Support de formation

Administration Système Linux1

Préparation à l'examen LPI 101

version 0.2 – française v 0.1

Table des matières

INTRODUCTION **6**
À propos de la version française 6
Remerciements 6
Financement 6
Marques déposées 6
Historique 6
Dramatis Personae 7
Objectifs 7
Plan de formation 7
Pré-requis et profil des élèves 7
Le programme de certification LPI 8
À l'attention du formateur 8
Pas de garantie 8
Sources d'information 8
Conventions de notation 8
LES OBJECTIFS DÉTAILLÉS POUR L'EXAMEN DE CERTIFICATION LPI 101 ..... **10**
Sujet 101 : Architecture système 10
Sujet 102 : Installation de Linux et gestion de paquetages 12
Sujet 103 : Commandes GNU et Unix 14
Sujet 104 : Disques, Système de fichiers Linux , Arborescence de fichiers standard 19
INSTALLATION **24**
Le CD d'installation 25
Installations locales 26
Installation réseau 26
Disque de récupération 26
Plan de partitionnement 27
Plusieurs systèmes d'exploitation sur une machine 29
Exercices et résumé 29
CONFIGURATION MATÉRIELLE **32**
Allocation de ressources 33
Cartes d'extension 34
USB 34
SCSI 35
Cartes réseau 37
Configuration des modems 38
Configuration des imprimantes 42
Cartes son 43
Exercices et résumé 44
GESTION DES DISQUES **47**
Les disques 47
Partitions 47
Les programmes de partitionnement 48
Chargeurs de démarrage ("Master Boot Record") 49
Gestion des partitions 51
Gestion des quotas 52

Exercices et résumé....52
LE SYSTÈME DE FICHIERS LINUX....56
La structure du système de fichiers....56
Formatage et cohérence des systèmes de fichier....57
Contrôle de l'utilisation des disques....59
Droits d'accès et attributs des fichiers....60
Résumé et exercices....64
LA LIGNE DE COMMANDE....68
Présentation....68
Le shell interactif....68
Variables....69
Entrées, sorties et redirections....70
Méta-caractères et guillemets....71
L'historique des commandes....72
Autres commandes....73
Exercices et résumé....75
GESTION DES FICHIERS....79
Se déplacer dans le système de fichiers....79
Recherche de fichiers et de répertoires....79
Gestion des répertoires....81
Utilisation de cp et mv....81
Liens symboliques et liens physiques....82
touch et dd....83
Résumé et exercices....83
GESTION DES PROCESSUS....87
Visualisation des processus en cours....87
Modification des processus....88
Les processus et le shell....90
Résumé et exercices....91
TRAITEMENT DU TEXTE....94
le couteau suisse cat....94
Petits outils....94
Manipulations du texte....96
Résumé et exercices....98
INSTALLATION DES LOGICIELS....100
Introduction....100
Bibliothèques statiques et partagées....101
Installation à partir des sources....103
RPM : RedHat Package Manager....106
Gestion des paquets Debian....109
La commande alien....114
Résumé et exercices....115
TRAITEMENT AVANCÉ DU TEXTE....119
Les expressions rationnelles....119
La famille grep....120
Travailler avec grep....120
egrep et fgrep....120

sed, l'éditeur en continu....121
Résumé et exercices....122

UTILISATION DE VI....124
Les modes de vi....124
Éléments de texte....124
Insertion de texte....125
Couper / Coller....126
Copier / Coller....126
Recherche et remplacement....126
Annuler et rétablir....127
Lancer une commande du shell....127
Enregistrer et quitter....127
Résumé et exercices....128

LE SERVEUR X....130
Introduction....130
Configuration de X11R6....131
Contrôle des clients X....133
Lancement du serveur X....135
Le gestionnaire d'affichage (display manager)....135
Dépannage les clients X....139
Choix d'un gestionnaire de fenêtres....139
Résumé et exercices....140

Introduction

À propos de la version française

Ce document est une traduction du guide mis à disposition par le centre de formation technique LinuxIT : à l'adresse
Ce cours de formation est disponible dans sa version originale ainsi que dans sa version française en licence GNU FDL. Il est valable pour la version 2 de la certification LPIC-1 mais il reste très utile et parfaitement adaptable pour la préparation de la LPIC-1 dans sa version 3 (depuis avril 2009).

Pour participer à la traduction de ce guide ou des autres guides de formation aux LPI, contactez Éric Deschamps.

Remerciements

Le cours d'origine a été mis à disposition par le centre de formation technique LinuxIT www.linuxit.com. Nous remercions particulièrement Andrew Meredith qui en a eu l'idée le premier. Nous remercions également nos élèves qui nous ont aidé à faire passer les aspects les plus techniques de Linux par leurs nombreuses questions. Cela nous a amené à ajouter des illustrations pour présenter des concepts de façon plus pédagogique. Enfin, un grand merci à Paul McEnery pour ses conseils techniques et pour avoir commencé certains des chapitres les plus difficiles comme ceux qui traitent du serveur X (101), des modems (102) et du noyau Linux (102).
Ce manuel est disponible en ligne sur . Merci aux volontaires de Savannah pour leur support et pour l'hébergement.

Financement

La réalisation de ces documents a été financée en partie par UNDP-APDIP International Open Source Network and International Development Research Centre Canada.

Marques déposées

Historique

- Première version (v 0.0) : octobre 2003, révision par Adrian Thommasset.
- Mis à jour en janvier 2004 après révision par Andrew Meredith.
- Novembre 2004 : ajout d'une partie sur les cartes d'extension dans le chapitre sur la configuration du matériel par Adrian Thommasset.
- Décembre 2004 : ajout de la table des matières et des objectifs visés, par Adrian Thommaset.
- Janvier 2005 : ajout d'un glossaire de termes, commandes et fichiers, visible en fin de chaque chapitre, par Adrian Thommaset.
- Juin 2005 : Adaptation du document à partir des recommandations de SerNet en vue d'un agrément par LPI, par Andrew Meredith. Ajout d'informations par Andrew D Marshall, mise à jour par Adrian

Thomasset. Ajout de la partie sur les outils Debian par Duncan Thomson.

- Août 2005 : ce support pédagogique dans la version anglaise 0.2 a reçu l'agrément LATM (LPI Approved Training Materials – support pédagogique agréé LPI) par SerNet.
- Mai 2008 : début de la traduction française par Éric Deschamps
- Mai 2011 : début de la relecture par Guillaume Laurès

Dramatis Personae

- Adrian Thomasset <adriant<CHEZ>linuxit<POINT>com>
- Andrew Meredith <andrew<CHEZ>anvil<POINT>org>
- Andrew D Marshall <admarshall<CHEZ>gmail<POINT>com>
- Duncan Thomson <thom-ci0<CHEZ>paisley<POINT>ac<POINT>uk>
- Eric Deschamps <erdesc<CHEZ>formation-lpi<POINT>com>
- Marie Sarraute
- Giovanni Dick
- Yoan Issartel <69ivanov<CHEZ>gmail<POINT>com>
- Guillaume Laurès <glaures<CHEZ>laureconseil<POINT>fr>

Objectifs

Le but principal de ce guide est d'apporter des explications, exemples et exercices aux personnes préparant l'examen LPI 101 pour la certification LPI 1 (LPIC-1). Trois sources sont utilisées pour la réalisation de ce guide :

- les objectifs de l'examen LPI 101 ;
- les critères pour les supports de formations agréés par LPI ;
- Le guide de l'auteur du projet de documentation Linux (LDP ou TLDP).

Les objectifs pour l'examen LPI 101 et les critères d'agrément LATM pour les documents de formation sont résumés ci-dessous. Vous pouvez également les trouver en ligne à l'adresse : . Le Guide de l'auteur LDP [] donne un ensemble de directives à ceux qui souhaitent publier des guides pratiques (HOWTO) ou des manuels via le plus grand système de documentation de GNU/Linux, le Linux Documentation Project.
Le deuxième objectif principal de ce guide, à égalité avec le premier, est issu du challenge de l'auteur du LDP [LDP-AG, 4.1. Writing the Text] : “de créer un ensemble lisible, divertissant, et compréhensible à partir d'informations brutes”.

Plan de formation

Ce cours est prévu pour accompagner des cours pratiques de préparation de l'examen LPI 101. Généralement, la formation prend entre 24 et 32 heures, coupées en sessions consécutives de 8 h mais le cours est adaptable et vous pouvez l'utiliser dans d'autres situations.

Pré-requis et profil des élèves

Nous partons du principe que les élèves ont :

- une expérience de plusieurs années d'utilisation d'ordinateurs basés sur une architecture Intel x86, y compris une bonne connaissance des composants matériels et de leurs interactions avec les composants du système d'exploitation ;

- une culture générale de l'informatique et des réseaux, comme les systèmes binaire et hexadecimal, les unités de mesure les plus courantes (octets, Ko vs Kbit, Mhz, etc.) les structures de système de fichiers, des bases sur la gestion de réseau Ethernet et Internet, le matériel, etc.
- une expérience pratique de plus de 3 mois d'utilisation de GNU/Linux, BSD ou tout autre système Unix, en travaillant en ligne de commande (sur un terminal texte ou dans une console), localement ou à distance.

Les personnes disposant de moins d'expérience mais motivées et prêtes à prendre plus de temps pour rattraper leurs lacunes peuvent également suivre ce manuel. C'est une tâche ardue mais pas impossible.
Les commandes et leur utilisation sont présentées avec leurs références et des exemples ainsi que des exercices et leurs réponses vous mettant dans les conditions de l'examen. Tous les exercices sont facultatifs, les plus recommandés étant présentés ou référencés dans le contenu du cours.

Le programme de certification LPI

Il y a actuellement (NdT : au moment de l'écriture du document, pas de sa traduction) deux niveaux de certification LPI. Le premier niveau LPIC-1 s'obtient en passant les examens LPI 101 et 102. De la même façon, vous obtiendrez le niveau 2 (LPIC-2) en passant les examens LPI 201 et 202. Il n'y a aucun pré-requis pour passer les examens LPI 101 et 102. Vous devrez cependant les avoir obtenus pour passer les examens du LPIC-2.

À l'attention du formateur

Ce manuel ne contient pas de recommandations pour le formateur. Vous noterez les points suivants : les exercices des chapitres 'Gestion du matériel' et 'Le système de fichiers Linux' partent du principe qu'une nouvelle partition peut être créée. A l'installation, assurez vous qu'une grande partition étendue, possédant au moins 100 Mo d'espace libre, sera disponible après la création de toutes les autres partitions.
Les paquets suivants sont nécessaires aux exercices :

- rpm-build
- shareutils

Pas de garantie

Ce manuel vous est délivré sans aucune garantie.

Sources d'information

- www.lpi.org
- www.linux-praxis.de
- www.lpiforums.com
- www.tldp.org
- www.fsf.org
- www.linuxit.com

Conventions de notation

Les commandes et noms de fichiers apparaissent dans le texte en gras.
Les symboles <> indiquent un argument obligatoire.

Les symboles [] indiquent un argument optionnel.

Les commandes pouvant être tapées directement sont mises en valeur comme suit :

Introduction

```
commande
```

Les objectifs détaillés pour l'examen de certification LPI 101

INFORMATION IMPORTANTE : Ceci est la liste des 101 objectifs actuels, qui sont en vigueur **depuis** le 1er avril 2009.
Un poids qui varie de 1 à 10 est affecté à chaque objectif, il représente son importance par rapport aux autres. Les objectifs de poids le plus important seront sujets à davantage de questions durant l'examen.

Sujet 101 : Architecture système

101.1 Déterminer et configurer les paramètres du matériel

Poids 2

Description Les candidats doivent être capables de déterminer et de configurer le matériel et les périphériques sous GNU/Linux.

Domaines de connaissance les plus importants :

- Activer et désactiver les périphériques intégrés.
- Configurer les systèmes avec ou sans périphérique externe tels que les claviers.
- Savoir différencier les types de périphériques de stockage de masse.
- Paramétrer correctement les identifiants matériels pour les différents périphériques, en particulier le périphérique de démarrage.
- Connaître les différences entre les périphériques qui peuvent être connectés à froid ou à chaud.
- Déterminer les ressources matérielles des périphériques.
- Outils et commandes permettant d'obtenir des informations sur les périphériques (par exemple : lsusb, lspci, etc.)
- Outils permettant de manipuler les périphériques USB.
- Compréhension des concepts sysfs, udev, hald et dbus.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- /sys
- /proc
- /dev
- modprobe
- lsmod
- lspci
- lsusb

101.2 Démarrage du système

Poids 3

Description Les candidats doivent pouvoir se repérer dans les phases de démarrage d'un système Linux.

Domaines de connaissance les plus importants

- Fournir des commandes au chargeur de démarrage et passer des paramètres d'amorçage au noyau.
- Démontrer sa connaissance des séquences d'amorçage depuis le lancement du BIOS jusqu'à l'achèvement des séquences de démarrage.
- Consulter les événements de la phase de démarrage dans les journaux (logs).

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- /var/log/messages
- dmesg
- BIOS
- bootloader
- kernel
- init

101.3 Changer de niveau d'exécution et arrêter ou redémarrer le système

Poids 3

Description Les candidats doivent être capables de gérer le niveau d'exécution du système. Cet objectif inclut le passage en mode mono-utilisateur (single user), l'arrêt et le redémarrage du système. Les candidats doivent être capables de prévenir les utilisateurs avant de changer de niveau d'exécution et de terminer les processus correctement. Cet objectif comprend également le paramétrage du niveau d'exécution par défaut.

Domaines de connaissance les plus importants

- Paramétrer le niveau d'exécution par défaut.
- Passer d'un niveau d'exécution à un autre, y compris en mode mono-utilisateur.
- Arrêter et redémarrer le système en ligne de commande.
- Prévenir les utilisateurs avant un changement de niveau d'exécution ou pour d'autres événements systèmes importants.
- Terminer les processus correctement.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- /etc/inittab
- shutdown
- init

- /etc/init.d
- telinit

Sujet 102 : Installation de Linux et gestion de paquetages

102.1 Concevoir un schéma de partitionnement du disque dur

Poids	2
Description	Les candidats doivent être capables de concevoir un schéma de partitionnement du disque dur pour un système Linux.

Domaines de connaissance les plus importants

- Allouer les systèmes de fichiers et l'espace d'échange (swap) sur des partitions ou des disques séparés.
- Ajuster le schéma de partitionnement à l'usage prévu du système.
- S'assurer du bon emplacement de la partition /boot qui contient le chargeur de démarrage.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- / : le système de fichiers racine (root)
- /var
- /home
- espace d'échange (swap)
- points de montage
- partitions

102.2 Installation d'un gestionnaire de démarrage

Poids	2
Description	Les candidats doivent être capables de sélectionner, d'installer et de configurer un gestionnaire de démarrage.

Domaines de connaissance les plus importants

- Fournir des lieux d'amorçage alternatifs et sauvegarder les options de démarrage.
- Installer et configurer un chargeur de démarrage tel que GRUB.
- Interagir avec le chargeur de démarrage en ligne de commande.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- /boot/grub/menu.lst
- grub-install
- MBR

- superblock
- /etc/lilo.conf
- lilo

102.3 Gestion des bibliothèques partagées

Poids 1

Description Les candidats doivent être capables de déterminer les bibliothèques dont dépendent les programmes et les installer en cas de besoin.

Domaines de connaissance les plus importants

- Identifier les bibliothèques partagées.
- Identifier l'emplacement typique des bibliothèques systèmes.
- Charger des bibliothèques partagées.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- ldd
- ldconfig
- /etc/ld.so.conf
- LD_LIBRARY_PATH

102.4 Utilisation du gestionnaire de paquetage Debian

Poids 3

Description Les candidats doivent être capables de gérer les paquetages en utilisant les outils de gestion de paquetage Debian.

Domaines de connaissance les plus importants

- Installer, mettre à jour et désinstaller des paquetages binaires Debian.
- Rechercher des paquetages contenant des fichiers ou des bibliothèques spécifiques installés ou non.
- Obtenir des informations sur un paquetage Debian comme la version, le contenu, les dépendances, l'intégrité du paquetage, et l'état d'installation (que le paquetage soit installé ou non).

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- /etc/apt/sources.list
- dpkg
- dpkg-reconfigure
- apt-get
- apt-cache

- aptitude

102.5 Utilisation de gestionnaire de paquetage RPM et YUM

Poids 3

Description Les candidats doivent être capables de gérer les paquetages avec les outils RPM et YUM.

Domaines de connaissance les plus importants

- Installer, ré-installer, mettre à jour et supprimer les paquetages avec RPM et YUM.
- Obtenir des informations sur un paquetage RPM comme la version, le contenu, les dépendances, l'intégrité du paquetage, la signature et l'état d'installation.
- Déterminer les fichiers relatifs à un paquetage donné, et rechercher à quel paquetage appartient un fichier donné.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- rpm
- rpm2cpio
- /etc/yum.conf
- /etc/yum.repos.d/
- yum
- yumdownloader

Sujet 103 : Commandes GNU et Unix

103.1 Travail en ligne de commande

Poids 4

Description Les candidats doivent être capables de travailler en ligne de commande. C'est l'utilisation du shell bash qui sera traitée dans cet objectif.

Domaines de connaissance les plus importants

- Utiliser des commandes ou des séquences de commandes pour réaliser des tâches simples en ligne de commande.
- Utiliser et modifier l'environnement du shell, en particulier savoir définir, exporter et référencer des variables d'environnement.
- Utiliser et éditer l'historique des commandes.
- Lancer des commandes comprises ou non dans le chemin par défaut.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- .

- bash
- echo
- env
- exec
- export
- pwd
- set
- unset
- man
- uname
- history

103.2 Traitement de flux de type texte par des filtres

Poids 3

Description Les candidats doivent être capables d'appliquer des filtres à un flux de type texte.

Domaines de connaissance les plus importants

- Envoyer des fichiers textes ou des sorties de commandes à des filtres textuels pour les modifier en utilisant des commandes UNIX appartenant au paquetage GNU "textutils".

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- cat
- cut
- expand
- fmt
- head
- od
- join
- nl
- paste
- pr
- sed
- sort
- split
- tail
- tr

- unexpand
- uniq
- wc

103.3 Effectuer une gestion de base sur les fichiers

Poids	4
Description	Les candidats doivent être capables d'utiliser les commandes Linux de base pour gérer les fichiers et les répertoires.

Domaines de connaissance les plus importants

- Copier, déplacer et détruire des fichiers ou des répertoires.
- Copier plusieurs fichiers et répertoires récursivement.
- Supprimer des fichiers et répertoires récursivement.
- Utiliser de manière simple et avancée les caractères de joker (*,?,[..]).
- Utiliser la commande pour rechercher des fichiers sur la base de leurs types , de leurs tailles ou de leurs dates (de création, de modification et d'accès).
- Utiliser les commandes tar, cpio et dd.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- cp
- find
- mkdir
- mv
- ls
- rm
- rmdir
- touch
- tar
- cpio
- dd
- file
- gzip
- gunzip
- bzip2
- file globing (transformation d'un motif générique en une liste de noms de fichiers correspondants)

103.4 Utilisation des flux, des tubes (pipes) et des redirections

Poids 4

Description Les candidats doivent être capables de rediriger des flux et de les connecter dans le but de traiter efficacement ces données textuelles. Les tâches à effectuer comprennent les redirections de l'entrée standard, de la sortie standard et de la sortie standard des erreurs, connecter la sortie d'une commande à l'entrée d'une autre, utiliser la sortie d'une commande comme paramètre pour une autre commande et envoyer le résultat en même temps sur la sortie standard et dans un fichier.

Domaines de connaissance les plus importants

- Redirection de l'entrée standard, de la sortie standard et de la sortie standard des erreurs.
- Connecter la sortie d'une commande à l'entrée d'une autre commande.
- Utiliser la sortie d'une commande comme paramètre pour une autre commande.
- Envoyer simultanément le résultat d'une commande vers la sortie standard et vers un fichier.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- tee
- xargs

103.5 Création, surveillance et destruction de processus

Poids 4

Description Les candidats doivent être capables d'effectuer une gestion de base des processus.

Domaines de connaissance les plus importants

- Exécuter un processus en avant-plan et en arrière plan.
- Indiquer qu'un programme doit continuer son exécution après la déconnexion.
- Contrôler les processus actifs.
- Sélectionner et trier les processus à afficher.
- Envoyer des signaux aux processus.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- &
- bg
- fg
- jobs

- kill
- nohup
- ps
- top
- free
- uptime
- killall

103.6 Modification des priorités des processus

Poids 2

Description Les candidats doivent être capables de gérer les priorités des processus.

Domaines de connaissance les plus importants

- Savoir la priorité par défaut affectée à un processus créé.
- Exécuter un programme avec une priorité plus haute ou plus basse que celle par défaut.
- Changer la priorité d'un processus en cours d'exécution.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- nice
- ps
- renice
- top

103.7 Recherche dans des fichiers texte avec les expressions rationnelles

Poids 2

Description Les candidats doivent être capables de manipuler des fichiers et des données de type texte en utilisant des expressions rationnelles. Cet objectif comprend la création d'expressions rationnelles contenant plusieurs caractères spéciaux. Il comprend également l'utilisation d'outils à base d'expressions rationnelles pour effectuer des recherches dans un système de fichiers ou dans le contenu d'un fichier.

Domaines de connaissance les plus importants

- Créer des expressions rationnelles simples contenant plusieurs éléments de notation.
- Utiliser des outils à base d'expressions rationnelles pour effectuer des recherches dans un système de fichiers ou dans le contenu d'un fichier.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- grep
- egrep
- fgrep
- sed
- regex

103.8 Édition de fichiers texte avec "vi"

Poids 3

Description Les candidats doivent être capables d'éditer le contenu de fichiers texte en utilisant "vi". Cet objectif comprend le déplacement dans "vi", les modes de "vi", l'insertion, la modification, la destruction, la copie et la recherche de texte.

Domaines de connaissance les plus importants

- Se déplacer dans un document édité avec "vi".
- Utiliser les modes "vi" de base à savoir le mode commande, le mode insertion et le mode remplacement.
- Insérer, modifier, détruire, copier et rechercher du texte.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- vi
- /, ?
- h,j,k,l
- i, o, a
- c, d, p, y, dd, yy
- ZZ, :w!, :q!, :e!

Sujet 104 : Disques, Système de fichiers Linux , Arborescence de fichiers standard.

104.1 Création de partitions et systèmes de fichiers.

Poids 2

Description Les candidats doivent être capables de créer des partitions et des systèmes de fichiers. Ceci inclut la prise en charge des partitions d'échange (swap).

Domaines de connaissance les plus importants

- Utiliser la commande "mkfs" pour configurer une partition et créer différents types de système de fichiers comme :

- ext2
- ext3
- xfs
- reiserfs v3
- vfat

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- fdisk
- mkfs
- mkswap

104.2 Maintenir l'intégrité des systèmes de fichiers

Poids	2
Description	Les candidats doivent être capables de maintenir l'intégrité d'un système de fichiers, ainsi que les données supplémentaires associées à la journalisation.

Domaines de connaissance les plus importants

- Vérifier l'intégrité du système de fichiers.
- Contrôler l'espace libre et les inodes.
- Réparer les problèmes simples du système de fichiers.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- du
- df
- fsck
- e2fsck
- mke2fs
- debugfs
- dumpe2fs
- tune2fs
- xfs tools (tels que xfs_metadump et xfs_info)

104.3 : Contrôle du montage et du démontage des systèmes de fichiers

Poids	3
Description	Les candidats doivent être capables de configurer le montage, le démontage et les options de montage des systèmes de fichiers.

Domaines de connaissance les plus importants

- Monter et démonter manuellement les systèmes de fichiers.
- Configurer le montage des systèmes de fichiers au démarrage du système.
- Configurer les options de montage des systèmes de fichiers. Par exemple, permettre aux utilisateurs de monter et démonter un système de fichiers amovible.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- /etc/fstab
- /media
- mount
- umount

104.4 Gestion des quotas de disque

Poids	1
Description	Les candidats doivent être capables de gérer les quotas disque des utilisateurs.

Domaines de connaissance les plus importants

- Configurer un quota de disque pour un système de fichiers.
- Éditer, vérifier et générer des rapports d'utilisation de quotas des utilisateurs.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- quota
- edquota
- repquota
- quotaon

104.5 Gérer les permissions et les propriétaires des fichiers

Poids	3
Description	Les candidats doivent être capables de gérer les droits d'accès et les propriétaires des fichiers.

Domaines de connaissance les plus importants

- Gérer les permissions d'accès sur des fichiers classiques, des fichiers spéciaux comme sur des répertoires.
- Utiliser des modes comme suid, sgid et sticky bit pour garantir la sécurité.
- Savoir changer le mode de création de fichiers par défaut.
- Utiliser le groupe propriétaire pour donner des permissions aux membres d'un groupe.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- chmod
- umask
- chown
- chgrp

104.6 Créer et changer les liens symboliques et physiques sur les fichiers

Poids	2
Description	Les candidats doivent être capables de gérer des liens symboliques et physiques sur un fichier.

Domaines de connaissance les plus importants

- Créer les liens.
- Identifier les liens symboliques des liens physiques.
- Copier versus lier les fichiers
- Utiliser des liens pour les tâches d'administration système.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- ln

104.7 Recherche de fichiers et placement des fichiers aux endroits adéquats

Poids	2
Description	Les candidats doivent être familiarisés avec l'arborescence standard de fichiers FHS (Filesystem Hierarchy Standard), y compris la place adéquate d'un fichier et les classifications des répertoires.

Domaines de connaissance les plus importants

- Comprendre l'emplacement correct d'un fichier dans le FHS.
- Rechercher les fichiers et les commandes dans l'arborescence Linux
- Connaitre l'emplacement des fichiers et des répertoires importants dans l'arborescence Linux.

Liste non-exhaustive de termes, fichiers et utilitaires utilisés pour cet objectif

- find
- locate
- updatedb
- whereis
- which
- type

- /etc/updatedb.conf

Installation

Pré-requis

Aucun

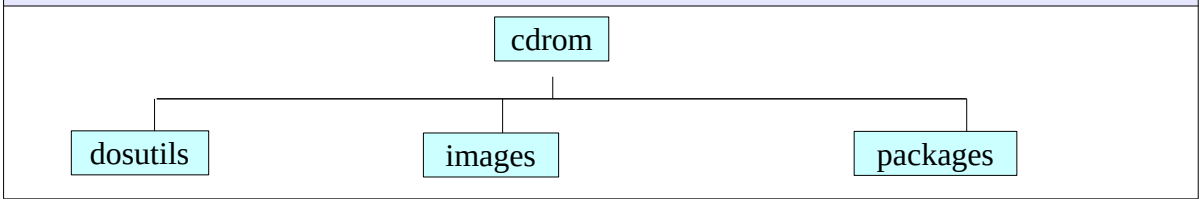
Objectifs

- Comprendre l'organisation d'un CD d'installation Linux type
- Effectuer différents types d'installation
- Organiser un partitionnement simple (voir également p .Error: Reference source not found)

Le CD d'installation

Les différentes distributions Linux utilisent des noms de répertoires différents sur les CDs d'installation. La structure générique est la suivante :

Structure générique d'un CD d'installation Linux



packages: ce répertoire contient les paquets au format binaire. Voici les noms correspondants pour les distributions les plus courantes :

debian: **dist**
mandrake: **Mandrake**
redhat: **RedHat**
suse: **suse**

Tous les logiciels installés sur le système à l'installation viennent de ces paquets logiciels. Consultez la partie traitant des gestionnaires de paquets page Error: Reference source not found pour plus de détails.

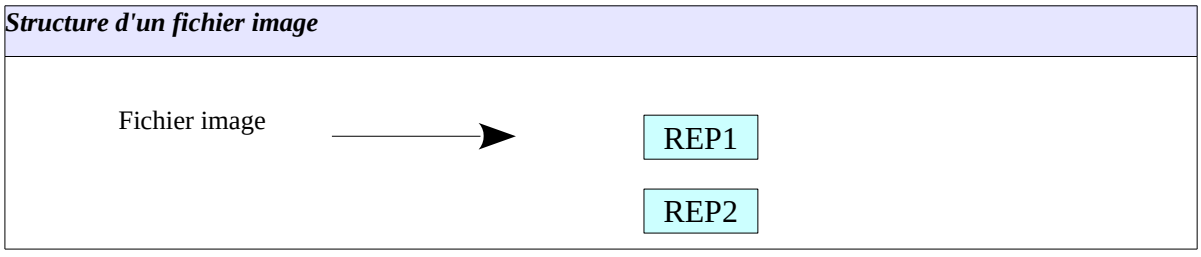
images: ce répertoire contient plusieurs “images”. Ce sont des fichiers spéciaux contenant des systèmes de fichiers. Un exemple d'image est un disque virtuel contenant des modules du noyau nécessaires pour démarrer votre système (initrd). Il y a différents types d'images qui permettent de :

- démarrer le programme d'installation ;
- gérer un matériel ou un besoin spécifique via des modules du noyau complémentaires ;
- dépanner le système (image rescue).

Pour installer à partir d'une disquette, vous devez copier certaines images sur la disquette. Sous Linux, on utilise le programme **dd**, **rawrite** permet de le faire sous DOS.

Une image est un fichier spécial qui peut contenir des sous-répertoires (comme un fichier archive).

Structure d'un fichier image



Une image peut être montée sur un pseudo-périphérique. Si l'image est nommée *Image*, la commande suivante vous permettra de voir le contenu de ce fichier dans le répertoire **/mnt/floppy** :

```
mount -o loop /path/to/Image /mnt/floppy
```

dosutils: ce répertoire contient les programmes DOS qui pourront être utilisés pour préparer l'installation de Linux, comme le programme **rawrite.exe** dons nous venons de parler. Vous y trouverez également **fips** qui vous permet de partitionner un disque C:\ en deux sans perte de données et à la condition que le système de fichiers soit FAT et pas NTFS.

Installations locales

L'installation la plus courante et la plus facile est l'installation locale. La plupart des distributions sont disponibles sous la forme d'une image ISO d'un Cdrom avec un script d'installation automatique. Sur les machines sans lecteur de Cdrom, il est également possible d'installer à partir d'une disquette.

Installation à partir d'un CD-ROM

Modifiez le paramétrage du BIOS de votre machine pour démarrer à partir du CD. Le menu du programme d'installation vous permet de faire une installation avancée ou basique.

Installation à partir d'une disquette

Si vous ne pouvez pas démarrer à partir d'un CD-ROM vous allez devoir créer la disquette d'installation. Cela peut arriver lorsque le CD n'est pas amorçable ou si vous avez téléchargé une image de votre distribution à un format autre qu'ISO.

Création de la disquette d'installation		
`dd if=/path/to/<image_name>`	`of=/dev/fd0`	Sous Linux
`rawrite.exe`		sous Windows (pas NT)

Pour les distributions RedHat, les images d'installation sont dans le répertoire **images**. L'image de base est boot.img. Vous en trouverez de plus spécifiques comme **bootnet.img** ou **pcmcia.img**.

Sur une distribution Suse, l'image de la disquette est dans le répertoire **disks** et l'image est appelée **bootdisk**.

Installation réseau

Pour une installation de RedHat vous aurez besoin d'une disquette d'installation spécifique. Faites une disquette de démarrage en utilisant l'image **bootnet.img**.

```
dd /mnt/cdrom/images/bootnet.img of=/dev/fd0
```

Pendant la première partie de l'installation en mode texte, vous devrez paramétrer le clavier et réseau. La suite de l'installation se fait par FTP, NFS ou HTTP. Vous pouvez faire une installation en mode graphique en utilisant un protocole qui permet de monter une source réseau sur un point de montage local (NFS) mais si vous utilisez les protocoles de retrait de fichier (FTP ou HTTP), l'installation se fera en mode texte. Ceci n'est plus forcément le cas pour la plupart des distributions modernes.

Notez également que l'installation réseau est possible directement à partir du CD pour la plupart des distributions modernes (pour Mandrake à partir du disque 2 et en introduisant le paramètre askmethod au démarrage du CD de Fedora).

Disque de récupération

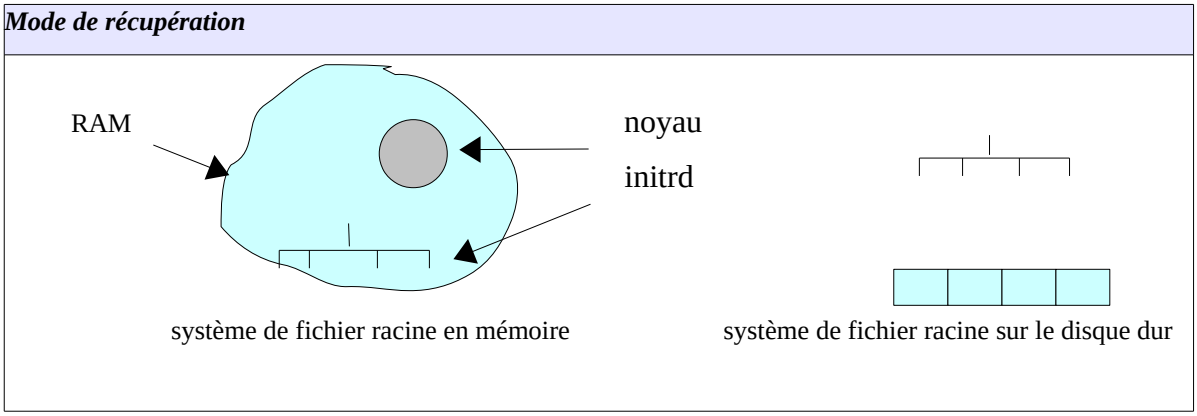
Si vous ne pouvez plus démarrer votre système Linux, vous pouvez démarrer en utilisant un disque de récupération. C'est une petite version de Linux qui montera un système de fichier virtuel minimal en mémoire.

Le système d'exploitation fonctionne entièrement en mémoire vive (RAM). Il permet d'accéder à votre système de fichier racine sur le disque dur. La plupart des disques de récupération le trouvent automatiquement. Par exemple si la racine se trouve sur la première partition logique du premier disque IDE (/dev/hda5), le disque de récupération peut la monter sur un sous-répertoire du système de fichier en mémoire, par exemple /mnt/system.

Changement de perspective

Dans cette situation, nous avons **deux systèmes de fichiers racines**. Pour utiliser celui présent sur le disque dur comme racine de votre système, vous devez changer de perspective (changer de racine). C'est ce que fait la commande **chroot** :

```
chroot /mnt/sysimage
```



C'est parti

Ancienne méthode :

1. créer une disquette amorçable en utilisant l'image **boot.img** et en la copiant avec : dd if=boot.img of=/dev/fd0
2. copier l'image rescue?img sur une deuxième disquette : dd if=rescue.img of=/dev/fd0
3. démarrer le système avec la disquette l'image **boot.img**
4. à l'invite de LILO, tapez « linux rescue ». Vous devriez voir quelque chose comme :
   Insert root file system disk:
5. insérer la disquette avec rescue.img et appuyer sur entrée
6. attendre l'invite du shell qui indique que le système a fini de démarrer
7. vous aurez peut-être encore à déterminer quel système de fichier est la racine (non présenté ici)

Nouvelle méthode :

1. insérer le CD d'installation de votre distribution (Suse, RedHat, Mandriva, Debian, ...)
2. tapez « linux rescue » à l'invite
3. suivez les instructions
4. le système de fichier racine devrait être détécté
5. si votre système de fichier racine est monté sur **/mnt/sysimage**, tapez la commande suivante :

```
chroot /mnt/sysimage
```

Plan de partitionnement

Le système d'exploitation utilise un mécanisme appelé montage pour accéder aux différentes ressources sur le disque dur. Sur les systèmes d'exploitation de type UNIX, cela se fait en attachant une partition à un répertoire, appelé point de montage.

L'illustration un peu plus bas vous montre un plan de partitionnement envisageable. Des ressources différentes, partitions, partages réseau, CD-ROMs, etc. sont attachés sur différents points de montage.

Du point de vue de l'utilisateur, tout est vue comme une arborescence simple composée de répertoires et sous-répertoires.

Arborescence du système de fichiers

La racine de l'arborescence est appelée root et est représentée par un slash : « / ». Le point de montage / est également le premier répertoire attaché à une ressource (périphérique racine) par le système d'exploitation.

Une fois la racine montée, les répertoires et sous-répertoires présents sur ce périphérique peuvent également être utilisés comme points de montage pour d'autres périphériques pour compléter l'arborescence des répertoires.

