steria

40th anniversary 1969 - 2009

www.steria.com

Formation Unix de base

Introduction à Unix

Date 29/01/2011

V1.1

Formation Unix de base

Remerciements

Ce cours s'inspire, en partie, du cours « *d'Introduction à Unix* » (°), de Messieurs :

- Bruno Pouliquen (Bruno.Pouliquen@univ-rennes1.fr)
- Denis Delamarre (Denis.Delamarre@univ-rennes1.fr)
- Pierre Le Beux (Pierre.Lebeux@univ-rennes1.fr)
- *De la Faculté de médecine de Rennes, DESS TIMH,*
- Et Eric Drouin pour son « Portail du Collège virtuel » (pour le 'awk'),

⇒ que nous remercions ici.

(°) http://www.med.univ-rennes1.fr/~poulique/cours/unix

Formation Unix de base

- **Les intervenants STERIA**

- Benjamin LISAN, formateur (Capacity Planning et Performance DOSSI », Projet CAP5, STERIA).
- Les élèves analystes d'exploitation (AE) (STERIA).

- Règles de couleurs de fond de page, dans ce documents
- La partie en fond vert clair est la partie introduction.
- Les parties en rose clair, sont celles spécifiques à SFR.
- La partie terminale en bleu clair est la partie des annexes.

Formation Unix de base

- **But**

- Permettre aux pilotes d'être autonome face aux problèmes courants _ de niveau 1 _ rencontrés sur les serveurs Unix, gérés par eux.

Formation Unix de base

- **Préparation**

- Recueil de vos besoins
  - Questionnaire
  - Idées

- Connaître votre niveau de connaissance par rapport à l'ordinateur.

Contenu de la Formation

- Les notions de base Unix, en particulier :
- Droits sur les fichiers,
- les « partitions » disques appelées systèmes de fichiers ou 'file system' etc.

- les commandes de bases Unix (les commandes « shell »).
- Ce cours est une introduction à Unix, en particulier au langage de commande d'Unix, le « shell ».
- les répertoires et les fichiers « système » qu'il faut analyser en cas d'un incident ou de PB (ex. fichier « /var/adm/messages » etc. …).

Contenu de la Formation (suite)

- En particulier, nous aborderons les commandes suivantes :

- ‘ls’, lister des fichiers,
- ‘find’, pour trouver des fichiers,
- ‘ln’, pour créer des liens symbolique.
- « vi », l’éditeur,
- ‘top’, activité du système,
- ‘df –k’, pour visualiser les partitions appelées ‘file system’,
- ‘who’, ‘finger’, lister les utilisateurs.
- ‘awk’, ‘grep’, ‘sed’, pour traiter les chaînes de caractères.
- ‘ssh’, pour se connecter aux serveurs.
- ‘who –r’, ‘uptime’, pour savoir quand un serveur a rebooté.
- ‘pwd’, ‘cd’, répertoire courant et changement de répertoire,
- ‘cp’, ‘mv’, ‘rm’, copie, déplacement et suppression de fichier

Formation Unix de base

Plan

- Introduction
- Connexion
- Mot de passe
- Syntaxe générale des commandes
- Manuel en ligne : man
- Quelques commandes : who, finger, date, cal
- Système de fichiers (filesystem)
- Dénomination des fichiers
- Structure du système de fichier
- Chemins d'accès
- Répertoire de travail
- Quelques commandes : cd
- mkdir, rmdir : Création/destruction de répertoires
- ls : liste du contenu (fichiers et sous-répertoires) d'un répertoire
- cat, more
- Recherche : grep
- egrep
- sed

Plan de la Formation Unix de base

Plan (suite)

- wc
- Manipulation des fichiers : copie : cp
- Liens : ln
- Mouvements de fichiers : mv
- Destruction de fichiers : rm
- Redirection d'entrée / sortie
- Enchaînement de commandes: « | »
- Protection des fichiers (chmod, chown …).
- Droits d'accès
- Principales commandes
- Editeur de texte : vi
- Afficher le début et la fin d'un fichier texte : head et tail
- Compresser des fichiers (pour gagner de la place) : compress, gzip
- crontab
- cut, paste, tr, tee
- du (occupation du répertoire en Ko).
- Comparaison de fichiers texte (ascii) : diff, cmp, vim

Plan de la Formation Unix de base

Plan (suite & fin)

- Outils de compression et de décompression de fichiers : compress, gzip, zip / uncompress, ungzip, unzip
- La commande d'archivage tar
- sort, uniq
- DNS, nslookup
- NTP, ntpq, ntptrace
- uname
- ifconfig
- touch
- Politique de sécurité des accès chez SFR : sudo
- Processus : gestion des processus, ps, kill etc.
- Rédaction de scripts shell, écriture, sh (ksh), etc. nohup etc.
- Annexe : histoire d'Unix.
- Annexe : write
- Annexe : mail
- Annexe : sauvegarde système (par ufsdump (SUN)).
- Annexe : principales commandes Unix (shell)
- Annexe : Bibliographie

Formation Unix de base

Introduction

- UNIX : Est un « système d'exploitation », c'est à dire :
- Un logiciel permettant une utilisation efficace et commode d'un ordinateur.
- qui exploite au mieux la puissance de la machine.
- *UNIX un système « ~ ouvert », donnant beaucoup de possibilités à l'utilisateur.*
- *UNIX un système complexe.*
- *UNIX est normalement sans virus.*

↑ Une grande diversité d'Unix

Normalement, pas de virus avec Unix.

Note : si vous voulez en savoir plus sur l'histoire d'Unix, se reporter à l'annexe.

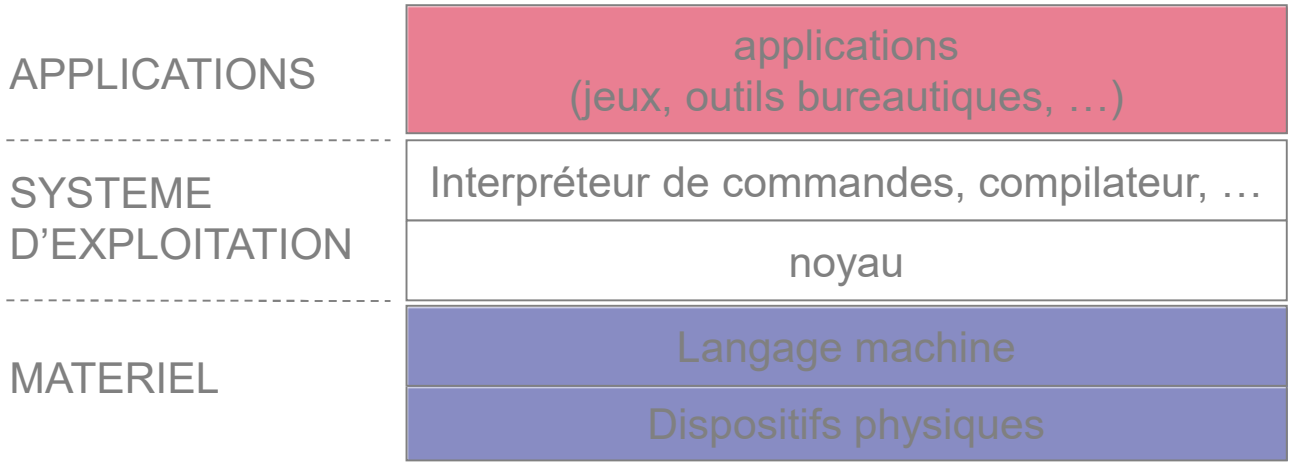
Système d'exploitation

- C'est l'interface entre l'utilisateur et le matériel
- Ses fonctions principales sont :
  - Contrôle des ressources (allocation et gestion du CPU et de la mémoire)
  - Contrôle des processus (des tâches)
  - Contrôle des périphériques
  - …
- Il contient des outils de gestion utilisables par les applications, tels que la manipulation de fichiers, gestion d'impressions, date…

Système d'exploitation

- Exemples de systèmes d'exploitation:
  - Unix, DOS, Windows, Mac OS, Linux, OS/2, BSD, ...

- Architecture-type:



Système d'exploitation

- Propriétés du système d'exploitation Unix
  - multi-tâches
  - multi-utilisateurs
  - multi-postes
  - Libre (et gratuit) pour certaines versions d'Unix : Linux etc. !!

- Ouverture/Fermeture d'une session
  - Travailler sous Unix implique une connexion au système
  - Login:
    - Identification de l'utilisateur: *login* + *mot-de-passe*
    - droits accordés par le *super-utilisateur* (**root**)
  - Logout:
    - **NE PAS ETEINDRE** une machine "sauvagement"
    - commandes « logout » ou « exit », au niveau de la console Unix (affichée à l'écran) ou au niveau de sa cession utilisateur, pour la quitter (en sortir).

Formation Unix de base

Connexion

- Il faut se connecter au système UNIX via le protocole réseau « `ssh` » (chez SFR) :
- Le système procède alors à votre identification (« login ») :

- Exemple :

```
ssh su1363.sfr.com
login : dupond
password : milou
```

Remarques :

- - Le nom est limité à 8 caractères *(sur certains Unix),*
- - Le mot de passe également *(sur certains Unix),*
- - Il y a différenciation des minuscules et majuscules,
- - Le mot de passe n'apparaît évidement pas à l'écran.
- Lorsque la connexion est établie apparaît alors l'invite (ou « prompt ») de la machine (ici dans notre exemple, le prompt du serveur est « `su1363` ») :

```
[root@su1363:/]
```

- Ensuite pour se déconnecter il suffit de taper la commande UNIX : « **exit** » ou « **logout** » (Ctrl-D fonctionne aussi).

Voir politique de sécurité concernant la connexion aux serveurs chez SFR (compte nominatif, sudo & co) plus loin.

➔Mot de passe

Lors de la première connexion il est bon (et même obligatoire) de changer son mot de passe, qui vous sera personnel. La séquence est la suivante :

```
noemed% passwd
Changing password for dupond on noemed.
Old password: milou
New password: loumi
Retype new password: loumi
```

A partir de ce moment le nouveau mot de passe est actif.

Remarques :

Afin d'éviter de se faire « craquer » son mot de passe, il est conseillé d'utiliser un mot de passe ayant les caractéristiques suivantes :

- *Plus de 6 caractères*
- *Au moins deux caractères non alphabétiques (0,1,2...,9,&,',!,%,@, ...)*
- *Eviter les mots du langage courant*

Exemples de mauvais mots de passe :

```
milou ab azerty robert35 1234 brest ...
```

➔Syntaxe générale des commandes

Le système d'exploitation UNIX offre à l'utilisateur un ensemble de commandes, l'appel de ces commandes répond toujours à la même syntaxe :

```
commande [options] [paramètres]
```

La commande est un mot clé en minuscules où toute faute de frappe constitue une erreur de syntaxe et provoque donc un message d'erreur (bien faire attention aux espaces).

Exemple :

```
noemed% faitcequejeveux
faitcequejeveux:Command not found
```

Elle (ou il ?) n'a rien compris !

Une commande peut être limitée à un mot clé ou peut être suivie d'options et/ou de paramètres facultatifs.

➔Syntaxe générale des commandes (suite)

Les paramètres et les options obéissent également à une certaine syntaxe.
Pour les options sur toutes les commandes Unix, elles sont en règle générale toujours précédées d'un tiret (-) et suivies d'au moins un caractère. Il peut y avoir plusieurs options. *Exemple :*

```
noemed% ls -a
```
(« **li**st **a**ll » : lister tout)

Paramètres:

Pour les paramètres ou arguments, il n'y a pas de syntaxe générale:

- Certaines commandes ne comporte pas de paramètres.
- D'autres en ont un obligatoire (nom de fichier par exemple).
- D'autres ont des paramètres facultatifs avec une syntaxe précise.

➔Manuel en ligne : man

Sous UNIX chaque commande est documentée. Cette documentation est accessible à l'aide de la commande « man » (pour manuel) suivi de la commande dont on veut consulter la documentation.

Exemple :

```
noemed% man cat
CAT(1V)  USER COMMANDS     CAT(1V)

NAME

cat - concatenate and display

SYNOPSIS
cat [ - ] [ -benstuv ] [ filename...       ]
DESCRIPTION
cat reads each filename in sequence and displays it
on       the standard output.      Thus:
(...)
```

La commande *cat* permet de visualiser le contenu d'un ou plusieurs fichiers.

➔Quelques commandes : who, finger

Information sur les utilisateurs :

Qui est là et qui êtes vous : (who)

```
noemed% who
dupond  ttyp0   Feb 18 11:44    (lichen)
seka    ttyp1   Feb 18 07:37    (plb)
poulique ttyp3 Feb 18 08:54    (dim)
picault ttyp4   Feb 18 09:08    (asterix)
```

On peut également demander qui on est :

```
noemed% whoami
dupond
```

ou

```
noemed% who am i
noemed!dupond  ttyp0   Feb 18 11:44 (lichen)
```

➔Quelques commandes : who, finger (suite)

•La commande finger est quasiment identique :

```
noemed% finger
Login       Name              TTY Idle   When    Where
dupond    Dupond Albert (DES  p0      2 Wed 11:44 lichen
seka      Seka Louis-Paul     p1 5:03 Wed 07:37 plb
poulique Pouliquen Bruno- ex p3 2:22 Wed 08:54 dim
picault  Picault Christelle  p4 3:29 Wed 09:08 asterix
```

•Mais elle donne plus d'informations sur un utilisateur particulier :

```
noemed% finger picault
Login name: picault         In real life: Picault
Christelle (Stage DEA Image-IA Caen)
Directory: /users/picault Shell: /bin/csh
On since Feb 18 09:08:36 on ttyp4 from asterix
3 hours 33 minutes Idle Time
Mail last read Wed Feb 18 09:09:25 1997
No Plan.
```

Formation Unix de base

➔Une commande pour connaître votre identification sur le serveur : id

Exemples :

```
AIX 5.2.0.0 (su0188) [root]/
# id
uid=0(root) gid=0(system) groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp),33001(prsudo1)
AIX 5.2.0.0 (su0188) [root]/
#

[oracle@su0349:/users/oracle] id
uid=202(oracle) gid=202(dba)
[oracle@su0349:/users/oracle]

su0832:/users/strb2126 [strb2126]> id
uid=41026(strb2126) gid=47111(gpstr711)
su0832:/users/strb2126 [strb2126]> sudo su -
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   February 2004
You have mail.
No news.
[root@su0832:/]id
uid=0(root) gid=1(other)
[root@su0832:/]
```

Si aucun opérande utilisateur est fourni à la commande « id », l'utilitaire « id » affiche, sur la sortie standard, le nom d'utilisateur et l'ID (identifiant) du groupe et de l'utilisateur et les noms de groupe du processus appelant. « id » affiche aussi les groupes secondaires de l'utilisateur (si l'on lui a déjà rattaché plusieurs groupes secondaires _ voir notion de groupes d'utilisateurs, plus loin dans ce document).

➔Commandes : date, cal

Consulter la date du jour : date

Pour obtenir la date et l'heure en anglais :

```
noemed% date
Thu Sep 25 15:43:46 WET 1997
```

La date standard est en anglais et sous forme : jour, mois, n° du jour, heure, zone horaire et année. Mais on peut en changer son format, comme ci-après :

date [+Format] : (Exemple : **date +%T**).

1) sans argument affiche la date système,

2) avec argument, modifie la date système.

```
[root@su1363:/] date '+DATE: %m/%d/%Y%nTIME:%H:%M:%S'
DATE: 11/19/2010
TIME:15:47:39
[root@su1363:/]
```

➔Date, cal (suite)

La commande Calendrier (cal) :

Cette commande permet de visualiser le calendrier d'une année ou d'un mois en anglais.
Syntaxe :
cal [no-jour no-mois] année

```
Exemple :
noemed% cal 1 96        # <= jour du calendrier de janvier 1996
January 96
 S  M Tu  W Th  F  S
              1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

On peut en principe visualiser les calendriers de l'an 1 jusqu'à l'an 9999. Vous pouvez ainsi connaître le jour de la semaine de votre naissance !

Système de fichiers

Qu'est-ce qu'un fichier :

Un fichier est une suite non structurée de caractères, stockée sur une mémoire auxiliaire.

En fait pour UNIX tout est fichier :

- **Un fichier de données (exemple : la liste des étudiants).**
- **Un programme (Toutes les commandes UNIX sont des fichiers) Un répertoire (qui permet de regrouper des fichiers).**
- **Un périphérique : l'écran ou le clavier par exemple, un disque dur, une disquette, un CD-ROM.**
- **Un lien vers un autre fichier.**
- **Un lien vers un fichier éloigné (via le réseau).**

Le système de fichiers est un ensemble de logiciels système qui gère tous les fichiers connus de la machine.

Système de fichiers

Pour gérer et retrouver ces fichiers, il faut des répertoires (« directory » en anglais).

Quand on dit que le système de fichiers est hiérarchisé, cela veut dire qu'il peut y avoir plusieurs niveaux de répertoires qui permettent de se déplacer dans un arbre de fichiers-répertoire pour arriver aux "feuilles" qui correspondent aux fichiers de données proprement dit.

Enfin, pour le système de fichiers, tout élément peut être protégé (en lecture, écriture ou exécution) voire inaccessible (protégé en lecture/écriture).

Dénomination des fichiers

Chaque fichier doit posséder un nom unique : un nom de fichier est caractérisé par une chaîne de caractères allant jusqu'à 255 caractères : lettres, chiffres, points ("."), trait de soulignement ("_")...

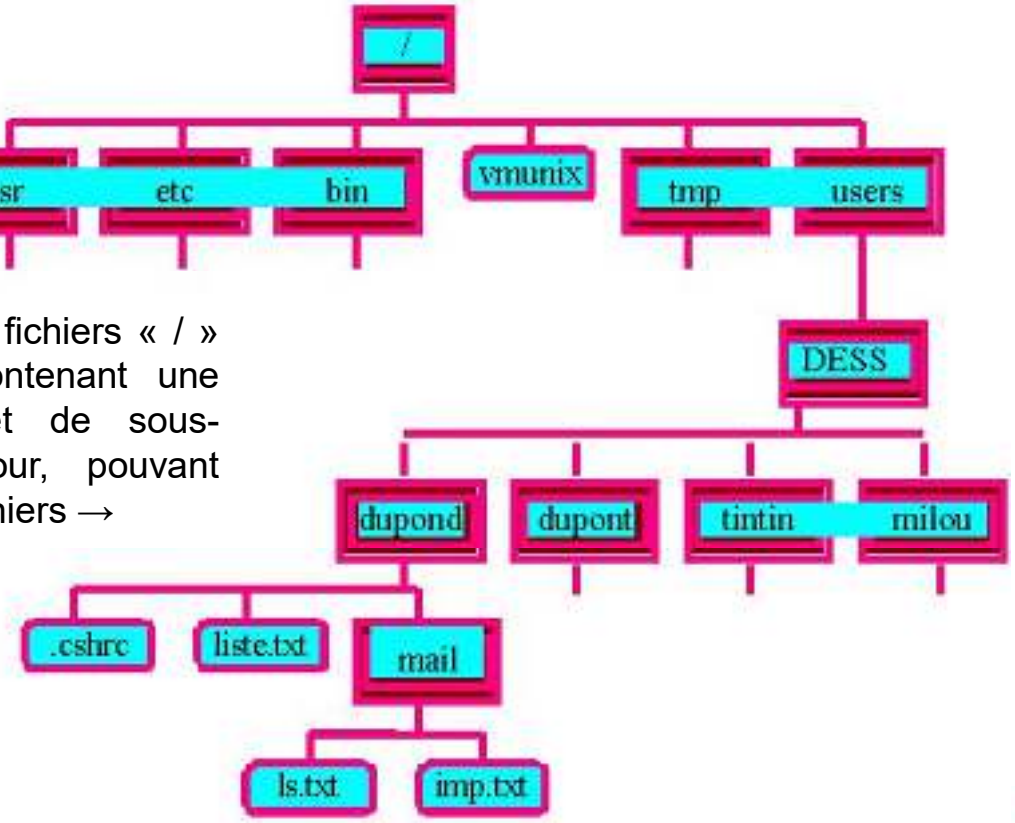
ATTENTION :
- les MAJUSCULES sont distinguées des minuscules (!).
- Eviter de commencer par les caractères spéciaux et le blanc (espace) pour les noms de fichiers.
- il n'y a pas de gestion de versions de fichiers, il faut donc le gérer soi-même en changeant de noms: par exemple garder le nom suivi d'un numéro de version : lettre1, lettre2, ou datées...
- les caractères « . » ou « .. » correspondent à des fichiers. On peut utiliser le point comme séparateur, celui-ci est généralement réservé pour indiquer le type du fichier (ou plus exactement dans la partie *extension* du nom du fichier) :

```
lettre.txt, programme.c, triangle.pas
```

Structure du système de fichiers

La structure est arborescente: la racine est le répertoire /, les nœuds sont les répertoires et les feuilles sont, en général, les fichiers.

A droite, exemple du système de fichiers « / » (root) _ d'un Unix BSD _, contenant une arborescence de répertoires et de sous-répertoires, chacun, à leur tour, pouvant contenir des répertoires ou des fichiers →



Formation Unix de base

➔Système de fichiers sur les serveurs de SFR

Un serveur d'application, chez SFR, se découpe en 3 ensembles de systèmes de fichiers (filesystems) :

• Les systèmes de fichiers du système d'exploitation;
• Le système de fichiers utilisateurs;
• Les systèmes de fichiers des applications;

• Les systèmes de fichiers utilisateurs et applicatifs sont appelés systèmes de fichiers du «**socle de base** ».

Note : ces informations, dans cette page et celles qui suivent, sont extraites du document de norme SFR « **Socle Unix. UNIX : Normalisation des arborescences** » de AKHATTAB Brahim.

Formation Unix de base

➔Système de fichiers sur les serveurs de SFR

Les versions d'Unix sont actuellement en voie de standardisation (version convergente). *Mais aujourd'hui encore, le fait de passer d'un système constructeur à un autre fait apparaître des différences*. Cependant certains répertoires sont normalisés et ils tendent à contenir des fichiers similaires. Ainsi nous retrouvons :

/usr
Contient la plupart des **programmes Unix standards** (binaires), le « **man** » en ligne et les **librairies**. Le sous répertoire standard « **/usr/local** » contient des fichiers locaux ne faisant pas partie de la livraison du système. Typiquement, nous retrouvons des procédures d'administration système *personnalisées* de la machine.

/dev
Contient **l'entrée de chaque périphérique** connu du système (unité de sauvegarde, disques, imprimante). Le nom de ces périphériques est différent d'un système constructeur à un autre !

➔Système de fichiers sur les serveurs de SFR

/var

Contient les différents fichiers traces générés. Sa taille varie suivant l'activité des processus. On y retrouve les fichiers d'**accounting**, **syslog**, les requêtes générées par la commande **crontab**, mais aussi les fichiers du **spool** d'impression. Ce répertoire est souvent un système de fichiers (en général, toujours).

/tmp

Contient les fichiers temporaires dont la durée de vie ne devrait pas excéder celle du processus créateur. Ce répertoire est le plus souvent utilisé par le système d'exploitation. **ATTENTION :** Les fichiers temporaires générés par d'autre processus autres que système (utilisateur, application) ont leur propre environnement de stockage sous **/varsoft/<*appli_n*>/tmp**.

/etc

Contient les fichiers critiques de démarrage mais aussi les fichiers de configuration du système. Certains systèmes tels que HP, placent les fichiers de démarrage des produits sous le répertoire « /sbin ». A part certains fichiers standards tels que « inittab » ou « bcheckrc », l'arborescence sous ce répertoire est différente pour chacun des systèmes constructeurs.

Formation Unix de base

➔Système de fichiers sur les serveurs de SFR

Le système, et les systèmes de fichiers du système d'exploitation (HPUX, Solaris, AIX, Linux …) sont toujours installés sur un disque interne au serveur et mirroré.

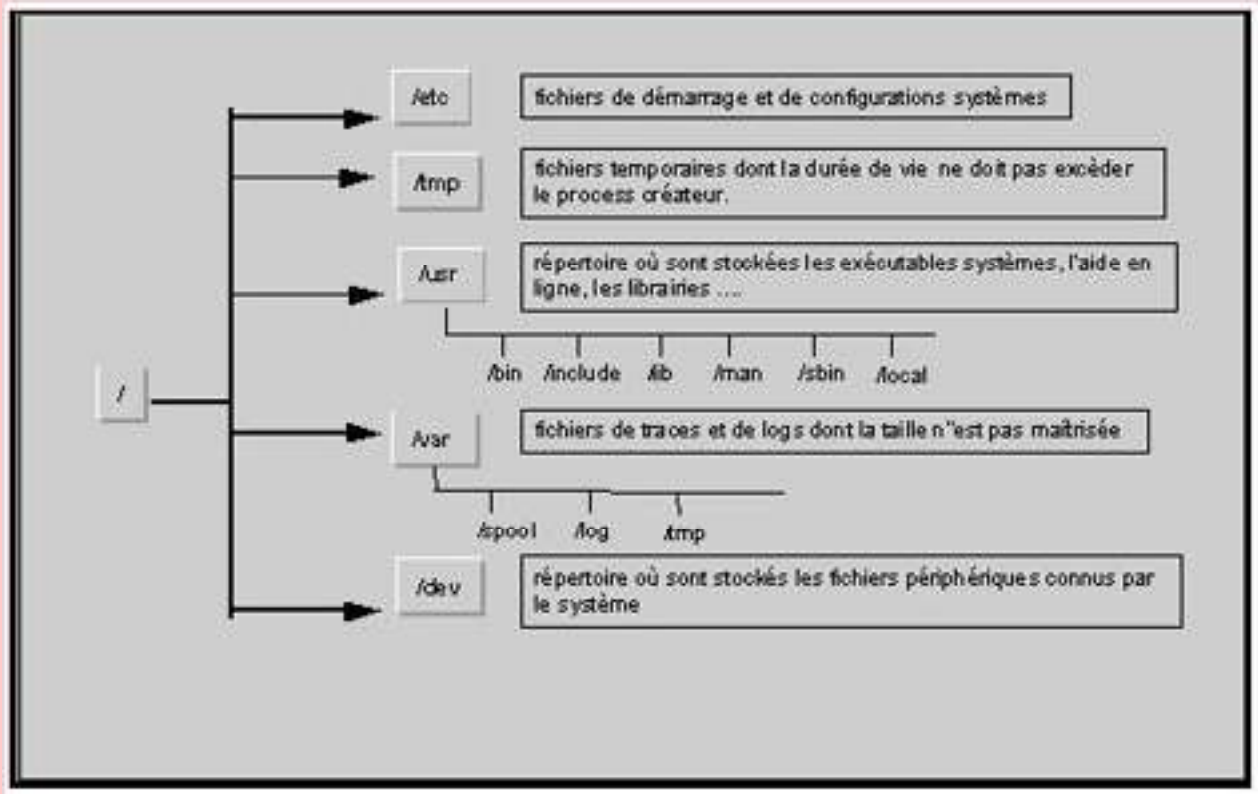


Figure 1. Répertoires systèmes (il manque juste, dans ce schéma, le répertoire « /opt » présents sur tous les systèmes Unix).

➔Système de fichiers sur les serveurs de SFR

➔Système de fichiers utilisateurs

•Un utilisateur qui se connecte à un système doit posséder un environnement de travail.
•Le point de montage « /users » _ pour le système de fichiers utilisateur _ a été retenu.
•Chez SFR, le système de fichiers « /users » est créé sur un disque externe d'une baie de disques EMC².

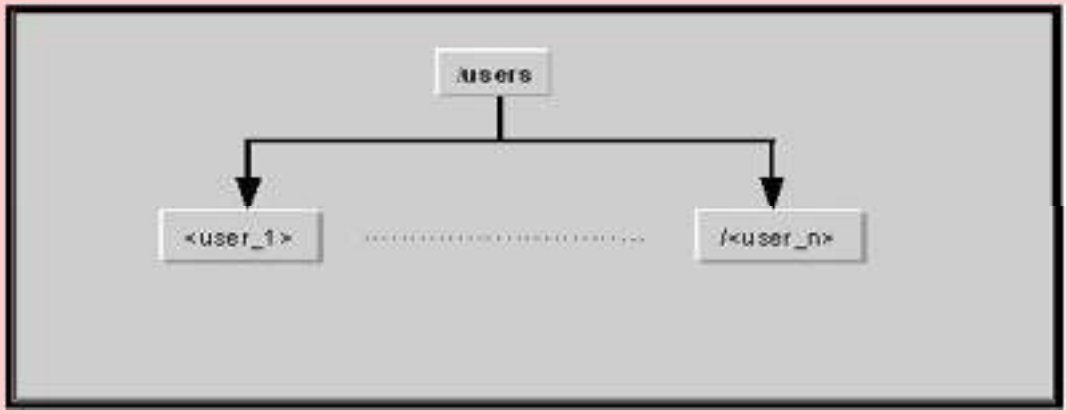


Figure 2. Répertoires utilisateurs

Formation Unix de base

➔Système de fichiers sur les serveurs de SFR

➔Les systèmes de fichiers applicatifs

Une application se compose :

•d'éléments statiques
- un ensemble de produits (1 à N).
- un ensemble d'utilitaires et de fichiers de configurations personnalisés.

•d'éléments dynamiques
- des fichiers de données gérés en interne (catalogue).
- des fichiers de données gérés par un SGBD (Sybase, Oracle).
- des fichiers traces (log, tmp, trc …).
- des espaces d'échanges (d'entrée / sortie).

Comme il peut y avoir plusieurs applications sur le même serveur, l'arborescence mis en place respecte le principe de non pollution d'une application par une autre.

Formation Unix de base

➔Système de fichiers sur les serveurs de SFR

Note : Une *application*, dans ce document, fait référence aussi bien aux logiciels « maison », qu'aux services d'exploitation, aux outils.

L'organisation des systèmes de fichiers applicative est la suivante :

- ***/product*** : Où sont rangés tous les produits,
- ***/catalog*** : Où sont stockées les données gérées en interne par les applications ou les points d'entrée vers les systèmes de fichiers de type /[prefixe]data<indice>_<instance>[suffixe].
- ***/database*** : où sont stockés les points d'entrées vers les systèmes de fichiers de données gérées par un SGBD (Sybase, Oracle)
- ***/varsoft*** : Où sont rangées toutes les données temporaires, les fichiers de traces, de log, d'historisation, d'archives, générés par les différentes applications, le répertoire de livraison des applications, un espace de travail pour les administrateurs et les fichiers à archiver.
- ***/interfaces*** : Où sont rangés tous les fichiers échangés entre applications.
- **/[prefixe]data<indice>_<instance>[suffixe]** : Où sont stockées les données de l'application ou les fichiers de données de bases de données.

Formation Unix de base

➔Système de fichiers (filesystem) chez SFR

Chacun des points d'entrée (indiqués page précédente), depuis le répertoire racine « / », doit faire l'objet d'un système de fichiers. . Ces systèmes de fichiers sont créés sur des disques EMC².

Note : L'installation des produits ou logiciels (dce, Haute dispo, C …), ajoutés au système d'exploitation de base, qui sont non « customisables » (non paramétrables par l'utilisateur final) et dont les fonctions sont communes aux autres applications, est réalisée dans les répertoires préconisés par le constructeur.

Exemple : En bleu, les systèmes de fichiers (**filesystem),** propre à SFR, sur un serveur HP-UX :

