Collection Savoirs francophones
Série Technologies de l'information

Préparation à l'examen 101 pour la certification de l'Institut professionnel de Linux, niveau junior (LPIC-1)

Zied Bouziri (2e édition)

Niry H. Andriambelo, Andrei Boyanov, Nicolas Larrousse (1re édition)

Pour citer cet ouvrage

Z. Bouziri, N. H. Andriambelo, A. Boyanov, N. Larrousse (2010). *Préparation à l'examen 101 pour la certification de l'Institut professionnel de Linux, niveau junior (LPIC-1)*. Agence universitaire de la Francophonie, Paris. Disponible sur le Web : www.lpi-francophonie.org/spip.php?article234.

Première édition :

N. H. Andriambelo, A. Boyanov, N. Larrousse (2007). *Institut professionnel de Linux. Support de formation LPIC 101*. Agence universitaire de la Francophonie, Éditions des archives contemporaines, Paris. 167 p. ISBN 978-2-914610-51-3

Les auteurs remercient Véronique Pierre pour son appui à la relecture et à la mise en forme de l'ouvrage.

Agence universitaire de la Francophonie (AUF)
Direction de l'innovation pédagogique et de l'économie de la connaissance
4 place de la Sorbonne
75005 PARIS
France
www.auf.org

Accès et utilisation

Cet ouvrage est diffusé exclusivement au format numérique, gratuitement. Il est téléchargeable au format PDF sur le site **LPI Francophonie**, www.lpi-francophonie.org.

Le contrat Creative Commons BY-NC-SA sous lequel il est mis à disposition vous donne un certain nombre de droits, mais vous impose également de respecter un certain nombre de conditions.

Les droits

Vous êtes libre de reproduire, distribuer et communiquer cet ouvrage, tel quel ou après modification. L'ouvrage peut vous être fourni dans un format numérique modifiable sur simple demande, à envoyer à innovation@lpi-francophonie.org.

Les conditions à respecter

- **BY = Paternité** (*by*) : les noms des auteurs et éditeurs de l'ouvrage devront toujours être mentionnés, en utilisant le modèle donné (*cf.* page précédente), ceci même si vous apportez des modifications et, dans ce cas, d'une manière qui ne risque pas de suggérer qu'ils soutiennent ou approuvent les modifications apportées ;
- **NC = Pas d'utilisation commerciale** (*Non Commercial*) **:** toute diffusion payante, même après modification, est interdite ;
- **SA = Partage des conditions initiales à l'identique** (*Share Alike*) : si vous modifiez, transformez ou adaptez cet ouvrage, vous n'avez le droit de distribuer la création qui en résulte qu'en donnant les mêmes droits, et sous les mêmes conditions.

À chaque réutilisation ou distribution de cet ouvrage, ou de toute œuvre qui en serait dérivée, vous devez faire apparaître clairement au public les conditions contractuelles de sa mise à disposition. La meilleure manière de les indiquer est un lien vers cette page web :

http://creativecommons.org/licenses/by-nc-sa/2.0/fr/

Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits sur cette œuvre.

Rien dans ce contrat ne diminue ni ne restreint le droit moral de l'auteur.

Table des matières

Chapitre 1. Architecture matérielle et gestion des périphériques

A. Architecture de base de l'ordinateur

Objectifs	⇒ Se familiariser avec les notions de base de l'architecture matérielle des ordinateurs.
Points importants	Les ordinateurs de type PC ont des éléments fonctionnellement semblables bien qu'ils soient de modèles différents ou fabriqués par des constructeurs différents.
Mots clés	BIOS, CPU, mémoire, ROM, RAM, bus, dmesg

Les ordinateurs de type PC sont constitués d'un ensemble de composants assurant des fonctions semblables.

a) Le processeur (*Central Processing Unit* ou CPU)

Le processeur est un circuit électronique qui assure les fonctions centrales de l'ordinateur. C'est lui qui exécute les instructions constituant les différentes tâches demandées à l'ordinateur.

b) Mémoires

Les mémoires sont des composants électroniques pouvant garder des informations temporairement ou à long terme.

Les **mémoires centrales** sont utilisées pour stocker les informations nécessitant un accès rapide par le processeur. On distingue les **mémoires vives** (*Random Access Memory* ou **RAM**) et les **mémoires mortes** (*Read Only Memory* ou **ROM**).

Les **mémoires de masse** sont utilisées pour stocker les informations à plus long terme comme les disques, les disquettes (cf. section « disques »).

c) Le BIOS (*Basic Input/Output System*)

Le BIOS est un petit programme qui réside en mémoire morte et qui, après la mise sous tension de l'ordinateur, effectue un inventaire et un test des matériels présents. Il permet également, selon les versions, de les paramétrer.

d) Les bus

L'unité centrale d'un PC communique avec les contrôleurs de périphériques par le biais des **bus**. Un contrôleur de périphérique permet de piloter plusieurs périphériques qui lui sont rattachés.

e) Les disques

Les disques sont des périphériques de stockage. On peut citer les disques durs, les disquettes, les cédéroms, le DVD-ROM, etc.

B. Introduction à la gestion des périphériques

Objectifs	⇒ Savoir déterminer les ressources matérielles des périphériques. ⇒ Savoir utiliser des outils permettant de lister des informations sur les périphériques (par exemple : lsusb, lspci, etc.). ⇒ Savoir utiliser des outils permettant de manipuler les périphériques USB. ⇒ Comprendre les concepts de sysfs, udev, hald, D-Bus.
Points importants	Dans les noyaux 2.6 et grâce à la combinaison des systèmes de fichiers virtuels /proc et /sys, il est possible d'obtenir un instantané du système et de toutes ses périphériques.
Mots clés	ISA, PCI, AGP, IRQ, DMA (canaux), /proc, /sys, lspci, lsusb, udev

a) Les bus

Un PC utilise généralement plusieurs types de bus :

- Le bus **ISA** (*Industry Standard Architecture*) permet de transférer à 8 Mo/s et a une fréquence de 8,33 MHz. Les cartes d'extension (*slots*) supportées sont de 8 ou 16 bits.
- Le bus **PCI** (*Peripheral Component Interconnect*) est plus rapide. Il fonctionne à 33 Mhz en permettant des transferts jusqu'à 132 Mo/s en 32 bits.

- Le bus **AGP** (*Accelerated Graphical Port*), de type PCI, est réservé aux cartes graphiques. Il peut fonctionner à plus de 33 MHz.

Le bus local permet essentiellement d'accéder à la mémoire. Ce type de bus utilise la même fréquence que le processeur. Plusieurs contrôleurs nécessitant une vitesse rapide sont branchés sur ce type de bus, en occurrence les contrôleurs PCI et AGP.

b) Allocation de ressources

Un système informatique alloue des ressources aux différents contrôleurs de périphériques afin qu'elles puissent communiquer avec lui.

Lorsqu'un périphérique veut communiquer avec le processeur, il envoie une **interruption**. Cette interruption déclenche l'exécution d'un sous-programme du pilote du périphérique qui va demander du temps CPU. Le CPU interrompra alors l'activité en cours pour exécuter les demandes du périphérique. Ces interruptions sont identifiées par un numéro, l'**IRQ** (*Interrupt Request Number*), qui varie de 0 à 15.

Les canaux **DMA** (*Direct Memory Access*) permettent à un contrôleur de périphérique de transférer les données directement à la mémoire sans passer par le CPU. Ces canaux améliorent la performance dans la mesure où ils permettent des transferts rapides parfois simultanés (par opposition aux transferts octet par octet transitant par le processeur).

Les **adresses d'entrées/sorties** (*I/O ports*) sont utilisées pour que le CPU puisse communiquer avec les périphériques en lecture/écriture ou écriture. Les adresses d'entrées/sorties sont souvent de 0x100 à 0x3ff.

Les paragraphes suivants décrivent les systèmes de fichiers /proc et /sys qui permettent d'avoir des informations sur les périphériques du système telles que les ressources allouées et les pilotes utilisés.

c) Le pseudo système de fichiers /proc

/proc réside dans la mémoire système. Il n'a pas d'existence sur les disques. Il contient des fichiers qui fournissent des informations importantes sur l'état du système telles que les informations relatives aux processus, aux paramètres du noyau et aux périphériques.

Le noyau conserve les informations relatives aux ressources allouées, à savoir les interruptions reçues, la liste des canaux DMA et les entrées-sorties en cours d'utilisation, respectivement dans les fichiers suivants :

- /proc/interrupts

```
[maitre@maestro maitre]$ more/proc/interrupts
 CPU0
 0:  220494 IO-APIC-edge  timer
```



```
 1:  196    IO-APIC-edge  i8042
 2:  0      XT-PIC        cascade
 8:  1      IO-APIC-edge  rtc
 12: 9331   IO-APIC-edge  i8042
 14: 6609   IO-APIC-edge  ide0
 15: 3287   IO-APIC-edge  ide1
 16: 0      IO-APIC-level  uhci_hcd
 17: 767    IO-APIC-level  yenta, eth0, Intel 82801DB-ICH4
 18: 2      IO-APIC-level  uhci_hcd, ohci1394
 19: 0      IO-APIC-level  uhci_hcd
 23: 0      IO-APIC-level  ehci_hcd
NMI: 0
LOC: 220418
ERR: 0
MIS: 0
[maitre@maestro maitre]$
```

- /proc/dma

```
[maitre@maestro maitre]$more /proc/dma
 4: cascade
[maitre@maestro maitre]$
```

- /proc/ioports

```
[maitre@maestro maitre]$ more /proc/ioports
0000-001f : dma1
0020-0021 : pic1
0040-005f : timer
0060-006f : keyboard
0070-0077 : rtc
0080-008f : dma page reg
00a0-00a1 : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
```

d) Le pseudo système de fichiers /sys ou sysfs

Avec le noyau 2.6, la plupart des informations relatives aux périphériques ont été déplacées vers /sys.

Sysfs permet de connaître des informations sur les périphériques du système et leurs pilotes, il est également utilisé pour configurer certaines fonctionnalités du noyau.

/sys est organisé en un ensemble de répertoires, chacun contient un certain nombre de fichiers qui contiennent en général une seule valeur. Certains liens symboliques sont présents, parcourant plusieurs branches de l'arborescence du /sys.

Le *tableau 1* décrit les sous-répertoires du système de fichiers /sys.

Tableau 1. Le système de fichiers /sys

Sous-répertoire	Description
/sys/bus	peuplé de liens symboliques, représentant la manière dont chaque périphérique appartient aux différents bus
/sys/class	montre les périphériques regroupés en classes, comme les périphériques réseau par exemple
/sys/block	contient les périphériques de type bloc

Le démon udevd écoute les messages du noyau concernant les changements d'état du périphérique.

udevd utilise ces informations, ainsi que les informations sur le périphérique fournies par le noyau au travers du système de fichiers /sys, pour effectuer les opérations de chargement des modules de pilotes de périphériques, de chargement des *firmwares* et de création des fichiers spéciaux de périphériques dans le répertoire /dev.

D-Bus, système de communication inter-processus, utilise sysfs pour la diffusion d'événements système tels que « nouveau matériel ajouté » ou « file d'attente changée ».

Le démon hald est connecté à D-Bus afin d'offrir une API que les applications peuvent utiliser pour découvrir, surveiller et invoquer des opérations sur les périphériques. Cette API constitue une couche d'abstraction matérielle (*Hardware Abstraction Layer* ou HAL).

e) Les cartes PCI

Les cartes d'extension branchées sur les bus AGP ou PCI sont détectées par le système d'exploitation au démarrage. Les ressources qui leur sont allouées dépendent des spécifications des bus sur lesquels elles sont connectées et les conflits de ressources sont donc gérés quasi automatiquement. Ces ressources ainsi que d'autres informations sur ces périphériques peuvent être visualisées dans les systèmes de fichiers /proc et /sys, ou bien en utilisant les commandes dmesg et lspci.

