uncompress -c` *`fichier`*`{.Z} >` *`fichier`* **ou**
`zcat` *`fichier`*`{.Z} >` *`fichier`*
Idem, sauf que *fichier*.Z est préservé

`gzip` *`fichier(s)`*
Utilitaire de compression du GNU. Compresse *fichier(s)* (algorithme Lempel-Ziv, mais dans autre format qu'avec utilitaire `compress`) sur fichier(s) de nom(s) *`fichier`*`.gz`, et le(s) *fichier (s)* original(aux) disparaît(ssent) du répertoire
`-v` = affiche pourcentage de compression
`-l` = indique contenu du fichier compressé *fichier*.gz
`-h` = affiche liste des options de gzip
`-9` = la meilleure (mais la plus lente) des compressions
`gzip -c` *`fichier`* `>` *`fichier`*`.gz`
Idem, sauf que le *fichier* original est préservé
`gunzip` *`fichier(s)`*`{.gz}`
Utilitaire universel de décompression du GNU pour fichiers compressés avec `gzip`, mais aussi capable de traiter des fichiers compressés avec `zip, compress` et `pack` (la détection est automatique). Avec la commande ci-dessus, décompresse *fichier(s)*.gz sans conserver fichier (s) compressé(s)
`-v` = mode verbose
`-h` = affiche liste des options de gunzip
`gunzip -c` *`fichier`*`{.gz} >` *`fichier`*
Idem, sauf que *fichier*.gz est préservé
`unzip` *`fichier`*`{.zip}`
Décompression de *fichier*.zip provenant du monde MS-DOS (compressés avec `pkzip`)

`pack` *`fichier(s)`*
Utilitaire de compression System V. Compresse *fichier(s)* (algorithme Huffman, moins

efficace que Lempel-Zif) sur fichier(s) de nom(s) *fichier*.**z**, et le(s) *fichier(s)* original(aux) disparaît(ssent) du répertoire
- = affiche statistiques de compression

`unpack` *fichier(s)*`{.z}`
Utilitaire de décompression System V pour fichiers compressés avec `pack`. Décompresse *fichier(s)*.z sans conserver fichier(s) compressé(s)

`pcat` *fichier*`{.z} > `*fichier*
Idem, sauf que *fichier*.z est préservé

`sh` *fichier*`.shar`
Décompression d'une "shell-archive" (fichier composite qui est en fait un Bourne-shell script). Attention à examiner le contenu de l'archive **avant** de la déballer (il peut s'agir d'un fichier malicieux du type "Cheval de Troie" !). Voir aussi outils `shar` et `unshar` du GNU.

Encodage/décodage binaire <-> hexadécimal

Un tel encodage est nécessaire si l'on veut envoyer un fichier binaire (exécutables, images...) ou contenant des caractères 8 bits (car. accentués) par le canal de la messagerie qui n'accepte que des fichiers ASCII 7-bits. Il faut ensuite les décoder à la réception.

`uuencode` *fichierA fichierB* `> `*fichierC*`.uu`
Encode *fichierA* binaire sur *fichierC*.uu hexadécimal. Dans *fichierC*.uu se trouve enregistré le nom *fichierB* sous lequel `uudecode` rechargera le fichier binaire (nom *fichierB* qui peut être différent de *fichierA* mais que l'on nomme le plus souvent de façon identique). L'expansion en taille du fichier est d'environ 35%

`uudecode` *fichierC*`.uu`
Decode *fichierC*.uu hexadécimal et crée fichier binaire de nom *fichierB* défini dans l'en-tête du fichier encodé. `uudecode` ne prend en considération que ce qui se trouve entre la ligne `begin` et la ligne `end` du fichier encodé et saute tout le reste.

Exemple complet :

On pourrait bien évidemment "piper" entre elles plusieurs des commandes ci-dessous...

