Supplément sur Linux – Commandes et script

1. Architecture de Linux

Le système d'exploitation Linux n'est pas un bloc monolithique fixé une fois pour toutes. En réalité, de nombreuses composantes travaillent ensemble, rédigés par des personnes différentes et assemblés en distributions. Ce n'est que vu de l'extérieur que le noyau de Linux semble être une unité indissociable. Les distributions ont néanmoins toutes le même noyau du système d'exploitation, et de nombreuses applications communes. [1]

1.1. Le noyau

Est la partie du code en mémoire. Le noyau s'occupe de la :[2]

1.1.1. Gestion de la mémoire

La mémoire physique est étendu virtuellement. Les programmes ou les sections de programmes inutilisés sont déchargés vers le disque dur et chargés en mémoire vive lorsque l'exécution l'exige.

1.1.2. Gestion des fichiers

Linux possède un système hiérarchisé de fichiers dont la structure interne peut varier. Des espaces disques d'autres systèmes peuvent être montés.

1.1.3. Gestions des périphériques

L'accès aux périphériques se fait par l'entremise de fichiers qui constituent l'interface entre les pilotes de périphériques et le noyau.

1.1.4. Gestion des programmes et des processus

Linux assure que chaque programme fonctionne en toute indépendance.

1.1.5. Gestion des droits d'accès

Les données enregistrées dans le système de fichiers doivent être protégées contre les accès non autorisés.

1.2. Le shell

Le shell est la liaison la plus élémentaire entre vous, l'utilisateur, et le système d'exploitation. Vous tapez les commandes qui seront interprétées par le shell et transmises au système d'exploitation. Il existe plusieurs versions de shell :[3]

1.2.1. bash

C'est le shell standard de Linux le Bourne Again Shell

1.2.2. sh

Le shell original le Bourne shell

1.2.3. csh

Le C shell utilise une interface de programmation différente de bash

1.2.4. ksh

Le Korn shell est un des plus populaire sous Unix. Il est compatible avec bash

1.2.5. tcsh

Le C shell amélioré

[1] Michael WIELSCH & Jens PRAHM, H.-G. ESSER, *La Bible Linux*, Micro Application, page 47.
[2] Idem page 46.
[3] Idem page 36.

1.2.6.zsh
Le Z shell est compatible avec bash

1.3. L'interface graphique
Xfree86 constitue une version libre de X Window développé à l'origine au célèbre MIT. À l'époque, en 1987, le but était de réaliser une interface utilisateur graphique qui, de manière souple et sans grandes exigences sur le plan du matériel, devait s'adapter au plus grand nombre de plate forme. Les plus en vogue sont :

1.3.1.Gnome
Le projet GNOME a pour but de construire un environnement complet et convivial basé sur des logiciels libres.

1.3.2.KDE
K Desktop Environnement

2. Les répertoires de Linux
Les répertoires de Linux diffèrent quelque peu des répertoires Unix standard.

2.1. /
Le répertoire racine contient tous les sous répertoire.

2.2. /boot
Au démarrage du système, le programme d'amorçage examinera, entre autres, le répertoire /boot. Parmi les objets recherchés figure le fichier map, par lequel LILO (le gestionnaire d'amorçage de Linux) déterminera l'emplacement du noyau sur le disque dur. Sans cette information, LILO ne peut pas le charger en mémoire vive.

2.3. /bin
Le répertoire binaries contient les commandes les plus importantes.

2.4. /dev
Le répertoire device ne contient pas de fichiers dans le sens classique du mot. Il contient des fichiers de périphériques par lesquels vous communiquez avec les appareils raccordés à l'ordinateur. Aucun utilisateur, pas même le root, ne peut s'adresser directement au matériel. Les fichiers de périphériques constituent des interfaces avec les pilotes de périphériques et n'ont donc pas de contenu.

2.5. /etc
Seuls les fichiers de configuration doivent se trouver là : passwd, group, hosts, etc. Notez le répertoire /etc/X11 qui est le répertoire de l'interface graphique. Xfree y dépose ses informations de configuration. Les administrateurs ont intérêt à noter le répertoire /etc/skel dont les fichiers se copieront dans le répertoire personnel des nouveaux utilisateurs.

2.6. /home
Le répertoire personnel des utilisateurs figurera souvent sous le répertoire /home/VotreNom. L'avantage est que l'utilisateur pourra bénéficier de son propre système de fichiers. Le deuxième avantage est qu'à partir du répertoire personnel, peu de droits en écriture seront distribués. On pourra même définir des quotas de mémoire.

2.7. /lib
Répertoire des bibliothèques partagées.

2.8. /opt
Répertoire des programmes complémentaires qui ne font pas partie intégrante de Linux. Dans la distribution SUSE, KDE figure dans ce répertoire…

2.9. /proc
Le répertoire des pseudo-systèmes de fichiers. Les fichiers n'occupent aucune place sur le disque dur. Ces fichiers ne sont que des constructions logiques, ils pointent vers des programmes en mémoire vive qui lisent directement des informations système central, sans les déposer sur le disque dur.

2.10. /root
Le répertoire de l'administrateur. Il ne faut pas confondre le root, le répertoire /, et le répertoire /root…

2.11. /sbin
C'est le répertoire des commandes d'administration. Au démarrage de Linux, de nombreux programmes et fichiers d'administrations sont cherchés, lu et exploités, ce qui aboutit à l'affichage final de la bannière login. En plus du noyau du système /vmlinuz, les fichiers les plus important pour démarrer Linux figurent dans les répertoires /etc et /sbin. Nous pouvons distinguer trois domaines clés :

2.11.1. Les commandes systèmes générales
Tels que : init, swapon, swapoff, mkswap, getty

2.11.2. Les commandes de démarrage et d'arrêt du système
Tel que : shutdown, fastboot, fasthalt, reboot.

2.11.3. Les commande gérant l'espace du disque dur
Tel que : fsck, e2fsck, mkfs, mkefs et fdisk

2.12. /tmp
Le répertoire des fichiers temporaires peut aussi être créé en RAM. Il est accessible par tous en lecture et écriture.

2.13. /var
Est le répertoire des données variables dans lequel Linux dépose des données variables, se modifiant rapidement ou fréquemment. L'utilisateur peut y écrire. Les fichiers temporaires d'impressions peuvent s'y retrouver.

2.14. /usr
Le répertoire des données sensible est un répertoire qui contient une série de répertoire dans lesquels Linux conserve des données très importantes. Notez la présence de : /usr/doc qui contient la documentation de Linux et /usr/games contient des jeux…

3. Retour sur le système de fichiers
Petit rappel avant de commencer.

3.1. Chemin absolu et relatif
Lorsqu'est créé sur le système un nouvel utilisateur, celui-ci se voit attribuer un répertoire qui devient son répertoire personnel : c'est à dire un répertoire propre de cet utilisateur.

Le répertoire actif est le répertoire par lequel l'utilisateur a directement accès par défaut : c'est le répertoire où se trouve l'utilisateur dans l'arborescence. Lors de sa connexion, le répertoire actif est son répertoire personnel.

Pour donner un chemin d'accès, on peut commencer par la racine /, on parle alors de chemin absolu; ou on peut commencer au répertoire courant, on parle alors de chemin relatif.

Au niveau du répertoire actif, il existe deux pseudonymes ou allias : le point (.) et le deux points (..). Le premier représente le répertoire courant et le second représente le répertoire parent.

4. Niveau de connexion

Le niveau de connexion est mesuré par les droits accordés.

4.1. user

L'usager courant a généralement peu de droits.

4.2. root

Le root ou administrateur est le dieu du système. Il a tous les droits.

5. Connexion et déconnexion

La procédure de connexion est similaire sur toutes les plates-formes. Vous devez obligatoirement vous loger avec un nom et un mot de passe.

5.1. Login, MDP et procédure

Explication en cours…

5.2. exit

Pour vous déconnecter, n'oublier pas la commande exit.

6. Quelques commandes du Shell

Voici la description de certaines commandes du Shell.

6.1. Information système

Les commandes d'informations du système sont des commandes d'ordre général.

MAN

Manuel d'aide en ligne.

```
usage: man [-c|-f|-k|-w|-tZT périphérique] [-adlhu7V] [-Mchemin]
[-Pvisualisateur] [-Sliste] [-msystème] [-pchaîne] [-Llocale]
[-eextension] [section] page
```

Options	Explication
`-a, --all`	Trouve toutes les pages adéquates.
`-d, --debug`	Affiches des messages de déboguage.
`-e, --extension`	Limite la recherche au extension `extension'.
`-f, --whatis`	Équivalent de whatis.
`-k, --apropos`	Équivalent de apropos.
`-w, --where, --location`	Affiche l'emplacement des pages du manuel.
`-l, --local-file`	Interprète l'argument 'page' comme un nom de fichier.
`-u, --update`	Force une vérification de cohérence du cache.
`-r, --prompt string`	Donne une chaîne d'invite au visualisateur `less'.
`-c, --catman`	Utilisé par catman pour reformater les pages trop vieilles.
`-7, --ascii`	Affiche un équivalent ASCII de certains caractères latin1.
`-t, --troff`	Utilise groff pour formater les pages.
`-T, --troff-device`	Utilise groff avec le périphérique sélectionné.
`-Z, --ditroff`	Utilise groff en le forçant à produire ditroff.
`-D, --default`	Redonne leur valeur par défaut à toutes options.
`-M, --manpath chemin`	Positionne le chemin de recherche des pages.
`-P, --pager visualisateur`	Utilise le programme `visualisateur' pour l'affichage des

	pages.
-S, --sections liste	Utilise la liste des sections séparées par des virgules
-m, --systems système	Cherche les pages du manuel provenant d'autres systèmes UNIX.
-L, --locale locale	Défini la localisation pour cette recherche.
-p, --preprocessor chaîne	La chaîne indique le pré-processeur à utiliser. e - [n]eqn p – pic t - tbl g – grap r - refer v – vgrind
-V, --version	Affiche la version.
-h, --help	Affiche ce message-ci.

DATE

Afficher la date courante selon le FORMAT spécifié, ou initialiser la date du système.

```
Usage:     date [OPTION]... [+FORMAT]
   ou:     date [OPTION] [MMJJhhmm[[SS]AA][.ss]]
```

Options	Explication
-d, --date=CHAÎNE	Afficher la date selon la description donnée par la CHAÎNE, excluant le mot réservé `now'
-f, --file=FICHIER	Identique à --date pour chaque ligne du FICHIER de dates
-I, --iso-8601[=SPECS-TEMPS]	Produire un format de sortie date/heure selon la norme ISO-8601 SPECS-TEMPS=`date' (ou manquant) pour la date seulement, `hours', `minutes', or `seconds' pour la date et à la précision voulue
-r, --reference =FICHIER	Utiliser la date de modification du FICHIER comme date de référence
-R, --rfc-822	Afficher la date selon le format respectant les spécifications du RFC-822
-s, --set=FORMAT	Initialiser la date selon le FORMAT décrit
-u, --utc, --universal	Afficher ou initialiser selon le système de temps universel (T.U.)
--help	Afficher l'aide-mémoire
--version	Afficher le nom et la version du logiciel

Les options du FORMAT d'affichage de la date sont les suivantes :

Options	Explication
%%	a literal %
%a	locale's abbreviated weekday name (Sun..Sat)
%A	locale's full weekday name, variable length (Sunday..Saturday)
%b	locale's abbreviated month name (Jan..Dec)
%B	locale's full month name, variable length (January..December)
%c	locale's date and time (Sat Nov 04 12:02:33 EST 1989)
%d	day of month (01..31)
%D	date (mm/dd/yy)
%e	day of month, blank padded (1..31)
%h	same as %b
%H	hour (00..23)
%I	hour (01..12)
%j	day of year (001..366)

%k	hour (0..23)
%l	hour (1..12)
%m	month (01..12)
%M	minute (00..59)
%n	a newline
%p	locale's AM or PM
%r	time, 12-hour (hh:mm:ss [AP]M)
%s	seconds since 00:00:00, Jan 1, 1970 (a GNU extension)
%S	second (00..60)
%t	a horizontal tab
%T	time, 24-hour (hh:mm:ss)
%U	week number of year with Sunday as first day of week (00..53)
%V	week number of year with Monday as first day of week (01..52)
%w	day of week (0..6); 0 represents Sunday
%W	week number of year with Monday as first day of week (00..53)
%x	locale's date representation (mm/dd/yy)
%X	locale's time representation (%H:%M:%S)
%y	last two digits of year (00..99)
%Y	year (1970...)
%z	RFC-822 style numeric timezone (-0500) (a nonstandard extension)
%Z	time zone (e.g., EDT), or nothing if no time zone is determinable

By default, date pads numeric fields with zeroes. GNU date recognizes the following modifiers between `%' and a numeric directive.