La liste des ressources allouées par le système d'exploitation au démarrage est conservée dans le fichier /var/log/dmesg. La commande dmesg affiche le contenu de ce fichier, où sont stockés les messages du noyau. Elle permet donc également d'afficher les ressources allouées par le noyau.

La commande lspci, en examinant le contenu du fichier /proc/bus/pci, affiche un résumé synthétique des bus et cartes d'extension PCI détectés au démarrage par le système. On utilise en particulier les deux options -v, qui affiche les ressources allouées par le système à ces cartes (IRQ et adresse d'entrée/sortie), et -b qui affiche les ressources allouées par le BIOS.

Dans l'exemple suivant, nous allons examiner les caractéristiques de la carte réseau attachée à notre machine.

```
# lspci | grep -i ethernet
03:00.0 Ethernet controller: Marvell Technology Group Ltd.
Device 4355 (rev 12)
```

La sortie de cette commande donne l'identifiant du périphérique PCI.

On peut avoir plus d'informations sur ce périphérique, tels que le pilote associé et les ressources allouées.

```
# lspci -v -s 03:00.0
03:00.0 Ethernet controller: Marvell Technology Group Ltd.
Device 4355 (rev 12)
Subsystem: Toshiba America Info Systems Device ff50
Flags: bus master, fast devsel, latency 0, IRQ 218
Memory at c0100000 (64-bit, non-prefetchable) [size=16K]
I/O ports at 3000 [size=256]
Capabilities: [48] Power Management version 3
Capabilities: [5c] Message Signalled Interrupts: Mask- 64bit+
Queue=0/0 Enable+
Capabilities: [c0] Express Legacy Endpoint, MSI 00
Capabilities: [100] Advanced Error Reporting <?>
Capabilities: [130] Device Serial Number 00-23-8b-ff-ff-a3-fb-bd
Kernel driver in use: sky2 Kernel modules: sky2
```

La commande udevinfo permet d'interroger les propriétés du périphérique réseau à partir de sa représentation sysfs.

```
# udevinfo -p /sys/class/net/eth0/ -a
.....
  looking at device '/class/net/eth0':
  KERNEL=="eth0"
  SUBSYSTEM=="net"
```

```
.. ...
ATTR{address}=="00:23:8b:a3:fb:bd"
ATTR{broadcast}=="ff:ff:ff:ff:ff:ff"
....
looking at parent device
'/devices/pci0000:00/0000:00:1c.1/0000:03:00.0':
KERNELS=="0000:03:00.0"
SUBSYSTEMS=="pci"
DRIVERS=="sky2"
ATTRS{vendor}=="0x11ab"
ATTRS{device}=="0x4355"
ATTRS{subsystem_vendor}=="0x1179"
ATTRS{subsystem_device}=="0xff50"
ATTRS{class}=="0x020000" ATTRS{irq}=="218"
.....
```

f) USB

L'**USB** (*Universal Serial Bus*) est une interface qui permet de relier des périphériques à un PC. Elle possède les caractéristiques suivantes :

- connexion à chaud (« *hotpluggable* ») ;
- jusqu'à 126 périphériques peuvent être reliés au PC ;
- vitesse de transfert comprise entre 1,5 et 12 Mbit/s pour la version 1 et allant jusqu'à 480 Mbit/s pour la version 2.

Les types de périphériques pouvant être connectés sont les suivants :

- *hub* ;
- moniteur, imprimante, scanner, caméra, appareil photo ;
- périphériques audios ;
- HID (*Human Interface Device*) : clavier, souris, *joystick* ;
- périphériques de stockage : lecteur de disquettes, disques, lecteur de carte, etc. ;
- des ports série (*serial converter*).

Les **contrôleurs USB** sont intégrés à la carte mère. Ils peuvent être compatibles avec les types suivants :

- OHCI (*Open Host Controller Interface*) de Compaq (USB 1) ;
- UHCI (*Universal Host Controller Interface*) d'Intel (USB 1) ;
- EHCI pour la version 2 de la norme USB.

Les **modules du noyau** correspondant à chaque type de contrôleur sont respectivement `usb-ohci.o`, `usb-uhci.o` et `ehci-hdc.o`, mais pour qu'un

périphérique fonctionne correctement, il faudrait charger, en plus du pilote du contrôleur USB utilisé, le pilote de ce périphérique.

La commande `lsusb` fournit des informations sur les périphériques USB reliés à la machine.

```
$ lsusb
Bus 008 Device 003: ID 03f0:2b17 Hewlett-Packard LaserJet 1020
Bus 008 Device 002: ID 04f2:b008 Chicony Electronics Co., Ltd
Bus 008 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 002: ID 1058:0704 Western Digital Technologies,
Inc.
Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Dans l'exemple qui suit on peut constater qu'une imprimante USB a été récemment connectée au système :

```
# dmesg | tail
[31997.733610] usb 8-1: new high speed USB device using ehci_hcd
and address 7
[31997.886063] usb 8-1: configuration #1 chosen from 1 choice
[31997.887761] usb 8-1: New USB device found, idVendor=03f0,
idProduct=2b17
[31997.887773] usb 8-1: New USB device strings: Mfr=1,
Product=2, SerialNumber=3
[31997.887779] usb 8-1: Product: HP LaserJet 1020 [31997.887783]
usb 8-1: Manufacturer: Hewlett-Packard
[31997.887788] usb 8-1: SerialNumber: FN16V3G
[31998.044821] usblp0: USB Bidirectional printer dev 7 if 0 alt
0 proto 2 vid 0x03F0 pid 0x2B17
[31998.044821] usbcore: registered new interface driver usblp
```

La commande `udevinfo` permet de visualiser des informations détaillées sur cette imprimante, tels que le pilote associé et les ressources allouées :

```
# udevinfo -p /sys/class/usb/lp0/ -a
.....
looking at device '/class/usb/lp0':
  KERNEL=="lp0"
```

```
  SUBSYSTEM=="usb"
  DRIVER==""
.....
looking at parent device
'/devices/pci0000:00/0000:00:1d.7/usb8/8-1/8-1:1.0':
  KERNELS=="8-1:1.0"
  SUBSYSTEMS=="usb"
  DRIVERS=="usblp"
.....
  ATTRS{ieee1284_id}=="MFG:Hewlett-Packard;MDL:HP LaserJet
  1020;CMD:ACL;CLS:PRINTER;DES:HP LaserJet 1020;
  FWVER:20050309;"
```

C. Exercices

1. **Quelle commande permet d'afficher les messages de démarrage du système ?**

   - ❑ `mess`.
   - ❑ `dmesg`.
   - ❑ `bootmsg`.
   - ❑ `lsmsg`.

2. **Quelle(s) commande(s) permet(tent) de d'afficher les ressources utilisées par une carte PCI connectée à votre machine ?**

   - ❑ `lspci`.
   - ❑ `udev`.
   - ❑ `udevinfo`.
   - ❑ `less /etc/modules.conf`.

Chapitre 2. Le démarrage de Linux

Objectifs	⇒ Comprendre les étapes du démarrage de Linux pour mieux connaître le fonctionnement du système en général. ⇒ Savoir résoudre plus facilement des problèmes concernant le matériel et l'administration de la machine. ⇒ Connaître les constituants du processus de démarrage – LILO, le noyau, `init` et son fichier de configuration `inittab`.
Points importants	Comprendre le mode démarrage de Linux est fondamental, ne serait-ce que pour être capable de démarrer la machine depuis d'autres sources (cédérom, clé USB …) afin de résoudre des problèmes aussi bête que la perte du mot de passe administrateur.
Mots clés	runlevels, init, telinit, inittab, initdefault, sysinit, respawn, wait, LILO

A. Les niveaux de démarrage

Commençons dans le sens inverse du démarrage. Nous allons d'abord considérer les différent états dans lesquels le système peut se retrouver après le démarrage pour expliquer ensuite la manière de procéder pour y parvenir.

Le système Linux, comme tous les systèmes de type Unix, peut fonctionner sous plusieurs modes. On appelle ces différents modes des **niveaux d'exécution** (*runlevels*), comme par exemple « *single user mode* » pour assurer la maintenance du système ou « *multi user mode* » pour un fonctionnement standard.

Les différents niveaux d'exécution sont numérotés de 0 à 6. Le *tableau 2* décrit l'utilisation standard des niveaux de démarrage :

Tableau 2. Niveaux d'exécution standard

Niveau d'exécution	Description
Runlevel 0	*shut down* – arrête la machine
Runlevel 1	*single user mode* – mode de maintenance (un seul utilisateur peut se connecter au système)
Runlevel 2	*multi user* sans démarrer tous les services réseau
Runlevel 3	*multi user*
Runlevel 4	–
Runlevel 5	*multi user* avec interface graphique (*Display manager*)
Runlevel 6	*reboot* – redémarrage du système

Attention, certaines distributions, par exemple Debian, utilisent des niveaux d'exécutions différents.

Pour passer d'un niveau à un autre on utilise soit la commande `init` soit la commande `telinit`. Le programme `init` est celui qui est lancé le premier au démarrage du système. Le processus `init` qui lui correspond a toujours un PID de 1 :

```
ps aux | grep init
root 1 0.0 0.3 1272 220 ? S May16 0:05 init
```

Le numéro « 1 » situé après le nom d'utilisateur « *root* » représente l'identifiant du processus (le PID).

On abordera plus loin la manière de configurer le niveau de démarrage par défaut (c'est-à-dire le « *runlevel* » qui sera choisi par `init` au démarrage du système par défaut). On verra également comment indiquer au processus `init` de choisir un autre niveau de démarrage.

Pour passer à un autre niveau (par exemple le niveau 2) nous allons utiliser la commande `init` avec l'argument 2 :

```
# init 2
```

B. Configuration du démarrage par /etc/inittab et /etc/rcN.d

Les actions qui seront lancées après avoir exécuté la commande précédente sont définies en deux endroits : /etc/inittab et /etc/rc2.d.

Le fichier /etc/inittab contient des actions spécifiques pour chaque niveau d'exécution, plus d'autres configurations que l'on verra plus tard. La ligne suivante est normalement présente dans le fichier /etc/inittab :

```
l2:2:wait:/etc/rc.d/rc 2
```

Cette ligne provoque l'exécution de « /etc/rc.d/rc 2 » chaque fois que le système entre dans le niveau 2 (le deuxième champ de la ligne indique le niveau d'exécution).

Regardons maintenant le contenu du répertoire /etc/rc2.d. La commande précédente va démarrer tous les services de ce répertoire commençant par un « S » et va arrêter tous les services qui commencent par un « K ». L'ordre de démarrage/arrêt des services est défini par les deux chiffres à la suite de la première lettre.

En réalité les fichiers dans les répertoires /etc/rcN.d sont des liens symboliques vers des scripts qui se trouvent dans le répertoire /etc/init.d/. Ces scripts prennent comme argument au moins les commandes `start` et `stop`, respectivement pour démarrer et arrêter le service.

Regardons maintenant plus en détail le contenu du fichier /etc/inittab. Le format des lignes de ce fichier est le suivant :

```
id : niveau : action : commande
```

avec :

- `id` : identifiant de la ligne. Cette valeur est choisie de manière aléatoire ;
- `niveau` : niveau de démarrage pour lequel cette ligne doit être exécutée. Peut contenir plusieurs niveaux, par exemple « 2345 » ;.
- `action` : définit ce que la ligne fait ;
- `commande` : la commande à exécuter.

Analysons une partie du contenu d'un fichier inittab :