1) `compress -c prgm > prgm.Z`
Compresse fichier binaire prgm sur prgm.Z en laissant intacte prgm
2) `uuencode prgm.Z prgm.Z > prgm.Z.uu`
Encodage de prgm.Z sur fichier ASCII prgm.Z.uu. On pourrait ici éventuellement fractionner prgm.Z.uu en plusieurs morceaux avec commande `split`...
3) `mail -s "Sujet..." dupond@dgr.epfl.ch < prgm.Z.uu` **puis**
`rm prgm.Z prgm.Z.uu`
Envoie prgm.Z.uu à dupond puis détruit fichiers devenus inutiles
4) `mail`, "`r` *No*", "`s message.uu`", "`d` *No*", "`q`"
Lecture du mail, puis sauvegarde sur fichier, puis destruction du message et sortie de mail
5) `uudecode message.uu`
Décodage de message.uu sur prgm.Z
6) `uncompress prgm` **puis**
`rm message.uu`
Décompression de prgm.Z sur prgm puis destruction des fichiers devenus inutiles

Ecriture, resp. lecture d'un fichier-archive "tar"

Lorsque l'on veut transmettre un répertoire entier voire toute une arborescence, on crée généralement

un fichier d'archive "tar". Il s'agit d'un fichier binaire que l'on peut encore compresser (et encoder).

`tar cvf . fichier.tar`
Ecrit toute l'arborescence courante sur *fichier*.tar [*tape archive*]
`tar tf fichier.tar`
Affiche le contenu de l'archive *fichier*.tar
`tar xvf fichier.tar`
Charge contenu de l'archive *fichier*.tar dans le répertoire courant.
A la place de "`uncompress fichier.tar.Z`" suivi de "`tar xvf fichier.tar`", on peut aussi plus simplement faire "`zcat fichier.tar.Z | tar xvf -`"

---

8. Processus, jobs et traitement "batch"

Généralités

Un **processus** est l'instance d'une commande ou d'un programme en cours d'exécution. Chaque processus est exécuté par le système de façon concurrente et indépendamment des autres processus tournant sur la même machine (notions de "multi-tâche" et "temps partagé"). Le processus est confiné dans un espace-mémoire qui lui est propre (notion de protection mémoire), mais les processus peuvent communiquer entre eux ou avec les périphériques et le réseau par l'intermédiaire d'appels système.

Pour toute commande lancée par l'utilisateur, le shell (lui-même un processus) "fork" un nouveau processus, sauf s'il s'agit de commandes internes au shell (built-in) qui sont directement exécutées par le shell.

Les processus, comme les fichiers, appartiennent à celui qui les a créés, et l'utilisateur ne peut donc manipuler, avec les commandes décrites ci-dessous, que ses propres processus. Il existe un certain nombre de processus-système, appelés **daemons**, qui font "vivre" Unix (paging/swapping, réseau, impression...); seul l'administrateur système peut agir sur ces processus (de même que sur ceux de tous les utilisateurs).

Lorsque le nombre de processus augmente, les performances diminuent et les temps de réponse augmentent. Il peut arriver que le nombre de processus soit trop élevé par rapport à la taille de la mémoire vive; le système évacue alors temporairement sur disque tout le contexte-mémoire des processus les moins actifs (**swap**).

Unix utilise aussi la technique de la **mémoire virtuelle** qui permet l'exécution de programmes dont la taille est supérieure à celle de la mémoire vive disponible. Le principe est de découper le programme en "pages" de taille fixe et ne charger en mémoire que les pages nécessaires, à un instant donné, pour l'exécution du programme, les autres pages non utilisées étant temporairement déchargées sur disque (**pagination**).

Un **job** est un travail créé et contrôlé par un shell. Il peut être composé de plusieurs processus. Exemple : "`ls -l | more`" : job composé de 2 processus différents reliés par un "tube".

Attention : la plupart des possibilités et commandes ci-dessous sont propres au C-shell et T-shell !

Manipulation de processus

`ps {PID}`

Ecrit sur la sortie standard les informations concernant les processus {ou le processus de *PID* spécifié}, notamment le **PID** (processus ID : No unique dans tout le système), son état, le temps CPU utilisé, la commande exécutée
Sous Unix BSD (SunOS...) :
`-a` = liste tous les processus (sinon, seulement ceux de l'utilisateur)
`-u` = fourni des informations étendues
`-x` = inclu processus non liés à un terminal (p.ex. graphiques)
Exemple : "`ps -aux | grep $USER`" affiche tous les processus de l'utilisateur
Sous Unix System V (Solaris 2...) :
`-elf` = plus ou moins équivalent aux options `-aux` en Unix BSD

`kill -9 PID`
Tue processus de *PID* spécifié (en fait lui envoie le signal d'arrêt inconditionnel SIGKILL)

`at -f {-c} {-s} script heure {date}` **ou**
`commande > fichier_sortie | at heure {date}`
Exécute *commande* ou *script* à *heure* {et *date*} spécifiée. Si *commande* génère des résultats sur sortie standard, il faut rediriger la sortie sur *fichier_sortie*
`-c` = exécuter script par C-shell
`-s` = exécuter script par Bourne-shell

Voir aussi commande `crontab` pour exécution de travaux à intervalles réguliers (horaires quotidien, hebdomadaire, mensuel...)