```
# Selection de la 1ere colonne du 'df -k', puis le 2eme champ compris entre '/' de la
# colonne, puis on garde que l'unique occurrence des champs en doublon, puis on elimine les lignes
# vides et enfin on rajoute un caractere '/' au debut de chaque ligne du resultat.
root@su0044:/ # df -k| awk '{print $1}'| cut -d/ -f2|uniq|grep -v "^$"|sed "s:^:/:"
/catalog
/interfaces
/opt
/product
/tmp
/users
/usr
/var
/varsoft
/database
/stand
root@su0044:/ #
```

Formation Unix de base

➔ Dénomination des serveurs (chez SFR)

Différence entre noms de serveurs chez SFR : entre cuxxxx, suxxxx et iuxxxx :

- suxxxx : serveur physique.
- iuxxxx : serveur virtuel ou instance virtuelle, hébergé par un châssis (°) :

➢ *LDOM : technologie de virtualisation SUN (logical domains).*

➢ *VMWARE : système propriétaire (de la société VMWare) de virtualisation (°) logicielle des architectures de type x86.*

- cuxxxx : châssis (serveur hébergeant plusieurs instances virtuelles).

(°) La **virtualisation** consiste à faire fonctionner sur un seul ordinateur plusieurs systèmes d'exploitation comme s'ils fonctionnaient sur des ordinateurs distincts. On appelle **serveur privé virtuel** (*Virtual Private Server* ou VPS) ou encore environnement virtuel (*Virtual Environment* ou VE), ces ordinateurs virtuels.

Formation Unix de base

➔Règles de nommage de serveurs chez SFR

I	U / W	XXXX
S	U / W	XXXX
C	U	XXXX

I	Instance virtuelle : 1) LDOM, 2) VMWARE
S	1) Serveur physique / 2) partition Mseries (SUN M5000, M3000, M9000).
C	Châssis
U	Unix et Linux
W	Windows
A	Reverse proxy (serveurs Unix)

Exemples :
SU0008
IU0015
CU1100

➔Chemins d'accès

LES NOMS DE CHEMINS : (pathname)

Il suffit de donner les noms des fichiers-répertoire de chaque niveau séparé par des caractères `/` **(*slash*).**

Exemples :

`/`
`bin` **: 1er niveau du répertoire** `bin`
`/users/dupond` **: 2ème niveau : répertoire** `dupond`**.**
`/users/dupond/mail/imp.txt` **: Le fichier** `imp.txt` **se trouvant dans le répertoire mail du répertoire** `dupond` **du répertoire** `users` **...**

Répertoire de travail

ANNUAIRE DE TRAVAIL ET ANNUAIRE PERSONNEL

Lorsque l'on se connecte sur une machine UNIX (login) l'utilisateur est directement placé au niveau de son répertoire personnel (home directory):

`/users/`*`vous`*

Ce répertoire devient alors votre répertoire de travail.
Pour connaître votre répertoire de travail utiliser la commande `pwd` (print working directory):

```
noemed% pwd
/users/dupond
```

➔Quelques commandes : cd

Changement de répertoire de travail (change-dir)
Permet de se positionner à un autre endroit de la hiérarchie.

Exemple :

```
noemed% pwd
/users/dupond
noemed% cd /tmp
noemed% pwd
/tmp
```

`cd` **sans aucun paramètre vous repositionne sur votre répertoire personnel :**

```
noemed% cd
noemed% pwd
/users/dupond
```

➔Quelques commandes : cd (suite)

Chemin absolu, chemin relatif

Pour dénommer un fichier on peut utiliser la dénomination absolue, c'est à dire en partant de la racine (/) :

```
/users/dupond/mail/imp.txt
```

Mais si notre répertoire de travail est `/users/dupond` il suffira de nommer ce fichier :

`mail/imp.txt` **(nommage relatif)**

Et si le répertoire de travail est `/users/dupond/mail` :

`imp.txt`

Répertoire spécial : `..`

Les deux points signifient « répertoire père »

Répertoire très utile pour les chemins relatifs :

`cd ../milou` **devient équivalent à** `cd /users/milou`

Formation Unix de base

➔Commande : touch

touch est une ***commande Unix*** standard permettant de modifier la date (*timestamp*) de dernier accès et de dernière modification d'un ***fichier***. Cette commande permet également de créer un fichier vide.
Lorsque *touch* est utilisé sans options, la commande utilise la date courante pour mettre à jour la date du fichier. Les dates mises à jour sont les dates d'accès et de modifications. Exemples :

•Voici par exemple la mise à jour de la date du fichier « Unix_Certif":

```
$ ls -l
total 0 -rw-r----- 1 ikipou ikipou 0 2007-07-30 07:58 Unix_Certif
$ touch Unix_Certif
$ ls -l total 0 -rw-r----- 1 ikipou ikipou 0 2008-01-20 16:01 Unix_Certif
```

Nb: la commande « > **Unix_Certif** » a le même résultat que la commande « touch **Unix_Certif** ».

•Si un fichier spécifié en option n'existe pas, un fichier vide est créé avec l'horodatage courant :

```
$ ls -l
total 0
$ touch Android
$ ls -l
total 0 -rw-r--r-- 1 ikipou ikipou 0 2008-01-20 16:04 Android
```

mkdir, rmdir : Création / destruction de répertoires

mkdir : Création d'un nouveau répertoire (**m**a**k**e ***dir****ectory*). Exemple: création d'un répertoire sous le répertoire de travail :

```
noemed% mkdir temporaire
```

ou

```
noemed% mkdir /users/dupond/temporaire
```

rmdir : Suppression d'un répertoire (qui doit être vide). Exemple:

```
noemed% rmdir temporaire
```

ls (suite)

•Liste le contenu du répertoire (**ls**t) :
ls (sans paramètre) : liste le répertoire de travail
ls *répertoire* : liste le contenu du répertoire donné
Exemple :

```
noemed% ls /users/dupond
liste.txt       mail
```

Liste de tous les fichiers : (- a pour all)

```
noemed% ls -a
.       ..      .cshrc
liste.txt       mail
```

On trouve ici les fichiers « **.** » et « **..** » utilisés par le système pour retrouver le répertoire courant et le répertoire père. De même, le fichier `.cshrc` est affiché (fichier servant à des initialisations automatiques).
Si l'on désire toutes les informations sur les fichiers standards, on utilisera l'option -l (long).

ls (suite)

Si l'on désire toutes les informations sur les fichiers standards, on utilisera l'option -l (**l**ong) :

```
noemed% ls -l
```

On peut cumuler l'option -l avec l'option -a :

```
noemed% ls -al (ou ls -a -l)
drwxr-xr-x    3  dupond      512       Feb  18  17:47  .
drwxr-xr-x    2  root        7168      Feb  18  17:54  ..
-rw-r--r--    1  dupond      733       Feb  18  17:45  .cshrc
-rw-r--r--    1  dupond      327       Feb  18  17:47  liste.txt
drwxr-xr-x    2  dupond      512       Feb  18  17:48  mail
```

Ceci signifie (en résumé) que le fichier `liste.txt` fait 327 octets (soit 327 caractères pour un fichier de texte), et qu'il a été modifié pour la dernière fois le 18 février à 17h 47.

cat

•Visualisation de fichiers (con**cat**enate) : cat

syntaxe : `cat` ***noms-de-fichier***
Exemples :

```
cat liste.txt
cat mail/imp.txt mail/ls.txt
```

Lecture de portions de fichiers **(head, tail, more)**

```
head [-n] fichier
tail [-n] fichier more fichier
```

Note :
Les crochets `[ ]` indiquent qu'il s'agit d'un paramètre facultatif (ici `-n` représente le nombre de lignes à visualiser). S'il n'est pas présent, il y a visualisation des 8 premières (ou dernières) lignes.

cat (suite)

Exemples:

```
noemed% head -3 liste.txt
Bonnefoy Isabelle
Channac Bertrand
Collet Jean-Yves
noemed% tail -4 liste.txt
Sebti Mohammed
Tiennot-Trebaol Dominique
Tillard Eric
Turmel Valerie
noemed% more mail/*.txt
```

La commande « `more` » (ou « `pg` » sur certains Unix) permet d'afficher page par page (d'une quarantaine de ligne) le résultat d'une commande ou du contenu d'un fichier à l'écran. Exemples : `more` *`fichier`*

```
cat /etc/hosts | more
```

Recherche d'une chaîne : grep (commande de type « filtre »)

Le programme `grep` est un utilitaire standard UNIX qui parcourt un ensemble de fichiers, selon un *modèle* de texte arbitraire, spécifiée par une *expression régulière*.

Syntaxe : **grep** [*options*] *motif* [*fichiers*]

Le *motif* ou *modèle* (en anglais « *pattern* ») peut être une simple chaîne de caractères, mais peut être également une expression contenant des caractères spéciaux.
(voir page suivante sur la définition d'un *motif* et d'une *expression régulière*).

Exemple :

```
noemed% grep 'Isabelle' liste.txt
Bonnefoy Isabelle
Kauffer Isabelle
```

Quelques caractères spéciaux (ou méta-caratères) :

. : Le point « **.** » correspond à n'importe quel caractère unique.
^ : L'accent circonflexe « **^** » correspond au début de ligne (i.e. « commence par »).
$: le symbole dollar « **$** » correspond à la fin de ligne (signifie « finit par »).

Recherche d'une chaîne : grep (suite) (« Filtre »)

Expressions régulières (*qui indiquent ce que l'on recherche*) :

`syntaxe:` **`egrep`** `'`*`expression`*`'` *`fichier`*

Une *expression régulière* est un *motif* ou *critère de recherche*, constitué de :

- Un caractère : Exemple : **`egrep 'A'`** : ici toutes les lignes contenant un A majuscule.
- Un ensemble de caractères, Par exemples ci-après :
- `[a-z]:` tout caractère alphabétique minuscule.
- `[aeiouy]` : toute voyelle minuscule.
- `[a-zA-Z0-9]` : tout caractère alphanumérique (alphabétiques min / maj et numériques).
- Un caractère spécial, comme ceux-ci-après :

❑ `.` : n'importe quel caractère
❑ `\n` : le caractère « retour-chariot »
❑ `\t` : le caractère tabulation
❑ `\b` : le caractère espace

Recherche d'une chaîne : grep (suite) (« Filtre »)

Quelques opérateurs d'expressions régulières : Exemples :

- `^` : début de ligne. Ex.: `egrep '^[0-9]'` : lignes commençant par un chiffre
- `$` : fin de ligne. Ex.: `egrep ' $'` : Lignes finissant par un espace
- `?` : 0 ou 1 fois. Ex.: `egrep 'anti ?bacterien'` (avec un espace ou sans espace)
- `*` : 0 ou n fois. Ex.: `egrep 'anti *bacterien'` (avec plusieurs espaces ou sans)
- `+` : 1 ou n fois. Ex.: `egrep ' [0-9]+ '` (Un chiffre entouré d'espaces)
- `()` : parenthèses. Ex.: `egrep ' ([AGCT][AGCT][AGCT]-)+ '` (ici, reconnaît les chaînes d'ADN : « ACA-AGC-AAA », mais pas les chaînes CAGE ou TATA !)
- `|` : ou (inclusif) : Ex.: `egrep 'anti *(bacterien|biotique)'` : on recherche les lignes contenant toutes la chaîne 'anti' et l'un de ces 2 chaînes 'bacterien' ou 'biotique', toutes ces 2 chaînes étant situées dans la même ligne et étant séparée par des espaces.

Note : pour plus d'informations sur les motifs & expressions régulières, voir l'annexe sur « grep ».

Exemples :

```
egrep ' [0-9]+ *[Ff]( |$)' mon_texte
```

=> retrouve toutes les lignes contenant un prix (4F, 51 F, 123 f, ...)

```
egrep '(^| )[Ll]e *[Bb]eux( |$)' rep_telephone
```

=> retrouve « Le Beux » (mot entier, avec ou sans la première lettre majuscule, avec ou sans espace). Voir exemples, page suivante → :

Formation Unix de base

Recherche d'une chaîne : grep (suite) (Filtre)

Exemples simples :

`grep BOB tmpfile`	{recherche la chaîne 'BOB' à l'intérieur de chacune des lignes du fichier 'tmpfile'}
`grep -i -w blkptr *`	{recherche les fichiers, pour le mot *blkptr*, qu'il soit en majuscule ou minuscule} -w : Recherches pour telle expression, tel un mot ou un mot entouré par \< et \>.
`grep run[-]time *.txt`	{recherche le mot 'run time' or 'run-time' dans tous les fichiers, ayant comme extension « .txt »}
`who | grep root`	{redirige le résultat de **who** vers **grep**, en recherchant la chaîne *root*}

Recherche d'une chaîne : grep (suite) (Filtre)

Exemples simples :

`grep smug files`	{Recherche dans les fichiers *files* les lignes contenant le texte 'smug'}
`grep '^smug' files`	{recherche les occurrences 'smug' en début de ligne}
`grep 'smug$' files`	{'smug' en fin de ligne}
`grep '^smug$' files`	{les lignes ne contenant que 'smug'}
`grep '\^s' files`	{les lignes commençant avec '^s', "\" annule la caractéristique « joker » du caractère spécial ^}
`grep '[Ss]mug' files`	{recherche 'Smug' ou 'smug'}

Recherche d'une chaîne : grep (suite) (Filtre)

Exemples simples :

`grep 'B[oO][bB]' files`	{recherche BOB, Bob, BOb ou BoB }
`grep '^$' files`	{recherche les lignes blanches}
`grep '[0-9][0-9]' file`	{recherche 2 paires de chiffres : 00, 01, .. 20…}

Recherche d'une chaîne : grep (suite) (Filtre)

Exemples plus complexes :

Les noms commençant par K :

```
noemed% grep '^K' liste.txt
Kauffer Isabelle
Kervran Stephane
```

Les prénoms finissant par 'ie' :

```
noemed% grep 'ie$' liste.txt
Delangle Stephanie
Lecor Nathalie
Muller Sylvie
Turmel Valerie
```

Les noms de plus de 10 caractères :

```
noemed% grep '^.......... ' liste.txt
Lebouteiller Rachel
```

Options : `-i` (pas de différenciation majuscule / minuscule)
`-v` (affiche les lignes qui ne sont pas reconnues)

Recherche d'une chaîne : grep (suite) (Filtre)

Exemples plus complexes (suite) :

Options : `-i` (pas de différenciation majuscule / minuscule)
`-v` (affiche les lignes qui ne sont pas reconnues)

Remarque :
On est parfois amenés à rechercher des expressions plus complexes.
On utilisera alors un outil associé : `egrep`

```
[root@su0063:/] egrep "shminfo_shmmax|shminfo_shmseg|seminfo_semmni|seminfo_semmns" /etc/system
set shmsys:shminfo_shmseg=256
set shmsys:shminfo_shmmax=16777216
set semsys:seminfo_semmni=256
set semsys:seminfo_semmns=500
[root@su0063:/]
```

Recherche d'une chaîne : grep (suite) (Filtre)

- Exemples :

```
# cat annuaire.txt
ACHERES, ZAC du Magasin -203/209 avenue du Général de Gaulle - 78260 Achères, Tel 01 30 06 72 62
RENNES, 1 rue de Jouanet CS 70331 - Route de Fougères - 35703 Rennes Cedex 7, Tel 02 23 21 37 00
PARIS LA DEFENSE - SEQUOIA, Tour Séquoia - 1, place Carpeaux - 92915 Paris La Défense Cedex, Tel 01 71 07 70 00
PARIS NANTERRE - RIVE DEFENSE, 5 rue Noël PONS - 92000 Nanterre, Tel 01 71 53 60 00
untel 99 ans
telephoner au 99.28.42.15
Michel LE BEUX 01.42.65.66.63
# egrep '[Ll][Ee] *[Bb][Ee][Uu][Xx]' annuaire.txt
Michel LE BEUX 01.42.65.66.63     # <= on retrouve bien la chaîne de caractère « Le Beux ».
# cat prix
couteau 5 euros
cahier 4F
caculette 51 F
telephone portable 124 f
stylo 1 euros
# egrep ' [0-9]+ *[Ff]( |$)' prix.txt
cahier 4F
caculette 51 F
telephone portable 124 f
#
```

Recherche d'une chaîne : grep (suite) (Filtre)

•Exemples :

```
# cat biochimie.txt
anti bacterien
anti            bacterien4
antibacterien2
antibiotique3
anti biotique
CAGE
TATA
ACA-AGC-AAA
ACT-TGC-AGT
TCG-GAC-TGA
GGC-TAT-GTC
CCG-AGC-AAC
TGA-ACG-TCA
AGC-CTG-ACG
CCG-ATA-CAC
GGC-TCG-TTA
# egrep 'anti ?bacterien'  biochimie.txt # <= Ici on recherche donc « anti bacterien » ou « antibacterien ».
anti bacterien
antibacterien2
# egrep 'anti *bacterien'  biochimie.txt # <= on recherche « antibacterien » ou « anti » et « bacterien » séparés
                                         # par des espaces.
anti bacterien
anti            bacterien4
antibacterien2
# egrep 'anti *(bacterien|biotique)' biochimie.txt
anti bacterien
anti            bacterien4
antibacterien2
antibiotique3
anti biotique
# → www.steria.com
```

```
# egrep '([AGCT][AGCT][AGCT]-)+' biochimie.txt
ACA-AGC-AAA
ACT-TGC-AGT
TCG-GAC-TGA
GGC-TAT-GTC
CCG-AGC-AAC
TGA-ACG-TCA
AGC-CTG-ACG
CCG-ATA-CAC
GGC-TCG-TTA
#
```

➔Remplacement des chaînes de caractères par d'autres : sed (Filtre)

sed : Editeur de flot de données, en mode en batch (traitement par lot) et de fichiers texte.

Exemples:

Substitution :

```
sed –e 's/foo/bar' myfile.txt
```

Remplacer la première occurrence du mot 'foo' par 'bar' sur chaque ligne du fichier nommé « *myfile.txt* ».

➔Recherche et remplacement de chaîne : sed (Filtre)

`syntaxe: sed 's/expression/remplace/g' fichier`

L'expression est du même type que celle définie pour egrep.

`sed 's/antibiotique/antibacterien/g'` : Remplace toute occurrence de « antibiotique » par « antibacterien ».

=> options :

`sed 's/expression/remplace/'` : ne remplace qu'une fois par ligne.

`sed 's/...(exp2).../...\1.../'` : le symbole `\1` signifie que l'on garde le contenu de ce qui était dans la parenthèse (Note : Le « \1 » ne marche pas avec « sed » en Solaris).

Exemples :

`sed 's/lebeux/le beux/g' annuaire` : On force un espace

`sed 's/beurre doux/beurre sale/g'` : Traducteur normand->breton

`sed 's/mot de passe.*//g' doc` : Enlève les mots de passe de la doc

➔Recherche et remplacement de chaîne : sed (suite)

(Filtre)

Exemple : commande pour le passage en nouvelle numérotation à 10 chiffres (en rajoutant 2 chiffre au début du n° de tel) (voir exemple page suivante) :

```
sed    ’s/tel 99/tel 02.99/g’ annuaire > nvannuaire
```

=> Problèmes : a) la ligne « `untel 99 ans` » remplacée à tord.
Et b) la ligne « téléphoner au 99.28.42.15 » n’est pas remplacée.