```
id:3:initdefault:
# Initialisation du système
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Que faire à l'occurrence de CTRL-ALT-DEL
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```



```
# Démarrage des terminaux virtuels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Démarrage de l'interface graphique au niveau 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Certaines lignes du fichier ne sont pas affichées dans cet exemple. Voilà les actions possibles avec leurs significations :

- `initdefault` : le niveau de démarrage par défaut ;
- `sysinit` : commande à exécuter à l'initialisation du système ;
- `wait` : exécuter la commande spécifiée et attendre qu'elle se termine pour continuer ;
- `respawn` : exécuter la commande au début et la re-exécuter dès qu'elle se termine.

À chaque modification du fichier /etc/inittab, le processus `init` doit relire son fichier de configuration pour prendre en compte les modifications. Cela se fait par la commande suivante :

```
# init q
```

C. LILO et le processus de démarrage

Au démarrage du PC le BIOS est lancé et détecte les différents périphériques. Lorsque le BIOS se termine, il charge en mémoire un petit programme (*boot loader*). Si le système est chargé à partir du disque dur, ce programme est enregistré sur le premier secteur du disque dur. Si Linux est installé sur la machine, ce programme peut être le chargeur standard de Linux **LILO** (*LInux LOader*).

LILO charge le noyau en mémoire et peut lui passer certains paramètres. De plus, il donne la possibilité de choisir parmi plusieurs noyaux compilés et configurés au préalable.

Le noyau chargé par LILO commence par se décompresser, il assure principalement l'initialisation du matériel et exécute le premier processus du système, `init`.

L'indication du chargement de LILO est donnée par l'invite (*LILO prompt*) qui permet de choisir le noyau à charger et d'indiquer les paramètres à passer au noyau ainsi qu'au processus `init` :

```
Boot: linux init s
```

L'exemple ci-dessus montre comment demander à `init` d'initialiser le système en « *single user mode* ». On peut constater sur cet exemple que le noyau chargé est nommé « linux » : il s'agit très souvent du nom du noyau par défaut.

Configuration de LILO

Pour configurer de manière permanente des paramètres du noyau nous allons modifier le fichier de configuration de LILO, /etc/lilo.conf. Dans ce fichier, comme nous l'avons déjà vu précédemment, à chaque noyau correspond une section. C'est dans cette section que nous pouvons configurer les paramètres à passer au noyau au démarrage, on utilise à cet effet le mot-clé `append` :

```
append= "pci=biosirq"
append="ram=16M"
append="/dev/hdc=ide-scs
```

Après chaque modification, il ne faut pas oublier d'installer le nouveau LILO (muni des nouveaux paramètres) avec la commande `/sbin/lilo`.

D. Le processus de démarrage

Voici les différentes étapes de la mise en route d'un ordinateur :

a) BIOS

Le programme BIOS est livré avec le PC et est spécifique au constructeur de la carte mère. Il est indépendant de Linux et ne fait donc pas partie de l'objet de notre étude. Remarquons simplement qu'il charge le « *boot loader* » LILO et qu'il détecte le matériel installé sur la machine.

b) LILO

C'est le petit programme qui permet de choisir quel noyau charger et quels paramètres lui passer. Il y a bien sûr un choix par défaut qui sera pris en compte dans le cas où l'utilisateur n'intervient pas.

c) Le noyau

Une fois que LILO charge le noyau à partir du fichier qui a été spécifié dans /etc/lilo.conf le noyau prend la main.

La première action pour le noyau est de le décompresser. Ce n'est pas toujours obligatoire que le noyau soit compressé mais il l'est souvent du fait de la limitation mémoire due à l'architecture des PC (d'où l'utilité de l'instruction `make bzImage`).

Puis le noyau initialise le matériel présent sur l'ordinateur. Dans un premier temps, il s'agit juste du matériel dont les *drivers* sont présents directement dans le noyau. Les autres matériels seront initialisés plus tard lors du chargement des modules.

À la suite de cette opération, le noyau monte le système de fichier racine (/) en lecture seule. À partir de ce moment les programmes indispensables comme ceux que l'on trouve dans /sbin/ et /bin/ sont disponibles.

Le noyau est alors prêt à créer le premier processus du système, `/sbin/init`.

d) init

Une fois créé, ce processus, comme son nom l'indique, initialise le système, en se basant sur la configuration contenue dans le fichier /etc/inittab.

Le premier programme exécuté par `init` est celui indiqué sur la ligne contenant l'action `sysinit`. Le plus souvent ce premier programme est `/etc/rc.s/rc.sysinit` qui normalement, avant d'effectuer le montage des systèmes de fichiers (indiqués dans /etc/fstab), vérifie leur intégrité à l'aide de l'utilitaire `/sbin/fsck`. Il charge ensuite les modules à partir de /etc/modules.conf (ou /etc/conf.modules sur les précédentes versions de Linux).

Puis `init` exécute les programmes configurés dans /etc/inittab pour le niveau de démarrage considéré. Ce niveau de démarrage est soit le niveau par défaut (configuré par l'action `initdefaut` du fichier /etc/inittab), soit celui passé comme argument par `lilo`. Pour les niveaux d'exécution « multi-utilisateurs », il crée les terminaux virtuels, par la commande `/sbin/mingetty` ou `/sbin/getty`. Et pour tous les niveaux, il démarre les services appropriés, d'après la configuration de /etc/rcN.d où N est le niveau de démarrage.

E. Exercices

1. **Quel est le chemin d'accès complet au fichier contenant la configuration du niveau de démarrage par défaut ?**

2. **Quelle commande affiche le niveau de démarrage précédent et celui en cours ?**

3. **Quelle action du fichier /etc/inittab est utilisée pour définir le niveau d'exécution par défaut ?**

   - ❑ runlevel
   - ❑ default
   - ❑ defaultinit
   - ❑ initdefault

4. **Modifier le niveau de démarrage par défaut.**

   OBJECTIFS

   ⇒ S'exercer à la modification de la configuration de /etc/inittab.

   ⇒ Comprendre les niveaux de démarrage.

   ⇒ Comprendre le niveau de démarrage par défaut.

   SOLUTION

   1) Nous allons mettre le niveau de démarrage par défaut à 5. Pour cela il suffit d'ouvrir le fichier /etc/inittab avec un éditeur de texte (vi par exemple) et de modifier la ligne contenant le mot clé `initdefault`.

   2) Il faut ensuite redémarrer Linux pour tester la nouvelle configuration.

5. **Configurer un nouveau terminal virtuel sur la console /dev/tty7 pour les niveaux de démarrage 2 et 3**

   OBJECTIFS

   ⇒ S'exercer à la modification de la configuration de /etc/inittab

   ⇒ Comprendre les niveaux de démarrage.

   ⇒ Savoir configurer le démarrage de différents programmes pour différents niveaux de démarrage.

   SOLUTION

   1. Modifier le fichier /etc/inittab pour ajouter une ligne identique aux lignes correspondantes au programme.

   2. Ne pas oublier de lancer la commande `init q` pour informer le processus `init` des modifications.

6. **Passer à un autre niveau de démarrage**

   OBJECTIFS

   ⇒ Comprendre les niveaux de démarrage.

   ⇒ Savoir passer d'un niveau de démarrage à un autre.

   SOLUTION

   1. Utiliser la commande `init N` (ou `telinit N`) en changeant `N` avec le niveau de démarrage voulu.

   2. Se reporter aux pages du manuel relatives à `init` et `telinit` pour voir les options possibles.

7. **Démarrer la machine en mode mono-utilisateur**

   OBJECTIFS

   ⇒ Savoir passer des options au noyau et à `init` au démarrage de la machine.

   ⇒ Savoir démarrer avec un niveau de démarrage différent du niveau par défaut.

   SOLUTIONS

   1. Au démarrage de la machine passer à Linux l'option `init s`.

   2. Si l'invite de LILO n'apparaît pas appuyer sur la touche <Ctrl> durant le démarrage.

Chapitre 3. Installation et gestion de paquetages

A. Les principes d'installation

Objectifs	⇒ Comprendre les étapes de l'installation du système et des paquetages. ⇒ Savoir identifier les différents types d'installations courantes.
Points importants	Il existe plusieurs types d'installation du système d'exploitation Linux selon l'utilisation prévue du poste. Les bons choix effectués lors de l'installation facilitent la maintenance et la gestion d'un poste de travail.
Mots clés	Choix du type d'installation.

En général, une installation du système d'exploitation Linux passe par les étapes suivantes :

1. Démarrage de l'installation, avec choix de la langue d'installation et du type de clavier. L'installation peut être effectuée à partir du système local lui-même (cédérom ou disquette) ou à partir du réseau (démarrer avec un système minimal à partir d'une disquette ou d'un cédérom qui permettra de configurer l'accès réseau). Les fichiers d'installation eux-mêmes se trouvent sur une autre machine du réseau : le serveur ;
2. Partitionnement du disque dur : le schéma de partitionnement dépendra de la taille du disque ainsi que du type d'utilisation du poste. Les outils de partitionnement intégrés dans l'interface d'installation peuvent être `disk druid`, `fips`, `fdisk`, `sfdisk` ou `cfdisk` selon les distributions ;
3. Choix des paquetages à installer ;
4. Configuration de l'interface graphique : choix des cartes graphiques, des gestionnaires de fenêtres et paramétrage de l'affichage. L'installation de la souris s'effectue également à cette étape ;

5. Configuration du réseau local : installation des cartes réseaux. Configuration réseau du poste ;
6. Choix du fuseau horaire ;
7. Configuration de l'imprimante ;
8. Définition du mot de passe du super-utilisateur (*root*) : la création éventuelle des premiers utilisateurs peut être effectuée lors de cette étape ;
9. Création de la disquette de démarrage ;
10. Configuration du chargeur de démarrage.

B. Les chargeurs de démarrage : LILO/GRUB et le MBR

Objectifs	⇒ Comprendre le fonctionnement des chargeurs de démarrage. ⇒ Connaître le fonctionnement des deux types de chargeur. ⇒ Savoir identifier les fichiers de configuration. ⇒ Être capable de modifier les fichiers de configuration.
Points importants	Les deux principaux chargeurs de démarrage sont LILO et GRUB qui permettent de charger un des systèmes d'exploitation installés sur la machine.
Mots clés	LILO, /etc/lilo.conf, /boot/grub/grub.conf, grub-install

Au démarrage d'un ordinateur, un programme de chargement situé dans le BIOS charge la routine de démarrage stockée sur le MBR (*Master Boot Record*) qui est situé en général sur le premier secteur du disque.

Ce MBR contient :

- un chargeur (*loader*) qui va à son tour charger un autre programme chargeur (*second stage loader*) propre au système d'exploitation à charger ;
- la description des tables de partitions du disque en cours ;
- une valeur numérique (*magic number*) parfois utilisée pour vérifier l'intégrité du MBR lui-même.

Les deux principaux chargeurs du système Linux sont **LILO** (*Linux Loader*) et **GRUB** (*Grand Unified Bootloader*), tous les deux installés dans le MBR.

Une fois lancé, LILO procède à la deuxième étape du processus de chargement qui se situe généralement dans /boot/boot.b. Ce fichier permet également le démarrage d'autres systèmes d'exploitation. Le fichier de

configuration de LILO est /etc/lilo.conf. Ce fichier est lu à l'exécution de la commande /sbin/lilo qui crée, à partir des configurations lues, le fichier /boot/map. Ce dernier contient les emplacements physiques des blocs où se trouve le programme de démarrage, étant donné qu'à ce stade le chargeur n'est pas capable d'accéder au système de fichiers.

Le fichier /etc/lilo.conf comporte plusieurs lignes de la forme `nom=valeur`.

Exemple de fichier /etc/lilo.conf :