Manipulation de jobs

`commande`
Exécute *commande* en avant-plan (foreground). Le shell lui-même est suspendu
`commande &`
Exécute *commande* en arrière-plan (background). Elle s'exécute donc de façon "détachée" concurremment avec le shell. Le job en question peut survivre au shell et à la session à partir desquels il a été lancé
(Identique à la séquence : "`commande <CR> <ctrl-Z> bg`)
`(commande1 ; commande2) &`
Exécute séquentiellement *commande1* et *commande2* en arrière-plan

`<ctrl-C>`
Avorte le job courant (celui qui s'exécute en avant-plan) et fait revenir le shell en avant-plan
`<ctrl-Z>`
Suspend le job courant (celui qui s'exécute en avant-plan) (état "stopped") et fait revenir le shell en avant-plan

`jobs`
Liste les jobs du shell courant (suspendus ou s'exécutant en arrière-fond) avec leurs **jobID** (numérotés dans le cadre du shell auquel ils sont rattachés) [*jobs status*]
`-l` = affiche encore le PID
`bg {%jobID}` **ou**
`%{jobID} &`
Continue en arrière-plan [*background*] le dernier job suspendu {ou celui de *jobID* spécifié}
`fg {%jobID}` **ou**
`%{jobID}`
Ramène en avant-plan [*foreground*] le dernier job détaché ou suspendu {ou celui de *jobID* spécifié}

`stop %jobID`
Suspend le job de *jobID* spécifié
`kill {-9} %jobID`
Tue le job de *jobID* spécifié (avec `-9`, arrêt inconditionnel)

`wait {jobID}`
Attend que tous les jobs s'exécutant en arrière-plan {ou uniquement celui de *jobID* spécifié} se terminent

On ne peut pas manipuler les jobs depuis un autre shell que celui auquel ils sont rattachés (shell depuis lequel ils ont été lancés).

---

9. Usage des périphériques de stockage (disquettes, cassettes, CD-ROM...)

Généralités

De façon générale, les outils de manipulation de disquettes (3"1/2), cassettes (Exabyte 8mm, QIC 1/4"...), CD-ROM... sous Unix sont nombreux et en général incompatibles entre eux. Pour lire un média, il est donc important de savoir avec quel outil (commande) il a été écrit. Les commandes principales sont :

- `tar` [*tape archive*]
- `bar`
- `cpio` [*copy input output*]
- `dump/restore` (il faut être utilisateur `root`)
- `cp`
- ...

Utilitaires "mntdisk" pour disquettes 3"1/2 et CD-ROM

Une série d'utilitaires, sous la dénomination "mntdisk", sont installés au DGR pour manipuler (formater, monter... sans devoir être utilisateur "privilégié") des **disquettes 3"1/2 HD** (haute densité 1.4 MB) en format Unix et DOS ainsi que des **CD-ROMs**. Les commandes y relatives sont les suivantes :

`fdmount` *`option(s)`*
Usage de disquettes 3"1/2 HD formatées **Unix**
(Si vous n'arrivez pas à monter disquette Unix ainsi, il se peut qu'elle aie été écrite directement avec "tar". Essayez : `tar tvf /dev/fd0`)
`dosmount` *`option(s)`*
Usage de disquettes 3"1/2 HD formatées **DOS**
`cdmount` *`option(s)`*
Usage de **CD-ROM** 5"1/4
Options valables pour ces commandes :
`-h` = formatage et initialisation disquette **DOS** en haute densité (efface disquette !) (identique à commande "`fdformat -d /dev/fd0c`")
`-hi` = formatage et initialisation disquette **Unix** en haute densité (efface disquette !) (identique à commande "`fdformat /dev/fd0c`")
`-m` = montage du disque dans l'arborescence Unix, respectivement sur :
- `/floppy` pour disquettes Unix

- `/pcfs` pour disquettes DOS
- `/cdrom` pour CD-ROM

`-r` = montage du disque en lecture seulement
`-u` = démontage du disque
`-e` = éjection du disque (identique à commande "`eject`" sur Sun)

Exemples pour une disquette DOS :