```
sed 's/tel.*99\./tel 02.99\./' annuaire > nvannuaire
```

Note : « .* » signifie n’importe quel caractère, « \. » signifie qu’on recherche le caractère « . ».
(On inhibe le caractère « joker », i.e. spécial, de ce caractère « . » pour la commande « `sed` »).

⇒ Résultat : Même chose que précédemment mais plus robuste (voir exemple page suivante).

Autres exemples :

```
Sed 's/99(.[0-9][0-9])+/02.99\1/g' annuaire > nvannuaire
```

Formation Unix de base

➔Recherche et remplacement de chaîne : sed (suite) : Exemples :

```
# cat annuaire    # <= affichage à l'écran du contenu du fichier « annuaire »
ACHERES, ZAC du Magasin -203/209 avenue du Général de Gaulle - 78260 Achères, Tel 01 30 06 72 62
RENNES, 1 rue de Jouanet CS 70331 - Route de Fougères - 35703 Rennes Cedex 7, Tel 02 23 21 37 00
PARIS LA DEFENSE - SEQUOIA, Tour Séquoia - 1, place Carpeaux - 92915 Paris La Défense Cedex, Tel 01 71 07 70 00
PARIS NANTERRE - RIVE DEFENSE, 5 rue Noël PONS - 92000 Nanterre, Tel 01 71 53 60 00
untel 99 ans
téléphoner au 99.28.42.15
Michel LE BEUX 01.42.65.66.63
# sed   's/tel 99/tel 02.99/g' annuaire
ACHERES, ZAC du Magasin -203/209 avenue du Général de Gaulle - 78260 Achères, Tel 01 30 06 72 62
RENNES, 1 rue de Jouanet CS 70331 - Route de Fougères - 35703 Rennes Cedex 7, Tel 02 23 21 37 00
PARIS LA DEFENSE - SEQUOIA, Tour Séquoia - 1, place Carpeaux - 92915 Paris La Défense Cedex, Tel 01 71 07 70 00
PARIS NANTERRE - RIVE DEFENSE, 5 rue Noël PONS - 92000 Nanterre, Tel 01 71 53 60 00
untel 02.99 ans       # => Problèmes : a) la ligne « untel 99 ans » remplacée à tord.
téléphoner au 99.28.42.15      # => et b) la ligne « téléphoner au 99.28.42.15 » n'a pas été remplacée.
Michel LE BEUX 01.42.65.66.63
# sed 's/tel.*99\./tel 02.99\./' annuaire
ACHERES, ZAC du Magasin -203/209 avenue du Général de Gaulle - 78260 Achères, Tel 01 30 06 72 62
RENNES, 1 rue de Jouanet CS 70331 - Route de Fougères - 35703 Rennes Cedex 7, Tel 02 23 21 37 00
PARIS LA DEFENSE - SEQUOIA, Tour Séquoia - 1, place Carpeaux - 92915 Paris La Défense Cedex, Tel 01 71 07 70 00
PARIS NANTERRE - RIVE DEFENSE, 5 rue Noël PONS - 92000 Nanterre, Tel 01 71 53 60 00
untel 99 ans
tel 02.99.28.42.15   # <= ici la ligne a bien été remplacé correctement.
Michel LE BEUX 01.42.65.66.63
#
```

➔Annexe : Commande ‘sed’ (compléments) (Filtre)

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).

➔Annexe : Commande ‘sed’ (compléments) (suite) (Filtre)

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]/ →/
```

➔Annexe : Commande ‘sed’ (compléments) (suite et fin) (Filtre)

Exemples (plus compliqués) :

```
# 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"
```

Compte du nombre de mots, de lignes, d'octets : wc

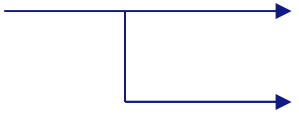
- **wc** : Compte le nombre de lignes, de mots et d'octets (de caractères) dans un fichier ou les fichiers spécifiés par le paramètre du fichier :
- Exemple:

```
$ wc *.txt
# compte les lignes, les mots, les octets dans tous les fichiers txt
wc -l /etc/passwd
# Compter le nombre d'utilisateurs dans votre système

[root@su0349:/users]
 ¤ ls |wc -l
     379
[root@su0349:/users]
 ¤ ps -ef|wc -l
     126
[root@su0349:/users]
 ¤
```

➔Ecrire des commandes ou le résultat d'une commande dans un fichier : tee -

Vous pouvez exécuter une commande, avec un affichage à l'écran et une copie dans un fichier texte. Ce qui peut être très pratique pour garder un trace ou pour aider a résoudre un problème. Pour cela il faut lancer votre commande avec un ou plusieurs arguments suivi de : « `| tee votreFichier` » (°).

Syntaxe: `./commande_shell | tee votreFichier`



⇒Sortie du résultat de la commande à l'écran et

⇒ dans le fichier *votreFichier*

Dans cet exemple la commande « `# apt-get update` » va copier son résultat dans le fichier texte « `apt-update.txt` » se trouvant dans le répertoire « `/home/libordux` »:

```
# apt-get update | tee /home/libordux/apt-update.txt
```

Voici une astuce Bash pour enregistrer toutes les activités du shell dans un fichier:

```
$ bash -i 2>&1 | tee fichier
```

(°) Note : « `| tee -a votreFichier` » avec l'option « `-a` » rajoute à la fin de votre fichier *votreFichier*.

Formation Unix de base

➔Renommer une commande : alias

alias : Renomme ou définit une commande.

Syntaxe : **alias variable=commande**

Exemple : **alias ll='ls -l'**

Besoin : Je souhaite créer un raccourci ou un alias pour une commande, tel que lorsque je tape "t" la commande suivante s'exécute: « telnet localhost 2500 ».

Solution : Il suffit de taper dans un terminal : **alias t='telnet localhost 2500**'.

Et pour que ce soit permanent de rajouter cette ligne de commande, dans ton fichier de lancement de ton environnement de ton login **".bashrc"** _ si ton shell est du bash _, « **.profile** » _ si ton shell est du Bourn shell (sh) ou du Korn shell (ksh) _ ou « **.cshrc** » _ si ton shell est du « c shell ».

➔Manipulation des fichiers : cp (copie)

Commande `cp` (**co**p**y**)

Syntaxe : **`cp`** *`fichier nouveau-fichier`*

Exemple :

```
noemed% cp liste.txt nouvelle_liste.txt
noemed% ls -l
total 3
-rw-r--r--     1 dupond 309 Feb 19 09:52 liste.txt
drwxr-xr-x     2 dupond 512 Feb 19 09:12 mail
-rw-r--r--     1 dupond 309 Feb 19 10:06 nouvelle_liste.txt
```

Le système a bien créé un nouveau fichier avec les mêmes caractéristiques que l'ancien.

Liens : ln

Commande `ln` (**ln**k) :
Syntaxe : `ln -s` *`fichier référence`*
Permet de faire une copie logique du fichier, en général on utilise les liens symboliques (option -s) :

```
noemed% ln -s liste.txt lst
noemed% ls -l
total 4
-rw-r--r--     1 dupond 309 Feb 19 09:52 liste.txt
lrwxrwxrwx     1 dupond       9 Feb 19 10:10 lst -> liste.txt
drwxr-xr-x     2 dupond 512 Feb 19 09:12 mail
-rw-r--r--     1 dupond 309 Feb 19 10:06 nouvelle_liste.txt
```

Le fichier `lst` n'est qu'un fichier virtuel, qui ne contient que la référence vers le fichier réel.
Avantage : Toute modification du fichier d'origine est répercutée sur le fichier virtuel.
Les liens sont souvent utilisés pour les répertoires:
`/users/DESS` est un lien vers `/pub/users/DESS`

➔Mouvement déplacement d'un fichier : mv

Commande `mv` (move) : Syntaxe : `mv ancien nouveau`

Deux utilisations essentielles : renommer un fichier et déplacer un fichier d'un répertoire vers un autre.

a) Renommer un fichier : Exemples :

```
noemed% mv nouvelle_liste.txt nvliste.txt noemed% ls -l
total 4
-rw-r--r--  1    dupond   309      Feb      19       09:52    liste.txt
Lrwxrwxrwx  1    dupond   9        Feb      19       10:10    lst -> liste.txt
drwxr-xr-x  2    dupond   512      Feb      19       09:12    mail
-rw-r--r--  1    dupond   309      Feb      19       10:06    nvliste.txt
```

b) Déplacer un fichier. Exemple : Le fichier imp.txt replacé dans le répertoire courant:

```
noemed% mv mail/imp.txt .
```

Cette commande aurait pu s'écrire:

```
noemed% cd mail
noemed% mv imp.txt ..
```

Destruction d'un fichier : rm

Commande `rm` (**rem**ove).

syntaxe : **`rm`** *`fichiers`*

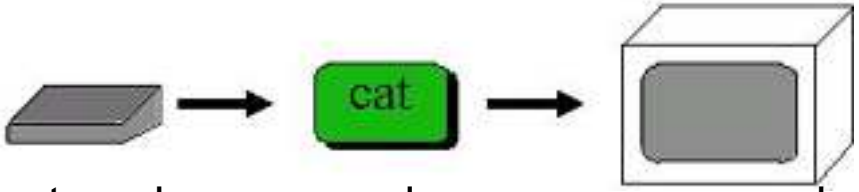
Exemple :

```
noemed% rm nvliste.txt lst
```

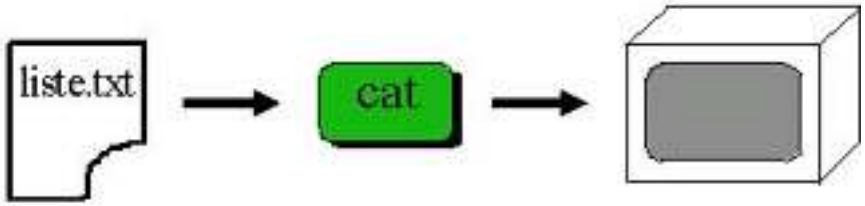
Attention ! La commande « rm » supprime immédiatement le fichier, sans poser aucune question => de ce fait, elle est dangereuse et doit donc être utilisée avec précaution.

Redirection d'entrée / sortie

Tout processus (programme) sous UNIX possède deux organes de communication standards appelés « Entrée standard » et « Sortie standard ».
Exemple : la commande `cat` prend tout caractère tapé au clavier pour l'afficher sur l'écran...



Quand on tape la commande `cat liste.txt`, la commande prend comme entrée standard le fichier `liste.txt`



➔Redirection d'entrée / sortie

Pour détourner l'entrée standard, on utilise le caractère <,
pour la sortie standard, le caractère >.

`cat liste.txt` est équivalente à `cat < liste.txt`

Pour copier le fichier liste.txt on peut utiliser la commande :

```
cat < liste.txt > nvliste.txt
```

On peut également créer un nouveau fichier en utilisant la commande :

```
cat > fichier.txt
```

Remarque : On peut également rajouter des lignes dans un fichier en utilisant la commande : `cat >> fichier.txt`

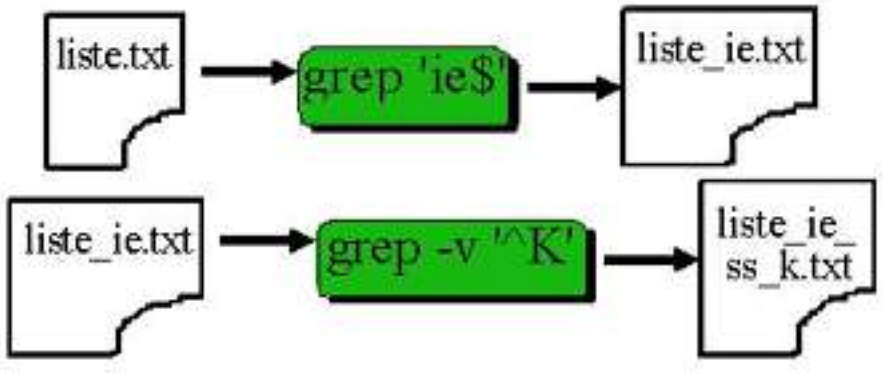
➔Enchaînement de commandes : |

On est souvent amené à enchaîner plusieurs commandes UNIX. Par exemple : avec la commande `grep` on voudrait connaître toutes les lignes qui finissent par "ie" mais qui ne commencent pas par K.
Deux solutions :
1) Deux commandes et un fichier intermédiaire

```
grep 'ie$' liste.txt > liste_ie.txt
```

puis

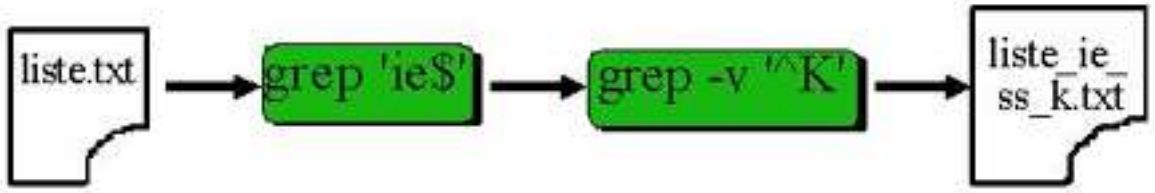
```
grep -v '^K' liste_ie.txt > liste_ie_ss_k.txt
```



➔Enchaînement de commandes : | (suite)

2) Utilisation du "pipe" (caractère " | ")permettant de rediriger la sortie standard d'une commande vers l'entrée standard de la suivante :

```
grep 'ie$' liste.txt | grep -v '^K' > liste_ie_ss_k.txt
```



Protections et droits sur les fichiers

Le système de fichiers UNIX gère des droits d'accès caractérisés par trois possibilités d'utilisation :

Lecture (R pour **r**ead)
Ecriture (W pour **w**rite)
Exécution (X pour e**x**ecute)

Pour un répertoire, X indique la possibilité d'entrer dans ce répertoire.
Chaque fichier est caractérisé par trois triplets d'accès qui décrivent les droits :

du propriétaire (u pour **u**sers)
du groupe (g pour **g**roupe)
des autres (o pour **o**thers)

Protections et droits sur les fichiers

Les droits associés à un fichier apparaissent lorsque l'on utilise la commande ls -l.
Exemple :

```
noemed% ls -lg (g pour lister également le groupe possesseur)
total 2
-rw-r--r-- 1 dupond dess97 309 Feb 19 09:52 liste.txt
drwxr-x--- 2 dupond dess97 512 Feb 19 10:22 mail
```

Le type de fichier est caractérisé par **d** pour un répertoire et **-** (tiret) pour un fichier normal.

On s'aperçoit que le fichier `liste.txt` est lisible par tout le monde (utilisateur, groupe et autres), mais qu'il n'est modifiable que par l'usager dupond.
Par contre le répertoire mail n'est pas lisible pour les autres utilisateurs. Il est lisible par les usagers du groupe dess97 `rwx`**`r-x`**`---`, mais pas modifiable.
L'usager dupond a tous les droits sur le répertoire mail : **`rwx`**`r-x---`

➔Droits d'accès sur les fichiers

Cette opération est possible par le propriétaire du fichier grâce à la commande chmod (**ch**ange-**mod**e) :

syntaxe: **chmod** [*droits*] *fichier*

Les droits sont précisés sous forme de paramètres :
`u` (**u**sers), `g` (**g**roup), `o` (**o**thers), `a` (**a**ll) avec :

`+` Droit accordé
`-` Droit enlevé
`=` Droit fixé

Suivi du droit associé :

`r` (lecture)
`W` (écriture)
`x` (exécution)

➔Droits d'accès sur les fichiers (suite)

Exemples :

```
noemed% chmod o-r liste.txt
noemed% ls -l liste.txt
-rw-r----- 1 dupond   309 Feb 19 09:52 liste.txt

noemed% chmod a=rw liste.txt
noemed% ls -l liste.txt
-rw-rw-rw-     1 dupond       309 Feb 19 09:52 liste.txt
noemed% chmod g+w mail/*
noemed% ls -l mail
total 2
-rw-rw-r--      1 dupond          365        Feb   19   09:12    imp.txt
-rw-rw-r--      1 dupond          391        Feb   19   09:12    ls.txt
```

➔Système de fichier ou File system (FS)

Un **système de fichiers** (*file system* ou *filesystem* en anglais) ou **système de gestion de fichiers** est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, disquette, CD-ROM, clé USB, disques SSD, etc.).

Une telle gestion des fichiers permet de traiter, de conserver des quantités importantes de données ainsi que de les partager entre plusieurs programmes informatiques. Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.

Il permet d'organiser les supports (ou volumes) stockant les données _ disques …

Il existe d'autres façons d'organiser les données, par exemple les bases de données (notamment base de données relationnelle) et les fichiers indexés.

➔Afficher les FS montés et leur taille : df

df : Affiche des informations sur la volumétrie des FS :
df : informations sur l'occupation du disque ou/ et du FS.
df : Peut aussi afficher le nombre de fichiers et de blocs disques libres.

```
[root@su1363:/kernel] df -h /catalog
Filesystem             size   used  avail capacity  Mounted on
STE0993/catalog         66G   462M    38G     2%    /catalog
[root@su1363:/kernel] df -a /catalog
/catalog           (STE0993/catalog   ):80083281 blocks 80083281 files
[root@su1363:/kernel]
```

➔Afficher les FS montés et leur taille : df (suite)

Exemple :

```
[root@iu0263:/var/tmp/btsys] df -h |head -100 |egrep -v "platform|mnttab|swap|objfs|devices|ctfs|proc"
Filesystem             size   used  avail capacity  Mounted on
/dev/md/dsk/d10        4.9G   3.6G   1.3G    74%    /
fd                       0K     0K     0K     0%    /dev/fd
/dev/md/dsk/d30        3.9G   370M   3.5G    10%    /var
/dev/md/dsk/d50        1.9G   622M   1.3G    33%    /opt
nas002-2066.phys.pack:/vol/V0264APP001/qopt_sp_com    23G     0K    23G     0%    /opt/SP-common
nas002-2066.phys.pack:/vol/V0264APP001/qopt_sp   7.2G     0K   7.2G     0%    /opt/SP
nas002-2066.phys.pack:/vol/V0264APP001/qvar_sp   7.2G     0K   7.2G     0%    /var/SP
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_product    10G   2.5G   7.5G    26%    /product
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_varsoft   3.0G    26M   3.0G     1%    /varsoft
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_users   512M   1.4M   511M     1%    /users
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_interfaces   1.0G    96K   1.0G     1%    /interfaces
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_catalog   7.0G    45M   7.0G     1%    /catalog
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_Sysload   2.0G     0K   2.0G     0%    /catalog/Sysload
nas002-2066.phys.pack:/vol/v0993STE001/qiu0263_database    32M     0K    32M     0%    /database
[root@iu0263:/var/tmp/btsys]
```

➔Afficher les FS montés et leur taille : bdf (HPUX)

```
[root@su0995: / ] bdf
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     393216  198089  182963   52% /
/dev/vg00/lvol1     319125   77984  209228   27% /stand
/dev/vgS0STE_030/lvvarsoft
                    8388608 1737336 6445346   21% /varsoft
/dev/vgS0STE_030/lv_varsoft_mqm
                    1048576   58911  927833    6% /varsoft/mqm
/dev/vgS0STE_030/varsoft_delivery
                    1048576  598987  421502   59% /varsoft/delivery
/dev/vgS0STE_030/varsoft_delivry_CLFY
                    8388608 6879516 1416791   83% /varsoft/delivery/CLFY
/dev/vgMCLFYPAC1_030/lvvarsoft_bba
                    1032192    2304 1021856    0% /varsoft/bba
/dev/vgMCLFYPAC1_030/lvarsoftCLFY13v2
                    15728640 9760280 5921752   62% /varsoft/CLFY-BAK
/dev/vgMCLFYPAC1_030/lvvarsoftCLFY13v1
                    15728640 10034912 5649432   64% /varsoft/CLFY
/dev/vg00/lvol7    6291456 2386601 3661702   39% /var
/dev/vg00/lvol8    2097152    1614 1964574    0% /var/adm/crash
/dev/vg00/lvol6    4096000 1448243 2482430   37% /usr
/dev/vgS0STE_030/lvusers
                    2097152 1258190  786571   62% /users
/dev/vg00/lvol5    2064384  982290 1031706   49% /tmp
/dev/vgS0STE_030/lvproduct
                    8388608 7830724  544524   93% /product
/dev/vgS0STE_030/lv_product_mqm
                    1048576   12707  971155    1% /product/mqm
/dev/vgMCLFYPAC1_030/lvproduct_bba
                    1032192    2304 1021856    0% /product/bba
/dev/vgMCLFYPAC1_030/lvproductCLFY13v2
➔ www.steria.com    22192128 6925192 15157840   31% /product/CLFY-BAK
[root@su0995: / ]
```

Formation Unix de base

➔Volume en Ko du contenu d'un répertoire : du -s

Usage: `du [-a][-d][-k][-r][-o|-s][-L] [fichier ...]`

•**Exemple :**

```
[root@su0832:/] df -k /data/APP/RDT/DATA_12B
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/vx/dsk/datadg/APP_RDT_DATA_12B
                     134951936 132936554 1889851    99%
/data/APP/RDT/DATA_12B
[root@su0832:/] cd /data/APP/RDT/DATA_12B
[root@su0832:/data/APP/RDT/DATA_12B] du -s *
2       BASEQUALIF
366692  COMMUN
724     DTM_SCORE
1045838 GDM
61066942        KPI3G # <= ce répertoire contient environ ~ 58,24 Go !
0       lost+found
24274416        MYCOM
252732  QOSRDT
70780   QOSRDT_bak
269954  SNOOP
4       TEMP_DAF
178423698       trace_appels
[root@su0832:/data/APP/RDT/DATA_12B]cd KPI3G
[root@su0832:/data/APP/RDT/DATA_12B/KPI3G] du -s *
45219688        Archive
0       DATA
15847254        ENCOURS
[root@su0832:/data/APP/RDT/DATA_12B/KPI3G]cd Archive
[root@su0832:/data/APP/RDT/DATA_12B/KPI3G/Archive]ls |wc -l
     154
[root@su0832:/data/APP/RDT/DATA_12B/KPI3G/Archive]
```

➔Commandes de recherche d'information sur un fichier : whereis, whatis, file

•La commande `whereis`, recherche la localisation du fichier testé par `whereis`, voire l'emplacement du code source et l'emplacement de sa page de manuel :

```
# whereis find              # <= sur linux.
/usr/bin/find /usr/share/man/man1p/find.1p.gz /usr/share/man/man1/find.1.gz
# which gedit
/usr/bin/gedit
#
[root@su0349:/] whereis find            # <= sur Sun Solaris
find: /usr/bin/find /usr/man/man1/find.1
[root@su0349:/]
```

• La commande `whatis` renvoie des informations sur le pages du manuel (man) relatives à la commande ou aux binaire placé après la commande `whatis` :

```
# whatis lp          # <= sur linux.
lp           (4) - line printer devices
lp(lp-cups)  (1) - print files
#
```

```
[root@su0349:/] whatis lp      # <= sur Sun Solaris
lp        lp (1)           - submit print request
lp        lp (7d)          - driver for parallel port
lp        lp (1)           - submit print request
lp        lp (7d)          - driver for parallel port
/usr/5bin/man/windex: Not a directory
[root@su0349:/]
```

Formation Unix de base

➔Commandes de recherche d'information sur un fichier : whereis, whatis, file

•La commande `file` donne le type du fichier testé par cette commande. Exemple :

```
# file *.txt
form.txt: news or mail text
friend.txt: news or mail text
ihave.txt: news or mail text
index.txt: ASCII Java program text, with very long lines, with CRLF line terminators
jargon.txt: news or mail text
mad.txt: news or mail text
more.txt: news or mail text
news.txt: Non-ISO extended-ASCII C program text, with very long lines, with CRLF line terminators
newsdata.txt: Non-ISO extended-ASCII English text, with very long lines, with CRLF line terminators
qad.txt: news or mail text
refrence.txt: news or mail text
robots.txt: ASCII text, with CRLF line terminators
stopwords.txt: ASCII English text, with CRLF line terminators
submit.txt: news or mail text
wm.txt: news or mail text
yhelp.txt: news or mail text
#
[root@su0349:/usr/bin] file whois
whois:          ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
[root@su0349:/usr/bin]
```

su1083:root: /var/adm/sa # file /opt/wbem/lbin/cimprovagt
/opt/wbem/lbin/cimprovagt: PA-RISC1.1 shared executable dynamically linked
su1083:root: /var/adm/sa #

➔Recherche de fichiers : 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.

➔Nettoyer ou purger un file system

Il est souvent nécessaire de 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 …), afin de réduire le volumétrie d'un FS :

```
cd /répertoire_racine_de_la_recherche
```

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
```

➔Nettoyer ou purger un file system (suite)

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

```
find . -name "*log" -type f -exec rm {} \;  # <= commande 'rm' à éviter
```

B) diminuer la taille d'un fichier log :

```
tail -1000 fic.log > /tmp/toto : on garde les 100 dernières lignes.
compress fic.log               : on en conserve une archive compressée.
mv /tmp/toto fic.log
```

➔Nettoyer ou purger un file system (suite)

Commande spéciale pour purger le FS « / » (le FS « racine » ou « root ») :
=> Trouver les fichiers du FS « / » (root), ayant plus de 10000000 caractères :

1) Sous Solaris :

```
find / -mount -local -size +10000000c -exec ls -l {} \;
```

Note : les options « **-mount -local** » sont à préciser dans la commande `find`.

2) Sous HP-UX :

```
braconne:root /> find / -xdev -size +10000000c
/core
braconne:root /> ls -l core
-rw-------   1 root       root       29432088 Jan 24 07:00 core
braconne:root /> df -k /
/                    (/dev/vg00/lvol3       ) :   101846 total allocated Kb
                                                    9116 free allocated Kb
                                                   92730 used allocated Kb
                                                      91 % allocation used

braconne:root /> gzip core
braconne:root /> ls -l core*
-rw-------   1 root       root       2905218 Jan 24 07:00 core.gz
braconne:root /> bdf /
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol3     102400   66825   33402   67% /
braconne:root />
```

➔Les éditeurs de textes sous Unix

ed : éditeur ligne
vi , **view** , **vedit** : éditeur plein écran, le plus utilisé.
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.
emacs : éditeur plein écran (on doit l'installer).

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.

L'éditeur de texte vi

`vi` (visual) est un éditeur de texte dit plein écran, qui a été développé à l'université de BERKELEY et qui est devenu standard sous UNIX.
Il fonctionne page par page, d'où son nom visual, au lieu de ligne par ligne. Un curseur permet de se deplacer dans le texte sur tout l'écran.

`vi` ne fonctionne pas directement en mode frappe de texte. Il travaille en mode commande. Tout ce qu'on tape est une commande d'édition. Par exemple `x` supprime le caractère courant et `dd`, la ligne courante. Pour insérer du texte il faut taper `i` (insert) par exemple et ensuite le texte. On termine par la touche escape `<esc>`, qui joue un rôle privilégié.
Comme c'est un éditeur plein écran, il y a un curseur qui indique le caractère courant (donc le mot et la ligne courants). On lance vi par :

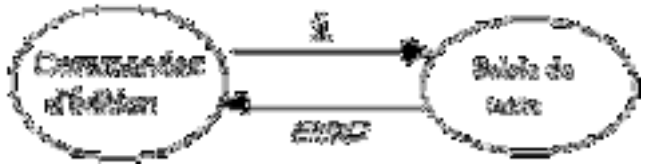
```
% vi [fichier]
```

L'éditeur vi

=> pour éditer le texte contenu dans fichier, sinon un nouveau texte qu'on saisira (et qu'on sauvegardera dans un fichier après). Le texte est mis dans un buffer (une zone mémoire), qui supportera temporairement les modifications apportées. Si un texte ne remplit pas tout l'écran, vi affiche un ~ en début des lignes restantes (toutes les lignes si aucun texte).

Pour rappel, il y a deux modes pour l'utilisation de « `vi` »:

- mode commande
- mode frappe.



Quand vous lancez l'éditeur, vous êtes en mode commande.
Sauf indication contraire, les commandes n'attendent pas la touche `Entrée` pour être exécutées.

➔L'éditeur vi (suite)

➔Mode frappe (insertion)

`i` insérer du texte à l'endroit du curseur : **i***texte*<esc>
`a` ajouter du texte après le curseur : **a***texte*<esc>
`A` ajoute du texte en fin de la ligne courante : **A***texte*<esc>
`o` ouvre une nouvelle ligne au-dessus de / avant la ligne courante : **O***texte*<esc>
`o` ouvre une nouvelle ligne au-dessous de / après la ligne courante : **o***texte*<esc>

Après un de ces caractères, vous êtes en mode frappe. Vous pouvez taper le texte que vous voulez. Pour revenir en mode commande, utilisez la touche `Esc`.

Sous `vim`, une mention en bas de l'écran vous indique quand vous êtes en mode insertion. Sous `vim`, vous pouvez utiliser les flèches pendant que vous êtes en mode insertion, ce n'est pas le cas pour `vi` (`vi` date d'avant qu'il y ait les flèches sur le clavier).

➔L’éditeur vi (suite)

➔Mode frappe (insertion) (suite)

Attention cependant, les lignes continuent jusqu'à ce que vous tapez Entrée, c'est-à-dire qu'il n'y a pas de retour a la ligne automatique, a moins que vous l'ayez programme!
En tapant du texte, on doit donc se souvenir de taper Entrée a la fin des lignes.

➔L'éditeur vi (suite)

➔Mode frappe (insertion) (suite)

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>

➔L'éditeur vi (suite)

➔Mode commande

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

➔L'éditeur vi (suite)

➔Mode commande (suite)

Mode ligne	
:w	Sauvegarde le fichier
:wq	Sauvegarde et quitte le fichier
:q	Quitte sans sauvegarder
:q!	Quitte sans rien sauvegarder du tout

L'éditeur vi (suite)

Sauf pour `ZZ`, ces commandes nécessitent la touche Entrée :

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)

➔L'éditeur vi (suite)

➔Affichage

Ctrl-L	Re-affiche l'écran (mode commande ou mode frappe)
z <CR>	Re-affiche l'écran avec la ligne courante en haut
z-	Re-affiche l'écran avec la ligne courante en bas
z.	Re-affiche l'écran avec la ligne courante au centre

➔L'éditeur vi (suite et fin)

➔Commandes de recherche

Les commandes de cette section doivent être suivies de la touche entrée :

/xxx	recherche xxx en avançant a partir du curseur
?xxx	recherche xxx en reculant (xxx est n'importe quelle expression régulière ... par exemple /a-zA-Z trouvera tous les majuscules qui suivent une minuscule. C'était juste pour vous laisser apercevoir les possibilités de vi ... man regexp :-))
/ et ?	répètent la dernière recherche. Notez en passant que ce syntaxe est le même que dans less ! C'est bien utile pour chercher dans les man volumineux.
:%s/regexp1/regexp2/g	remplace regexp1 par regexp2 dans tout le fichier. Le pourcent signifie "tout le fichier", vous pouvez le remplacer par "m,n" : première et dernière ligne de la recherche. Le g fait plusieurs remplacements par ligne si nécessaire, si vous omettez /g seul la première occurrence sera remplacée.

Formation Unix de base

➔Affichage du début d'un fichier texte : head

• « head » : Afficher les n premières lignes d'un fichier texte :

head -n *<nombre de ligne> <fichier>*

Exemple :