`-' (hyphen) do not pad the field
`_' (underscore) pad the field with spaces

WHO
Afficher la liste des usagers présentement branchés sur le système.

```
Usage: who [OPTION]... [ FICHIER| PARAM1 PARAM2 ]
```

Options	**Explication**
-H, --heading	Afficher les en-têtes de colonnes
-i, -u, --idle	Ajouter le temps d'inactivité de l'usager selon le format HEURE:MINUTES, . ou 'vieux'
-l, --lookup	Utiliser la forme canonique des noms des hôtes via le DNS
-m	Seulement du poste (hostname) et de l'usager associé à 'stdin'
-q, --count	Afficher tous les comptes actifs et le nombre d'usagers présents sur le système
-s	(ignorée)
-T, -w, --mesg	Ajouter le statut du message usager avec +, - ou ?
--message	Identique à -T
--writeable	Identique à -T
--help	Afficher l'aide-mémoire
--version	Afficher le nom et la version du logiciel

Si FICHIER n'est pas spécifié, utiliser /var/run/utmp. /var/log/wtmp comme FICHIER est d'usage courant. Si PARAM1 et PARAM2 sont fournis, -m est assumé: `am i' ou `mom likes' sont d'usage courant.

CAL
Afficher un calendrier. Si aucun argument n'est spécifié, il affiche le mois courant.

```
Usage: cal [-jy] [month [year]]
```

Options	Explication
-j	Display julian dates (days one-based, numbered from January 1).
-y	Display a calendar for the current year.

A single parameter specifies the year (1 - 9999) to be displayed; note the year must be fully specified: ``cal 89'' will not display a calendar for 1989. Two parameters denote the month (1 - 12) and year. If no parameters are specified, the current month's calendar is displayed.

6.2. Gestion de fichiers

Les commandes de gestion de fichiers servent à manipuler des fichiers.

CAT

Concaténer le(s) FICHIER(s), ou de l'ENTRÉE standard, vers la sortie standard.

```
Usage:     cat [OPTION] [FICHIER]...
```

Options	Explication
-A, --show-all	Équivalent à -vET
-b, --number-nonblank	Numéroter que les lignes non vides
-e	Équivalent à -vE
-E, --show-ends	Afficher $ à la fin de chaque ligne
-n, --number	Numéroter toutes les lignes
-s, --squeeze-blank	Afficher jamais plus qu'une seule ligne vide
-t	Équivalent à -vT
-T, --show-tabs	Afficher les caractères TAB comme ^I
-u	(ignoré)
-v, --show-nonprinting	Utiliser la notation ^ et M- , excepté pour LFD et TAB
--help	Afficher l'aide-mémoire
--version	Afficher le nom et la version du logiciel

Sans FICHIER, ou quand le FICHIER est -, lire de l'entrée standard. Peut être utilisé avec les opérateurs d'indirection > (sortie), < (entrée) EX.: cat > coco (**ctrl D** pour terminer la saisie, enregistre des données dans un fichier).

CP

Copier le fichier SOURCE vers une DESTINATION, ou plusieurs fichiers SOURCE vers un RÉPERTOIRE.

```
Usage:     cp [OPTION]... SOURCE DESTINATION
  ou:      cp [OPTION]... SOURCE... RÉPERTOIRE
```

Options	Explication
-a, --archive	Identique à -dpR
-b, --backup	Archiver avant de détruire
-d, --no-dereference	Préserver les liens
-f, --force	Détruire les destinations existantes, ne pas demander confirmation
-i, --interactive	Demander confirmation avant d'écraser
-l, --link	Établir des liens sur les fichiers au lieu de copier
-p, --preserve	Préserver les attributs des fichiers si possible
-P, --parents	Accoler le chemin du répertoire source au

REPERTOIRE

Options	Explication
`-r`	Copier récursivement, les non-répertoires comme des fichiers
`--sparse=DATE`	Contrôler la DATE de création des fichiers dispersés
`-R, --recursive`	Copier récursivement les répertoires
`-s, --symbolic-link`	Créer des liens symboliques au lieu de copier
`-S, --suffix=SUFFIXE`	Écraser le suffixe usuel d'archivage par le SUFFIXE
`-u, --update`	Copier seulement les vieux ou les nouveaux fichiers
`-v, --verbose`	Utiliser le mode bavard et indiquer ce qui a été fait
`-V, --version-control=MOT`	Écraser le contrôle de version usuel par le MOT
`-x, --one-file-system`	Demeurer sur ce système de fichiers
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

Par défaut, les fichiers SOURCES dispersés sont détectés par le biais d'une heuristique grossière et le fichier CIBLE correspondant est aussi construit de façon dispersé. Il s'agit d'un comportement sélectionné par l'option --sparse=auto. Spécifiez --sparse=always pour créer un fichier CIBLE dispersé lorsque le fichier SOURCE contient de longues séquences de d'octets de valeur zéro. Utilisez --sparse=never pour inhiber la création de fichiers dispersés.

Le suffixe d'archive est ~, à moins que l'option SIMPLE_BACKUP_SUFFIX soit utilisée. Le contrôle de version VERSION_CONTROL peut être initialisée selon les valeurs suivantes:

Options	Explication
`t, numbered`	Faire des archives numérotées
`nil, existing`	Numéroter si des archives numérotées existent, ne pas numéroter autrement
`never, simple`	Toujours faire des archives de type simple

Un cas spécial où "cp" archive la SOURCE lorsque les options `force' et `backup' sont utilisées et que la SOURCE et la DESTINATION portent le même nom qu'un fichier régulier existant.

LS

Afficher les informations au sujet des FICHIERS (du répertoire courant par défaut). Trier les entrées alphabétiquement si aucune des options -cftuSUX ou --sort n'est utilisée.

```
Usage:     ls [OPTION]... [FICHIER]...
```

Options	Explication
`-a, --all`	Afficher les noms cachés débutant par .
`-A, --almost-all`	Ne pas inclure dans la liste . et ..
`-b, --escape`	Afficher en octal les caractères non-graphiques en utilisant des séquences d'échappement
`--block-size=TAILLE`	Utiliser la TAILLE de blocs
`-B, --ignore-backups`	Ne pas inclure dans la liste, les entrées se terminant par ~
`-c`	Lister les fichiers triés selon leur date de modification; avec -l: les afficher avec la date de modification du 'inode'
`-C`	Afficher en colonnes
`--color[=PARAM]`	Afficher les fichiers avec une couleur selon leur type à l'aide d'un des paramètres suivants: never, always ou

	auto
`-d, --directory`	Lister les noms de répertoires plutôt que leur contenu
`-D, --dired`	Générer une sortie adaptée pour le mode 'dired' de Emacs
`-f`	Ne pas trier, autoriser
`-aU, interdire -lst`	
`-F, --classify`	Ajouter un caractère pour taper chaque entrée
`--format=MODE`	Afficher selon le MODE suivant: -x croisé,-m avec virgules, -x horizontal, -l long, -1 en colonne simple, -l en mode bavard, -C vertical
`--full-time`	Afficher avec la date et l'heure complètes
`-g`	(ignorée)
`-G, --no-group`	Inhiber l'affichage des informations de groupe
`-h, --human-readable`	Afficher les tailles dans un format lisible par un humain (i.e. 1K 234M 2G)
`-H, --si`	Idem mais utiliser un multiple de 1000 au lieu de 1024
`--indicator-style=CODE`	Ajouter en suffixe l'indicateur selon le CODE: none (par défaut), classify (-F), file-type (-p)
`-i, --inode`	Afficher le numéro d'index de chaque fichier
`-I, --ignore=PATRON`	Ne pas inclure dans la liste les entrées concordant avec le PATRON de shell
`-k, --kilobytes`	Utiliser des blocs de 1024 octets, et non pas de 512 octets malgré l'option

POSIXLY_CORRECT

Options	**Explication**
`-l`	Utiliser le format long d'affichage
`-L, --dereference`	Afficher les entrées pointées par des liens symboliques
`-m`	Remplir la largeur par une liste d'entrées séparée par des virgules
`-n, --numeric-uid-gid`	Afficher les valeurs numériques des 'UID' et des 'GID' plutôt que leur nom
`-N, --literal`	Afficher les caractères bruts (i.e. ne pas les traiter comme des caractères de contrôle)
`-o`	Utiliser le format long pour l'affichage sans les informations de groupe
`-p`	Ajouter un caractère pour taper chaque entrée
`-q, --hide-control-chars`	Afficher ? au lieu de caractères non-graphiques
`-Q, --quote-name`	Encapsuler chaque nom d'entrée entre guillemets
`--quoting-style=MOT`	Utiliser le style d'encapsultation selon le MOT clé suivant: literal, shell, shell-always, c, escape
`-r, --reverse`	Inverser l'ordre lors du trie
`-R, --recursive`	Afficher les sous-répertoires récursivement
`-s, --size`	Afficher la taille de chaque fichier en blocs
`-S`	Trier selon la taille des fichiers
`--sort=CODE`	Trier selon le CODE suivant: -c pour ctime, -X pour extension, -U pour aucun, -S pour la taille, -t pour la date, -v pour la version, -c pour le statut, -u pour la date d'accès, -u pour l'accès
`--time=CODE`	Afficher les temps d'accès en mots au lieu de date de modification: atime, access, use, ctime ou status
`-t`	Trier par la date de modification; avec -l: afficher 'mtime'
`-T, --tabsize=TAILLE`	Utiliser la tabulation de la TAILLE pour chaque colonne au lieu de 8
`-u`	Trier selon la date du dernier accès; avec -l: afficher

	'atime'
`-U`	Ne pas trier: afficher selon l'ordre original des entrées d'un répertoire
`-v`	Trier par version
`-w, --width=LARGEUR`	Utiliser la LARGEUR d'écran au lieu des valeurs courantes
`-x`	Afficher les entrées par lignes plutôt que par colonnes
`-X`	Trier alphabétiquement par extension des entrées
`-1`	Afficher un fichier par ligne
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

Par défaut, la couleur n'est pas utilisée pour distinguer les différents types de fichiers. Cela est équivalent à l'utilisation de l'option --color=none. L'utilisation de l'option --color sans le paramètre WHEN est équivalent à l'utilisation de --colors=always. Avec l'option --color=auto, les codes de couleur sont transmis vers la sortie standard si celle-ci est reliée à un terminal (tty).

MV
Renommer la SOURCE selon DESTINATION, ou déplacer les fichiers SOURCES vers le RÉPERTOIRE.

```
Usage:     mv [OPTION]... SOURCE DESTINATION
   ou:     mv [OPTION]... SOURCE... RÉPERTOIRE
```

Options	**Explication**
`-b, --backup`	Archiver avant de détruire
`-f, --force`	Détruire les destinations, sans demander confirmation
`-i, --interactive`	Demander confirmation avant d'écraser
`-S, --suffix=SUFFIXE`	Écraser le suffixe d'archivage usuel en utilisant SUFFIXE
`-u, --update`	Déplacer seulement les vieux ou les tous nouveaux fichiers
`-v, --verbose`	Utiliser le mode bavard et indiquer ce qui a été fait
`-V, --version-control=CODE`	Écraser le contrôle de version par le CODE
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

Le suffixe d'archive est ~, à moins que l'option SIMPLE_BACKUP_SUFFIX soit utilisée. Le contrôle de version VERSION_CONTROL peut être initialisée selon les valeurs suivantes:

Options	**Explication**
`t, numbered`	Faire des archives numérotées
`nil, existing`	Numéroter si des archives numérotées existent, ne pas numéroter autrement
`never, simple`	Toujours faire des archives de type simple

RM
Enlever (remove) les FICHIER(s).

```
Usage: rm [OPTION]... FICHIER...
```

Options	**Explication**
`-d, --directory`	Enlever le répertoire, même si non vide (usager root seulement)
`-f, --force`	Ignorer les fichiers inexistants, ne pas demander de

	confirmation
`-i, --interactive`	Demander une confirmation avant chaque destruction
`-r, -R, --recursive`	Enlever le contenu des répertoires récursivement
`-v, --verbose`	En mode bavard expliquer ce qui a été fait
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

MKDIR
Créer des RÉPERTOIRES, s'ils n'existent pas déjà.

```
Usage:    mkdir [OPTION] RÉPERTOIRE...
```

Options	**Explication**
`-m, --mode=MODE`	Utiliser le MODE des permissions d'accès (comme avec chmod), et non pas le mode rwxrwxrwx - umask
`-p, --parents`	Si l'exécution est sans erreur parce qu'existant: créer des répertoires parents si nécessaire
`--verbose`	Utiliser le mode bavard et afficher un message pour chaque répertoire créé
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

RMDIR
Enlever les RÉPERTOIRES, s'ils sont vides.

```
Usage:    rmdir [OPTION]... RÉPERTOIRE...
```

Options	**Explication**
`--ignore-fail-on-non-empty`	Ignorer les échecs qui sont causées par un répertoire qui n'est pas vide
`-p, --parents`	Enlever les parents explicites de répertoires s'ils sont vides
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

DU
Produire un sommaire de l'utilisation de l'espace disque de chaque FICHIER et récursivement dans les répertoires.

Usage: du [OPTION]... [FICHIER]...