- "`dosmount -h`" : formatage (efface !) disquette
- "`dosmount -m`" : montage disquette
- "`unix2dos` *`fichier_Unix fichier_DOS`*" : conversion (cf. ci-dessous)
- "`cp` *`fichier_DOS`* `/pcfs`" : copie de *fichier_DOS* sur disquette
- "`ls -lR /pcfs`" : consultation de tout ce qu'il y a sur disquette
- "`dosmount -ue`" : démontage et éjection de la disquette

Le SI-DGR dispose d'un système d'**écriture de CD-ROM** particulièrement intéressant pour l'archivage de données volumineuses (capacité d'un CD-ROM : 600 MB ; coût du média : env. 30.- ; durée pour graver le CD : env. 30 min.). Adressez-vous directement au SI-DGR en cas d'intérêt, ce système n'étant pas en libre service.

Manipulation de disquettes Unix avec "bar"

Sun recommande l'utilisation de la commande "`bar`" pour la manipulation de disquettes Unix. Celle-ci offre davantage d'options que la commande "`tar`".

`fdformat`
: Formatage Unix préalable (nécessaire !) de la disquette (efface contenu !)

`bar tvf /dev/rfd0`
: Affichage du contenu de la disquette

`bar cvf /dev/rfd0 {`*`fichier(s) ou répertoire(s)`*`}`
: Copie *fichier(s)* ou *répertoire(s)* sur la disquette en écrasant ce qui se trouverait déjà (commande qui ne peut donc pas être passée une seconde fois pour "ajouter" des fichiers). Si la quantité de données à copier excède la capacité du média (débordement), il y a demande automatique d'introduction d'un nouveau média (ce qui n'est pas possible avec `tar`).

`bar xvfp /dev/rfd0 {`*`fichier(s)`*`}`
: Copie le contenu de la disquette {ou seulement *fichier(s)* spécifié(s)} dans le répertoire courant en préservant les caractéristiques d'origine du fichier (option `p`)

`eject`
: Ejecte la disquette

Caractères accentués et terminaisons de lignes

Le codage des **caractères 8 bits** (caractères accentués...) est différent dans les mondes Unix, Macintosh et DOS. Lorsqu'un fichier texte contenant des car. 8 bits doit être transféré d'un environnement à un autre, on peut utiliser l'utilitaire "`recode`" du GNU (voir man-page) ou évent. la commande Unix "`tr`" pour convertir ces caractères.

De même, les **terminaisons de lignes** sont aussi différentes dans ces 3 environnements :

- `<LF>` sous Unix (car. ASCII de code octal 12)
- `<CR>` sur Macintosh (car. ASCII de code octal 15)
- `<CR><LF>` sous MS-DOS

Les utilitaires suivants permettent une **conversion** aisée de fichiers **Unix** <-> **DOS** :

`unix2dos` *`fichier_Unix fichier_DOS`*
: Conversion de *fichier_Unix* en *fichier_DOS* (car. 8 bits et terminaisons de ligne)

`dos2unix` *`fichier_DOS fichier_Unix`*
: Conversion de *fichier_DOS* en *fichier_Unix* (car. 8 bits et terminaisons de ligne)

Voir aussi les utilitaires GNU-`recode` et `charconv` qui offrent davantage de possibilités.

Cassettes

On recommande la commande "`tar`" qui, pour ce qui est des options principales, fonctionne comme "`bar`" (cf. ci-dessus). On travaillera cependant sur les devices suivants (sur nos stations Sun) :

- Exabyte 2.5 GB : `/dev/rst0` (rembobinage bande avant chaque écriture), `/dev/nrst0` (sans rembobinage)
- Exabyte 5 GB : `/dev/rst8` (rembobinage bande avant chaque écriture), `/dev/nrst8` (sans rembobinage)
- QIC (cassette 1/4") : `/dev/rst1` (rembobinage bande avant chaque écriture), `/dev/nrst1` (sans rembobinage)

En général :

`mt -f` *`périphérique`* `offline`
: Ejection de la cassette

Pour les devices en mode non-rewind (`/dev/nrst0, /dev/nrst8, /dev/nrst1`) :

`mt -f /dev/nrst0 fsf {`*`n`*`}`
: Saute par-dessus 1 {ou *n*} marques de fin de fichier (pour passer d'une archive-tar à une autre)

`mt -f /dev/nrst0 rewind`
: Rembobine bande

---

10. Messagerie et communication inter-utilisateurs

Dans les commandes qui suivent, il faut fournir, à la place de *machine* :