Ce processus se produit de la façon suivante :

1. le chargeur de démarrage charge le noyau et lui indique où se trouve la racine (cf « Démarrage de Linux » dans le cours pour la préparation de l'examen LPI 102).
2. Les autres répertoires sont montés à partir des informations du fichier /etc/fstab (voir p.Error: Reference source not found)

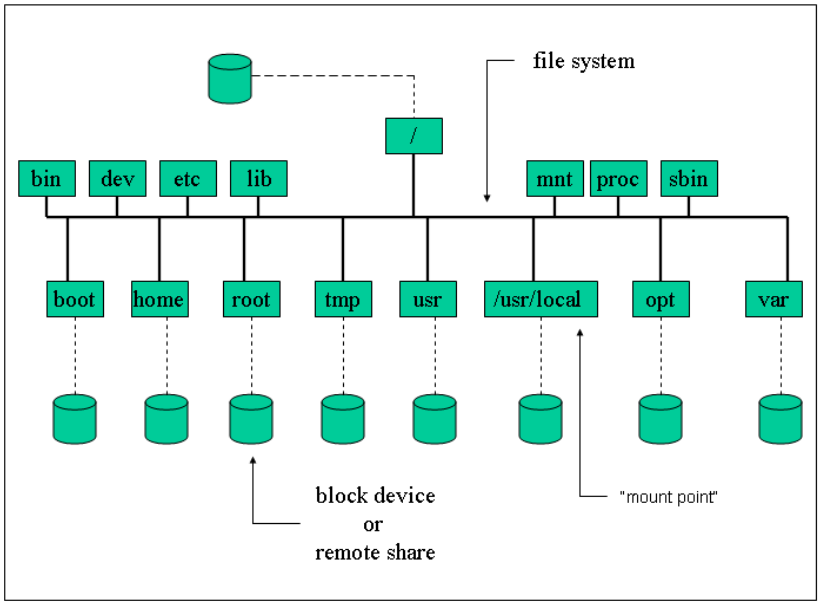


Illustration 1: Points de montage sur le système de fichiers

Partitionnement du disque

Lorsque vous installez Linux vous devez créer votre partitionnement. Cette étape de l'installation se fait généralement en passant par un outil graphique comme Yast ou DiskDruid. Ces programmes ont trois fonctions :

- créer des partitions d'une taille donnée ;
- sélectionner le type de système de fichiers (voir p.Error: Reference source not found) ;
- définir le point de montage pour chaque partition

Si vous utilisez le mode expert pour une installation, vous utiliserez sans doute fdisk (voir p.Error: Reference source not found) pour uniquement créer les partitions.

Au minimum, vous aurez besoin de deux partitions : une partition racine et une partition d'échange (pour la mémoire paginée). Il n'y a pas véritablement de règle le partitionnement, il se choisit en général là partir de a fonction de l'ordinateur (poste de travail, serveur de mail etc.).

La partition SWAP

Vous devez choisir la taille de la partition d'échange à sa création. Encore un fois, il n'y a pas de règle. La taille nécessaire dépend du type d'application qui seront lancées sur le PC (bureautique, serveur, applications 3D, etc.). En général cependant, pour un noyau 2.4 avec une quantité mémoire vive assez moyenne (c'est à dire moins de 256 Mo), on créée une partition d'échange du double de la taille de la RAM. Sur un vieux noyau 2.2, nous aurions donné à cette partition la même taille que la quantité de RAM.

En général, la pagination se fait à partir d'une partition. Dans la table des partitions, la valeur hexadécimale

correspondant à la partition SWAP est 82.

REMARQUE
Contrairement aux partitions utilisées pour le stockage de données, une partition swap n'est jamais montée. Il n'y a donc pas de point de montage pour ce type de partition. Pour créer une partition SWAP à l'installation, il suffit de sélectionner le système de fichier SWAP (ou partition d'échange). Une fois le système démarré, les informations concernant les partitions SWAP sont dans le fichier /proc/swaps.

Vous pouvez également créer des fichiers d'échange au lieu des partitions (voir LPI 201). C'est souvent utilisé en situation d'urgence sur un système en marche mais rarement à installation.

Plusieurs systèmes d'exploitation sur une machine

(Cette partie n'est pas nécessaire à la préparation de l'examen).

Si Windows est déjà installé sur le système, le programme d'installation configurera automatiquement LILO ou GRUB pour permettre de démarrer les deux systèmes.

Pré-installation

Avant toute chose vous devriez lancer un programme de défragmentation du disque. Vous aurez ainsi l'assurance que les données utilisées par Windows se trouvent au début du disque.

Ensuite, utilisez un programme comme PartitionMagic ou fips pour partitionner le disque en deux. Les programmes Windows se trouvent sur la première partition. La seconde partition devra être suffisamment grande pour contenir Linux.

Remarque : l'espace moyen nécessaire pour une distribution Linux est de 4 Go.

Démarrer l'installation à partir de DOS

Pour les Windows 9x, redémarrez votre ordinateur en commande DOS. Si vous installez une RedHat, tapez la commande E:\DOSUTILS\AUTOBOOT.BAT qui lancera le programme d'installation. Pour installer une Suse, tapez la commande E:\setup.exe sous DOS.

Le disque dur vu par Windows

Sous Windows, le système ne voit que les partitions FAT et NTFS. Les partitions Linux sont inaccessibles.

Le disque dur vu par Linux

Sous Linux, la partition Windows devrait être nommée /dev/hda1 (pour la première partition du premier disque dur). Par défaut, cette partition n'est pas montée. Vous pouvez créer un répertoire /dos ou /mnt/dos et la monter et y accéder.

Exercices et résumé

Questions (réponses p.Error: Reference source not found)

Oui ou non

1. Le programme rawrite fonctionne sous Linux et est utilisé pour copier un fichier image sur une disquette
2. À la préparation d'un nouveau plan de partitionnement sur un disque vierge, n'importe quelle partition peut être choisie comme partition racine

Glossaire

Terme	*Description*
Système de fichiers virtuel	Un système de fichiers est une structure de données qui permet d'organiser les données sur un disque. Pour l'utilisateur, les données se trouvent dans une arborescence de répertoires dont la racine est appelée « root » et est notée « / ». Cette structure est également appelée système de fichiers virtuel puisque l'arborescence ne change pas selon soit l'organisation du disque ou le partitionnement. Sous un système DOS, la situation est radicalement différente : par exemple si votre disque est découpé en 4 partitions, les utilisateurs devront savoir que les données peuvent se trouver sur C:\, D:\, E:\, ou F:\ et dans ce cas, le premier lecteur de CD-Rom sera nommé G:\.
Point de montage	Répertoire sur lequel est lié une partition pour la rendre accessible au système.
Plan de partitionnement	Le partitionnement est réalisé à l'installation. Il comprend le découpage du disque en partitions et le choix des points de montage pour les relier au système de fichier virtuel sur lequel le système d'exploitation est installé. La FHS (FileSystem Hierarchy Standard – Norme de hiérarchie du système de fichiers) indique où sont installés les composants logiciels, où se trouvent les répertoires personnels des utilisateurs etc. Vous devez en tenir compte pour le découpage de votre système en fonction de son rôle. Par exemple la plupart des logiciels sont installés dans le répertoire /usr. Par conséquent, assurez-vous que cette partition est suffisamment grande, au minimum 2
Mode de récupération	Il s'agit d'un système Linux minimal avec juste assez d'outils pour accéder au disque. On utilise en général ce mode en démarrant à partir d'un CD-Rom d'installation, le système fonctionne entièrement en mémoire vive.
root (/)	Répertoire le plus haut de l'arborescence sur lequel est attaché la partition racine. L'ensemble des répertoires et sous-répertoires peut se trouver sur cette partition ou d'autres sous-répertoires peuvent servir de point de montage pour d'autres partitions, cela dépend du plan de partitionnement choisi à l'installation.

Commandes

Commande	*Description*
chroot	Va dans un répertoire (comme avec cd) et considère ce répertoire comme la racine (/). Par défaut, **chroot** tentera de lancer le shell Bash **/bin/bash**, mais vous pouvez spécifier d'autres commandes à lancer (voir « serveurs chrootés » dans LPI202).
dd	Outil qui permet de copier des fichiers tout comme des parties d'un périphérique (disque dur, CD-ROM ou disquette). Les CD d'installation contiennent en général des fichiers que l'on appelle des fichiers image qui peuvent être recopiées sur disquette en utilisant cette commande.
fips.exe	Programme disponible sur la plupart des CD d'installation utilisé pour retailler une

	partition FAT, afin de récupérer de la place pour installer Windows et Linux en fouble amorçage.
rawrite	Equivalent à **dd** sous DOS

Exercices

1. Procédez à une installation locale à partir d'un CD-ROM en suivant la stratégie indiquée ci-dessous. Seuls les utilisateurs avancés familiers de l'éditeur de texte vi et de la gestion de paquets devraient tenter les étapes FACULTATIVES.
   (i) Type d'installation : choisissez « personnalisée »
   (ii) Partitionnement : partitionnez le disque manuellement avec Disk Druid

   Voici une proposition pour un espace disque de 3 Go. Si vous disposez de plus d'espace, faites une partition /usr plus grande et installez plus de paquets que ceux indiqués à l'étape (iv). ATTENTION : laissez une partition libre d'au moins 100 Mo, nous en aurons besoin plus tard./boot 20M
   / 250M
   /usr 2300M
   /home 50M
   /tmp 100M
   /var 150M
   SWAP 128M Remarquez que SWAP est le type de système de fichier et que cette partition n'est pas montée (voir p.28)
   (iii) (FACULTATIF) Installez LILO sur /dev/hda2 ou ne l'installez pas, en tout cas ne l'installez pas sur le MBR (/dev/hda)
   nous souhaitons délibérément que le système n'arrive pas à démarrer. Le chargeur de démarrage sera remis à l'étape 2 (i)
   (iv) Paquets à installer (les noms diffèrent suivant la distribution) :
   « Interface graphique X » + « Environnement de bureau GNOME » OU « Environnement de bureau KDE »
   « Editeurs »
   « Outils graphiques pour Internet »
   « Développement logiciel » (Ceci a son importance puisque nous compilerons des paquets plus tard
   (v) Ne créez pas la disquette de démarrage
2. (FACULTATIF) Dépanner le système
   (i) Redémarrez à partir du CD-ROM d'installation. À l'invite, tapez :
   linux rescue
   (ii) Lisez les informations à l'écran. À l'invite, utilisez la commande chroot comme nous l'avons vu plus tôt.
   (iii) Le paquet lilo doit être installé. Éditez le fichier /etc/lilo.conf avec vi. Vous devriez avoir :

```
boot=/dev/fd0
prompt
linear
timeout=50
image=/boot/vmlinuz-<kernel-version>
 label=linux
 read-only
 root=/dev/<root-partition>
```

   (iv) lancez /sbin/lilo. En cas d'erreur, vous aurez peut-être à remplacer linear par lba32

Configuration matérielle

Pré-requis

Aucun

Objectifs

- Comprendre les affectations de ressources matérielles (interruptions – IRQ, ports d'entrée / sortie et accès direct à la mémoire DMA)
- Avoir une vue d'ensemble des matériels comme les cartes d'extension et les périphériques USB et SCSI
- Détection des cartes réseau et des imprimantes (sans configuration)
- Comprendre les étapes de base pour la configuration des modems et des cartes son

Allocation de ressources

Pour que les périphériques d'un PC puissent communiquer directement avec les ressources du système, essentiellement le processeur, le système alloue des ressources (canaux, lignes) à chaque périphérique. Ces ressources sont les requêtes d'interruption matérielle (IRQ), les adresses d'entrée sortie (I/O) et les canaux d'accès directs à la mémoire (DMA).

Interruptions (IRQs) : Les périphériques utilisent les interruptions matérielles pour demander du temps processeur. Le processeur arrête son activité et procède aux instructions demandées par le matériel. Sur PC il y a 16 interruptions directement connectées sur les périphériques, elles vont de **0** à **15**.

Entrées/sorties (I/O address) : Les entrées / sorties correspondent à des adresses spécifiques dans la mémoire système. Le processeur et le périphérique communiquent en écrivant et en lisant à ces adresses.

Accès direct à la mémoire (DMA) : certains périphériques accèdent à la mémoire système au travers d'un canal d'accès direct à la mémoire. Ainsi ils accèdent aux données sans passer par le processeur, ce qui peut améliorer les performances.

Liste des ressources allouées

Le noyau inscrit dans les fichiers suivants sur /proc les ressources utilisées par le matériel :

/proc/dma
/proc/interrupts
/proc/ioports
/proc/pci

Les commandes **lspci** et **dmesg** permettent également d'avoir la liste des ressources allouées.

lspci : Liste les informations du chipset pour tous les périphériques PCI. Vous obtiendrez les IRQs et les ports d'entrées / sorties avec l'option **-v**. L'option -**b** affiche les ressources affectées par le BIOS et non plus par le noyau.

dmesg : Affiche les messages du noyau en continu. Cette commande affiche également les messages affichés au démarrage du système à l'étape de chargement du noyau (kernel stage). À cette étape le noyau recherche les périphériques et charge les pilotes (modules ou inclus au noyau) automatiquement. Ces informations se trouvent dans /var/log/dmesg.

Ressources types

Périphérique	*Port d'entrée/sortie*	*IRQ*
/dev/ttyS0	0x03f8	4
/dev/ttyS1	0x02f8	3
/dev/lp0	0x378	7
/dev/lp1	0x278	5
soundcard	0x220	

Affectation de ressources manuelle

REMARQUE :
L'exemple suivant est très courant, mais comme les modules du noyau ne seront vus que pour les LPI 102, certains auront des difficultés à le comprendre. Vous pouvez sans problème passer cet exemple et vous rendre à la partie 2.

Exemple : configuration de deux cartes réseau

1. Vous pouvez passer des paramètres au noyau au démarrage pour les pilotes inclus au noyau. Dans cet exemple typique, vous disposez de deux cartes réseau, mais une seule est détectée. La commande suivante indique au noyau :
   - qu'une carte utilise l'IRQ 10 et le port d'entrée sortie 0x300 ;
   - que l'autre carte utilise l'IRQ 9 et le port d'entrée sortie 0x340

```
ether=10,0x300,eth0     ether=9,0x340,eth1
```

Vous pouvez taper cette commande en option au prompt de LILO ou GRUB. Sinon, comme pour les paramètres de mémoire vus précédemment, ajoutez ces informations au champ `append=` dans **/etc/lilo.conf** ou **/etc/grub.conf** (ou /boot/grub/grub.conf).

Vous pouvez constater que l'instruction `ether=` est une commande générique pour le noyau comme `root=`, `mem=` ou `init=`. Notez également que vous n'avez pas besoin de donner d'autres informations sur les cartes réseau (Intel, netgear, etc.).

2. Vous pouvez passer les mêmes paramètres aux modules (pilotes de périphériques chargés dynamiquement) en utilisant **/etc/modules.conf** ou **/etc/conf.modules**. Si l'on part du principe que dans l'exemple précédent les deux cartes utilisent le module e100.o, alors le fichier **/etc/modules.con**f contiendrait :

```
alias eth0 e100
alias eth1 e100


options eth0 io=0x300 irq=10
options eth1 io=0x340 irq=9
```

Cartes d'extension

Les cartes d'extension les plus courantes sont les cartes ISA et PCI. Il n'y a presque plus rien à faire pour les configurer depuis les noyaux 2.4. Cependant, pour les bus ISA et avec des noyaux plus anciens, il était nécessaire de scanner le bus pour détecter les cartes présentes (son, ethernet, etc.).

Le programme **pnpdump** fournit par le paquet isapnptools permet de scanner le bus ISA pour détecter les périphériques « Plug and Play ». Il renvoie en sortie le chipset de la carte ainsi que les paramètres d' adresses d'entrée sortie, DMA et IRQ. Si la sortie du programme est sauvegardée dans le fichier **/etc/isapnp.conf**, les périphériques ISA sont configurés au démarrage par **isapnp**.

Depuis les noyaux 2.4, l'initialisation des périphériques PnP est géré par le module isapnp.o

USB

L'Universal Serial Bus (USB) est un bus informatique plug-and-play à transmission série servant à brancher des périphériques informatiques à un ordinateur. Il existe quatre types de périphériques USB :

- Les périphériques d'affichage (écrans, etc.)
- Les périphériques de communication (modems, etc.)
- Les périphériques audio (cartes son)
- Les périphériques de stockage de masse (disques, etc.)
- Les interfaces Homme / Machine (HID)

Ces périphériques sont branchés sur un port USB qui est géré par un contrôleur USB. Linux gère les contrôleurs USB depuis les noyaux 2.2.7 (voir le Linux USB sub-system HOWTO)

Les contrôleurs USB

Il y a 3 types de contrôleurs USB :

Contrôleur USB	*Module du noyau*
OHCI (Compaq)	`usb-ohci.o`
UHCI (Intel)	`usb-uhci.o`
EHCI (USB v 2.0)	`ehci-hdc.o`

La commande **lsusb** donne la liste des périphériques USB connectés sur la machine :

```
lsusb
Bus 001 Device 001: ID 0000:0000
Bus 001 Device 002: ID 04a9:1055 Canon, Inc.
```

Hotplug (branchement à chaud)

Hotplug est un système utilisé pour tenir à jour l'état du système lorsque vous branchez ou débranchez des périphériques amovibles. Dans la plupart des cas, le noyau signale les évènements en envoyant des paramètres au script **/sbin/hotplug**.

Le script **hotplug** lance les scripts situés dans le répertoire /etc/hotplug.d (le script par défaut est **default.hotplug**) qui lance à son tour l'agent approprié défini dans **/etc/hotplug**. Les noms des agents correspondent aux type de connexions : ieee1394, net, pci et usb.

Le journal ci-dessous montre ce qu'il se passe lorsqu'on branche une caméra USB :

1ère étape : Les modules du noyau identifient l'évènement et les identifiants de produit et de vendeur :

```
13:26:19  kernel: hub.c: new USB device 00:07.2-1, assigned address 5

13:26:19  kernel: usb.c: USB device 5 (vend/prod 0x4a9/0x3058) is not claimed by any
active driver.
```

2ème étape : les arguments sont passés au script default.hotplug :

```
13:26:19 default.hotplug[10507]: arguments (usb) env (DEVFS=/proc/bus/usb OLDPWD=/
PATH=/bin:/sbin:/usr/sbin:/usr/bin ACTION=add PWD=/etc/hotplug HOME=/ SHLVL=2
DEVICE=/proc/bus/usb/001/005 PRODUCT=4a9/3058/1 TYPE=255/255/255 DEBUG=yes
_=/bin/env)
```

3ème étape : l'agent usb (usb.agent) associe le matériel à une caméra usb (en utilisant usb.usermap)

```
13:26:19  default.hotplug[10507]: invoke /etc/hotplug/usb.agent ()

13:26:23  usb.agent[10507]: Setup usbcam for USB product 4a9/3058/1

13:26:23 usb.agent[10507]: Module setup usbcam for USB product 4a9/3058/1

13:26:38 devlabel: devlabel service started/restarted
```

À partir de cet exemple, vous pouvez voir que la première étape utilise les modules du noyau et que les étapes suivantes utilisent hotplug. Il faut également avoir une bonne carte des matériels USB (usb.usermap) pour pouvoir utiliser le matériel.

usbmgr

Il existe sur les systèmes Debian une alternative à hotplug : usbmgr. Voici les fichiers principaux à connaître :

/usr/sbin/usbmgr	Le service à l'écoute des évènements en rapport avec l'USB
/usr/sbin/dump_usbdev	Un outil permettant de lister les périphériques USB connectés (comme lsusb)
/etc/usbmgr/usbmgr.conf	Fichier de configuration contenant les identifiants de matériel

SCSI

Types d'interfaces SCSI

Il y a 2 (NdT : maintenant 3) types d'interfaces SCSI

- SCSI-1 : interface 8 bits avec un bus supportant 8 périphériques, contrôleur inclus, donc 7 périphériques réels ;
- SCSI-2 : interface 16 bits (WIDE). Le bus peut gérer 16 périphériques, contrôleur inclus (donc 15).

Les périphériques SCSI sont identifiés de façon unique à partir du triplet que l'on nomme SCSI_ID :

- le canal SCSI
- l'identifiant du périphérique
- le numéro logique LUN (Logical Unit Number)

Le canal SCSI

Chaque carte carte SCSI utilise un canal de données sur lequel s'attachent les périphériques (disque, CDROM, etc.). Les canaux sont numérotés à partir de 0.

Le numéro identifiant du périphérique

Un identifiant unique (Device ID) est attribué à chaque périphérique manuellement à l'aide de cavaliers directement sur le périphérique. Sur une carte SCSI-1, les identifiants vont de 0 à 7 et donc de 0 à 15 pour une carte SCSI-2.

Le numéro logique LUN

Les numéros logiques sont utilisés pour faire la différence entre les périphériques à l'intérieur d'un périphérique identifié : partitions d'un disque ou lecteur DAT pour un robot qui en comprend plusieurs. C'est devenu rare de nos jours avec la baisse des coûts des cartes SCSI et leur plus grande capacité.

Détection du matériel

La liste des périphériques détectés se trouve dans le fichier **/proc/scsi/scsi**. L'exemple suivant est tiré du SCSI-2.4-HOWTO :

```
Attached devices:

  Host: scsi0 Channel: 00 Id: 02 Lun: 00

    Vendor: PIONEER  Model: DVD-ROM DVD-303  Rev: 1.10

    Type:   CD-ROM             ANSI SCSI revision: 02

  Host: scsi1 Channel: 00 Id: 00 Lun: 00

    Vendor: IBM      Model: DNES-309170W     Rev: SA30

    Type:   Direct-Access      ANSI SCSI revision: 03
```

La commande **scsi_info** affiche le SCSI_ID et le modèle du périphérique à partir des informations de **/proc/scsi/scsi**. À partir de l'exemple précédent, voici la sortie de la commande :

```
 scsi_info /dev/sda

SCSI_ID="0,0,0"

MODEL="IBM DNES-309170W"
```

```
FW_REV="SA30"
```

Démarrer à partir d'un disque SCSI

Par défaut le système démarre à partir du périphérique ayant le SCSI_ID 0. Vous pouvez modifier ce comportement dans le BIOS de la carte qui est accessible au démarrage.

Si le système est composé de disques IDE et SCSI, vous devrez commencer par paramétrer le BIOS de la machine pour démarrer sur un périphérique SCSI.

Cartes réseau

Le pilote de la carte réseau (NIC) doit être compilé avec le noyau (ou en module). Vous trouverez des informations sur la ou les cartes utilisées sur votre machine avec :

dmesg, **lspci, scanpci**, **/proc/interrupts**, **/sbin/lsmod**.ou **/etc/modules.conf:**

- dmesg

```
dmesg

Linux Tulip driver cersion 0.9.14 (February 20, 2001)

PCI: Enabled device 00:0f.0 (0004 ->0007)

PCI: Found IRQ 10 for device 00:0f.0

eth0: Lite-On 82cl68 PNIC rev 32 at 0xf800, 00:0A:CC:D3:6E:0F,

IRQ 10

eth0: MII transceiver #1 config 3000 status 7829 advertising
```

- /proc/interrupts

```
cat /proc/interrupts

0:    8729602          XT-PIC  timer

1:          4          XT-PIC  keyboard

2:          0          XT-PIC  cascade

7:          0          XT-PIC  parport0

 8:          1          XT-PIC  rtc

10:     622417          XT-PIC  eth0

 11:          0          XT-PIC  usb-uhci

   14:     143040          XT-PIC  ide0

15:        180          XT-PIC  ide1
```

- lsmod

```
/sbin/lsmod
```

```
     Module                  Size  Used by

     tulip                  37360   1 (autoclean)
```

Dans l'exemple ci-dessus la carte est basée sur un chipset Tulip, que l'adresse d'entrées / sorties esy 0xf800 et que l'IRQ est 10. Ces informations peuvent être utilisées pour insérer le module avec une adresse d'entrées/sorties différente en utilisant **modprobe** ou **insmod** ou en inscrivant les valeurs dans **/etc/modules.conf** (dans ce cas les paramètres seront utilisés au prochain démarrage).

Configuration des modems

Il nous faut d'abord détecter le modem. S'il s'agit d'un modem externe, il vous suffit de connaître le port série utilisé. Par contre, s'il s'agit un modem PCI, nous devons connaître l'adresse d'entrées sorties et l'interruption qu'il utilise.

Le modem

Pour les modems externes, allez à la partie suivante : « le port série ».

Vous pouvez détecter les modems PCI à l'aide de la commande **lspci** (l'exemple qui suit est tiré du PCI-Modem micro HOWTO).

```
lspci -v

          ----- snip -----

   00:0c.0 Serial controller:US Robotics/3Com 56K FaxModem Model 5610 (rev 01)
     (prog-if 02 [16550])

     Subsystem: US Robotics/3Com USR 56k Internal FAX Modem (Model 2977)

     Flags: medium devsel, IRQ 11
```

Notez que l'adresse d'entrées sorties est 0xe800 et l'IRQ 11. Nous pouvons utiliser ces informations et les attribuer à un port série.

Le port série

Le modem communique à travers une interface série. Les informations sont transmises à travers le réseau téléphonique sous la forme de séquences de bits (série) sur deux fils (entrant et sortant). Les informations entrantes sont repassées de série en parallèle pour le bus du PC et vice versa pour les données qui partent de l'ordinateur. La traduction est effectuée par une puce **UART** située au niveau port série sur la carte mère ou directement sur la carte PCI pour un modem interne.

Pour connaître les ports série détectés au démarrage, entrez la commande suivante :

```
 dmesg | grep ttyS



   /dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4

     /dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
```

Pour le moment, ces ports utilisent les adresses couramment utilisées par des ports série.

REMARQUE

Configuration matérielle

Pour les modems externes, ne considérez que les ports série avec les IRQ 3 ou 4. Les adresses d'entrée sorties ci-dessus sont également standard.

Le tableau ci-dessous présente les équivalents entre les ports COM de DOS et les ports série sur Linux :

DOS	Linux
COM1	/dev/ttyS0
COM2	/dev/ttyS1
COM3	/dev/ttyS2

Vous pouvez aussi utiliser la commande **setserial** pour rechercher les périphériques série. L'option -g affiche les ports série utilisés.

```
setserial -g /dev/ttyS[01]


  /dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
   /dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
```

Si vous avez un modem interne détecté par lspci page 38, vous devez garder en mémoire l'IRQ et l'adresse d'entrées sorties utilisées :

Adresses utilisées p 38 par le modem interne	
Adresse d'entrées sortie	0xe800
IRQ	11

Cette fois-ci nous allons utiliser **setserial** pour attribuer ces adresses à un port série virtuel :

```
setserial /dev/ttyS4 port 0xe800 irq 11 autoconfig
```

L'option *autoconfig* permet de paramétrer automatiquement l'UART. Vous pouvez sauver cette commande dans un script que vous appellerez *serial.rc* qui se chargera de configurer le port série à chaque démarrage.

On utilise généralement un lien symbolique **/dev/modem** pointant vers le port série utilisé par le modem.

Création manuelle du lien symbolique :

```
ln -s /dev/ttyS1 /dev/modem
```

On utilise aussi **setserial** pour configurer la vitesse du port série.

*Option de **setserial***	***Description***
spd_hi	Utiliser 56 kbaud au lieu de 38,4 kbaud
spd_vhi	Utiliser 115kbaud au lieu de 38,4 kbaud
spd_shi	Utiliser 230 kbaud au lieu de 38,4 kbaud
spd_warp	Utiliser 460kbaud au lieu de 38.4 kbaud
spd_cust	Paramètre la vitesse à partir du diviseur (vitesse = vitesse de base en baud / diviseur)
spd_normal	Utiliser 38,4 kbaud

Par exemple, si vous voulez paramétrer la vitesse du port série /dev/ttyS4 à 115 kb :

```
setserial /dev/ttyS4 spd_vhi
```

Configuration des connections

Le script de configuration wvdialconf du paquet wvdial recherche les modems série ou USB sur la machine et génère un fichier de configuration de base.

Exemple de fichier /etc/wvdial.conf :

```
[Dialer Defaults]
Modem = /dev/ttyS1
Baud = 115200
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 S11=55 +FCLASS=0
; Phone = <Target Phone Number>
; Username = <Your Login Name>
; Password = <Your Password>
```

Pour commencer, remplacez *Defaults* par le nom de votre fournisseur d'accès, par exemple MonFAI. Entrez également votre nom d'utilisateur, mot de passe et numéro de téléphone, puis tapez la commande suivante :

NdT : décommentez ces lignes en retirant les « ; ».

```
wvdial WorldISP
```

Vous pouvez également utiliser **minicom** pour votre connexion. Il se configure en utilisant l'option -s :

```
minicom -s


                                [configuration]

            Filenames and paths

            File transfer protocols

            Serial port setup

            Modem and dialing

            Screen and keyboard

            Save setup as dfl

            Save setup as..

            Exit

            Exit from Minicom
```

Tout ce qui concerne le port série se trouve dans *« Serial port setup »* et les informations de connexion communiquées par votre fournisseur s'inscrivent dans **« Modem and dialing »**.

Vous pouvez vous connecter à d'autres machines une fois que le modem est paramétré et fonctionnel, mais il faudra lancer **pppd** pour établir une connexion réseau. **pppd** va créer l'interface réseau ppp0 et assigner une adresse IP (fixe ou via DHCP). Des programmes comme **wvdial** s'en chargent tous seuls.

Winmodems

Si jamais la configuration précédente ne fonctionne pas, vous avez de fortes chance (façon de parler) d'avoir un

winmodem, qui va avoir besoin de pilotes supplémentaires. Pour avoir une définition fleurie des *winmodem*, jetez un oeil au winmodems-and-Linux-HOWTO.

Un winmodem qui peut fonctionner sous Linux est aussi appelé un linmodem (allez voir le linmodem-HOWTO si vous souhaitez en savoir un peu plus).

RNIS

RNIS c'est le réseau téléphonique classique numérisé jusqu'à la prise du client. C'est le Réseau Numérique à Intégration de Services, commercialisé sous le nom de Numéris en France par France Telecom. Au lieu d'une ligne analogique à 56kbits/s, le RNIS offre un certain nombre de canaux (suivant les contrats) pour les données, les canaux B à 64 kbit/s, et un canal de plus faible débit, le canal D qui gère les informations de liaison. Le service de base dans un certain nombre de pays est RNIS2 : 2 canaux B et un canal D.

Vous pouvez vous connecter en RNIS de différentes façons sous Linux. La plus simple est d'utiliser un routeur RNIS qui se charge de la connexion à votre place et qui se branche directement à votre PC ou votre réseau en Ethernet.

Vous aurez besoin d'une carte adaptateur si votre poste doit être directement connecté sur le réseau RNIS. Les détails concernant les différentes interfaces sortent du programme de ce cours mais vous trouverez en général :

- Interface de commandes compatible AT

  Avec les adaptateurs série et certains adaptateurs USB vous pouvez utiliser les commandes t exactement comme si vous utilisiez un modem. Dans ce cas, vous configurez l'adaptateur comme un modem. C'est plus simple, mais moins performant que d'autres méthodes et les données sont traitées comme celles d'un modem. Certains caractères doivent être échappés avec les modems parce qu'ils sont utilisés. Sur une ligne RNIS et en utilisant une autre méthode, les données ne sont pas à filtrer.

- Cartes PCI/ISA/PCCARD et isdn4linux

  Le moyen le plus efficace pour vous connecter à votre réseau RNIS est d'utiliser un adaptateur branché directement sur un bus de votre PC. L'objectif du projet isdn4linux est de masquer les détails de la connexion et de se présenter à l'utilisateur comme une interface réseau classique. Le paquet **isdn4k-utils** contient tous les outils nécessaires. Sur une distribution basée sur RedHat, le programme **system-config-network** gère la configuration.

ADSL

L'ADSL ou liaison numérique à débit asymétrique a très largement remplacé les connexions RNIS et les lignes louées (T1, etc.) pour les connexions haut débit. Tout comme pour le RNIS, l'utilisateur de Linux a plusieurs choix stratégiques. Le plus simple est d'utiliser un routeur ADSL autonome, auquel vous vous branchez directement en Ethernet. Dans ce cas, le routeur est traité comme n'importe quel routeur. De nombreux routeurs ADSL vendus actuellement sont des machines fonctionnant sous Linux. Si pour une raison ou une autre vous n'utilisez pas cette méthode, vous devrez acheter un modem ADSL et choisir un accès pouvant utiliser le protocole PPoE (Protocole Point à Point sur Ethernet). Vous trouverez différents projets de gestion de modems ADSL basés sur différents chipsets. Mais chacun utilise ses outils de configuration et son environnement.

- PPPoe

  L'ADSL n'est pas un protocole : il fonctionne plutôt comme un panier interconnectant des protocoles réseau sur PPP (Protocole Point à Point). Si l'équipement et le compte ADSL sont compatibles, les utilisateurs peuvent utiliser PPPoE pour réaliser la connexion. L'équipement ADSL se charge des couches en dessous de la session PPP et transmet les paquets PPP à travers ces couches. Les outils et les informations de connexion nécessaires se trouvent dans le paquet pppoe.

Configuration des imprimantes

Le processus d'impression est vu en détail dans le cours de préparation LPI102. Du point de vue matériel, les imprimantes sont automatiquement détectées au démarrage, ce que vous pouvez voir sur la sortie de **dmesg**.

L'impression sous Linux s'effectue en deux étapes. D'abord, les données brutes sont filtrées et mises à un format postscript. Ensuite, l'impression est prise en charge par le programme ghostscript ou **gs**.

Utilisation de printtool

printtool créé une entrée dans /etc/printcap. Les informations les plus importantes sont le chemin d'accès au filtre d'entrée (if), le répertoire de spool (sd) et le périphérique d'imprimante (lp).

Si **printtool** n'arrive pas à détecter le port parallèle correspondant à votre imprimante, utilisez **dmesg** pour voir ce que le noyau avait trouvé au démarrage.

L'exemple suivant présente une imprimante connectée sur le premier port parallèle /dev/lp0 :

```
parport0: PC-style at 0x378 (0x778) [SPP,ECP,ECPEPP,ECPPS2]

parport0: detected irq 7; use procfs to enable interrupt-driven operation.

parport_probe: succeeded

parport0: Printer, HEWLETT-PACKARD DESKJET 610C

lp0: using parport0 (polling)
```

Un fichier /etc/printcap :

```
# This file can be edited with the printtool in the control-panel.
##PRINTTOOL3## LOCAL cdj550 300x300 a4 {} DeskJet550 3 {}
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/lp/filter:
```

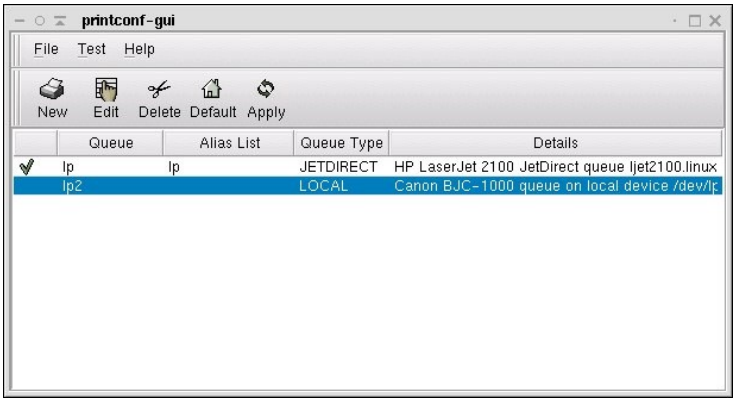


Illustration 2: printtool, l'interface GTK de configuration des imprimantes

Utilisation de cups

Cups est un nouvel outil d'administration et de configuration des imprimantes. Les fichiers de configuration sont situés dans **/etc/cups**. Le processus d'impression est le même sauf que **cups** utilise ses propres filtres que

l'on trouve dans **/usr/lib/cups**.

Cups se configure à partir d'une interface WEB, service qui utilise le port TCP 631.

Cups remplace également le service lpd.

REMARQUE
Les imprimantes locales sont détectées au démarrage du système qu'elles parallèles ou USB. Vous pouvez afficher ces informations avec la commande dmesg.

Cartes son

Il y a deux projets pour la gestion du son sous Linux : OSS (Open Sound System) et ALSA (Advanced Linux Sound Architecture). OSS est un projet commercial de gestion du son sur d'autres systèmes UNIX. Des pilotes modifiés ont été introduits au noyau à partir de la version 2.0.
ALSA est un projet beaucoup plus récent intégré seulement à partir du noyau 2.6. Pour les noyaux plus anciens, il faut recompiler le noyau pour intégrer la gestion d'ALSA, sauf pour certaines distributions qui ont très rapidement adopté ALSA, comme Suse.

Dans la plupart des cas, le son fonctionne à l'installation du système. La plupart des distributions Linux incluent des outils de configuration du son graphiques.

Détection de la carte son

Pour changer on va utiliser dmesg pour voir si le noyau a détecté une carte son :

```
 dmesg | grep -i  audio

NeoMagic 256AV/256ZX audio driver, version 1.1p

Initialized NeoMagic 256ZX audio in PCI native mode
```

REMARQUE
Il est possible que cette commande ne vous renvoie rien. Dans ce cas, regardez plus en détail la sortie de dmesg pour tenter de déterminer quel périphérique correspond à votre carte son.

Utilisation de sndconfig (objectif LPI101)

Pour configurer la carte précédente, nous devons trouver le module correspondant aux informations de **dmesg**. Avec OSS, ce module est associé à un fichier périphérique qui sera utilisé par les programmes, sound-slot-0 pour le premier périphérique de son.

Ce travail est fait automatiquement par des programmes de configuration. Nous allons voir **sundconfig**, qui est au programme du LPI101.

sndconfig est un programme RedHat de configuration des périphériques de son avec les modules OSS. Vous aurez sans doute besoin de l'installer car la plupart des distributions ne l'incluent plus à l'installation de base. Tapez :

```
sndconfig
```

Au lancement, un menu graphique (ncurses) apparaît et sndconfig vous demande s'il doit lancer la recherche de cartes sons sur votre machine. Allez sur « OK ».