```
[root@su0832:/etc]cat hosts
#
# Internet host table
#
127.0.0.1       localhost
10.42.209.17    su0832  su0832adm svg402-979adm svg402-979
loghost
10.42.67.16     su0832cli
10.42.145.18    su0832svg svg402-979svg
10.42.188.34    su0832nfs
10.42.145.7     su0467 svg101-979svg1 svg101-979
10.42.145.3     su0349svg

164.17.42.136   nebraska
10.42.145.13    su0969svg svg302-979svg
[root@su0832:/etc]head -9 hosts
#
# Internet host table
#
127.0.0.1       localhost
10.42.209.17    su0832  su0832adm svg402-979adm svg402-979
loghost
10.42.67.16     su0832cli
10.42.145.18    su0832svg svg402-979svg
10.42.188.34    su0832nfs
10.42.145.7     su0467 svg101-979svg1 svg101-979
[root@su0832:/etc]
```

Formation Unix de base

➔Affichage de la fin d'un fichier texte : tail

• « tail » : Afficher les n dernières lignes d'un fichier texte :

tail -n *+<n° ligne debut> <fichier>*

Exemples:

```
[root@su0832:/etc]cat hosts
#
# Internet host table
#
127.0.0.1       localhost
10.42.209.17    su0832  su0832adm svg402-979adm svg402-979 loghost
10.42.67.16     su0832cli
10.42.145.18    su0832svg svg402-979svg
10.42.188.34    su0832nfs
10.42.145.7     su0467 svg101-979svg1 svg101-979
10.42.145.3     su0349svg

164.17.42.136   nebraska
#
10.42.145.13    su0969svg svg302-979svg

# rajout pour CTLM a la demande de Franck DEGOULET, le mar. 03/08/2010
11:09
10.42.221.4 su0419adm psv1ac2pr1
10.42.90.4 su0419cli
10.42.221.7 su0421adm bsv1ac2pr1
10.42.90.7 su0421cli
[root@su0832:/etc]tail -5 hosts
# rajout pour CTLM a la demande de Franck DEGOULET, le mar. 03/08/2010
11:09
10.42.221.4 su0419adm psv1ac2pr1
10.42.90.4 su0419cli
10.42.221.7 su0421adm bsv1ac2pr1
10.42.90.7 su0421cli
[root@su0832:/etc]
```

•Autre exemple :

Contenu du fichier toto :

```
azerty1
azerty2
azerty3
azerty4
azerty5
azerty6
azerty7
azerty8
```

```
# tail -n +3 toto | head -n 4
azerty3
azerty4
azerty5
azerty6
```

Formation Unix de base

➔Commandes pour compresser un fichier : compress, gzip, zip
(pour gagner de la place disque).

• « **compress** », « **gzip** », « **zip** » : compresser un ou plusieurs fichiers :

Exemples:

```
gzip filename(s) / compress filename(s) / zip -r filename_arch.zip file(s)
gzip filename_arch.gz file1 file2 file3 /users/u087751/formation.txt

[root@su1401: /varsoft/edi2/err ] tail -1000 trace.log > trace.log2
[root@su1401: /varsoft/edi2/err ] gzip trace.log
[root@su1401: /varsoft/edi2/err ] mv trace.log2 trace.log
[root@su1401: /varsoft/edi2/err ] ls -l trace.log*
-rw-r--r--   1 root       sys          239051 Jan 26 13:42 trace.log
-r--r--r--   1 nedt       editique   10263091 Nov 28 17:56 trace.log.gz
[root@su1401: /varsoft/edi2/err ]
```

•« **uncompress** », « **ungzip** », « **unzip** » : décompresser un ou plusieurs fichiers :
(commandes inverses des précédentes).

```
gunzip filename.gz  / unzip filename_arch.zip / uncompress filename.Z
```

Formation Unix de base

➔Commandes pour compresser un fichier : compress, gzip, zip

(pour gagner de la place disque).

- Exemple d'utilisation de "compress" et de "uncompress" :

```
[root@su0832:/etc]ls -l hosts*
lrwxrwxrwx   1 root     root          12 jui 25  2006 hosts -> ./inet/hosts
-r--r--r--   1 root     sys          329 mai  6  2008 hosts.03aout2010
[root@su0832:/etc]compress hosts.03aout2010
[root@su0832:/etc]ls -l hosts.03aout2010*
-r--r--r--   1 root     sys          227 mai  6  2008 hosts.03aout2010.Z
[root@su0832:/etc]uncompress hosts.03aout2010.Z
[root@su0832:/etc]ls -l hosts.03aout2010
-r--r--r--   1 root     sys          329 mai  6  2008 hosts.03aout2010
[root@su0832:/etc]
```

La commande d'archivage: tar

•La commande tar (tape archive) regroupe, en un seul fichier, toute une arborescence.

•Principale option :

```
c : crée une nouvelle archive.
x : extrait des fichiers d'une archive.
t : affiche le contenu d'une archive.
r : ajoute des fichiers à une archive.
u : met à jour les fichiers de l'archive.
d : compare les fichiers contenus dans l'archive à ceux présents sur le disque dur.
v : mode verbeux.
vv : mode verbeux (donne beaucoup plus d'informations).
k : N'écrase aucun fichier.
f fichier : spécifie le nom du fichier d'archive à lire ou écrire.
Z : Pour compresser / décompresser avec gzip avant d'archiver.
Z : Pour compresser / décompresser avec compress.
```

Syntaxe : `tar -options nom_du__fichier_archive_tar repertoire(s)`

Exemple d'un archivage : `tar -cvf /tmp/sacha.tar /home/sacha`
Exemple de l'affichage du contenu : `tar -tvf /tmp/sacha.tar`
Exemple d'extraction du contenu : `tar -xvf sacha.tar`

Note : La commande **tar** permet de rassembler plusieurs fichiers pour les stocker. Elle est avant tout destinée à l'archivage, mais elle peut aussi être employée pour copier des fichiers, des répertoires.

➔Commande ou utilitaire ‘awk’ (Filtre)

➔awk est un langage de programmation utilisé pour manipuler les données et pour générer des rapports.

➔Les données peuvent provenir de l'entrée standard, un ou plusieurs fichiers, ou comme sortie d'un processus.

➔awk analyse un fichier (ou entrée) ligne par ligne, de la première à la dernière ligne, recherchant les lignes qui correspondent à un modèle spécifique et effectue des actions sélectionnées (précisées entre accolades « { » et « } ») sur ces lignes.

➔S'il ya un modèle (alias « pattern », c’est-à-dire une chaîne de caractères à rechercher), mais sans aucune action spécifique, toutes les lignes qui correspondent au modèle sont affichés.

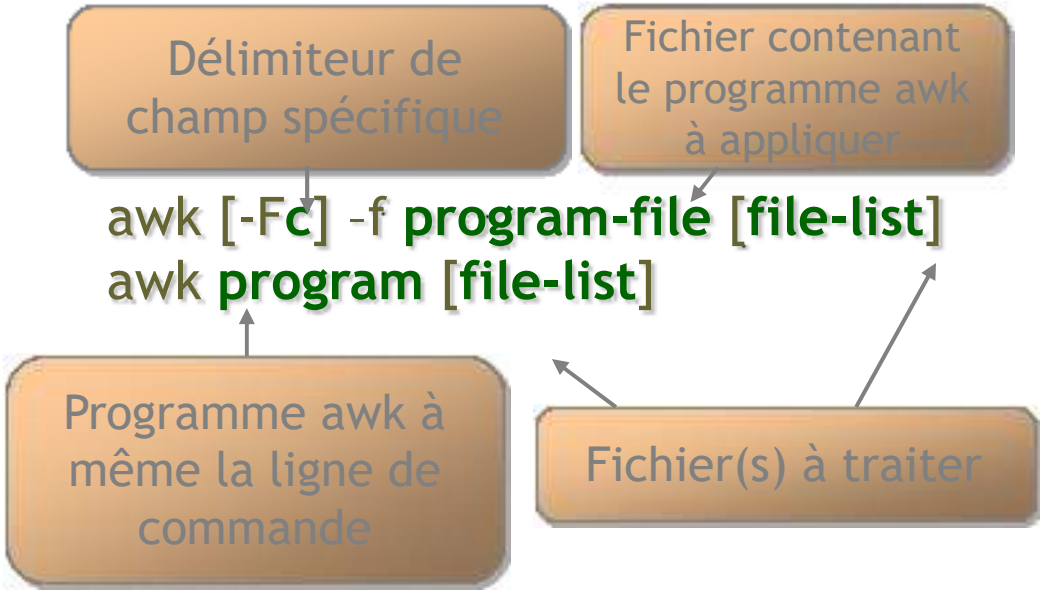
➔Si il ya une action sans modèle (i.e. pattern), toutes les lignes, en entrée, spécifiées par l'action sont exécutées.

➔Commande 'awk' (Filtre)

Utilité

L'utilitaire **awk** recherche un ou plusieurs chaînes de caractères (**patterns**) dans un ou plusieurs fichiers et applique certaines **actions** sur les enregistrements sélectionnés.

Syntaxe



Note: délimiteurs par défaut: SPACE *et* TAB

(Filtre)

➔Commande ‘awk’

Awk : Un langage de manipulation de fichiers texte avec (ayant) des colonnes.

Imprimer les deux premiers champs dans l'ordre inverse:

```
awk '{print $ 2, $ 1}' fichier
```

Affiche des lignes de plus de 72 caractères:

```
awk 'length> 72' fichier
```

Imprimer longueur de la chaîne de la 2e colonne

```
awk '{print length($2)}' fichier
```

(Filtre)

➔Commande 'awk'

Patterns (modèles)

Expressions régulières, **/regex/**, avec ou sans opérateurs **~** (=) et **!~** (!=)

exemples:

/tion/
*enregistrements **contenant** la chaîne **tion***

$1 ~ /tion/
*enregistrements dont le **1er champ contient tion***

$3 !~ /tion/
*enregistrements dont le **3e champ ne contient pas** la chaîne **tion***

(Filtre)

➔Commande 'awk'

Patterns (suite)

Opérateurs relationnels (fonctionnent sur les nombres ainsi que sur les lettres): **<, <=, ==, !=, >=** et **>**

exemples:

$1 > $2

*enregistrements où la valeur **1er champ** est **supérieure** à celle du **2e***

"$4" == "alpha"

*enregistrements dont le **4e champ est alpha***

➔Commande 'awk' (Filtre)

Patterns (suite)

Opérateurs booléens: || (ou) et **&&** (et)

exemple:

$3 >= 10 && $3 < 20

*enregistrements où la valeur **3e champ** est **supérieure ou égale à 10 et inférieure à 20***

Intervalles (spécifiés avec une virgule)

exemple:

/début/,/fin/

*enregistrements inclus **entre les bornes /début/ et /fin/**. Les bornes sont **incluses** dans l'intervalle. Le traitement de l'intervalle est **récurrent**.*

➔Commande 'awk' (Filtre)

Patterns (suite)

Identificateurs uniques: BEGIN et END

*l'identificateur **BEGIN** permet d'effectuer une action **avant** de commencer le traitement des fichiers.*
*l'identificateur **END** permet d'effectuer une action **après** que le traitement des fichiers soit complété.*

exemple:

BEGIN { print "Début du traitement" }
*affiche la chaîne **"Début du traitement"** sur la sortie standard au lancement du programme, **avant** de commencer le traitement du/des fichier(s).*

➔Commande 'awk' (Filtre)

Actions

- Action **par défaut: print**

  print *affiche sur la sortie standard.*
  il est possible de rediriger la sortie avec **>**, **>>** *et* **|**.

- Il est possible de spécifier **plusieurs actions sur une même ligne** avec **;** :

  pattern { **action; action; action;** }

- Les actions peuvent être composées d'**opérations logiques et arithmétiques**, de **fonctions**, de **variables**, de **tableaux associatifs**, de **structures itératives** et de **structures conditionnelles**.

(Filtre)

➔ Commande ‘awk’ (suite)

Imprime les champs dans l’ordre inverse :

```
awk '{ for (i = NF; i > 0; --i) print $i }' file
```

imprime tous les lignes entre la paire start/stop :

```
awk '/start/, /stop/' file
```

imprime toutes les lignes dont le 1er champs est différent du précédent :

```
awk '$1 != prev { print; prev = $1 }' file
```

➔ Commande ‘awk’ (suite)

(Filtre)

Imprime la colonne 3 si colonne 1 > colonne 2:

```
awk '$1 > $2 {print $3}' file
```

Imprime la ligne si la colonne 3 > colonne 2:

```
awk '$3 > $2' file
```

Compte le nombre de lignes si col 3 > col 1

```
awk '$3 > $1 {print i + "1"; i++}' file
```

Imprime le numéro de séquence et après la colonne 1 du fichier:

```
awk '{print NR, $1}' file
```

Commande de tri : sort

- Utilisé pour trier alphabetiquement le contenu d'un fichier the contents d'un fichier, ligne par ligne :

```
sort filename
```

ou

```
cat filename | sort
```

ou

```
sort -u filename
```
: ← - u : Après le tri, on ne garde pas les doublons

Commande de tri : uniq

Filtrer les lignes répétées (en double) dans un fichier :

`uniq` *`filename`*

ou

`cat` *`filename`* `| sort`

ou

`uniq -u` *`filename`* : ← - u : Après le tri, on ne garde pas les doublons

➔Fusionner le contenu de 2 ou plusieurs fichiers texte : paste

Contenu du fichier *file1*:	Contenu du fichier *file2*:

`Jack Wallen`
`Jessica Wallen`
`Johnny Wallen`
`Jeri Wallen`
	`123-45-6789`
	`234-56-7890`
	`345-67-8901`
	`456-78-9012`

En exécutant la commande « `paste file1 file2 > file3` », le contenu du fichier *file3* sera le suivant :

```
Jack Wallen      123-45-6789
Jessica Wallen   234-56-7890
Johnny Wallen    345-67-8901
Jeri Wallen      456-78-9012
```

➔Comparaison de fichiers texte (ascii) : diff, cmp

• Par exemple, comparer la version actuel du fichier « /etc/system » avec une version plus ancienne de ce fichier, par la commande « diff » :

```
[su0963@root:/etc] diff system system.sav.20081015
112c112
< set semsys:seminfo_semmnu=16384   # <= le paramètre seminfo_semmnu est passé de la valeur 10000 à 16384.
---
> set semsys:seminfo_semmnu=10000
123,126c123,126
< set msgsys:msginfo_msgmax=65535
< set msgsys:msginfo_msgmnb=2097152
< set msgsys:msginfo_msgssz=128
< set msgsys:msginfo_msgtql=8192
---
> set msgsys:msginfo_msgmax=4096
> set msgsys:msginfo_msgmnb=65535
> set msgsys:msginfo_msgssz=32
> set msgsys:msginfo_msgtql=1024
132c132
< set rlim_fd_cur=10000  # <= le paramètre rlim_fd_cur est passé de la valeur 1024 à la valeur 10000.
---
> set rlim_fd_cur=1024
[su0963@root:/etc]
```

➔Comparaison de fichiers texte (ascii) : cmp, vim (suite)

•La commande « **cmp** » (sans argument) donne un résultat plus simple : elle indique uniquement les lignes qui sont différentes entre les 2 fichiers à comparer, sans autre précisions:

```
[su0963@root:/etc] cmp system system.sav.20081015
system system.sav.20081015 differ: char 2693, line 112
[su0963@root:/etc]
```

En utilisant la commande « **vim** » (i.e. « vi » en mode lecture, pour Linux), de la façon suivante, on obtient un résultat plus agréable (voir son résultat page suivante) :
⇒Avec "**vim**", vous ouvrez votre 1er fichier, puis dans **vim** vous tapez la commande :

```
:vert diffsplit nom_2nd_fichier
```

⇒Vous aurez le résultat souhaité, à savoir 2 fenêtres, séparé par un long trait vertical au milieu des 2 fenêtres. Exemple :