- soit le nom IP (Internet) de la machine (ex : "`dgrsun15.epfl.ch`")
- soit directement son adresse IP au cas où il y a un problème de Domain Name Server (DNS) (ex : "`128.178.162.17`")

Messagerie

Pour faire de la messagerie électronique (e-mail) depuis les stations Sun, vous pouvez utiliser à choix :

- utilitaire `mailtool` Sun (recommandé, mais nécessite station ou émulation X-window)
- commande `mail` (peu conviviale)

La forme SMTP des adresses de messagerie électronique est du type :

`prenom.nom@site.organisation.pays`

soit, pour les utilisateurs du DGR :

`prenom.nom``@dgr.epfl.ch`

Il est interdit de faire usage de caractères accentués dans les champs d'adresse ("To:", "From:" et "Cc:") ainsi qu'au niveau du "Subject:". Donc utiliser "a" à la place de "à", "e" à la place de "é" ou "è", "c" à la place de "ç", etc... Il n'y a cependant, au niveau des adresses, pas de distinction entre caractères majuscules et minuscules qui peuvent être utilisés indifféremment.

Vous pouvez trouver l'adresse de messagerie exacte de tout usager de l'EPFL dans l'annuaire CSO de l'EPFL, et de tout autre usager dans une haute école suisse (voire même à l'étranger) dans l'annuaire X-500 de Suisse et du monde, tous deux accessibles depuis l'application Mosaic.

On peut personnaliser son environnement de messagerie au moyen du fichier `~/.mailrc` (p.ex. créer des alias d'adresses selon la syntaxe : "`alias` *`surnom prenom.nom@site.organisation.pays`*"). Encore un conseil : n'oubliez pas de détruire les messages lus qui n'ont plus d'intérêt afin de ne pas encombrer votre espace disque (qui est limité !). Et pour en savoir davantage sur la messagerie électronique à l'EPFL, cliquez ici.

Forums de discussion

Concernant l'usage des forums de discussion Internet, voir le chapitre Usenet/News de ce support de cours.

Communication interactive

`talk` *`username@machine`*
: Etablissement d'une session de communication interactive avec utilisateur *username* travaillant sur *machine*. Le correspondant doit être couramment connecté (on peut le vérifier avec la commande `rusers`). Il existe différentes versions incompatibles du protocoles "talk"; si ça ne marche pas avec `talk`, essayez la commande `ytalk`. Voir aussi la commande "`write` *`username`*" et le système "`irc`" [*Internet Relay Chat*].

`rusers {`*`machine`*`}`
: Affiche la liste des utilisateurs connectés sur les machines locales {ou sur la *machine* spécifiée} [*remote users*]. Voir aussi commande `rwho`
`-l` = produit sortie de type `who`

`finger {{`*`username`*`}{@`*`machine`*`}}`
: Affiche des informations sur l'utilisateur *username* ayant un compte sur *machine* (notamment contenu du fichier `~/.plan`)

---

11. Autres outils de communication

Généralités

Unix est le premier système d'exploitation a avoir intégré à un si haut degré les possibilités de communication entre systèmes (cf. célèbre slogan de Sun : "the Network is the Computer"). Certains services réseau sont même totalement transparents pour l'utilisateur (nous ne les décriront pas davantage ici), parmi lesquels :

- partage de disques à travers le réseau par **NFS** (Network File System) ou AFS/DFS (Distributed File System)
- mécanisme **NIS** (Network Information System) de gestion centralisée des username/password et autres fichiers d'administration (anciennement dénommé **Yellow Pages**)
- mécanisme **DNS** (Domain Name Server) de résolution de noms de machine...
- synchronisation de l'horloge des machines (**NTP**, Network Time Protocol)

Les commandes ci-dessous nécessitent que vous ayez un compte (username/password) sur la machine distante, et que celle-ci soit accessible via le **protocole TCP/IP**, c'est-à-dire qu'elle soit sur le même réseau local IP (internet avec un petit "i") ou sur le réseau mondial **Internet**.

Commandes

`ftp {-n} {`*`machine`*`}`
: Initialisation d'une session de transfert de fichier avec une autre *machine* [*file transfer protocol*]. Avec `-n`, ne fait pas login automatique (intéressant lorsque l'on utilise `ftp` dans shell-script). Le fichier de configuration de ftp est `~/.netrc`.
Voir aussi la version X-window `ftptool` de cet outil.