Options	**Explication**
`-a, --all`	Afficher le décompte pour tous les fichiers, pas seulement pour les répertoires
`--block-size=TAILLE`	Utiliser la TAILLE de blocs
`-b, --bytes`	Afficher la taille en octets
`-c, --total`	Produire le grand total
`-D, --dereference-args`	Ne pas tenir compte des CHEMINS lorsqu'il y a des liens symboliques
`-h, --human-readable`	Afficher les tailles dans un format lisible par un humain (i.e. 1K 234M 2G)
`-H, --si`	Idem mais utiliser un multiple de 1000 au lieu de 1024
`-k, --kilobytes`	Utiliser des blocs de 1024 octets, et non pas de 512 octets malgré l'option

POSIXLY_CORRECT

Options	Explication
`-l, --count-links`	Dénombrer les tailles aussi souvent qu'il y a de liens directs
`-L, --dereference`	Ne pas tenir compte de tous les liens symboliques
`-m, --megabytes`	Utiliser des blocs de 1024K-octets, et non pas de 512 octets malgré l'option
`-S, --separate-dirs`	Ne pas inclure la taille des sous-répertoires
`-s, --summarize`	Afficher seulement un total pour chaque type de paramètre
`-x, --one-file-system`	Escamoter les répertoires de différents
`-X FICHIER, --exclude-from=FICHIER`	Exclure les fichiers qui concordent avec le nom du FICHIER
`--exclude=EXPRES`	Exclure les fichiers qui concordent avec l'expression
`--max-depth=N`	Afficher le total pour un répertoire (ou un fichier, avec l'option --all) seulement si N a moins de niveau dans la ligne de commande;
`--max-depth=0`	Est identique à --summarize systèmes de fichiers
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

DF

Afficher les informations à propos du système de fichiers sur lequel réside chaque FICHIER ou de tous les systèmes de fichier par défaut.

```
Usage:    df [OPTION]... [FICHIER]...
```

Options	Explication
`-a, --all`	Inclure les systèmes de fichiers ayant 0 bloc
`--block-size=TAILLE`	Utiliser la TAILLE de blocs
`-h, --human-readable`	Afficher les tailles dans un format lisible par un humain (i.e. 1K 234M 2G)
`-H, --si`	Idem mais utiliser un multiple de 1000 au lieu de 1024
`-i, --inodes`	Lister les informations sur les 'inodes' plutôt que sur l'utilisation des blocs
`-k, --kilobytes`	Utiliser des blocs de 1024 octets, et non pas de 512 octets malgré l'option

POSIXLY_CORRECT

Options	Explication
`-m, --megabytes`	Utiliser des blocs de 1024K-octets, et non pas de 512 octets malgré l'option
`--no-sync`	Ne pas effectuer une synchronisation avant d'obtenir les informations d'utilisation des disques (par défaut)
`-P, --portability`	Utiliser le format de sortie POSIX
`--sync`	Demander une synchronisation avant d'obtenir les informations d'utilisation des disques (par défaut)
`-t, --type=TYPE`	Limiter l'affichage au TYPE de système de fichiers
`-T, --print-type`	Afficher le type du système de fichiers
`-x, --exclude-type=TYPE`	Limiter l'affichage en excluant le TYPE de système de fichiers
`-v`	(ignorée)
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

PWD
Affiche le nom du répertoire de travail courant.

```
Usage:    pwd [OPTION]...
```

Options	Explication
`--help Print`	A usage message on standard output and exit successfully.
`--version`	Print version information on standard output then exit successfully.

6.3. Gestion des droits

Voici quelques commandes servant à gérer les droits d'accès, d'écriture et de lecture ainsi que les droits de propriétés des fichiers.

CHMOD
Changer le modificateur des droits d'accès d'un fichier ou d'un répertoire.

```
Usage:     chmod [OPTION]... MODE[,MODE]... FICHIER...
   ou:     chmod [OPTION]... MODE_OCTAL FICHIER...
   ou:     chmod [OPTION]... --reference=FICHIER_RÉFÉRENCE...
```

Options	Explication
`-c, --changes`	Utiliser le mode bavard mais rapporter seulement les modifications lorsqu'elles surviennent
`-f, --silent, --quiet`	Supprimer la plupart des messages d'erreur
`-v, --verbose`	Produire un diagnostic pour chaque fichier traité
`--reference=FICHIERS`	Utiliser le mode des FICHIERS de référence au lieu de valeurs
`-R, --recursive`	Modifier récursivement fichiers et répertoires
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

Chaque MODE se compose d'une ou plusieurs des options: ugoa (user, group, others, all), un des symboles +-= et d'une ou plusieurs des options: rwx (read, write, execute).

Calcul de la valeur octale des droits d'accès

	r (read)	**w (write)**	**x (execute)**	**Total**
Propriétaire	400	200	100	700
Groupe	40	20	10	70
Autres	4	2	1	7

Signification des droits d'accès pour un fichier normal. [4]

Droits	**Explication**
`r`	Le contenu du fichier peut être lu (read).
`w`	Le contenu du fichier peut être modifié. Le fait que le fichier peut être supprimé ne fait pas parti de ses caractéristiques propres, mais de celles du répertoire où il est placé (write).
`x`	Le fichier contient un programme et peut être exécuté. On distingue deux types de programmes : les programmes binaires et les scripts (execute).

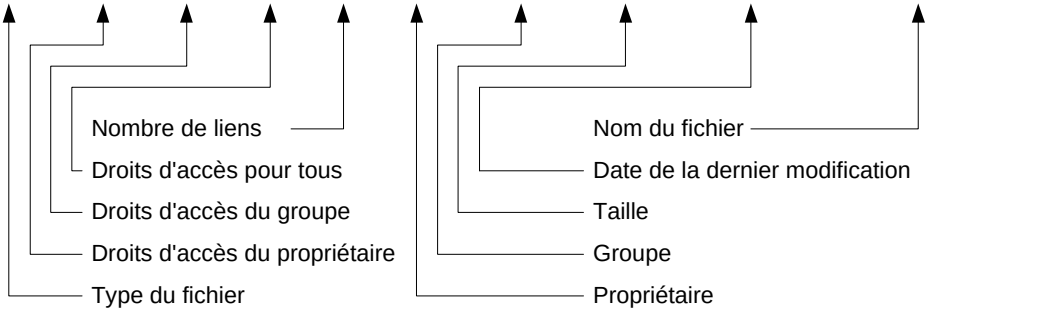
[4] M. WIELSCH & H.G. ESSER et T. FORSTER, *Linux toutes distributions*, Micro Application, PC Poche, 1999, page152.

Signification des droits d'accès pour un répertoire. [5]

Droits	Explication
r	Les éléments du répertoire sont accessibles en lecture. Cette autorisation est nécessaire pour la commande `ls` et pour le shell pour établir par exemple les modèles de critères de recherche sur les noms de fichiers (read).
w	Les éléments du répertoire sont modifiables. L'utilisateur peut créer de nouveaux fichiers dans ce répertoire et peut supprimer les fichiers existants, cette dernière possibilité étant indépendante des autorisations d'accès aux fichiers proprement dits (write).
x	Le nom du répertoire peut apparaître dans un chemin d'accès. Par conséquent, vous n'aurez pas accès à un répertoire par la commande `cd` si vous ne disposez pas au minimum de l'autorisation d'exécution. Tous les fichiers et répertoires contenu dans ce répertoire sont totalement verrouillés si cette autorisation fait défaut (execute).

Affichage des droits (`ls -l`)

```
d     rwx   r-x   r-x   4 csimard  users   4096 Aug 29 09:15 .
d     rwx   ---   ---  13 csimard  users   4096 Aug 29 08:17 ..
d     rwx   r-x   r-x   2 root     root    4096 Aug 29 09:56 cours9
d     rwx   r-x   r-x   2 csimard  users   4096 Aug 18 09:54 notes
-     rw-   r--   r--   1 csimard  users  19289 Aug 18 09:54 commandes.txt
```



CHOWN
Changer l'appartenance de chaque FICHIER au PROPRIÉTAIRE et/ou au GROUPE.

```
Usage:     chown [OPTION]... PROPRIÉTAIRE[.[GROUPE]] FICHIER...
   ou:     chown [OPTION]... [GROUPE] FICHIER...
   ou:     chown [OPTION]... --reference=FICHIER_RÉFÉRENCE...
```

Options	Explication
`-c, --changes`	Utiliser le mode bavard mais rapporter seulement les modifications lorsqu'elles surviennent
`--dereference`	Modifier les références de chaque lien symbolique, plutôt que le lien symbolique lui-même
`-h, --no-dereference`	Modifier les liens symboliques au lieu des fichiers

[5] M. WIELSCH & H.G. ESSER et T. FORSTER, *Linux toutes distributions*, Micro Application, PC Poche, 1999, page152..

	référencés (disponible seulement sur les systèmes offrant l'appel système lchown)
`-f, --silent, --quiet`	Supprimer la plupart des messages d'erreur
`--reference=FICH`	Utiliser l'appartenance du propriétaire et du groupe du FICHIER de référence au lieu de valeurs explicites PROPRIÉTAIRE.GROUPE
`-R, --recursive`	Modifier récursivement fichiers et répertoires
`-v, --verbose`	Indiquer ce qui a été fait
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

Si non spécifié, le propriétaire demeure le même. Si non spécifié, le groupe d'appartenance reste inchangé, autrement il est attribué au groupe d'établissement de session lorsqu'un point est présent. Un ':' peut remplacer le point.

CHGRP
Changer le groupe d'appartenance de chaque FICHIER au GROUPE.

```
Usage:     chgrp [OPTION]... GROUPE FICHIER...
   ou:     chgrp [OPTION]... --reference=FICHIER RÉFÉRENCE...
```

Options	**Explication**
`-c, --changes`	Utiliser le mode bavard mais rapporter seulement les modifications lorsqu'elles surviennent
`-h, --no-dereference`	Modifier les liens symboliques au lieu des fichiers référencés (disponible seulement sur les systèmes offrant l'appel système lchown)
`-f, --silent, --quiet`	Supprimer la plupart des messages d'erreur
`--reference=FICH`	Utiliser le groupe de référence des FICHIERS au lieu d'une valeur de groupe
`-R, --recursive`	Modifier récursivement fichiers et répertoires
`-v, --verbose`	Produire un diagnostic pour chaque fichier traité
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

UMASK
Modifier la valeur octale de base lors de la création d'un fichier ou d'un répertoire.

```
Usage: umask VALEUR
```

Masque d'un fichier

Valeur maximale	`rw-rw-rw-`	666
A retirer (`umask 022`)	`----w--w-`	022
Résultat	`rw-r--r--`	644

Masque d'un répertoire

Valeur maximale	`rwxrwxrwx`	777
A retirer (`umask 022`)	`----w--w-`	022
Résultat	`rwxr-xr-x`	755

6.4. filtre

CUT
Copier des sections de certaines lignes d'un fichier.

```
Usage:     cut -b   byte-list,  --bytes=byte-list  [-n]  [--help]
         [--version] [file...]
ou:        cut -c   character-list,   --characters=character-list
         [--help] [--version] [file...]
ou:     cut -f  field-list, --fields=field-list [-d delim] [-s]
         [--delimiter=delim]  [--only-delimited][--help] [--version] [file...]
```

The byte-list, character-list, and field-list are one or more numbers or ranges (two numbers separated by a dash) separated by commas. The first byte, character, and field are numbered 1. Incomplete ranges may be given: `-m' means `1-m'; `n-' means `n' through end of line or last field.

Options	Explication
`-b, --bytes byte-list`	Print only the bytes in positions listed in byte-list. Tabs and backspaces are treated like any other character; they take up 1 byte.
`-c, --characters character-list`	Print only characters in positions listed in character-list. The same as -b for now, but internationalization will change that. Tabs and backspaces are treated like any other character; they take up 1 character.
`-f, --fields field-list`	Print only the fields listed in field-list. Fields are separated by a TAB by default.
`-d, --delimiter delim`	For -f, fields are separated by the first character in delim instead of by TAB.
`-n`	Do not split multibyte characters (no-op for now).
`-s, --only-delimited`	For -f, do not print lines that do not contain the field separator character.
`--help`	Print a usage message and exit with a status code indicating success.
`--version`	Print version information on standard output then exit.