```
[su0963@root:/etc] vim system
:vert diffsplit system.20081015
```

Formation Unix de base

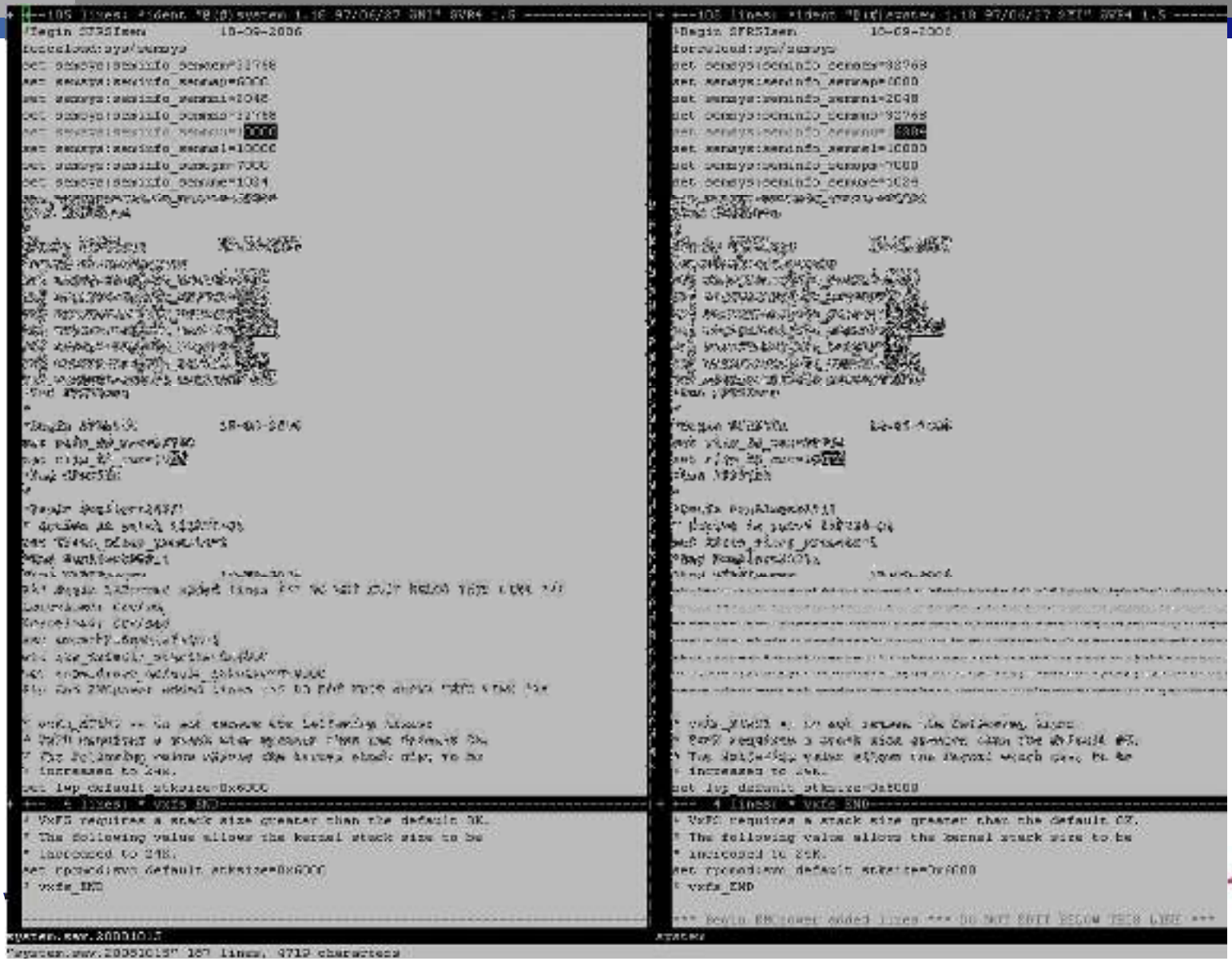
```
+ +--105 lines: *ident "@(#)system 1.18 97/06/27 SMI" SVR4 1.5 ---------------|+ +--105 lines: *ident "@(#)system 1.18 97/06/27 SMI" SVR4 1.5 ---------------
  *Begin SFRSIsem          18-09-2006                                         |  *Begin SFRSIsem          18-09-2006
  forceload:sys/semsys                                                        |  forceload:sys/semsys
  set semsys:seminfo_semaem=32768                                             |  set semsys:seminfo_semaem=32768
  set semsys:seminfo_semmap=6000                                              |  set semsys:seminfo_semmap=6000
  set semsys:seminfo_semmni=2048                                              |  set semsys:seminfo_semmni=2048
  set semsys:seminfo_semmns=32768                                             |  set semsys:seminfo_semmns=32768
  set semsys:seminfo_semmnu=10000                                             |  set semsys:seminfo_semmnu=16384
  set semsys:seminfo_semmsl=10000                                             |  set semsys:seminfo_semmsl=10000
  set semsys:seminfo_semopm=7000                                              |  set semsys:seminfo_semopm=7000
  set semsys:seminfo_semume=1024                                              |  set semsys:seminfo_semume=1024
  set semsys:seminfo_semvmx=65534                                             |  set semsys:seminfo_semvmx=65534
  *End SFRSIsem                                                               |  *End SFRSIsem
  *                                                                           |  *
  *Begin SFRSImsg          18-09-2006                                         |  *Begin SFRSImsg          18-09-2006
  forceload:sys/msgsys                                                        |  forceload:sys/msgsys
  set msgsys:msginfo_msgmni=2048                                              |  set msgsys:msginfo_msgmni=2048
  set msgsys:msginfo_msgmap=1026                                              |  set msgsys:msginfo_msgmap=1026
  set msgsys:msginfo_msgmax=4096                                              |  set msgsys:msginfo_msgmax=65535
  set msgsys:msginfo_msgmnb=65535                                             |  set msgsys:msginfo_msgmnb=2097152
  set msgsys:msginfo_msgssz=32                                                |  set msgsys:msginfo_msgssz=128
  set msgsys:msginfo_msgtql=1024                                              |  set msgsys:msginfo_msgtql=8192
  set msgsys:msginfo_msgseg=65535                                             |  set msgsys:msginfo_msgseg=65535
  *End SFRSImsg                                                               |  *End SFRSImsg
  *                                                                           |  *
  *Begin SFRSIfd           18-09-2006                                         |  *Begin SFRSIfd           18-09-2006
  set rlim_fd_max=32768                                                       |  set rlim_fd_max=32768
  set rlim_fd_cur=1024                                                        |  set rlim_fd_cur=10000
  *End SFRSIfd                                                                |  *End SFRSIfd
  *                                                                           |  *
  *Begin SunAlert28311                                                        |  *Begin SunAlert28311
  * Active le patch 112254-01                                                |  * Active le patch 112254-01
  set TS:ts_sleep_promote=1                                                   |  set TS:ts_sleep_promote=1
  *End SunAlert28311                                                          |  *End SunAlert28311
  *End SFRSIparam          18-09-2006                                         |  *End SFRSIparam          18-09-2006
  *** Begin EMCpower added lines *** DO NOT EDIT BELOW THIS LINE ***          |  ----------------------------------------------------------------------
  forceload: drv/sd                                                           |  ----------------------------------------------------------------------
  forceload: drv/ssd                                                          |  ----------------------------------------------------------------------
  set emcp:bPxEnableInit=1                                                    |  ----------------------------------------------------------------------
  set lwp_default_stksize=0x6000                                              |  ----------------------------------------------------------------------
  set rpcmod:svc_default_stksize=0x6000                                       |  ----------------------------------------------------------------------
  *** End EMCpower added lines *** DO NOT EDIT ABOVE THIS LINE ***            |  ----------------------------------------------------------------------
                                                                              |
  * vxfs_START -- do not remove the following lines:                         |  * vxfs_START -- do not remove the following lines:
  * VxFS requires a stack size greater than the default 8K.                  |  * VxFS requires a stack size greater than the default 8K.
  * The following value allows the kernel stack size to be                   |  * The following value allows the kernel stack size to be
  * increased to 24K.                                                        |  * increased to 24K.
  set lwp_default_stksize=0x6000                                              |  set lwp_default_stksize=0x6000
+ +--  4 lines: * vxfs_END----------------------------------------------------|+ +--  4 lines: * vxfs_END----------------------------------------------------
  * VxFS requires a stack size greater than the default 8K.                  |  * VxFS requires a stack size greater than the default 8K.
  * The following value allows the kernel stack size to be                   |  * The following value allows the kernel stack size to be
  * increased to 24K.                                                        |  * increased to 24K.
  set rpcmod:svc_default_stksize=0x6000                                       |  set rpcmod:svc_default_stksize=0x6000
  * vxfs_END                                                                 |  * vxfs_END
    ➔www.steria.com                                                         |
  ----------------------------------------------------------------------------|  *** Begin EMCpower added lines *** DO NOT EDIT BELOW THIS LINE ***
system.sav.20081015                                                           |  system
"system.sav.20081015" 187 lines, 4719 characters
```

➔**Comparaison de fichiers texte (ascii) : vim (suite)**

➔En rouge souligné, les zones de texte différents entre les 2 fichiers.

steria

Formation Unix de base



➔Com-paraison de fichiers texte (ascii) : vim (suite et fin)

➔ En surligné noir => les caractères différents entre les 2 fichiers.

steria

➔Découper des champs sélectionnés de chaque ligne d'un fichier : cut

(Filtre)

Figure 4-4 Using the cut command

← Extraction des champs d'un fichier en utilisant la redirection de sortie: la commande « cut » (couper) supprime des colonnes ou des champs à partir d'un fichier.

Substitution de caractères : tr (Filtre)

- recopie **stdin** sur **stdout** en substituant des caractères
- syntaxe: **tr [-cds] [s1 [s2]]**
- options:
  - **-c** (complément de **s1**)
  - **-d** efface les car. de **s1**
  - **-s** tte séquence dans **s1** est substituée par un car. unique dans **s2**
- ex:
  - **tr A-Z a-z < essai**
    remplace les majuscules par des minuscules
  - **tr A-Z a-z < essai | tr -sc a-z '\012'**
    remplace les majuscules par des minuscules, puis remplace tout ce qui n'est pas une lettre minuscule par un retour chariot ('\012')

Formation Unix de base

Le shell (interpréteur et langage de commande Unix)

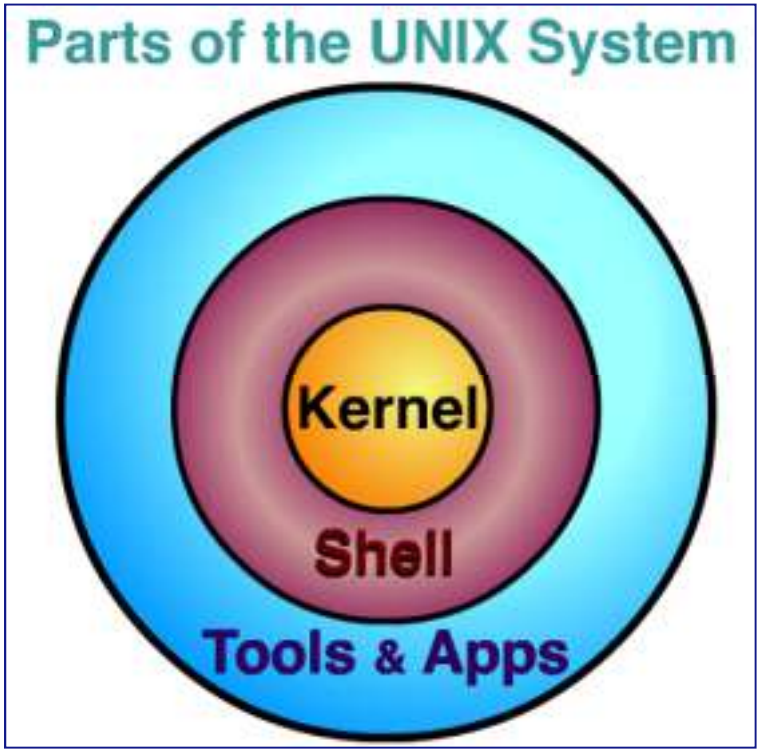
- Une fois connecté, le système ouvre une session à notre nom et attend nos instructions via un programme spécial:
- Le Shell = interpréteur de commandes
  - interface utilisateur “de base” (interlocuteur avec le syst.)
  - interprétation ligne à ligne
  - plusieurs shells: sh (Bourne shell), csh (C shell), tcsh, bash, ksh (Korn shell), zsh, ...
  - configurable: fichiers d’environnement (commençant par un “`.`”), présent dans le répertoire d’accueil de l’utilisateur (la « HOME DIRECTORY » (°)) :
    - “`.login`”
    - “`.logout`”
    - “`.bashrc`”
  - langage de programmation

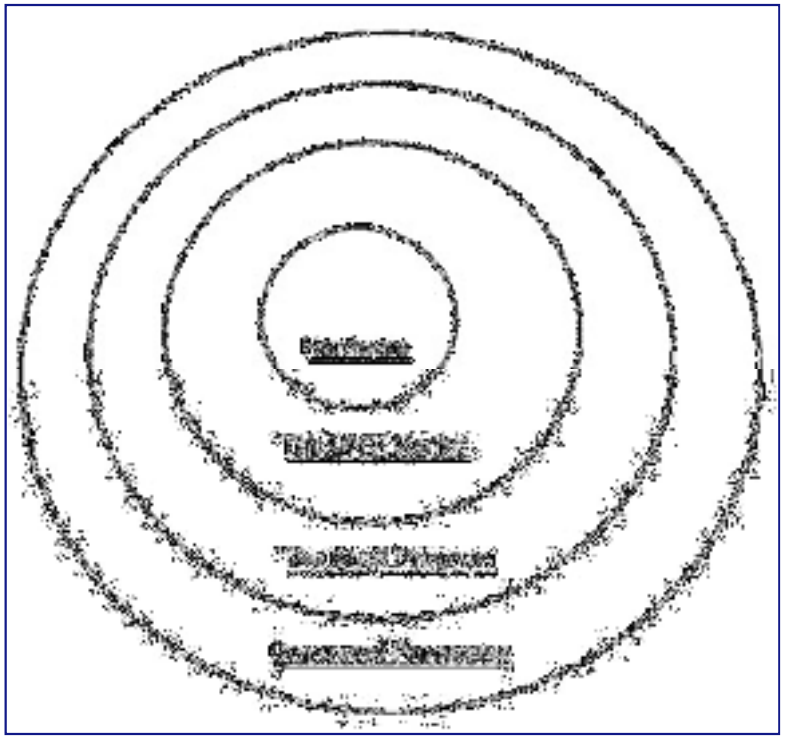
- shell par défaut sous Linux : bash
- shell par défaut sous Solaris : sh ou bash

(°) renseigné dans la variable Unix « $HOME ».

Formation Unix de base

➔Le shell (interpréteur et langage de commande Unix) (suite)





↑ Où se situe l'interpréteur shell (interface) entre l'homme / les applications et outils et le noyau (le cœur) du système d'exploitation (O.S.) Unix (le « kernel ») _ schéma des couches, en pelures d'oignon successives.

Formation Unix de base

➔**Le langage shell un langage très compact, mais assez ésotérique (°)**

Les commandes sont très courtes :

➔« cp » pour « copy ».

➔« rm » pour « remove ».

➔« mv » pour « move »

➔« tar » pour « tape archive record ».

➔Etc. etx.

Humour Unixien.

(°) c'est-à-dire difficile à comprendre pour les non initiés (très « chinois »).

➔Ecriture des scripts et leurs exécution

On crée, en général, les scripts, à exécuter, contenant des commandes shell, sous l'éditeur « vi » (voir présentation de « vi » plus haut dans ce document).
Exemple :

```
# vi script1.sh    # voir un exemple de script page suivante.
```

On lui donne éventuellement les droits d'exécution, par exemple avec cette commande :

```
# chmod 755 script1.sh      ou bien     # chmod u+x g+x o+w script1.sh
```

Puis on exécute le scripts, en le lançant de la façon suivante :

```
# sh script1.sh #(en Bourne shell)ou # ksh script1.sh (en Korn shell).

Ou bien

# ./script1.sh  # <= si on lui a déjà donné des droits d'exécution
```

Formation Unix de base

➔Exécuter une commande à l'abri de hangups ou kill : nohup

Pour éviter que l'exécution d'une commande puisse interrompue, on lance cette commande en la faisant précéder de la commande « nohup », comme dans l'exemple ci-après :

Exemple: `nohup program`

Pour arrêter votre tâche (jobs) :

```
[ivr64~root] stop pid_de_tache(s)
[ivr64~root]
```

Si vous souhaitez exécuter une commande et fermer la session, en cours d'exécution de la commande (sans que l'exécution de cette dernière s'interrompe), vous lancez votre commande en « `nohup` » (en exécution non interruptible).

Si vous voulez lancer la commande en arrière-plan (en « background ») et vous voulez retourner immédiatement à votre travail (et retourner à votre prompt shell « `$` » ou « `#` »), vous lancer alors (la commande était terminée par le « et » commercial : «**&** »):

```
$ nohup program &
$
```

Pour vérifier que votre tâche (jobs) est toujours lancée en arrière-plan :

```
[ivr64~root] jobs
[ivr64~root]
```

➔Planification de l'exécution de tâches (jobs) : at

`at` : exécute des commandes à un moment précis.
`atrm` : supprime des tâches (jobs), identifiés par leur numéro de job.

`at -m 01:35 < atjob` : Exécute les commandes listées dans le fichier `'atjob'` à 01h35, en plus de toutes les sorties qui sont générées, par mail du job, pour l'utilisateur qui exécute la tâche. Lorsque cette commande a été achevée avec succès, vous devrez obtenir un prompt qui ressemble à celui de l'exemple ci-dessous :

```
commands will be executed using /bin/csh
job 1072250520.a at Wed Dec 24 00:22:00 2003
```

`at -l` : Cette commande va lister chacune des tâches planifiées, comme ci-après :

```
# at -l
1072250520.a Wed Dec 24 00:22:00 2003
#
```

`at -r` : Supprime le travail (la tâche ou le job) vient d'être créé.
ou
`atrm 23` : Supprime la tâche (le job) n° 23.

Formation Unix de base

➔Afficher ou ré-exécuter la/les commande(s) que vous venez de passer : history

```
# history | more
1 service network restart
2 exit
3 id
4 cat /etc/redhat-release
# !4
cat /etc/redhat-release # <= la commande est ré-exécutée.
Fedora release 9 (Sulphur) # <= résultat de l'exécution du « cat … ».
#
```

Exemple de scripts

```
# script check_instance_bge.ksh : pour afficher la vitesse et le mode des pattes réseau :
                          (si pattes de type bge).
#!/bin/ksh
for i in 0 1 2
do
echo "instance :" bge$i
echo "******************************"
echo "1000 full duplex"
ndd -get /dev/bge$i adv_1000fdx_cap
echo "1000 half duplex"
ndd -get /dev/bge$i adv_1000hdx_cap
echo "100 full duplex"
ndd -get /dev/bge$i adv_100fdx_cap
echo "100 half duplex"
ndd -get /dev/bge$i adv_100hdx_cap
echo "10 full duplex"
ndd -get /dev/bge$i adv_10fdx_cap
echo "10 half duplex"
ndd -get /dev/bge$i adv_10hdx_cap
echo "mode autoneg"
ndd -get /dev/bge$i adv_autoneg_cap
echo "******************************"
echo "################################"
done
```

Exemple de scripts

```
# script check_instance_ce.ksh : pour afficher la vitesse et le mode des pattes réseau :
                          (si pattes de type ce).
#!/bin/ksh
for i in 0 1 2 3 9
do
echo "instance :" ce$i
echo "******************************"
ndd -set /dev/ce instance $i
echo "1000 full duplex"
ndd -get /dev/ce adv_1000fdx_cap
echo "1000 half duplex"
ndd -get /dev/ce adv_1000hdx_cap
echo "100 full duplex"
ndd -get /dev/ce adv_100fdx_cap
echo "100 half duplex"
ndd -get /dev/ce adv_100hdx_cap
echo "10 full duplex"
ndd -get /dev/ce adv_10fdx_cap
echo "100 half duplex"
ndd -get /dev/ce adv_10hdx_cap
echo "mode autoneg"
ndd -get /dev/ce adv_autoneg_cap
echo "******************************"
echo "##############################"
done
```

Formation Unix de base

➔Exemple de scripts

#script **purge_utmpx_wtmpx.ksh** : script de purge des fichiers **utmpx** et **wtmps** (dans /var/tmp) :

```
#!/bin/ksh
# ------------------------------------------------------------------------
# Nom du script : purge_utmpx_wtmpx.ksh  Auteur : B. LISAN  Date : 09/06/10
# But : si taille d'un des 2 fichiers "/var/adm/utmpx" , "/var/adm/wtmpx"
# depasse > ~ 10Mo => on purge les 2 fichiers
# Condition execution : cron   date/heure execution : chaque 1er du mois a 0h00
# 0 0 1 * * * /var/tmp/btsys/purge_utmpx_wtmpx.ksh >/dev/null 2>&1
# ------------------------------------------------------------------------
TAILLE_MAX=10240 # en Ko (correspond a 10Mo)

# taille de "/var/adm/utmpx" en Ko
taille_utmpx=` /bin/du -k /var/adm/utmpx | /usr/xpg4/bin/awk '{print $1}'`
# taille de "/var/adm/wtmpx" en Ko
taille_wtmpx=` /bin/du -k /var/adm/wtmpx | /usr/xpg4/bin/awk '{print $1}'`

#echo "taille utmpx=$taille_utmpx Ko"
#echo "taille wtmpx=$taille_wtmpx Ko"

# si la taille d'un des 2 fichiers est superieur ~ a 10Mo, on purge les 2
if [ "$taille_utmpx" -gt $TAILLE_MAX -o "$taille_wtmpx" -gt $TAILLE_MAX ]
then
  # ls -l /var/adm/utmpx
  # ls -l /var/adm/wtmpx
   > /var/adm/utmpx
   > /var/adm/wtmpx
fi
```

Formation Unix de base

➔ **Politique de sécurité concernant la connexion aux serveurs chez SFR**
(compte nominatif, sudo & co)

Selon la nouvelle politique d'accès au serveurs (°), chez SFR :

1. Il faut se connecter au serveurs en ssh
2. Il sera interdit de se connecter directement avec le compte « root » super-utilisateur.
3. Il faudra s'y connecter avec son compte nominatif appelé GAU. Ce compte Unix utilisateur commence toujours par « strb », par exemple : strb2126 (il vous est attribué, ainsi que son mot de passe, par le service GAU).

(°) serveurs Unix du parc SI Mobile (environnements de PFV _ pack _ & PFP _ prod).

Formation Unix de base

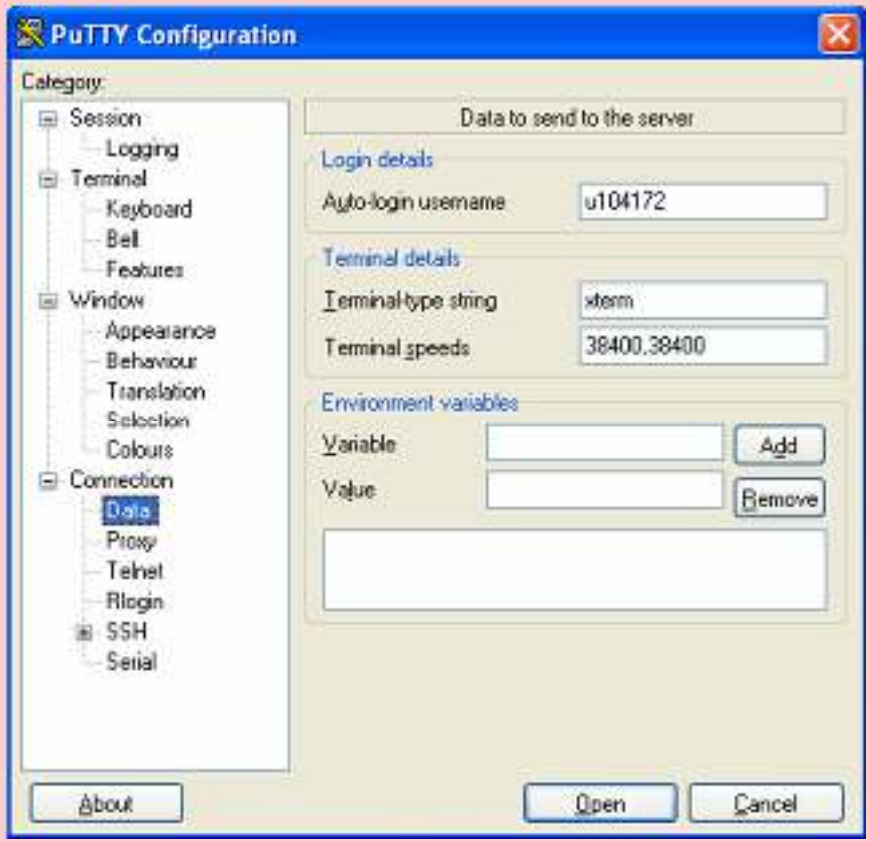
➔Politique de sécurité concernant la connexion aux serveurs chez SFR (suite)
(compte nominatif, sudo & co)

Utilisation du compte GAU pour se connecter

1- Lancer l'utilitaire PuTTy
2- Entrer le nom du serveur dans la zone « Hostname »
3- Cliquer sur « Open »
4- Entrer le compte GAU puis le mot de passe

Pour utiliser automatiquement le compte GAU lors des connexions :

1- Aller dans Connexion / Data
2- Dans « Auto-login username » entrer son compte GAU
3- Puis dans « Session » cliquer sur « Default Settings » et enfin « Save »



Formation Unix de base

➔Politique de sécurité concernant la connexion aux serveurs chez SFR (suite)

(compte nominatif, sudo & co)

Accéder aux privilèges

· **L'équivalent du « su - »**

Pour basculer complètement vers un utilisateur (y compris charger le « .profile » qui est souvent nécessaire pour lancer des traitements ou démarrer des applications) :

sudo –u *utilisateur* –i

Toutefois, elle n'est pas disponible sur tous les serveurs, il faudra alors lancer aux choix :

- pour *bash* : **sudo –u *utilisateur* –H /bin/bash –-login**
- pour *ksh* : **sudo –u *utilisateur* –H /bin/ksh –-login**

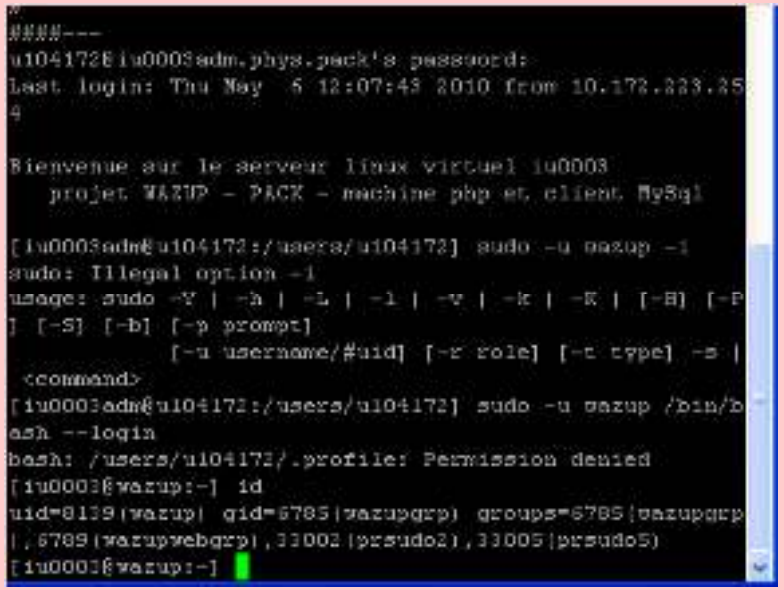
· **L'équivalent du « su utilisateur –c commande »**

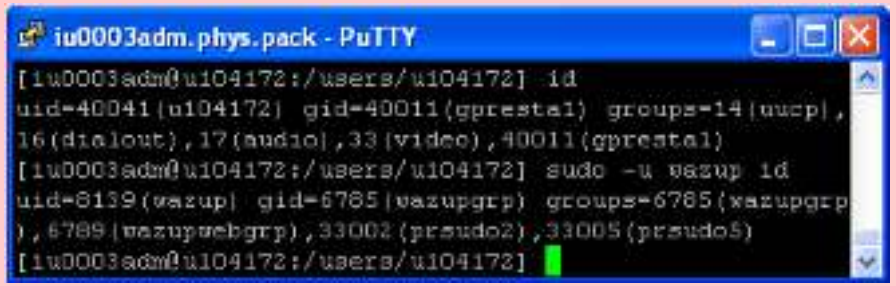
Pour lancer une commande (comme le lancement d'un script nécessitant un utilisateur en particulier) :

sudo –u *utilisateur commande*

Remarques :

- **Ne pas** remplacer « login » par votre login
- Le --login peut être remplacé par « -l » sur certains serveurs





Formation Unix de base

➔Politique de sécurité concernant la connexion aux serveurs chez SFR (suite)
(compte nominatif, sudo & co)

Liste des privilèges sudo 1- Se connecter sur les serveurs 2- Taper la commande **sudo –l** 3- Le résultat indique les différents privilèges en langage sudo	(%prsudo1, %prsudo2) ALL ➔ root + socle système (%prsudo2) ALL ➔ socle système (!%prsudo1, !%prsudo2, %prsudo3) ➔ Comptes applicatifs Autres ➔ Actes délégués (accès à une collection de script)

Formation Unix de base

➔Politique de sécurité concernant la connexion aux serveurs chez SFR (suite)

Exemple →

```
login as: strb2126

    - CECI EST UN SYSTEM INFORMATIQUE PRIVE -

[. . . ]

strb2126@su0832's password:
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   February 2004
###############################################################################################################
# /users PEUT arriver a saturation, Merci de faire le necessaire en vous debarrassant des anciens fichiers      #

*         /\          EN CAS DE REBOOT: MERCI DE      *
*        //\\                                          *
*       //  \\        CONTACTER LE BT-SYS SAS          *
*      //    \\MACHINE SVG402 STORAGE NODE NETWORKER   *
*     // #####\\              DATAZONE SVG101           *
*    //     #   \\            PREVENEZ LE BT-SAS        *
*   //      #    \\            PAR COM MAIL TEL         *
*  //       #     \\AVANT TOUTE REBOOT OU MANIPULATION *
* //_______________\\       SUN MICROSYSTEM           *
*****************************************************
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   February 2004
###############################################################################################################
No news.
su0832:/users/strb2126 [strb2126]> sudo su -
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   February 2004
###############################################################################################################
# /users PEUT arriver a saturation, Merci de faire le necessaire en vous debarrassant des anciens fichiers      #