```
boot = /dev/hda
prompt
default = linux
timeout = 120

image = /boot/vmlinz
label = linux
root =  /dev/hda5

other = /dev/hda1
label = windows
table = /dev/hda
```

Les principales options sont les suivantes :

- `boot` : localisation de LILO (dans le MBR ou au début d'une autre partition Linux, /dev/hda représente le MBR) ;
- `install` : chemin d'accès du chargeur. La valeur par défaut de cette option est /boot/boot.b ;
- `prompt` : si cette option est présente, LILO affichera une invite et attendra une saisie de l'utilisateur ;
- `default` : nom de l'image à charger par défaut ;
- `timeout` : utilisé avec l'option `prompt`, permet de définir un délai d'attente avant le démarrage automatique ;
- `image` : chemin d'accès au noyau ;
- `label` : nom de l'image pouvant être introduit par l'utilisateur si l'option prompt est activée ;
- `root` : la partition qui contient le système de fichiers racine ;
- `read-only` : pour monter la partition en lecture seule afin que le programme de vérification de l'intégrité du disque puisse s'exécuter proprement ;
- `append` : permet de donner des paramètres aux modules du noyau ;

- `linear/lba32` : permet de commander LILO afin qu'il lise le disque en utilisant le système LBA (*Linear Block Adressing*) qui est utile pour les disques de grande taille ;
- `lba32` : permet de dépasser la limite des premiers 1 024 cylindres et donc de « *booter* » un noyau se situant n'importe où sur le disque dur ;
- `other` : permet d'indiquer la partition qui abrite un système d'exploitation autre que Linux.

L'autre chargeur, GRUB, est aussi installé dans le MBR. Pour changer les options de GRUB, on peut utiliser la commande /sbin/grub qui lancera un interpréteur de commande propre à GRUB. Les modifications seront enregistrées dans le fichier de configuration principale de GRUB, /boot/grub/grub.conf ou /boot/grub/menu.lst (qui peut si nécessaire être édité directement). Ce fichier de configuration est lu au démarrage par le programme /sbin/grub-install.

Il existe deux types de sections pour le fichier /boot/grub/grub.conf :

1. Section globale (*global*) :
- `default` : nom de l'image à charger automatiquement (la première entrée est identifiée par 0),
- `timeout` : délai d'attente de l'invite en secondes ;

2. Section image :
- `title` : nom de l'image,
- `root` : la partition qui contient le chargeur de démarrage du système (*second stage loader*) et la racine du système de fichiers,
- `kernel` : chemin du noyau en partant de la racine définie avec l'option *root*,
- `ro` : lecture seule,
- `initrd` : chemin du « *initial root disk* ».

Voici un exemple de fichier de configuration /boot/grub/grub.conf :

```
default=0
timeout=10
title Red Hat Enterprise Linux AS (2.6.8-1.523)
      root (hd0,0)
      kernel /vmlinuz-2.6.8-1.523 ro
root=/dev/VolGroup00/LogVol00 rhgb quiet
      initrd /initrd-2.6.8-1.523.img

# section to load Windows
```

```
title Windows
      rootnoverify (hd0,0)
      chainloader +1
```

C. Le gestionnaire de paquetage Debian

Objectifs	⇒ Savoir installer, mettre à jour et désinstaller des paquetages binaires Debian. ⇒ Savoir obtenir des informations sur un paquetage Debian : version, contenu, dépendances du paquetage, état d'installation... .
Points importants	Le système de gestion de paquetage Debian offre deux outils de bases pour l'installation et la suppressions des paquetage Debian : `DPKT` (*Debian package tool*) et `APT` (*Advanced package tool*). L'extension des fichiers paquetage Debian est `.deb`.
Mots clés	/etc/apt/sources.list, dpkg, dpkg-reconfigure, apt-get, apt-cache, aptitude

a) DPKG

DPKG est l'outil de gestion de paquetage pour la distribution Debian. Il permet d'installer, de désinstaller, de visualiser, de configurer et de construire des paquetages Debian. Les options courantes de la commande `dpkg` sont :

- `-i nom-application.deb` : installe l'application `nom-application.deb` ;
- `-r mon-application.deb` : désinstalle l'application `nom-application.deb` ;
- `-l | grep appli` : cherche si le paquetage `appli` est installé. Sans le `grep`, liste tous les paquetages ;
- `-L nom-application` : liste les fichiers du paquetage `nom-application` (s'il est installé) et leur emplacement ;
- `--unpack` : permet de désarchiver mais de ne pas effectuer l'opération de configuration du paquetage.

La commande `dpkg-reconfigure` permet de reconfigurer un paquetage déjà installé.

Les options contrôlant le comportement de la commande `dpkg` sont décrites dans le fichier /etc/dpkg/dpkg.cfg.

La base de données des paquetages Debian se trouve dans l'arborescence /var/lib/dpkg.

b) apt-get

Le système de gestion de paquetages Debian est doté d'un autre outil de gestion avancé appelé **APT** (*Advanced Package Tool*). APT utilise toujours la commande `dpkg` mais ajoute des fonctionnalités supplémentaires : la définition de la source des applications à installer (disques locaux, cédérom ou sur Internet par le protocole HTTP ou FTP) et la gestion des dépendances.

La définition des sources des applications à installer s'effectue dans le fichier /etc/apt/sources.list.

Voici un exemple du contenu de ce fichier :

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-
free
deb http://non-us.debian.org/debian-non-US stable/non-US main
contrib non-free
deb http://security.debian.org stable/updates main contrib non-
free
```

Les options de fonctionnement générales de la commande `apt` sont décrites dans le fichier apt.conf. Le fichier de configuration principal se trouve dans /etc/apt/apt.conf (ou parfois /etc/apt.conf), les options personnelles peuvent être modifiées dans un fichier apt.conf se trouvant dans le répertoire home de l'utilisateur.

`apt-get` se connecte à tous les sites indiqués dans /etc/apt/sources.list, et recherche la liste des programmes disponibles.

Les options de base de la commande `apt` sont :

- `apt-get install nom-paquetage` : installe le paquetage `nom-paquetage` ;
- `apt-get remove nom-paquetage` : désinstalle le paquetage `nom-paquetage` ;
- `apt-get --purge remove nom-paquetage` : désinstalle `nom-paquetage` et ses fichiers de configuration ;
- `apt-get install nom-paquetage1 nom-paquetage2-` : installe `nom-paquetage1` et désinstalle `nom-paquetage2` ;
- `apt-get remove nom-paquetage1 nom-paquetage2+` : désinstalle `nom-paquetage1` et installe `nom-paquetage2` ;
- `apt-get --reinstall nom-paquetage1` : réinstalle le paquetage `nom-paquetage1` ;

- `apt-get update` : met à jour la liste des paquetages disponibles ;
- `apt-get upgrade` : met à jour tous les paquetages pouvant être mis à jour ;
- `apt-get -u upgrade` : affiche en plus la liste des paquetages qui vont être mis à jour ;
- `apt-get dist-upgrade` : met à jour le système tout entier (nouvelle version de la Debian) ;
- `apt-get source nom-paquetage` : télécharge le paquetage source `nom-paquetage` ;
- `apt-get -b source nom-paquetage` : télécharge le paquetage source `nom-paquetage` et le compile ensuite ;
- `apt-get build-dep nom-paquetage` : télécharge les dépendances du paquetage source `nom-paquetage` qui va être compilé.

D'autres commandes de la famille `apt` existent. Elles permettent d'avoir des informations sur les paquetages. Voici les options courantes de ces commandes :

- `apt-show-versions -u` : affiche une liste des paquetages pouvant être mis à jour ;
- `apt-cache search foobar` : recherche dans la liste des paquetages disponibles les occurrences de `foobar` ;
- `apt-cache show nom-paquetage` : affiche la description de `nom-paquetage` ;
- `apt-cache depends package` : montre les dépendances de `package` ;
- `apt-file search nom-fichier` : affiche le nom du paquetage qui fournit `nom-fichier` ;
- `apt-file list package` : affiche le contenu de `package`.

c) aptitude

Le programme `aptitude` est une interface en mode texte pour la gestion des paquetages Debian. Il permet à l'utilisateur de connaître la liste des paquetages et de réaliser des tâches d'administration comme l'installation, la mise à jour et la suppression des paquetages.

Le programme `aptitude` fonctionne en mode interactif ou à partir de la ligne de commande.

D. Le gestionnaire de paquetage RPM et YUM

Objectifs ⇒ Savoir installer, ré-installer, mettre à jour et supprimer les paquetages avec RPM et YUM.
⇒ Savoir obtenir des informations sur un paquetage RPM : version, contenu, dépendances, intégrité du paquetage, signature et état d'installation (si le paquetage est installé ou non).
⇒ Savoir déterminer les fichiers relatifs à un paquetage donné, ainsi que rechercher le paquetage auquel appartient un fichier donné.

Mots clés rpm, rpm2cpio, /etc/yum.conf, /etc/yum.repos.d/, yum, yumdownloader

a) RPM (*Red Hat Package Manager*)

RPM est utilisé originellement par la distribution Red Hat mais actuellement employé par bon nombre de distributions.

La gestion des paquetages est principalement réalisée par la commande `rpm`. RPM stocke sa base de données dans le répertoire `/var/lib/rpm`. Les noms des paquetages au format RPM respectent souvent la syntaxe `nom-version-release-architecture.rpm`. Exemple : `xsnow-1.41-1.i386.rpm`

Voici les options courantes de la commande RPM :

- `-i` (ou `--install`) : installe un paquetage ;
- `-U` (ou `--update`) : met à jour un paquetage déjà installé ou l'installe s'il n'est pas encore présent dans le système ;
- `-e` (ou `--erase`) : désinstalle un paquetage ;
- `-q` (ou `--query`) : envoie une requête sur un paquetage afin d'afficher des informations ;
- `-V` (ou `--verify`) : vérifie l'intégrité d'un paquetage ;
- `-F` (ou `--freshen`) : met à jour un paquetage déjà installé ;
- `--version` : affiche la version de la commande `rpm` ;
- `--help` : affiche les options de la commande `rpm`.

Options à utiliser avec l'option `-q` (ou `--query`) :

- `c` : affiche la liste des fichiers de configuration d'un paquetage donné ;
- `f` : affiche le nom du paquetage auquel appartient un fichier donné ;
- `i` : affiche les informations relatives à un paquetage ;
- `l` : affiche tous les fichiers et répertoires relatifs à un paquetage ;
- `p` : spécifie que la requête est spécifique au fichier du paquetage ;
- `b` : crée un paquetage `rpm` à partir d'un répertoire contenant les fichiers sources ;

- `--rebuild` : crée un paquetage à partir d'un fichier de source `rpm` ;
- `--requires PACKAGE` : pour connaître la liste des paquetages dépendants d'un paquetage.

Les options spéciales :

- `--nodeps` : installe un paquetage sans se soucier des dépendances ;
- `--force` : force la mise à jour ;
- `--import` : importe le fichier de signature d'un paquetage ;
- `--checksig` : vérifie l'authenticité du paquet par sa signature ;
- `h` : ajoute l'état d'avancement d'un processus en cours ;
- `v` : mode bavard ;
- `a` : applique l'option à tous les paquetages installés.

Le fichier `rpmrc` contrôle les actions de la commande `rpm`. Il permet de définir l'architecture machine et le système sur lequel la commande `rpm` s'exécute. Il contient également des informations allant du chemin des bases de données « rpm » jusqu'aux noms des personnes ayant développé les paquetages.

Plusieurs fichiers rpmrc sont présents sur le système, /usr/lib/rpm/rpmrc, /etc/rpmrc ainsi que des fichiers rpmrc personnalisés dans le répertoire *home* de l'utilisateur. Ils sont traités dans l'ordre énoncé précédemment, les différentes options étant modifiées par les derniers fichiers considérés.

Le contenu final est affiché par l'option `-showrc` de la commande `rpm`.

```
rpm --showrc
ARCHITECTURE AND OS:
build arch            : i586
compatible build archs: i686 i586 i486 i386 noarch
build os              : Linux
compatible build os's : Linux
install arch          : i686
install os            : Linux
compatible archs      : i686 i586 i486 i386 noarch
compatible os's       : Linux
RPMRC VALUES:
macrofiles            : /usr/lib/rpm/macros:/usr/lib/rpm/i686-
linux/macros
```

La base de données des paquetages RPM se trouve dans l'arborescence /usr/lib/rpm.