: On se valide d'abord sur la *machine* distante avec la commande :
`user` *`username`* `<CR>` *`password`*
Puis on peut faire usage des **commandes** suivantes :
`ls` ou `dir` = afficher le contenu du répertoire courant sur la machine distante
`cd` *`répertoire`* = se déplacer dans *répertoire* sur machine distante
`pwd` = affiche nom du répertoire courant sur machine distante
`binary` = met `ftp` dans mode qui préserve les fichiers binaires lors du transfert (images, sons...)
`ascii` = (contraire de `binary` et état par défaut sur Sun) convertit fichiers texte en fonction des différentes représentations de caractères sur machines locale et distante
`{m}get` *`fichier{s}`* = récupérer *fichier{s}*
`{m}put` *`fichier{s}`* = envoyer *fichier{s}*
`{m}del` *`fichier{s}`* = détruire *fichier{s}* sur machine distante
`!` *`commande`* = exécute *commande* sur machine locale
`?` {*`commande`*} = aide {sur *commande*}
`quit` = sortie de ftp

Voir aussi l'utilitaire `ftptool` (Sun) doté d'une interface X11 conviviale qui permet notamment de transférer des arborescences complètes.

`telnet {`*`machine`*`}` **ou**
`rlogin {-l` *`username`*`} machine`
: Etablit connexion interactive sur *machine* distante [*remote login*]. Utiliser `rlogin` si le système d'exploitation de la machine distante est Unix, sinon utiliser en principe `telnet`.
Terminer la connexion avec `logout`, `exit` ou `<ctrl-D>` suivant la machine

: `<ctrl-]>` = passage au mode commande (affichage prompt `telnet>`)

Autres commandes utiles

La machine distante doit être de type Unix et il peut être utile l'on aie (pour commandes `rlogin, rsh, rcp`), dans le compte de cette machine, un fichier d'autorisation `~/.rhosts` pour éviter authentification par mécanisme username/password. Ce fichier aura la syntaxe :

```
machine1 {username}
machine2 {username}
...
```

et doit avoir la protection `rw-------` (`600`) pour fonctionner (et être inacessible aux autres utilisateurs, pour des raisons de sécurité évidentes).

`ping` *machine*
: Test si *machine* est accessible ou non

`rsh` *machine* `{-l` *username*`}` *commande*
: Exécution d'une *commande* sur *machine* distante (sans faire de login) en récupérant sa sortie standard sur la machine locale [*remote shell*]

`rexec` *machine commande*
: Comme `rsh` sauf que le mécanisme offert par le fichier `~/.rhosts` n'est pas utilisé et que l'on nous demande interactivement *username* et *password*

`rcp {-l` *username*`}` {*machine*`:}`*source* {*machine*`:}`*destination*
: Copie de fichier(s) entre machine locale et machine distante ou vice-versa [*remote copy*]
`-r` = copie de toute une arborescence de répertoires (récursivement)
`-p` = préserve dates de modification et protections

---

12. Commandes diverses

`bc`
: Calculatrice interactive à précision arbitaire. En sortir avec `<ctrl-D>`. Passer commande `scale=`*nb_déc* : pour fixer le nombre de décimales
L'entrée/sortie des valeurs peut se faire en base *n* = 2 (binaire), 8 (octal), 10 (décimal), 16 (hexadécimal) : passer commande :
`ibase=`*n* : base en entrée (saisie)
`obase=`*n* : base en sortie (affichage résultats)

Autres calculettes : voir commande `dc` et applications X-window `xcalc` (à utiliser avec option `-rpn`pour mode HP) et `calctool`

`date`
: Affiche sur sortie standard la date et l'heure courante

`uname`
: Affiche sur sortie standard des informations sur le système
`-n` = nom internet de la machine
`-s` = nom du système d'exploitation
`-r` = version du système d'exploitation
`-a` = toutes ces informations

`who`
: Affiche la liste des utilisateurs connectés sur la machine courante.
Voir aussi commandes `w`, `whoami`, `users` et `rusers`

`id`
Affiche No d'utilisateur (UID), No de groupe (GID), appartenance à des groupes secondaires

`echo {"}chaîne de caractères{"}`
`echo $`*`variable`*
Affiche *chaîne de caractères* ou contenu de *variable* sur la sortie standard
`-n` = sans faire de saut de ligne à la fin de l'affichage
Exemple : pour faire un "beep" depuis un shell-script : "`echo -n '<ctrl-G>'`" en Unix-BSD, ou "`echo '\007\c'`" en Unix-System V

`stty` *`option(s)`*
Définition des caractéristiques du terminal
`-a` = afffiche les settings courants
`sleep` *`n`*
Effectue une attente de *n* secondes
`time` *`commande`*
Exécute *commande* puis affiche temps écoulé, temps de calcul...