*         /\          EN CAS DE REBOOT: MERCI DE      *
*        //\\                                          *
*       //  \\        CONTACTER LE BT-SYS SAS          *
*      //    \\MACHINE SVG402 STORAGE NODE NETWORKER   *
*     // #####\\              DATAZONE SVG101           *
*    //     #   \\            PREVENEZ LE BT-SAS        *
*   //      #    \\            PAR COM MAIL TEL         *
*  //       #     \\AVANT TOUTE REBOOT OU MANIPULATION *
* //_______________\\       SUN MICROSYSTEM           *
*****************************************************
You have mail.
No news.
[root@su0832:/]
```

Formation Unix de base

➔Politique de sécurité concernant la connexion aux serveurs chez SFR (suite)

(compte nominatif, sudo & co)

Remplir le formulaire eDsi « Créer un nouvel utilisateur »

1- Aller sur le portail eDsi http://edsi
2- Sélectionner « Nouvelle demande », puis « Droits et Accès » et « GAU »
3- Dans le menu déroulant choisir « Demande d'accès individuelle »
4- Dans « Besoin » choisir « Créer un nouvel utilisateur »
5- Remplir les champs « Société » et « Entité »
6- Dans Serveurs, indiquer soit une liste exhaustive de serveur, soit un périmètre spécifique type Bloc Applicatif, TMA etc.

Source : Manuel Utilisateur Sudo + Selfcare.pdf

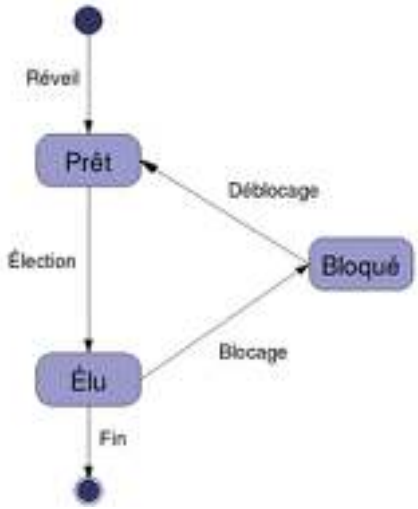
Remarques
- Le champ « nom du compte » peut indiquer un compte eTrust si vous disposiez d'un tel compte (le compte ne doit plus être utilisé et sera alors supprimé)
- Les demandes sont traitées en 48H
- Pour des demandes plus spécifiques vous pouvez contacter le CAU via sa BAL générique CAU (CAU@sfr.com)

Formation Unix de base

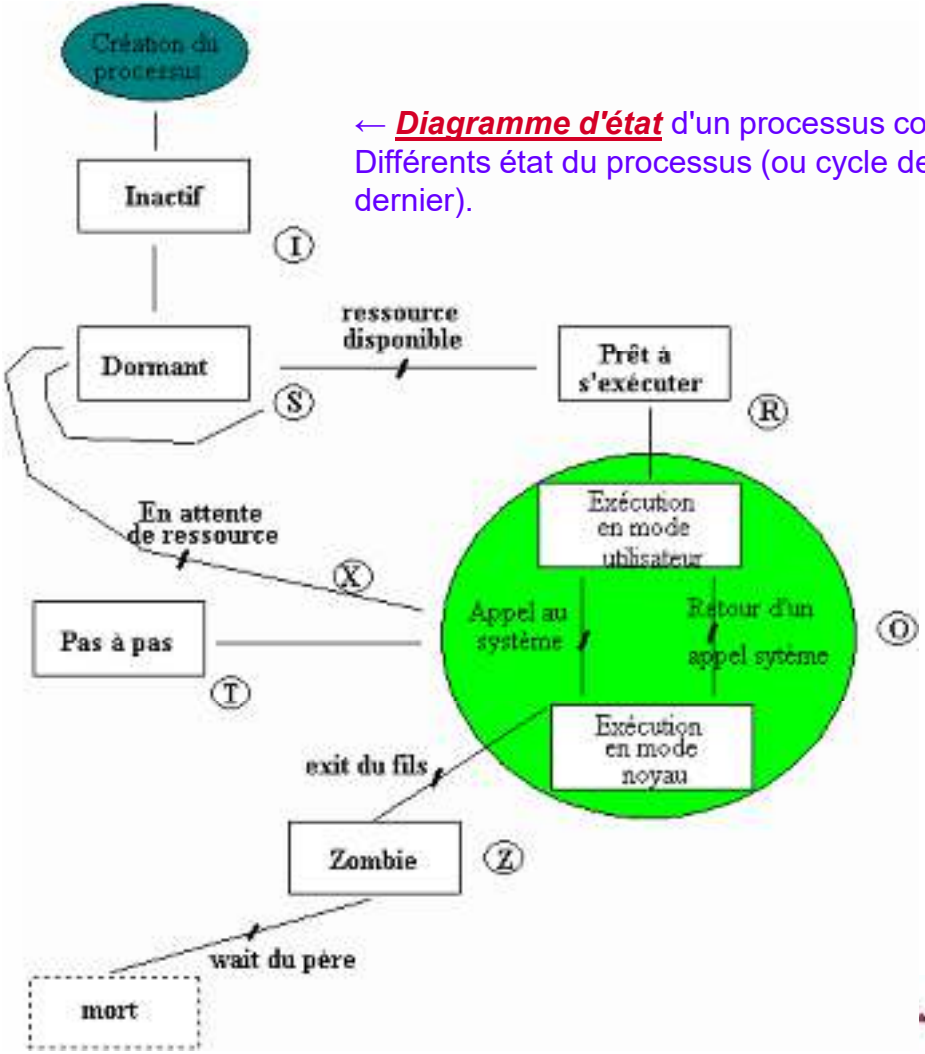
- **Processus : définitions**
- **Processus** (en anglais, process) : en informatique, une opération, plus ou moins complexe, exécutable par un ordinateur.
- Un processus peut être démarré par un utilisateur par l'intermédiaire d'un **périphérique** ou bien par un autre processus.
- Autre définition : C'est un programme (constitué d'instructions élémentaires) en cours d'exécution, en mémoire vive de l'ordinateur.
- Il peut avoir plusieurs états : run (en cours d'exécution), sleep (dormant), wait (en attente), zombi (~ quasiment mort) etc. (voir page suivante).

Formation Unix de base

➔Processus : états



↑ ***Diagramme d'état*** d'un processus simple.



← ***Diagramme d'état*** d'un processus complexe. Différents état du processus (ou cycle de vie de ce dernier).

Formation Unix de base

➔Tuer les processus : kill

Taper la commande "ps" ou "ps -ef", qui liste de tous les processus en cours d'exécution (ps donne un rapport instanné de tous les processus en cours).
2. Dans la liste des processus en cours affichés par la comande « `ps` », retrouver le programme que vous souhaitez mettre à mort et noter son numéro ID de processus (pid).
3.Alors taper « `kill` *`pid`* », avec « *`pid`* » : le numéro du processus id réel.
Exemple:

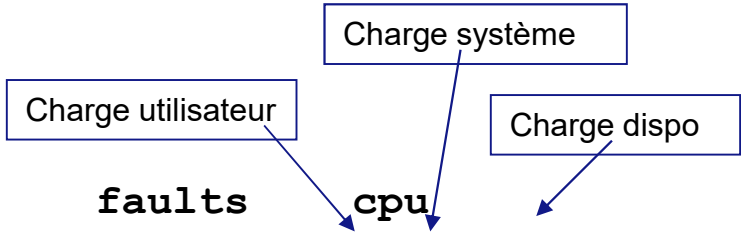
```
[root@iu0263:/] ps -e
PID TTY          TIME CMD
    0 ?           0:08 sched
    1 ?           1:01 init
    2 ?           0:00 pageout
    3 ?         561:49 fsflush
[ . . . ]

 4899 ?           0:00 sshd
[root@iu0263:/] kill 4899
[root@iu0263:/]
```

<= cette commande va tuer le processus « `sshd` » (pid 4899).

Attention !!! La commande « rm » est dangereuse => Si vous tuez le mauvais processus, vous pouvez rendre votre application ou votre serveur instable.

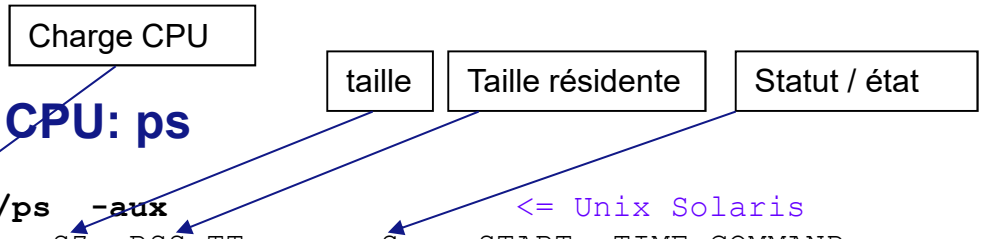
➔Mesurer la charge CPU: vmstat

. vmstat 5 **<= Unix BSD**




```
procs  memory     page                    disk          faults      cpu
r b w avm  free  re  at  pi  po  fr  de  sr  s0  s1  s2  s3  in   sy   cs  us  sy  id
0 0 0 0    2496 0   6   16  0   1   0   0   0   0   0   1   168  336  31  2   7   91
0 0 0 0    2504 0   4   0   0   0   0   0   0   0   0   0   86   231  22  1   2   97
0 0 0 0    2352 0   2   0   0   0   0   0   0   0   0   0   79   211  22  0   1   98
0 0 0 0    2296 0   0   0   0   0   0   0   0   0   0   0   61   205  20  1   2   97
^C.
```

```
[root@su1445:/] vmstat 5      <= Solaris
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po  fr de sr m1 m1 m1 m2   in   sy   cs us sy id
 0 5 0 16657552 31831256 510 4169 1255 298 297 0 0 1 1 1 1 5786 352422 8364 12 1 87
 0 8 0  3064264 22407952 488 54494 2 0 0 0 0 0  0  0  0 9806 928665 19454 39 8 53
 0 7 0  3212544 22559240 161 36375 0 76 30 0 0 0 0 0  0 10105 763968 18090 31 6 63
 0 6 0  3109320 22459896 166 42622 0 29 6 0 0 0 0  0  0 9378 899800 19287 36 6 58
 0 7 0  3186328 22535816  51 36631 2 3 3 0 0  0  0  0  0 9335 808613 17024 31 5 64
^C
[root@su1445:/]
```



➔Mesurer la charge CPU: ps

```
[root@su1445:/] /usr/ucb/ps  -aux                    <= Unix Solaris
USER       PID %CPU %MEM    SZ   RSS TT       S    START  TIME COMMAND
oracle    5025  1.5 16.0 1588295210485400 ?        O 18:03:12  1:08 oraclePMSVA1P1 (DE
sabxfer   1073  1.2  1.7 11887041111600 ?        S   Jan 11 178:19 /bin/java -DTRACE=
oracle   24318  0.9 16.0 1591593610518384 ?        S 15:42:15 23:08 oraclePMSVA1P1 (LO
oracle    3811  0.8 15.9 1594674410429856 ?        O   Jan 12 203:52 oraclePMSVA1P1 (LO
oracle    6741  0.7 16.0 1591619210518616 ?        S 15:51:36 23:18 oraclePMSVA1P1 (LO
oracle   25395  0.7 16.0 1593560010484480 ?        O 18:15:20 124:13 oraclePMSVA1P1 (LO
oracle   29870  0.6 16.0 1591580810518256 ?        O 14:24:00 30:54 oraclePMSVA1P1 (LO
oracle   19684  0.6 15.9 1591825610401376 ?        S   Jan 12 148:55 oraclePMSVA1P1 (LO
[  . . . ]

[root@su1445:/]
```

Mesurer la charge CPU : top

« top » en Unix Solaris :

```
[root@su1445:/] top
last pid: 29534;  load avg:  21.9,  20.8,  19.9;       up 475+18:06:22
10:49:33
1030 processes: 1009 sleeping, 1 stopped, 20 on cpu
CPU states: 60.6% idle, 33.5% user,  5.9% kernel,  0.0% iowait,  0.0% swap
Memory: 64G phys mem, 17G free mem, 31G total swap, 31G free swap

   PID USERNAME LWP PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
 18616 sabxfer    1   0    0   15G   10G cpu/61   3:06  1.57% oracle
 16061 sabxfer   11   0    0   15G   10G cpu/42 197:52  1.56% oracle
   351 oracle     1  60    0   15G   10G sleep   18:53  1.08% oracle
  2132 oracle     1  59    0   15G   10G sleep   41:24  1.00% oracle
 10890 oracle     1  50    0   15G   10G sleep    1:50  0.98% oracle
 19794 oracle     1  40    0   15G   10G cpu/48 265:00  0.91% oracle
 22949 oracle     1  50    0   15G   10G sleep   24:32  0.89% oracle
 21854 oracle     1  54    0   15G   10G sleep  119:02  0.87% oracle
 28891 root       1  19    0 4664K 4448K cpu/54   0:42  0.87% prstat
   116 oracle     1  50    0   15G   10G sleep   36:29  0.87% oracle
  2986 oracle     1   0    0   15G   10G cpu/53  96:55  0.81% oracle
 14326 oracle     1  59    0   15G   10G sleep    2:43  0.80% oracle
  6763 oracle     1  60    0   15G   10G sleep  119:50  0.80% oracle
  3801 oracle     1   0    0   15G   10G cpu/44 268:28  0.79% oracle
 23557 oracle     1  20    0   15G   10G sleep   69:48  0.77% oracle
```

Formation Unix de base

Mesurer la charge mémoire et cpu : prstat

```
[su0238@root:/] prstat
   PID USERNAME  SIZE   RSS STATE   PRI NICE      TIME  CPU PROCESS/NLWP
 22933 zabbix   4696K 1600K sleep    44    5   3:20:24 0.4% zabbix_agentd/1
   593 root     8344K 1672K sleep    59    0   3:01:32 0.3% rmserver/6
  4637 root     3848K 3232K cpu0     59    0   0:00:00 0.1% prstat/1
  1002 root     6528K 2504K sleep    59    0   0:32:22 0.1% dm_primer/1
   950 root       28M   21M sleep    59    0   0:37:36 0.0% ccnfagent/3
   567 root     6952K 1488K sleep    59    0   0:13:54 0.0% mstragent/1
 23788 root       30M   17M sleep    58    0   0:25:11 0.0% msragent/10
 22555 root     7336K 3008K sleep    59    0   0:00:00 0.0% sshd/1
   943 root       37M   25M sleep    59    0   0:12:02 0.0% caf/30
 22558 root     1792K 1224K sleep    59    0   0:00:00 0.0% ksh/1
 22935 zabbix   4464K 1528K sleep    44    5   0:10:39 0.0% zabbix_agentd/1
   954 root       29M   22M sleep    59    0   5:18:43 0.0% ccsmagtd/4
   948 root     2576K 1528K sleep    59    0   0:02:38 0.0% camf/1
  3940 root     5376K 4312K sleep    59    0   0:00:12 0.0% picld/68
   154 root     7080K 4024K sleep    59    0   0:02:42 0.0% nscd/31
   945 root       21M   13M sleep    59    0   0:00:09 0.0% cfsmsmd/12
   660 root       33M   21M sleep    58    0   0:01:36 0.0% nsrexecd/4
   616 root     8080K  544K sleep    59    0   0:00:00 0.0% rmserver/1
 22936 zabbix   4464K 1536K sleep    44    5   0:20:34 0.0% zabbix_agentd/1
 22937 zabbix   4480K 1520K sleep    44    5   0:01:01 0.0% zabbix_agentd/1
 11994 root     6272K 3232K sleep    59    0   0:00:28 0.0% p_ctmat/1
 22934 zabbix   4464K 1528K sleep    44    5   0:06:05 0.0% zabbix_agentd/1
   530 root     4648K 1576K sleep    59    0   0:00:04 0.0% sshd/1
   615 root     7952K  736K sleep    59    0   0:00:00 0.0% rmserver/1
   419 root       10M 4072K sleep    59    0   0:04:10 0.0% snmpd/1
   436 root     3904K 2184K sleep    59    0   0:00:16 0.0% rpc.metad/1
   […]
   344 daemon   5152K 1504K sleep    59    0   0:00:13 0.0% nfsmapid/3
   386 root     2944K  936K sleep    59    0   0:00:00 0.0% automountd/2
   153 root     4016K 1440K sleep    59    0   0:00:00 0.0% devfsadm/6
   326 root     3240K 1288K sleep    59    0   0:00:14 0.0% cron/1
Total: 60 processes, 341 lwps, load averages: 0.06, 0.06, 0.05
[su0238@root:/]
```

➔Planification d'une tâche : cron et crontab

L'utilitaire « crontab » gère un accès utilisateur avec "cron" (voir cron(1M)) par copie, création, liste et suppression du fichier (ou de la table) « crontab » de planification d'exécution de jobs. S'il est appelé sans option, « crontab » copie le fichier spécifié, sur la sortie standard et, si aucun fichier est spécifié, dans le répertoire qui stocke toutes les « crontabs » utilisateurs. Si « crontab » est appelé avec un nom de fichier, il écrase une entrée de « crontab » existante pour l'utilisateur qui l'a appelé.

Format d'entrée de la table "crontab" :
Un fichier "crontab" contient des lignes, chacune ayant six champs. Les champs sont séparés par des espaces ou des tabulations. Les 5 premiers sont, chacun, constitués de nombres entiers signifiant respectivement :

- Minute (0-59),
- Heure (0-23),
- Jour du mois (1-31),
- Mois de l'année (1-12),
- Jour de la semaine (0-6 with 0=Sunday).

← Dans ces champs, on peut avoir 1) une astérisque (signifiant toutes les valeurs), 2) un nombre entier ou 3) une suite de valeur séparés par des « , ».

➔Cron et crontab (suite)

Minute (0-59)

Heure (0-23)

Mois de l'année (1-12)

Commande à exécuter

⇒ Exemple de contenu de la table « crontab » :

```
[su0917@root:/] crontab -l
#ident  "@(#)root       1.21    04/03/23 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5/kprop_script ___slave_kdcs___
##45 23 * * * /opt/SUNWexplo/bin/explorer > /dev/null 2>&1
##################
# Nouveau PRAI
42 18 * * * /product/prai/shl/prai_create_send_config.sh >/varsoft/prai/logs/pra1 2>&1
55 18 * * * /product/prai/shl/prai_create_send_carto.sh >/varsoft/prai/logs/pra2 2>&1
00 01 19 * * /product/prai/shl/prai_create_send_image.sh >/varsoft/prai/logs/pra3 2>&1
# Scanner Config
##0 20 * * * /opt/sfrsi/scanner/config.sh >/dev/null 2>&1
##45 23 * * * /opt/sfrsi/scanner/lecturesar.sh >/dev/null 2>&1
##################
```

➔Planification d'une tâche : cron et crontab (suite)

Exemple 1: purge des fichiers "core" :

1) Cet exemple purge les fichiers "core" chaque matin, les jours de semaine à 3h15 du matin:

```
15 3 * * 1-5 find $HOME -name core 2>/dev/null | xargs rm -f
```

2) Exemple 2: Mail de voeux d'anniversaire :

```
0 12 14 2 * mailx john%Happy Birthday!%Time for lunch.
```

3) Exemple 3: Spécifier des jours du mois et de la semaine :

Cet exemple

```
0 0 1,15 * 1
```

exécutera une commande sur le 1er et le 15 de chaque mois, aussi bien que chaque lundi :

Pour spécifier des jours par un seul champs, les autres champs devront être positionnés à la valeur « * » (astérisque). par exemple:

```
0 0 * * 1
```

exécutera une commande seulement les lundi.

➔Planification d'une tâche : cron et crontab (suite)

⇒ Les tables « crontab » du super-utilisateur 'root' et des utilisateurs (en Solaris) se trouvent dans ce répertoire :

```
[root@su0349:/]
 ¤ ls -l /var/spool/cron/crontabs  <= répertoire
total 46
drwxr-xr-x   2 root     sys          512 Aug 16 14:16 .
drwxr-xr-x   4 root     sys          512 Mar  1  2006 ..
-rw-------   1 root     adm          248 Apr  8  2010 adm
-rw-------   1 root     other        424 Nov  2  2009 btsys
-rw-------   1 root     other        223 Dec 18  2006 digiadm
-r--------   1 root     root         452 Mar  1  2006 lp
-r--------   1 root     other          0 Feb 13  2007 lpetit
-rw-------   1 root     mysql        293 May 19  2009 mysql
-rw-------   1 root     root       13166 Aug 16 14:16 root <= crontab de root
-rw-------   1 root     sys          308 Mar  1  2006 sys
-r--------   1 root     sys          404 Mar  1  2006 uucp
-rw-------   1 root     root         208 Jun 24  2010 webservd
[root@su0349:/]
```

⇒ Où se trouve la log des traces d'exécution du cron :

/var/cron/log Contient des informations historiques sur l'exécution du cron.

➔Visualisation de la configuration d'un serveur

`uname -a`

Cette commande permet d'obtenir des informations sur l'architecture, la sous-architecture et le type de machine.

`prtconf -v`

Cette commande permet d'obtenir des informations sur l'architecture, la sous-architecture et le type de machine. Elle fournit toute la configuration matérielle de l'équipement. Voir aussi la commande « `prtdiag` ».

`sysdef -i`

Cette commande fournit toute la configuration matérielle de l'équipement, ainsi que la valeur de certaines valeurs positionnées dans le noyau (pour les IPC, par exemple).

Voir exemples pages suivantes.

➔Visualisation de la configuration matérielle : uname

Hostname (nom du serveur vu du réseau).	`[root@iu0263:/etc] hostname`
	`iu0263`
	`[root@iu0263:/etc] uname -n`
	`iu0263`
	`[root@iu0263:/etc]`

Marque	`[root@iu0263:/etc] uname`
	`SunOS`
	`[root@iu0263:/etc]`

Modèle	`[root@iu0263:/etc] uname -i  # en HP-UX : /bin/model`
	`SUNW,T5240`
	`[root@iu0263:/etc]`

Toutes informations sur le serveur	`[root@iu0263:/] uname -a`
	`SunOS iu0263 5.10 Generic_127127-11 sun4v sparc SUNW,T5240`
	`[root@iu0263:/]`

Version de Solaris

Niveau patch kernel

Architecture serveur

Formation Unix de base

➔Visualisation de la configuration d'un serveur : psrinfo, prtconf

valeur	SUN	HP
Nombre de CPU	[root@iu0263:/] **/usr/sbin/psrinfo -p** 1 [root@iu0263:/]	*HPUX11.11: /bin/echo "processor_count/D" | adb -k /stand/vmunix /dev/mem|tail -1* *HPUX11.23: /usr/contrib/bin/machinfo|grep 'Number of CPUs'*
Cadence CPU	[root@iu0263:/etc] **psrinfo -pv** The physical processor has 8 virtual processors (0-7) UltraSPARC-T2+ (cpuid 0 clock 1165 MHz) [root@iu0263:/etc]	*HPUX11.11: /bin/echo "runningprocs/D" | adb -k /stand/vmunix /dev/mem|tail -1* *HPUX11.23: /usr/contrib/bin/machinfo|grep 'processor family'*
Taille de la RAM	[root@iu0263:/etc] **/usr/sbin/prtconf | grep 'Memory size'** Memory size: 8192 Megabytes [root@iu0263:/etc]	*HPUX11.11:* *expr `echo "phys_mem_pages/D" | adb -k /stand/vmunix /dev/mem|tail -1|cut -f2` * 4 / 1024* *HPUX11.23: /usr/contrib/bin/machinfo|/usr/bin/grep "Memory"*

Formation Unix de base

➔Visualisation de la configuration matérielle : prtconf

```
¤ prtconf -v
System Configuration:  Sun Microsystems  sun4v
Memory size: 32640 Megabytes
System Peripherals (Software Nodes):

SUNW,SPARC-Enterprise-T5220
    System properties:
        name='fm-capable' type=int items=1
            value=00000009
        name='relative-addressing' type=int items=1
            value=00000001
        name='MMU_PAGEOFFSET' type=int items=1
            value=00001fff
        name='MMU_PAGESIZE' type=int items=1
            value=00002000
        name='PAGESIZE' type=int items=1
            value=00002000
    Driver properties:
        name='pm-hardware-state' type=string items=1 dev=none
            value='no-suspend-resume'
        name='fm-errcb-capable' type=boolean dev=none
        name='fm-ereport-capable' type=boolean dev=none
    scsi_vhci, instance #0
        System properties:
            name='class' type=string items=1
                value='root'
            name='load-balance' type=string items=1
                value='round-robin'
            name='auto-failback' type=string items=1
                value='enable'
            name='device-type-scsi-options-list' type=string items=8
                value='HP      HSV101' + 'symmetric-option' + 'HP      HSV2' + 'symmetric-option' + 'HP      OPEN' + 'symmetric-option' +
'EMC     SYMMETRIX'
[...]
[root@su1512:~]
 ¤
```

```
[  . . . ]
Hardware properties:
            name='initiator-interconnect-type' type=string items=1
                value='iSCSI'
            name='scsi-initiator-id' type=int items=1
                value=00000007
            name='scsi-selection-timeout' type=int items=1
                value=000000fa
            name='scsi-watchdog-tick' type=int items=1
                value=0000000a
            name='scsi-tag-age-limit' type=int items=1
                value=00000002
            name='scsi-reset-delay' type=int items=1
                value=00000bb8
            name='scsi-options' type=int items=1
                value=00107ff8
        Device Minor Nodes:
            dev=(98,0)
                dev_path=/iscsi:devctl
                    spectype=chr type=minor
    pseudo, instance #0
        System properties:
            name='instance' type=int items=1
                value=00000000
            name='class' type=string items=1
                value='root'
        Device Minor Nodes:
            dev=(2,0)
                dev_path=/pseudo:devctl
                    spectype=chr type=minor
[root@su1512:~]
 ¤
```

Formation Unix de base

➔Visualisation de la configuration matérielle : sysdef -i

```
 ¤ sysdef -i
*
* Hostid
*
  84d3fca2
*
* sun4v Configuration
*
*
* Devices
*
scsi_vhci, instance #0
        ssd, instance #1
        ssd, instance #2
        ssd, instance #3
        ssd, instance #4
        ssd, instance #5
        ssd, instance #6
        ssd, instance #7
        ssd, instance #8
        ssd, instance #9
        ssd, instance #10
        ssd, instance #11
        ssd, instance #12
        ssd, instance #13
        ssd, instance #14
packages (driver not attached)
        SUNW,builtin-drivers (driver not attached)
        deblocker (driver not attached)
        disk-label (driver not attached)
        terminal-emulator (driver not attached)
        dropins (driver not attached)
        SUNW,asr (driver not attached)
        kbd-translator (driver not attached)
        obp-tftp (driver not attached)
        zfs-file-system (driver not attached)
        hsfs-file-system (driver not attached)

[ . . . ]
* Time Sharing Scheduler Tunables
*
60      maximum time sharing user priority (TSMAXUPRI)
SYS     system class name (SYS_NAME)
[root@su1512:~]
```

➔Visualisation de la configuration matérielle : prtdiag

```
example% /usr/platform/sun4d/sbin/prtdiag
System Configuration: Sun Microsystems sun4d SPARCserver 1000
System clock frequency: 40 MHz
Memory size: 64Mb
Number of XDBuses: 1
CPU Units: Frequency Cache-Size Memory Units: Group Size
A: MHzMB B: MHzMB 0: MB 1: MB 2: MB 3: MB

________ _______ _________ ___ _____ _____ _____ _____
Board0: 50 1.0 50 1.0 32 0 0 0
Board1: 50 1.0 50 1.0 32 0 0 0
============================SBus Cards=========================
Board0: 0: dma/esp(scsi) 'SUNW,500-2015'
lebuffer/le(network) 'SUNW,500-2015'
1: <empty>
2: cgsix 'SUNW,501-1672'
3: <empty>
Board1: 0: dma/esp(scsi) 'SUNW,500-2015'
lebuffer/le(network) 'SUNW,500-2015'
1: <empty>
2: <empty>
3: <empty>
No failures found in System
===============================================================
```

Formation Unix de base

Structure des réseaux (chez SFR)

- Un **réseau informatique** est un ensemble d'***équipements*** reliés entre eux pour échanger des informations.
- Chez SFR, les réseaux sont un ensemble liens (circuits) et équipements physiques permettant de relier les ordinateurs (les serveurs entre eux) et de faire transiter des informations entre eux.

Il existe plusieurs types de réseaux chez SFR :

- **Réseau interne** (LAN Intranet), en général, accessible par la plupart des utilisateurs SFR.

=> En production, il y a 3 types de réseau Intranet, chez SFR :

- CLI : réseau utilisé par les utilisateurs (clients des machines)
- ADM : réseau d'administration utilisé par les administrateurs systèmes et A.E.
- SVG : réseau utilisé pour la sauvegarde des serveurs.

- **Réseau situé en DMZ**, réseau protégés par des pare-feux, d'accès en général restreint (°).

(°) En informatique, une **zone démilitarisée** (ou **DMZ**, de l'anglais *demilitarized zone*) est un sous-réseau isolé par un pare-feu. Ce sous-réseau contient des machines se situant entre un réseau interne (LAN - postes clients) et un réseau externe (typiquement, Internet).

Formation Unix de base

➔Structure des serveurs (chez SFR)

Pattes réseaux :
Les serveurs ont besoin de relier à différents réseaux informatiques de SFR, par le biais de prises réseau RJ45, qu'on appellent « pattes réseau ».

Les serveurs ont, en général, 3 pattes réseaux, en prod, ou 2, en recette ou pack :

•Patte CLI : patte permettant de se connecter réseau utilisé par les utilisateurs (clients des machines) (en prod, recette et pack).
•Patte ADM : patte permettant de se connecter au réseau d'administration utilisé par les administrateurs systèmes et A.E (en pack).
•Patte SVG : patte permettant de se connecter (d'être relié) au réseau utilisé pour la sauvegarde des serveurs (en prod, en recette).
•Parfois, il existe des serveurs ayant plus de 3 pattes réseaux, comportant alors des pattes additionnelles CLU, NFS, NAS, TEC …, pour se connecter à des réseaux particuliers (spéciaux) nommés CLU, NFS, NAS, TEC … ou encore BBA, SRV etc. etc.

Formation Unix de base

➔Structure des serveurs (chez SFR) (suite)

Pattes réseaux (suite) :
Pour des raisons de sécurité, certaines serveurs ont des pattes doublées ou multipliées, connectées sur le même réseau (ADM, CLI, SVG, …). Si une patte physique tombe en panne, une autre patte physique prend le relais pour permettre au serveur de continuer de communiquer sur le réseau spécifique (ADM, CLI, SVG, …) :

• En Solaris, la technologie de distribution de charge entre patte réseau, est appelée IPMP (°).
• En HP, elle s'appelle VLAN Tagging (°°).

(°) **IPMP**, ou **Réseau d'IP multipathing**, est un service fourni par Solaris pour fournir la défaut-tolérance et la distribution de charge pour des cartes d'interface de réseau (ou NICs).
(°°) Un **VLAN** est une façon de créer des réseaux logiques indépendants au sein d'un réseau physique. Un VLAN Tagging est la pratique consistant à insérer un ID de VLAN dans un en-tête de paquet afin de déterminer les VLAN (Virtual Local Area Network) auquel le paquet appartient.

Formation Unix de base

➔résolution DNS

Le **Domain Name System** (ou **DNS**, système de noms de domaine) est un service permettant d'établir une correspondance entre une adresse IP et un nom de domaine (qui correspond souvent au nom du serveur) et, plus généralement, de trouver une information _ adresse IP, serveur DNS … _ à partir d'un nom de domaine.

Pour vérifier l'association entre un nom et une adresse IP, plusieurs commandes sont disponibles suivant les systèmes d'exploitation utilisés. Pour exemple sur un Unix, la commande nslookup est disponible, via l'invite de commande Unix :

```
[root@su0349:/]
 ¤ nslookup su1216
Server:         10.16.112.8
Address:        10.16.112.8#53

su1216.sfr.com  canonical name =
su1216adm.sfr.com.
Name:   su1216adm.sfr.com
Address: 10.43.196.7
[root@su0349:/]
```

```
[root@su0349:/]
 ¤ nslookup 10.43.196.7
Server:         10.16.112.8
Address:        10.16.112.8#53

7.196.43.10.in-addr.arpa  name =
su1216adm.sfr.com.