Sur notre ordinateur, la carte a été détectée :

Neomagic Corporation | NM2360 [MagicMedia 256ZX Audio]

Si sndconfig n'a rien trouvé, il vous présente la liste de fabricants et des cartes gérées par OSS. Dans ce cas, utilisez lspci pour obtenir le modèle et vérifiez qu'il est compatible à la page http://www.opensound.com/osshw.html

Une fois la carte sélectionnée, **sndconfig** essaie de charger le module du noyau correspondant et de jouer un son (surprise !). Si tout a fonctionné, **sndconfig** met à jour **/etc/modules.conf** (vu dans la préparation de l'examen LPI102). Voici dans notre cas, la ligne ajoutée par sndconfig :

```
alias sound-slot-0 nm256_audio
```

Exercices et résumé

Révision (réponse p .Error: Reference source not found)

Oui ou non :

1. La partition racine d'un système Linux doit toujours se trouver sur un disque IDE : _____
2. Un système Linux peut utiliser n'importe quel périphérique USB du moment où le noyau a été compilé avec le support USB : _____

Glossaire

Terme	Définition
DMA	L'accès direct à la mémoire est utilisé par certains composants matériels pour effectuer des opérations de lecture / écriture directement en mémoire, sans utiliser le processeur.
Adresses d'entrées / sorties	Zones mémoires prédéfinies et réservées utilisées par les périphériques et le processeur pour des opérations de lecture / écriture.
IRQ	Signal envoyé par un périphérique au processeur pour exécuter une tâche en priorité : arrêt de l'exécution actuelle, exécution de la tâche prioritaire (par exemple saisie clavier) et reprise de la tâche précédente.
Allocation de ressources	Ensemble des ressources : DMA, adresses d'entrées/sorties et IRQ attribuées à un matériel
SCSI	Interface de transfert de données entre un matériel et le bus du PC. Par exemple, les périphériques peuvent être un disque dur, un lecteur de bandes, un lecteur Cédérom, un graveur de CD, un scanner...
USB	Connexion haute vitesse standard permettant de connecter des périphériques externes à l'ordinateur sans avoir besoin de redémarrer. Le principe est un contrôleur sur l'ordinateur avec un hub (concentrateur) initial. On branche ensuite les périphériques sur ce concentrateur ou d'autres concentrateurs, avec un maximum de 127 périphériques connectés (concentrateurs compris) par contrôleur.

Ressources

The Winmodems-and-Linux HOWTO

The Serial HOWTO

The Modem HOWTO

The Linux USB sub-system (http://www.linux-usb.org/)

SCSI terminology (http://www.scsita.org/terms/scsiterms.html)

The Linux 2.4 SCSI subsystem HOWTO

The Ethernet HOWTO

The Printing HOWTO

The Sound HOWTO

The isdn4linux project (http://www.isdn4linux.de/)

The Roaring Penguin PPPoE project (http://www.roaringpenguin.com/penguin/open_source_rp-pppoe.php)

Fichiers

Fichier	*Description*
/etc/isapnp.conf	Fichier de configuration de **isapnp** – voir **isapnp.conf(5)**
/proc/dma	Liste des canaux DMA utilisés
/proc/interrupts	Liste des interruptions utilisées
/proc/ioports	Liste de adresses d'entrées / sorties utilisées
/proc/pci	Informations sur le bus PCI
/etc/hotplug/usb.usermap	Liste des périphériques USB reconnus
/var/log/dmesg	Fichier journal des messages du noyau depuis le démarrage
/proc/scsi/scsi	Informations sur les périphériques SCSI – voir **scsi_info(8)**

Commandes

Commande	*Description*
dmesg	Affiche les messages du noyau depuis le démarrage
hotplug	Programme utilisé par le noyau pour traiter les évènements relatifs au matériel – voir **hotplug(8)**
isapnp	Programme utilisé pour initialiser les cartes ISA avant les noyaux 2.4 – voir **isapnp(8)**
lspci	Liste des périphériques PCI – voir **lspci(8)**
lsusb	Liste les périphériques USB – voir **lsusb(8)**
pnpdump	**pnpdump(8)** – affiche les informations sur les cartes ISA Plug-and-Play
scsi_info	**scsi_info(8)** – affiche la description des périphériques SCSI
setserial	**setserial(8)** – affiche ou règle les paramètres des ports série

usbmgr	Service utilisateur qui charge ou décharge les modules USB. C'est l'alternative à hotplug qui est généralement utilisée sur Debian.
usb.agent	Agent **hotplug** traitant les évènements relatifs à l'USB
usbmodules	**usbmodules(8)** – liste les pilotes du noyau qui devraient pouvoir gérer les périphériques USB connectés. usbmodules peut être utilidé par /sbin/hotplug ou un de ses agents (en général /etc/hotplug/usb/agent) quand les périphériques USB sont connectés à chaud.
wvdial	Outil utilisé pour les connexions PPP, voir **wvdial(1)**

Exercices

1. Utilisez **dmesg** pour voir les messages de **/var/log/dmesg**. Recherchez les mots clés USB, tty, ou ETH0
   - quels sont les noms des contrôleurs USB utilisés ?
   - Quelles interruptions sont utilisées par les deux premiers ports série ?
2. Consultez les fichiers suivants

   ```
   /proc/ioports
   /proc/interrupts
   /proc/pci
   /proc/dma
   ```
3. Le bus PCI
   - À partir des commandes **lspci -v** et **scanpci -v**, déterminez le type de votre carte ethernet
   - Vérifiez qu'il y a autant d'entrées de « bus » avec la commande lspci et /proc/pci
4. Outils USB
   - Déterminez le type de contrôleur USB sur votre système (xHCI) à partie des commandes **lsmod** et **lsusb** ;
   - Utilisez **usbmodules** pour afficher les modules du noyau pouvant utiliser l'interface
5. Périphériques SCSI
   - À partir du contenu du fichier **/proc/scsi/scsi** suivant, déduisez la sortie de la commande **scsi_info** (voir p.Error: Reference source not found):

```
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
Vendor: PHILIPS  Model: CDRW48A          Rev: P1.3
Type:   CD-ROM                           ANSI SCSI revision: 02
```

Gestion des disques

Pré-requis

- Expérience de la procédure d'installation d'un système sous Linux (voir Installation)

Objectifs

- Comprendre la différence entre les partitions primaires, étendue et logiques
- Utilisation des outils de partitionnement avant et après l'installation
- Installation et configuration des chargeurs de démarrage LILO et GRUB
- Comprendre les points de montage et le rôle du fichier **/etc/fstab**

Les disques

Sur un système Linux, les disques sont représentés par des fichiers dans le répertoire **/dev**. Le noyau communique avec les périphériques en utilisant une paire de numéros majeur et mineur. Les numéros majeurs sont listés dans **/proc/devices**. Par exemple, le numéro majeur pour le premier disque dur est **3** :

```
Block devices:
  1 ramdisk
  2 fd
  3 ide0
```

Les descripteurs de périphériques de **/dev** commencent par **hd** (IDE) ou **sd** (SCSI). Un lecteur de bande SCSI commencerait par **st**, etc. Les systèmes contiennent généralement plus d'un périphérique bloc, donc une lettre est ajoutée pour préciser le périphérique utilisé.

descripteur	*Périphérique bloc physique*
hda	Maître sur le premier contrôleur IDE
hdb	Esclave sur le premier contrôleur IDE
hdc	Maître sur le second contrôleur IDE
hdd	Esclave sur le second contrôleur IDE
sda	Premier disque SCSI
sdb	Second disque SCSI

Remarque : si vous insérez un disque SCSI avec un ID SCSI (par exemple 4) compris entre ceux de deux disques existants (par exemple 3 et 7), le descripteur du disque d'ID le plus haut se trouvera déplacé (par exemple il passera de *sdb* à *sdc*), ce qui peut causer quelques inconvénients non négligeables.

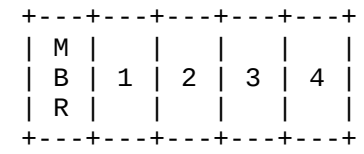
Partitions

Les disques peuvent être divisés en partitions. Celles-ci sont représentées par un numéro qui est ajouté au descripteur du disque :

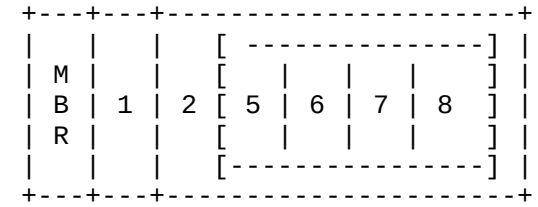
Descripteur	**Partition**
hda1	Première partition sur le premier disque
hda2	Seconde partition sur le premier disque
sdc3	Troisième partition sur le troisième disque SCSI

Il est possible de créer **4 partitions principales** sur les disques IDE. Une de ces partitions peut être **étendue**. Les **partitions étendues** permettent de créer autant de **partitions logiques** que désiré. On peut aller jusqu'à 64

partitions sur un disque IDE, 16 sur un disque SCSI.
Exemple 1 : 4 partitions principales



Exemple 2 : 2 partitions principales dont une partition étendue (no 2) contenant 4 partitions logiques (no 5 à 8)



Sortie type de fdisk -l :

```
Disque /dev/sda: 163.9 Go, 163928604672 octets
255 heads, 63 sectors/track, 19929 cylinders
Units = cylindres of 16065 * 512 = 8225280 bytes
Identifiant disque: 0x0009e33d

Périphérique Amorce    Début       Fin      Blocs    Id  Système
/dev/sda1               1         122      979933+   83   Linux
/dev/sda2             123        3769    29294527+   83   Linux
/dev/sda3            3770        7416    29294527+   83   Linux
/dev/sda4            7417       19929   100510672+    f   W95 Etendu (LBA)
/dev/sda5            7417        9848    19535008+   83   Linux
/dev/sda6            9849       10456     4883728+   83   Linux
/dev/sda7           10457       11064     4883728+   83   Linux
/dev/sda8           11065       12280     9767488+   83   Linux
/dev/sda9           12281       12523     1951866    83   Linux
/dev/sda10          12524       12766     1951866    82   Linux swap / Solaris
/dev/sda11          12767       19929    57536766    83   Linux
```

Sur ce disque, il y a 4 partitions principales. La quatrième partition (/dev/sda4) est étendue et n'est pas utilisée directement, elle contient seulement les 7 partitions logiques sur lesquelles se trouvent les systèmes de fichiers.
Remarque : faites bien la distinction entre les différents types de partition. De plus, il est important de comprendre la convention de nommage pour les disques IDE.

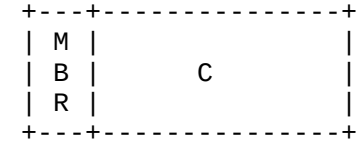
Les programmes de partitionnement

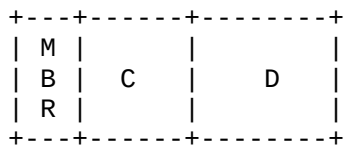
Avant l'installation (non nécessaire à l'examen)

- PartitionMagic
- Fips

Fips ne sait gérer que les partitions en *FAT16* et *FAT32*. *PartitionMagic* est beaucoup plus puissant, il sait traiter les partitions UNIX.
NdT : vous pouvez également utiliser *parted* et ses interfaces *Qparted* ou *Gparted* qui feront très bien le travail.

Il n'est pas nécessaire de partitionner si vous avez déjà deux partitions DOS/Windows C:\ et D:\ et que la partition D:\ est vide (et assez grande).





Pendant l'installation (non nécessaire pour la préparation de l'examen)

Pendant l'installation de Linux, vous pouvez créer vos partitions et les associer à des points de montage.
Les utilisateurs expérimentés peuvent le faire en deux étapes :

- Utiliser *fdisk* pour créer les partitions
- Associer un point de montage à chaque partition

La plupart des distributions utilisent un programme facile d'accès vous permettant de réaliser ces deux étapes en une seule :

- **diskdrake**(Mandrake)
- **DiskDruid** (RedHat)

Enfin, pour les débutants et les administrateurs systèmes pressés, les dernières distributions réaliseront automatiquement le plan de partitionnement.

Sur un système en cours de fonctionnement

Sur un système installé, utilisez **fdisk** pour faire vos nouvelles partitions. Voyons la syntaxe de fdisk sur un exemple :

Partitionner le premier disque dur

- fdisk /dev/hda
- tapez **m** pour l'aide, puis créez une nouvelle partition avec **n**.
- tapez **w** pour sauver les modifications sur le disque
- **redémarrez**

Voici les 4 points à suivre pour créer de nouvelles partitions. Le quatrième point est généralement oublié. Cela force le système à relire la table des partitions dans le **MBR**.
Remarque : vous devez formater la nouvelle partition avec **mkfs** ou **mke2fs** avant de pouvoir l'utiliser.

C'est la fin de notre petit tour sur les outils de partitionnement. Attaquons-nous maintenant aux chargeurs de démarrage.

Chargeurs de démarrage ("Master Boot Record")

Le MBR est placé sur le premier secteur du disque (512 octets) et contient la table des partitions ainsi qu'un chargeur d'amorçage. Au démarrage, le chargeur (bootloader) cherche la partition active dans la table de partitions et charge le premier secteur de cette partition.

Le chargeur de démarrage pour Linux : LILO

Grosso-modo, il y a trois choses à connaître sur LILO :

- LILO : installé dans le MBR, il charge la seconde étape de démarrage (second stage loader), en général **/boot/boot.b**
- **/etc/lilo.conf** : c'est le fichier de configuration de LILO. En voici les options :

- **boot** <nowiki>*</nowiki>où installer LILO (Le MBR du premier disque est /dev/hda)
- **install** deuxième chargeur (par défaut **boot.b**)
- **prompt** invite utilisateur (présente éventuellement un choix de système à démarrer)
- **default** nom de l'image lancée par défaut
- **timeout** utilisé conjointement avec prompt, pause avant le démarrage par défaut (en dixièmes de secondes)
- **image** <nowiki>*</nowiki>chemin d'accès au noyau à lancer (il est possible d'utiliser other pour lancer un autre chargeur)
- **label** <nowiki>*</nowiki>Nom de l'image, c'est le nom que l'utilisateur peut entrer au démarrage
- **root** <nowiki>*</nowiki>nom du périphérique contenant le système de fichiers racine /
- **read-only** <nowiki>*</nowiki>monter le système de fichiers en lecture seule pour que **fsck** fonctionne normalement
- **append** paramètres pour le chargement des pilotes inclus au noyau (statiques)
- **linear/lba32** ces options s'excluent mutuellement. Les deux options indiquent à LILO de lire le disque en utilisant l'adressage par bloc logique (LBA). linear est typiquement utilisé sur les disques très grands. lba32 est utilisé pour accéder à des données au delà des premiers 1024 cylindres au démarrage (voir aussi p 10(UNDEFINED REFERENCE: "1024 cylinder (gloss)")).

- **/sbin/lilo** : c'est le programme binaire qui installe LILO dans le MBR en fonction de la configuration dans /etc/lilo.conf.

Lancez **/sbin/lilo** à chaque modification de **/etc/lilo.conf**.

GRUB (Grand Unified Bootloader)

GRUB s'installe également dans le MBR. On peut modifier le contenu du MBR en utilisant le shell de grub **/sbin/grub** ou en utilisant le fichier de configuration de grub **/boot/grub/grub.conf** qui est lu par **/sbin/grub-install**.
Des informations détaillées sur GRUB sont disponibles dans les pages **info**.

Principales parties de /boot/grub/grub.conf

- **General** / Global
  - **default** : image qui démarre par défaut (la première entrée est 0)
  - **timeout** : temps d'affichage de l'invite en secondes
- **Image**
  - **title** : nom de l'image
  - **root** : racine du système de fichiers
  - **kernel** : chemin du noyau à partir du root défini précédemment (par ex. /vmlinuz)
  - **ro** : read-only – lecture seule
  - **initrd** : chemin d'accès au disque virtuel en mémoire (ramdisk)

Exemple de fichier grub.conf

```
default=0
```

```
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title   Linux (2.4.18-14)
        root (hd0,0)
        kernel /vmlinuz-2.4.18-14 ro root=/dev/hda5
        initrd /initrd-2.4.18-14.img
```

Gestion des partitions

Au démarrage, le fichier **/etc/fstab** attribue les points de montage aux périphériques de type bloc.

Format de /etc/fstab

- Périphérique ./. point-de-montage ./. système-de-fichier ./. options nombre-dump ./. nombre-fsck

Exemple de /etc/fstab

```
LABEL=/         /             ext2     defaults              1 1
LABEL=/boot     /boot         ext2     defaults              1 2
LABEL=/home     /home         ext3     defaults              1 2
/dev/fd0        /mnt/floppy   auto     noauto,owner          0 0
LABEL=/usr      /usr          ext2     defaults              1 2
LABEL=/var      /var          ext3     defaults              1 2
none            /proc         proc     defaults              0 0
none            /dev/shm      tmpfs    defaults              0 0
none            /dev/pts      devpts   gid=5,mode=620        0 0
/dev/hdc9       swap,pri=-1   swap     defaults              0 0
/dev/cdrom      /mnt/cdrom    iso9660  noauto,owner,kudzu,ro 0 0
```

La commande **mount** est utilisée pour rendre un périphérique accessible sur un répertoire (point de montage).
La syntaxe de **mount** est :

```
mount -t [SYSTÈME-FICHIERS] -o [OPTIONS] PÉRIPHÉRIQUE RÉPERTOIRE
```

Par exemple on montera un CD-ROM sur le point de montage /mnt/cdrom avec :

```
mount -t iso9660 /dev/cdrom  /mnt/cdrom
```

Sur un système en fonctionnement, le fichier **etc/fstab** sert aussi de raccourci pour assigner un périphérique à un répertoire :

```
mount /dev/cdrom
```

La commande **mount** lit **/etc/fstab** et déduit le système de fichier et le point de montage. Vous noterez que dans l'exemple de **fstab** précédent, nous accédons à certains périphériques via une étiquette (LABEL). Pour affecter une étiquette à un périphérique, utilisez la commande **tune2fs** :

```
tune2fs -L /usr/local /dev/hdb12
```

Quelques options de mount

- **rw, ro** : lecture/écriture et lecture seule
- **noauto** : le périphérique n'est pas monté au démarrage
- **users** : le périphérique est lisible et démontable par tous les utilisateurs
- **user** : seul l'utilisateur spécifié peut démonter le périphérique
- **owner** : le propriétaire du périphérique devient celui qui l'a monté
- **usrquota** : active les quotas utilisateurs sur le périphérique
- **grpquota** : active les quotas de groupe sur le périphérique

Remarque : pour l'examen, souvenez-vous que **mount -a** montera tous les systèmes de fichier de **/etc/fstab** qui ne sont pas déjà montés et qui n'ont pas l'option **noauto**.
La commande **umount** démonte un périphérique. Sa syntaxe est :

```
umount PÉRIPHÉRIQUE ou POINT-DE-MONTAGE
```

Aviez-vous remarqué que cette commande comporte une faute d'orthographe anglaise !
Par exemple, les deux commandes suivantes démontent le CD-ROM :

```
umount /dev/cdrom
```

ou

```
umount /mnt/cdrom
```

Gestion des quotas

Les outils de gestion des quotas sous Linux permettent aux administrateurs de mettre en place les quotas sans devoir redémarrer le système. Voici les étapes à suivre :

- Éditez **/etc/fstab** et ajoutez l'option **usrquota**
- Remontez la partition :

```
mount  -o remount  <nowiki><device> </nowiki>
```

- Initialisez les données sur les quotas : un fichier préliminaire **aquota.user** est créé à la base du répertoire

```
quotacheck -ca
```

- Éditez les quotas pour chaque utilisateur :

```
edquota -u <user>
```

Vous devez fixer une limite dépassable (soft) et une limite fixe (hard) pour l'espace disque utilisé (en blocs) et pour le nombre d'inodes pour chaque utilisateur. Le système autorisera l'utilisateur à dépasser la limite basse durant un certain temps (période de grâce). A l'expiration de cette période, la limite basse sera considérée comme limite haute.

- Appliquez les quotas

```
quotaon –a
```

Les utilisateurs peuvent connaître leur statut de quota avec la commande **quota**. L'administrateur système peut générer des rapports avec **repquota** ou **quotastats**.

Exercices et résumé

Questions de révision

Oui ou non :

1. Lorsque LILO est installé et que vous avez modifié le fichier lilo.conf, il n'est pas nécessaire de relancer /sbin/lilo :
2. Lorsque GRUB est installé et que vous avez modifié le fichier grub.conf, il n'est pas nécessaire de relancer grub-install :
3. Si la racine est au-delà de la limite du cylindre 1024, il est possible que votre système Linux ne démarre pas :
4. On ne peut utiliser les quotas que sur une partition dans son ensemble :

Glossaire

Terme	Description
partition	partie indépendante d'un disque dur qui peut soit être utilisée directement pour stocker des données, soit être divisée en partitions logiques. Toutes les partitions montées et manipulées par les systèmes d'exploitation

	ont des systèmes de fichiers séparés. De plus, elles sont considérées comme des périphériques indépendants. C'est pourquoi une partition sera parfois nommée "périphérique" parfois "système de fichiers".
partition principale	partition décrite dans l'un des enregistrements originaux de la table des partitions. Il n'y en a que 4 de disponible, donc les disques ne peuvent contenir que 4 partitions principales.
partition étendue	partition principale dont la description contient une liste de partitions, ce qui permet de créer des partitions nommées partitions logiques au delà des 4 primaires.
partition logique	partition contenue dans une partition étendue.
MBR	1er secteur du disque dur (512 octets) contenant le chargeur de démarrage et la table des partitions.
limite des 1024 cylindres	les anciens BIOS utilisent l'adressage CHS (pour Cylinder/Head/Sector = Cylindres / Têtes / Secteurs) : 10 bits pour le nombre de cylindres, 8 pour les têtes et 6 pour les secteurs. Ceci permet d'accéder à des disques d'une taille maximale de $(2^{10})*(2^8)*(2^6-1)*512$ octets ce qui correspond à 8,5 Go. Lorsque vous lancez **/sbin/lilo**, les adresses sont données sous la forme CHS pour que le BIOS puisse les lire, à moins que vous n'utilisiez les modes linear ou lba32. Si la deuxième partie du chargeur d'amorçage **/boot/boot.b** est plus loin que le 1024ème cylindre après le MBR, le système ne pourra pas démarrer.
chargeur d'amorçage (ou de démarrage)	code stocké sur les 512 premiers octets du disque, lu par le BIOS et utilisé pour démarrer un système d'exploitation.
GRUB	GRand Unified Bootloader : chargeur d'amorçage permettant, au même titre que LILO, de choisir au démarrage de son ordinateur entre plusieurs systèmes d'exploitation.
LILO	LInux boot LOader.
quotas	restrictions définies sur le nombre d'inodes et la taille de disque pour l'utilisation d'un périphérique par un groupe ou un utilisateur.
grâce	délai de tolérance pour les dossiers ou les fichiers qui ont dépassé la limite basse des quotas fixés.

Fichiers

Fichier	Description

/boot/boot.b	deuxième partie (second stage) du chargeur d'amorçage LILO.
grub.conf	fichier de configuration de grub. **NdT :** vous trouverez également **menu.lst**.
/etc/fstab	**fstab(5)** – Le fichier fstab contient des informations décrivant les différents systèmes de fichiers. fstab est uniquement lu par les programmes, l'écriture est laissée à l'administrateur, qui doit créer et maintenir lui-même ce fichier.
/etc/lilo.conf	fichier de configuration lu par le programme d'installation du chargeur de démarrage **/sbin/lilo**
/proc/devices	périphériques vus sur le système et numéros majeurs associés
aquota.user	fichier de description des quotas se trouvant à la racine d'un périphérique sur lequel les quotas sont activés

Commandes

Commande	Description ou **apropos**
/sbin/lilo	cf **lilo(8)** – installe le chargeur de démarrage
edquota	cf **edquota(8)** – éditeur des quotas utilisateurs
fdisk	**fdisk(8)** – gestionnaire de tables de partitions pour Linux
grub-install	cf **grub-install(8)** - installe GRUB sur votre disque
mount	cf **mount(8)** – monte un système de fichiers
quotas	cf **quota(1)** – affiche l'utilisation et les limites pour un périphérique
quotacheck	cf **quotacheck(8)** – balaye un système de fichiers, crée, vérifie et répare les fichiers de description des quotas
quotaon / quotaoff	cf **quotaon(8) / quotaoff** – active ou désactive les quotas
quotastats	cf **quotastats(8)** – programme permettant de consulter les statistiques sur les quotas
repquota	cf **repquota(8)** – rapport résumé sur les quotas pour un système de fichiers
tune2fs	cf **tune2fs(8)** – Ajuster les paramètres des systèmes de fichiers ext2...
usrquota,grpquota	n'est pas une commande mais une option dans **/etc/fstab** qui permet d'activer les quotas sur un

	périphérique
umount	**umount(8)** – démonter des systèmes de fichiers

Travaux pratiques

1. créez une nouvelle partition en utilisant **fdisk** sur **/dev/hda**

   ```
   fdisk /dev/hda
   ```

   Conseils : Pour créer une nouvelle partition, tapez **n**. Le type de la partition est par défaut **83** (Linux). Pour écrire la table de partition, tapez **w**. Le système doit relire la table de partition : REDÉMARREZ !

2. Formatez une de vos partitions

   ```
   mkfs <périphérique>
   i. Créez un répertoire data à la racine
      mkdir /data
   ii.Modifiez /etc/fstab et reliez le point de montage /data à la partition que vous
      avez précédemment formatée
      <périphérique>   /data  ext2  defaults   0 2
   ```

3. Forcez **mount** à lire **/etc/fstab** :

   ```
   mount –a
   ```

   Si ça ne fonctionne pas, vérifiez le fichier **fstab** et assurez-vous que **/data** existe **2(i)**.

4. Suivez les étapes décrites dans le cours pour activer les quotas sur ce périphérique

   À la deuxième étape, tapez **mount** et consultez sa sortie. Quelle option de **/etc/fstab** indique que les quotas sont activés sur le périphérique ?

   À la troisième étape, quel fichier est créé dans le répertoire **/data** ? _ *Avant de vérifier les quotas pour les utilisateurs normaux, ajoutez les permissions en lecture/écriture sur **/data** :chmod o+rw /dataDans les cas extrêmes, il pourrait être plus simple de redémarrer et de laisser les scripts de démarrage créer le fichier **aquota.user** ou **aquota.group**. Si les commandes **quotas**, **repquota** ou **quotastats** n'indiquent rien, assurez-vous que tout le monde a accès en lecture écriture sur **/data** : chmod a+rw /data***(FACULTATIF)** Monter un partage NFSLe formateur a un partage NFS. Déterminez quel répertoire est partagé et modifiez **/etc/fstab** pour monter ce partage sur **/mnt/nfs**. Utilisez l'option **noauto** pour ne pas monter le partage au démarrage.Changement de chargeur de démarrageDésinstallez LILO du MBR ou de la disquette :lilo –uadaptez l'exemple de **grub.conf** donné plus haut pour qu'il convienne à votre système.Installez GRUB sur votre disquette avec la commande :grub-install /dev/fd0

Réponses aux questions

1. **Non** : il faut relancer /sbin/lilo à chaque modification de /etc/lilo.conf
2. **Oui** : On n'installe qu'une fois GRUB
3. **Non** : la limite du cylindre 1024 affecte la deuxième partie du chargeur d'amorçage (**NdT** : dans /boot)
4. **Oui** : On ne peut pas fixer de quota sur un répertoire

Le système de fichiers Linux

État :
Pré-requis

- Avoir déjà installé Linux (voir Installation)

Objectifs

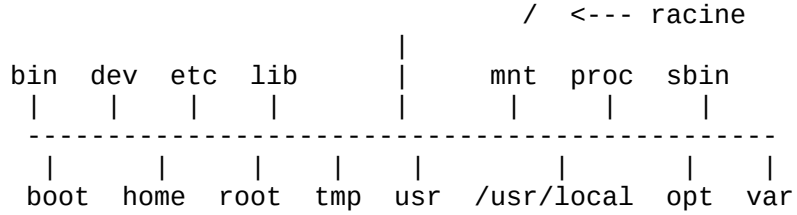
- Introduction aux répertoires de base et aux concepts issus de la FHS (Filesystem Hierarchy Standard) ou norme de la hiérarchie des systèmes de fichiers
- Formater une partition en EXT2 ou EXT3
- Contrôler l'espace libre par périphérique ou répertoire
- Comprendre les droits sur les fichiers et les répertoires dans le monde UNIX

La structure du système de fichiers

Un système de fichiers est similaire à une arborescence. La racine de l'arborescence est au sommet et les feuilles en dessous.
Comme nous l'avons déjà vu, vous devez donner un point de montage aux partitions que vous créez. En général, on le fait au moment de l'installation. Pour mieux comprendre où se trouvent les choses, jetons un œil à l'arborescence typique d'un système de fichiers Linux.

Au sommet de l'arborescence, on trouve la racine (*root*) : /. C'est un peu comme le C:\ sous DOS, sauf que C:\ est également le premier périphérique, alors que la racine peut correspondre à n'importe quel périphérique de votre système.

Répertoires de base




```
Répertoires dont le point de montage peut être un périphérique autre que celui de la
racine :

  boot  home  root  tmp  usr  /usr/local  opt  var
```

Les répertoires de base sont les premiers sous-répertoires sous la racine. Sur un système de type RedHat, ils sont installés par un paquet *rpm* du nom de *filesystem* :

```
$ rpm -ql filesystem
```

Au démarrage du système, le noyau monte la partition racine (/). Pour pouvoir monter et vérifier les autres partitions et systèmes de fichiers, un certain nombre de programmes comme *fsck*, *insmod* ou *mount* doivent être accessibles.
Remarque : les répertoires /dev, /bin, /sbin, /etc et /lib doivent être sur la partition racine. De plus, la racine doit contenir un répertoire /proc vide. Il est utilisé par le noyau pour informer sur le statut du système d'exploitation (processus, statistiques d'utilisation de la mémoire, etc.).

Rôle des répertoires de base

- /bin et /sbin : contient les binaires nécessaires au démarrage et les commandes essentielles

- /dev : fichiers périphériques ou fichiers spécifiques
- /etc : fichiers et répertoires de configuration spécifiques à la machine
- /lib : bibliothèques partagées pour les binaires de /bin et /sbin. Contient également les modules du noyau.
- /mnt ou /media : points de montage pour les systèmes de fichiers externes
- /proc : informations du noyau. En lecture seule sauf pour /proc/sys.
- /boot : contient le noyau Linux, le System.map (carte des symboles du noyau) et les chargeurs d'amorçage secondaires.
- /home (facultatif) : répertoires utilisateurs, avec, en général, une copie du contenu de /etc/skel.
- /root : répertoire de l'utilisateur *root*.
- /tmp : fichiers temporaires.
- /usr : *User Specific Ressource*. Contenu essentiellement statique et partageable. NdT : /usr est composé de sous-répertoires bin, sbin, lib, slib et autres qui contiennent l'essentiel des programmes et bibliothèques de votre système non essentielles ni nécessaires au démarrage.
- /usr/local ou /opt : programmes et bibliothèques supplémentaires. NdT : en général, c'est dans ces répertoires que l'on place les programmes qui ne font pas partie des paquets des distributions.
- /var : données variables comme les spool ou les journaux. Les sous-répertoires peuvent être soit partageables (comme /var/spool/mail) soit non partageables (comme /var/log).

/var/www, /var/ftp ou /srv : pages web ou fichiers ftp anonymes.

Formatage et cohérence des systèmes de fichier

Il est nécessaire de créer un système de fichiers pour organiser les données sur une partition. La procédure d'installation vous demande quel type de système de fichier vous souhaitez utiliser.
Vous avez un choix important de système de fichiers. Le format **ext2**, aussi connu comme le format natif pour Linux, est utilisé par défaut. Pour les programmes d'installation plus récents, ext3 (NdT et même aujourd'hui ext4 !) est utilisé par défaut. Ext3 est en fait un système ext2 journalisé.

Pour la mémoire paginée (swap), vous n'avez pas le choix, vous devez utiliser le système de fichiers *swap*.

Ext2

Regardons ext2 d'un peu plus prêt. Ext2 est constitué de blocs de 1024 octets (par défaut). Sans aller trop loin dans les détails, il y a 3 types de blocs : * les superblocs : répétés tous les 8193 blocs, les superblocs contiennent des informations sur la taille des blocs, les inodes libres, le dernier montage, etc. * les inodes : contient des pointeurs sur les blocs de données. Les 12 premiers blocs de données sont accédés directement. Si les données vont au-delà des 12 Ko, alors les inodes indirectes font le relai. * les blocs de données : correspondant aux fichiers ou aux répertoires, ils contiennent les données.

Outils de formatage

Les systèmes de fichiers gérés par le noyau nous permettent de lire les disques déjà formatés. Pour créer des systèmes de fichiers sur un système en fonction, nous devons installer et utiliser les outils correspondants. Pour ext2, le programme de formatage est mkfs.ext2 ou mke2fs. De la même façon, pour le système de fichiers de *Silicon Graphics* xfs, vous utiliserez mkfs.xfs que vous devrez sans doute installer.

Le programme mkfs agit comme une interface pour les outils spécifiques à chaque système de fichier. Sa syntaxe est :

```
mkfs –t <type de système> <périphérique>
```

Vous noterez de nouveau qu'ext3 est un système de fichiers ext2 auquel on a ajouté la journalisation (voir plus loin les exercices).

Exemple 1 : formatage en jfs

```
mkfs -t jfs /dev/hda12
```

Exemple 2 : formatage en ext2fs

```
mke2fs /dev/hda11    [or mkfs –t ext2 /dev/hda11]
```

Cohérence des systèmes de fichiers

Vous devez utiliser **fsck** quand le système de fichiers est endommagé ou corrompu, mais cela nécessite au minimum de démonter la partition ou de la monter en lecture seule.

Comme mkfs, fsck se comporte comme une interface qui détecte automatiquement le type de système de fichiers de la partition. Ensuite, les programmes fsck.ext2, fsck.ext2, etc. sont appelés. Comme EXT2 est le système de fichiers principal de Linux, il existe une commande e2fsck qui ne gère que ce système de fichiers.

Vous pouvez préciser le type de système de fichiers en utilisant la syntaxe suivante :

```
fsck –t <type de système> <périphérique>
```

Exemple : vérification d'un système de fichiers reiserfs sur le périphérique /dev/sdb10 :

```
fsck –t reiserfs /dev/sdb10
fsck.reiserfs /dev/sdb10
```

Commandes de déboguage des systèmes de fichiers

Les commandes debugs et dumpe2fs sont rarement utilisées mais elles peuvent être utiles pour vous donner des informations de bas niveau sur un système de fichiers ext2 ou ext3.

```
debugfs [ -b tailledebloc ] [ -s superbloc ] [ -f cmd_file ] [ -R request ] [ -V ] [ [ -w ] [ -c ] [ -i ] [ périphérique ] ]
```

Le programme debugfs est un débogueur de système de fichiers interactif. Il peut être utilisé pour examiner ou modifier l'état d'un système de fichiers ext2/3.

Une fois dans le shell de debugfs, vous pouvez utiliser les commandes internes pour parcourir l'arborescence, examiner les données d'inode, supprimer des fichiers, créer des liens, récupérer le journal d'ext3, etc. Cette commande est très puissante et doit être utilisée avec précaution, en général seulement pour essayer d'avancer alors que fsck a échoué.

```
dumpe2fs [ -bfhixV ] [ -ob superbloc ] [ -oB tailledebloc ] <périphérique>
```

dumpe2fs affiche les informations des superblocs et des groupes de blocs pour le système de fichiers sur le périphérique.

```
dumpe2fs /dev/hda1

dumpe2fs 1.35 (28-Feb-2004)
Filesystem volume name:   /boot1
Last mounted on:          <not available>
Filesystem UUID:          d741042c-3eaf-49ee-94c1-7dd8c5459764
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
needs_recovery sparse_super
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              25584
Block count:              102280
Reserved block count:     5114
Free blocks:              80564
Free inodes:              25537
```

```
First block:              1
Block size:               1024
Fragment size:            1024
Reserved GDT blocks:      256
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1968
Inode blocks per group:   246
Filesystem created:       Sat May  7 10:40:51 2005
Last mount time:          Sun May 29 04:08:01 2005
Last write time:          Sun May 29 04:08:01 2005
Mount count:              10
Maximum mount count:      -1
Last checked:             Sat May  7 10:40:51 2005
Check interval:           0 (<none>)
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
Default directory hash:   tea
Directory Hash Seed:      50108791-6a0a-41ff-9608-0485c993eaf9
Journal backup:           inode blocks


Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
  Block bitmap at 259 (+258), Inode bitmap at 260 (+259)
  Inode table at 261-506 (+260)
  0 free blocks, 1942 free inodes, 2 directories
  Free blocks:
  Free inodes: 27-1968

[....]
```

Contrôle de l'utilisation des disques

Avec mount et df

Ces deux outils fonctionnent au niveau des périphériques et non pas au niveau des répertoires. Les commandes **mount** et **umount** mettent à jour la liste des systèmes de fichiers montés dans **/etc/mtab**.
mount sans option affiche la liste des systèmes de fichiers actuellement montés, qui est équivalente au contenu de **/etc/mtab**. Le noyau garde également une trace des systèmes de fichiers montés dans **/proc/mounts**.

df' affiche également les systèmes de fichiers montés mais il affiche également l'espace disque utilisé et disponible. Par défaut, l'espace est donné en **blocs de 1K**.

```
df -h

Filesystem            Size  Used Avail Use% Mounted on
/dev/hda9             289M  254M   20M  93% /
/dev/hda2              23M  7.5M   14M  35% /boot
none                   62M     0   61M   0% /dev/shm
/dev/hda5             1.4G  181M  1.1G  13% /share
/dev/hda7             787M   79M  669M  11% /tmp
/dev/hda3             4.3G  3.4G  813M  81% /usr
/dev/hda6             787M  121M  627M  17% /var
//192.168.123.2/share   12G  8.8G  3.7G  71% /mnt/smb
```

Utilisation de du

Cette commande affiche l'utilisation du disque, mais cette fois en se basant sur les répertoires, et non plus les périphériques. Vous noterez que *du* ne peut pas afficher l'espace disponible sur le disque puisque cette information n'est accessible qu'au niveau du périphérique.
La commande suivante affiche la taille utilisée pour /etc en utilisant une unité compréhensible par les humains (-h) et n'affiche que le total (-s : summary).

```
du -sh /etc
62M     /etc/
```

Droits d'accès et attributs des fichiers

Changement des permissions et des propriétaires

Exemple :

```
-rw-rw-r-- 1 utilisateur groupe 13281 2007-03-09 11:00 STATS.ods
    /\           /\        /\
   chmod        chown  chown/chgrp
```

Comme vous pouvez le voir dans l'exemple précédent, **chmod** est utilisé pour modifier les permissions. Il y a trois types de propriétaire pour chaque fichier ou répertoire :

- u : un utilisateur existant dans /etc/passwd
- g : un groupe existant valide de /etc/group
- o : les autres

Exemple :

```
-rw-rw-r--    1 jade   sales         24880 Oct 25 17:28 libcgic.a
```

Modification des permission avec chmod :

```
chmod g=r,o-r libcgic.a
chmod g+w  libcgic.a
```

Modification de l'utilisateur et du groupe avec chown et chgrp :

```
chown  root  libcgic.a
chgrp  apache libcgic.a
```

Remarque : Les trois outils **chmod**, **chown** et **chgrp** ont l'option très pratique **-R** qui applique les modifications récursivement sur les répertoires et fichiers indiqués.