Le répertoire /usr/lib/rpm/ contient également les raccourcis des commandes les plus utilisées dans le processus de création des paquetages comme la commande `patch`. Ces commandes se trouvent dans le fichier macros.

On peut extraire des données d'un paquetage RPM sans l'avoir installé. Cela peut être utile pour récupérer le code source d'un paquetage ou extraire des polices ou d'autres données.

Pour accomplir cette tâche on utilise la commande `rpm2cpio` qui prend comme seul argument le fichier RPM. Le résultat est envoyé vers la sortie standard qui peut être redirigée vers une archive cpio :

```
rpm2cpio vsftpd-2.1.0-2.fc11.i586.rpm > vsftpd-2.1.0-
2.fc11.i586.cpio
```

On peut ensuite utiliser la commande `cpio`, avec l'option `-i` pour extraire les fichiers et `-make-directories` pour créer des répertoires :

```
cpio -i -make-directories vsftpd-2.1.0-2.fc11.i586.cpio
```

Le résultat est une extraction des fichiers de l'archive cpio dans le répertoire courant. Ce sera une série de sous-répertoires qui imitent la structure de l'arborescence Linux à savoir usr, lib, etc et ainsi de suite, ou bien une archive tar contenant du code source.

b) YUM (*Yellow dog Updater Modified*)

YUM est un outil permettant l'installation, la mise à jour et la suppression des paquetages rpm.

YUM gère les dépendances entre les paquetages. Il trouve ces paquetages sur différentes sources appelées dépôts : des fichiers images ISO Red Hat, le réseau Red Hat, le site Dags, etc.

La configuration de YUM se fait à travers le fichier /etc/yum.conf qui contient deux types de sections :

- `[main]` : définit les options globales de configuration ;
- `[repository]` : définit la configuration pour chaque dépôt.

Des fichiers de configuration supplémentaires sont également lus à partir des répertoires configurés par l'option `reposdir`, du fichier yum.conf, dont la valeur par défaut est le répertoire /etc/yum/repos.d.

Les options de base de la commande `yum` sont :

- `yum install paquetage(s)` : installe la dernière version d'un paquetage ou d'un groupe de paquetages en s'assurant que toutes les

dépendances sont satisfaites ;

- `yum update paquetage(s)` : met à jour les paquetages indiqués. Si aucun nom de paquetage n'est fourni avec la commande, `update` met à jour tous les paquetages installés. Lors de la mise à jour des paquetages, `yum` s'assure que toutes les dépendances sont satisfaites ;
- `yum remove paquetage(s)` ou `yum erase paquetage(s)` : supprime du système le(s) paquetage(s) indiqué(s), ainsi que tous les paquetages qui en dépendent ;
- `yum search chaine` : cherche des paquetages dont la description, le résumé, le nom, ou le nom de l'empaqueteur, contiennent la chaîne indiquée. Très pratique pour chercher un paquetage quand on n'en connait pas le nom mais que l'on connaît quelques mots clés s'y rapportant ;
- `yum list` : affiche diverses informations sur les paquetages.

La commande `list` peut prendre plusieurs arguments :

- `yum list [all | glob_exp1] [glob_exp2] [...]` : affiche tous les paquetages disponibles et installés ;
- `yum list available [glob_exp1] [...]` : affiche tous les paquetages disponibles dans le(s) dépôt(s) pouvant être installés ;
- `yum list updates [glob_exp1] [...]` : affiche tous les paquetages dont des mises à jour sont disponibles dans le(s) dépôt(s) ;
- `yum list installed [glob_exp1] [...]` : affiche les paquetages mis à jour ;
- `yum list extras [glob_exp1] [...]` : affiche les paquetages installés sur le système qui ne sont disponibles dans aucun dépôt du fichier de configuration ;
- `yum list obsoletes [glob_exp1] [...]` : affiche les paquetages installés sur le système qui sont rendus obsolètes par des paquetages des dépôts du fichier de configuration ;
- `yum list recent` : affiche les paquetages récemment ajoutés aux dépôts.

Toutes les options décrites précédemment prennent en argument des noms de paquetages ou des expressions génériques du shell (avec des jokers). Par exemple, `yum list available 'foo*'` affichera tous les paquets disponibles dont le nom commence par « foo ». L'utilisation des apostrophes permet de s'affranchir du développement de l'expression par le shell.

On peut utiliser la commande `yumdownloader` suivie du nom de paquetage, et la dernière version de ce paquetage sera téléchargée dans le répertoire

courant. Cela peut être pratique si on a besoin de mettre à jour un système qui n'est pas connecté à Inetrnet.

E. Gestion des bibliothèques

Objectifs	⇒ Savoir gérer les librairies dynamiques.
Points importants	Les bibliothèques, partagées ou non, tiennent une place importante dans le bon fonctionnement du système Linux. Ce sont des éléments de programmes contenant des fonctions utilisées par d'autres programmes.
Mots clés	/etc/ld/so.conf, ldd, ldconfig, LD_LIBRARY_PATH

Les bibliothèques (*libraries*) sont des fonctions généralement partagées utilisées par un programme binaire sous sa forme exécutable. Il existe deux sortes de bibliothèques : les bibliothèques statiques, qui sont incluses dans le fichier image de l'exécutable, et les bibliothèques dynamiques ou partagées, quand les codes du programme ne sont pas inclus dans le fichier image de l'exécutable. Les bibliothèques statiques se présentent sous forme de fichier avec l'extension `.a` tandis que les bibliothèques partagées sont identifiées par l'extension `.so` (*shared object*). Les bibliothèques dynamiques peuvent être appelées par plusieurs programmes simultanément, et elles sont associées au processus seulement durant l'exécution.

Les bibliothèques dynamiques sont chargées par l'utilitaire `ld.so` en utilisant l'ordre de recherche suivant :

- les répertoires mentionnés dans la variable d'environnement `LD_LIBRARY_PATH` ;
- le fichier cache /etc/ld.so.cache qui contient la liste des bibliothèques des répertoires de recherche. Il est mis à jour par la commande `ldconfig` qui scrute les différents répertoires mentionnés dans le fichier /etc/ld.so.conf ;
- les répertoires /lib et /usr/lib, usr/local/lib, etc.

La commande `ldd` permet d'avoir la liste des bibliothèques partagées nécessaires à un exécutable.

F. Exercices

8. Quelle commande allez-vous utiliser pour ré-installer le chargeur de démarrage GRUB ?

❑ grub

- ❑ `install-grub`
- ❑ `grub-install`
- ❑ `grub-setup`

9. Comment chercher un paquet précis dans les dépôts Debian ?

- ❑ `apt-cache search`
- ❑ `apt search`
- ❑ `apt-get search`
- ❑ `apt-get -s`

10. Vous venez juste de télécharger le fichier `foo_1.5-20_i386.deb`. Quelle commande installera le paquet sur votre système ?

- ❑ `apt-get -d install foo`
- ❑ `aptitude foo_1.5-20_i386.deb`
- ❑ `dpkg -i foo_1.5-20_i386.deb`
- ❑ `apt-get install foo_1.5-20_i386.deb`

11. Quelle commande va lister les fichiers contenus dans `foo-1.5.20.i386.rpm` ?

- ❑ `rpm -qpl foo-1.5.20.i386.rpm`
- ❑ `rpm -qf foo-1.5.20.i386.rpm`
- ❑ `rpm -qi foo-1.5.20.i386.rpm`
- ❑ `rpm -ql foo-1.5.20.i386.rpm`

12. Comment chercher un paquet précis dans les dépôts avec YUM ?

- ❑ `yum-cache search`
- ❑ `yum search`
- ❑ `yum -s`
- ❑ `yum-cache -s`

13. À quoi sert la commande `ldconfig` ?

- ❑ À prendre en compte les nouvelles librairies dont le chemin est contenu dans le fichier /etc/ld.so.conf.
- ❑ À configurer les chemins des librairies systèmes.
- ❑ À préparer la compilation d'un programme.

Chapitre 4. Système de fichiers

A. Disques durs et partitionnement

Objectifs

⇒ Comprendre la logique de Linux en ce qui concerne la gestion du partitionnement.

⇒ Savoir réparer le partitionnement du disque avant l'installation.

⇒ Maîtriser les commandes de partitionnement.

Mots clés

MBR, fdisk

Un disque dur est composé de plateaux reliés à un moteur central, avec des têtes de lecture de part et d'autre de chacun des plateaux. Sur chaque plateau se trouvent des pistes cylindriques découpées en secteurs. L'adressage d'un secteur est une référence au cylindre, à la tête de lecture utilisée, à la piste, et enfin au secteur.

À l'installation, un disque dur n'est ni partitionné, ni formaté. **Partitionner** signifie définir sur le disque un ou plusieurs espaces, ou **partitions**, et **formater** signifie préparer une partition à recevoir des informations en utilisant un système de fichiers défini.

a) Les partitions

Une partition est définie par son type, son emplacement de début de partition et enfin soit sa taille, soit son emplacement de fin de partition. Un partitionnement est réversible (non physique).

Une seule partition est activée à la fois au niveau du BIOS : cette activation indique où le BIOS doit aller chercher le noyau du système d'exploitation pour le démarrage.

Il existe trois sortes de partitions :

- les partitions principales : leur nombre est limité à quatre et elles

supportent tous types de systèmes de fichiers ;

- la partition étendue : elle ne peut contenir que des partitions logiques et ne peut pas recevoir de systèmes de fichiers. Elle ne peut exister que s'il existe une partition principale ;
- les partitions logiques : elles sont contenues dans une partition étendue. Elles ne sont pas limitées en nombre et acceptent tous types de systèmes de fichiers.

b) Organisation des partitions sous Linux

Les descripteurs de disques durs dans le répertoire `/dev` commencent par `hd` pour les périphériques de type IDE ou par `sd` pour les périphériques de type SCSI. Une lettre additionnelle est ajoutée au descripteur pour désigner le périphérique.

Il y a généralement deux contrôleurs IDE en standard sur un PC, chaque contrôleur supportant deux périphériques (disques, lecteur de cédérom/DVD, lecteur ZIP...).

Tableau 3. Désignation des périphériques IDE

	Primaire	Secondaire
Maître	a	C
Esclave	b	D

Pour le périphérique maître sur le contrôleur primaire : `hda`

Pour le périphérique esclave sur le contrôleur secondaire : `hdd`.

Les périphériques SCSI sont désignés en fonction de leur position dans la chaîne SCSI (`sda`, `sdb`, `sdc`, etc.).

On utilise la commande `fdisk` pour configurer une nouvelle partition. Par exemple, pour le premier disque IDE :

```
fdisk /dev/hda
```

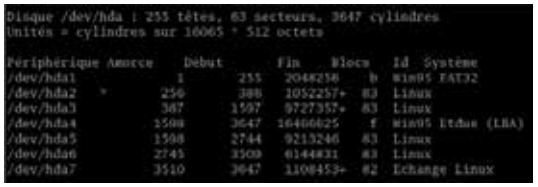
Voici une liste des différentes commandes internes de `fdisk` :

- `a` : (dés)active un indicateur « *bootable* » ;
- `b` : édite le libellé de disque bsd ;
- `c` : (dés)active l'indicateur de compatibilité DOS ;
- `d` : supprime une partition ;
- `l` : répertorie les types de partition connus ;
- `m` : affiche la liste des commandes ;

- `n` : ajoute une nouvelle partition ;
- `o` : crée une nouvelle table de partition DOS vide ;
- `p` : affiche la table de partition ;
- `q` : quitte le programme sans enregistrer les modifications ;
- `s` : crée un nouveau libellé de disque Sun vide ;
- `t` : change l'ID système d'une partition ;
- `u` : change l'unité d'affichage/saisie ;
- `v` : vérifie la table de partition ;
- `w` : écrit la table sur le disque et quitte le programme ;
- `x` : fonctions supplémentaires (experts seulement).