FIND
La commande find sert à rechercher de façon récursive des fichiers dans des répertoires et selon les options spécifiées.

```
Usage: find [répertoire(s)] [critères de sélection]
```

Options	Explication
`-name`	Recherche par le nom
`-type`	Recherche par le type : `f` (normal) `d` (directory - répertoire) `c` (caractère) `b` (bloc)
`-user`	Recherche par le nom d'utilisateur
`-group`	Recherche par le nom de groupe
`-size`	Recherche par la taille `-size n` (taille exacte de bloc) `-size nc` (taille exacte en octet) `-size +n` (plus de n) `-size -n` (moins de n)
`-atime` `-mtime`	Recherche selon : access time (dernier accès)

`-ctime`	modification time (dernière modification) creation time (date de création) + ou - détermine le nombre de jours de plus ou de moins selon le type de recherche
`-perm`	Recherche selon la permission
`-link`	Recherche par nombre de lien
`-exec`	Permet l'exécution de la commande qui suit immédiatement le mot \exec sur chacun des fichiers trouvés. Notez que l'option -exec doit être la dernière de la commande find. Le nom de la commande suivant \exec doit être suivi de "\;" Les deux accolades "{ }" devront suivre le nom de la commande donné pour spécifier que les arguments de la commande sont les fichiers trouvés.
`-ok`	Identique à \exec sauf que celle-ci requiert une confirmation avant l'exécution de la commande
`-print`	Affiche la liste de fichiers trouvés. Cette option est nécessaire, sur certains systèmes, si l'on veut obtenir un affichage à l'écran

GREP
Recherche un PATTERN ou PATRON dans un FICHIER ou une sortie standard.

```
Usage:    grep [OPTION]... PATRON [FICHIER] ...
```

Options	**Explication**
`-E, --extended-regexp`	PATTERN is an extended regular expression
`-F, --fixed-strings`	PATTERN is a set of newline-separated strings
`-G, --basic-regexp`	PATTERN is a basic regular expression
`-e, --regexp=PATTERN`	Utiliser le PATRON comme expression régulière
`-f, --file=FILE`	Obtenir le PATRON du FICHIER
`-i, --ignore-case`	Ignorer la distinction de la casse
`-w, --word-regexp`	Forcer l'appariement du PATRON que sur des mots complets
`-x, --line-regexp`	Forcer l'appariement du PATRON que sur des lignes entières
`-z, --null-data`	Terminer la ligne de données par ZÉRO et non pas par un retour de chariot

Miscellaneous:

Options	**Explication**
`-s, --no-messages`	Suppress error messages
`-v, --invert-match`	Select non-matching lines
`-V, --version`	Print version information and exit
`--help`	Display this help and exit
`--mmap`	Use memory-mapped input if possible

Contrôle de sortie:

Options	Explication
`-b, --byte-offset`	Afficher les adresses relatives des lignes traitées
`-n, --line-number`	Afficher les numéros de lignes des lignes traitées
`-H, --with-filename`	Afficher le nom de fichier pour chaque concordance
`-h, --no-filename`	Supprimer le préfixe du nom de fichier sur la sortie
`-q, --quiet, --silent`	Supprimer tout affichage en sortie
`-a, --text`	Ne pas supprimer la sortie binaire
`-d, --directories=ACTION`	Traiter les répertoires selon l'ACTION 'read' (lecture), 'recurse' (récursivité), ou 'skip' (escamotage).
`-r, --recursive`	Équivalent à --directories=recurse.
`-L, --files-without-match`	Afficher seulement les noms des fichiers ne contenant pas de concordance
`-l, --files-with-matches`	Afficher seulement les noms des fichiers contenant des concordances
`-c, --count`	Afficher seulement le décompte des lignes concordantes par fichier
`-Z, --null`	Afficher l'octet ZÉRO après le nom du fichier

Context control:

Options	Explication
`-B, --before-context=NUM`	Print NUM lines of leading context
`-A, --after-context=NUM`	Print NUM lines of trailing context
`-C, --context[=NUM]`	Print NUM (default 2) lines of output context unless overridden by -A or -B
`-NUM`	Same as --context=NUM
`-U, --binary`	Do not strip CR characters at EOL (MSDOS)
`-u, --unix-byte-offsets`	Report offsets as if CRs were not there (MSDOS)

`egrep' means `grep -E'. `fgrep' means `grep -F'. With no FILE, or when FILE is -, read standard input. If less than two FILEs given, assume -h. Exit status is 0 if match, 1 if no match, and 2 if trouble.

Les caractères spéciaux sont:

Caractères	Signification
`[ ]`	Les crochets délimitent un ensemble de caractères représentant l'emplacement d'un seul caractère qui peut être n'importe quel de l'ensemble.
`[ ^ ...]`	Négation des caractères de l'ensemble.
`.`	Un caractère quelconque.
`*`	Un caractère de répétition.
`$`	Une fin de ligne.
`^`	Un début de ligne.

PASTE

Copie des lignes de fichiers.

```
Usage:    paste [-s] [-d delim-list] [--serial]
        [--delimiters=delim-list] [--help] [--version] [file...]
```

Options	Explication
`-s, --serial`	Paste the lines of one file at a time rather than one line from each file.
`-d, --delimiters delim-list`	Consecutively use the characters in delim-list

	instead of TAB to separate merged lines. When delim-list is exhausted, start again at its beginning.
`--help`	Print a usage message and exit with a status code indicating success.
`--version`	Print version information on standard output thenexit.

SORT
Écrire la concaténation triée de tous les FICHIERS sur la sortie standard.

```
Usage:    sort [OPTION]... [FICHIER]...
```

Options	**Explication**
`+POS1 [-POS2]`	Débuter avec la clé de position POS1, et terminer *avant* POS2 (désuet) les numéros de champs et les positions relatives des caractères sont comptés à partir de zéro (contrairement au décompte à partir de un de l'option -k)
`-b`	Ignorer les blancs de tête dans les champs ou les clés triés
`-c`	Vérifier si un fichier soumis a déjà été trié, si oui ne pas trier
`-d`	Considérer seulement les caractères [a-zA-Z0-9] comme clés
`-f`	Considérer les minuscules comme des majuscules et comme clés
`-g`	Comparer selon la valeur numérique générale, implique -b
`-i`	Considérer seulement les caractères [\040-\0176] comme clés
`-k POS1[,POS2]`	Débuter à la position POS1 et terminer *à* POS2, les numéros de champs et les positions relatives des caractères sont comptés à partir de un (contrairement au décompte à partir de zéro de la forme +POS1)
`-m`	Fusionner les fichiers triés, ne pas trier
`-M`	Comparer selon (inconnu) < `JAN' < ... < `DÉC', implique -b
`-n`	Comparer selon la valeur numérique de la chaîne, implique -b
`-o FICHIER`	Produire le résultat dans le FICHIER au lieu de la sortie standard
`-r`	Inverser le résultat des comparaisons
`-s`	Stabiliser le trie en inhibant la dernière comparaison
`-t SÉPARATEUR`	Utiliser le SÉPARATEUR au lieu de la transition non blanc à blanc
`-T RÉPERTOIRE`	Utiliser le RÉPERTOIRE temporaire, non pas $TMPDIR ou /tmp
`-u`	Avec -c, vérifier l'ordonnancement strict avec -m, afficher seulement la première séquence identique
`-z`	Terminer les lignes avec un octet de valeur 0, pour la commande find find -print0
`--help`	Afficher l'aide-mémoire
`--version`	Afficher le nom et la version du logiciel

POS is F[.C][OPTS], where F is the field number and C the character position in the field, both counted from one with -k, from zero with the obsolescent form. OPTS is made up of one or more of Mbdfinr; this effectively disables global -Mbdfinr settings for that key. If no key is given, use the entire line as the key. With no FILE, or when FILE is -, read standard input.

TAIL
Affiche les 10 dernières lignes de FICHIER dans la sortie standard.

```
Usage:    tail [OPTION]... [FICHIER]...
```

Options	**Explication**
`--retry`	Keep trying to open a file even if it is inaccessible when tail starts or if it becomes inaccessible later -

	- useful only with -f
`-c, --bytes=N`	Output the last N bytes
`-f, --follow[={name|descriptor}]`	Output appended data as the file grows; -f, --follow, and --follow=descriptor are equivalent
`-n, --lines=N`	Output the last N lines, instead of the last 10
`--max-unchanged-stats=N`	See the texinfo documentation (the default is 5)
`--max-consecutive-size-changes=N`	See the texinfo documentation (the default is 200)
`--pid=PID`	With -f, terminate after process ID, PID dies
`-q, --quiet, --silent`	Never output headers giving file names
`-s, --sleep-interval=S`	With -f, sleep S seconds between iterations
`-v, --verbose`	Always output headers giving file names
`--help`	Display this help and exit
`--version`	Output version information and exit

If the first character of N (the number of bytes or lines) is a `+', print beginning with the Nth item from the start of each file, otherwise, print the last N items in the file. N may have a multiplier suffix: b for 512, k for 1024, m for 1048576 (1 Meg). A first OPTION of –VALUE or +VALUE is treated like -n VALUE or -n +VALUE unless VALUE has one of the [bkm] suffix multipliers, in which case it is treated like -c VALUE
or -c +VALUE.

With --follow (-f), tail defaults to following the file descriptor, which means that even if a tail'ed file is renamed, tail will continue to track its end. This default behavior is not desirable when you really want to track the actual name of the file, not the file descriptor (e.g., log rotation). Use --follow=name in that case. That causes tail to track the named file by reopening it periodically to see if it has been removed and recreated by some other program.

TEE
Copier de l'entrée standard vers chaque FICHIER, et également vers la sortie standard.

```
Usage:    tee [OPTION]... [FICHIER]...
```

Options	**Explication**
-a, --append	Accoler la sortie au(x) FICHIER(s), sans les écraser
-i, --ignore-interrupts	Ignorer les signaux d'interruption
--help	Afficher l'aide-mémoire
--version	Afficher le nom et la version du logiciel

TR
Transformer une chaîne de caractère en une autre.

```
Usage:    tr [OPTION]... [CHAÎNE1] [CHAÎNE2]
```

Options	**Explication**
-d	Détruis (delete) les éléments de la chaîne

7. La General Public Licence

La transcription française de la licence GPL se trouve à l'Annexe D. De plus, vous trouverez le site de GNU à l'adresse : http://www.gnu.org/

8. Commandes Linux avancées

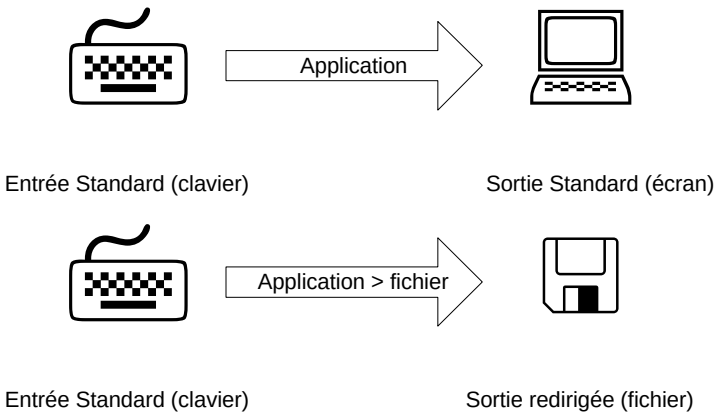
Voyons maintenant quelques notions plus avancées.

8.1. La redirection [6]

La redirection des entrées-sorties est une des principales caractéristiques du shell, dont la puissance n'a d'égal que la simplicité. La réputation de Linux, comme système d'exploitation souple et performant, est due en grande partie aux possibilités offertes de rediriger directement ou indirectement, les entrées-sorties. [7]

Beaucoup d'utilitaires Linux envoient des informations à l'écran. Il est parfois difficile d'avoir une vue d'ensemble immédiate de ces informations. Dans ce cas, il serait intéressant de pouvoir envoyer ces données dans un fichier, de les rediriger. Grâce à un éditeur de texte, vous pourrez ensuite les modifier ou en prendre connaissance tranquillement. [8]

Toutes les commandes utilisent des canaux d'entrées-sorties pour lire des données ou transmettre leurs informations. Le canal d'entrée utilisé en général pour la lecture est lié au clavier. Linux pilote les canaux d'entrées-sorties de manière indépendante pour chaque utilisateur, chacun voyant son clavier lié à un canal d'entrée. Linux gère de la même façon les canaux de sortie. Le canal de sortie par défaut est lié à l'écran devant lequel est assis l'ordinateur.[9]



Caractères de redirection	Nom
< fichier	Redirection d'entrée par fichier
> fichier	Redirection de sortie vers fichier
<< fichier	Redirection d'entrée ajout au fichier (rare)
>> fichier	Redirection de sortie ajout au fichier
n > fichier	Envoi un canal (0, 1, 2) vers fichier. Cette commande est très pratique pour éviter les messages d'erreur à l'écran. Ex: `2> err.txt` Envoi tous les messages d'erreurs vers err.txt

[6] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 164.
[7] M. WIELSCH & H.G. ESSER et T. FORSTER, *Linux toutes distributions*, Micro Application, PC Poche, 1999, page 79.
[8] Idem.
[9] Idem.

8.2. Les canaux

Voici les différents canaux dont dispose chaque application.