Notation symbolique et octale

Les permissions sont soit la lecture read=r, l'écriture write=w et l'exécution execute=x. Voici les valeurs octales correspondant à ces permissions :

Symbolique	Octal	Binaire
r	4	'100'
w	2	'010'
x	1	'001'

Ces permissions s'appliquent à l'utilisateur, au groupe et aux autres. Un élément a un ensemble de 3 permissions groupées pour chacune de ces catégories.

Exemple : Interprétation d'une permission 755 ou -rwxr-xr-x :

- utilisateur : rwx, c'est à dire 4+2+1 = **7**
- groupe r-x, c'est à dire 4+1 = **5**
- autres r-x, c'est à dire 4+1 = **5**

Les permissions standard

Les systèmes UNIX créent des fichiers et répertoires avec des permissions standard comme suit :

Fichiers | 666	-rw-rw-rw-
Répertoires | 777	-rwxrwxrwx

Umask

Chaque utilisateur peut définir un umask qui altère la permission standard. Pour obtenir les permissions d'un fichier, on soustrait (*) la valeur octale du **umask** à la valeur octale de la permission standard. Cette permission n'a pas de nom mais on pourrait l'appeler la permission effective.
(*) Même si la soustraction fonctionne dans la plupart des cas, vous devriez noter que la formule utilisée pour calculer les permissions est :

```
Permissions finales = Permissions standard ET(logique) NON umask
```

Sur les systèmes où les utilisateurs appartiennent à différents groupes, la valeur du umask peut être 002. Pour les systèmes où tous les utilisateurs sont dans le groupe users, le umask a toutes les chances d'être fixé à 022.

Le SUID

On peut assigner une permission spéciale à un programme de façon à ce qu'il soit toujours lancé en tant que le propriétaire du fichier. On appelle cette permission **SUID**, ce qui signifie « set user ID » (définition de l'identifiant utilisateur). La valeur symbolique pour cette permission est **s** et la valeur numérique **4000**.
Les outils d'administration peuvent avoir le **SUID** défini de façon à permettre aux utilisateurs de modifier des fichiers système.

Par exemple, la commande **passwd** peut être lancée par n'importe quel utilisateur pour lui permettre de changer son mot de passe, qui sera écrit soit dans **/etc/passwd**, soit dans **/etc/shadow**. Pourtant, ces deux fichiers appartiennent à l'utilisateur root et leurs permissions respectives sont **644** et **600**.

On résout ce problème en plaçant le bit SUID sur la commande passwd, ce qui la force à être lancée en tant que root et lui permet de modifier les fichiers /etc/passwd et /etc/shadow.

Le SUID sur passwd :

```
ls -l $(which passwd)
-r-s--x--x    1 root     root        18992 Jun  6  2003 /usr/bin/passwd
```

Remarque : dans la commande précédente, le SUID est affiché dans sa forme symbolique. Vous pouvez obtenir plus d'information sur un fichier en utilisant **stat** et en regardant la représentation octale des permissions :

```
stat /usr/bin/passwd
File: `/usr/bin/passwd'
Size: 18992    Blocks: 40   IO Block: 4096   regular file
Device: 305h/773d    Inode: 356680      Links: 1
Access: (4511/-r-s--x--x)  Uid: ( 0/ root)   Gid: ( 0/ root)
```

Attention : La permission SUID pose souvent des problèmes de sécurité. En voici un exemple :

1. Un utilisateur cherche à lire les mails du root. D'abord, il change sa variable d'environnement MAIL :

```
export MAIL=/var/spool/mail/root
```

1. Ensuite, il lance la commande mail :

```
 mail
 /var/spool/mail/root: Permission denied
```

Ça ne fonctionne pas, ça aurait été trop facile ! Mais si un administrateur s'est laissé convaincre de placer le bit SUID sur la commande mail, les commandes précédentes permettraient à n'importe quel utilisateur de lire les mails de n'importe quel autre (root compris) !
Les exemples suivants sont également dangereux, pourquoi ?

```
chmod 4755 /bin/cat
```

```
chmod u+s /bin/grep
```

Le SGID

Le **SGID** est une permission similaire au SUID mais pour le groupe. Sa valeur symbolique est **s** et sa valeur octale est **2000**.
Si vous placez le SGID sur un répertoire, les fichiers qui y seront ensuite créés appartiendront au groupe propriétaire de ce répertoire. Il n'est donc pas nécessaire d'utiliser **newgrp** pour modifier le groupe du processus utilisé pour la création du fichier (voir l'exercice plus loin).

Exemples :

```
chmod 2755 /home/data
chmod g+s /bin/wc
```

Le sticky bit

La permission sticky bit avec la valeur **1000** a l'effet suivant :

- appliqué à un répertoire, il empêche les utilisateurs de supprimer les fichiers dont ils ne sont pas propriétaires (idéal pour un répertoire partagé par un groupe) (**NdT :** le sticky bit est typiquement utilisé pour /tmp).
- appliqué à un fichier, il était utilisé pour charger le fichier en mémoire pour en accélérer les futurs accès ou exécutions. La valeur symbolique pour un exécutable est **t**, **T** pour un fichier non exécutable. Aujourd'hui comme le cache des systèmes de fichier est plus courant et plus performant, le sticky bit pour les fichiers tend à ne plus être supporté.

Exemples :

```
chmod 1666 /data/store.txt
chmod o+t /bin/bash
```

Les attributs sur les fichiers

Au côté des permissions standard, il existe un autre système pour modifier la façon dont on peut accéder à un fichier. Les attributs de fichiers ne s'affichent pas avec la commande **ls** mais avec la commande **lsattr**. On utilise la commande **chattr** pour paramétrer ou supprimer ces attributs.
Les attributs suivants existent. Veuillez noter les cas d'utilisation : * « A » : Quand on accède à un fichier avec l'attribut A, sa valeur *atime* (temps d'accès) n'est pas modifiée. Ceci évite un certain nombre d'accès disques, ce qui est plutôt pratique pour des fichiers temporaires. Attention cependant, certains outils comme *tmpwatch* se basent sur la valeur du atime pour déterminer l'utilisation récente d'un fichier, donc si le atime n'est pas mis à jour l'état du fichier sera mal interprété.

- « a » : L'accès en écriture aux fichiers avec l'attribut « a » est limité en ajout (append). Seul le super utilisateur ou un processus disposant de la capacité CAP_LINUX_IMMUTABLE peut définir ou supprimer cet attribut. L'utilisation la plus évidente est pour les fichiers journaux système, pour empêcher un intrus de retirer les traces de leur passage. Cependant, vous devriez garder à l'esprit qu'un intrus a besoin des droits root pour éditer ces fichiers journaux. Par conséquent, s'il les a, il peut retirer l'attribut « a », modifier les fichiers puis rétablir l'attribut « a ».
- « c » : un fichier avec l'attribut « c » est automatiquement compressé par le noyau. Vous accédez à vos données décompressées en lecture, et l'écriture sur le fichier compresse les données avant de les stocker sur le disque. Note : Même si l'attribut est défini sur un fichier et qu'il est affiché avec la commande lsattr, il n'est pas traité par les pilotes du noyau pour Ext2 ou Ext3.
- « D » : les modifications sur un répertoire avec l'attribut « D » sont écris de façon synchrone sur le disque. C'est l'équivalent de l'option dirsync de mount mais ici appliquée à un sous-ensemble de fichiers. Avec cet attribut, les opérations suivantes sont synchrones dans le répertoire : create, link,

unlink, symlink, mkdir, rmdir, mknod et rename.

- « d » : un fichier avec l'attribut « d » ne sera pas sauvegardé quand la commande dump sera lancée.
- « i » : un fichier avec l'attribut « i » (immutable = immuable) ne peut pas être modifié. Il est impossible de le supprimer ou de le renommer, ni de créer un lien vers ce fichier, ni d'écrire dans le fichier. Seul le superutilisateur ou un processus ayant la capacité CAP_LINUX_IMMUTABLE peut définir ou retirer cet attribut.
- 'j' : les données d'un fichier ayant l'attribut « j » sont inscrites dans le journal ext3 avant d'être inscrites dans le fichier si le système de fichiers est monté avec l'option « data=ordered » ou « data=writeback ». Lorsque le système de fichiers est monté avec l'option « data », l'attribut est sans effet puisque les données de tous les fichiers sont écrites dans le journal. Seul le super utilisateur ou un processus ayant la capacité CAP_SYS_RESSOURCE peut définir ou retirer cet attribut.
- « s » : lorsqu'un fichier avec l'attribut « s » est supprimé, ses blocs sont remis à zéro et réinscrits sur le disque. Note : comme pour l'attribut « c », cet attribut n'est pas traité par les pilotes de système de fichiers ext2 et ext3.
- « S » : les données d'un fichier ayant l'attribut « S » sont inscrites sur le disque de façon synchrones. C'est l'équivalent de l'option de « sync » de mount appliquée à un sous-ensemble de fichiers. Cet attribut est généralement utilisé pour les fichiers dits « cooked files », les fichiers dans lesquels les serveurs de bases de données stockent leurs informations. Ainsi, on contourne l'ensemble des deux systèmes de cache des pilotes du noyau et le cache de la base de données, optimisé, écrit directement sur le disque.
- « T » : l'attribut « T » n'est disponible qu'à partir des noyaux 2.6.x. Son but est d'indiquer le sommet de la hiérarchie, fonctionnalité utilisée par l'algorithme « Orlov block allocator ». Les nouvelles règles d'allocation de fichiers sur les systèmes de fichiers ext2 et ext3 rapprochent les sous-répertoires de façon à accélérer les accès sur une arborescence, si l'arborescence a été créée avec un noyau 2.6.
- « t » : le dernier bloc d'un fichier avec l'attribut « t » ne pourra pas être partagé avec d'autres fichiers (pour les systèmes de fichiers qui gèrent le « tail-merging »). C'est nécessaire pour des applications comme LILO qui accèdent directement au système de fichier et qui ne sont pas capables d'interpréter les blocs partagés. Note : au moment de l'écriture de la documentation, ext2 et ext3 ne font pas, en dehors de rustines très expérimentales, de « tail-merging ».
- « u » : lorsqu'on supprime un fichier ayant l'attribut "u", son contenu est sauvegardé. Ainsi, l'utilisateur peut demander sa récupération. Encore un attribut qui est géré par tout sauf le noyau.

Exemple :

```
# lsattr testfile
            ------------- testfile

# chattr +i testfile
# lsattr testfile
----i-------- testfile
# rm -f testfile
rm: cannot remove `testfile': Operation not permitted

# chattr -i testfile
# rm -f testfile
# ls testfile
ls: testfile: No such file or directory
```

Résumé et exercices

Questions de révision

Oui ou Non

1. Le répertoire /usr doit toujours se trouver sur la partition racine puisqu'il contient des données essentielles au démarrage
2. Il est possible que certaines opérations ne fonctionnent pas si le répertoire personnel d'un utilisateur se trouve ailleurs que dans /home
3. La création d'un système de fichiers sur une partitin détruit toujours toutes les données de cette partition

Voir les réponses

Glossaire

Terme	Description
Répertoires de base	Sous-répertoires se trouvant directement sous la racine
Bloc de données	Bloc utilisé pour stocker les données sur le système de fichiers. Ils sont référencés par les inodes
Sous-répertoires de la racine essentiels	Répertoires devant se trouver sur le système de fichiers racine pour le démarrage du système
ext2	Second extended file system : système de fichiers par défaut sur Linux (**NdT** : aujourd'hui remplacé par Ext3 puis Ext4)
ext3	Third extended filesystem : évolution de ext2 avec les mêmes fonctionnalités plus un système de journalisation
Permissions sur les fichiers	Attributs stockés dans l'inode du fichier pour les droits en lecture, écriture et exécution attribués au propriétaire (u), au groupe (g) ou aux autres (o) ainsi que des permissions plus avancées comme le SUID, le SGID et le "sticky bit"
FHS	Filesystem Hierarchy Standard (« norme de la hiérarchie des systèmes de fichiers »). Ce standard consiste en un ensemble de règles et de recommandations pour le placement des fichiers et des répertoires sur les systèmes de la famille UNIX. L'objectif de ces règles est d'améliorer l'interopérabilité des applications, outils d'administration ou de développement, et des scripts, ainsi que d'uniformiser la documentation sur ces systèmes. Voir
Inode	Bloc utilisé pour stocker les informations sur les fichiers, répertoires et les liens symboliques. La localisation des données des fichiers, les permissions, l'horodatage et le type de fichier (répertoire, fichier ou

	lien symbolique) se trouvent dans l'inode.
Orlov block allocator	Algorithme qui améliore les performances des systèmes de fichiers EXT2/EXT3. Disponible uniquement sur les noyaux 2.6. Cette fonctionnalité a été portée de BSD puis améliorée par Alexander Viro, Andrew Morton, et Ted Ts'o. Son auteur originel est Grigory Orlov. Cette fonctionnalité peut être activée avec la commande chattr sous Linux
Superbloc	Le superbloc est lu au montage du système de fichiers. Il contient différentes informations comme la taille des blocs du système de fichiers (1024 par défaut), le nombre d'inodes libres, le nombre de montage et le nombre maximum de montages utilisés pour déterminer si une vérification du système de fichiers doit être lancée

Fichiers

Fichier	Description
/etc/mtab	Fichier utilisé **par mount** pour conserver la trace des périphériques montés
/proc/mounts	Fichier utilisé **par le noyau** pour conserver la trace des périphériques montés

Commandes

Commande	Description (apropos)
chattr	Modifier les attributs des fichiers d'un système de fichiers Linux
chgrp	Changer le groupe propriétaire d'un fichier
chmod	Changer les permissions d'un fichier
chown	Modifier le propriétaire et le groupe d'un fichier
df	Indiquer l'espace occupé par les systèmes de fichiers
du	Évaluer l'espace disque occupé par des fichiers
e2fsck	Vérifier un système de fichiers Linux ext2/ext3/ext4
fsck	Vérifier et réparer un système de fichiers Linux
lsattr	Afficher les attributs des fichiers sur une partition Linux de type ext2
mke2fs	Créer un système de fichiers ext2/ext3/ext4
mkfs	Créer un système de fichiers Linux
mount	Tous les fichiers accessibles sur un système de fichiers

	Unix sont rangés sur une arborescence unique, la hiérarchie des fichiers dont la racine est /. Ces fichiers peuvent être répartis sur différents périphériques. La commande mount est utilisée pour attacher le système de fichiers d'un périphérique sur l'arborescence. À l'inverse, la commande umount détache ce système de fichiers de l'arborescence
umask	Outil utilisé pour configurer le masque de création de mode de fichier - voir help umask

Travaux pratiques

Systèmes de fichiers

1. Créez deux nouvelles partitions de plus de 50 Mo sur **/dev/hda** (ou /dev/sda) en utilisant **fdisk**.
   - Conseils :
     - Pour créer une nouvelle partition, tapez **n**. Le type de partition par défaut est **83** (Linux)
     - Pour écrire la table des partitions, tapez **w**.
     - Le système doit lire votre nouvelle partition : redémarrez votre ordinateur !
2. Formatez la première partition en **ext2** et la seconde en **reiserfs**.
   - Conseil : Le programme **mkfs** est une interface à **mkfs.ext2** ou **mkfs.reiserfs**, etc. Sa syntaxe est :

     ```
     mkfs –t <type_de_système> <périphérique>
     ```
3. Créez deux répertoires dans **/mnt** et montez vos deux partitions
   - ```
     mkdir /mnt/ext2
     mkdir /mnt/reiserfs
     ```
4. Vérifiez l'état de votre système :
   - Utilisez **mount** pour lister les périphériques montés. Les permissions configurés dans **fstab** sont également visibles.
   - Utilisez **df** pour voir le nombre total de blocs utilisés. L'option **-k** converti le nombre de blocs en kilo-octets (taille par défaut pour un bloc sur ext2)
   - Lancez **fsck** sur l'un des deux systèmes de fichiers créés. Le programme **fsck** est une interface à **fsck.ext2**, **fsck.ext3**, **fsck.reiserfs**, etc. Sa syntaxe est :

     ```
     fsck <périphérique>
     ```
5. Aller plus loin : passer de **ext2** à **ext3**
   - Passez le périphérique monté sur **/mnt/ext2** en ext3 avec **tune2fs**, qui ajoutera un journal au système de fichiers existant. Assurez-vous d'adapter votre fichier **/etc/fstab** pour prendre en compte cette modification.

     ```
     tune2fs –j /dev/hdaN
     ```
     À ce stade, le système a ajouté un **journal** à la partition **/dev/hdan**, qui est devenue une partition **ext3**. Ce procédé est non-destructif et réversible. Si vous montez un système de fichiers **ext3** en **ext2**, le fichier **.journal** est écrasé. Vous pouvez le recréer avec **tune2fs**.
     ```

Permissions sur les fichiers

1. Connectez vous en utilisateur normal (pas root). Créez un fichier en utilisant **touch** et vérifiez que les permissions effectives sur ce fichier sont **664**.
2. Changez le **umask** en **027**. Si vous créez un nouveau fichier, quelles seront les permissions effectives sur ce fichier ? Où configure-t-on la valeur du **umask** ? Suivant les systèmes, ce sera dans **/etc/profile** ou **/etc/bashrc**.
3. Créez deux utilisateurs :

   ```
   useradd util1
   useradd util2
   ```
   Initialisez leurs mots de passe avec **passwd util1** et **passwd util2**
4. Créez le groupe **ventes**

   ```
   groupadd ventes
   ```
5. Ajoutez les utilisateurs au groupe ventes

   ```
   gpasswd -a util1 ventes
   gpasswd -a util2 ventes
   ```
6. Créez un répertoire /news avec comme groupe propriétaire **ventes** et en lecture/écriture pour ce groupe.

   ```
   mkdir -m 770 /news ; chown .ventes /news
   ```
7. Placez le GID sur le répertoire /news

   ```
   chmod g+s /news
   ```
   Quelles sont les permissions symboliques (-rwxr-xr-x) sur /news ? (Utilisez ls -ld /news)

   Vérifiez qu'un membre du groupe ventes n'a pas besoin de taper **"newgrp ventes"** pour créer des fichiers avec les bonnes permissions. Les membres du groupe ventes peuvent-ils modifier n'importe quel fichier de ce répertoire ?
8. Ajoutez le **sticky-bit** sur **/news**. Vérifiez que seuls les propriétaires peuvent modifier leurs fichiers dans ce répertoire. Comment se présentent les permissions dans /news ?
9. En tant que root, placez le SUID root sur **xeyes**. Connectez-vous en utilisateur normal et vérifiez que le programme est lancé par le root.

   ```
   chmod u+s `which xeyes`
   ```
   Connectez-vous en utilisateur normal et lancez **xeyes**, puis :

   ```
   ps aux | grep xeyes
   ```
   Le programme devrait être lancé par root.

Réponses aux questions

1. **Non** : aucun programme placé dans /usr n'est nécessaire au démarrage du système
2. **Non** : on peut placer le répertoire personnel d'un utilisateur n'importe où sur le système
3. **Oui**

La ligne de commande

Pré-requis

- Aucun

Objectifs

- Introduction au shell bash et aux concepts de base comme l'exécution interactive de programmes
- Distinguer les variables locales et globales (exportées)
- Manipuler les données à partir des tubes et des autres opérateurs de redirection
- Comprendre les caractères de substitution et les expansions de nom de fichiers (file globbing)

Présentation

La ligne de commande est un moyen simple d'interagir avec un ordinateur. Le shell interprète les commandes tapées au clavier. Le prompt, ou l'invite de commande, qui se termine par un $ ou un # pour l'administrateur, indique que le shell attend les commandes de l'utilisateur.
Le shell est également un langage de programmation qu'on peut utiliser pour lancer des tâches automatiquement. Les programmes shell sont appelés des scripts.

Shells les plus courants

Le Bourne shell	/bin/sh
Le "Bourne again" shell	/bin/bash
Le Korn shell	/bin/ksh
Le C shell	/bin/csh
Tom's C shell	/bin/tcsh

le programme du LPI se concentre essentiellement sur le Bash, puisque c'est l'un des shells les plus couramment utilisés.

Le shell interactif

Les commandes shell suivent généralement la syntaxe suivante :
commande [options] {paramètres}

Afficher du texte à l'écran

Le bash utilise la commande **echo** pour afficher du texte à l'écran :

```
echo “Voici une courte ligne.”
```

Chemins relatif et absolu

Le shell interprète le premier "mot" de toute chaîne de caractère donnée comme une commande. Si cette chaîne est un chemin relatif ou absolu vers un fichier exécutable, alors le programme est exécuté. Si ce premier mot n'est pas précédé de /, alors le shell recherchera dans les répertoires définis dans la variable **PATH** et tentera de lancer la première commande qui correspondra à la chaîne.
Par exemple, si la variable **PATH** contient les répertoires /bin et /usr/bin, alors le système ne trouvera pas la chaîne xeyes puisque xeyes se trouve dans /usr/X11R6/bin/xeyes. Dans ce cas, vous devez entrer le **chemin absolu** de la commande :

```
/usr/X11R6/bin/xeyes
```

Mais on peut également utiliser le **chemin relatif**. Par exemple, si l'utilisateur est dans le répertoire où se trouve le programme xeyes, il peut taper :

```
./xeyes
```

Variables

Les variables du shell sont équivalentes aux variables utilisées dans les langages de programmation. Les noms de variables ne doivent contenir que des caractères alphanumériques. Par exemple, CREDIT=300 assigne tout simplement la valeur 300 à la variable nommée CREDIT.

1. initialiser une variable : **Nom_Variable=valeur (sans espace !)**
2. appeler une variable : **$Nom_Variable**

```
CREDIT=300
echo $CREDIT
```

On peut supprimer la valeur d'une variable avec la commande **unset**.

export, set et env

Il y a deux types de variables : les variables **locales** et les variables **globales** (exportées).
Les variables locales ne sont accessibles que sur le shell actif. Les variables exportées ou globales sont accessibles à la fois par le shell actif et par tous les processus fils lancés à partir de ce shell.

Les commandes **set** et **env** listent les variables définies.

Les commandes set et env

set	Liste toutes les variables
env	Liste les variables exportées

Une variable globale est globale dans le sens où elle est accessible par tous les processus fils.

Variable locale	Variable locale		
Parent	Fils	Parent	Fils
VAR=val	VAR=?	VAR=val	VAR=val

Exemple : exportez la variable CREDIT puis testez si elle est listée par **set** ou **env**.

```
export CREDIT
env | grep CREDIT
```

Lancez un nouveau shell (processus fils) (NdT : en tapant **bash**) et vérifiez que la variable CREDIT est accessible. Peut-on lancer n'importe quel SHELL et être sûr que la variable CREDIT est toujours déclarée ?

Liste de variables courantes pré-définies :

Variable pré-définie	Signification
DISPLAY	Utilisé par X pour identifié iù lancer une application cliente
HISTFILE	Chemin vers le fichier utilisateur .bash_history
HOME	Répertoire personnel de l'utilisateur
LOGNAME	Identifiant de connexion de l'utilisateur
PATH	Liste des répertoires utilisés par le shell pour exécuter des programmes quand une commande est entrée dans chemin
PWD	Répertoire de travail actuel

SHELL	Shell utilisé (bash dans la plupart des distributions Linux
TERM	Émulation de terminal utilisée

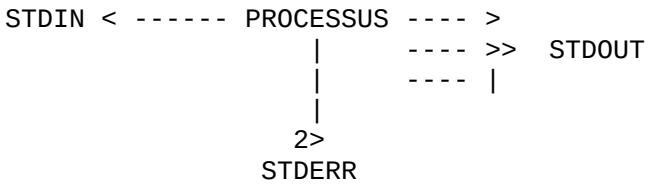
Variables spéciales

Les variables suivantes sont relatives à la gestion des processus :

Variable	Signification
$!	PID du dernier processs fils
$$	PID du shell actif
$?	valeur de sortie de la dernière commande : 0 = succès, 1 sinon

Entrées, sorties et redirections

Les processus UNIX ouvrent trois **descripteurs de fichiers** standards (NdT : correspondant aux **flux standards**) qui permettent de traiter les entrées et sorties. Ces descripteurs standards peuvent être redéfinis pour chaque processus. Dans la plupart des cas, le descripteur **stdin** (entrée standard) est le clavier, et les deux descripteurs de sortie, **stdout** (sortie standard) et **stderr** (l'erreur standard), sont l'écran.
Un processus et ses 3 descripteurs de fichiers



Valeurs numériques pour stdin, stdout et stderr

flux	Valeur numérique
stdin	0
stdout	1
stderr	2

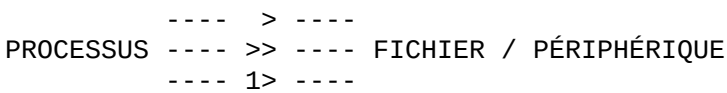
Redirection de la sortie standard

programme > fichier
Les données vont de gauche à droite.

```
fdisk –l > partitions.txt
```

Ceci lance **fdisk** et redirige la sortie vers le fichier partitions.txt. La sortie n'est pas visible a l'écran. Notez que le shell lit cette commande à partir de la droite : le fichier partitions.txt est d'abord créé s'il n'existait pas auparavant, écrasé dans le cas contraire car l'opérateur ">" est utilisé.
L'opérateur ">>" ajoute la sortie standard à un fichier.

Redirection de la sortie standard :



Redirection de l'entrée standard

programme < fichier
Dans ce cas, les données vont de droite à gauche. L'opérateur "<" ne peut être utilisé qu'avec **stdin** donc on ne peut pas l'utiliser avec les flux de sortie.

Si le fichier contient les instructions p, m et q (une instruction par ligne), alors dans l'exemple suivant fdisk affichera la table des partitions de /dev/hda, puis affichera l'aide puis quittera :

```
fdisk /dev/hda  < instructions
```

Redirection de l'entrée standard :

```
PROCESSUS ---- <  ---- FICHIER / PÉRIPHÉRIQUE
          ---- 0< ----
```

Redirection de l'erreur standard

programme 2> fichier_erreur
stdin, stdout et stderr sont représentés respectivement par 0, 1 et 2. Cela nous permet de choisir le flux d'erreur standard :

```
find / 2> /dev/null
```

Redirection de l'erreur standard :

```
PROCESSUS ---- 2> ---- FICHIER / PÉRIPHÉRIQUE
```

Les tubes

programme1 | programme2
Les tubes sont représentés par l'opérateur "|". Les données vont de gauche à droite. La figure suivante indique comment la sortie standard su premier processus est redirigée vers l'entrée standard du second processus.

Redirection à partir d'un tube :

```
PROCESSUS1 (stdout) ---- | ---- (stdin) PROCESSUS2
```

Exemple :

```
cat /var/log/messages | less
```

Note : Les redirections de sortie multiples sont analysées de droite à gauche, ainsi les commandes suivantes ne sont pas équivalentes :

```
commande  2>&1  > fichier
commande  >fichier  2>&1
```

La commande tee

commande | tee FICHIER
La commande **tee** est utilisé après un tube et prend comme paramètre un nom de fichier. La sortie standard de la commande précédente est alors écrite dans le fichier (NdT : comme avec ">") mais **tee** laisse le flux vers la sortie standard. La sortie standard est dupliquée.

Méta-caractères et guillemets

Les méta-caractères ont un sens spécial pour le shell. Ils sont la plupart du temps utilisés comme jokers, pour correspondre à plusieurs noms de fichiers ou de répertoires en utilisant un minimum de lettres.
Les caractères d'entrée (<), de sortie (>) et le tube (|) sont également des caractères spéciaux ainsi que le dollar ($) utilisé pour les variables. Nous ne les listerons pas ici, mais notez que ces caractères sont rarement utilisés pour nommer des fichiers standards.

Caractères génériques ou jokers

- Le caractère ***** remplace n'importe quel nombre de caractères :

```
ls  /usr/bin/b*
```
Liste tous les programmes commençant par "b".

- Le caractère **?** remplace n'importe quel caractère unique :

```
ls  /usr/bin/?b*
```
Liste tous les programmes ayant un "b" pour seconde lettre.

- **[]** est utilisé pour définir une plage de valeurs :

```
ls  a[0-9]
```
liste tous les fichiers commençant par un "a" suivis d'un chiffre en seconde position.
```
ls  [!Aa]*
```
liste tous les fichiers qui ne commencent pas par un "a" ni par un "A"

- **{chaîne1,chaîne2};** même si ce n'est pas simplement un joker de nom de fichiers, on peut l'utiliser pour filtrer des noms de fichiers :

```
ls  index.{htm,html}
```

Guillemets et caractères d'échappement

On peut annuler la signification des jokers en utilisant des caractères d'échappement, qui sont également des caractères génériques.
L'antislash (\), qu'on appelle le **caractère d'échappement**, annule le sens de tous les caractères génériques, en forçant le shell à les interpréter littéralement.

Les guillemets simples, ou apostrophes (' ') annule le sens de tous les caractères génériques sauf l'antislash.

Les guillemets (doubles) (" ") sont les guillemets faibles mais annulent la plupart des méta-caractères entourés à l'exception du tube (|), de l'antislash (\) et des variables ($var).

Les guillemets inversés `

Les guillemets inversés exécutent la commande entourée. L'exemple suivant définit la variable TIME à partir de la commande **date**.
```
TIME="La date d'aujourd'hui est le `date +%a:%d:%b`"
echo $TIME
La date d'aujourd'hui est le mar.:06:juil.
```
Pour exécuter des commandes, on peut également utiliser **$()**. La commande sera exécutée et traitée comme une variable.
```
TIME=$(date)
```

L'historique des commandes

Pour voir la liste des commandes que vous avez tapé, vous pouvez utiliser la commande interne de bash **history**.
```
history
1 ls
2 grep   500 /etc/passwd
```
history liste les commandes en cache ainsi que celles sauvées dans ~/.bash_history. Lorsque que l'utilisateur quitte le shell, les commandes en cache sont inscrites dans ~/.bash_history.
Vous pouvez récupérer les commandes tapées en utilisant les flèches directionnelles (haut et bas) de votre clavier. Vous pouvez également utiliser des raccourcis emacs qui vous permettent d'exécuter et même de modifier ces lignes.

Raccourcis clavier Emacs pour l'historique des commandes :

Ctrl+p	Ligne précédente (équivalent à la flèche haute)
Ctrl+n	Ligne suivante (Next) (= flèche basse)

Ctrl+b	Aller au caractère précédent sur la ligne (Back) (= flèche gauche)
Ctrl+f	Aller au caractère suivant (Forward) (=flèche droite)
Ctrl+a	Aller au début de la ligne (= touche Orig)
Ctrl+e	Aller en fin de ligne (End) (= touche Fin)

Le point d'exclamation **!** peut être utilisé pour relancer une commande.
Exemple :

```
!x                      lance la dernière commande de l'historique commençant par un
"x"
!2                      lance la commande n°2 de l'historique
!-2                     lance l'avant dernière commande
!!                      lance la dernière commande
^chaine1^chaine2        lance la dernière commande en remplaçant chaine1 par chaine2
```

Autres commandes

alias

Vous pouvez créer des alias pour des commandes nécessitant beaucoup de paramètres. La syntaxe d'**alias** est :
`alias monprog='commande [options]{paramètres}'`
alias tout seul donne la liste des alias déjà définis.

Auto-complètement de commandes

En appuyant sur **TAB**, le shell complète les commandes que vous avez commencé à taper.

<< est une redirection de EOF (fin de fichier)

Par exemple
`cat << stop`
accepte des entrées jusqu'à ce que "stop" soit tapé.

Commandes en chaîne

`commande1; commande2; commande3`
Ces trois commandes sont exécutées en séquence sans se préoccuper du résultat de la commande précédente.
`commande1 && commande2 &&  commande3`
Chaque commande sera exécutée seulement si la commande précédente a réussi (code de sortie 0) (**NdT** : && est un ET logique).
`commande1 || comande2 ||  commande3`
La commande suivante sera lancée uniquement si la précédente a échoué (code de sortie différent de 0) (**NdT** : || est un OU logique).

La commande "exec"

exec est une commande interne du bash utilisée pour lancer d'autres commandes. Normalement, quand vous lancez une commande, un sous-processus est créé. Si vous utilisez la commande **exec** pour lancer le nouveau programme, il remplace le processus qui l'a lancé, c'est à dire le shell pour un script ou un shell interactif. Quand la commande se termine, le contrôle n'est pas redonné au shell, mais retourne au processus qui avait lancé le shell.

```
echo $$
414

$ bash
```

```
$ echo $$
455

$ echo hello
hello
$ echo $$
455

$ exec echo hello
hello
$ echo $$
414
```

L'exemple précédent montre que le contrôle revient au second shell (processus 455) quand on lance la commande echo normalement, et au premier shell (processus 414) lorsqu'on utilise **exec**.

Les pages de manuel et la base whatis

Les pages de manuel sont organisées en sections, voici les principales sections que l'on devrait trouver dans une page de manuel :

NOM	Le nom du sujet du manuel suivi par une brève ligne de description
SYNOPSYS	La syntaxe de la commande
DESCRIPTION	Une description plus détaillée
OPTIONS	Un tour de toutes les options possibles et leur fonction
FICHIERS	Fichiers en lien avec le sujet (fichiers de configuration, etc.)
VOIR AUSSI	Autres pages de manuel en lien avec le sujet

La base de données **whatis** est un index de la description courte (section NOM) de l'ensemble des pages de manuel du système. Elle est actualisée par une tâche quotidienne cron. La base de données d'indexation **whatis** contient les deux champs suivants :

```
nom (clé)  –  description courte
```

La syntaxe de whatis est :

```
whatis <chaîne>
```

La sortie de la commande est la section NOM des pages de manuel pour lesquels la chaîne correspond à nom (clé).

La commande **man** permet également dinterroger la base de données d'indexation whatis en utilisant la syntaxe suivante :

```
man -k <chaîne>
```

Cette commande est équivalente à **apropos**. Contrairement à la commande whatis, ces commandes recherchent les correspondances dans les champs "nom" et "description courte". Si la chaîne correspond à un mot dans l'un des deux champs, la requête précédente renverra la section NOM entière.

Exemples : (les correspondances seront indiquées en gras quand je saurais le faire dans un espace de code...)