Ci-dessous, un exemple de table de partitionnement obtenu avec l'option « `l` » de `fdisk` :

Figure 1. Exemple de partitionnement



Le système d'exploitation utilise une zone d'échange (*swap*) sur le disque comme une extension de la mémoire physique. Selon les besoins, il y aura donc un échange entre la mémoire physique et la zone *swap*.

Linux utilise deux types de partitions : Linux (*Linux native*) et Echange Linux (*swap*) comme on peut le constater sur la *figure 1*. La première partition est une partition qui peut contenir un système Windows et la quatrième une partition de type étendu qui permet de créer des partitions logiques (*cf.* supra).

B. Arborescence des fichiers sous Linux

Objectifs ⇒ Connaître l'organisation de l'arborescence standard des fichiers sous Linux.

Mots clés boot, bin, lib, dev, etc, proc, root, home, tmp, usr, var, local

a) Les répertoires de base

Les répertoires de base de l'arborescence standard de fichiers sont les suivants (*N. B.* la racine du système est représentée par « / ») :

- /boot : contient principalement le fichier binaire du noyau ainsi que les ressources nécessaires à son lancement au démarrage ;
- /dev : contient les fichiers des périphériques (*devices*) de la machine ainsi que des fichiers spéciaux ;
- /etc : répertoire très important où sont stockés tous les fichiers de configuration du système en général et des différents démons en particulier. Il s'agit *du* répertoire à sauvegarder pour pouvoir restaurer la configuration d'une machine ;
- /home : répertoire où sont stockés par défaut les répertoires **home** des utilisateurs du système ;
- /proc : contient les informations nécessaires au noyau. C'est une arborescence virtuelle généralement en lecture seule sauf proc/sys ;
- /root : répertoire **home** du super-utilisateur (*root*) ;
- /tmp : permet aux applications et aux utilisateurs d'avoir un espace d'échange où ils peuvent stocker leurs fichiers temporaires. Il est effacé à chaque redémarrage de la machine (« *reboot* ») ;
- /usr : contient les fichiers nécessaires aux applications, la documentation, les manuels, les fichiers sources ainsi que des librairies généralement statiques et générées à l'installation des logiciels standards de la distribution ;
- /usr/local : arborescence qui sert à installer les logiciels supplémentaires ;
- /var : contient les fichiers journaux des différents démons (donc variable) ainsi que les spools de *mail*, d'impression, de *cron*, etc.

b) Les autres répertoires

- /bin et /sbin : contiennent l'ensemble des binaires indispensables au démarrage de la machine et les commandes essentielles d'administration ;
- /lib et /usr/lib : contiennent les librairies nécessaires aux commandes précédentes.

C. Formatage et types de systèmes de fichiers

Objectifs ⇒ Connaître les différents types de systèmes de fichiers reconnus par Linux et leurs spécificités.

⇒ Maîtriser la création d'un système de fichiers sur une partition.

Mots clés ext2, ext3, reiserfs, vfat, xfs, mkfs, tune2fs

Les principaux types de système de fichiers supportés par Linux sont présentés dans le *tableau 4*.

Tableau 4. Commandes de création de systèmes de fichiers

Système de fichiers	Commande de création
ext2	mke2fs ou mkfs.ext2
ext3	mke2fs -j ou mkfs.ext3
reiserfs	Mkreiserfs
Xfs	mkfs.xfs
vfat	mkfs.vfat

Le système de fichiers ext3 est une simple extension du format standard ext2 de Linux : il intègre un **journal** qui enregistre toutes les opérations effectuées sur le disque. Ceci permet une récupération plus rapide et sûre du système en cas d'arrêt brutal de la machine.

L'instruction générale de création d'un système de fichiers est :

```
mkfs -t type de fichier partition
```

Il existe des commandes équivalentes pour chaque type de systèmes de fichiers, par exemple `mkfs.ext3`, `mkfs.vfat` (voir *tableau 4*).

Exemples de formatage de la partition hda1 avec création d'un système de fichiers de type ext3 (les trois commandes sont équivalentes) :

```
mkfs.ext3 /dev/hda1
mkfs -t ext3 /dev/hda1
mke2fs -j /dev/hda1 # création du journal spécifique
```

Il est aussi très facile de transformer une partition ext2 en ext3 avec l'instruction `tune2fs` pour créer le journal :

```
tune2fs -j /dev/hda1
```

D. Contrôle de l'intégrité du système de fichiers et réparation

Objectifs ⇒ Maîtriser la remise en état d'un système de fichiers endommagé.

Mots clés fsck, e2fsck, debugfs, dumpe2fs

Si le système de fichiers est endommagé ou corrompu, l'utilitaire `fsck` est utilisé pour vérifier et corriger le système.

L'instruction générale de vérification du système de fichiers est :

```
fsck -t type-de-fichier partition
```

De même que précédemment, il existe des commandes équivalentes pour chaque type de systèmes de fichier, par exemple `fsck.ext3`.

L'option `-i` permet de laisser l'utilitaire `fsck` essayer de corriger lui-même les problèmes rencontrés. Un expert pourrait mieux faire, mais en général l'opération se déroule bien.

L'exemple qui suit permet de vérifier l'intégrité d'un système de fichiers reiserfs :

```
fsck.reiserfs /dev/hda1
```

Une vérification de toutes les partitions est faite au démarrage du système par la commande `fsck`.

La commande `e2fsck` est équivalente à `fsck -t ext2`.

En cas de problèmes plus important, il est possible d'utiliser l'utilitaire `debugfs`. Il est utilisé pour examiner et modifier l'état d'un système de fichiers formaté en ext2. Il permet par exemple de retrouver des inodes de fichiers supprimés (récemment) et de les restaurer.

L'utilitaire `dumpe2fs` permet d'afficher les informations d'un système de fichiers formaté en ext2. Il est par exemple ainsi possible de connaître la date du dernier montage d'un système de fichiers.

E. Montage et démontage d'un système de fichiers

Objectifs ⇒ Savoir monter et démonter un système de fichiers sous Linux.

Mots clés mount, umount, du, df, /etc/fstab

a) Montage et démontage manuel

Pour pouvoir utiliser un système de fichiers, celui-ci doit être monté sur un point de montage de l'arborescence Linux : son contenu est alors accessible comme un simple répertoire.

Le système d'exploitation réalise alors diverses tâches de vérification afin de s'assurer que tout fonctionne correctement.

La commande mount accepte deux arguments :

- le premier est le fichier spécial correspondant à la partition contenant le système de fichiers ;
- le second est le répertoire sous lequel il sera monté (point de montage).

Il peut être nécessaire de spécifier le type de fichiers avec l'option -t au cas où Linux ne parviendrait pas à le déterminer automatiquement.

La commande umount permet le démontage du système de fichiers.

Voici un exemple de montage et de démontage d'une clé USB de type « flashdisk » décrite par le fichier device sda :

```
mount /dev/sda1 /mnt/flashdisk
umount /mnt/flashdisk
```

b) Montage et démontage automatique

Le fichier /etc/fstab est utilisé pour le montage automatique des systèmes de fichiers au moment du démarrage du système.

Chaque ligne du fichier fstab décrit la manière de montage d'un système de fichiers, et ceci à travers six champs séparés par des espaces.

```
#cat /etc/fstab
LABEL=/     /           ext3    defaults        0 1
/dev/hda5   /home       ext3    defaults        0 2
none        /proc       proc    defaults        0 0
/dev/cdrom  /media/cdrom iso9660 ro,noauto,owner 0 0
/dev/hda3   /usr        ext3    defaults        0 2
/dev/hda6   /var        ext3    defaults        0 2
/dev/hda2   swap        swap    defaults        0 0
```

- Le premier champ donne le **nom de périphérique** ou l'**étiquette** (LABEL) associé à ce périphérique. Ce premier champ peut aussi inclure des systèmes de fichiers distants, dans ce cas la notation d'un chemin NFS (*Network File System*) est utilisée : serveur:/chemin_distant. Ceci indique que /chemin_distant est un répertoire partagé via NFS sur une machine distante dont le nom est « serveur » ;
- Le second champ indique le **point de montage**, qui est le chemin d'accès dans l'arborescence Linux ;
- Le troisième champ décrit le **type de système de fichiers**, par exemple ext2, ext3, reiserfs, iso9660, etc. ;
- Le quatrième champ indique les **options de montage**. Il s'agit d'une liste d'options séparées par des virgules, ce sont en fait les options de

la commande mount. Le mot clé « default » indique la combinaison des options rw, suid, dev, exec, auto, nouser, et async (voir le manuel de la commande mount pour plus de détails sur chacune de ces options) ;

- Le cinquième champ est destiné théoriquement à être utilisé par l'utilitaire dump pour déterminer les **systèmes de fichiers à sauvegarder**. Mais en pratique ce champ n'est pas exploité et il est presque toujours à 0 ;
- Le sixième champ est utilisé par l'utilitaire fsck pour déterminer l'**ordre de vérification de l'intégrité des systèmes de fichiers** lors du démarrage du système. Le système de fichier racine doit avoir la valeur 1, les autres systèmes de fichiers ont la valeur 2 et seront vérifiés à la suite. Si ce champ vaut 0, fsck ne vérifie pas le système de fichier.

Les commandes mount et umount utilisent le fichier fstab. Il est important que les données de ce fichier soient complètes et exactes. Par exemple on peut monter un système de fichier en spécifiant seulement le point de montage ou le nom de périphérique. Ainsi, au lieu d'exécuter la commande :

```
#mount -t iso9660 -o ro,noauto,owner, /dev/cdrom /media/cdrom
```

on peut écrire

```
#mount /media/cdrom
```

et les autres informations sont extraites automatiquement par la commande mount à partir de fichier fstab.

La commande mount -a monte tous les systèmes de fichiers répertoriés dans le fichier fstab. Cette commande est généralement exécutée au moment de démarrage du système.

c) Contrôle de système de fichiers

La commande df permet de connaître le taux d'utilisation de toutes les partitions montées du système. L'option -h (*human readable*) facilite la lecture en utilisant des unités de taille plus commodes (Mo, Go, To …).

La commande du (*disk usage*) est très pratique pour connaître l'espace occupé par une arborescence. L'option -s permet d'afficher le total pour chaque élément et l'option -k de l'afficher en kilo-octets :

```
du -ks /usr/local
```

F. Les droits sur les fichiers et les répertoires

Objectifs ⇒ Comprendre la gestion des droits sur les fichiers et répertoires sous Linux.
⇒ Savoir protéger les fichiers et les répertoires.

Mots clés chmod, umask, chattr, suid, sgid, sticky bit, chattr

Linux permet de spécifier les **droits** dont disposent les utilisateurs sur un fichier ou un répertoire par la commande `chmod`.

On distingue trois catégories d'utilisateurs :

- `u` : le propriétaire (*user*) ;
- `g` : le groupe ;
- `o` : les autres (*others*).

Ainsi que trois types de droits :

- `r` : lecture (*read*) ;
- `w` : écriture (*write*) ;
- `x` : exécution ;
- `-` : aucun droit.

Exemples :

- pour donner le droit de lire le fichier liste.txt au propriétaire du fichier :

```
chmod u+r liste.txt
```

- pour autoriser une personne du groupe propriétaire du fichier à modifier le fichier :

```
chmod g+w liste.txt
```

- pour autoriser les autres utilisateurs à exécuter le fichier commande.sh :

```
chmod o+x commande.sh
```

Les droits d'accès peuvent aussi s'exprimer en notation octale. Les correspondances sont indiquées dans le *tableau 5*.