Numéro du canal	Fonction
0	Entrée standard (clavier)
1	Sortie standard (écran)
2	Sortie d'erreur standard (écran)

8.3. Les tubes

Pour établir une liaison directe entre le canal de sortie standard d'une commande et le canal d'entrée standard d'une autre, il existe le signe | (le tube ou pipe ou pipeline). Toutes les données renvoyées par la commande placée à gauche de ce signe à travers le canal de sortie standard sont envoyés au canal d'entrée standard de la commande placée à droite. [10]

Ex: `commande 1 | commande 2 | commande3 ...`

9. Exercices supplémentaires avec les commandes Linux

9.1. Exercices avec grep

```
root@cours11 -->cat fruits
pomme
poire
orange
pamplemousse
fraise
banane

root@cours11 -->grep ^[a-f] fruits
fraise
banane

root@cours11 -->grep ^[aeiou] fruits
orange

root@cours11 -->grep ^[^aeiou] fruits
pomme
poire
pamplemousse
fraise
banane

root@cours11 -->grep ^.o fruits
pomme
poire

root@cours11 -->grep ..*m fruits
pomme
pamplemousse

root@cours11 -->grep se$ fruits
pamplemousse
fraise


root@cours11 -->grep -v ^p fruits
orange
fraise
banane
```

[10] M. WIELSCH & H.G. ESSER et T. FORSTER, *Linux toutes distributions*, Micro Application, PC Poche, 1999, page89.

```
root@cours11 -->grep -v -c ^p fruits
3

root@cours11 -->grep -vc ^p fruits
3
```

9.2. Exercices avec find

```
root@cours11 -->pwd
/home/csimard/garneau/cours11

root@cours11 -->ls -l
total 16
drwxr-xr-x   2 root     root         4096 Aug 29 09:56 .
drwxr-xr-x   4 csimard  users        4096 Aug 29 09:15 ..
-rw-r--r--   1 root     root           17 Aug 29 09:56 blabla
-rw-r--r--   1 root     root           46 Aug 29 09:16 fruits

root@garneau -->pwd
/home/csimard/garneau

root@garneau -->ls -l
total 16
drwxr-xr-x   4 csimard  users        4096 Aug 29 09:15 .
drwx------  13 csimard  users        4096 Aug 29 08:17 ..
drwxr-xr-x   2 root     root         4096 Aug 29 09:56 cours11
drwxr-xr-x   2 csimard  users        4096 Aug 18 09:54 notes

root@garneau -->find cours11 -name fruits -print
cours11/fruits

root@garneau -->find cours11 -name fruits
cours11/fruits

root@garneau -->find . -type d
.
./notes
./cours11

root@garneau -->find . -user root
./cours11
./cours11/fruits
./cours11/blabla

root@garneau -->find cours11 -user root
cours11
cours11/fruits
cours11/blabla

root@garneau -->find . -user csimard
.
./notes
./notes/commandes2
./notes/commandes.txt
./notes/commandes.txt~

root@notes -->ls -l
total 136
drwxr-xr-x   2 csimard  users        4096 Aug 18 09:54 .
drwxr-xr-x   4 csimard  users        4096 Aug 29 09:15 ..
-rw-r--r--   1 csimard  users       19289 Aug 18 09:54 commandes.txt
-rw-r--r--   1 csimard  users       19660 Aug 18 09:52 commandes.txt~
-rw-r--r--   1 csimard  users       82840 Aug 18 09:33 commandes2

root@garneau -->find . -group root
```

```
./cours11
./cours11/fruits
./cours11/blabla

root@garneau -->find . -group users
.
./notes
./notes/commandes2
./notes/commandes.txt
./notes/commandes.txt~

root@garneau -->find . -size +20c
.
./notes
./notes/commandes2
./notes/commandes.txt
./notes/commandes.txt~
./cours11
./cours11/fruits

root@garneau -->find . -size -20c
./cours11/blabla

root@/root -->find . -ctime +30
(le contenu est très long...)

root@garneau -->ls
.  ..  cours11  notes
root@garneau -->mkdir pasrapport
root@garneau -->ls
.  ..  cours11  notes  pasrapport
root@garneau -->chmod 000 pasrapport
root@garneau -->ls -l
total 20
drwxr-xr-x   5 csimard  users        4096 Aug 29 10:30 .
drwx------  13 csimard  users        4096 Aug 29 08:17 ..
drwxr-xr-x   2 root     root         4096 Aug 29 10:03 cours11
drwxr-xr-x   2 csimard  users        4096 Aug 18 09:54 notes
d---------   2 root     root         4096 Aug 29 10:30 pasrapport

root@garneau -->find . -perm 000 -type d
./pasrapport

root@garneau -->find . -type d -exec ls -l {} \;
total 12
drwxr-xr-x   2 root     root         4096 Aug 29 10:03 cours11
drwxr-xr-x   2 csimard  users        4096 Aug 18 09:54 notes
d---------   2 root     root         4096 Aug 29 10:30 pasrapport
total 128
-rw-r--r--   1 csimard  users       19289 Aug 18 09:54 commandes.txt
-rw-r--r--  1 csimard  users       19660 Aug 18 09:52 commandes.txt~
-rw-r--r--   1 csimard  users       82840 Aug 18 09:33 commandes2
total 36
-rw-r--r--   1 root     root        28160 Aug 29 10:03 Cours11.sdw
-rw-r--r--   1 root     root           17 Aug 29 09:56 blabla
-rw-r--r--   1 root     root           46 Aug 29 09:16 fruits
total 0
```

9.3. Exercices avec cut

L'option type est soit une colonne caractère "-c" ou un champ mot "-f". Notez que la numérotation commence à 1. Il est possible de sélectionner un intervalle (-c2-4) ou une liste d'intervalles (-c2-4, 4-8)

```
root@cours11 -->cat voitures
```

```
Mercedes        neuve   55000
Camaro Z28      neuve   45000
Chevrolet       usee    17000
Lamborghini     neuve   555000
Tricycle        neuve   25

root@cours11 -->cut -c2 voitures
e
a
h
a
r

root@cours11 -->cut -c1-3 voitures
Mer
Cam
Che
Lam
Tri

root@cours11 -->cut -f1-2 voitures
Mercedes        neuve
Camaro Z28      neuve
Chevrolet       usee
Lamborghini     neuve
Tricycle        neuve
```

9.4. Exercices avec paste

Il existe deux options possibles -d qui permet de définir un novueau délimiteur et -s qui permet de tout coller sur la même ligne (subsequent lines).

```
root@cours11 -->cut -f1 voitures > v1

root@cours11 -->cut -f3 voitures > v2

root@cours11 -->cat v1 v2
Mercedes
Camaro Z28
Chevrolet
Lamborghini
Tricycle
55000
45000
17000
555000
25

root@cours11 -->paste v1 v2 > voitures2

root@cours11 -->cat voitures2
Mercedes        55000
Camaro Z28      45000
Chevrolet       17000
Lamborghini     555000
Tricycle        25

root@cours11 -->cut -f1 voitures | paste -s
Mercedes            Camaro  Z28         Chevrolet           Lamborghini
Tricycle
```

9.5. Exercices avec tr

La commande tr permet de changer les caractères spécifiés d'un fichier par d'autres.

Syntaxe : tr "caractère(s) à trouver" "caractères(s) à implanter"

```
root@cours11 -->cat fruits
pomme
poire
orange
pamplemousse
fraise
banane

root@cours11 -->cat fruits | tr "o" "z"
pzmme
pzire
zrange
pamplemzusse
fraise
banane

root@cours11 -->cat fruits | tr "[a-z]" "[A-Z]"
POMME
POIRE
ORANGE
PAMPLEMOUSSE
FRAISE
BANANE

root@cours11 -->tr "\012" " " < fruits
pomme poire orange pamplemousse fraise banane
```

10. Introduction à la programmation du shell Bash de Linux

"Shell" est le terme Unix anglophone utilisé pour désigner une interface avec le système d'exploitation. C'est à dire, quelque chose qui vous permet de communiquer avec l'ordinateur par l'entremise du clavier et de l'écran. Le shell est un programme séparé du système d'exploitation. De ce fait, il est possible de choisir le shell que l'on préfère et de l'installer. Le shell généralement utilisé par les distributions de Linux est le Bash (Bourne again shell). [11]

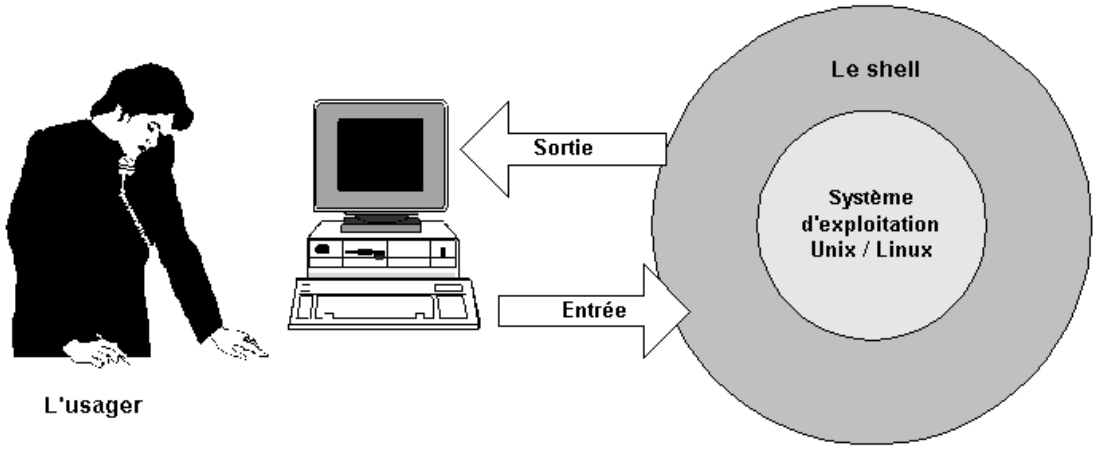
Dans le cadre de ce cours, nous emploierons les termes shell, interpréteur de commande et console comme synonyme.

Pour connaître la version du bash utilisé par votre distribution Linux vous pouvez faire la commande suivante en console. Notez que vous devez respecter la casse des commandes dans l'univers Linux…

```
echo $BASH_VERSION
```

En somme, l'interpréteur de commande est une interface textuelle. C'est à dire qu'elle n'accepte que des commandes en format texte et ne présente que des sorties dans ce format. Par comparaison, les interfaces graphiques (GUI) présentent des icônes, des fenêtres, des graphiques et acceptent les entrées à partir d'une souris. [12]

[11] Traduction libre de : Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page *ix*.
[12] Idem page 2.



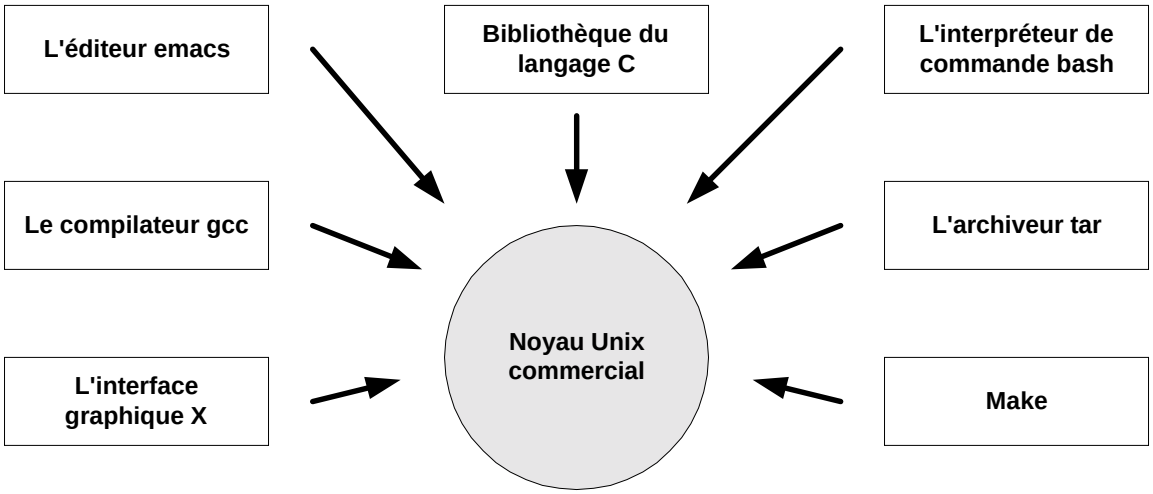
L'usager ne communique pas directement avec le système d'exploitation mais avec l'interface qui elle, interprète ses commandes en un format compréhensible pour le système d'exploitation.

11. Historique

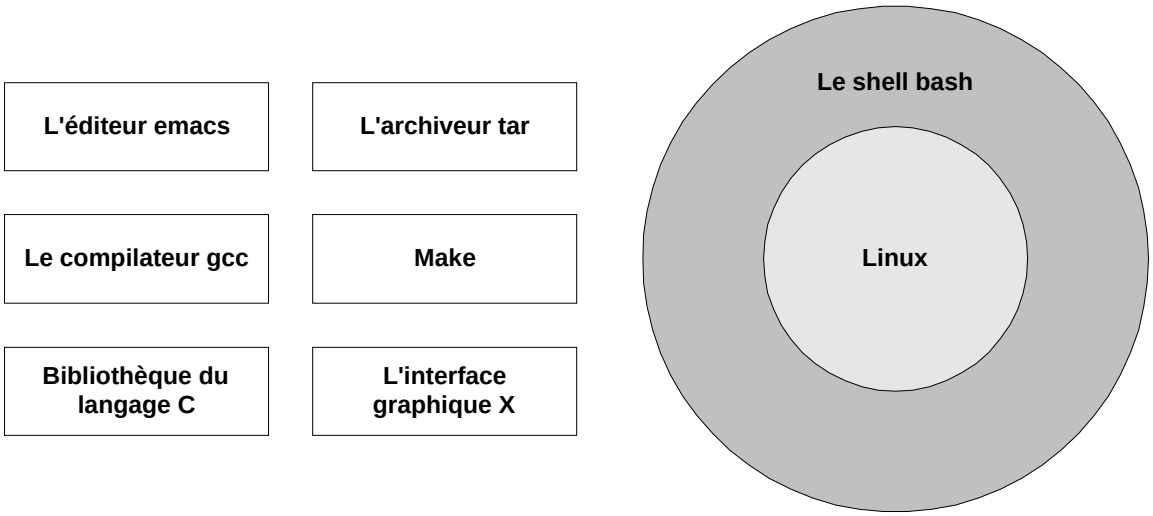
Le shell Bash, écrit par Brian Fox, fait partie du projet GNU (acronyme récursif de GNU's Not Unix) démarré en janvier 1984 par Richard Stallman. Le projet GNU [13] avait pour but de créer un système complet et compatible UNIX pouvant être distribué librement. Son financement a nécessité la création de la FSF (Free Software Foundation). En 1991 le noyau Linux de Linus Torvald a été inséré c'est pourquoi on appelle le tout GNU / Linux.