```
whatis lilo
lilo                (8)  - Installer le chargeur de démarrage
lilo.conf [lilo]    (5)  - Fichier de configuration pour lilo

man -k  lilo
grubby              (8)  - Outil en ligne de commande pour configurer grub, lilo, et
elilo
lilo                (8)  - Installer le chargeur de démarrage
lilo.conf [lilo]    (5)  - Fichier de configuration pour lilo
```

La **FHS** recommande de placer les pages de manuel dans **/usr/share/man**. Cependant, on peut rechercher dans d'autres répertoires en utilisant la variable d'environnement MANPATH, que l'on configure dans **/etc/man.config**. Chaque répertoire est ensuite divisé en sous-répertoires correspondants aux sections du manuel. (**NdT** : il ne faut pas confondre les sections **du manuel** avec les sections **d'une page de manuel** que nous venons de voir).

Sections du manuel

Section 1	Programmes exécutables ou commandes de l'interpréteur de commandes (shell)
Section 2	Appels système, par exemple mkdir(2)
Section 3	Appels de bibliothèque, par exemple stdio(3)
Section 4	Fichiers spéciaux (situés généralement dans /dev)
Section 5	Formats des fichiers et conventions
Section 6	Jeux
Section 7	Divers
Section 8	Commandes de gestion du système
Section 9	Sous-programmes du noyau

Pour accéder à une section du manuel en particulier, il faut utiliser la syntaxe suivante :

```
man N command
```

Exemples :

```
man mkdir
man 2 mkdir

man crontab
man 5 crontab
```

Exercices et résumé

Questions de révision

Oui ou Non

1. Si la variable PATH n'est pas correctement paramétrée, les utilisateurs ne peuvent lancer les programmes qu'en tapant leur chemin complet ou relatif : _On peut "tuber" la sortie standard STDOUT dans un fichier :_Une fois qu'un flux de donnée est passé dans un tube, ce flux n'est généralement plus visible sur STDOUT : _Toutes les commandes tapées dans le SHELL sont sauvegardées dans une base MySQL :_ Voir les réponses

Glossaire

Terme	Description
commandes chaînées	Plusieurs commandes tapées sur une seule ligne en utilisant des séparateurs? Suivant le séparateur, le shell lancera les commandes différemment
méta-caractère	Caractère qui n'est pas interprété littéralement par le shell parce qu'il a un sens
Expansion de nom de fichier (file globbing)	Terme utilisé pour le traitement de plusieurs fichiers

	avec l'utilisation des méta-caractères. Le nom vient du sous-programme **glob** sur les anciens shell UNIX qui était utilisé pour "développer" les caractères génériques donnés en ligne de commande
Redirection et tubes	Opérations qui manipulent les flux de données et les descripteurs de fichiers d'un processus. Une redirection implique un processus et un fichier, alors qu'un tube n'implique qu'un processus
stderr, stdin, stdout	Nom des descripteurs de fichiers disponibles pour chaque processus pour diriger les messages d'erreur, lire les flux d'entrée et écrire les sorties (hors erreurs)
caractères génériques	Les méta-caractères suivants utilisés pour correspondre à plus d'un caractère lorsqu'on travaille en ligne de commande : *, ?, "{}" ou "[]"

Commandes

Commande	Description (ou apropos)
alias	configure un alias pour une commande ou une séquence de commandes (voir **man builtins** ou **help alias**)
echo	Affiche du texte sur STDOUT
env	Liste les variables exportées (voir **man builtins** ou **help env**)
exec	Commande intégrée au shell utilisée pour lancer un programme mais en remplaçant le processus qui l'a appelé au lieu de créer un processus fils
export	exporte la valeur de la variable dans l'environnement pour les commandes exécutées par la suite (voir **man builtins** ou **help export**)
history	Affiche la liste de l'historique des commandes avec le numéro des lignes (voir **man builtins** ou **help history**)
tee	Lire depuis l'entrée standard et écrire sur la sortie standard et dans des fichiers
set	Liste toutes les variables dans l'environnement du shell courant (voir aussi **man builtins** et **help set**)

Travaux pratiques

Attention : Pour suivre les exercices, vous aurez besoin des commandes **uuencode** et **uudecode**. Vous les trouverez dans le paquet **shareutils**.

Stdin-stdout-stderr

Tapez les commandes suivantes et représentez si possible les séquences d'exécution sous la forme d'un

diagramme équivalent à ceux utilisés dans ce chapitre.

```
ls /etc ; df > /tmp/out.1
(ls /etc ; df) > /tmp/out.2

find /etc -type f  2> /dev/null | sort

tr [a-z] [A-Z] < /etc/passwd | sort > /tmp/passwd.tmp

cat /tmp/passwd.tmp | tr [A-Z] [a-z]
```

Ligne de commande

1. Affichez tous les fichiers de /usr/X11R6/bin qui ne commencent pas par un x

   ```
   ls /usr/X11R6/bin/[!x]*
   ```
2. La commande xterm a les options suivantes :

   Créez un alias pour que la commande **su** ouvre un nouveau xterm en couleur et qu'il vous demande le mot de passe root.

   ```
   alias su=″xterm -bg orange -fg brown -e su - &″
   ```
   Où placeriez-vous cet alias pour le conserver sur le système ?
3. Vous pouvez coder des fichiers (**NdT** : en fait, les convertir de binaires en texte) avec **uuencode**. Le fichier codé est redirigé vers stdout. **Par exemple** :

   ```
    uuencode /bin/bash super-shell > uufichier
   ```
   Cette commande convertit /bin/bash et produira un fichier nommé super-shell quand on utilisera **uudecode** sur le fichier codé uufichier.

   - Envoyez le fichier convertit uufichier par mail à un utilisateur local : vous pouvez soit utiliser uuencode et un tube | ou sauvegarder la sortie de uuencode dans un fichier uufichier, ce que l'on vient de faire, et utiliser un redirection de STDIN <.
   - Coupez uufchier en 5 :

   ```
   uuencode /bin/bash super-shell > uufichier
   split –b 150000 uufichier  fichier-coupe
   ```
   - Vous récupérerez des fichiers fichier-coupe.aa, fichier-coupe.ab, etc. Pour récupérer le fichier uufichier avec les données d'origine, faites :

   ```
   cat fichier-coupe.* > uufichier.new
   ```
   - Enfin, lancez uudecode sur ce fichier et regardez si ça fonctionne.

   ```
   uudecode   uufichier.new
   ```
   Cette commande devrait créer un fichier binaire nommé **super-shell**.
4. Quel outil indique le chemin complet vers un exécutable en examinant la variable PATH ?

Variables

1. Suivez les instructions suivantes :

   i. Assignez la valeur "virus" à la variable ALERT :

   ```
   ALERT=virus
   ```
   ii. Vérifiez que la variable est définie en utilisant la commande **set** :

   ```
   set |grep ALERT
   ```
   iii. La variable ALERT est-elle listée si vous utilisez **env** au lieu de **set** ? Ensuite, tapez "bash". Pouvez-vous accéder à votre variable ALERT ?

   ```
   bash
   ```

```
echo $ALERT
```

iv. Notez la value de ALERT : (est-ce vide ?)

v. Tapez **exit** ou (^D) pour retourner à votre session.

vi. Utilisez la commande **export** pour rendre la variable ALERT globale

```
export ALERT
```

vii. Vérifiez que la variable est globale avec **env**.

```
env | grep ALERT
```

viii. Lancez un nouveau shell bash et vérifiez que ALERT est définie dans ce shell :

```
bash
echo $ALERT
```

ix. Dans ce nouveau shell, modifiez la variable ALERT et exportez la :

```
export ALERT=green
```

x. Quittez ce shell. Quelle est la valeur de ALERT dans le shell d'origine ?

xi. À l'invite de commande, tapez les lignes suivantes :

- ```
  CREDIT01=300;CREDIT02=400
  for VAR in CREDIT01 CREDIT02;do echo $VAR;done
  ```
  Remarquez que la variable VAR est référencée avec $VAR.
- Relancez cette (deuxième) commande
- Relancez cette (deuxième) commande en remplaçant CREDIT01 par $CREDIT01

xii. Modifiez la variable PS1 pour afficher le chemin complet de votre répertoire de travail.

(**Indication** : la valeur de PS1 est [\u@ \W]\$, il vous suffit de remplacer \W par \w).

- ```
  PS1='[\u@\h \w ]\$ '
  ```

À quoi ressemble votre variable PS2 ?

Réponses aux questions

1. **Oui**
2. **Non** : STDOUT peut être redirigé vers un fichier
3. **Oui**
4. **Non** : les commandes sont en général stockées dans le fichier **.bash_history** du répertoire personnel de l'utilisateur.

Page consultée fois

Gestion des fichiers

État :
Pré-requis

- La ligne de commande
- Compréhension du système de fichiers Ext2

Objectifs

- Se déplacer efficacement sur le système de fichiers pour créer, supprimer et trouver des fichiers ou des répertoires
- Faire la distinction entre les liens physiques et les liens symboliques

Se déplacer dans le système de fichiers

Chemins absolus et relatifs
On peut accéder à un répertoire ou un fichier en donnant son chemin complet, qui commence à la racine (/), ou en donnant son chemin relatif partant du répertoire courant.

- **Chemin absolu** :
  - indépendant du répertoire de travail de l'utilisateur
  - commence par /
- **Chemin relatif** :
  - dépend de l'endroit où se trouve l'utilisateur
  - ne commence **pas** par /

Comme pour tout système de fichiers structuré, un certain nombre d'outil vous aident à parcourir le système. Les deux commandes suivantes sont des commandes internes du shell :

- **pwd** : (Print Working Directory) affiche le répertoire actuel en chemin absolu
- **cd** : la commande pour changer de répertoire (Change Directory)

Recherche de fichiers et de répertoires

Dans cette partie, nous décrirons les commandes **find**, **which**, **whereis** et **locate**.

find

Syntaxe :

```
find <REPERTOIRE> <CRITERE> [-exec <COMMANDE> {} \;]
```

Le paramètre REPERTOIRE indique à **find** d'où lancer la recherche et CRITERE peut être le nom du fichier ou du répertoire que nous recherchons.
Exemples :

```
            find /usr/X11R6/bin -name  ¨x*¨.
find / -user 502
```

Les lignes correspondantes sont listées sur la sortie standard. On peut également lancer une commande sur cette sortie, comme supprimer le fichier ou changer le mode de permission, en utilisant l'option **-exec** de **find**. Par exemple, pour supprimer tous les fichiers appartenant à l'utilisateur 502 :

```
find / -type f -user 502 –exec rm –f {} \;
```

xargs

On voit souvent en **xargs** le compagnon de **find**. En fait, **xargs** traite chaque ligne de la sortie standard comme paramètre pour une autre commande. On pourrait utiliser **xargs** pour supprimer tous les fichiers appartenant à un utilisateur avec :

```
find / -type f -user 502 | xargs rm –f
```

Remarque : Certaines commandes comme **rm** ne savent pas traiter les paramètres trop longs. Il est parfois nécessaire de supprimer tous les fichiers d'un répertoire avec :

```
ls |xargs rm -f
```

Options courantes pour find

-type	spécifier le type de fichier (répertoire, lien symbolique, fichier, etc.)
-name	nom du fichier
-user	propriétaire
-atime, ctime, mtime	dates d'accès, de création et de modification (multiples de 24 heures)
-amin, cmin, mmin	moments d'accès, de création et de modification (multiples de 1 minute)
-newer FICHIER	fichiers plus récents que FICHIER

locate

Syntaxe :

```
locate <CHAINE>
```

locate liste tous les fichiers et répertoires qui correspondent à l'expression.

```
locate X11R
```

La recherche avec **locate** est très rapide. En fait, **locate** interroge la base de données **/var/lib/slocate/slocate.db**. Cette base de données est tenue à jour par une tâche quotidienne cron (cronjob) qui lance **updatedb**.

Le fichier **/etc/updatedb.conf** est lu par **updatedb** lorsqu'il est lancé manuellement pour déterminer les systèmes de fichiers et le répertoires dont il ne doit pas tenir compte (montages NFS et /tmp par exemple).

which

Syntaxe :

```
which chaîne
```

which retourne le chemin complet de la commande dont le nom est **chaîne** en parcourant les répertoires définis dans la variable **PATH** de l'utilisateur uniquement.

whereis

Syntaxe :

```
whereis chaîne
```

Cette commande affiche le chemin absolu des sources, binaires ainsi que de les pages de manuel pour les fichiers correspondant à **chaîne** en se basant sur le **PATH** ainsi que sur des répertoires couramment utilisés.

ls

Options courantes pour ls

-i	affiche l'inode

-h	avec -l, afficher les tailles dans un format lisible par un humain (par exemple 1K 234M 2G)
-n	affiche les valeurs numériques des identifiants du propriétaire (UID) et du groupe (GID)
-p	ajoute l'indicateur « / » aux répertoires
-R	affiche récursivement les sous-répertoires
-S	trie selon la taille des fichiers
-t	trie selon la date de modification
-u	trie la date de dernier accès (**NdT** : voir **man ls** car -u agit différemment suivant l'option avec laquelle elle est couplée)

Gestion des répertoires

Création d'un répertoire avec mkdir

Vous pouvez définir les droits avec l'option **-m** de **mkdir**. Une autre option couramment utilisée et utilile de **mkdir** est **-p** qui crée les sous-répertoires quand c'est nécessaire.
Exemple :

```
mkdir –p docs/programmes/versions
```

Ndt : la commande précédente crée le répertoire programmes s'il n'existait pas, ce que mkdir ne ferait pas sans l'option -p.

Suppression des répertoires

On utilise soit **rmdir** soit **rm -r** pour supprimer les répertoires. En tant que root, vous aurez peut-être à spécifier l'option **-f** pour forcer la suppression de tous les fichiers.
NdT : **rmdir** supprime un répertoire vide alors que **rm -r** supprime un répertoire et son contenu.

Remarques :

- rm –rf /rep1/* supprime tous les fichiers et sous-répertoires et laisse le répertoire rep1 vide
- rm –rf /rep1/ supprime tous les fichiers et sous-répertoires y compris rep1

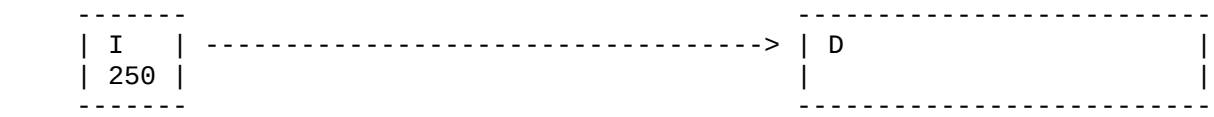
Utilisation de cp et mv

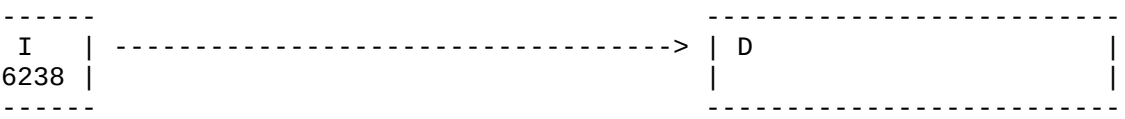
cp

Syntaxe:

```
cp [options] fichier1 fichier2
cp [options] fichiers répertoire
```

Il est important de noter que **cp** *fichier1 fichier2* crée une nouvelle copie de *fichier1* et laisse *fichier1* inchangé.
Illustration : fichier1 avec l'inode 250 est copié sur fichier2, en dupliquant les données sur un nouveau bloc et en créant une nouvelle inode 6238 pour fichier2.





Vous pouvez également copier plusieurs fichiers dans un répertoire, à partir d'une liste ou en utilisant des caractères génériques. Le tableau suivant donne la liste des options les plus courantes.

Options les plus courantes pour cp

-d	ne suit pas les liens symboliques (quand il est utilisé avec -R
-f	force
-i	interactif : demande confirmation avant d'écraser
-p	conserve les attributs des fichiers
-R	copie récursivement les répertoires

Remarques :

- cp –r /rep1/* /rep2/ copie tous les fichiers et sous-répertoires en omettant rep1
- cp –r /rep1/ /dir2/ copie tous les fichiers et sous-répertoires y compris rep1

mv

Syntaxe :

```
mv [options] anciennom nouveaunom
mv [options] source destination
mv [options] source répertoire
```

La commande **mv** peut à la fois déplacer et renommer les fichiers et les répertoires. Si *anciennom* est un fichier et *nouveaunom* un répertoire, le fichier *anciennom* est déplacé dans ce répertoire.
Si la *source* et la *destination* sont sur le même système de fichiers, alors le fichier n'est pas copié, mais les informations de l'inode sont mises à jour pour tenir compte du nouveau chemin. Les options les plus courantes de **mv** sont **-f** pour forcer l'écrasement et **-i** pour demander confirmation à l'utilisateur.

Liens symboliques et liens physiques

Liens symboliques

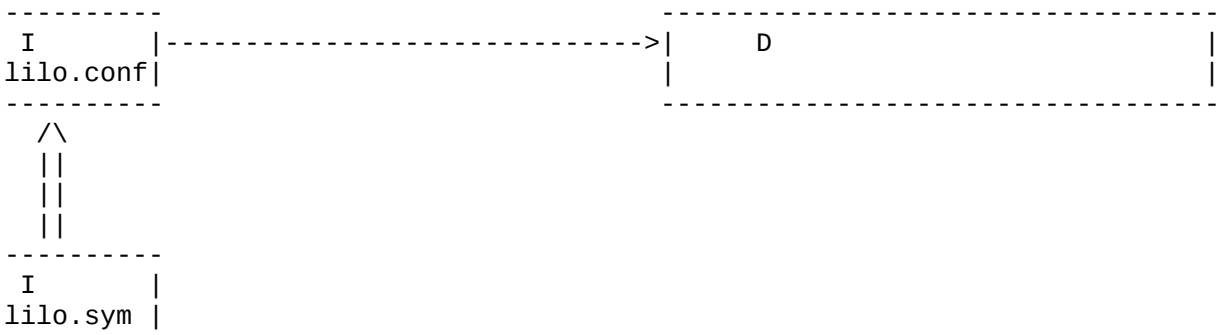
Un lien symbolique vers un fichier ou un répertoire crée une nouvelle inode qui pointe vers le même bloc de données.

```
ln -s lilo.conf lilo.sym
```

Voici ces deux fichiers. Notez que le nombre de références est 1 pour chacun des fichiers.

```
-rw-------   1  root  root  223 Nov  9 09:06   lilo.conf
lrwxrwxrwx   1  root  root  9 Nov  9 09:06  lilo.sym -> lilo.conf
```

Illustration : Lien symbolique vers un fichier



```
------------
```

Les liens symboliques peuvent pointer vers des fichiers ou répertoires présents sur un autre système de fichier.

Liens physiques

Un lien physique est un nom supplémentaire pour une même inode. Ainsi, le nombre de références s'accroît de un à chaque nouveau lien physique.

```
ln lilo.conf lilo.link
```

Dans la liste suivante, notez que le nombre de références est de 2 et que les deux fichiers ont la même taille (**NdT** : ainsi que la même inode, ce que vous pouvez vérifier avec **ls -i**). Dans les faits, ces deux fichiers sont parfaitement identiques.

```
-rw-------  2 root  root  223  Nov  9 09:06 lilo.conf
-rw------- 2 root  root  223  Nov  9 09:06 lilo.link
```

Les liens physiques ne peuvent être créés que sur le même système de fichier.

touch et dd

touch

touch est une commande permettant de créer et de modifier les fichiers.
Syntaxe :

```
touch {options} fichier(s)
```

Le fichier est créé s'il n'existait pas. Vous pouvez également changer la date d'accès avec l'option **-a**, la date de modification avec **-m** et **-r** copie les attributs de date d'un autre fichier.
Exemples :

- crée deux nouveaux fichiers :

  ```
  touch fichier1.txt fichier2.txt
  ```

- copie les attributs de date de *lilo.conf* sur *monfichier* :

  ```
  touch monfichier -r /etc/lilo.conf
  ```

- crée un fichier nommé *-erreurs* en utilisant l'option -- :

  ```
  touch -- -erreurs
  ```

dd

Cette commande copie un fichier avec une taille de bloc modifiable. On peut également l'utiliser pour effectuer des conversions, comme avec **tr**. Les options principales de **dd** sont **if=** (fichier d'entrée : input file), **of=** (fichier de sortie : output file) et **conv=** (conversion). conv peut prendre les symboles lcase (minuscules), ucase (majuscules) et ascii.
Exemple :

```
dd if=/mnt/cdrom/images/boot.img of=/dev/fd0
```

Notez que contrairement à **cp**, la commande **dd** copie des portions de données brutes d'un périphérique. Par conséquent, **dd** préserve le système de fichier sous-jacent. **cp** se contente de traiter des données et les transfère d'un système de fichier à un autre.
*** FIXME : ajouter les schémas ***

Résumé et exercices

Questions de révision

Oui ou non

1. La commande **cd -** vous replace dans le répertoire où vous vous trouviez précédemment ? _La commande **cd ~** (en bash) est la commande la plus courte pour aller dans votre répertoire personnel ?

_On peut créer deux nouveaux répertoires /rep1./rep2 avec **mkdir** sans aucune option : _la commande **updatedb** met à jour la base de données de **locate** :_La syntaxe pour créer un lien symbolique nommé LIEN pointant vers FICHIER est : _ *ln -s LIEN FICHIER*Les commandes **cd /etc** et **cd ./etc** sont toujours équivalentes :_Voir les réponses

Fichiers

Fichier	Description
/etc/updatedb.conf	fichier de configuration pour la commande updatedb
/var/lib/slocate/slocate.db	base de donnée pour **locate** (ou **slocate** : "secure locate")

Commandes

Commande	Description (apropos)
cd	change de répertoire courant - voir **help cd**
cp	cp(1) – Copie des fichiers et des répertoires
dd	copie et convertit des fichiers. Généralement utilisé pour copier le contenu d'un périphérique disque sur un autre périphérique ou un fichier
find	find(1) – Recherche des fichiers dans une hiérarchie de répertoires
ln	ln(1) – Crée des liens entre des fichiers
locate	commande utilisée pour rechercher des fichiers et des répertoires dans la base de données locate
ls	ls(1) – Affiche le contenu de répertoires
mkdir	mkdir (1) - Crée des répertoires
mv	Déplace ou renomme des fichiers
pwd	pwd (1) - Affiche le nom du répertoire de travail actuel
rm	rm (1) - Effacer des fichiers et des répertoires
touch	Crée un fichier vide ou modifie l'horodatage d'un fichier s'il existe
updatedb	commande utilisée pour mettre à jour la base de données **locate**
whereis	whereis(1) – Recherche les fichiers exécutables, les sources et les pages de manuel d'une commande
which	which(1) – localise une commande

Travaux pratiques

Navigation

1. Créez un nouveau répertoire *etc* dans */tmp*

   ```
   mkdir /tmp/etc
   ```
2. Créez un fichier nommé *nouveau* dans */tmp/etc* (avec touch, cat ou vi).
3. Allez à la racine (cd /)
4. Testez les commandes qui affichent le contenu de *nouveau* :

   ```
   cat etc/newfile
   cat /etc/newfile
   cat tmp/etc/newfile
   cat /tmp/etc/newfile
   ```
5. Supprimez le répertoire */tmp/etc* avec **rmdir**. Refaites le point 1 puis supprimez */tmp/etc* avec **rm**.

Faire de la place sur le système de fichier

Dans le but de récupérer de l'espace sur le périphérique qui contient le répertoire */usr/share/doc*, nous allons copier les données de */usr/share/doc* sur un périphérique inutilisé. Ensuite, nous ferons de la place en supprimant le répertoire */usr/share/doc* et nous créerons un lien symbolique de */usr/share/doc* vers le nouveau périphérique.

1. Créez un répertoire */spare* sur lequel nous monterons notre disque de rechange (l'une des partitions crées au cours des exercices précédents ferait parfaitement le travail).

   ```
   mkdir /spare
   mount <périphérique> /spare
   ```
2. Vérifiez avec **df -h /spare** et **du -hs /usr/share/doc** que la partition est suffisamment grande pour contenir les données existantes.
3. Ensuite, copiez le contenu de */usr/share/doc* sur */spare/*

   ```
   cp -a /usr/share/doc /spare
   ```
4. Vérifiez que les données ont bien été copiées, puis modifiez */etc/fstab* pour rendre le périphérique accessible au démarrage du système.
5. Supprimez */usr/share/doc* et créez un lien symbolique de */usr/share/doc* vers */spare/doc*

   ```
   ln -s /spare/doc /usr/share/doc
   ```
6. (facultatif) Faites la même chose avec **/home**. Rencontrez vous des problèmes ?

Recherche de fichiers

1. Copiez */etc/lilo.conf* sur */etc/lilo.conf.bak*
   i. retrouvez votre nouveau fichier avec **find**
   ii. utilisez **locate** pour retrouver ce nouveau fichier
   iii. mettez à jour la base de données **locate** et réessayez le ii.

Stratégie de sauvegarde (première étape)

1. Recherchez tous les fichiers de votre répertoire personnel qui ont été modifiés dans les dernières 24 heures.

   ```
   find /home –mtime –1 |tee list1 |wc ––lines (-1 signifie moins d'une journée)
   ```
   Nous verrons les outils d'archivage dans le LPI 102, mais la sortie de find peut être tubée directement

dans **cpio**.

Réponses aux questions

1. **Oui**
2. **Non** : cd sans paramètre permet également d'aller dans le répertoire utilisateur
3. **Non** : vous devez utiliser **mkdir -p** pour créer des répertoires qui contiennent des sous-répertoires
4. **Oui**
5. **Non** : la commande correcte est **ln -s FICHIER LIEN**
6. **Non** : les chemins commançant par un slash sont des chemins absolus

Page consultée fois

Gestion des processus

État :
Pré-requis

- La ligne de commande

Objectifs

- Trouver l'identifiant d'un processus (PID pour Process ID) à partir de différentes commandes
- Utiliser kill et killall efficacement et avec le signal approprié
- Gérer les tâches (jobs) à partir de la ligne de commande, en arrière plan et au premier plan

Visualisation des processus en cours

Les processus sont référencés par un identifiant unique, le **PID**. Ce nombre peut être utilisé pour changer la priorité d'un processus ou pour l'arrêter.
Un processus correspond à n'importe quel exécutable exécuté. Si le processus 2 a été lancé par le processus 1, on l'appelle un *processus fils*. Le processus qui l'a lancé est appelé *processus parent*.

L'arborescence des processus

La commande **pstree** donne une bonne illustration de la hiérarchie des processus parents et fils.
Exemple: extrait de la sortie de pstree

```
bash(1046)---xinit(1085)-+-X(1086)
                         `-xfwm(1094)-+-xfce(1100)---xterm(1111)---bash(1113)-+-
pstree(1180)
                                      |                                       |-
soffice.bin(1139)---soffice.bin(1152)-+
                                                                              
-soffice.bin(1153)
                                      |                                       |
|-soffice.bin(1154)
                                      |                                       |
|-soffice.bin(1155)
                                      |                                       |
|-soffice.bin(1156)
                                      |                                       |
`-soffice.bin(1157)
                                      |                                       `-
xclock(1138)
                                      |-xfgnome(1109)
                                      |-xfpager(1108)
                                      |-xfsound(1107)
                                      `-xscreensaver(1098)
```

Dans l'exemple précédent, tous les PIDs sont affichés; ils sont évidemment incrémentés. Les options les plus courantes de **pstree** sont **-p** pour afficher les PIDs et **-h** pour faire ressortir (en gras) les processus utilisateurs.

Recherche des processus en cours d'exécution

Une méthode plus directe pour déterminer les processus en cours d'exécution est d'utiliser **ps**. La plupart des utilisateurs utilisent une combinaison d'options qui marche pour presque toutes les situations.
Voici 3 combinaisons :

- **ps ux** affiche tous les processus lancés par l'utilisateur

- **ps T** processus lancés sous le terminal actif par l'utilisateur
- **ps aux** tous les processus du système

Nous vous recommandons de lire la **page de manuel de ps** et de choisir les options que vous préférez ! on peut utiliser **ps** avec les options de type UNIX et BSD :
Utilisation :

- ps - [arguments dans le style Unix98]
- ps [option dans le style BSD]
- ps --[options étendues dans le style GNU]
- ps -help pour un résumé des options

Résumé des options importantes

-a	affiche tous les processus liés à une console pour l'utilisateur actuel sauf les maîtres de session (NdT : **-a** est différent de **a** comme indiqué ci-dessus)
-e ou -A	affiche tous les processus
-f	donne le PPID (Parent Process ID) et le STIME (Start Time)
-l	affiche le format long
a	affiche tous les processus liés à une console et pour tous les utilisateurs
x	affiche également tous les processus non liés à une console

Mise à jour en continu des informations sur les processus

La commande **top** met à jour les informations sur les processus à une fréquence ajustable.
Pendant son utilisation, vous pouvez afficher une liste de commandes de **top** en tapant **h**. La barre espace met à jour les informations instantanément.

Vous pouvez également utiliser **top** pour changer la priorité d'un processus, comme nous le verrons dans la partie suivante.

Modification des processus

Arrêter les processus

On utilise la commande **kill** pour envoyer des signaux aux processus. Il existe 63 signaux. Le signal par défaut, nommé **SIGTERM**, termine le processus et a pour valeur numérique **15**.

- **kill** Syntaxe :

```
kill SIGNAL PID
```

Chaque processus peut choisir ou non de détecter un signal, à l'exception de **SIGKILL** qui est directement géré par le noyau. De nombreux services refinissent le sens de **SIGHUP** en :"relire le fichier de configuration".
Signaux les plus courants

1 ou SIGHUP	déconnecter le processus

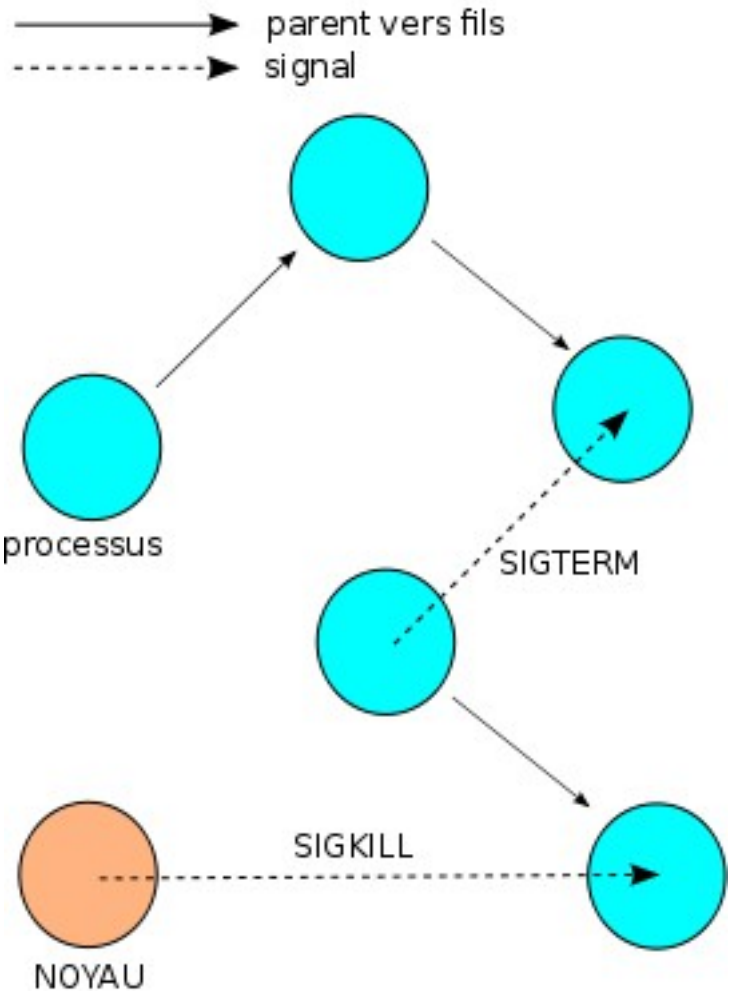
2 ou SIGINT	équivalent à **Ctrl+C** : interruption
3 ou SIGQUIT	quitter le processus
9 ou SIGKILL	tue le processus à travers un appel noyau
15 ou SIGTERM	termine un processus "gentiment". C'est le signal PAR DÉFAUT

On peut également arrêter un processus sans connaître son PID avec la commande **killall**.

- **killall** Syntaxe

```
killall SIGNAL NOM_PROCESSUS
```

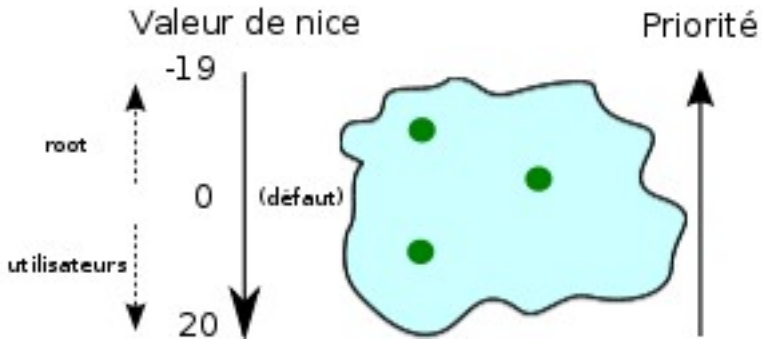
Illustration : signaux entre processus



Priorité des processus et valeurs de nice

Les valeurs de **nice** (NI pour nice indice) modifient la priorité pour le processeur et sont utilisés pour adapter la charge du processeur dans un environnement multi-utilisateur. Chaque processus est lancé avec la **valeur de nice par défaut : 0**. Ces valeurs sont comprises entre 19 (la plus petite) et -20 (la plus importante). (**NdT** : moins un processus est gentil, plus il consomme de puissance).
Seul le super-utilisateur root peut diminuer la valeur de nice (**NdT** : donc augmenter sa priorité) d'un processus. En conséquence, tous les processus étant lancés par défaut avec un nice à 0, seul le root peut définir des valeurs de nice négatives !

Illustration : Valeurs de nice et priorité des processus



On utilise la commande **renice** pour modifier la priorité d'un processus en cours d'exécution, et la commande **nice** pour définir la priorité d'un processus à son lancement.

- **nice** Syntaxe :

  ```
  nice –<NI>  <processus>
  ```
- **renice** Syntaxe :

  ```
  renice  <+/-NI>  -p <PID>
  ```

Notez que **renice** utilise les PID et peut gérer une liste de processus à la fois. L'option **-u** qui affecte tous les processus d'un utilisateur peut être très utile.
Exemples :

- passage des valeurs de nice à 1 pour les processus 234 et 765

  ```
  renice +1  -p 234 765
  ```
- lancer xclock avec une valeur de nice à -5

  ```
  nice  --5  xclock
  ```

Les processus et le shell

processus en arrière plan et au premier plan

Lorsque vous lancez un processus à partir du shell, vous quittez automatiquement l'interpréteur de commandes. Vous remarquerez que les commandes ne répondent pas. La raison à cela, c'est qu'il est possible de lancer des programmes au premier plan (**fg** pour foreground) ou à l'arrière plan (**bg**ou background) du shell.
Lorsqu'un programme est lancé au premier plan, il est possible de récupérer l'invite de commandes en interrompant le programme un moment. Le signal d'interruption est **Ctrl+Z**.

lancement et arrêt des tâches (jobs)

Un processus lancé à partir du shell est également appelé tâche ou job. Lorsque la tâche reçoit le signal **^Z** (**NdT** : SIGTSTP), le processus est stoppé et vous récupérez l'invite de commande. Pour relancer le programme en arrière plan, il vous suffit de taper **bg**.
Exemples :

- lancement d'xclock au premier plan, perte de l'invite de commande :

  ```
  [mike localhost  /bin]$xclock
  ```
- envoi du signal **^Z** à xclock :

  ```
  [1]+  Stopped                 xclock
  ```
- on récupère l'invite de commande, et on tape **bg**

  ```
  [mike localhost  /bin]$bg
  ```
- xclock est exécuté en arrière plan :

  ```
  [1]+ xclock &
  ```

```
[mike localhost  /bin]$
```

Notez la symbole **[1]+** ci-dessus. Le nombre correspond au numéro de job, auquel on peut se référer. Le signe **+** indique le dernier processus modifié. Un **-** indiquerait l'avant dernier processus modifié.
On ajoute **&** à une commande pour la lancer en arrière plan :

- ```
  xclock&
  [1] 6213
  ```

Lister les tâches (jobs)

La commande **jobs** donne la liste de tous les processus lancés à partir du shell actuel et affiche le numéro de job, son statut (lancé ou à l'arrêt), ainsi que les deux derniers processus modifiés :

- ```
  [1]-  Stopped                 xclock
  [2]   Running                 xman &
  [3]+  Stopped                 xload
  ```

Le numéro de job

On peut utiliser le numéro de job pour stopper ou lancer une sélection de tâches, grâce à la commande **fg**.
Exemple : on place le job 2 au premier plan et on tue le job 1

- ```
  fg  2          ou
  fg  %2     ou
  fg  %?xma
  kill –9 %1
  ```

Empêcher le HUP avec nohup

Enfin, il existe un programme appelé **nohup** qui se comporte comme un processus père indépendamment de la session utilisateur. Lorsqu'un utilisateur ferme sa session, le système envoie le signal **HUP** à tous les processus liés à la session.
Par exemple, si vous souhaitez empêcher que le système envoie le signal **HUP** à votre script bigbang qui cherche à calculer l'âge de l'univers, vous devriez le lancer comme ceci :

- ```
  nohup bigbang &
  ```

Résumé et exercices

Questions de révision

Oui ou Non

1. la commande kill lancée sur un processus tentera toujours de tuer le processus donné :
2. Les commandes "kill $(pidof xeyes)" et "killall xeyes" sont équivalentes : _Un programme lancé avec le caractère "&" sera lancé en tâche de fond : _La valeur de nice d'un processus est la même que sa priorité de processeur : _Voir les réponses

Glossaire

Terme	Description
processus en arrière plan `$commande &` Contrairement aux processus au premier plan, vous pouvez lancer d'autres commandes dans le shell sans devoir attendre que ce processus soit terminé	Processus lancé à partir du shell avec :

Gestion des processus

processus au premier plan

Processus lancé à partir du shell avec :