On peut utiliser la **notation octale** pour les droits avec la commande `chmod`, cela permet de positionner plusieurs types de droits en une seule commande.*Exemples* :

- attribuer les droits `rw-------` à tous les fichiers :

```
chmod 600 *
```


- attribuer les droits `rw-r--r--` à tous les fichiers :

```
chmod 644 *
```

- attribuer les droits `rwxr-x---` à tous les fichiers :

```
chmod 750 *
```

Tableau 5. Expression des droits sur les fichiers en notation octale

Droit	Notation octale
---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwx	7

Lorsqu'on crée un nouveau fichier, par exemple avec la commande `touch`, ce fichier possède certains droits par défaut. La commande `umask` permet de changer ces droits par défaut. Les droits sont indiqués de façon « inverse » en notation octale pour les droits de type `r` et `w` seulement.

Pour créer des fichiers en mode `rw-r--r--` :

```
umask 022
```

Pour créer des fichiers en mode `-------` :

```
umask 666
```

Les droits spéciaux

Il existe trois droits spéciaux, **SUID**, **SGID** et **Sticky Bit**. Ils peuvent être positionnés par la commande `chmod` (premier groupe de droits exprimés en octal) :

- **SUID** (*Set User ID*) : lorsque le bit SUID est positionné, une commande se lancera avec l'UID de son propriétaire, ce qui permet d'acquérir ses droits durant l'exécution de la commande : par exemple la commande `/usr/bin/passwd` permet d'acquérir les droits de *root* pour modifier le fichier `/etc/shadow`.

  Le bit SUID est positionné par l'option `s` de la commande `chmod`, ou bien en positionnant à « 1 » le premier bit du groupe des droits spéciaux (d'où la valeur 4 de l'exemple suivant) :

```
chmod 4755 /bin/cat
chmod u+s /bin/grep
```

- **SGID** (*Set Group Id*) : ce droit fonctionne de la même façon que le droit SUID en ce qui concerne les exécutables mais en donnant le droit du groupe.

  Le SGID peut aussi être attribué à un répertoire : dans ce cas tout fichier créé dans un répertoire portant le SGID aura comme groupe propriétaire le groupe du répertoire.

  Ce bit est positionné par l'option `s` de la commande `chmod`, ou bien en positionnant à « 1 » le deuxième bit du groupe des droits spéciaux (d'où la valeur « 2 » de l'exemple suivant) :

```
chmod 2755 /home/lpi102
chmod g+s /home/lpi102
```

- **Sticky Bit** : si le Sticky Bit est positionné sur un répertoire, seul le propriétaire d'un fichier contenu dans ce répertoire pourra le renommer ou le supprimer, mais tous les utilisateurs pourront y avoir accès. Le Sticky Bit est par exemple toujours positionné sur les répertoires `/tmp` ou `/var/mail/`.

  Ce bit est positionné par l'option `t` de la commande `chmod`, ou bien en positionnant à 1 le troisième bit du groupe des droits spéciaux (d'où la valeur 1 de l'exemple) :

```
chmod 1666 /home/lpi/partagé
chmod o+t /home/lpi/partagé
```

  Historiquement, le Sticky Bit positionné sur un fichier indiquait au système de le maintenir en mémoire à la suite d'un premier chargement pour des questions d'efficacité.

G. Modifier le propriétaire et le groupe sur les fichiers et les répertoires

Objectifs	⇒ Être capable de modifier le propriétaire et le groupe d'un fichier ou d'un répertoire.
Mots clés	chown, chgrp

Linux permet de spécifier le propriétaire d'un fichier ou d'un répertoire par la commande `chown`.

```
chown le_propriétaire /home/lpi/monfichier
```


Linux permet de spécifier le groupe d'un fichier ou d'un répertoire par la commande `chgrp`.

```
chgrp le_groupe /home/lpi/monfichier
```

H. Les quotas

Objectifs	⇒ Comprendre la notion de quota sur un système de fichiers. ⇒ Savoir gérer les quotas par utilisateur.
Mots clés	quota, /etc/fstab, usrquota, quotacheck, edquota, quotaon, quotaoff, repquota

L'attribution de quotas dans un système de fichiers est un outil qui permet de maîtriser l'utilisation de l'espace disque par les utilisateurs.

Les quotas consistent à fixer une **limite d'espace** pour un utilisateur ou un groupe d'utilisateurs (attention, cela ne fonctionne pas forcément avec les systèmes de fichiers de type `reiserfs`).

Pour cela, voici les étapes à suivre :

- éditer le fichier `/etc/fstab` et rajouter `usrquota` dans les options de montage ;
- remonter la partition sur laquelle on veut définir des quotas pour que le montage prenne en compte les nouvelles options :

```
mount -o remount device
```

- commencer la vérification des quotas avec la commande `quotacheck` :

```
quotacheck -ca
```

- éditer les quotas pour chaque utilisateur avec `edquota` :

```
edquota -u utilisateur
```

  La commande ouvre un éditeur (`vi` ou `emacs` selon le contenu de la variable EDITOR), qui permet de modifier directement les fichiers `aquota.user` ou `aquota.group` ;

- débuter la prise en compte des quotas par la commande `quotaon` ;

```
quotaon partition
```

- afficher un résumé des informations sur les quotas définis sur un système de fichiers et sur leur utilisation de ces quotas par les différents utilisateurs avec la commande `repquota`.

```
repquota partition
```

La commande `quotaoff` arrête la prise en compte des quotas.

I. Recherche de fichiers

Objectifs ⇒ Connaître et maîtriser les différents types d'outils de recherche de fichiers.
⇒ Savoir gérer les bases de données d'informations associées à ces outils.

Mots clés find, locate, slocate, whereis, which, whatis, updatedb, makewhatis, apropos, /etc/updatedb.conf

La recherche dans l'arborescence d'un système de fichiers peut se faire grâce à des utilitaires tel que `find`, `locate`, `which`, `whereis`, `whatis` et `apropos`.

a) find

La commande `find` est la plus ancienne commande de recherche de Unix, elle n'utilise pas de base indexée et son exécution peut donc parfois être longue car elle est très complète par ses critères de recherche.

La syntaxe générale est la suivante : `find <chemin> <critères>`.

Les principales options de recherche sont les suivantes :

- `-name` : par nom de fichiers ;
- `-type` : par type du fichier (`f` : fichier, `d` : répertoire ...) ;
- `-user` : utilistateur auquel appartiennent les fichiers recherchés ;
- `-atime` : par date de dernier accès aux fichiers ;
- `-mtime` : par date de dernière modification du contenu des fichiers ;
- `-ctime` : par date de dernier changement des fichiers : contenu mais aussi droits d'accès, propriétaire....

Pour rechercher le fichier Xinitrc dans tout le système (à partir de la racine) :

```
find / -name Xinitrc
```

Pour rechercher les fichiers de l'utilisateur 666 dans tout le système (à partir de la racine) :

```
find / -user 666
```


b) locate et slocate

La commande `locate` cherche tous les types de fichiers dans l'intégralité des systèmes de fichiers comme `find`, mais elle utilise une base de données. La base de données est automatiquement mise à jour par une commande de type `cron`, généralement la nuit, lorsque la machine est peu sollicitée.

On peut mettre à jour manuellement la base de données en utilisant la commande `updatedb` (on doit être *root* pour lancer cette commande). Les options de fonctionnement de la commande `updatedb` sont décrites dans le fichier /etc/updatedb.conf. On peut y décrire la racine de l'arborescence à indexer, les fichiers à exclure, l'emplacement de la base de données, etc.

```
updatedb
```

La recherche est donc très rapide et peut se faire à partir de fragments du nom :

```
locate monfichier_perdu
  /home/nicolas/trucs/monfichier_perdu
```

La commande `slocate` est la version sécurisée de `locate`. Elle fonctionne de la même manière, mais stocke également les droits d'accès associés aux fichiers ainsi que les informations de propriété (propriétaire et groupe) du fichier de façon à ne pas afficher dans le résultat de la recherche les fichiers auxquels l'utilisateur n'aurait pas accès.

c) which

La commande `which` est utilisée pour trouver l'emplacement d'une commande : elle effectue sa recherche par rapport au contenu de la variable `PATH`, et retourne le chemin du premier fichier correspondant.

```
which bash
  /bin/bash
```

Elle est très commode pour vérifier quelle version de la commande s'exécute réellement lorsqu'on l'appelle par son nom relatif.

d) whereis

La commande `whereis` fonctionne de façon similaire à `which`, mais elle peut aussi chercher dans les pages de manuel (`man`) et les codes sources.

```
bash: /bin/bash /usr/share/man/man1/bash.1.bz2
```

e) whatis

La commande `whatis` cherche des commandes dans l'intégralité des systèmes de fichiers comme `which`, mais elle utilise une base de données qui contient une courte description ainsi que des mots clés.

La base de données est créée en utilisant la commande `makewhatis` (on doit être *root* pour lancer cette commande).

```
makewhatis
```

La recherche est donc plus rapide et peut se faire à partir du nom ou d'un mot clé. La réponse contient une description rapide de la commande

```
whatis who

  who - show who is logged on
```

f) apropos

La commande `apropos` utilise la même base de données que `whatis`, mais donne plus d'informations :

```
apropos who

  w - show who is logged on and what they are doing
  who - show who is logged on
  whoami - print effective userid
```

J. Exercices

1. **Comment est représenté le périphérique IDE esclave du deuxième contrôleur ?**
   - ❑ hda
   - ❑ sdc.
   - ❑ hdc1.
   - ❑ hdd.
2. **L'arborescence _____ contient les données des utilisateurs.**
3. **Pour formater une partition, vous utilisez**
   - ❑ `disk druid`.

   - ❑ `fdisk`.
   - ❑ `mkfs`.
   - ❑ `fsck`.
4. **Quelles commandes permettent de tester un système de fichiers formaté en ext3 (plusieurs réponses) ?**
   - ❑ `fsck.ext3`.
   - ❑ `fsck -t ext3`.
   - ❑ `tune2fs -j`.
   - ❑ `e3fsck`.
5. **Quelle commande montre l'occupation d'un répertoire /usr/XXX ?**
   - ❑ `du -s /usr/XXX`.
   - ❑ `df -h /usr/XXX`.
   - ❑ `ls -iR /usr/XXX`.
6. **Quels droits sont positionnés sur un répertoire par la commande chmod 1777 un_repertoire (plusieurs réponses) ?**
   - ❑ Elle permet à tous les utilisateurs d'accéder au répertoire.
   - ❑ Elle interdit l'accès aux autres utilisateurs que le groupe.
   - ❑ Elle permet à tous de créer des fichiers dans ce répertoire, mais pas de modifier ou d'effacer ceux des autres.
7. **La commande _____ permet de vérifier les quotas d'un utilisateur.**
8. **La commande locate**
   - ❑ fonctionne comme `find` mais avec des options différentes.
   - ❑ est plus efficace que `find` car elle utilise un index.
   - ❑ permet de changer la langue par défaut de Linux.

Chapitre 5. GNU et les commandes Unix

A. Commandes générales

Objectifs	⇒ Maîtriser la ligne de commande. ⇒ Connaître les caractères spéciaux et la majorité des filtres de texte disponibles sur la ligne de commande.
Points importants	Les commandes Unix ont une syntaxe générale commune. La maîtrise de cette syntaxe permet de détecter très vite les éventuelles anomalies ou erreurs sur une commande.
Mots clés	~/.bash_history, /etc/shells, bash, csh, ksh, man, pwd, sh, tcsh

Les commandes Unix peuvent être exécutées depuis la ligne de commande.

Les instructions entrées sur la ligne de commande sont exécutées par un « interpréteur de commande » communément appelé « **shell** ».

Un premier shell est lancé par le programme `login` après l'authentification sur le système (après avoir saisi à l'invite `login` un nom d'utilisateur et un mot de passe associé valides).