---

13. Redirection et tube

Entrée standard et sortie standard

Les "canaux standards" utilisés par Unix sont :

- **entrée** standard (stdin) : par défaut connectée au clavier
- **sortie** standard (stdout) : par défaut connectée à l'écran
- standard **error** (stderr) : par défaut connectée à l'écran

Les shells et la plupart des commandes et utilitaires Unix lisent leurs données sur l'entrée standard et rendent leurs résultats sur la sortie standard. La façon d'indiquer une fin de données sur l'entrée standard par défaut (clavier) est `<ctrl-D>`.

Les mécanismes de **redirection** et de **tube** (**pipe**) élargissent ces possibilités. Le C-shell utilise à cet effet les métacaractères "`<`", "`>`" et "`|`".

Redirection

La redirection permet de lire les données dans un fichier plutôt qu'au clavier (redirection de l'entrée standard) et/ou d'écrire les résultats dans un fichier plutôt qu'à l'écran (redirection de la sortie standard).

Pour supprimer l'affichage de la sortie standard d'une commande, la rediriger sur le null-device `/dev/null`

`commande` `>{>}{!}{&}` *`fichier_sortie`*
Redirige la sortie standard vers *fichier_sortie*
`>>` = en mode append si *fichier* préexiste
`>!` = écrase *fichier* s'il préexiste (ce "`!`" est nécessaire si la variable C-shell `noclobber` est positionnée)
`&` = redirige également standard error sur *fichier*
`(`*`commande > fichier_sortie`*`) >&` *`fichier_erreur`*

Redirige séparément la sortie standard sur *fichier_sortie* et les erreurs sur *fichier_erreur*

commande `<` *fichier_entree*
: Substitue *fichier_entree* à l'entrée standard

`set noclobber`
: (En C-shell ou T-shell) Evite d'écraser les fichiers existants par une redirection (variable activée dans environnement DGR). Pour écraser un fichier, il faut alors faire usage du "`!`" aprés le ">"

Exemples :

- `ls -al > liste` : crée fichier contenant la liste des fichiers du répertoire courant
- `cat f1 f2 f3 > fusion` : concatène 3 fichiers, puis
  `cat f4 >> fusion` : ajoute 4ème fichier
- `wc < fichier` : compte le nombre de lignes d'un fichier
- `(pwd ; ls -l) > fichier` : redirige la sortie des 2 commandes `pwd` et `ls`
- `pwd ; ls -l > fichier` : ne redirige que la sortie de la seconde commande (`ls`)

Tube (pipe ou pipeline)

Le tube permet de connecter deux commandes en envoyant les résultats de la première commande (sortie standard) en entrée dans la seconde commande (entrée standard). Une commande placée derrière un tube est appelée **filtre**.

commande1 `|{&}` *commande2*
: La sortie standard de *commande1* est reliée à l'entrée standard de la *commande2*.
  Avec `&`, redirige également standard error

commande1 `|` `tee {-a}` *fichier* `|` *commande2*
: Idem avec copie des résultats de *commande1* sur *fichier* intermédiaire (en mode append avec `-a`)

Exemples :

- `ls | wc -w` : affiche le nombre de fichiers du répertoire courant
- `ls -lg | lpr` : imprime le contenu du répertoire courant
- `ls -lg | sort -n +4 -5 | more` : affichage par `more` du contenu du répertoire courant avec classement des fichiers par taille croissante
- `ps -aux | grep $USER` : affiche liste de tous les processus appartenant à l'utilisateur
- `progr < data | sort > result` : exécute programme utilisateur `progr` sur fichier de données `data` et envoie les résultats triés sur le fichier `result`

---

14. Variables

Variables locales

Les variables de ce type sont propres au shell courant. Elles sont dites **locales** (privées) et on les appelle aussi **variables-shell**. Essentiellement utilisées dans les shell-scripts, elles peuvent être de type : valeurs entières, chaînes de caractères, listes de valeurs, identificateurs.