[root@su0349:/]
 ¤
```

➔résolution NTP (serveur de temps) : ntpq

NTP : Le **Protocole d'Heure Réseau** (*Network Time Protocol* ou NTP) est un protocole qui permet de synchroniser, via un réseau informatique, l'horloge locale d'ordinateurs, sur une référence d'heure (en général, fourni par un autre serveur).

Vérifier par la commande '*ntpq*' qu'on accède bien à distance au serveur de temps :

```
[root@su0349:/] ntpq -p
     remote           refid      st t when poll reach   delay   offset    disp
==============================================================================
 LOCAL(0)        LOCAL(0)        10 l   55   64  377     0.00    0.000   10.01
 su0350svg       su0351adm.sfr.c  3 u   76 1024  166     5.40    4.560   17.82
+su0351svg       ntp_refserver1.  2 u  103  128  376     0.32    0.967    0.81
*ntp_refserver1. .GPS.            1 u  702 1024  377     9.38    0.838    1.11
[root@su0349:/] ntptrace
localhost: stratum 2, offset 0.000073, synch distance 0.01689
ntp_refserver1.sfr.com: stratum 1, offset 0.000809, synch distance 0.00000, refid 'GPS'
[root@su0349:/]
```

Note : le serveur précédé d'un '*' est actuellement celui utilisé.

➔résolution NTP (serveur de temps) : ntpq (suite)

```
[su1190@root:/] ntpq -p
     remote           refid      st t when poll reach   delay   offset    disp
==============================================================================
+su0349adm.sfr.c ntp_refserver1.  2 u  690 1024  377    15.87    0.486    1.63
*su0350adm.sfr.c ntp_refserver1.  2 u  335 1024  377    15.76   -0.415    0.87
+su0351adm.sfr.c ntp_refserver1.  2 u  413 1024  377    15.18    1.918    1.16
[su1190@root:/]
¤ ntpq -c peers
     remote           refid      st t when poll reach   delay   offset    disp
==============================================================================
*su0349adm.sfr.c ntp_refserver1.  2 u  236 1024  377    14.97    2.686    2.79
+su0350adm.sfr.c su0351adm.sfr.c  3 u  126 1024  377    15.50   -0.758    1.42
+su0351adm.sfr.c ntp_refserver1.  2 u  369 1024  377    15.37    1.869    1.82
[root@su1217:/etc]
 ¤
```

Quand cela ne marche pas ou que cela met un certain temps à se synchroniser :

```
# ntptrace
localhost: stratum 3, offset -0.016370, synch distance 0.517773 91.189.94.4: timed out,
nothing received ***Request timed out
# ntptrace
localhost: stratum 3, offset 0.021077, synch distance 0.620848
horlogegps.reseau.jussieu.fr: stratum 2, offset -0.000020, synch distance 0.100084
10.3.128.189: timed out, nothing received ***Request timed out
```

⇒*Il faut patienter quelques minutes, pour être sûr que la synchro ne fonctionne toujours pas.*

⇒Source : *http://www.system-linux.eu/index.php?post/2010/01/05/Mettre-vos-serveurs-%C3%A0-la-bonne-heure-avec-NTP*

Formation Unix de base

➔Commande de configuration des paramètres de l'interface réseau : ifconfig

➔Pour vérifier que les pattes réseau fonctionnent, il faut qu'elles soient **UP** et **RUNNING**, comme dans l'exemple ci-après.
➔On utilise pour cela la commande « **`ifconfig -a`** » (°) :

```
 [su0238@root:/] ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.172.209.134 netmask fffff000 broadcast 10.172.223.255  # <= patte ADM
        ether 0:14:4f:95:d3:8c
bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 10.172.66.161 netmask fffffe00 broadcast 10.172.67.255    # <= patte CLI
        ether 0:14:4f:95:d3:8d
[su0238@root:/]
```

Note : ici ce serveur su0238 est à Rennes (en pack, car réseau en 10.172.xxx.xxx) et possède 2 pattes réseau ADM et CLI.
Note2 : l'équivalent de "ifconfig –a" (SUN) pour HP-UX est : « `lanshow` ».
(°) C'est un peu l'équivalent de la commande « `ipconfig` » en MS-DOS.

Formation Unix de base

➔Outils MCO serveurs (SFR)

Page d'accès : http://phenope2/info_machine/secure/Accueil_MCO.php

Formation Unix de base

➔Outils Sysload (SP Analyst)

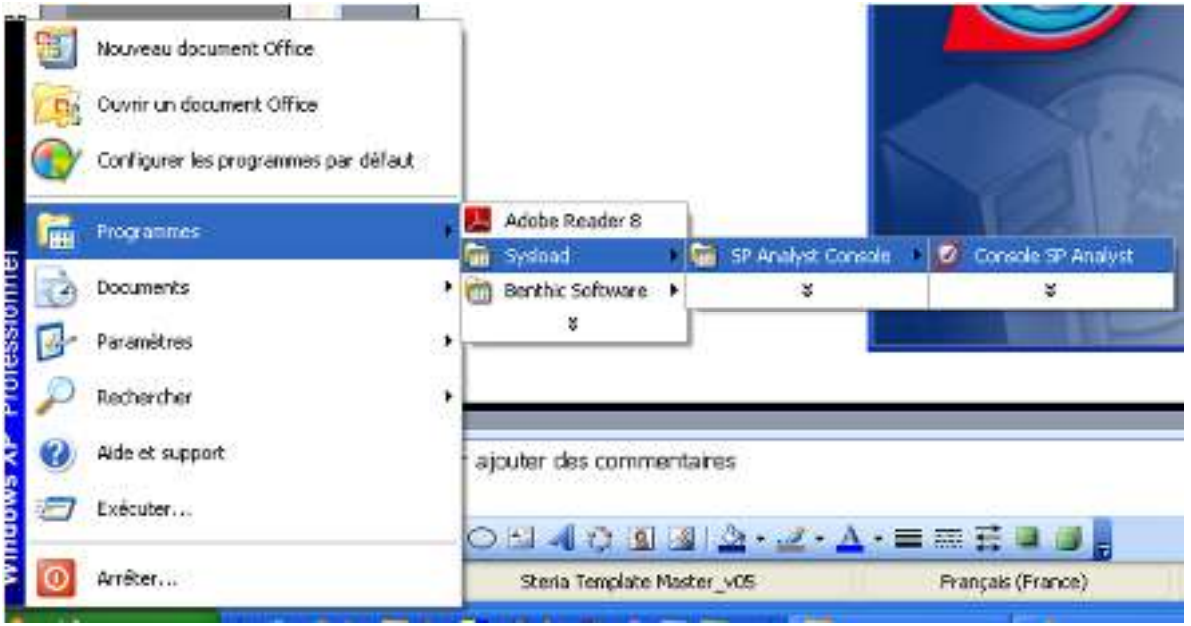
SP Analyst (de la société Sysload) est un outil de diagnostic de performance pour les serveurs physiques et virtuels. SP Analyst est un outil de productivité utilisé par les équipes informatiques pour diagnostiquer des problèmes de performance et de gestion de la capacité des ressources du serveur.

•Pour lancer votre client Sysload SP Analyst à partir de votre poste PC ↓ :

Note : bien sûr, il faut que le client Sysload SP Analyst ait été déjà installé sur votre poste PC, par l'équipe Bureautique (pour pouvoir le faire marcher sur votre PC).

Sinon, faire un demande dans ce sens, sur le site :

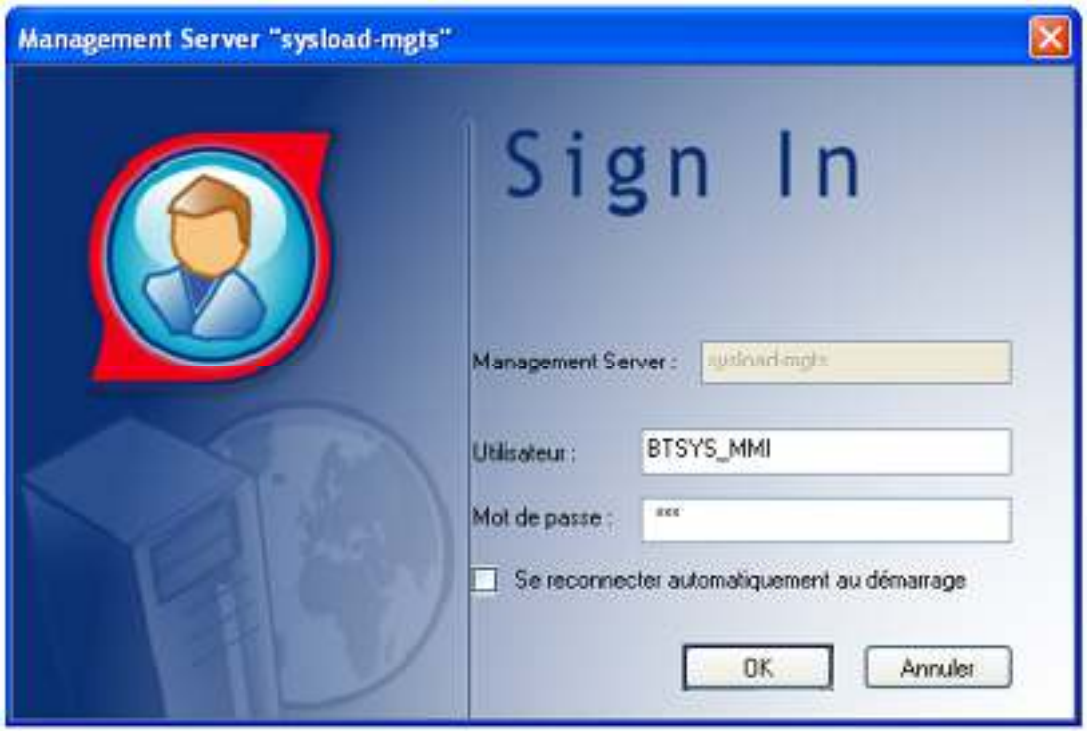
http://e-dsi



Formation Unix de base

➔Outils Sysload (SP Analyst)

Ecran de saisie du Login et passe de Sysload → :



SYSLOAD (client) : BTSYS_MMI / mmi

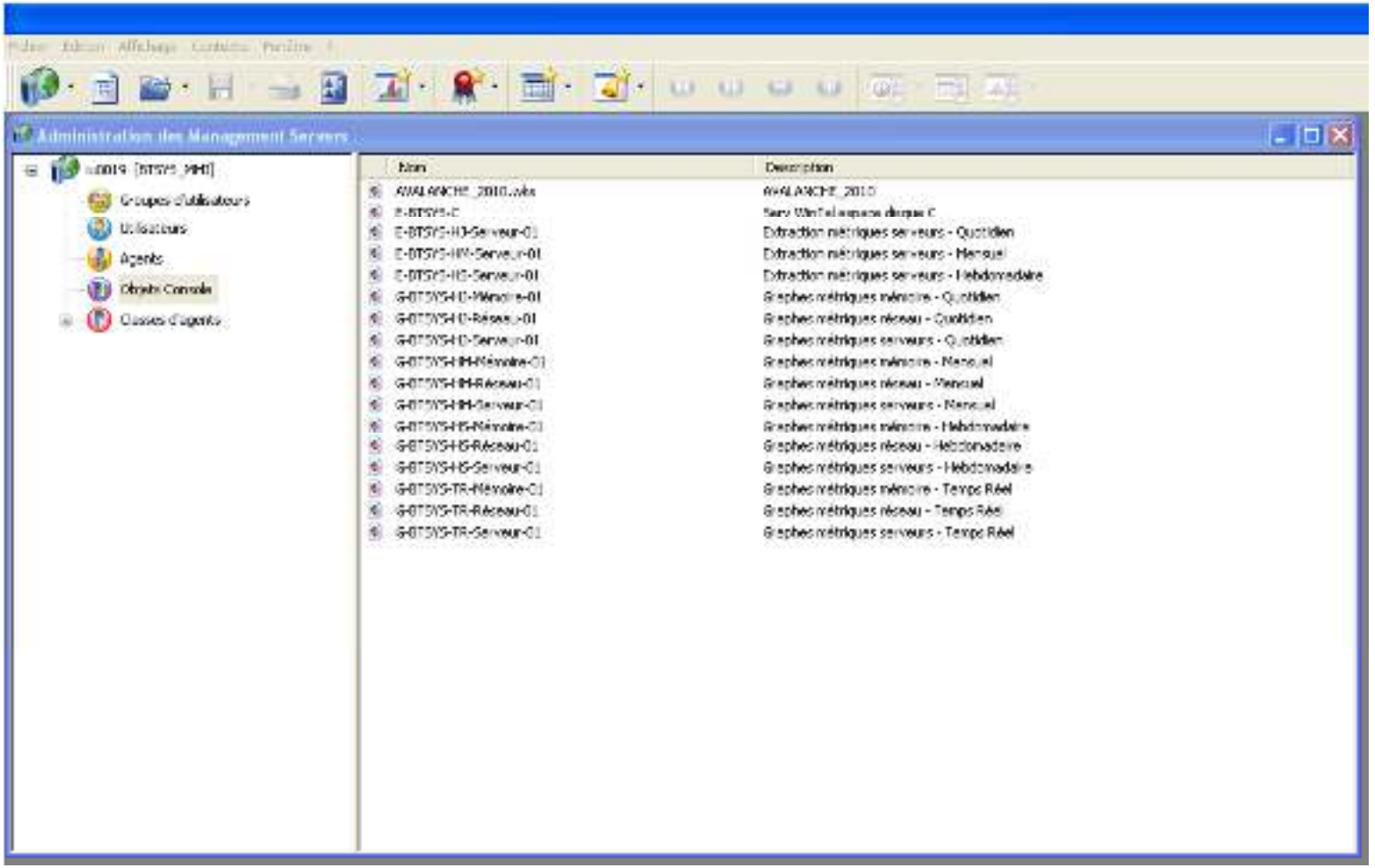
Formation Unix de base

➔Outils Sysload

• Ce qu'il s'affiche après que l'on ait saisi le login et passe →

• On fait le choix par exemple de « Graphes métriques serveurs – Quotidien »

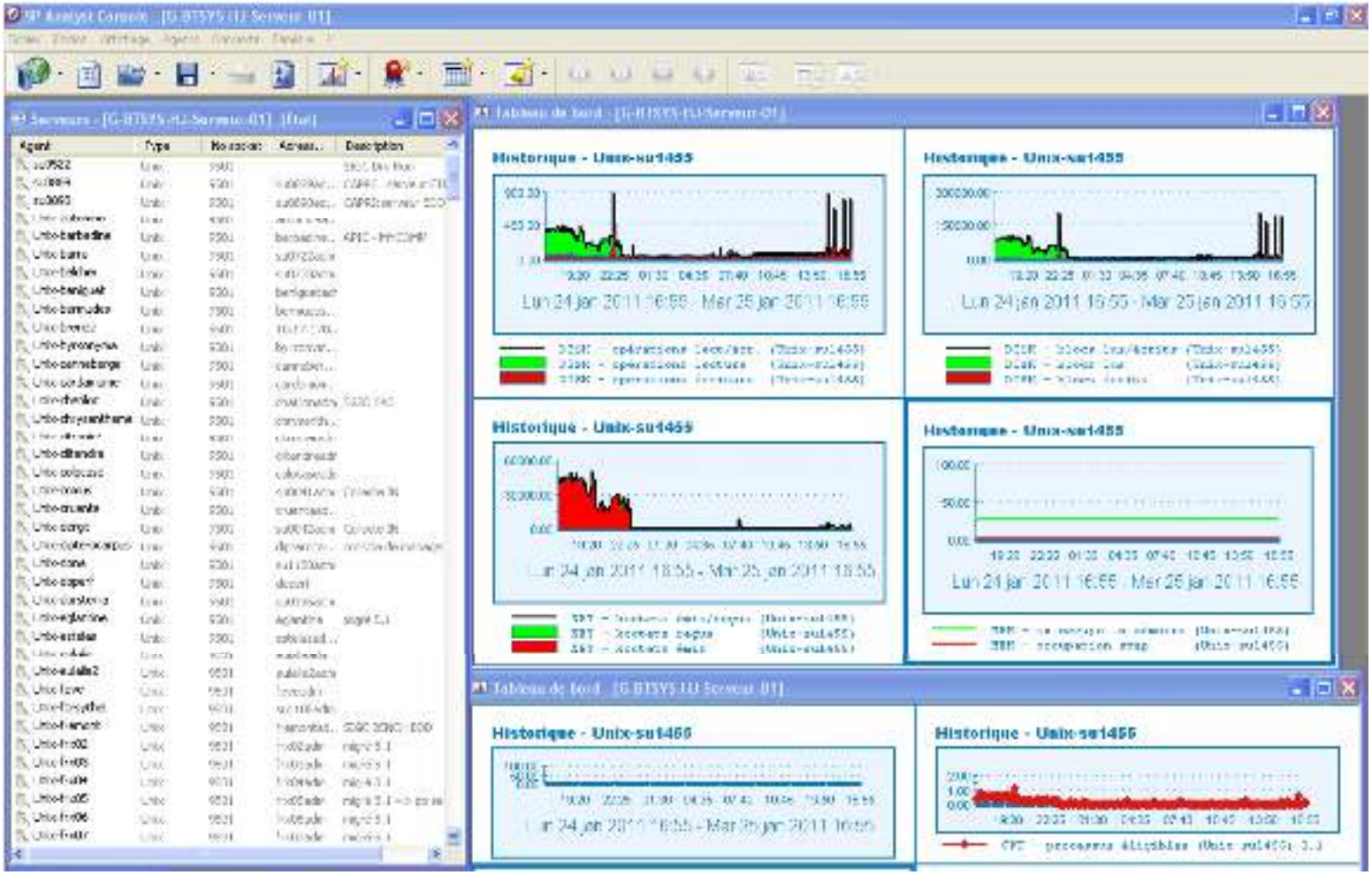
•Voir page suivante.



Sysload User guide : référence à compléter (?).

Formation Unix de base

➔Outils Sysload



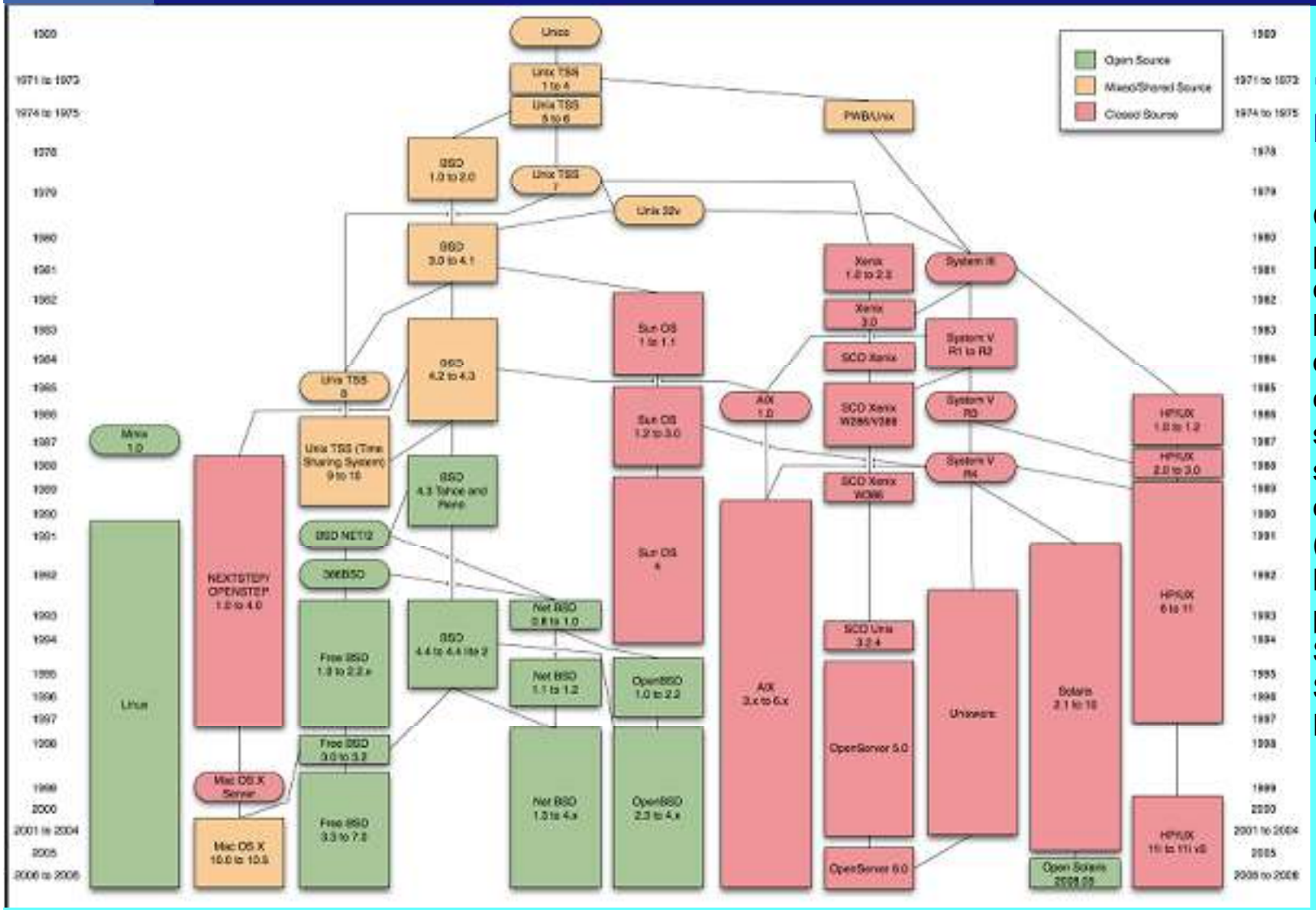
Attention !! : Sysload devrait être abandonné au profit de l'outil de métrologie zabbix (un Open Source gratuit).

Formation Unix de base

Annexe: Historique

- 1969 : Ken Thompson et Dennis Ritchie créent un nouveau système inspiré de MULTICS (Multiplexed Information and Computing System).
- Version 4 écrite en langage C.
- FAMILLE UNIX (nom déposé par ATT), environnement de temps partagé multiutilisateurs et multiprocessus.
- Famille BSD 4.3 (universitaire)
- Famille UNIX SYSTEME V (industriel)
- Dans la famille SYSTEM V (Unix propriétaires) :
  - Solaris (SUN)
  - HP-UX (HP)
  - AIX (IBM)
- Normalisation internationale menée par l'OSF (Open Software Foundation), regroupant les principaux constructeurs (IBM, DEC, Bull, HP ...).
- Apparition d'un Unix gratuit : Linux (freeware)
- Nombreuses améliorations du système initial, notamment avec l'ajout de composantes :
- Réseau : UUCP, puis TCP/IP (notamment le protocole NFS permettant de partager des ressources entre machines)
- Interface : X11 (interface graphique), puis OSF/MOTIF
- Langages de commandes « shells » (Bourne-shell, Korn-shell, C-shell).

Formation Unix de base



➔Annexe: Historique

Comme nous pouvons le constater l'histoire d'Unix est très complexe et a ses gourous et ses chapelles concurrentes (Unix libre _ BSD, Linux _, propriétaire _ Système V, Solaris, AIX, HPUX etc.).

Annexe : write

Envoi d'un message directement à un autre utilisateur connecté.