```
$commande
```

Une fois le processus lancé, le shell doit attendre qu'il se termine avant de pouvoir lancer une autre commande

processus orphelin

Processus dont le père s'est terminé. Un processus orphelin est "adopté" par init

PID

Nombre associé au processus

processus zombie

processus terminé mais que le parent considère comme encore présent, en attendant le prochain appel système wait(). L'appel système wait() effectué par le processus père peut modifier l'état du processus fils en terminé, si le processus s'est terminé. En général, un processus zombie ne dure pas longtemps. Cependant, des bogues ou anomalies peuvent faire durer des processus zombies, qui monopolisent des ressources système même si le processus est physiquement terminé !

Commandes

Commande	Description (apropos)
bg	reprendre en arrière plan un job suspendu
Ctrl+Z	combinaison de touches utilisée pour suspendre le processus actif au premier plan
fg	placer un job au premier plan (il devient ainsi le processus actif)
jobs	liste des processus lancés à partir du terminal actif
kill	envoyer un signal spécifié à un processus en utilisant son PID
killall	envoyer un signal spécifié à un processus en utilisant son nom
nice	lance un processus en modifiant sa priorité
nohup	nohup (1) - Exécuter une commande en la rendant insensible aux déconnexions, avec une sortie hors terminal
ps	ps(1) - affiche un instantané des processus actifs. Si vous souhaitez que ce soit mis à jour régulièrement, utilisez top
pstree	affiche une arborescence des processus actifs en partant (par défaut) d'init
renice	modifie la priorité d'exécution d'un processus
top	top (1) - Afficher les tâches

Travaux pratiques

1. Vérifiez la valeur de nice de votre x-terminal. Changez cette valeur avec **top** ou **renice**
2. Quel signal est équivalent à ^Z ? (Vous obtiendrez la liste des lignaux avec **kill -l**)
3. Quel signal, redéfini pour la plupart des services, les force à relire leur fichier de configuration ?
4. Quel est le signal envoyé par défaut à un processus en utilisant **kill** ou **killall** ?
5. Quel signal est directement géré par le noyau et ne oeut pas être redéfini ?
6. Avant de continuer, connectez vous à un terminal virtuel (tty1 à tty6). Nous souhaitons lancer un script qui coninuera à être exécuté quand nous nous déconnecterons, en utilisant le processus père **nohup**.
   - dans le répertoire */tmp*, créez un fichier nommé impression contenant :

     ```
     compte=0
     while (true) do
         echo iteration numero $compte
         let compte+=1
     done
     ```
   - Nous commençons par taper les lignes suivantes, sans utiliser **nohup** :

     ```
     cd /tmp
     ./impression &
     exit
     ```
   - Il est possible que vous ne voyiez pas la ligne de commande en tapant *exit* mais ça devrait vous déconnecter. Reconnectez-vous et vérifier que *impression* n'est plus exécuté :

     ```
     ps ux |  grep  impression
     ```
   - Ensuite, lancez la commande avec :

     ```
     nohup /tmp/impression &
     exit
     ```
   - Reconnectez-vous puis testez ces commandes :

     ```
     ps ux |grep impression
     tail -f ~/nohup.out
     Ctrl+C
     killall impression
     ps ux|grep impression
     tail -f ~/nohup.out
     ```

Réponses aux questions

1. **Non** : kill envoie un signak aux processus. Certains signaux mettent simplement le processus en pause, d'autres forcent les services à relire leur fichier de configuration. Cependant, la signal par défaut (15 ou SIGTERM) tente de terminer le processus.
2. **Oui**
3. **Oui**
4. **Non**

Traitement du texte

État :
Pré-requis

- La ligne de commande

Objectifs

- Manipuler efficacement les fichiers et les flux de données pour le contenu comme souhaité (c'est à dire en triant ou en formatant le contenu)
- Améliorer vos compétences dans l'utilisation de la ligne de commande en mémorisant et en comprenant de simples outils de manipulation des textes

le couteau suisse cat

l'éditeur cat

La commande **cat** peut être utilisée comme un éditeur de texte rudimentaire.
Exemple :

```
cat  > petit-message
nous sommes curieux
de rencontrer
des pingouins à Prague
Crtl+D
```

Vous noterez l'utilisation de **Ctrl+D**. Cette commande est utilisée pour clore la saisie.

le lecteur cat

On utilise plus couramment **cat** pour envoyer du texte vers *stdout*.
Les options les plus courantes sont :

- -n numéroter chaque ligne de la sortie
- -b numéroter uniquement les lignes non vides
- -A afficher le retour charriot

Exemple :

```
cat  /etc/resolv.conf
search    mydomain.org
nameserver   127.0.0.1
```

tac lit à l'envers

tac fait la même chose que **cat** à l'exception qu'elle lit de la dernière ligne à la première.
Exemple :

```
tac petit-message
des pingouins à Prague
de rencontrer
nous sommes curieux
```

Petits outils

Utilisation de head ou tail

On utilise souvent les commandes **head** et **tail** pour analyser les fichiers journaux. Par défaut, ces commandes

affichent 10 lignes. En voici les utilisations les plus courantes :
Exemples :

- afficher les 20 première lignes de */var/log/messages* :

```
head -n 20 /var/log/messages
head -20  /var/log/messages
```

- afficher les 20 dernières lignes de /etc/aliases

```
tail -20  /etc/aliases
```

tail a une option supplémentaire qui nous permet d'afficher la fin d'un texte en commençant par une ligne donnée.

- afficher le texte en partant de la ligne 25 de /var/log/messages

```
tail +25 /etc/log/messages
```

Exercice :

- Pour un texte de 90 lignes, comment utiliseriez-vous **tail** et **head** pour afficher les lignes 50 à 65 ? Y-a-t'il une seule façon de s'y prendre ?

Enfin, **tail** peut afficher un fichier en continu avec l'option **-f**. C'est extrêmement pratique pour suivre les modifications d'un fichier en temps réel.

Compter les lignes, les mots et les octets

La commande **wc** compte le nombre d'octets, de mots et de lignes dans les fichiers. Les options suivantes vous permettent de sélectionner ce qui vous intéresse :
Options de wc

-l	compte le nombre de lignes
-w	compte le nombre de mots (words)
-c	compte le nombre d'octets
-m	compte le nombre de caractères

NdT : les options **-c** et **-m** me rendent perplexe, mais il n'y a rien à y faire, juste à les connaître...
Remarque : sans argument, **wc** compte ce qui est saisi dans *stdin*.

Numérotation des lignes

La commande **nl** affiche la même chose que **cat -b**.
Exemples :

- numéroter toutes les lignes y compris les lignes vides :

```
nl -ba /etc/lilo.conf
```

- numéroter uniquement les lignes avec du texte :

```
nl -bt  /etc/lilo.conf
```

Remplacer les tabulations par des espaces

On utilise la commande **expand** pour remplacer les tabulations par des espaces. **unexpand** est utilisé pour l'opération inverse.

Afficher les fichiers binaires

Il y a nombre d'outils pour ça. Les plus courants sont **od** (octal dump) et **hexdump**.

Découper les fichiers

La commande **split** peut découper un fichier en plusieurs fichiers plus petits à partir de critères comme la taille ou le nombre de lignes. Par exmple, nous pouvons découper */etc/passwd* en fichiers de 5 lignes chacuns :

```
split -l 5 /etc/passwd
```

Cette commande va créer des fichiers appelés xaa, xab, xac, xad, etc., chaque fichier contenant au plus 5 lignes. Il est possible de donner un préfixe plus significatif que "x", comme "pass-5" :

```
split -l 5 /etc/passwd passwd-5
```

Cette commande crée des fichiers identiques à la commande précédente, mais ils sont désormais nommés passwd-5aa, passwd-5ab, passwd-5ac, passwd-5ad, ...

Éliminer les lignes successives en doublon

La commande **uniq** n'envoie à *STDOUT* qu'une version des lignes successives identiques.

Exemple :

```
uniq > /tmp/UNIQUE
ligne 1
ligne 2
ligne 2
ligne 3
ligne 3
ligne 3
ligne 1
^D
```

Le fichier */tmp/UNIQUE* a le contenu suivant :

```
cat /tmp/UNIQUE
ligne 1
ligne 2
ligne 3
ligne 1
```

 **Remarque** : Nous pouvons voir dans l'exemple précédent qu'en utilisant **"uniq"**, les lignes en doublon qui ne se suivent pas sont envoyées à *STDOUT*.

Quel est le contenu de */tmp/UNIQUE* si nous envoyons d'abord *STDIN* à **sort** (voir "'sort"' un peu plus loin) comme ici :

```
sort | uniq > /tmp/UNIQUE
```

Manipulations du texte

Les outils suivants modifient la mise en page du texte.

Sélectionner les champs et les caractères avec cut

La commande **cut** peut extraire une plage de caractères ou de champs de chaque ligne d'un texte.

L'option **-c** est utilisée pour manipuler les caractères.

Syntaxe :

```
cut –c {plage1,plage2}
```

Exemple :

```
cut -c5-10,15- /etc/passwd
```

La commande précédente extrait les caractères 5 à 10 puis 15 jusqu'à la fin pour chaque ligne de **/etc/passwd**.

On peut spécifier le séparateur de champ (espace, virgule, etc.) d'un fichier ainsi que les champs à extraire. Ces options sont définies respectivement par les options **-d** (delimiter) et **-f** (field).

Syntaxe :

```
cut -d {séparateur}  -f {champs}
```

Exemple :

```
cut -d: -f 1,7 --output-delimiter=" " /etc/passwd
```

Cette commande extrait les 1er et 7e champs de /etc/passwd séparés par un espace. Le délimiteur de sortie est le même que le délimiteur d'entrée d'origine (**NdT** : par défaut, c'est la tabulation). L'option **--output-delimiter** vous permet de le changer.

Jointure de texte

La commande la plus facile est **paste** qui concatène deux fichiers l'un à la suite de l'autre.
Syntaxe :

```
paste   texte1   texte2
```

Avec **join** vous pouvez en plus préciser quels champs vous souhaitez.
Syntaxe :

```
join  -j1 {champ_no}  -j2{champ_no}  texte1  texte2              ou
join   -1 {champ_no}  -2{champ_no}  texte1  texte2
```

Le texte n'est envoyé à la sortie que si les champs sélectionnés correspondent.

 Les comparaisons se font ligne par ligne et le processus s'arrête dès qu'il n'y a pas de correspondance, même s'il y a d'autres correspondances à la fin du fichier.

Trier la sortie

Par défaut, **sort** trie le texte par ordre alphabétique. Pour effectuer un tri numérique, utilisez l'option **-n**.

Mise en forme de la sortie avec fmt et pr

Vous pouvez modifier le nombre de caractères par ligne avec **fmt**. Par défaut **fmt** joins les ligne et génère des lignes de 75 caractères.
Options de fmt

-w	(width) nombre de caractères par ligne
-s	découpe les lignes longues mais sans les remplir
-u	sépare chaque mot par une espace et chaque phrase par deux espaces

On peut paginer les longs fichiers pour qu'ils correspondent à une taille donnée avec la commande **pr**. On peut contrôler la longueur des pages (66 lignes par défaut), la largeur (par défaut 72 caractères) ainsi que le nombre de colonnes.
Lorsqu'on produit un texte sur plusieurs colonnes, chaque colonne est tronquée uniformément en fonction de la largeur de page spécifiée. Cela veut dire que des caractères sont supprimés à moins d'avoir édité le texte de façon à éviter cela.

Convertir les caractères

La commande **tr** convertit un ensemble de caractères en un autre.
Exemple :

- convertir les majuscules en minuscules

  ```
  tr 'A-B' 'a-b'  < fichier.txt
  ```

- changer de délimiteur dans /etc/passwd

  ```
  tr  ':'  ' ' < /etc/passwd
  ```

 **Remarque** : **tr** a seulement deux arguments ! Le fichier n'est pas un argument.

Résumé et exercices

Questions de révision

Oui ou Non

1. Les commandes **"cat FICHIER"** et **"cat < FICHIER"** affichent toutes les deux le contenu de FICHIER : _La commande **"last FICHIER"** affiche les 10 dernières lignes de FICHIER : _Lorsque vous modifiez les lignes d'un fichier en utilisant **cut**, ces modifications ne sont effctives que pour *STDOUT* : _Lorsqu'on utilise **uniq** sur un fichier, les lignes en doublon successives sont éliminées dans le fichier : _Voir les réponses

Commandes

Commande	Description (apropos)
cat	cat (1) - Concaténer des fichiers et les afficher sur la sortie standard
cut	cut (1) - Supprimer une partie de chaque ligne d'un fichier
expand	expand (1) - Convertir les tabulations en espaces
fmt	fmt (1) - Formater simplement du texte
head	head (1) - Afficher le début des fichiers
join	join(1) – Fusionner les lignes de deux fichiers ayant des champs communs
nl	nl(1) – Numéroter les lignes d'un fichier
od	od(1) – Afficher le contenu d'un fichier en octal ou sous d'autres formats
paste	paste(1) – Regrouper les lignes de différents fichiers
sort	sort(1) – Trier les lignes de fichiers texte
split	split(1) – Découper un fichier en différentes parties
tac	tac(1) – Concaténer et afficher des fichiers à l'envers
tail	tail(1) – Afficher la dernière partie de fichiers
tr	tr(1) – Convertir ou éliminer des caractères
unexpand	unexpand(1) – Convertir les espaces en tabulations
uniq	uniq(1) – Signaler ou éliminer les lignes répétées
wc	wc(1) – Afficher le nombre de lignes, de mots et d'octets d'un fichier

Travaux pratiques

1. À l'aide de **cat**, saisissez le texte suivant dans le fichier *message* :

```
cat >> message
ligne 1
```

```
^D
```

- Faites la même chose mais utilisez le mot clé STOP à la place du contrôle de fin de fichier (^D)

```
cat >> message << STOP
ligne 2
STOP
```

- Ensuite, ajoutez le texte suivant avec la commande **echo** :

```
echo ligne 3 >> message
```

2. Créez le fichier *index* avec deux champs REFERENCE et TITRE séparés par un espace :

```
001 Utiliser_Linux
```

- Créez un second fichier *tarifs* avec deux champs REFERENCE et PRIX séparés par un espace :

```
001 9.99
```

- Utilisez **join** pour afficher les champs REFERENCE, TITRE et PRIX

3. À l'aide de **tr**, remplacez toutes les virgules par des points-virgules dans /etc/passwd
   - Faites la même chose avec **cut**
4. Affichez les lignes 70 à 85 de */var/log/messages* en utilisant **head** et **tail**.
5. Utilisez les commandes **cut**, **grep** et **ifconfig** pour n'afficher que l'adresse IP de la première interface réseau *eth0*.
6. Créez le répertoire */tmp/fichiers*

```
mkdir /tmp/fichiers
```

- Créez 50 fichiers dans ce répertoire :

```
count=0
while [ $compte -lt 50 ]; do
touch /tmp/fichiers/$compte.txt
let compte+=1
done
```

- Nous souhaitons changez les extensions txt en dat. Pour ce faire, nous devons taper les commandes suivantes :

```
for FICHIER in $(ls *.txt)
do
NOM_FICHIER=$(echo $FICHIER| cut -d. -f1)
mv  $FICHIER $NOM_FICHIER.dat
done
```

Réponses aux questions

1. **oui**
2. **non** : il faut utiliser **tail**
3. **oui** : c'est le cas avec toutes les commandes de traitement de texte
4. **non** : toutes les commandes de traitement de texte modifient les données vers *STDOUT* et ne modifient pas le fichier original

Installation des logiciels

Pré-requis

- La ligne de commande

Objectifs

- Comprendre l'utilisation du fichier Makefile lorsqu'on compile de gros projets à partir des sources
- Manipuler efficacement les sources des programmes et lancer les commandes appropriées leur compilation
- Résolution des problèmes relatifs aux bibliothèques partagées (ou dynamiques)
- Utiliser les gestionnaires de paquets pour rechercher, ajouter, supprimer ou vérifier des logiciels (**NdT** : RPM et Debian)

Introduction

Commençons par un petit exemple de code. Même si nous n'avons pas besoin d'une compréhension avancée du langage C, ces exemples pourront vous aider à résoudre des situations courantes.

- Le fichier main.c :

```
#include<stdlib.h>

int main(){
   Bonjour();
}
```

- Le fichier Bonjour.c :

```
#include<stdio.h>

void Bonjour(){
        printf("Salut ! \n");
}
```

Vous noterez que *main.c* est incomplet puisque la fonction *Bonjour()* n'est pas définie. De même, Bonjour.c ne contient pas de déclaration "main". Ces fichiers sont interdépendants. Cependant, on peut compiler des fichiers objets (.o) qui sont comme des fichiers binaires non exécutables et qu'on peut utiliser pour construire une application.

- Compilation des fichiers objets :

```
gcc –c main.c
gcc –c Bonjour.c
```

Ceci générera deux fichiers main.o et Bonjour.o qu'on peut désormais utiliser pour construire l'application app.

- Compilation de *app* :

```
gcc –o app main.o Bonjour.o
```

L'option **-o** spécifie simplement un nom pour le code compilé. Si on ne précise pas de nom, le fichier s'appelle par défaut *a.out*.

On peut automatiser toutes ces étapes en utilisant un fichier **Makefile**. Voici un Makefile minimal pour compiler notre programme *app*.

- Makefile

```
SHELL = /bin/sh
CC = /usr/bin/gcc
```

```
app: main.o Bonjour.o
        $(CC) –o app main.o Bonjour.o
main.o: main.c
        $(CC) –c main.c
Bonjour.o: Bonjour.c
        $(CC) –c Bonjour.c
```

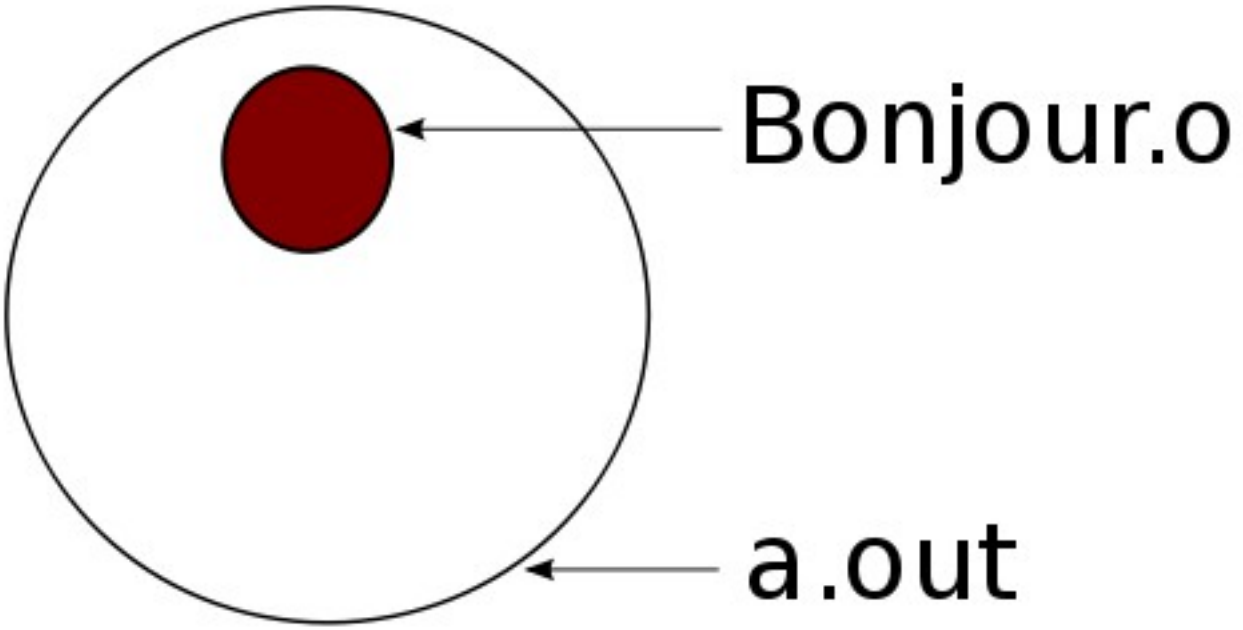
Bibliothèques statiques et partagées

Lorsque des fonctions sont souvent utilisées, elles sont stockées dans des bibliothèques. On peut alors lier du code qui utilise ces fonctions aux bibliothèques qui les contiennent pendant la compilation. Les bibliothèques peuvent être liées *statiquement* ou *dynamiquement* au code.
La compilateur **gcc** a de nombreuses options pour lier les les bibliothèques. Cependant, par défaut **gcc** lie les fichiers donnés à la ligne de commande qui ne comportent pas l'extension **.c** (seuls les fichiers **.c** sont traités comme du code).

Illustration : lien par défaut

```
gcc main.c Bonjour.o
```

Cette commande produit un fichier *a.out* exécutable lié statiquement à l'objet Bonjour.o.
Illustration d'une application liée statiquement (a.out) :



Bibliothèques statiques

Les bibliothèques statiques sont des archives de fichiers **.o**. Ces archives sont créées avec la commande **ar** et comportent une extension **.a**.
Illustration : ajout d'un fichier objet à une archive

```
ar rcs libfoo.a fichier1.o fichier2.o
```

Bibliothèques dynamiques / partagées

Une bibliothèque partagée est une bibliothèque qui est chargée par le programme quand celui-ci est exécuté. On dit également que la bibliothèque est chargée dynamiquement.
Illustration : Création d'une bibliothèque partagée :

```
gcc –c –fPIC Bonjour.c.c     crée le fichier objet
gcc –shared –W1,soname,libfoo.so.1 –o libfoo.so.1.0 Bonjour.o
```

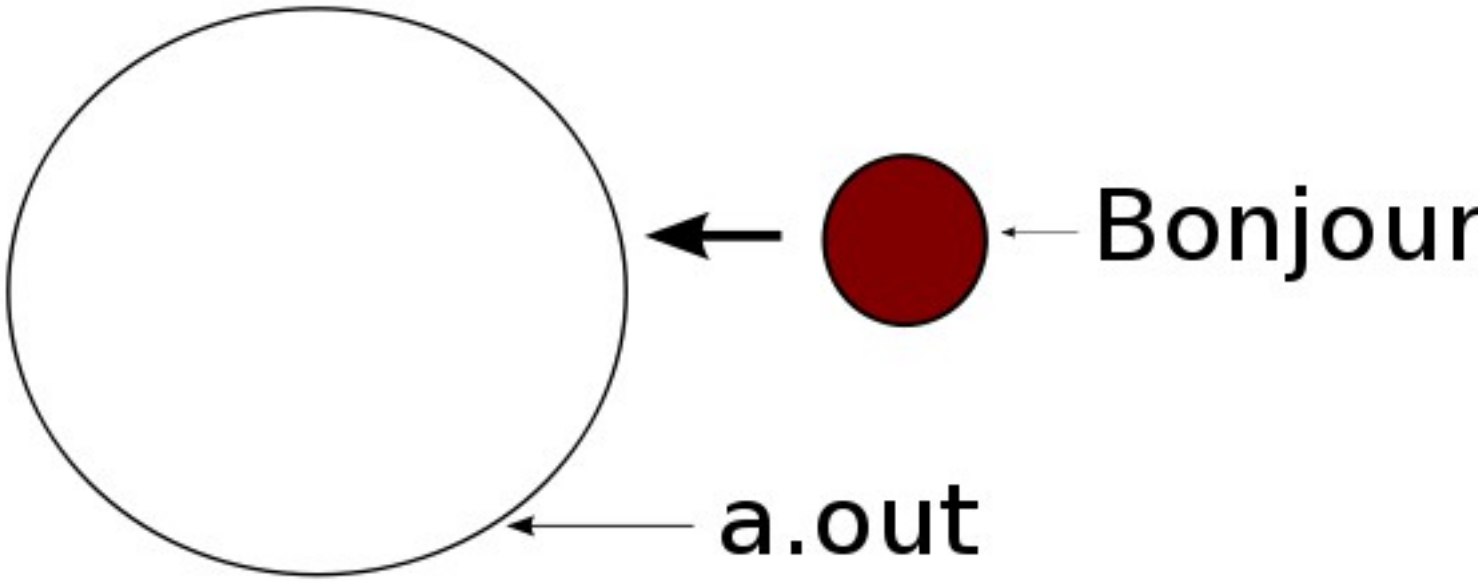
L'option **-fPIC** génère du code indépendant de la position (PIC : Position Independent Code).
Illustration : compilation avec une bibliothèque partagée

```
gcc main.c libfoo.so.1.0
```

Cette commande produit un fichier exécutable **a.out**. Cependant, si vous essayez de le lancer, il vous affichera l'erreur suivante :
Illustration : Erreur due à une bibliothèque partagée non trouvée

```
./a.out: error while loading shared libraries: libfoo.so.1.0: cannot open shared object file: No such file or directory
```

Cette erreur illustre ce qui se passe quand l'éditeur de liens n'arrive pas à trouver la bibliothèque dynamique *libfoo.so.1.0*. Nous verrons comme résoudre ce problème dans la prochaine partie.
Illustration d'une application liée dynamiquement (a.out) :



Le processus chargé d'attacher une bibliothèque dynamique au lancement, appelé la *liaison*, est géré par l'éditeur de lien (ou **linker**) **ld.so**. Comment l'éditeur de liens sait où trouver *libfoo.so* ? La réponse à cette question se trouve juste en-dessous.

Attribution des noms aux bibliothèques partagées et chargement dynamique

Pour comprendre comment les bibliothèques sont gérées sous Linux, nous allons nous baser sur l'exemple ci-dessous.

- Tout d'abord, la bibliothèque est générée avec la commande suivante :

  ```
  gcc -shared -soname,libfoo.1 -o libfoo.1.0
  ```

- Voici à quoi correspondent les noms des fichiers
- Enfin, on se rappellera que la liste des es bibliothèques est maintenue par la commande **ldconfig** qui lit le fichier de configuration **/etc/ld.so.conf** pour générer le cache **/etc/ld.so.cache**. **ld.so** est le chargeur et éditeur de liens dynamique sous Linux.

Pour connaître les bibliothèques partagées nécessaires à l'exécution d'un programme, on utilise la commande **ldd**.
Exemple :

```
ldd a.out
        libfoo.so.1.0 => not found
```

```
        libc.so.6 => /lib/libc.so.6 (0x40028000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Vous noterez que *libfoo.so.1.0* est introuvable. En effet, *a.out* doit charger dynamiquement cette bibliothèque mais l'éditeur de liens **ld.so** ne la connaît pas.
En fait l'éditeur de liens utilise une base de donnée appelée *ldcache* dont le contenu est de la forme :

```
soname =>/chemin/vers/bibliothèqe
```

La commande **ldconfig -p** permet d'afficher le contenu du *ldcache* :

```
ldconfig -p
        libaudiofile.so.0 (libc6) => /usr/lib/libaudiofile.so.0
        libaudiofile.so (libc6) => /usr/lib/libaudiofile.so
        libaudio.so.2 (libc6) => /usr/X11R6/lib/libaudio.so.2
        libattr.so (libc6) => /usr/lib/libattr.so       ..........
```

Le *ldcache* est généré à l'amorçage du système par la commande **ldconfig**. Par défaut, **ldconfig** examine les répertoires */lib* et */usr/lib* pour construire le *ldcache*.
Si d'autres répertoires contiennent des bibliothèques, comme /usr/local/lib, /opt/lib ou /usr/X11R6/lib, vous devez répertorier ces répertoires dans le fichier **/etc/ld.so.conf** pour informer **ldconfig** qu'il doit les prendre en considération lorsqu'il génère le cache.

Que se passe-t-il au lancement d'une application ?

L'application demande les bibliothèques dont il a besoin à l'éditeur de lien en utilisant le *soname*. Ensuite, l'éditeur de lien interroge le *ldcache* et associe le *soname* avec le chemin complet vers la bibliothèque, puis il établit le lien entre la bibliothèque et l'application.

Que se passe-t-il si le *ld-cache* ne contient pas le chemin complet vers la bibliothèque ?

En général, le programme n'arrive pas à se lancer et affiche un message d'erreur du type "cannot open shared object file: No such file or directory". Cependant, on peut aussi définir la variable globale **LD_LIBRARY_PATH** et lui assigner le nom du répertoire contenant la bibliothèque.

À partir de ça, nous savons comment résoudre le problème que nous avions rencontré avec notre programme en utilisant l'une des deux méthodes suivantes :

1. Si vous devez tester temporairement le programme, définissez la variable **LD_LIBRARY_PATH** :
   ```
   export LD_LIBRARY_PATH=$(pwd)
   ```
2. Si vous êtes administrateur et que vous souhaitez que la bibliothèque soit accessible à tout le monde, copiez le fichier *libfoo.so.1.0* puis lancez **ldconfig** pour mettre à jour le ldcache.

Notes :

- Le spécifications GNU conseillent de placer les bibliothèques dans */usr/local/lib/*. Ces recommandations sont généralement suivies par les développeurs et la plupart des sources installent les bibliothèques dans ce répertoire et les binaires dans /usr/local/bin/. On utilise les commandes **"make install"** et **"make uninstall** pour installer et supprimer ce code.
- La **FHS** (norme de la hiérarchie des systèmes de fichiers que nous avons introduit dans les système de fichiers) recommande de placer les bibliothèques dans */usr/lib/* et les binaires associés dans */usr/bin/*. Cette convention est suivie par les distributions Linux. Dans les faits, le code mûr et stable est placé dans */usr/* plutôt que dans */usr/local/* et les deux standards ne sont pas en contradiction. L'installation et la suppression de ce code se fait en utilisant les commandes de gestion des paquets **rpm** ou **deb**.

Remarque : Sur certaines distributions, **ldconfig** n'examine pas */usr/local/lib*. il suffit d'ajouter le répertoire dans **/etc/ld.so.conf** et... redémarrer ?

Installation à partir des sources

Les projets open source sont souvent distribués sous forme d'archives tar. De nombreux environnements de

développement (glade, kdevelopp, etc.) génèrent les fichiers qui facilitent la compilation et l'installation d'un projet.

Les archives non compressées

Les archives non compressées portent l'extension **.tar**. Par exemple, si le projet a été développé dans un répertoire appelé *mon-projet-v.1/*, alors la commande suivante archivera le répertoire avec ses fichiers et sous-répertoires :

```
tar c mon-projet-v.1/ > mon-projet-v.1.tar
```

ou

```
tar cf  mon-projet-v.1.tar mon-projet-v.1/
```

Comme la plupart des projets sont plutôt gros et téléchargeables sur Internet, on les trouve plus souvent sous une forme compressée.

Compression

Les trois outils utilisés pour la compression sont **compress** (ancien), **gzip** et **bzip2**. Contrairement au zip de windows, ces outils ne permettent de compresser que des fichiers. Cependant, comme une archive est un fichier qui contient toutes les informations pour pouvoir récupérer des répertoires, on peut utiliser ces commandes avec les archives. On appelle souvent les archives compressées "tarball".

Outil de compression	Outil de décompression'	Décompression cat	Extension de fichier
compress	uncompress	zcat	.Z
gzip	gunzip	zcat	.gz
bzip2	bunzip2	bzcat	.bz2

Exemples :

```
compress -v FICHIER1
FICHIER1:  -- replaced with FICHIER1.Z Compression: 40.29%
gzip  -v FICHIER2
FICHIER2:   53.4% -- replaced with FICHIER2.gz
bzip2 -v FICHIER3
FICHIER3:    2.326:1,  3.439 bits/byte, 57.01% saved, 605504 in, 260320 out.
```

Remarques :

1. Lorsque vous compressez un fichier, l'extension .Z, .gz ou .bz2 est ajoutée au nom du fichier
2. Ces commandes de compression ne fonctionnent qu'avec des fichiers, pas avec des répertoires
3. On ne peut compresser qu'un fichier à la fois (pas de caractère générique !)

On peut également utiliser les commandes **zcat** et **bzcat**pour décompresser des fichiers, cependant les fichiers décompressés sont envoyés à *STDOUT* donc il est nécessaire de faire une redirection :

```
zcat FICHIER.Z > FICHIER1
```

Archives et compression

Outil de compression	Option de tar	Extension de l'archive
compress	Z	.tar.Z ou .tgZ
gzip	z	.tar.gz ou.tgz
bzip2	j	.tar.bz2

Le tableau précédent présente les options de tar **Z**,**z** et **j** qui font appel aux outils de compression appropriés lorsque c'est nécessaire.
Les deux exemples suivants sont équivalents :

```
tar cf  mon-projet-v.1.tar mon-projet-v.1/
bzip2  mon-projet-v.1.tar
tar cjf  mon-projet-v.1.tar.bz2 mon-projet-v.1/
```

Utilisation de tar

Nous savons désormais comment créer des archives. Il nous reste à connaître les options les plus courantes de **tar**.

Opération	Création	Extraction	Test
Options minimales	c ou cf	xf	tf
Autres Options	v,Z,z,j	v,Z,z,j	v,Z,z,j

Exemples : extractions

```
tar xvjf monprojet-v.1.tar.bz2
tar xzf autre-projet-v.2.0.tar.gz
```

Exemples : tests

```
tar tjf monprojet-v.1.tar.bz2
tar tzf autre-projet-v.2.0.tar.gz
```

Exemples alternatives utilisant **zcat** ou **bzcat**

```
bzcat monprojet-v.1.tar.bz2 | tar xf -
zcat autre-projet-v.2.0.tar.gz | tar tf -
```

Fichiers courants dans les sources des projets

Une fois l'archive d'un projet extraite, vous devriez trouver les fichiers suivants :

- **configure** : c'est un script qui détermine l'architecture système utilisée. Il vérifie également que tout ce qui est nécessaire à la compilation du projet est présent : compilateur, bibliothèques et en-têtes. Ces informations sont ensuite écrites dans des fichiers appelés **Makefile**. La méthode la plus sûre pour lancer ce script est de taper :

  ```
  ./configure'
  ```

  Vous pouvez également décider où vous installerez le projet avec l'option **--prefix**. Par défaut, la plupart des projets s'installent dans */usr/local*. Si vous souhaitez installer le projet compilé dans votre répertoire personnel, vous devriez taper :

  ```
  ./configure –prefix=$HOME
  ```

- **Makefile** : ce fichier agit comme un fichier de configuration pour la commande **make**. Les principales informations qu'il contient sont :
  - le nom du compilateur et les options de compilation
  - le chemin pour les bibliothèques partagées et les fichiers d'en-tête
  - les liens entre les fichiers source (.c) et les fichiers objets (.o)

Compilation du projet

Si les fichiers que nous venons de voir sont présents, alors vous avez une bonne chance de pouvoir installer le programme sur votre ordinateur en suivant les étapes suivantes :

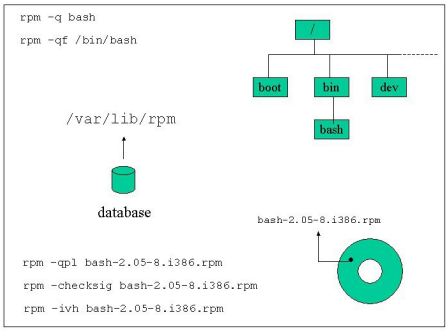
```
./configure
make
make install
```

Il est très fortement recommandé de lancer **./configure** et **make** en utilisateur normal. **make install** ne peut être lancé qu'en tant que root pour les répertoires d'installation protégés en écriture (*/usr* et */usr/local*).
Le script **./configure** a de nombreuses options. Pour une installation personnalisée, consultez ces options avec :

```
./configure --help
```

RPM : RedHat Package Manager

La plupart des distributions Linux utilisent un système de gestion de paquets pour installer, mettre à jour ou rechercher des programmes. Les systèmes de gestion de paquets Debian et RPM sont les plus populaires. Nous verrons dpkg et les outils Debian dans la prochaine partie.
Illustration : Fonctions d'un gestionnaire de paquets



Conventions de noms pour les paquets

Il n'y a pas de convention stricte, mais la plupart des paquets rpm sont sous la forme :
`nom-version_programme-version_paquet.architecture.rpm`
L'architecture indique au choix le type d'architecture système du binaire (i386, ppc, ia64, noarch,...) ou si le paquet contient les sources du programme (src).

Modes majeurs ou mineurs

Suivant leur place dans la ligne de commande, les options changent de signification. **rpm** fait la distinction entre la première option et les suivantes.
La première option donnée à **rpm** est son **mode majeur**. Par exemple dans la commande `rpm -iv A.rpm`, l'option **"i"** est l'option majeure qui entrainera l'installation du paquet A.

De même, une option qui n'est pas en première position est en mode mineur. par exemple dans `rpm -qpi`, l'option **"i"** est en mode mineur et récupèrera des informations sur le paquet A comme son auteur et son type de licence.

Voici les **modes majeurs** pour rpm :

Option courte	Option longue	Description
-i	--install	Installe le paquet
-U	--update	Met à jour (update) ou installe le paquet
-F	--freshen	Met à jour uniquement les paquets installés
-V	--verify	Vérifie la taille, les signatures MD5, les permissions, les types, etc.
-q	--query	Effectue des requêtes sur les paquets installés et désinstallés, ou sur des fichiers
-e	--erase	Désinstalle le paquet

Et voici les **modes mineurs** pour rpm :

Option courte	Description
a	s'applique à tous les paquets installés
c	avec **q** : liste les fichiers de **c**onfiguration
d	avec **q** : liste les fichiers de **d**ocumentation
f	avec **q** : recherche le paquet qui a installé le fichier donné
h	Affiche 50 marques de hachage quand l'archive du paquetage est déballée
i	avec **q** : affiche les **i**nformations sur un paquet
l	avec **q** : liste les fichiers et répertoires d'un paquet
p	avec **q** : interroge un paquetage <fichier_paquetage> non installé
v	bavard

Modes de recherche

Il y a trois types de recherches : les paquets non installés, les paquets installés et les fichiers.