`set`
: Ecrit la liste des variables locales sur sortie standard

`set` *`variable=valeur`*
: (en **C-shell**) Affecte *valeur* à *variable*. Si la valeur est une chaîne contenant des espaces, la mettre entre guillemets

`set` *`variable=‘commande‘`*
: Met la sortie standard de la *commande* sur *variable*

`set` *`variable=(val1 val2 val3)`*
: (en **C-shell**) Affecte liste de valeurs à *variable*

`set` *`variable[n]=valn`*
: (en **C-shell**) Affecte *n*-ème composante de *variable* (la variable doit préexister !)

`variable=valeur`
: (en **Bourne -shell**) Affecte *valeur* à *variable*. (Il ne doit pas y avoir d'espace entre *variable*, signe = et *valeur*).

`unset` *`variable`*
: Efface *variable*

`$`*`variable`*
: Retourne contenu de *variable* (que l'on peut p.ex. afficher avec : "`echo $`*`variable`*")

`$`*`variable[i-j]`*
: Retourne composantes *i* à *j* de *variable*

`$#`*`variable`*
: Retourne nombre de composantes de *variable*

Variables d’environnement

Tout processus Unix possède des variables dites d'**environnement** (ou variables **globales**). Le shell étant un processus, il en possède donc aussi... et c'est en fait essentiellement à ce niveau qu'on les défini ! Les variables d'environnement ont la propriété d'être automatiquement transmises par copie à tous les processus et shell-fils créés à partir du shell où elles sont définies.

Elles servent principalement à définir/personaliser l'environnement de travail et peuvent être modifiées interactivement ou par des shell-scripts. Les types possibles sont les mêmes que pour les variables locales. La coutume veut que le nom des variables d'environnement soit généralement défini en caractères majuscules.

`setenv`
`printenv {`*`VARIABLE`*`}`
: Ecrit la liste des variables d'environnement {ou la valeur de la *VARIABLE* spécifiée} sur sortie standard

`setenv` *`VARIABLE valeur`*
: (en **C-shell**) Affecte *valeur* à *VARIABLE*

`setenv` *`VARIABLE val1`*`:`*`val2`*`:`*`val3`*
: (en **C-shell**) Affecte liste de valeurs à *VARIABLE*

`VARIABLE=valeur` **puis**
`export` *`VARIABLE`*
: (dans un script **Bourne-shell**) Affecte *valeur* à *VARIABLE* puis place cette variable dans l'environnement. (Il ne doit pas y avoir d'espace entre *VARIABLE*, signe = et *valeur*).

`unsetenv` *`VARIABLE`*
: (en **C-shell**) Efface variable d'environnement *VARIABLE*

`unset` *`VARIABLE`*
: (en **Bourne-shell**) Efface variable d'environnement *VARIABLE*

`$`*`VARIABLE`*
: Retourne le contenu de la variable d'environnement *VARIABLE* (que l'on peut p.ex. afficher avec : "`echo $`*`VARIABLE`*")

Variables d’environnement courantes

On donne ci-dessous la signification d’un certain nombre de variables d’environnement courantes. Certaines d’entre-elles sont automatiquement initialisées, au DGR, par le prologue-système lors de la procédure de connexion. Certaines sont propres au C-shell ou T-shell.

- `PATH` : chemin de recherche des commandes et utilitaires (liste des répertoires dans lesquels le shell doit chercher les exécutables des commandes passées par l'utilisateur)
- `MANPATH` : chemin d'accès aux pages de manuel
- `LD_LIBRARY_PATH` : liste des répertoires où l'éditeur de lien `ld` recherche libriaires
- `PRINTER` : imprimante par défaut (architecture BSD)
- `LPDEST` : imprimante par défaut (architecture System V)
- `DISPLAY` : display d'affichage, utilisé par tous les clients X11
- `EDITOR` : éditeur de texte par défaut utilisé par plusieurs utilitaires (par défaut `vi`)
- `TERM` : type de terminal (en principe `xterm` ou `vt100`, `vt220`), variable utilisée notamment par éditeurs (point d'entrée dans fichier `/etc/termcap`)
- `HOME` : path du home-directory
- `USER` : username de l'utilisateur
- `SHELL` : shell par défaut
- `ENV` : en Korn-shell, nom du script qui est exécuté chaque fois qu'un nouveau shell est invoqué (p.ex. `$HOME/.kshrc`)
- `PS1` : prompt du Bourne-shell (en C-shell, utiliser la commande : `set prompt="`*`prompt`*`"`)
- ...