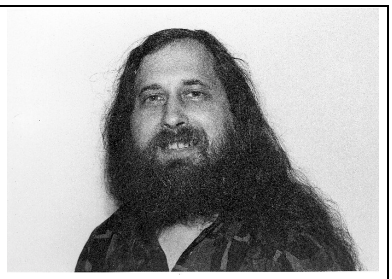
[13] Voir : http://www.fsf.org/home.fr.html

Projet GNU entre 1984 et 1991



Projet GNU / Linux à partir de 1991



Linus Torvald Créateur du noyau Linux. Il a étudié en Finlande.		**Richard Stallman** Fondateur du projet GNU. Créateur d'Emacs et de gcc. Il a étudié au MIT.	

12. Concepts de base

Voici un rappel de quelques notions nécessaires avant de commencer.

12.1. Le répertoire maison (home directory)

Généralement, l'usager se connecte sous Linux dans un certain répertoire. C'est son répertoire maison. Prenons par exemple l'usager `NomUsager`. Celui-ci se connectera sans doute dans le répertoire maison suivant :

```
/home/NomUsager
```

Une façon simple et rapide de retourner dans son répertoire maison est d'utiliser le tilde (~) qui représente le chemin absolu du répertoire maison général. Ainsi, le répertoire suivant est équivalent au précédent.

```
~/NomUsager
```

L'utilisation du tilde est intéressante. Certains systèmes UNIX utilisent le répertoire `/users` comme répertoire maison.

12.2. Répertoire de travail

Le répertoire de travail est le répertoire sur lequel porte nos commandes. On peut le modifier avec la commande cd. L'adressage relatif et l'adressage absolu sont utilisable.

12.3. Les commentaires dans les scripts

Les commentaires sont toujours précédés du dièse (#).

Voici les commentaires qui doivent se retrouver à l'entête des fichiers script :

```
#************************************************************
#   Fichier        : Nom du fichier script
#   Projet         : Nom du TP
#   Auteur(s)      : Votre Nom [ les autres noms ]
#   Groupe         : Identifiant du groupe
#   Cours          : Systèmes d'exploitation
#   École          : Le nom de votre établissement d'enseignement
#   Session        : Session et année
#   Notes          : [ Explications supplémentaires ]
#************************************************************
```

Voici les commentaires qui doivent se retrouver à l'entête des fonctions.

```
#************************************************************
#   Fonction       : Nom de la fonction
#   Objectif       : Objectif de la fonction
#   Notes          : [ Explications supplémentaires ]
#************************************************************
```

12.4. Rappel sur les substitutions de caractères

Vous pouvez utiliser les substitutions suivantes : [14]

Variables	Explication
?	Substitution d'un seul caractère
*	Substitution d'une série de caractères
[liste]	Substitution de tous les caractères de liste
[!liste]	Substitution de tous les caractères ne faisant pas parti de liste.

12.5. Caractères spéciaux

Voici les caractères spéciaux utilisables dans les scripts. [15]

Caractère	Explication
~	Répertoire maison
#	Commentaire
$	Expression d'une variable
&	Background job
*	Substitution de caractères
(	Démarrer un sous-shell
)	Terminer un sous-shell
\	Forcer l'affichage du caractère spécial
|	Pipe (dans le sens des commandes…)
[	Début d'une substitution de caractères
]	Fin d'une substitution de caractères
{	Début d'une fonction
}	Fin d'une fonction
;	Séparateur de commandes
' '	Simple guillemets
" "	Double guillemets
<	Redirection d'entrée
>	Redirection de sortie
/	Séparateur de répertoire
?	Substitution d'un seul caractère
!	Négation

12.6. Commandes de contrôles

Les commandes de contrôles diffèrent d'un système à l'autre. Elles servent généralement à envoyer un signal à propos d'un script en cours. Les commandes de contrôles suivantes peuvent-être utilisées : [16]

Commandes de contrôles	Nom	Explication
CRTL-c	intr	Arrête la commande en cours
CRTL-d	eof	Fin de l'entrée
CRTL-\	quit	Arrête la commande en cours
CRTL-s	stop	Arrête l'affichage à l'écran
CRTL-q	start	Redémarre l'affichage
DEL ou CRTL-?	erase	Efface le dernier caractère
CRTL-u	kill	Efface la ligne de commande
CRTL-z	susp	Suspend la commande en cours

[14] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 11.
[15] Idem page 21.
[16] Idem page 25.

La plus populaire est sans doute CRTL-c qui termine le script en cours.

13. L'environnement

Un environnement est une collection de concept qui exprime le fait qu'un système informatique, ou tout autre outil, est créé pour être compréhensible, cohérent et ergonomique. Ainsi, nous pouvons personnaliser l'environnement du Shell. [17]

13.1. Les fichiers

Lors de votre entré sur le système, le Shell récupère ses informations d'environnement de trois fichiers principaux. Ces trois fichiers sont: `.bash_profile`, `.bash_logout` et `.bashrc`. Le premier contient le profile de l'usager. Le second contient des commandes de fermeture du Shell. Le troisième contient l'environnement du sous Shell. L'importance de ces fichiers est relative à la distribution de Linux que vous avez. Dans tous les cas, lorsque le fichier `.bashrc` est édité, les changements se répercutent sur votre compte. Pour forcer le système d'exploitation à récupéré l'information d'environnement récemment modifié du fichier `.bashrc` vous devez faire la commande suivante :

```
source .bashrc
```

Votre environnement sera alors immédiatement mis à jour.

13.2. Les alias

Un alias est un raccourci pratique pour une commande complexe. Les alias se trouvent dans le fichier `.bashrc` ou `.bash_profile`. L'utilisation d'un alias est très simple :

```
alias nomAlias="commande"
```

Ex: `alias`
Ex: `alias datej="date '+Nous sommes %A le %e %B %Y' "`

Ajoutez le dernier exemple dans votre fichier `.bashrc`.

13.3. Les variables d'environnement

Voici une liste très concise des variables du Shell. Elles sont toujours écrites en majuscules.

- HISTFILE : Le nom de la commande du fichier historique.
- HISTFILESIZE : Le nombre maximum de ligne conservé dans le fichier historique.
- HISTSIZE : Le nombre de lignes conservé dans le fichier de commandes.
- BASH : Contient le répertoire utilisé pour invoquer cette instance.
- BASH_ENV : Le fichier d'environnement lorsque le Shell est invoqué.
- BASH_VERSION : La version du bash utilisé.
- PATH : Les répertoires de base lors d'une recherche d'une commande.
- PS1 : Le prompt primaire.
- PS2 : Le prompt secondaire.
- SHELL : Le répertoire du shell.

Ex: `echo $BASH_VERSION`

Cette commande affiche la version du Bash utilisée.

[17] Debra CAMERON & Bill ROSENBLATT & Eric RAYMOND, *GNU Emacs*, O'Reilly, USA, 1996, page 57.

13.4. Les variables du prompt

Le prompt accompagne l'utilisateur lorsqu'il est en mode console. Il contient certaines informations intéressantes. Il y a deux variables importantes qui déterminent l'apparence du prompt: PS1 et PS2. La première est le prompt primaire et la seconde le prompt secondaire.

Variables	Explication
\a	Caractère ASCII bell (007)
\d	La date
\e	Caractère d'échappement (033)
\H	Le Hostname
\h	Le Hostname jusqu'au premier "."
\n	Passer une ligne
\s	Nom du Shell
\T	L'heure 12 heure HH:MM:SS
\t	L'heure 24 heure HH:MM:SS
\@	L'heure 12 heure am, pm
\u	Nom de l'usager
\v	La version du bash
\V	La version complète du bash
\w	Le répertoire courant
\W	Le nom de base du répertoire courant
\#	Le numéro de la commande courante
\!	Le numéro historique de la commande
\$	Imprime # pour le root ou $ autrement
\nnn	Le code octal du caractère
\\	Imprime un backslash
\[	Débute une séquence de non impression
\]	termine une séquence de non impression

Ex: `echo $PS1`
Ex: `PS1="\s FXG -->"`

Modifions ensemble le prompt en changeant les valeurs de PS1 dans le fichier `.bashrc`. Insérez la ligne suivante.

```
PS1="\u dans \W -->"
```

13.5. Le path

Le PATH est la série de répertoire que le Shell parcourt lorsqu'il reçoit une instruction. Chacun des répertoires du path est séparé par des deux points (`:`).

Ex: `echo $PATH`
Ex: `PATH=$PATH":/home/NomUsager/bin"`

Modifions ensemble le path inscrit dans le fichier `.bashrc` pour qu'il puisse exécuter les scripts dans le répertoire `/bin` de l'usager. Pour ce faire, créez d'abord le répertoire `/bin`. En console, faites la commandes suivantes dans votre répertoire `/home/NomUsager` :

```
mkdir bin
```

Puis, ouvrez votre fichier `.bashrc` avec Emacs et ajoutez-y la ligne suivante à la fin :

```
PATH=$PATH":/home/NomUsager/bin"
```

Maintenant, tous les script situé dans le répertoire `/home/NomUsager/bin` pourront être exécutés.

14. Programmation de base du shell

Abordons maintenant la programmation de base des scripts avec le shell Bash de Linux.

14.1. Les scripts

Un script est un fichier exécutable renfermant plusieurs commandes. Dans le monde Linux, un fichier exécutable est déterminer par ses attributs et non par son extension comme dans le monde DOS. Ainsi, parmi les fichiers suivants :

```
- rw-      r-- r--   1 csimard  users  19289 Aug 20 10:54 prog1.exe
- rw-      r-- r--   1 csimard  users  19289 Aug 21 11:54 prog1
- rwx      r-x r-x   1 csimard  users  19289 Aug 22 11:56 prog1.txt
- rwx      r-x r-x   1 csimard  users  19289 Aug 23 07:56 prog1
```

Seuls les deux derniers sont exécutables.

Une façon rapide de changer les attributs d'un fichier normal en fichier exécutable est d'utiliser la commande suivante :

```
chmod +x NomFichier
```

Elle ajoute les attributs exécutables à tous les types d'usagers pour ce fichier. La forme d'un script dans un fichier est toujours la suivante :

```
Fonction 1
Fonction 2
Fonction n
Le code principal
```

14.2. Les fonctions

La déclaration d'une fonction se trouve toujours avant le script principal. L'avantage des fonctions est qu'elles ont leurs propres variables positionnelles et locales au besoin. Les fonctions prennent les formes suivantes :

```
function NomFonction
{
   # Code ici
}
```

ou

```
nomFonction ()
{
    # Code ici
}
```

Pour envoyer des paramètres du script principal à la fonction :

```
NomFonction paramètre1 paramètre2
```

Notez que l'utilisation des paramètres dans la fonction se fait par l'entremise des variables positionnelles de la fonction. Ainsi, le `paramètre1` sera la variable positionnelle 1 de la fonction…

14.3. Les variables

Une variable est un contenant nommé dont la valeur contenue peut être modifiée. Dans le monde du shell Bash celles-ci sont des chaînes de caractères. Leur valeur peut être obtenue

en précédant le nom de la variable par le signe $. En somme, `$variable` renvoi le contenu de `variable`. En fait, la syntaxe stricte demanderait que l'on écrive sous la forme `${variable}`. Vous devez utiliser la syntaxe stricte pour faire afficher le contenu d'une variable système. Nous verrons la notion de syntaxe plus en détail ultérieurement.

14.3.1. Initialisation des variables

Il y a différentes façons d'initialiser une variable. La première par une valeur quelconque :

```
variable="valeur"
```

La seconde par la sortie d'une commande :

```
variable="$(commande)"
```

La troisième par une demande à l'usager :

```
read variable
```

Nous verrons cette dernière option plus en détail ultérieurement.

14.3.2. Les variables d'environnement

Les variables d'environnement sont accessibles depuis tout script.

Ex: `${UID}`

Renvoi le numéro de l'usager du système. Le root porte le numéro 0…

Ex: `${USER}`

Renvoi le nom de l'usager du système.