```
noemed% write poulique
Est-ce qu'on peut utiliser l'imprimante ?
<Control-D>
```

<Control-D> représente pour UNIX le caractère de fin de fichier

L'utilisateur à qui est destiné ce message le verra apparaître sur son écran...

En retour, il peut répondre :

```
message from poulique@noemed on ttyp0 at 15:55 ... Non, elle n'est
pas encore configurée...
```

Annexe : mail

Messagerie électronique permettant la communication par boîte aux lettres entre utilisateurs.

Envoi de mail :

syntaxe : **mailx *utilisateur***
Subject: ***Sujet du message***
corps du message
<un point en début de ligne>

Exemple :

```
noemed% mailx dupond
Subject: Impression
La configuration de l'imprimante est finie
vous pourrez l'utiliser demain
.
```

➔Annexe : mail (suite)

Lecture de sa boîte aux lettre :

`mailx` (sans paramètre)

Si aucun message n'est présent dans la boîte aux lettres il affiche :

```
No mail for dupond
```

Sinon il affiche la liste des messages (uniquement les sujets), chaque message ayant un numéro il suffit de taper ce numéro pour lire son contenu...

```
noemed% mailx
Mail version SMI 4.0 Thu Jul 23 13:52:20 PDT 1992   Type ? for help.
"/usr/spool/mail/dupond": 2 messages 2 new
>N  1 administ@noemed.univ-rennes1.fr Wed Feb 18 16:14 10/381 Commande ls
 N  2 poulique@noemed.univ-rennes1.fr Wed Feb 18 16:15 10/355 Impression
? _
```

Annexe : mail (suite)

Pour lire le premier message :

```
& 1
Message 1:
From administ Wed Feb 18 16:14:36 1997
Received: by noemed.univ-rennes1.fr; Wed, 18 Feb 97
16:14:35 GMT (4.1/VERsept92)
Date: Wed, 18 Feb 97 16:14:35 GMT
From: administ@noemed.univ-rennes1.fr (Administrateur
des Macintosh )
To: dupond@noemed.univ-rennes1.fr
Subject: Commande ls
Status: R
Est-ce que tu connais l'option permettant de lister toutes les
informations dans la commande ls ?
```

➔Annexe: mail (suite)

Pour répondre directement à un message : r

```
& r
To: administ@noemed.univ-rennes1.fr
Subject: Re:   Commande ls
Non, je n'en sais rien du tout !
.
```

Annexe : mail (suite)

Pour supprimer le message courant : d

Pour quitter le programme mailx : q

Pour quitter sans toucher aux messages : x

ATTENTION: selon les unix ce programme s'appelle mail ou mailx

Formation Unix de base

➔Annexe : Commandes à proscrire (!) :

A) Commandes en rouge pour supprimer les fichiers inutiles à éviter :

```
find . \( -name core -o -name toto \) -exec rm -f {} \ ;
```

Ou encore :

```
rm -f *
```

Les commandes de suppression de fichiers sont brutales sous Unix, d'autant qu'elle ne vous demande aucune confirmation, avant, avant de supprimer le fichier. Dès que le fichier est détruit, on ne peut plus le restaurer, car il n'existe pas de commande Unix type « **recover** » en Unix.
Ici chez SFR, on ne peut récupérer (par 'recover') les fichiers supprimés par erreur qu'avec le logiciel '**networker**'. Et encore, il faut que le fichier ait été sauvé par networker, avant (La sauvegarde **networker** se fait, en général, chaque nuit).

➔Annexe: Recherche d'une chaîne : grep (commande « filtre »)

Motifs et expressions régulières (pour le grep) :

•Le programme ou utilitaire `grep` utilise le langage « **grep** ».

•Le *motif* ou *modèle* (en anglais « *pattern* ») est le *critère de recherche*, traduit dans ce *langage*.

•Le `grep` procède à une recherche/sélection/remplacement où, au lieu de chercher une *chaîne de caractères* donnée, il cherche une *chaîne* dans le document qui correspond à un *motif* donné (i.e. un *critère*), selon des *règles précises*, en partant du début du document.

•Une *expression régulière* est un *motif* (critère) permettant de décrire un ensemble de *chaînes*. Elles sont construites comme des opérations arithmétiques, en utilisant des *opérateurs* divers pour combiner des expressions plus petites.

Exemples de caractères utilisées par les expressions régulières :

•Caractères « **[«** et « **] »** : Une *liste de caractères*, encadrée par « **[«** et « **] »** peut être mise en correspondance avec n'importe quel *caractère unique* appartenant à la liste. Si le premier caractère de la liste est l'accent circonflexe « **^ »** alors la mise en correspondance se fait avec n'importe quel caractère *absent* de cette liste.

•Par exemple, l'expression régulière **[0123456789]** convient pour n'importe quel chiffre unique. L'expression **[0-9]** est équivalente à l'expression **[0123456789]** .

➔Annexe: Recherche d'une chaîne : grep (« filtre »)

(suite)

Motifs et expressions régulières (pour le grep) (suite) :

• Les *symboles* **\<** et **\>** correspondent respectivement à une chaîne vide en début et en fin de mot.
• Le *symbole* **\b** correspond à une chaîne vide à l'extrémité d'un mot,
• **\B** correspond à une *chaîne* vide ne se trouvant *pas* à une extrémité de mot.
• Une *expression régulière*, correspondant à un *caractère unique*, peut être suivie par l'un des *opérateurs de répétition* suivants :

? : L'élément précédent est facultatif et doit être mis en correspondance *une fois au maximum* (pour que le *critère* ou le *motif* soit vérifié).
* : L'élément précédent doit être mis en correspondance *zéro ou plusieurs fois*.
+ : L'élément précédent doit être mis en correspondance *au moins une fois*.
{n} : L'élément précédent doit être mis en correspondance *exactement n fois*.
{n,} : L'élément précédent doit être mis en correspondance *n fois ou plus*.
{,m} : L'élément précédent est *facultatif* et doit être mis en correspondance *m fois au plus*.
{n,m} : L'élément précédent doit être mis en correspondance *au moins n fois, mais au plus m fois.*

➔ Annexe: Commande 'awk' (suite) (Filtre)

Imprime chaque ligne après effacement du 2nd champ :

```
awk '{$2 = ""; print}' file
```

Imprime hi 28 fois :

```
yes | head -28 | awk '{ print "hi" }'
```

Note : la commande « **yes** » affiche d'une façon répétitive une chaîne spécifiée. Par exemple, « **yes** » (sans argument) affichera par défaut « y y y y y … » (selon une colonne verticale).
« **yes toto** » affichera répétitivement « toto toto toto toto … » (les « toto » disposés sur une colonne verticale).

Imprime hi.0010 à hi.0099 :

```
yes | head -90 | awk '{printf("hi00%2.0f \n", NR+9)}'
```

Remplace chaque champs par sa valeur absolue :

```
{ for (i = 1; i <= NF; i=i+1) if ($i < 0) $i = -$i print}
```

➔ Annexe : Commande ‘awk’ (suite)

(Filtre)

Actions (suite)

exemple:

```
$cat total.awk
BEGIN    { total = 0; dix = 0 }
         { total += $1 }
$1 == 10 { dix++ }
END      { printf "Total: %d, Dix: %d\n",\
           total, dix }
$awk –f total.awk nombres.txt
Total: 98, Dix: 2
$
```

➔ Annexe : commande ‘awk’ (suite) (Filtre)

Actions (suite)

exemple:

```
$cat pgvp.awk
BEGIN    { pgv = -1 }
$1 > pgv { pgv = $1 }
END      { if (pgv == -1)
           {
             printf "Valeurs negatives\n"
           }
           else
           {
             printf "Plus grande valeur: %d\n",\
             pgv
           }
         }
$awk –f pgvp.awk nombres.txt
Plus grande valeur: 23
$
```

➔Annexe: commande awk (suite) (Filtre)

Commentaires

Tout ce qui suit le symbole # sur une ligne est considéré comme un commentaire

exemple:

```
$cat commentaires.awk
BEGIN     { pgv = -1 # initialisation
          }
$1 > pgv  { # si 1er champ > valeur sauvegardee,
            # 1er champ devient valeur a battre
            pgv = $1 }
END       { printf "Plus grande valeur: %d\n",\
            pgv }
$awk -f pgvp.awk nombres.txt
Plus grande valeur: 23
$
```

Awk (suite) (annexe) Variables

(Filtre)

NR	numéro de l'enregistrement courrant
$0	enregistrement courrant
NF	nombre de champs dans l'enregistrement courrant
$1 à $n	les champs
FS	délimiteur de champs en entrée *par défaut*: SPACE *et* TAB
OFS	délimiteur de champs en sortie *par défaut*: SPACE
RS	délimiteur d'enregistrement en entrée *par défaut*: NEWLINE
ORS	délimiteur d'enregistrement en sortie *par défaut*: NEWLINE
FILENAME	nom du fichier en traitement

(Filtre)

Annexe : commande awk (suite)

Fonctions

length(str)

*retourne la **longueur** de la chaîne de caractères **str***
*si **str** n'est pas spécifié, retourne la **longueur** de **l'enregistrement courrant***

int(num)

*retourne la **portion entière** de **num***

index(str1,str2)

*retourne la **position** de **str2** dans **str1** ou **0 si str1 ne contient pas** la chaîne **str2***

(Filtre)

Annexe : commande awk (suite)

Fonctions (suite)

split(str,arr,del)

*construit **un tableau de jetons** (**arr**) où chacun de ces derniers est un champ de la chaîne **str** découpé à l'aide du délimiteur **del***
*retourne la **taille** du tableau créé*

sprintf(fmt,args)

*retourne la **chaîne args formatée** d'après le format **fmt** (fonctionne comme la fonction C du même nom)*

substr(str,pos,len)

*retourne une **chaîne de caractères** tirées de **str** à partir de la position **pos** et de longueur **len***

(Filtre)

Annexe : commande awk (suite)

Opérateurs arithmétiques

Tous les opérateurs suivants sont supportés:

*, /, %, +, -, =, ++, --, +=, -=, *=, /= et %=.

(Filtre)

Annexe : commande awk (suite)

Tableaux associatifs

Un tableau associatif utilise des chaînes de caractères (appelées **clés**) comme indices:

array[string] = value

Une construction particulière de boucle facilite le parcours des tableaux associatifs:

for (elem in array) action

Note: il est également possible de simuler un tableau traditionnel en utilisant des chaînes de caractères numériques...

(Filtre)

Annexe : commande awk (suite)

La fonction printf

Fonctionnement similaire à celui du langage C:

printf "control-string" arg_1, arg_2, ..., arg_n

La chaîne de formatage (control-string) des spécifications de conversion ayant la forme suivante:

%[-][x[.y]]conv

- **-** justification à gauche
- **x** largeur minimale du champ affiché
- **.y** nombre de décimales, s'il y a lieu
- **conv** type du paramètre

Annexe : commande awk (suite) (Filtre)

La fonction printf (suite)

Les types de paramètres supportés (**conv**) sont:

d	nombre entier
e	notation exponentielle
f	nombre à point flottant
g	utiliser le plus court entre **e** ou **f**
o	octal
s	chaîne de caractères
x	hexadécimal

dans la chaîne de formatage:
utiliser **\n** pour un changement de ligne (NEWLINE)
utiliser **\t** pour introduire une tabulation (TAB)

➔Annexe : Commande ‘nawk’ (Filtre)

Notes : nawk est le awk de Solaris avec plus de fonctionnalités. Elle permet de traiter des champs.
Le ‘awk’ est un langage interprété permettant de traiter des chaînes de caractères. Il est 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 .

➔Annexe : Commande 'nawk'(suite)

(Filtre)

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).

Formation Unix de base

➔Annexe : résolution DNS : dig

```
¤ dig su1216 aaa
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 710
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;su1216.                                IN      A

;; AUTHORITY SECTION:
.                       600     IN      SOA     su0052cli.phys.prod. gregoire\.vialaret.sfr.com. 14281 900 3600
604800 600

;; Query time: 5 msec
;; SERVER: 10.16.112.8#53(10.16.112.8)
;; WHEN: Mon Jan 24 17:29:36 2011
;; MSG SIZE  rcvd: 103

; <<>> DiG 9.3.4-P1 <<>> su1216 aaa
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 1354
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;aaa.                           IN      A

;; AUTHORITY SECTION:
.                       600     IN      SOA     su0052cli.phys.prod. gregoire\.vialaret.sfr.com. 14281 900 3600
604800 600

;; Query time: 3 msec
;; SERVER: 10.16.112.8#53(10.16.112.8)
;; WHEN: Mon Jan 24 17:29:36 2011
;; MSG SIZE  rcvd: 100
```

Formation Unix de base

➔résolution DNS : dig (suite)

```
[root@iu0290:/etc] dig -x 10.172.223.254

; <<>> DiG 9.3.5-P1 <<>> -x 10.172.223.254
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 1753
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;254.223.172.10.in-addr.arpa.   IN      PTR

;; AUTHORITY SECTION:
172.10.in-addr.arpa.    600     IN      SOA     su0051cli.phys.pack.
gregory\.stepanovsky.cegetelsi.fr. 10492                                900 3600 604800 600

;; Query time: 34 msec
;; SERVER: 10.16.108.136#53(10.16.108.136)
;; WHEN: Tue Jan  5 16:26:28 2010
;; MSG SIZE  rcvd: 132

[root@iu0290:/etc] nslookup 10.172.223.254
Server:         10.16.108.136
Address:        10.16.108.136#53

** server can't find 254.223.172.10.in-addr.arpa.: NXDOMAIN

[root@iu0290:/etc]
```

Formation Unix de base

➔Annexe : sauvegarde et restauration système

• **Sauvegarde système en SUN** :
« **`ufsdump`** » _ pour les FS système : / , /var, voire des FS /usr et /opt s'ils existent.
Exemple :

```
ufsdump 0uf - / | compress > root.dump.Z
ufsdump 0uf - /var | compress > root.dump.Z
```

ou bien

```
ufsdump 0uf - / |gzip > root.ufsdump.gz
ufsdump 0uf - /var |gzip > var.ufsdump.gz
```

• **Restauration des sauvegardes système réalisées par « `ufsdump` » (SUN)** :

```
cd /tmp/disk3 ; ufsrestore rf /tmp/disk5/var.dump
uncompress < root.dump.Z | ufsrestore rf -
```

ou bien

```
gunzip < root.dump.Z | ufsrestore rf -
```

mount -F ufs -o rw /dev/dsk/c1t2d1s0 /mntufsdump 0f - / | (cd /mnt ;ufsrestore xvf -)

Formation Unix de base

➔Annexe : sauvegarde et restauration système (suite)

Exemple : →

```
[luzerne-HR](root)# cd /net/su0349svg2
[luzerne-HR](root)# cd catalog/phenope/dumps
[luzerne-HR](root)# ls -ld luzerne su1104
luzerne: No such file or directory
su1104: No such file or directory
[luzerne-HR](root)# cd luzerne
[luzerne-HR](root)# pwd
/net/su0349svg2/catalog/phenope/dumps/luzerne
[luzerne-HR](root)# ufsdump 0uf - / | compress > root.dump.Z
  DUMP: Writing 32 Kilobyte records
  DUMP: Date of this level 0 dump: Wed Apr 08 11:27:02 2009
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/vx/rdsk/rootvol (luzerne:/) to standard output.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 5666622 blocks (2766.91MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]

  DUMP: 63.39% done, finished in 0:05
  DUMP: 5668990 blocks (2768.06MB) on 1 volume at 3080 KB/sec
  DUMP: DUMP IS DONE      # <= indication du bon déroulement de la sauvegarde
  DUMP: Level 0 dump on Wed Apr 08 11:27:02 2009
[luzerne-HR](root)#
[luzerne-HR](root)# ufsdump 0uf - /var | compress > var.dump.Z
  DUMP: Writing 32 Kilobyte records
  DUMP: Date of this level 0 dump: Wed Apr 08 12:03:13 2009
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/vx/rdsk/var (luzerne:/var) to standard output.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 9303060 blocks (4542.51MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 38.52% done, finished in 0:15
  DUMP: 9302910 blocks (4542.44MB) on 1 volume at 4166 KB/sec
  DUMP: DUMP IS DONE            # <= indication du bon déroulement de la sauvegarde
  DUMP: Level 0 dump on Wed Apr 08 12:03:13 2009
[luzerne-HR](root)#
```

➔Annexe : Principales commandes Unix

Commande	Explication
cal	Calendrier
cat	affichage d'un fichier
cd	changement de répertoire
chmod	changement des droits sur un fichier
cp	copie de fichier
date	affiche la date courante
df	Espace disponible sur les partitions (file system) ("df -k ." : connaître l'espace encore disponible)
diff	affichage de différences entre deux fichiers ("diff fichier1 fichier2")
du	Espace utilisé (y compris dans les sous-répertoires)
egrep	recherche de motif dans un fichier
emacs	éditeur pleine page
file	affiche le type de chaque fichier
find	recherche de fichiers (find . -name toto.txt)
finger	information sur les utilisateurs connectés
ftp	transfert de fichiers entre machines

Annexe : Principales commandes Unix (suite)

Commande	Explication
grep	recherche d'une chaîne de caractères dans un fichier
gzip	compression et décompression de fichiers
head	affichage des premières lignes d'un fichier
history	rappel des dernières commandes
ln	créer un lien d'un fichier à l'autre
logout	déconnexion
ls	contenu d'un répertoire
Mailx (ou **mail**)	lecture ou envoi de courrier électronique
man	manuel en ligne
mkdir	création d'un répertoire
more	affichage page par page d'un fichier
mv	déplacement, renommer un fichier
passwd	changement de mot de passe

Annexe : Principales commandes Unix (suite)

Commande	Explication
pwd	affichage du répertoire courant
ps	liste des processus
quota	espace disque disponible
rm	destruction d'un fichier
rmdir	destruction d'un répertoire
sed	recherche et remplacement
tail	affichage des dernières lignes d'un fichier
talk	discussion en direct entre utilisateurs
tar	gestion d'archives portables
telnet	connexion à distance
wc	comptage du nombre de lignes, de mots ou de caractères d'un fichier
who	liste des utilisateurs connectés
whoami	qui suis-je ?
write	envoi d'un message à un utilisateur

Formation Unix de base

➔Annexe: commandes utiles à l'écriture de scripts shell

commande	description synthétique
basename	Éliminer le chemin d'accès et le suffixe d'un nom de fichier.
chroot	
date	Affiche ou configure la date et l'heure du système.
dd	Convertit et copie un fichier.
df	Indique l'espace occupé par les systèmes de fichiers.
dirname	Ne conserve que la partie répertoire d'un nom de fichier.
du	Évaluer l'espace disque occupé par des fichiers.
echo	Affiche une ligne de texte.
env	Exécute un programme dans un environnement modifié ou affiche les variables de l'environnement utilisateur.
factor	Affiche les facteurs premiers d'un nombre.
groups	Affiche les groupes auxquels appartient un utilisateur.
hostid	Affiche le numéro d'identification de l'hôte actuel.
hostname	Nom de l'hôte (du serveur)
id whoami	Affiche les UID et GID effectifs et réels.
install	Copie des fichiers et positionne leurs attributs.
logname	Afficher le nom de connexion de l'utilisateur

Formation Unix de base

➔Annexe: commandes utiles à l'écriture de scripts shell (suite)

nice	Exécute un programme avec un niveau de priorité (politesse) modifié.
nohup	Exécute une commande en la rendant insensible aux déconnexions, avec une sortie hors terminal
pathchk	Vérifie la validité et la portabilité d'un nom de fichier
printenv	Affiche l'ensemble ou une partie des variables d'environnement.
printf	Formate et affiche des données.
readlink	Affiche la valeur d'un lien symbolique.
seq	Affiche une séquence de nombres.
shred	Écrit par dessus un fichier pour en camoufler le contenu, et optionnellement l'effacer.
sleep	Endort un processus pour une durée déterminée.
stat	Affiche l'état d'un fichier ou d'un système de fichiers.
stty	Modifie et afficher la configuration de la ligne de terminal.
sync	Vide les tampons du système de fichiers.
tee	Lit depuis l'entrée standard et écrit sur la sortie standard et dans des fichiers.
test	Vérifie le type d'un fichier, et comparer des valeurs.
touch	Modifie l'horodatage d'un fichier ; crée le fichier à vide, s'il n'existe pas.
tty	Affiche le nom de fichier du terminal associé à l'entrée standard.
uname	Affiche des informations sur le système.
users who	Afficher le nom des utilisateurs actuellement connectés sur cette machine.
yes	Affiche indéfiniment une chaîne de caractères jusqu'à ce que le processus soit tué.

Source : http://shell.figarola.fr/x1677.html

Formation Unix de base

➔Annexe : Définition d'un filtre Unix

Nombre de commandes Unix permettent de traiter un flux de données textuelles leur parvenant sur leur entrée standard, et restituent le résultat de leur opération sur la sortie standard : elles sont alors nommées *filtres*.

➔Les principaux filtres Unix

Cette section se bornera à citer les commandes incontournables pour l'écriture de scripts shell, il faudra donc se référer au man de chacune d'entre elles pour obtenir de plus amples renseignement quant à leur fonctionnement.

commande	description synthétique
awk	Recherche et traite des modèles (langage de programmation à part entière, développé infra.)
base64	Encoder/décoder des données et les afficher sur la sortie standard.
comm	Compare ligne à ligne deux fichiers triés.
compress uncompress gzip gunzip bzip2 bunzip2	Compressent et décompressent des données.
cpio tar	Archivent des données.
csplit	Découpe un fichier en sections définies par des lignes de contexte.
cut	Affiche des parties sélectionnées des lignes.
expand unexpand	Convertit les tabulations en espaces et vice-versa.
fmt	Formate simplement du texte.
fold	Coupe chaque ligne de texte à une longueur donnée.
grep egrep fgrep rgrep	Afficher les lignes correspondant à un motif donné (expressions rationnelles).
head	Affiche le début des fichiers.
join	Fusionne les lignes de deux fichiers ayant des champs communs.
nl	Numérote les lignes d'un fichier.
od	Affiche le contenu d'un fichier en octal ou sous d'autres formats.
paste	Regroupe les lignes de différents fichiers.

Formation Unix de base

Les principaux filtres Unix (suite)

commande	description synthétique
pr	Met en forme des fichiers de texte pour l'impression.
ptx	Génére un index croisé du contenu de fichiers.
rev	Renverse l'ordre des caractères pour chaque ligne.
sed	Filtre et transforme des textes (commande développé infra.)
shuf	Génère des permutations aléatoires.
sort tsort	Trie les lignes de fichiers texte
split	Découpe un fichier en différentes parties
sum cksum md5sum sha1sum sha224sum sha256sum sha384sum sha512sum	Calculent de sommes de contrôle (CRC).
tac	cat inversé...
tail	Affiche la dernière partie de fichiers.
tr	Convertit ou élimine des caractères.
uniq	Signale ou élimine les lignes répétées.
wc	Affiche le nombre de lignes, de mots et d'octets d'un fichier.
xargs	Construit et exécute des lignes de commandes à partir de l'entrée standard.

Source : http://shell.figarola.fr/x1677.html

Annexe : bibliographie

•**UNIX pour l'utilisateur :** Commande et langages de commandes, J.L. Nebut, édition technip. Description de l'interface entre l'utilisateur et le système

•**La programmation sous UNIX**, J.-M. Rifflet, ed. Mc Graw-Hill
Présentation des principales caractéristiques et des appels système.

•**Les bases de l'administration système,** Æleen Frisch, 2e

•Edition, avril 1996, ed. O'reilly, ISBN : 2-84177-008-7, 768 pages, 320F.

•**UNIX, une introduction en bref**, A. Strohmeier & P. Kipfer, PPUR 1993.

•**Le système UNIX** (traduction), S. Bourne, 1985, InterEditions.

•**L'environnement de programmation UNIX** (traduction), B. W. Kernighan & R. Pike, 1986, InterEditions.

•**UNIX : introduction**, B. Pouliquen, P. Le Beux, D. Delamarre, 1997, disponible sous Web à l'adresse (URL) : http://www.med.univ-rennes1.fr/~poulique/cours/unix

•**Guide UNIX**, Marc Schaefer, 1994
URL:http://www.pasteur.fr/other/computer/unix/unixguide_html

Annexe : bibliographie

•**Linux** : documentations en Français
•http://uhp.u-nancy.fr/linux/docs-france.html
•**Linux in a nutshell** (manuel de référence, version française), J. P. Hekman, 1997, ed O'Reilly, (isbn: 2-84177-031-1).

Références sur le langage '**awk**' :
•The GNU Awk manual
•Awk -- A Pattern Scanning and Processing Language (Original AWK paper)
•http://www-106.ibm.com/developerworks/library/l-awk1.html
•http://www-106.ibm.com/developerworks/library/l-awk2.html
•http://www-106.ibm.com/developerworks/library/l-awk3.html
•Sed and Awk 2nd Edition (O'reilly).
• awk (cours PowerPoint en français), portail du collège virtuel, http://www.colvir.net/prof/eric.drouin/infR06/notes_ed/awk.ppt

Références sur le langage '**grep**' :
•http://www.linux-kheops.com/doc/man/manfr/man-html-0.9/man1/grep.1.html
•http://technoflash.chez-alice.fr/SDATA/SEMIN/LEGR001.HTM

Cours Unix sur Internet

- Unix : introduction, www.med.univ-rennes1.fr/~poulique/cours/unix
- Cours vi et vim, http://perso.efrei.fr/~kockum/Cours-et-Faq/vi.html
- Cours d'Utilisation Unix, http://devernay.free.fr/cours/unix
- Cours Unix, http://aramis.obspm.fr/~semelin/enseignement/cours_unix.pdf
- Cours utilisateur Unix, http://dillen.thomas.free.fr/Cours/Cours%20UNIX%20(Olivier%20Hoarau%20-%20Linux-France).pdf
- Petit Guide Unix, http://www.emi.ac.ma/~ntounsi/COURS/UNIX/unix_ToC.html
- L'éditeur vi, http://www.emi.ac.ma/~ntounsi/COURS/UNIX/vi_ToC.html
- Sur le scripting shell (sur la rédaction de scripts shell) : Steve's Bourne / Bash shell scripting tutorial, http://steve-parker.org/sh/sh.shtml
- Doc PDF Linux Red Hat Enterprise 6, http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux

- Doc SFR :
- DOCUMENTS NORMES ET GUIDES, http://mysfrv2/communautes/NormesGuidesDOSSI/Pages/Home.aspx

Formation Unix de base

➔**Votre avis et vos commentaires sur la formations ou vos notes, ici :**

Formation Unix de base

➔**N’oubliez pas d’emporter votre manuel de formation Unix en vacances :**

www.steria.com

Merci de votre attention et de votre participation