Type de recherche	Option
paquet (fichier rpm)	-qp
paquet installé	-q
fichier (sur le système)	-qf

D'autres options vous permettent d'obtenir des information sur tous les fichiers installés **-l**, la documentation **-d**, les fichiers de configuration **-c**, etc.
Considérons par exemple le paquet routed-0.17.i386.rpm. Nous pouvons faire une requête sur le paquet et lister son contenu avant l'installation avec l'option **-l** :

```
rpm –qpl routed-0.17.i386.rpm
```

Une fois le paquet installé, on peut interroger le paquet avec :

```
rpm –ql routed-0.17     ou
rpm –ql routed
```

Enfin, si nous voulons trouver le nom du paquet qui a installé le fichier **/usr/sbin/routed**, on interrogera la base de données de **rpm** avec :

```
rpm –qf /usr/sbin/routed
```

Options spéciales

--nodeps	Ne pas vérifier les dépendances avant d'installer, mettre à jour ou de désinstaller les paquetages
--force	force une mise à jour
--test	n'installe pas ou ne met pas à jour, affiche juste les infromations sur stdout

--requires PAQUET	avec **q** liste les paquets dont dépend ce paquet
--whatrequires CAPACITÉ	avec **q** liste les paquets qui dépendent de ce paquet

Signatures des paquets

Vous pouvez vérifier la signature de chaque paquet distribué comme partie d'un projet. Par exemple, pour récupérer les clés de tous les développeurs impliqués dans le projet Fedora, il vous suffit de faire (seulement une fois) :

```
rpm –-import /usr/share/rhn/RPM-GPG-KEY-fedora
```

Vous pouvez désormais télécharger n'importe quel paquet à partir d'un mirroir FTP des RPMs du projet. Par exemple nous avons téléchargé **zlib-1.2.1.1-2.1.i386.rpm** à partir du sous-répertoire Fedora de ftp.mirror.ac.uk. Nous pouvons vérifier l'authenticité du fichier avec :

```
rpm --checksig /home/adrian/zlib-1.2.1.1-2.1.i386.rpm
/home/adrian/zlib-1.2.1.1-2.1.i386.rpm: (sha1) dsa sha1 md5 gpg OK
```

Intégrité des paquets

La commande suivante vérifie l'intégrité du paquet **bash** :

```
rpm –V bash
```

Cette commande ne nous retourne rien. Si nous tapons les commandes suivantes en tant que root :

```
chown bin /bin/bash
chmod 775 /bin/bash
```

Si nous re-vérifions l'intégrité du paquet bash, cette fois-ci nous avons :

```
rpm –V bash
.M...U..   /bin/bash
```

Le gestionnaire de paquet a comparé l'état de chaque fichier du paquet **bash** avec l'état de ces fichiers stocké dans sa base de données. Les modifications sur **/bin/bash** ont été identifiés.
Il est possible de faire un contrôle d'intégrité sur tous les paquets installés sur le système en ajoutant l'option **"a"** (--all) après **"V"** (--verify).

L'option --verify effectue un certain nombre de tests sur chaque fichier. Les caractères suivants permettent d'identifier les erreurs :

Caractère retourné	**Description de l'erreur**
.	le test s'est déroulé avec succès
||?||le test n'a pas pu être effectué|	
S	la taille du fichier a changé
M	le mode de permission ou le type de fichier a changé
5	le hachage MD5 du fichier a changé
D	le code majeur/mineur du périphérique ne correspond pas
L	le lien symbolic est cassé
U	le propriétaire du fichier a changé
G	le groupe propriétaire du fichier a changé
T	la date de modification (mtime) a changé

Pour aller plus loin : construire des paquets RPM (n'est pas au programme des LPIs)

: Ce paragraphe ne fait pas partie des objectifs pour l'examen LPI 101. En suivant cette partie, vous

rencontrerez peut-être des problèmes avec l'option `--rebuild`. Ce problème est du au fait que les nouvelles versions de RPM utiliser **rpmbuild** au lieu de **rpm** pour reconstruire des paquets.
Le code source de quantité de paquets RPM est également disponible en tant que paquet RPM est peut être utilisé pour construire un paquet binaire. La convention pour l'attribution du nom de fichier est :

```
nom-version_programme-version_paquet.src.rpm
```

Ces paquets contiennent au minimum deux fichiers : l'archive tar avec le code source et un fichier spec. Le fichier spec contient les instructions pour les appliquer les rustines (patchs), compiler et construire les paquets RPM. S'il faut appliquer un patch avant de compiler le paquet, alors le patch est inclut dans le paquet source.
Il y a trois méthodes pour construire un paquet RPM. Nous partirons d'un paquet nommé nom-version_programme-version_paquet.src.rpm.

Pour commencer, vous aurez besoin d'installer le paquet **rpm-build** :

- **Méthode 1** : Installer le paquet source avec :

  ```
  rpm –ivh nom-version_programme-version_paquet.src.rpm
  ```

  Vous retrouverez les fichiers dans les répertoires suivants :

  - /usr/src/redhat/SPECS
  - /usr/src/redhat/SOURCES

  Dans le répertoire /usr/src/redhat/SPECS, vous pouvez désormais voir un fichier appelé nom.spec (où "nom" est le nom du paquet). Pour commencer à construire le paquet name-version-release.i386.rpm, nous tapons la commande suivante :

  ```
  rpm –ba nom.spec
  ```

  Cette commande lance une série de scripts. L'archive dans /usr/src/redhat/SOURCES est décompressée dans /usr/src/redhat/BUILD. Si la compilation a réussi, le paquet binaire est sauvé dans /usr/src/redhat/RPMS/. Il y a plusieurs sous-répertoires correspondants aux différents processeurs, modèles et générations. Si la compilation n'est pas spécifique à l'architecture, le paquet est placé dans le sous-répertoire noarch.

- **Méthode 2** : Cette méthode suit la même chaîne d'évènements que la précédente, mais tout est fait en une seule commande :

  ```
  rpm –-rebuild  nom-version_programme-version_paquet.src.rpm
  ```

- **Méthode 3**: Parfois les développeurs mettent à disposition une archive tar avec un fichier spec. Si l'archive s'appelle nom-version_programme.tar.gz, vous pouvez rechercher le fichier spec avec la commande suivante :

  ```
  tar tzvf nom-version_programme.tar.gz | grep .spec
  ```

  Si l'archive contient un fichier spec, alors vous pouvez construire le paquet RPM en tapant :

  ```
  rpm –bt nom-version_programme.tar.gz
  ```

Gestion des paquets Debian

Les systèmes basés sur Debian utilisent le gestionnaire de paquets propre à Debian en lieu et place de la gestion des paquets RPM. Le gestionnaire de paquet Debian est plus rigoureux est plus configurable que les rpm, mais pour des raisons historiques est moins largement utilisé (**NdT** : peut-être encore vrai au moment de la rédaction du document mais ça me semble beaucoup moins vrai aujourd'hui).
L'approche utilisée pour la gestion de paquets sous Debian est très proche de celle des RPMs. La comande correspondant à **"rpm"** sur Debian est **"dpkg"**.

Conventions de noms pour les paquets

De même que pour les RPMs, les paquets Debian sont des fichiers appelés comme suit :

`nom_version-versionpaquet_architecture.deb`

La version du paquet indique la version Debian de la version du logiciel et l'architecture indique l'architecture du système (i386, sparc, all).

dpkg

dpkg est l'outil de base pour installer, construire, supprimer et gérer les paquets Debian. Cependant, on utilise plus souvent d'autres programmes qui contrôlent **dpkg** pour la gestion des paquets : **apt**, **dselect** (NdT : **aptitude**). **dpkg** prend plusieurs paramètres dans la ligne de commande : une action et zéro ou plusieurs options. Le paramètre <action> dit ce que **dpkg** doit faire et les options modifient l'action d'une manière ou d'une autre.
dpkg conserve des renseignements utiles sur les paquets disponibles. Cette information est divisée en trois classes : les **états**, les **états de la sélection** et les **drapeaux**. La modification de ces valeurs est principalement dévolue à dselect.

États des paquets

État	Description
installed	Le paquet est dépaqueté et correctement configuré.
half-installed	Le paquet est dépaqueté et la configuration a commencé mais, pour une quelconque raison, ne s'est pas terminée.
not-installed	Le paquet n'est pas installé sur le système.
unpacked	Le paquet est dépaqueté mais n'est pas configuré.
half-configured	Le paquet est dépaqueté et la configuration a commencé mais, pour une quelconque raison, ne s'est pas terminée.
config-files	Seuls les fichiers de configuration du paquet existent sur le système.

Drapeaux des paquets

Drapeau	Description
hold	dpkg laisse de côté un paquet marqué **hold**, à moins qu'il ne soit lancé avec l'option de forçage **--force-hold**.
reinst-required	Un paquet marqué **reinst-required** est défectueux et demande une réinstallation. **dpkg** ne peut supprimer de tels paquets, à moins qu'il ne soit lancé avec l'option de forçage **--force-reinstreq**.

Actions

Le cœur du fonctionnement de **dpkg** est le paramètre <action>. Le tableau suivant résume les principales actions de base dont vous êtes succeptibles d'avoir besoin.

Action	Description

-l	Affiche la liste des paquets installés sur le système, ou correspondant à un motif si précisé. Les trois premiers caractères de chaque ligne indiquent l'état, l'état de sélection et le drapeau pour le paquet
-s	Donne l'état du paquet indiqué
-I	Affiche des renseignements sur un paquet (fichier **.deb**)
-L	Affiche la liste des fichiers installés qui appartiennent à paquet
-S	Affiche le paquet qui inclue le fichier spécifié
-i	Installe ou met à jour et configure un paquet à partir d'un fichier **.deb**
--unpack	Dépaquète (uniquement) un paquet à partir d'un fichier **.deb**
--configure	Reconfiguration d'un paquet dépaqueté. Si l'on donne l'option **-a** ou **--pending** au lieu de paquet, tous les paquets dépaquetés mais non configurés sont configurés
-r	Supprime un paquet installé. L'action **-r** ou **--remove** supprime tout sauf les fichiers de configuration
-P	L'option -P ou --purge supprime tout, y compris les fichiers de configuration
--get-selections	Obtient la liste des sélections des paquets, et l'envoie sur la sortie standard
--set-selections	Modifie la liste des sélections des paquets en lisant un fichier sur l'entrée standard

Options

Toutes les options peuvent être soit précisées à la ligne de commande ou dans le fichier de configuration de **dpkg /etc/dpkg/dpkg.cfg**. Chaque ligne du fichier de configuration est soit une option (la même que le paramètre de la ligne de commande mais sans les "-"), soit un commentaire (commençant par "#").

Option	**Description**
-force-quelque-chose	Force **dpkg** à exécuter une action anormale (par exemple, ignorer les informations de dépendances --force-depends ou effectuer une mise à niveau inférieure avec --force-downgrade)
--refuse-quelque-chose	Refuser une action "normale" de **dpkg**
--ignore-depends	Ne tient pas compte de la vérification des dépendances pour les paquets spécifiés
--no-act	Faire tout ce qui doit être fait, mais n'écrire aucune

	modification (aussi : **--simulate**)
-R	Traite récursivement tous les simples fichiers qui correspondent au motif *.deb et qui se trouvent dans les répertoires et sous-répertoires spécifiés (utilisé avec **-i** ou **--unpack**

Fichiers

dpkg utilise un certain nombre de fichiers pour son fonctionnement, en commençant par son fichier de configuration **/etc/dpkg/dpkg.cfg**.
La liste des paquets avec leurs statuts est tenue dans les fichiers **/var/lib/dpkg/available** et **/var/lib/dpkg/status**.

Un fichier **.deb**, en plus des fichiers qui forment le paquet (programmes, bibliothèques, fichiers de configuration), inclue également un certain nombre de fichiers de contrôle. Ces fichiers permettent l'exécution de scripts avant et après l'installation ou la suppression du paquet, ainsi que les listes des fichiers et des fichiers de configuration. Une fois le paquet installé, ces fichiers sont placés dans **/var/lib/dpkg/info**.

Utilisation de dpkg

Pour installer un paquet à partir d'un fichier **.deb**, vous pouvez utiliser **dpkg** comme suit :
```
dpkg –i hello_2.1.1-4_i386.deb
```
OU
```
dpkg --unpack  hello_2.1.1-4_i386.deb
dpkg --configure hello
```
Pour supprimer le paquet *hello* ainsi que sa configuration :
```
dpkg –P hello
```
Tandis que la commande suivante supprime le paquet en laissant les fichiers de configuration :
```
dpkg –r hello
```
Pour obtenir la liste des paquets installés sur le systéme :
```
dpkg –l
```
Lorsque vous travaillez avec un fichier **.deb**, vous devez donner le nom du fichier, mais lorsque vous travaillez sur des paquets installés, vous ne donnez que le nom du paquet.

APT

La commande **dpkg** est adaptée pour l'installation de paquets individuels sans dépendances, mais lorsqu'il s'agit d'installer un certain nombre de paquets qui risquent d'avoir des dépendances, on préfère la suite de commandes **APT**.
APT est l'une des forces de **dpkg**. C'est un moyen simple pour installer et mettre à jour un système. **APT** utilise deux fichiers de configuration principaux :

Fichier	Description
/etc/apt/apt.conf	Options de configuration pour APT, comme la version de Debian installée, les paramètres du serveur mandataire (proxy), etc.
/etc/apt/sources	Sources utilisées pour les paquets : sur cédérom, sur le réseau, etc.

En général, on commence par configurer les sources d'APT, ce qui peut être fait avec :
```
apt-setup
```
Cette commande demande à l'utilisateur le mirroir à utiliser, et le teste, ou si vous utilisez des cdroms, la

commande suivante :
```
apt-cdrom
```
Une fois qu'APT sait où il trouve ses paquets, on utilise deux commandes pour la gestion des paquets : **apt-cache** et **apt-get**.

apt-cache

apt-cache est utilisé pour accéder aux informations du cache **APT** (placées dans **/var/cache/apt**).
Syntaxe :

```
apt-cache <action> chaîne
```
Le tableau suivant présente les actions courantes :

Action	Description
search	Recherche la chaîne dans les descriptions de paquets disponibles, et affiche une description courte des paquets correspondants
show	Affiche la description complète du paquet

apt-get

Tandis qu'**apt-cache** permet d'accéder aux informations sur les paquets disponibles, **apt-get** permet de mettre à jour ces informations, la récupération, l'installation et la suppression des paquets, et même de mettre à jour la distribution. Tout comme **apt-cache**, il faut indiquer à **apt-get** une action. Le tableau suivant décrit les actions les plus courantes pour **apt-get** :

Action	Description
update	Met à jour la liste des paquets à partir des sources définies dans /etc/apt/sources.list
install package	Installe le(s) paquet(s) indiqué(s) ainsi que leurs dépendances
upgrade	Met à jour tous les paquets pour lesquels une version plus récente existe
dist-upgrade	Met à jour la distribution (il vaut mieux lire les notes de version avant !)
remove	Supprime le(s) paquet(s) spécifié(s)

Utilisation d'APT

L'une des utilisations d'APT les plus courantes est la mise à jour système (par exemple pour installer les mises à jour de sécurité). On le fait généralement avec ces deux commandes :
```
apt-get update
apt-get upgrade
```
Une autre utilisation d'APT est l'installation de paquets, pour laquelle on utilise les commandes suivantes :
```
apt-get update            #met à jour la liste des paquets

apt-cache search frob      #recherche les paquets relatifs à frob

apt-cache show frobnicate  #affiche les informations sur un paquet en particulier

apt-get install frobnicate #installe le paquet frobnicate  et ses dépendances
```

La commande alien

La commande **alien** convertit les paquets Debian en paquets RedHat et vice-versa. Vous pouvez télécharge **alien** sur (NdT : aujourd'hui **alien** est distribué comme paquet sur Debian et autres)
Convertir un paquet Debian en rpm :

```
alien --to-rpm  paquet.deb
```

Convertir un rpm en deb :

```
alien --to-deb paquet.rpm
```

(NdT) : tâchez de vous rappeler qu'il faut utiliser **alien** en tant que **root** pour qu'il traite correctement les propriétaires et groupes des fichiers.

Résumé et exercices

Questions de révision

Oui ou Non

1. Lorsqu'on compile un programme à partir des sources, on doit compiler chaque partie du code dans l'ordre avec **gcc** : _Le programme **make** ne peut compiler un programme que s'il est lancé dans un répertoire contenant la **Makefile** approprié : _Les paquets contenant des binaires et ceux contenant des sources sont deux types de paquets RPM : _Il est recommandé de lancer **ldconfig** lorsqu'on installe des bibliothèques partagées à partir des source : _La commande **ldconfig** est utilisée pour mettre à jour le **ldcache** : _On peut effectuer des recherches en utilisant un gestionnaire de paquets sur les programmes installés à partir des sources : _Les outils **APT** permettent d'installer des paquets et des résoudre les dépendances : _Voir les réponses

Glossaire

Terme	Description
construction	terme utilisé lorsqu'on compile un projet à partir des sources, en général en tapant **make**
compiler	traduire les instructions écrites dans un langage de haut-niveau en code machine. La sortie de la compilation est appelée le code objet
bibliothèque dynamique	bibliothèque qui peut être chargée pendant l'exécution du programme
bibliothèque partagée	bibliothèque prévue pour être utilisée par plus d'un programme. Les bibliothèques partagées sont généralement dynamiques et portent l'extension .so (pour shared object)
bibliothèque statique	bibliothèque copiée dans l'exécutable au moment de la compilation. Les bibliothèques statiques portent l'extension .a (archive)
langage de haut niveau	langage de programmation lisible par les êtres humains et utilisé pour écrire du code source
éditeur de liens	1. programme utilisé pendant la compilation pour assembler les objets générés par le compilateur en un exécutable - voir ld(1)
1. programme qui charge dynamiquement les bibliothèques partagées nécessaires au lancement des programmes - voir ld.so(8)	
code objet	code produit par la compilation. Le code objet peut être soit un exécutable, soit il peut être lié à d'autre code objet pour former un exécutable
tarball	archive tar compressée

Fichiers

Fichier	Description
/etc/ld.so.conf	fichier de configuration de **ldconfig**
Makefile	fichier lu par **make** à la construction d'un programme
/etc/rpmrc	utilisé par **rpm** et **rpmbuild** (voir LPI 201), ce fichier contient des informations telles que l'architecture système et le chemin pour accéder aux macros et les utilitaires utilisés pour la gestion des paquets. Ce fichier se trouve en général dans le répertoire /usr/lib/rpm/
/usr/lib/rpm/	répertoire contenant les macros utilisées pour la gestion des paquets
/var/lib/rpm/	répertoire contenant les bases de données pour le gestionnaire de paquets (RPM)

Commandes

Commande	Description (apropos)
alien	**alien(1)** - Convertir ou installer un paquetage binaire d'une autre distribution. **alien** permet la conversion entre les formats de paquets RedHat (rpm), Debian (deb), Stampede (slp), Slackware (tgz) et Solatis (pkg). Si vous souhaitez utiliser un paquet provenant d'une autre distribution Linux que celle que vous avez installé, vous pouvez utiliser **alien** pour le convertir au format désiré et l'installer
les outils APT	Suite de commandes permettant d'effectuer des opérations avancées sur des paquets Debian situés sur cédérom ou serveur
configure	script généralement inclut dans les sources d'un projet pour générer le Makefile. Ce script tente de déterminer le type d'architecture, et les autres composants nécessaires à la construction du projet (compilateur, fichiers d'entête, ou bibliothèques)
dpkg	commande utilisée pour manipuler les paquets au format Debian
LD_LIBRARY_PATH	variable d'environnement utilisée par l'éditeur de liens (**ld.so**) contenant le chemin vers des bibliothèques partagées
ldconfig	programme qui génère le "ldcache" utilisé par l'éditeur de liens pour trouver les bibliothèques partagées. L'option **-p** permet d'afficher le contenu du cache
ldd	**ldd(1)** - Affiche les bibliothèques partagées nécessaires

make	**info make** - La commande **make** détermine les parties d'un gros programme à recompiler et lance les commandes pour les recompiler
rpm	commande utilisée pour manipuler les paquets au format RPM

Travaux pratiques

Dans l'exemple suivant, téléchargez un paquet RPM source (bash-2.05-8.src.rpm pour RedHat 7.2) à partir de www.rpmfind.net. (NdT : mettez à jour en fonction de la distribution RPM que vous utilisez)

1. Installation à partir de l'archive tar
   - Installez le contenu du paquet RPM sans compiler quoi que ce soit :

     ```
     rpm –ivh bash-2.05-8.src.rpm
     ```
   - dans le répertoire */usr/src/redhat/SOURCES*, décompressez l'archive avec :

     ```
     tar xvzf bash-2.05-8.tar.gz
     ```
   - Facultatif (mais recommandé) : appliquez les correctifs (patchs). La syntaxe dépend du répertoire où vous vous trouvez.

     de */usr/src/redhat/SOURCES* :

     ```
     patch –p0 –b < fichier.patch
     ```
     de */usr/src/redhat/SOURCES/bash-2.05-8*

     ```
     patch –p1 –b < fichier.patch
     ```
   - nous faisons le choix d'installer les fichiers dans un répertoire temporaire, par exemple */tmp/test*. Dans un environnement de production, nous placerions les fichiers dans */usr/local*. Créons le répertoire :

     ```
     mkdir /tmp/test
     ```
   - Enfin, compilons en suivant les étapes habituelles :

     ```
     ./configure –prefix=/tmp/test
     make
     make install
     ```
     Vous pouvez lister le contenu de **/tmp/test**/
2. Pour aller plus loin, nous allons reconstruire un paquet RPM binaire (non nécessaire pour l'examen LPI101)
   - `rpm –-rebuild  paquet.src.rpm`
     Le paquet créé devrait être dans **/usr/src/redhat/RPMS**
     - Vérifiez le contenu du paquet avec les options **-qpl**
     - installer le ou les paquets et lancez des requêtes sur le(s) paquet(s) installé(s)
     - désinstaller le(s) paquet(s)
3. Configurez **/etc/apt/sources** avec **apt-setup**. Utilisez les commandes **APT** et **dpkg** pour lancer des requêtes, installer et mettre à jour les paquets disponibles.

Réponses aux questions

1. **non** : on utilise **make**

2. **oui** : le Makefile est unique pour chaque projet. la commande **make** ne peut lire le Makefile qu'à partir du répertoire actuel, c'est à dire le répertoire à partir duquel la commande est lancée.
3. **oui**
4. **oui**
5. **oui**
6. **non**
7. **oui**

Traitement avancé du texte

Pré-requis

- La ligne de commande
- Traitement du texte

Objectifs

- Faire la distinction entre la syntaxe utilisée pour l'expansion des noms de fichiers (file globbing) et les expressions rationnelles
- Utiliser efficacement les commandes **grep**
- Comprendre des commandes **sed** simples

Présentation

On utilise **grep**, **fgrep** ou **egrep** pour rechercher un ou plusieurs mots dans un texte. Les mots utilisés pour la recherche sont une combinaison de lettres que l'on appelle les expressions régulières (ou expressions rationnelles). Les expressions rationnelles sont également reconnues par d'autres applications comme **sed** ou **vi**.

Les expressions rationnelles

Les expressions rationnelles traditionnelles (regex)

Une expression rationnelle est une séquence de caractères (ou atomes) utilisés pour correspondre à un motif. Les caractères utilisés sont soit des constantes (traitées littéralement) ou des caractères génériques (métacaractères).

Caractères génériques les plus courants

Caractères	Correspond à
\<CLÉ	Mots commençants par "CLÉ"
MOT\>	Mots finissant par "MOT"
^	Début de ligne
$	Fin de ligne
[Plage]	Plage de caractères ASCII
[^c]	Pas le caractère "c"
\[	Interpréter le caractère "[" littéralement
"ca*t"	Chaînes contenant "c" puis zéro ou plusieurs "a" puis "t"
"."	Correspond à n'importe quel caractère seul

Les regex étendues (eregex)

Les principales regex étendues sont : +, ?, () et |

Principales eregex

Caractères	Correspond à
"A1|A2|A3"	Chaînes contenant "A1" ou "A2" ou "A3"

"ca+t"	Chaînes contenant "ca" suivi par un ou plusieurs "a" puis "t"
"ca?t"	Chaînes contenant "ca" suivi par un ou zéro "a" puis "t"
"ca*t"	Chaînes contenant "c" puis zéro ou plusieurs "a" puis "t"

La famille grep

grep

La commande **grep** prend en charge les expressions rationnelles traditionnelles (regex).

egrep

La commande **egrep** prend en charge les expressions rationnelles traditionnelles et étendues (eregex).

fgrep

fgrep signifie "fast grep". **fgrep** interprète les chaînes littéralement : il ne prend pas en charge les expressions rationnelles.

Travailler avec grep

Syntaxe de grep :

```
grep MOTIF  FICHIER
```

Options les plus courantes de grep

-c	compte le nombre de lignes qui correspondent au MOTIF
-f	obtient le MOTIF à partir d'un fichier
-i	ignore la casse (NdT : pour le MOTIF aussi bien que pour les fichiers)
-n	indique le numéro de ligne du fichier en entrée
-v	affiche toutes les lignes sauf celles contenant le MOTIF
-w	correspondance exacte au MOTIF (NdT : **man grep**)

Par exemple, pour lister toutes les lignes non vides de */etc/lilo.conf* :

```
grep –v “^$” /etc/lilo.conf
```

egrep et fgrep

La commande **fgrep** ne prend pas en compte le sens particulier des expressions rationnelles. Par exemple :

```
fgrep ‘cat*’ FICHIER
```

ne trouvera que les mots contenant "cat*". Le principal avantage de **fgrep** est la possibilité de chercher à partir d'une liste de mots entrée ligne par ligne dans un fichier (disons LISTE). La syntaxe pour une commande de ce type serait :

```
fgrep –f LISTE FICHIER
```

La commande **egrep** prend en compte toutes les expressions rationnelles modernes. On peut également rechercher plusieurs mots clés entrés à la ligne de commande, séparés par des tubes. Par exemple :

```
egrep “linux|^image”   /etc/lilo.conf
```

sed, l'éditeur en continu

Voilà que nous allons parler de **sed**. C'est une commande ancienne, à l'origine la seule disponible sur les systèmes UNIX pour la manipulation de texte.
On utilise généralement **sed** pour rechercher et remplacer des motifs dans du texte. Cette commande prend en charge la plupart des expressions rationnelles.

Débuter avec sed

Syntaxe de sed :
```
sed   [options] ´commande'  [FICHIER_EN_ENTREE]
```
Le fichier en entrée est facultatif, puisque **sed** peut aussi travailler avec des redirections de fichiers et des tubes.
Voici quelques exemples, pour lesquels nous travaillerons sur le fichier *MODIF*.

- Suppression de toutes les lignes commentées :

  ```
  sed '/^#/ d ' MODIF
  ```

  Note

  Vous noterez que le motif de recherche est placé entre les deux slashs //.

- Remplacement de /dev/hda1 par /dev/sdb3 :

  ```
  sed 's/\/dev\/hda1/\/dev\/sdb3/g'  MODIF
  ```
  La commande **"s"** signifie "substitute" (remplacer). Le **"g"** signifie "global" et force la substitution à être exécutée plusieurs fois si nécessaire pour chaque ligne.

- Si la ligne contient le mot clé CLE, alors remplacer ":" par ";" globalement :

  ```
  sed ' /CLE/ s/:/;/g'  MODIF
  ```

Utilisation plus avancée de sed

Vous pouvez exécuter plusieurs commandes en commençant chacune par **-e**. Par exemple, (1) supprimer chaque ligne vide et (2) remplace "ANCIEN" par "NOUVEAU" sur chaque ligne du fichier *MODIF* :
```
sed –e '/^$/ d'  -e 's/ANCIEN/NOUVEAU/g'  MODIF
```
On peut également placer ces commandes dans un fichier, disons *COMMANDES*. Dans ce cas, chaque ligne est interprétée comme une nouvelle commande à exécuter (les guillemets sont inutiles).
Exemple de fichier COMMANDES :

```
1 s/ancien/nouveau/
/mot/ s/ancien/nouveau/g
23,25 d
```
Pour utiliser le fichier *COMMANDES*, on utilisera la syntaxe suivante :
```
sed  -f  COMMANDES  MODIF
```
C'est beaucoup plus compact qu'une très longue ligne de commande !
Résumé des options de sed

Option	**Description**
-e	exécute la commande suivante
-f	lit les commandes à partir d'un fichier
-n	n'affiche pas les lignes qui n'ont pas été éditées
Commandes sed	
d	(delete) supprime la ligne en entier

r	(read) ajoute le texte lu à partir d'un fichier
s	(substitute) remplacer
w	(write) écrit la sortie dans un fichier

Résumé et exercices

Questions de révision

Oui ou Non

1. l'expression rationnelle étendue "nucle?aire" correspond à nucleaire et nuclaire : _l'expression rationnelle "baza*r" correspond à bazaar et bazar mais pas bazor : _l'expression rationnelle étendue "nucle+aire" correspond à nucleaire mais pas à nuclaire : _l'expression rationnelle "baza*" correspond à bazaar et bazar et bazor : _Voir les réponses

Commandes

Commande	Description (apropos)
egrep	affiche les lignes contenant des motifs correspondant en utilisant les expressions rationnelles étendues
fgrep	affiche les lignes contenant des motifs correspondant en utilisant des chaînes littérales
grep	affiche les lignes contenant des motifs correspondant en utilisant les expressions rationnelles
sed	**sed (1)** - l'éditeur en continu est utilisé pour effectuer des transformations de texte de base sur un fichier en entrée (fichier ou entrée provenant d'une redirection)

Travaux pratiques

1. Créez un fichier *FICHIER* contenant :

```
Using grep,
fgrep and
egrep
to grep for 99% of the cats
% these are two
% commented lines
```

   - utilisez **grep** pour afficher les lignes non commentées
   - trouvez les lignes contenant exaxtement "grep" (pas "egrep" ni "frep". Utilisez "-w")
   - trouvez les lignes contenant des mots commençant par "a"

2. Expressions rationnelles : ajoutez les lignes suivantes à *FICHIER* :

```
ct
cat
caats
caaatss
ca+t
ca*t
ca?t
crate
```

```
carts
```

- consultez les sorties des commandes suivantes en utilisant **grep**, **egrep** et **fgrep** :

```
grep 'ca+t' FICHIER
grep 'ca?t' FICHIER
grep 'ca.t' FICHIER
grep 'caa*t' FICHIER
grep 'ca*r.' FICHIER
```

3. utilisez la commande **sed** pour effectuer les modifications suivantes dans FICHIER : (utilisez un fichier COMMANDE, puis recommencez le tout en ligne de commande)
   - première ligne, remplacez "grep" par "soap"
   - supprimez "fgrep" de la deuxième ligne
   - remplacez "egrep" par "water"
   - quatrième ligne, remplacez "grep for" par "wash"

   Enregistrez le résultat dans un fichier en utilisant l'option **"w"**.

Réponses aux questions

1. **oui** : l'expression "e?" se lit "correspond à 0 ou 1 e"
2. **oui** : l'expression "a*" se lit "correspond à 0 ou plusieurs a"
3. **oui** : l'expression "e+" se lit "correspond à 1 ou plusieurs e"
4. **oui** : notez que "a*" est placé à la fin de l'expression. Ceci signifie que "baz" suivi par tout ce qu'on veut correspond.

Utilisation de vi

Pré-requis

- Aucun

Objectifs

- comprendre les trois modes de vi
- introduction aux commandes d'éditions les plus courantes
- utilisation des expressions régulières et des commandes sed

vi est l'éditeur par défaut de la plupart des distributions Linux. Il est considéré comme aussi essentiel que **grep** ou **cat**, on le trouve donc dans le répertoire **/bin**.

Les modes de vi

Pour effectuer les opérations d'édition complexes comme le copier/collé, **vi** fonctionne dans trois modes différents.

Le mode normal

C'est un mode d'édition et de navigation. En général, les commandes sont simplement une lettre. Par exemple, utilisez **j** (jump) pour passer à la ligne suivante.
Tenez comme règle d'or que si vous souhaitez effectuer une opération plusieurs fois, vous pouvez faire précéder la commande par le nombre de fois. Par exemple **10j** saute 10 lignes.

Il arrive que les touches directionnelles du clavier ne soient pas convenablement gérées. Dans ce cas, il reste possible de se déplacer en utilisant les commandes **h j k l** qui auront les effets suivants :

touche	**direction**
h	gauche
j	bas
k	haut
l	droite

NdT : il y a un moyen mnémotechnique : **j** ressemble à une flèche basse.

Le mode ligne de commande

On entre dans ce mode en tapant sur deux points (**":"**). Les deux points apparaissent en bas à gauche de l'écran. Ce mode peut être utilisé pour rechercher du texte, enregistrer, quitter, ou lancé une commande shell.

Le mode d'insertion

Le plus simple pour entrer dans ce mode à partir du mode normal est de taper **i** ou **a**. Ce mode est le plus intuitif et est surtout utilisé pour saisir du texte dans un document.

> **Note**
>
> La touche **Echap** sort du mode d'insertion et retourne au mode normal

Éléments de texte

Le mode normal permet d'utiliser des éléments de texte (mots ou des paragraphes) comme mouvements, ce qui

permet d'appliquer des commandes d'édition aux documents sans utiliser de souris.

Mots, phrases et paragraphes

e resp. b	se déplacer à la fin (**e** : end) / au début (**b** : beginning) du mot actuel
(resp.)	se déplacer au début / à la fin de la phrase actuelle
{ resp. }	sé déplacer au début / à la fin du paragraphe actuel
w	Même chose que **e** mais inclue l'espace à la fin du mot

Début et fin

^	début de ligne
$	fin de ligne
1G	début du fichier
G	fin du fichier

On peut utiliser ces mouvements pour se déplacer dans le texte un mot (**w**), ou un paragraphe (**}**) à la fois, aller au début de la ligne (^) ou à la fin du fichier (**G**). Mais on pourra également utiliser ces élements pour des commandes de suppression ou de copie.

Insertion de texte

Si, à partir du mode normal, vous tapez **i**, vous pouvez saisir du texte dans votre document. Comme pour toutes les fonctionnalités de **vi**, il y a plusieurs manière d'y arriver. Le tableau ci-dessous liste tous les modes d'insertion possibles :
Commandes d'insertion

a	ajoute du texte, le curseur se plaçant après le caractère actuel
A	ajoute du texte en fin de ligne
i	insère du texte sur la position actuelle
o	insère du texte en créant une nouvelle ligne au dessous
O	insère du texte en créant une nouvelle ligne au dessus
s	supprime le caractère actuel et insère
S	supprime la ligne actuelle et insère

Lorsqu'on modifie un document, il est très pratique de supprimer la partie du texte que vous souhaitez remplacer juste avant d'entrer en mode d'insertion, ce qui est possible grâce à la commande **c** (change). Comme pour les autre commandes de ce paragraphe, **c** vous place en mode d'insertion, mais vous pouvez également préciser la portion de texte que vous souhaitez supprimer avant. Par exemple `c$` supprime le texte du curseur à la fin de ligne.
Il existe une autre commande pour remplacer un caractère (rien d'autre !) : **r**. Tout d'abord, placez le curseur sur le caractère à remplacer puis tapez **r** puis un autre caractère. Le nouveau caractère remplace l'ancien. Avec cette commande, vous restez en mode normal, vous ne passez pas en mode insertion.

Couper / Coller

Pour supprimer un caractère à partir du mode normal, on utilise **x**, **dd** pour supprimer la ligne actuelle. On peut ensuite copier l'élément supprimé avec la commande **p**.

> **Note**
>
> Presque toutes les commandes **vi** peuvent être répétées en spécifiant un nombre avant la commande. On peut également appliquer la commande à un éléments (mot, phrase ou paragraphe) en plaçant l'élément après la commande.

Exemples :

- supprimer un mot

  ```
  dw
  ```
- supprimer le texte de notre position actuelle à la fin de ligne

  ```
  d$
  ```
- supprimer le texte de notre position actuelle à la fin du paragraphe

  ```
  d}
  ```

On peut simultanément supprimer un élément et passer en mode d'insertion avec la commande **c**. Comme toujours, vous pouvez utiliser cette commande avec un élément de texte comme **w** ou **{**.

> **Note**
>
> Le dernier éléments supprimé est toujours placé dans le tampon (buffer) et peut être collé avec la commande **p**. C'est équivalent à une opération de copier/coller

Copier / Coller

La commande de copié dans **vi** en mode normal est **y** (pour yank, la lettre **c** ayant déjà été prise pour change), et la commande pour coller est **p** (paste).
Si on copie une ligne en entier, le texte collé sera inséré sur la ligne en dessous su curseur.

La sélection du texte se fait avec les éléments de texte **w**, **l**, **}**, **$**, etc. Il y a quelques exceptions, comme l'exemple suivant :

Exemples :

- copier le texte de la position actuelle à la fin de ligne

  ```
  y$
  ```
- copier la ligne en entier

  ```
  yy
  ```
- copier 3 lignes

  ```
  3yy
  ```

Recherche et remplacement

La recherche utilise les correspondances de motifs, nous devons donc utiliser les expressions rationnelles (regex). Tout comme beaucoup d'outils de manipulation de texte tels que **grep** et **sed**, **vi** prend en charge les expressions rationnelles.
Pour effectuer une recherche, on doit être en mode normal. La barre oblique / (slash) effectue la recherche vers l'avant et le point d'interrogation **?** effectue la recherche en arrière.

On peut également effectuer des opérations de recherche et remplacement. La syntaxe est similaire à **sed**.

Exemples :

- recherche vers le bas de mots commençant par "comp" dans tout le texte :

```
/\<comp
```

- recherche vers le haut de lignes commençant par la lettre "z" :

```
Upward search for lines starting with the letter z
?^z
```

- Recherche dans l'ensemble du texte du mot clé "VAR" et remplacement par "var" :

```
:% s/VAR/var
```

Annuler et rétablir

Au point où nous en sommes, ça vaut la peine d'indiquer qu'on peut toujours annuler les modifications ! En mode commande, on annule avec la commande **u** (undo) qui fonctionnent tant qu'on n'a pas enregistré le fichier. La commande pour rétablir (redo) est **^R**.

Lancer une commande du shell

En mode ligne de commande, tout ce qui suit le point d'exclamation **!** est considéré comme une commande shell.
Par exemple, alors que vous éditez *lilo.conf* ou *grub.conf*, vous souhaitez vérifier le nom de la partition racine. Vous pouvez taper :

```
:!df /
```

Enregistrer et quitter

La commande pour enregistrer est **:w** (write). Par défaut, le document est enregistré en entier. Il arrive que **vi** refuse d'enregistrer le fichier parce que vous n'avez pas les droits suffisants. Dans ce cas, on peut essayer de forcer l'écriture avec **:w!**.
On peut également spécifier un nom de fichier alternatif. On peut également enregistrer des parties du texte sur un autre fichier ou lire et coller du texte d'autres fichiers dans le document présent. Voici quelques exemples pour illustrer ces possibilités.