14.3.3. Les variables positionnelles

Lors de l'invocation d'un script, vous pouvez lui passer des arguments. Ces arguments envoyés au script deviendront des variables positionnelles. Elles sont numérotées de 1 à 9. La variable positionnelle 0 est réservée pour contenir le nom du script. Faisons ensemble le script `premier`. Allez dans votre répertoire `/home/NomUsager/bin` et inscrivez la ligne suivante dans un fichier nommé `premier`. Le script doit être dans un répertoire mentionné de votre `PATH` sans quoi il ne fonctionnera pas…

```
echo "Bonjour $1 $2 $3 !"
```

Puis, modifiez ses attributs pour le rendre exécutable.

```
chmod +x premier
```

ou

```
chmod 700 premier
```

Pour le démarrer, on tape `premier`. Mais si l'on tape `premier mon cher ami`, alors `mon` est la variable positionnelle 1, `cher` est la variable positionnelle 2 et `ami` est la variable positionnelle 3. En tapant `premier`, vous obtiendrez le résultat suivant :

```
Bonjour !
```

Par contre, en tapant `premier mon cher ami`, alors vous obtiendrez le résultat suivant :

```
Bonjour mon cher ami !
```

Il y a différentes façons d'afficher le contenu des variables positionnelles. Vous pouvez les mentionner par leurs numéros respectifs ou par la variable (@). Cette dernière inclus toutes les variables positionnelles sauf la variable positionnelle `0` (le nom du script).

Modifions un peu notre script et ajoutons quelques lignes :

```
clear
message="La vie est belle !"
echo "Bonjour $1 $2 $3 !"
echo "$message"
echo "$0 $1 $2 $3"
echo "$@"
echo "$# arguments"
echo "${UID}"
```

Devrait afficher ceci :

```
Bonjour mon cher ami !
La vie est belle !
/home/csimard/bin/premier mon cher ami
mon cher ami
3 arguments
501
```

14.3.4. Les variables locales et globales

Toutes les variables d'un script sont globales et sont accessibles à l'intérieur des fonctions. Pour déclarer une variable locale, accessible uniquement par la fonction, il faut précéder son nom du mot clé `local`.

Créons maintenant le script `deuxieme.` Il comporte des fonctions et utilise des variables locales et globales. Notez que l'entête du script et les entêtes des fonctions ont été supprimé afin d'alléger le texte… N'oubliez pas de les insérer dans votre code.

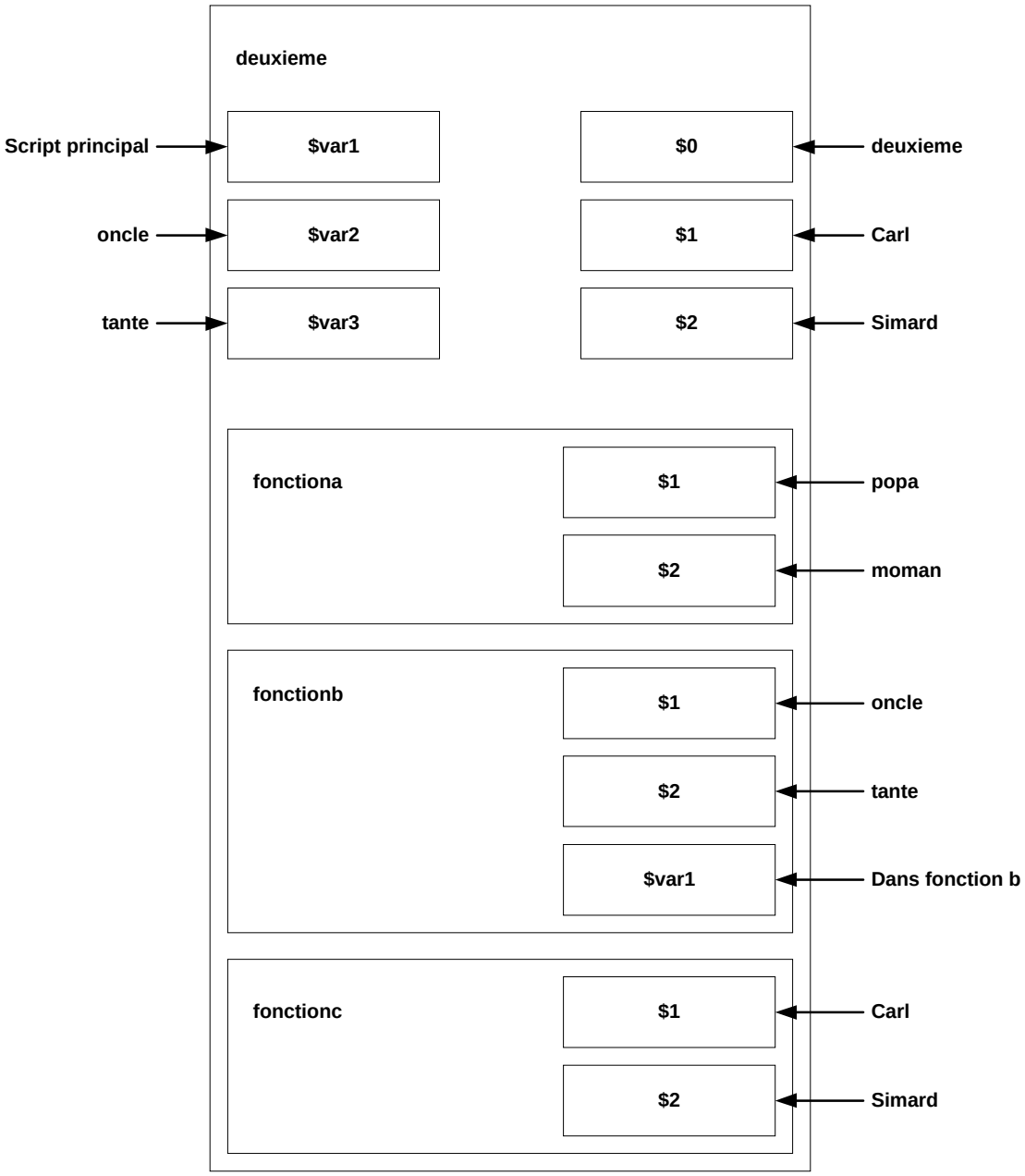
```
function fonctiona
{
	echo "Fonction a : $@"
	var1="Dans fonction a"
	echo "var1 : $var1"
}

function fonctionb
{
	echo "Fonction b : $1 $2"
	local var1="Dans fonction b"
	echo "var1 : $var1"
}


function fonctionc
{
	echo "Fonction c : $@"
}

clear
var1="Script principal"
var2="oncle"
var3="tante"
echo "Nom du script $0"
echo "Nombre d'arguments $#"
```

```
echo "Arguments du script $@"
echo -e "Variables debut : $var1 $var2 $var3 \n"

# Appel de la fonction a
fonctiona popa moman
echo -e "Var1 : $var1 \n"

# Appel de la fonction b
fonctionb $var2 $var3
echo -e "Var1 : $var1 \n"

#Appel de la fonction c
fonctionc $1 $2

echo -e "\nFINI"
```

En lançant le script `deuxieme` avec les arguments `Carl Simard`, il devrait afficher les lignes suivantes :

```
Nom du script deuxieme
Nombre d'arguments 2
Arguments du script Carl Simard
Variables debut : Script principal oncle tante

Fonction a : popa moman
var1 : Dans fonction a
Var1 : Dans fonction a

Fonction b : oncle tante
var1 : Dans fonction b
Var1 : Dans fonction a

Fonction c : Carl Simard

FINI
```

Voici une représentation graphique des variables du script `deuxieme` et des variables de ses différentes fonctions.



14.4. Opérations sur les chaînes

L'idée de base des opérations sur les chaînes est de vérifier l'existence d'une variable et de la substituer par une autre plus appropriée. Voici un tableau des opérations possibles. [18]

Opérations	Substitution réalisée
`${variable:-mot}`	Si la variable existe et n'est pas nulle, elle retourne sa valeur; sinon elle retourne le mot. Cette opération est utile pour retourner une variable par défaut.
`${variable:=mot}`	Si la variable existe et n'est pas nulle, elle retourne sa valeur; sinon, elle initialise la variable par le mot et retourne sa valeur. Notez que les variables positionnelles ne peuvent être initialisées de cette façon. Cette opération est utile pour initialiser une variable par défaut.
`${variable:?message}`	Si la variable existe et n'est pas nulle, elle retourne sa valeur; sinon elle affiche variable suivit de message et termine le script en cours. Si le message est omis, la commande affiche `parameter null or not set`.
`${variable:+mot}`	Si la variable existe et n'est pas nulle, elle retourne le mot; sinon, elle retourne null. Cette opération sert à tester l'existence d'une variable.

14.5. Les schémas

Les schémas permettent d'ajouter un peu d'esthétisme à vos scripts en voilant des parties de chaînes qui ne vous intéresse pas. Les opérations sur les schémas peuvent être intéressant lorsque vous désirer vous débarrasser du répertoire précédant le nom d'un script. Notez que vous pouvez utiliser les substitutions *, ? et [ ]. [19]

Opération sur les schémas	Explication
`${variable#schéma}`	Si le schéma ressemble au début de la variable, alors il cache la plus petite partie et retourne le reste.
`${variable##schéma}`	Si le schéma ressemble au début de la variable, alors il cache la plus grande partie et retourne le reste.
`${variable%schéma}`	Si le schéma ressemble à la fin de la variable, alors il cache la plus petite partie et retourne le reste.
`${variable%%schéma}`	Si le schéma ressemble à la fin de la variable, alors il cache la plus grande partie et retourne le reste.

Expression	Résultat
`${path##/*/}`	`long.nom.fichier`
`${path#/*/}`	`carl/notes/long.nom.fichier`
`$path`	`/home/carl/notes/long.nom.fichier`
`${path%.*}`	`/home/carl/notes/long.nom`
`${path%%.*}`	`/home/carl/notes/long`

[18] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 94.
[19] Idem page 99 – 100.

14.6. Les substitutions de commandes

La sortie d'une commande peut être utilisée comme étant la valeur d'une variable. La syntaxe d'une substitution de commande est la suivante[20] :

```
$(commande)
```

Tableau d'exemples :

Commandes	Équivalence ou explication
`$(pwd)`	`$PWD`
`$(ls $HOME)`	Liste les fichiers du répertoire HOME
`$(ls $(pwd))`	Liste les fichiers dans le répertoire courant

Pour mettre en pratique ce que nous venons d'apprendre, faisons ensemble le script `troisieme`. Il met en pratique des opérations sur les chaînes, des initialisations de variables par des sorties de commandes et un schéma.

```
clear
liste1=$(ls -m /)
liste2=$(ls / | sort -r | tr "\012" " ")
echo "Parametres : $0 $1 $2 $3"
echo "Variable 1 : ${1:-manquante}"
echo "Variable 2 : ${2:-manquante}"
echo "Variable 3 : ${3:-manquante}"
echo "Nom de la fonction : ${0##/*/}"
echo "Voici le contenu du repertoire racine : "
echo "$liste1"
echo "Voici le contenu du repertoire racine : "
echo "$liste2"
```

En lançant `troisieme Carl Simard`, vous devriez obtenir un affichage tel que celui-ci :

```
Parametres : /home/csimard/bin/troisieme Carl Simard
Variable 1 : Carl
Variable 2 : Simard
Variable 3 : manquante
Nom de la fonction : troisieme
Voici le contenu du repertoire racine :
bin, boot, dev, etc, home, lib, lost+found, mnt, opt, proc, root, sbin, tmp, usr, var
Voici le contenu du repertoire racine :
var usr tmp sbin root proc opt mnt lost+found lib home etc dev boot bin
```

La sortie de commande qui initialise liste2 pourrait se lire comme suit :

- Liste le répertoire racine (`/`)
- Met cette liste en ordre alphabétique inverse
- Change le caractère de fin de ligne par un espace

Une opération sur les chaînes permet de vérifier la présence d'une valeur et, le cas échéant de le faire savoir. De plus, le schéma (`/*/`) est utilisé pour cacher tout le répertoire du script.

15. Les structures de contrôles

On ne pourrait décemment programmer sans structures de contrôles. Le shell Bash de Linux en possède quelques-unes. Voici la liste :

[20] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 123.

15.1. Le if

Le if test une sortie de commande. Le if / elif / else complet se présente comme suit :

```
if condition
then
    #Code
elif condition
then
    #Code
else
    #Code
fi
```

Le if / else se présente comme suit :

```
if condition
then
    #Code
else
    #Code
fi
```

Le if tout court se présente comme suit :

```
if condition
then
    #Code
fi
```

15.2. La sortie de fonction et de programme

À la rencontre des mots clef `return` ou `exit`, une fonction retourne une valeur ou un programme se termine sans poser d'autres questions. [21]

```
function dire
{
    if condition
    then
       #Code
       return $variable #retourne une valeur résultante
    fi
{

function dire2
{
    if condition
    then
       #Code
       exit #termine le script
    fi
{
```

15.3. Test sur les variables et les fichiers

La seule chose que l'on peut tester avec une opération conditionnelle est la sortie d'une fonction. Heureusement, il existe différentes façons de réaliser un test, `test` étant en lui-même une commande Linux, avec les constructeurs `[ ]`. Ainsi la condition se présente sous la forme de [22] :

```
[ condition ]
```

Les opérateurs admis sont les suivants :

[21] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 115 et 116.
[22] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 117 et 118.