Exemples :

- enregistrer le document présent comme "nouveau" :

```
:w nouveau
```

- enregistrer les lignes 15 à 34 dans un fichier appelé "extrait" :

```
:w 15,24 extrait
```

- lire le fichier "extrait". Le texte sera collé au niveau du curseur :

```
:r extrait
```

: en mode ligne de commande :

- **.** correspond à la ligne actuelle
- **$** correspond à la fin du document

Les commandes suivantes correspondent aux différentes possibilités pour quitter **vi**

:wq	enregistre et quitte
:q!	quitte sans enregistrer
:x	quitte en enregistrant (si modification
:quit	pareil que **:q**

:exit ou **:e**	pareil que **:x**
ZZ	pareil que **:x**

Résumé et exercices

Questions de révision

Oui ou Non

1. La commande **l** (L minuscule) place le curseur un cran à gauche : _La commande **3dd** supprime 3 lignes : _La commande **3wd** supprime 3 mots : _La commande **:qw** enregistre et quitte : _Voir les réponses

Commandes

Action pour vi	**Description**
c^,$	début et fin de ligne
1G,G	début et fin de document
b,e	début et fin de mot
(,)	début et fin de phrase
{,}	debut et fin de paragraphe
w,W	mot et mot incluant les césures et la ponctuation
h,j,k,l	touches de direction : gauche, bas, haut, droite
:!	appelle une commande shell
:quit,:q	quitte
:quit!,:q!	quitte en abandonnant les modifications
:wq	enregistre et quitte
:exit,:x,:e,ZZ	sort (en enregistrant les modifications si nécessaire)
/,?	recherche vers l'avant et vers l'arrière
a,A,i,o,O,s,S	place en mode insertion
c	place en mode insertion en modifiant un élément
r	remplace un caractère seul en mode normal
d,dd	supprime un élément ou une ligne
x	supprime un caractère
y,yy	copie un élément ou une ligne
p	colle le contenu du tampon
u,^R	annuler, rétablir

Travaux pratiques

1. En tant que **root**, copiez */var/log/messages* dans */tmp*. En utilisant les fonctions de recherche et de remplacement de **vi**, faites débuter chaque ligne par " et finir avec " ;

2. Tapez **u** pour annuler les modifications
3. Copiez */etc/lilo.conf* dans */tmp*. Éditez le fichier et jouez avec les copier / coller (**yy**/**p**) et les couper / coller (**dd** /**p**)
4. Étudiez le résultat des commandes **:x**, **ZZ**, **:quit**, **:wq**, et **:q!** : quelles commandes enregistrent et n'enregistrent pas ?
5. Étudiez le résultat des différents modes d'nsertion : **A**, **a**, **O**, **o**, **S** et **s**
6. **Facultatif** : si vous en avez le temps, le paquet **vim-enhanced** installe un programme appelé **vimtutor** qui vous guidera à travers les options de **vi** les plus courantes.

Réponses aux questions

1. **non** - regardez comment les touches **hjkl** sont placées sur le clavieR. La touche **l** est à droite et déplace le curseur à droite.
2. **oui**
3. **non - 3w** déplace le curseur 3 mots plus loin, puis **d** est une commande incomplète. On peut supprimer 3 mots avec **d3w**
4. **non** - les commandes **:qw** sont lues de gauche à droite. Donc quitter (**q**) doit toujours être en dernier.

Le Serveur X

Pré-requis

- Aucun

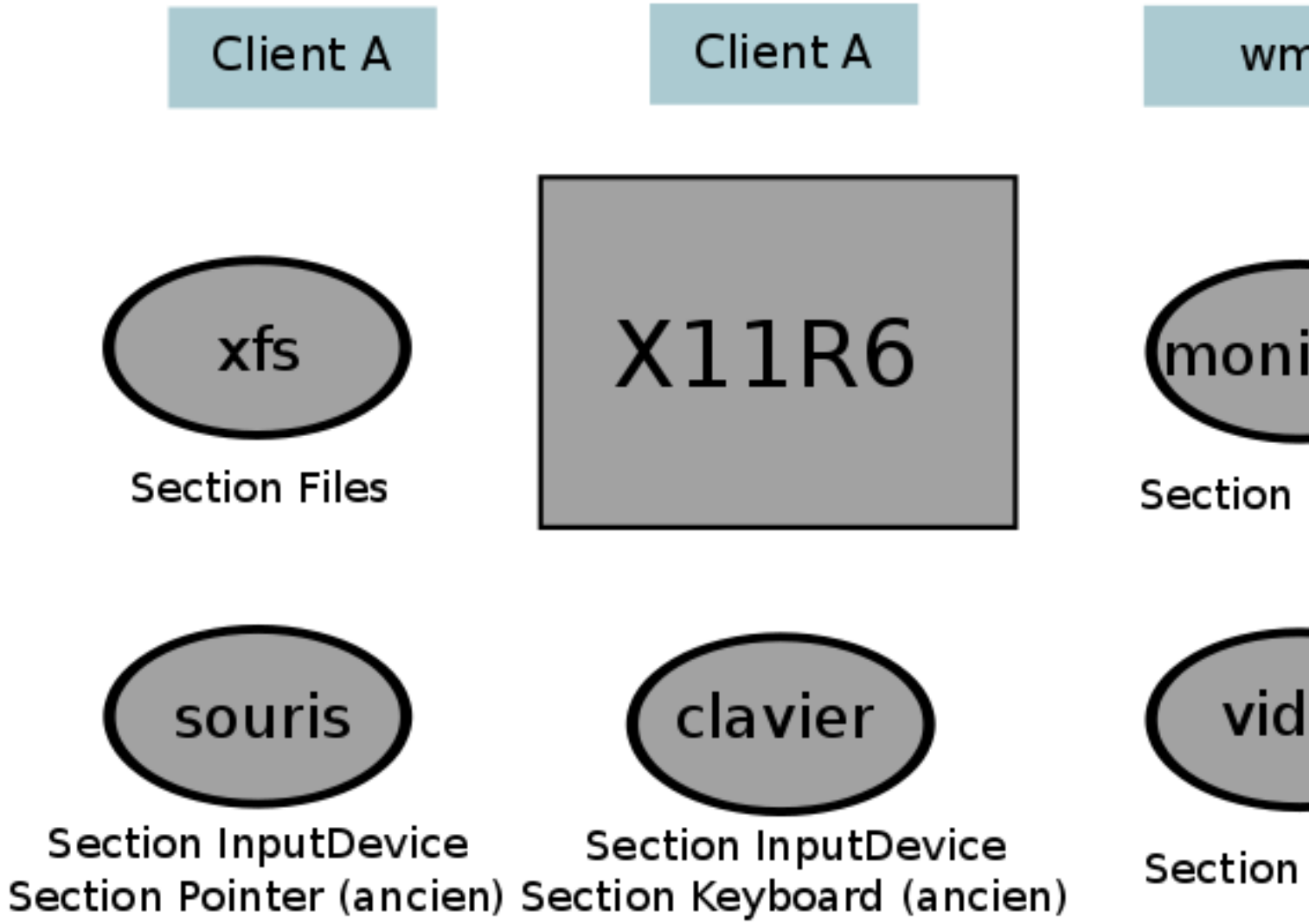
Objectifs

- Connaître les différents composants de le l'environnement "XWindow"
- Comprendre les fonctions de chaque fichier de configuration (l'édition des fichiers n'est pas nécessaire)
- Lancer des applications sur différents écrans
- Comprendre la fonction du gestionnaire d'affichage, y compris XDMCP

Introduction

Le système X Window était à l'origine le composant d'affichage graphique du projet Athena développé au MIT (Massachusetts Institute of Technology). C'est l'environnement graphique des systèmes UNIX. La version Linux du système X Window est basée sur l'adaptation libre de X Window version 11 révision 6 que l'on appelle couramment **X11R6**.
On appelle l'adaptation libre **xfree86** pour les systèmes 80386/80486 et suivants. Depuis Xfree86 a été adapté sur d'autres plate-formes comme System V/386 et 386BSD.

Illustration : Composants de X11R6 et sections de configuration



L'image précédente nous montre les différents composants du serveur **X11R6**. Le mot **Section** se réfère aux parties du fichier de configuration **XF86Config** que nous verrons plus loin.

Le deux clients représentés au dessus du serveur sont typiquement deux applications x comme x-term ou xclock. Le gestionnaire d'affichage (wm ou **window manager**) est également un client. Les gestionnaires d'affichage permettent de gérer les applications x comme des fenêtres : déplacement, sélection, réduction (iconification), etc.

Note

Remarque

La configuration du serveur **X11R6** est indépendante des clients du serveur. On configure les clients dans des fichiers de configuration spécifiques ou dans des fichiers globaux comme **Xdefaults** ou **Xresources**. La configuration du serveur contient uniquement ce qui touche au serveur de polices et les répertoires contenant les polices, à la souris, au clavier, aux résolutions prises en charge par le moniteur et à la profondeur des couleurs.

Configuration de X11R6

Xfree86 fourni en standard deux outils de configuration : **XF86Setup** et **xf86config**. D'autres éditeurs

proposent leur outils spécifiques comme :

- Xconfigurator, redhat-config-xfree86 (RedHat)
- XFdrake (Mandrake)
- sax (Suse)

Une fois le serveur configuré, on peut changer les paramètres horizontaux et verticaux du moniteur avec **xvidtune**.
Tous les outils que nous venons de mentionner créent et modifient le fichier de configuration **XF86Config**. Le serveur X lit ce fichier au démarrage, ce qui détermine son comportement. Il se situe généralement dans **/etc/X11/XF86Config**.

Il y a 11 sections de configuration dans ce fichier :

- ServerFlags
- Module
- InputDevice
- Device
- VideoAdapter
- Monitor
- Modes
- Screen
- ServerLayout
- DRI
- Vendor

Note

Remarque

Les noms de sections **Keyboard** et **Pointer** sont obsolètes mais sont encore reconnues pour des raisons de compatibilité. Elles ont été remplacées par les sections **InputDevice**.

L'une des premières section est la Section **Files**. Le mot clé **FontPath** indique s'il faut récupérer les polices à partir d'un répertoire local ou d'un serveur de polices. Le mot clé **RgbPath** est utilisé pour indiquer le chemin absolu vers le fichier **rgb** utilisé pour faire le lien entre les noms de couleurs et la notation RGB :

```
Section “Files”
        FontPath “/chemi/vers/repertoire/des/polices/”
        FontPath “trans/hote:port”
        RgbPath   "/chemin/vers/rgb"
EndSection
```

Où *trans* est le type de transport unix, *hôte* est le nom de domaine pleinement qualifié du serveur de polices, et *port* est le port de connexion, généralement 7100.
Exemple :

```
FontPath “unix/:7100”   # Serveur de police local
FontPath “unix/myfontserver.mydomain.com:7100”
```

Voici un exemple de fichier XF86Config :

```
Section "Files"
   RgbPath    "/usr/X11R6/lib/X11/rgb"
```

```
   FontPath
"/usr/X11R6/lib/X11/fonts/misc:unscaled,/usr/X11R6/lib/X11/fonts/75dpi:unscaled,/usr/X11R
6/lib/X11/fonts/100dpi:unscaled,/usr/X11R6/lib/X11/fonts/misc/"
EndSection

Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "keyboard"
EndSection

Section "InputDevice"
        Identifier  "Mouse0"
        Driver      "mouse"
        Option      "Protocol" "IMPS/2"
        Option      "Device" "/dev/psaux"
        Option      "ZAxisMapping" "4 5"
EndSection

Section "Monitor"
   Identifier      "Primary Monitor"
   VendorName      "Unknown"
   ModelName       "Unknown"
   HorizSync       31.5-37.9
   VertRefresh     55-90
   Modeline  "800x600"    40.00 800 840 968 1056 600 601 605 628 +hsync +vsync
EndSection

Section "Device"
   Identifier      "Primary Card"
   VendorName      "Unknown"
   BoardName       "None"
   VideoRam        2048
EndSection

Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 24
   BlankTime       0
   SuspendTime     0
   OffTime         0

   SubSection "Display"
      Depth        24
      Modes        "800x600"
   EndSubSection
   SubSection "Display"
      Depth        32
      Modes        "800x600"
```

Contrôle des clients X

Paramétrage des polices et des couleurs

Les clients X sont configurés via les fichiers **.Xresources** et **.Xdefaults**, situés dans les répertoires utilisateurs. Ils ne sont pas automatiquement créés, puisque des paramètres système par défaut sont également disponibles pour chaque programme.
Voici un extrait d'un **.Xresources** :

```
xterm_color*background: Black
xterm_color*foreground: Wheat
xterm_color*cursorColor: Orchid
xterm_color*reverseVideo: false
xterm_color*scrollBar: true
xterm_color*saveLines: 5000
xterm_color*reverseWrap: true
xterm_color*font: fixed
xterm_color.geometry: 80x25+20+20
xterm_color*fullCursor: true
xterm_color*scrollTtyOutput: off
xterm_color*scrollKey: on
xterm_color*VT100.Translations: #override\n\
<KeyPress>Prior : scroll-back(1,page)\n\
<KeyPress>Next : scroll-forw(1,page)
xterm_color*titleBar: false
```

Chacune de ces directives décrit comment un client est affiché. Chaque ligne est composée du nom du client suivi d'un astérisque et du paramètre X Window. Les utilisateurs peuvent donc personnaliser l'affichage des clients en configurant (avec attention) ce fichier.

La variable DISPLAY

Lorsqu'on lance une application (ou un client) X, il ou elle a besoin de savoir sur quel serveur il est lancé. Un serveur X est référencé comme un "display" (affichage). Par exemple, le premier serveur X que vous démarrez (disons avec **startx**) est appelé **:0**, le second serait **:1** et ainsi de suite. Le premier serveur X (ou display) sur l'hôte 192.168.1.99 est appelé **192.168.1.99:0**.
La plupart des clients X natifs comme xterm ou xclock disposent d'une option **-display** que l'on peut utiliser pour définir le serveur souhaité. Mais la méthode la plus simple est de définir la variable d'environnement **DISPLAY**.

Les deux commandes suivantes sont équivalentes :

```
xclock -display 192.168.1.99:0
```

```
DISPLAY=192.168.1.99:0 xclock
```

Cependant, le serveur X de l'hôte 192.168.1.99 n'autorisera pas le lancement de cette application tant que l'utilisateur qui a lancé le serveur X sur 192.168.1.99 ne l'aura pas explicitement autorisé en utilisant la commande **xhost**. Cette commande ajoute ou supprime des hôtes d'une liste de contrôle d'accès (ACL).
Exemple : Autoriser les clients X à partir de 192.168.1.7 de s'exécuter sur le serveur local

```
xhost + 192.168.1.7
     192.168.1.7 being added to access control list
```

Note

Remarque

xhost doit être utilisé avec **xauth** (qui n'est pas au programme de l'examen). Pour lancer un client de 192.168.1.7 sur notre serveur local, il nous faut encore taper les commandes suivantes sur le serveur :

xauth extract - $DISPLAY | ssh 192.168.1.7 xauth merge -

(En assumant que les noms d'utilisateurs sont les mêmes et que le nom d'hôte de la variable $DISPLAY peut être résolu).

Lancement du serveur X

On peut démarrer une session X à partir des deux méthodes suivantes :

1. En ligne de commande, une fois connecté sur la console, l'utilisateur lance le serveur X en tapant **startx**
2. Le système utilise un gestionnaire d'affichage, qui demande à l'utilisateur son nom d'utilisateur et son mot de passe. Cette méthode est généralement disponible pour un niveau d'exécution spécifique (sur les distributions de la famille RedHat, c'est le niveau d'exécution 5).

À partir de la ligne de commande

Le script **startx** lance **xinit**. Le script **xinit** prend eux rpincipaux paramètres (a) le serveur X et (b) le script **xinitrc**. Le script **xinitrc** lit (NdT : au sens de "source", **man bash**) les fichiers Xresources (qui contrôlent les applications X) et Xclients (choix du gestionnaire de fenêtres). La séquence de démarrage est donc la suivante :

```
startx --> xinit --> X -> xinitrc -> Xclients
```

En utilisant un gestionnaire d'affichage

Dans ce cas, le gestionnaire d'affichage est automatiquement lancé à un certain niveau d'exécution (5 pour RedHat). Nous allons tout d'abord décrire le processus de connexion, nous verrons les fonctionnalités plus avancées des gestionnaires d'affichage ensuite. Le processus de connexion se déroule selon les étapes suivantes :

```
xdm --> xlogin --> Xsession --> (facultatif) Xclients ou ~/.Xclients
```

Les différentes versions de gestionnaires d'affichage ainsi que les différentes distributions Linux peuvent suivre des étapes légèrement différentes. Vous noterez cependant, qu'en général, **startx** utilise **xinit** alors que **xdm** utilise **Xsession**.

> **Note**
>
> **Personnalisation**
>
> Chaque utilisateur peut personnaliser son environnement en utilisant un fichier **.xinitrc**. Ce fichier est fusionné avec le fichier système **xinitrc**.
>
> La commande **switchdesk** permet aux utilisateurs de se créer un fichier **.Xclients** personnalisé

Le gestionnaire d'affichage (display manager)

Les 3 principaux gestionnaires d'affichage sont **xdm** (génrique), **gdm** (GNOME) et **kdm** (KDE). D'après les objectifs des LPI, les fichiers de configurations se trouvent dans les répertoires suivants :

- **/etc/X11/xdm/**
- **/etc/X11/gdm/**
- **/etc/X11/kdm/** (voir le programme du LPI 101 (NdT : le serveur X est désormais au programme de l'examen 102)

Cependant, il est plus courant de trouver les fichiers de configuration pour **kdm** dans **/usr/share/config/kdm** (nous le voyons juste ci-dessous).

KDM

Ce gestionnaire d'affichage est installé avec l'environnement **KDE**. Il est basé sur le gestionnaire d'affichage générique **xdm** avec lequel il partage plusieurs fichiers de configuration. On retrouve ces fichiers de configuration dans **/usr/share/config/kdm**. Le fichier de configuration le plus important est **kdmrc**.
Le chemin d'accès au binaire est **/usr/bin/kdm**.

Fichiers de configuration pour KDM :

- **kdmrc**
- **Xaccess** (comme pour xdm)
- **Xservers** (comme pour xdm)
- **Xsession** (comme pour xdm)
- **Xsetup**
- **Xstartup**

GDM

Ce gestionnaire d'affichage est distribué avec l'environnement de bureau GNOME. Le fichier de configuration principal est **gdm.conf**.
Le chemin d'accès au binaire est **/usr/bin/gdm**.

Fichiers de configuration de GDM (/etc/X11/gdm) :

- **Sessions/**
- **gdm.conf**

XDM

Le gestionnaire d'affichage **xdm** fait partir de l'application **Xfree86**. Le fichier de configuration principal est **xdm-config**.
Le chemin d'accès au binaire est **/usr/bin/xdm**.

Fichiers de configuration de XDM :

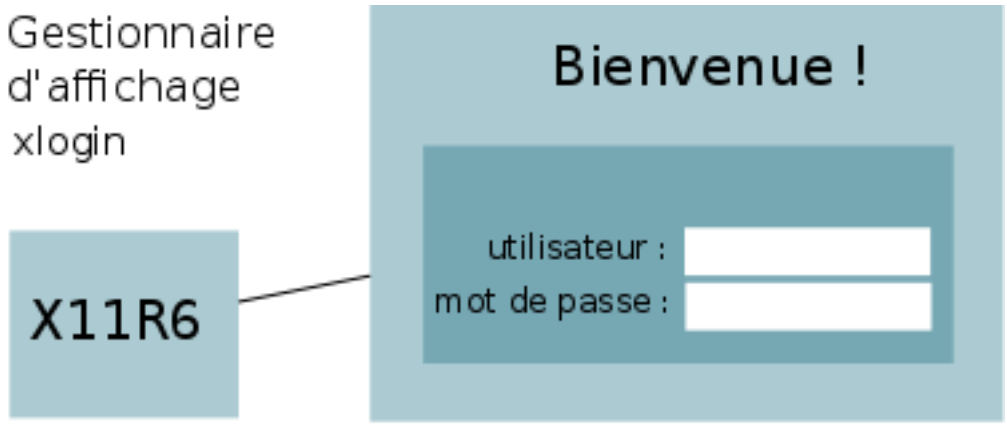
- **Xaccess**
- **Xresources**
- **Xsession**
- **xdm-config**
- **Xservers**

Nous verrons les fichiers de configuration de **xdm** plus en détail un peu plus loin.
En général, les gestionnaires d'affichages sont utilisés dans le niveau d'exécution 5 (NdT : pour les distributions de la famille RedHat)

Pour paramétrer le niveau d'exécution par défaut sur 5 dans **/etc/inittab** :

```
id:5:initdefault:
```

Les gestionnaires d'affichage permettent aux utilisateurs locaux de se connecter sur le système à partir d'une interface graphique. Mais on peut également les utiliser pour des connexions graphiques sur le réseau. Pour ce faire, ils utilisent le protocole **XDMCP** (X Display Manager Control Protocol). Par défaut, **XDMCP** est désactivé (nous l'activerons dans un exercice).
Le serveur X et le gestionnaire d'affichage



Fichiers de configuration

- **/etc/X11/xdm/Xrescources** : Le gestionnaire d'affichage étant également un client x, les polices, la couleur d'arrière plan et **xlogin** peuvent être configurés avec le fichier **Xrescources** que l'on trouve dans **/etc/X11/xdm/**. Lorsqu'on utilise **gdm**, le script **/etc/X11/gdm/Init/Default** lit (au sens de source) **Xresources**.
- **/etc/X11/xdm/Xservers** : Ce fichier se contente de faire la relation entre un affichage (display) et un serveur X. Par exemple, l'affichage (display) **:0** est interprété comme étant le serveur X local. Souvenez vous que **X** se lance toujours sur le premier **/dev/tty** libre.
- **/etc/X11/xdm/xdm-config** : C'est le fichier de configuration principal de **xdm**. On l'utilise aussi pour activer **XDMCP** (voir les exercices)
- **/etc/X11/xdm/Xaccess** : On utilise ce fichier pour activer **XDMCP**, ce qui permet aux hôtes distants de se connecter au serveur **X** local. Avec **-query**, recherche les autres serveurs (NdT : à revoir et vérifier).

Le fichier Xaccess

```
The Xaccess file
# $XConsortium: Xaccess,v 1.5 91/08/26 11:52:51 rws Exp $
#
# Access control file for XDMCP connections
# To control Direct and Broadcast access:
#
#       pattern
#
# To control Indirect queries:
#
#       pattern         list of hostnames and/or macros ...
#
# To use the chooser:
#
#       pattern         CHOOSER BROADCAST
#
# or
#
#       pattern         CHOOSER list of hostnames and/or macros ...
#
# To define macros:
#
#       %name           list of hosts ...
#
# The first form tells xdm which displays to respond to itself.
# The second form tells xdm to forward indirect queries from hosts matching
```

```
# the specified pattern to the indicated list of hosts.
# The third form tells xdm to handle indirect queries using the chooser;
# the chooser is directed to send its own queries out via the broadcast
# address and display the results on the terminal.
# The fourth form is similar to the third, except instead of using the
# broadcast address, it sends DirectQuerys to each of the hosts in the list
#
# In all cases, xdm uses the first entry which matches the terminal;
# for IndirectQuery messages only entries with right hand sides can
# match, for Direct and Broadcast Query messages, only entries without
# right hand sides can match.
#

*                                       #any host can get a login window

#
# To hardwire a specific terminal to a specific host, you can
# leave the terminal sending indirect queries to this host, and
# use an entry of the form:
#

#terminal-a     host-a

# The nicest way to run the chooser is to just ask it to broadcast
# requests to the network - that way new hosts show up automatically.
# Sometimes, however, the chooser can't figure out how to broadcast,
# so this may not work in all environments.
#

*               CHOOSER BROADCAST       #any indirect host can get a chooser

# If you'd prefer to configure the set of hosts each terminal sees,
# then just uncomment these lines (and comment the CHOOSER line above)
# and edit the %hostlist line as appropriate
#
#%hostlist      host-a host-b
#*              CHOOSER %hostlist       #
```

Le fichier Xervers

```
# $XConsortium: Xserv.ws.cpp,v 1.3 93/09/28 14:30:30 gildea Exp $
#
#
# $XFree86: xc/programs/xdm/config/Xserv.ws.cpp,v 1.1.1.1.12.2 1998/10/04 15:23:14
hohndel Exp $
#
# Xservers file, workstation prototype
#
# This file should contain an entry to start the server on the
# local display; if you have more than one display (not screen),
# you can add entries to the list (one per line).  If you also
# have some X terminals connected which do not support XDMCP,
# you can add them here as well.  Each X terminal line should
# look like:
#       XTerminalName:0 foreign
#
:0 local /usr/X11R6/bin/X
```

Un fichier Xrescources

Le gestionnaire d'affichage étant lui-même un client x, le fichier **Xressources** est similaire au fichier

.Xressources à l'execpetion qu'il contrôle l'affichage de l'écran de connexion.

```
! $XConsortium: Xresources /main/8 1996/11/11 09:24:46 swick $
xlogin*borderWidth: 3
xlogin*greeting: CLIENTHOST
xlogin*namePrompt: login:\040
xlogin*fail: Login incorrect
#ifdef COLOR
xlogin*greetColor: CadetBlue
xlogin*failColor: red
*Foreground: black
*Background: #fffff0
#else
xlogin*Foreground: black
xlogin*Background: white
#endif

XConsole.text.geometry: 480x130
XConsole.verbose:       true
XConsole*iconic:        true
XConsole*font:          fixed
```

Un fichier xdm-config

```
! $XFree86: xc/programs/xdm/config/xdm-conf.cpp,v 1.1.1.2.4.2 1999/10/12 18:33:29 hohndel
Exp $
!
DisplayManager.servers:         /etc/X11/xdm/Xservers
DisplayManager.accessFile:      /etc/X11/xdm/Xaccess
! All displays should use authorization, but we cannot be sure
! X terminals will be configured that way, so by default
! use authorization only for local displays :0, :1, etc.
DisplayManager._0.authorize:    true
DisplayManager._1.authorize:    true
!
DisplayManager*resources:       /etc/X11/xdm/Xresources
DisplayManager*session:         /etc/X11/xdm/Xsession
DisplayManager*authComplain:    false
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:     0
```

Dépannage les clients X

Il arrive que les clients X ne se ferment pas correctement et laissent des processus zombies. Un processus est dit zombie quand son parent s'est terminé et qu'il ne peut plus effacer les références au processus fils. On peut voir lorsque un processus père s'est fermé sans fermer le processus fils, en lançant la commande **ps** qui révèle que le processus fils appartient au **PID 1** (init). Il faut tuer ces processus parce qu'ils consomment des ressources. Seul l'utilisateur qui a lancé le processus ou le root peuvent le tuer et il peut être nécessaire d'utiliser le signal **-9** pour tuer ce type de processus.

Choix d'un gestionnaire de fenêtres

Dans le monde X Window, le bureau est généralement appelé l'écran. Il couvre complètement la zone d'affichage de votre moniteur. La fenêtre principale est le fond d'écran, typiquement utilisé pour afficher une couleur d'arrière plan ou une image. Le gestionnaire de fenêtres fourni une interface entre l'utilisateur et le serveur X. Il est presque impossible d'utiliser **X** sans gestionnaire de fenêtres, puisqu'il fournit la barre de titre et l'ensemble des boutons avec les quels vous manipules votre écran.
Le site regroupe des informations sur les gestionnaires de fenêtres. De nombreuses versions de ces gestionnaires de fenêtres sont disponibles sur (NdT : et notamment dans vos distributions GNU/Linux).

Voici une liste de gestionnaires de fenêtres :

- Enlightenment
- fvwm
- icewm
- amiWM
- mlvwm
- dfm
- olwm
- olvwm
- mwm
- Window Maker

Résumé et exercices

Questions de révision

Oui ou Non

1. Le fichier de configuration du gestionnaire de fenêtres est **XF86Config** : _Un client x peut être configuré pour être lancé sur n'importe quel serveur X accessible sur le réseau : _Le gestionnaire d'affichage est un programme qui gère l'affichage des pixels de votre interface graphique : _Un utilisateur qui a lancé un serveur X peut désactiver le contrôle d'accès avec la commande **xhost** : _Le protocole **XDMCP** est utilisé par les gestionnaires d'affichage pour afficher l'écran de connexion sur des hôtes distants : _Sur un système qui n'utilise pas de gestionnaire d'affichage, on lance généralement l'interface graphique avec la commande **xinit** : _Voir les réponses

Glossaire

Terme	Description
DISPLAY	variable d'environnement du shell utilisée pour indiquer aux clients X sur quel affichage (display), donc quel serveur X elles sont lancées
XDMCP	protocole fournissant un mécanisme standard permettant à un terminal de se connecter sur un serveur X distant
niveau d'exécution (runlevel)	configuration système qui permet de ne lancer qu'un groupe de processus - voir **init (8)**
client x ou application x	dans ce cours, on utilise ces termes pour les applications comme **xterm** ou **xclock** qui fonctionnent dans un environnement X
environnement de bureau	suite de programme comprenant un gestionnaire de fenêtre conçue à l'origine pour utiliser l'écran comme un 'bureau". Ces applications intégrées offrent généralement des fonctionnalités comme les

	copier/coller, couper/coller, glisser/déposer, etc. Par exemple, **XFCE**, **GNOME** et **KDE** sont des environnements de bureau.
gestionnaire d'affichage	application X lancée à un niveau d'exécution spécifique (5 dans le monde RedHat) affichant un écran de connexion graphique. Les gestionnaires d'affichage gèrent également le protocole **XDMCP**. Les gestionnaires d'affichage les plus courants sont **xdm** (générique, fourni avec **X11R6**), **gdm** (qui fait partie de la suite **GNOME**), et **kdm** (qui fait partie de la suite **KDE**)
gestionnaire de fenêtres	application X qui permet de déplacer, redimmensionner et "iconifier" les fenêtres. Certains gestionnaires de fenêtres proposent aussi une barre de tâches ainsi que des menus déroulants pour le lancement des applications. **twm**, **fluxbox**, **icewm** etc. sont des gestionnaires de fenêtre
gestionnaire de session	application qui permet d'enregistrer la session graphique lorsque l'utilisateur se déconnecte

Fichier de configuration de X11

Fichier	Description
XF86Config	fichier de configuration du serveur **X11R6**

Fichiers de configuration des gestionnaires d'affichage

Fichier	Description
gdm.conf	fichier de configuration de gdm
kdmrc	fichier de configuration de kdm
Xaccess	un des fichiers utilisé pour autoriser / interdire les accès **XDMCP**
xdm-config	fichier de configuration principal pour **xdm**
Xrescources	fichier permettant de personnaliser les gestionnaires d'affichage
Xservers	configure le nombre de gestionnaires d'affichage à lancer sur le système. Le fichier relie un "display" (:0 par défaut) à un serveur (en général X11R6)
Xsession	script utilisé par les gestionnaires d'affichage pour lancer un environnement graphique spécifique
Xclients	script utilisé par **Xsession** ainsi que **xdm** pour lancer le gestionnaire de fenêtre système

Fichiers de personnalisation

Fichier	Description
~/.Xresources ~/.Xdefaults	fichiers utilisés pour personnaliser l'affichage des applications X (position, taille de police, couleur, etc.)
~/.xinitrc	fichier utilisé par **startx** dans lequel l'utilisateur peut préciser le gestionnaire de fenêtre ou l'environnement de bureau à lancer
~/.Xclients	fichier utilisé par les gestionnaires d'affichage dans lequel l'utilisateur peut préciser le gestionnaire de fenêtre ou l'environnement de bureau à lancer

Commandes

Commande	Description (apropos)
gdm	**gdm(1) -GDM** se substitue à **XDM**. Contrairement à ses concurrents (X3DM, KDM, WDM), GDM a été écrit à partir de zéro et ne contient aucune ligne de code de XDM. GDM lance et gère les serveurs X, qu'il soient locaux ou distants (avec XDMCP)
kdm	gestionnaire d'affichage fourni avec l'environnement de bureau KDE. Il utilise de nombreux fichiers de configuration de **xdm**
startx	script qui appelle **xinit** et lance un serveur X avec le gestionnaire de fenêtre et le bureau
xauth	(ne fait pas partie des objectifs LPI) xauth(1) - Le programme xauth permet d'éditer les informations d'autorisation utilisées à la connexion au serveur X. Normalement, xauth n'est pas utilisé pour créer le fichier d'autorisation, c'est xdm qui s'en charge. Pour de plus amples informations, consultez **pam_xauth(8)**
xdm	X Display Manager, gestionnaire d'affichage faisant partie du serveur X
xf86config	script de configuration du serveur X : crée le fichier **XF86Config**
XF86Setup	programme qui crée ou modifie le fichier **XF86Config**
xhost	**xhost(1)** - la commande **xhost** est utilisée pour ajouter ou supprimer des noms d'utilisateurs ou des noms d'hôtes de la liste autorisée à se connecter au serveur X. Pour les hôtes, c'est un contrôle de sécurité assez rudimentaire, uniquement suffisant pour une station de travail mono-utilisateur
xinit	**xinit(1)** - le programme xinit est utilisé pour lancer le

	serveur X Window et le premier client X (en général, un gestionnaire de fenêtre) sur les systèmes qui ne lancent pas X directement à partir de /etc/init.d ou dans les environnements qui utilisent plusieurs systèmes graphiques. Lorsque ce premier client se ferme, xinit tue le serveur X et se termine
xinitrc	script lancé par **xinit** contenant une liste de clients X à démarrer. Les clients supplémentaire, comme les terminaux, xclock, etc., devraient être lancés en tâche de fond, alors que le premier client, en général, le gestionnaire de fenêtre, devrait être lancé au premier plan, de façon à éviter qu'xinitrc se ferme (voir **xinit (1)**)
xvidtune	**xvidtune(1)** - sans option, présente à l'utilisateur une interface lui permettant d'ajuster les modes vidéo. Il affiche également les paramètres dans un format permettant leur inclusion dans le fichier **XF86Config**

Travaux pratiques

NdT : Ces travaux pratiques ont été prévus pour des stations de travail en environnement de type RedHat. Avant de commencer, assurez vous que vous êtes en niveau d'exécution 3.

- `init 3`
- Connectez-vous sur la première console (c'est à dire **Alt +F1**)
- En tant que root, faites une sauvegarde du ichier de configuration **/etc/X11/XF86Config** existant et essayez les différents outils de configuration :
  - Redhat: Xconfigurator, redhat-config-xfree86 (8.0)
  - Mandrake : XFdrake
  - Suse: sax
  - XF86Setup
  - xf86config
  - X (avec l'option -configure)
- Lancez un serveur X en tapant **X**. Cela lancera **X11R6** seul, sans estionnaire de fenêtre. Retournez sur une console virtuelle (**Ctrl+Alt+F2**), relancez un shell puis tapez les commandes suivantes :

  ```
  export DISPLAY=localhost:0
  xterm&
  ```

  Retournez sur **X** en tapant **Ctrl+Alt+F7** (si vous n'avez pas modifié le nombre de consoles virtueles lancées dans **/etc/inittab**). Vous devriez voir un client xterm. Ensuite, tapez dans ce terminal (NdT : il faut que **twm** soit installé) :

  ```
  twm&
  ```

  Que s'est-il passé ? Pouvez-vous tuer **twm** sans tuer **X** ? Retournez sur une console virtuelle (**Ctrl+Alt+F2**) puis tapez :

  ```
  X :1
  ```

  Connectez vous sur une autre console virtuelle (tty3) puis tapez :

```
export DISPLAY=:1; xterm&
```

Vous avez désormais deux serveurs X lancés sur les "écrans" 0 et 1. Comment passez-vous de l'un à l'autre ?

- Utilisation de **XDMCP**

  Pour que cela fonctionne, assurez-vous que la ligne contenant un "*" est décommentée dans **/etc/X11/xdm/Xaccess**.

  Si vous utilisez **xdm** ou **kdm**, commentez la ligne suivante dans **xdm-config** :

  ```
  !DisplayManager.requestPort:     0
  ```

  En général, cette ligne est commentée et limite la connexion à l'écran 0 (plus sécurisé).

  Si vous utilisez **gdm**, vous devrez également éditer **gdm.conf** et ajouter :

  ```
  enable=true
  ```

  Ce qui désactive le paramètre de sécurité de **gdm**. Si votre IP est 1.2.3.4, les utilisateurs de votre réseau peuvent lancer une session sur votre serveur avec :

  ```
  X –query 1.2.3.4 :1
  ```

  ou

  ```
  X -indirect 1.2.3.4 :1
  ```

Réponses aux questions

1. **non** : **XF86Config** est le fichier de configuration du serveur X. En général, les fichiers de configuration des gestionnaires d'affichage se trouvent dans des sous-répertoires du répertoire utilisateur
2. **oui**, même si les pare-feu peuvent empêcher les clients X d'accéder aux serveurs distants.
3. **non**, les gestionnaires d'affichage gèrent l'affichage des écrans de connexion sur le serveur X local ou sur des serveurs distants avec **XDMCP**
4. **oui**, vous noterez que sur la plupart des distributions Linux récentes, vous devrez utiliser **xauth** avec **xauth**
5. **oui**
6. **non**, on utilise généralement **startx**