Opérateurs	Vrai si
`chaîne1 = chaîne2`	chaîne1 est égal à chaîne2 (notez qu'il n'y a qu'un seul "=")
`chaîne1 != chaîne2`	chaîne1 n'est pas égal à chaîne2
`chaîne1 < chaîne2`	chaîne1 est plus petit que chaîne2
`chaîne1 > chaîne2`	chaîne1 est plus grand que chaîne2
`-n chaîne1`	chaîne1 n'est pas null (a une longueur plus grande que zéro)
`-z chaîne1`	chaîne1 est null (a une longueur égale à zéro)

Ex :

```
if [ -n "$valeur" ]
then
    # Code
else
    #Code
fi
```

De plus, nous pouvons effectuer des tests sur des fichiers. Pour ce faire, nous avons droit aux opérateurs suivant[23] :

Opérateurs	Vrai si
`-d fichier`	Le fichier existe et c'est un répertoire (directory)
`-e fichier`	Le fichier existe
`-f fichier`	Le fichier existe et c'est un fichier régulier
`-r fichier`	Vous avez la permission en lecture (read) sur ce fichier
`-s fichier`	Le fichier existe et n'est pas vide
`-w fichier`	Vous avez la permission en écriture (write) sur ce fichier
`-x fichier`	Vous avez la permission en exécution (execute) sur ce fichier
`-O fichier`	Vous êtes propriétaire de ce fichier
`-G fichier`	Le groupe ID du fichier est le même que le vôtre
`fichier1 –nt fichier2`	Fichier1 est plus récent que fichier2
`fichier1 –ot fichier2`	Fichier1 est plus vieux que fichier2

Finalement, nous pouvons combiner différentes conditions avec les opérateurs logiques suivant :

Opérateurs	Fonctions
`&&`	Et logique
`||`	Ou logique
`!`	Négation

Ex :

```
if [ condition ] && [ ! condition ]
then
    # Code ici
fi
```

[23] Idem page 122.

Faisons ensemble le script `quatrieme`.

```
clear
echo -e "Script testant différentes conditions"
var1="Bonjour"
var2="Allo"
var3="Blabla"
repertoire="/home/csimard/bin"
fichier="quatrieme"
# Comparaison de deux variables
if [ "var1" = "var2" ]
then
    echo -e "Bonjour = Allo"
else
    echo -e "Bonjour != Allo"
fi
# Vérification de l'initialisation d'une chaîne
if [ -z "$var3" ]
then
    echo -e "Var3 n'est pas initialisé"
else
    echo -e "Var3 est initialisé avec : $var3"
fi
# La première condition est une sortie de commande
# Notez l'envoi des messages d'erreurs à la poubelle
if cd ${repertoire} 2> /dev/null
then
    echo -e "J'ai pu me rendre au répertoire $repertoire"
    # Le fichier est-il un répertoire
    if [ -d "$fichier" ]
    then
        echo -e "Le fichier $fichier est un répertoire"
    # Vérification de l'existence ET de l'exécutabilité du fichier
    elif [ -e "$fichier" ] && [ -x "$fichier" ]
    then
        echo -e "Le fichier $fichier existe et peut s'exécuter"
    else
        echo -e "Je n'ai rien d'autre à dire"
    fi
else
    echo -e "Je n'ai pu me rendre au répertoire $repertoire"
fi
```

Le résultat de son affichage est un peu moins long.

```
Script testant différentes conditions
Bonjour != Allo
Var3 est initialisé avec : Blabla
J'ai pu me rendre au répertoire /home/csimard/bin
Le fichier quatrieme existe et peut s'exécuter
```

15.4. Le for

La boucle for a la particularité de ne pas utiliser de chiffre mais la présence d'une liste de string dans une liste et d'initialiser un `nomVariable`, à chaque tour, avec la première chaîne restante de la liste[24]...

```
for nomVariable in liste
do
    # Code pouvant utiliser $nomVariable
done
```

Faisons ensemble le script `cinquieme`.

```
clear
echo -e "Petit script démontrant l'utilisation de la boucle for \n"
liste="Popa Moman Ti-mé Ti-Coune"
```

[24] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 126 à 132.

```
echo -e "La liste est : $liste \n"
# La boucle for prend la première chaîne de la liste
# elle la place dans la variable nom
# puis effectue une opération sur elle
# dans ce cas-ci un "echo" et ce,
# tant qu'il y a des chaînes dans la liste
echo -e "Maintenant, présentons-la en une seule colonne : \n"
for nom in ${liste}
do
    echo "$nom"
done
echo -e "\nOpération terminée \n"
```

Son affichage donne ceci :

```
Petit script démontrant l'utilisation de la boucle for

La liste est : Popa Moman Ti-mé Ti-Coune

Maintenant, présentons-la en une seule colonne :

Popa
Moman
Ti-mé
Ti-Coune

Opération terminée
```

15.5. Le case

Le case se présente sous la forme suivante :

```
case expression in
    patern1 )
            # Code ;;
    patern2 )
            # Code ;;
            * )
            # Code par défaut ;;
esac
```

Nous verrons un exemple complet un peu plus loin.

15.6. La boucle while

Les boucles while et until sont très pratiques :

La boucle while se termine par une condition fausse :

```
while condition
do
    # Code
done
```

Faisons ensemble le script `sixieme`. Il permet d'afficher les répertoires du PATH.

```
clear
path=$PATH:
echo -e "Voici les répertoires du PATH en liste \n"
echo -e "$path \n"
echo -e "Voici les répertoires du PATH en colonne \n"
while [ $path ]
do
    # On enlève la queue de la liste par un schéma
    echo "${path%%:*}"
    # On réinitialise la variable path
    # en supprimant sa tête par un schéma
    path=${path#*:}
```

```
done
echo -e "\nOpération terminée \n"
```

Le résultat du script `sixieme` est le suivant :

```
Voici les répertoires du PATH en liste

/usr/X11R6/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/games:/home/csimard/bi
n:/home/csimard/bin:/usr/X11R6/bin:/usr/games:/home/csimard/bin:/usr/X11R6/bin:/usr/ga
mes:/home/csimard/bin:/usr/X11R6/bin:/usr/games:/home/csimard/bin:

Voici les répertoires du PATH en colonne

/usr/X11R6/bin
/usr/local/bin
/bin
/usr/bin
/usr/X11R6/bin
/usr/games
/home/csimard/bin
/home/csimard/bin
/usr/X11R6/bin
/usr/games
/home/csimard/bin
/usr/X11R6/bin
/usr/games
/home/csimard/bin
/usr/X11R6/bin
/usr/games
/home/csimard/bin

Opération terminée
```

15.7. La boucle until

La boucle until se termine par une condition vraie :

```
until condition
do
    # Code
done
```

Voyons ensemble son application dans le script `septieme`.

```
clear
choix="n"
until [ $choix = "o" ] || [ $choix = "O" ]
do
    echo -e "Désirez-vous sortir de la boucle ? (oO/nN) : \c"
    read choix
done
```

Assurez-vous d'expliquer la façon de sortir à l'usager. Sans quoi, il risque fort de rester coincé. Voici l'affichage de `septieme`.

```
Désirez-vous sortir de la boucle ? (oO/nN) : n
Désirez-vous sortir de la boucle ? (oO/nN) : N
Désirez-vous sortir de la boucle ? (oO/nN) : o
```

La condition peut aussi être une commande

```
until commande
do
    # Code
done
```

Cette commande sera exécutée tant qu'elle ne réussira pas…

16. Les entrées sorties

Nous avons vu, jusqu'à maintenant, des scripts avec très peu d'interactivité. Pour faire agir l'usager d'un système vous devez pouvoir lui poser des questions et capter ses réponses. De plus, vous devez pouvoir sauvegarder des données dans un fichier.

16.1. La redirection

Vous pouvez utiliser toutes les redirections connues jusqu'ici. Ainsi, vous pouvez créer un fichier de façon interactive. Notez que `commande 2> /dev/null` redirige les messages d'erreur vers la poubelle[25]…

16.2. Les commandes d'entrée/sortie

Nous les avons vu tout au long des scripts précédents.

16.2.1. echo

Permet d'afficher quelque chose à l'écran. L'option `-e` "enable" permet d'utiliser les caractères spéciaux suivant :[26]

Caractères spéciaux	Explication
`\n`	Saut de ligne
`\\ ou \b`	Retour arrière
`\t`	Tabulation
`\c`	Fin de la sortie et annulation du saut de ligne
`\Octal`	Valeur octale d'un caractère spécial
`\a`	Fait sonner la cloche

16.2.2. read et unset

La commande `read` permet d'initialiser une variable par une demande à l'usager. Notez que la commande `unset` permet une remise à null de la variable.[27]

Ex: `read variable1`
Ex: `unset variable1`

16.2.3. pause

Pour améliorer le rendu de vos scripts, la commande `sleep n` peut être utilisée pour marquer une pause de n secondes.

17. La copie de votre fichier script sur disquette

Votre script est terminé et fonctionnel ? Maintenant, il ne vous reste plus qu'à en faire une copie de sauvegarde. Avant de pouvoir copier un fichier script sur votre disquette, vous devez monter son système de fichier.

```
mount /mnt/floppy
```

Puis, vous effectuer la copie de vos fichiers. En supposant que vous vous trouvez dans le répertoire qui contient le script, la commande devrait ressembler à celle-ci :

```
cp NomScript /mnt/floppy
```

N'oubliez pas de démonter votre système de fichiers par la commande suivante :

```
umount /mnt/floppy
```

[25] Cameron NEWHAM & Bill ROSENBLATT, *Learning the Bash Shell*, O'Reilly, USA, 1998, page 164.
[26] Idem page 170.
[27] Idem page 174.

18. Exercices supplémentaires

Exercice 1 : Le menu

```
#************************************************************
#   Fichier    : Menu
#   Projet     : Exercices supplémentaires
#   Auteur(s)  : Carl Simard
#   Groupe     :
#   Cours      : Systèmes d'exploitation
#   École      :
#   Session    :
#   Notes      : Permet de vous pratiquer un peu :-)
#************************************************************

#************************************************************
#   Fonction   : sortie
#   Objectif   : Termine le script
#   Notes      :
#************************************************************
function sortie
{
   clear
   echo "Bonne journee $1 $2"
   sleep 2
   clear
   exit
}

#************************************************************
#   Fonction   : choixun
#   Objectif   : Permet d'afficher Allo à l'usager
#   Notes      :
#************************************************************
function choixun
{
   clear
   echo "Allo"
   sleep 2
}

#************************************************************
#   Fonction   : choixdeux
#   Objectif   : Permet d'afficher Bonjour à l'usager
#   Notes      :
#************************************************************
function choixdeux
{
   clear
   echo "Bonjour"
   sleep 2
}
```

```
#************************************************************
#   Fonction   :
#   Objectif   : Boucle du menu principal
#   Notes      :
#************************************************************
while :
do
   clear
   echo -e "MENU PRINCIPAL \n"
   echo -e "Choix pour $1 $2 \n"
   echo -e "\t Option \t description \n"
   echo -e "\t 0 \t\t Sortie"
   echo -e "\t 1 \t\t Le allo"
   echo -e "\t 2 \t\t Le bonjour \n"
   echo -e "Votre choix : \c "
   read choix

   case $choix in
      0) sortie $1 $2 ;;
      1) choixun ;;
      2) choixdeux ;;
   esac

done
```

Exercice 2 : Le questionnaire

```
#*************************************************************
#   Fichier    : questionnaire
#   Projet     : Exercices supplémentaires
#   Auteur(s)  : Carl Simard
#   Groupe     :
#   Cours      : Systèmes d'exploitation
#   École             :
#   Session    :
#   Notes             : Permet de vous pratiquer un peu :-)
#*************************************************************

#*************************************************************
#   Fonction   : diredate
#   Objectif   : Affiche la date de naissance de l'usager
#   Notes             :
#*************************************************************
function diredate
{
   echo -e "Fonction naissance"
   echo -e "Vous êtes né le $1 $2 $3 \n"
}

#*************************************************************
#   Fonction   : direage
#   Objectif   : Affiche l'âge de l'usager
#   Notes             :
#*************************************************************
function direage
{
   echo -e "Fonction âge"
   echo -e "Vous avez $1 ans \n"
}

#*************************************************************
#   Fonction   : prenom
#   Objectif   : Affiche le prénom de l'usager
#   Notes             :
#*************************************************************
function prenom
{
  echo -e "Fonction prénom"
  echo -e "Votre prénom est $1 \n"

}

#*************************************************************
#   Fonction   : nom
#   Objectif   : Affiche le nom de l'usager
#   Notes             :
#*************************************************************
function nom
{
   echo -e "Fonction nom"
   echo -e "Votre nom est $1 \n"
}

#*************************************************************
#   Fonction   :
#   Objectif   : Script principal du questionnaire
#   Notes             :
#*************************************************************
clear
echo -e "Donnez votre âge : \c"
read age
echo -e "Donnez votre date de naissance : \c"
read naissance
echo -e "\n"
direage $age
diredate $naissance
```

```
prenom $1
nom $2
echo -e "$1 $2 ne le $naissance \n"
echo "Fin du questionnaire"
```