L'interpréteur de commande possède un « **prompt** » qui peut prendre plusieurs formes selon les configurations du compte de l'utilisateur connecté. De façon générale, le prompt est de la forme :

`utilisateur@nom-de-poste repertoire-courant$`.

Par défaut, ce prompt se termine par le caractère **$** pour un **utilisateur normal** et par un **#** pour le **super-utilisateur**.

Les commandes sont à entrer à la suite de ce *prompt* qui indique qu'un interpréteur de commande est prêt à les exécuter.

Les commandes Unix sont de la forme générale :

`nom-de-commande [ option(s) ] [ argument(s) ].`

Le nom de la commande est toujours précisé.

Une commande peut n'avoir ni option ni argument :

```
ls
```

Une commande peut être suivie par une ou plusieurs options :

```
ls -a -l
```

Une commande peut posséder un ou plusieurs arguments :

```
ls /dev/
```

Une commande peut être suivie d'une combinaison d'**options** et d'**arguments** :

```
ls -al /dev/
```

- une **option** est souvent représentée par une lettre précédée du caractère « - ». Plusieurs options peuvent être séparées par des espaces mais peuvent être aussi rassemblées pour former un mot précédé par le caractère « - » :

```
ls -l
ls -a -l
ls -al
```

- un **argument** représente souvent le chemin dans l'arborescence du système de fichiers. Ce chemin peut être **relatif** ou **absolu** :
  - le chemin **absolu** représente l'arborescence complète à partir de la racine « / ». Un chemin absolu commence toujours par « / », par exemple /var/log/messages ou /home/maitre/travail ;
  - le chemin **relatif** représente l'arborescence depuis le répertoire courant. Un chemin relatif ne commence jamais par « / », par exemple ../var/log/messages depuis /home et travail depuis /home/maitre.

a) Les interpréteurs de commandes

De nombreux interpréteurs de commandes sont actuellement disponibles. Voici quelques interpréteurs courants :

- `/bin/sh` : le « *bourne shell* » ;
- `/bin/bash` : le « *bourne again shell* » ;

- `/bin/ksh` : le « *korn shell* » ;
- `/bin/csh` : le « *C shell* » ;
- `/bin/tcsh` : le « *Tom's C shell* ».

Les interpréteurs disponibles sur un système donné sont listés dans le fichier /etc/shells.

b) Les commandes relatives aux répertoires et aux fichiers

- `pwd` : pour afficher le chemin absolu du répertoire courant ;
- `cd` : pour changer de répertoire, syntaxe :

```
cd chemin
```

  `cd` (sans option ni argument) permet de se déplacer vers le répertoire personnel de l'utilisateur courant (*home directory*) ;

- `ls` : pour lister le contenu d'un répertoire, syntaxe :

```
ls chemin
```

- `ls` (sans option ni argument) affiche le contenu du répertoire courant,
- l'option `-a` affiche en plus les fichiers cachés dont les noms commencent par un point,
- l'option `-l` permet un affichage long (type de fichier, droit, propriété, date de modification, taille du fichier, etc.),
- l'option `-i` affiche le numéro d'inode auquel est rattaché le fichier ;
- `mkdir` : pour créer un nouveau répertoire, syntaxe :

```
mkdir chemin
```

- `rmdir` : pour supprimer un répertoire vide, syntaxe :

```
rmdir chemin
```

- `touch` : pour changer les informations de date et d'heure d'un fichier. Crée un fichier vide lorsque le fichier passé en argument n'existe pas ;
- `cp` : pour copier le contenu d'un fichier, syntaxe :

```
cp chemin-fichier-source chemin-fichier-destination
```

- `mv` : pour déplacer ou renommer un fichier, syntaxe :

```
mv ancien chemin nouveau chemin
```


- `rm` : pour supprimer un fichier, syntaxe :

```
rm chemin
```

- l'option `-r` permet de supprimer un répertoire,
- l'option `-f` permet de forcer la suppression sans demande de confirmation de la part de l'utilisateur ;
- `ln` : pour effectuer un lien sur un fichier. Un lien permet de faire référence à un fichier donné par plusieurs noms différents. Il existe deux types de liens sous Unix :
- les liens **matériels** (*hardlink*) qui créent des noms différents pour désigner un même espace sur le disque. Le fichier sera effectivement supprimé lorsque le dernier lien sera détruit. Syntaxe :

```
ln chemin fichier source chemin fichier destination
```

- les liens **symboliques** qui sont eux-mêmes de petits fichiers qui contiennent un chemin d'accès vers un autre fichier : ils peuvent donc pointer vers un fichier qui n'existe pas. La différence essentielle est qu'un lien symbolique peut pointer vers un fichier appartenant à un autre système de fichiers. Syntaxe :

```
ln -s chemin fichier source chemin fichier-destination
```

- le caractère « ~ » représente le répertoire personnel de l'utilisateur courant :

```
cd ~
cd ~/travail
```

c) Les pages de manuel

Pour chaque commande, une page de manuel explique à quoi elle sert, comment elle fonctionne, et les différentes options disponibles.

La commande `man` affiche ces pages de manuel, syntaxe :

```
man commande
man ls
```

La commande `history` affiche les commandes précédemment lancées par l'utilisateur courant :

```
history
```

Elle utilise le contenu du fichier .bash_history qui est mis à jour après chaque commande. Il se trouve dans le répertoire personnel de l'utilisateur (~/.bash_history).

Le résultat de cette commande est la liste des précédentes commandes précédées d'un numéro. Chaque commande peut être ré-exécutée en tapant `!N`, où `N` est le numéro de ligne dans le résultat de la commande `history`. Pour ré-exécuter la dernière commande il suffit de taper `!!`

B. Les filtres

Objectifs	⇒ Comprendre le fonctionnement des filtres : entrée, sorties, arguments. ⇒ Connaître les commandes de filtre courantes. ⇒ Retenir les arguments classiques.
Points importants	Les filtres Unix font partie intégrante de son fonctionnement. C'est grâce à eux que l'on peut fabriquer des commandes très élaborées en personnalisant leur résultat.
Mots clés	cat, cut, expand, fmt, head, join, nl, od, paste, pr, sed, sort, split, tac, tail, tr, unexpand, uniq, wc, grep, regexp

Les **filtres** sont des commandes qui, à partir d'un flux d'entrées donné, effectuent des traitements avant d'afficher un résultat en sortie. On les nomme également **commandes de traitement de flux**.

La commande `cat` affiche seulement le contenu du fichier /etc/passwd sans altérer le fichier original :

```
cat /etc/passwd
```

La commande `nl` affiche en sortie les lignes du fichier /etc/passwd en les numérotant sans altérer le fichier original :

```
nl /etc/passwd
```

Voici une liste des filtres les plus courants :

- `cat` : pour concaténer le contenu d'un fichier et l'afficher ensuite sur la sortie standard, qui est par défaut l'écran. La commande `cat` sans argument (nom de fichier) attend les suites de caractères tapés sur l'entrée standard (le clavier) :

```
cat /etc/passwd
```

- `cut` : pour n'afficher que certaines colonnes (champs) d'un fichier donné. Cette commande utilise alors comme option le séparateur de champs et le numéro du champ à afficher pour chaque ligne. S'il n'est pas spécifié, le séparateur de champs par défaut est le caractère de tabulation.

  La commande ci-dessous permet d'afficher les noms d'utilisateurs du système. On utilise comme séparateur de colonnes le caractère « : » et on sélectionne la première colonne du fichier /etc/password.

```
cut -d : -f1 /etc/passwd
```

- `expand` : pour convertir les tabulations d'un fichier en espaces. Le processus inverse, conversion des espaces en tabulations, peut être effectué avec la commande `unexpand ;`
- `fmt` : pour formater les paragraphes dans un fichier ;
- `grep` : pour afficher les lignes contenant une occurrence de caractères donnée, syntaxe :

```
grep CHAINE chemin
```

  Pour afficher toutes les lignes contenant la chaîne de caractères « false » :

```
grep false /etc/passwd
```

  `CHAINE` peut être une expression régulière. Pour afficher toutes les lignes qui commencent par la chaîne « root » :

```
grep ^root /etc/passwd
```

  L'option `-v` permet de n'afficher que les lignes NE contenant PAS l'expression régulière mentionnée. Pour afficher les lignes ne commençant pas par la chaîne « root » :

```
grep -v ^root /etc/passwd
```

- `head` : pour afficher les premières lignes d'un fichier, syntaxe :

```
head -N chemin
```

  Sans option, `head` affiche les 10 premières lignes d'un fichier. Pour afficher les quatre premières lignes du fichier `/etc/passwd` :

```
head -4 /etc/passwd
```

- `join` : pour effectuer des jointures des lignes de deux fichiers différents dans un même champ ;

- `nl` : affiche en sortie les lignes d'un fichier précédées de leurs numéros respectifs ;
- `od` : formate un fichier en octal et en autres formats ;
- `paste` : pour fusionner deux fichiers différents en prenant chaque ligne de chaque fichier pour former une nouvelle ligne ;
- `pr` : pour formater de manière simple un fichier et le préparer pour une impression, syntaxe :

```
pr [option(s)] chemin
```

- option `-N` : affiche le texte en plusieurs colonnes. `N` définit le nombre de lignes,
- option `-wN` : le `N` derrière `-w` (*width*) indique le nombre de caractères par ligne, par défaut 72,
- option `-lN` : le `N` derrière `-l` (*length*) définit le nombre de lignes par page (par défaut 66),
- option `-hTexte` : `Texte` doit remplacer le nom du fichier dans l'en-tête de chaque page,
- option `-oN` : le N derrière -o (*offset*) définit le nombre de caractères de retrait à gauche,
- option `-n` : numérote les lignes.
- `sed` : utilitaire de traitement de données très puissant, capable d'utiliser les expressions régulières.

  Pour substituer toute « `chaîne1` » dans le fichier « `chemin1` » avec « `chaine2` » et envoyer le résultat dans le fichier « `chemin2` » :

```
sed 's/chaine1/chaine2/g' chemin1 > chemin2
```

- `regexp` : permet de tester une expression régulière en lui fournissant une chaîne de test. Cela permet de vérifier les expressions régulières employées avec les commandes classiques `ls`, `sed`, `awk`, etc.
- `sort` : trie le contenu d'un fichier par ligne :
- option `-n` pour effectuer un tri numérique,
- option `-r` pour effectuer un tri décroissant ;
- `split` : pour découper un fichier en plusieurs. On peut spécifier une taille de fichiers en option. Par exemple, la commande pour créer les fichiers petitfichieraa, petitfichierab… d'une taille maximum de 1,4 Mo (taille d'une disquette) est :

```
split -b 1.4m /home/maitre/grosfichier petitfichier
```


- `tac` : inverse de la commande `cat`, affiche le contenu du fichier dans l'ordre inverse des lignes ;
- `tail` : pour afficher les dernières lignes d'un fichier, syntaxe :

```
tail -N chemin
```

  Sans option, `tail` affiche les 10 dernières lignes d'un fichier.

  Pour afficher les 4 dernières lignes du fichier /etc/passwd :

```
tail -4 /etc/passwd
```

- `tr` : pour effectuer des conversions de caractères, par exemple minuscule/majuscule, retour chariot/passage à la ligne ...) ;
- `uniq` : affiche les lignes sans doublons ;
- `wc` : affichent des statistiques sur un fichier, nombre de caractères, nombre de mots et nombre de lignes.
- `-b` : affiche seulement le nombre de caractères ;
- `-w` : affiche seulement le nombre de mots ;
- `-l` : affiche seulement le nombre de lignes.

C. Utilisation de l'éditeur « vi »

Objectifs	⇒ Être capable d'effectuer des manipulations simples du contenu d'un fichier avec l'éditeur « vi ».
Points importants	L'éditeur « vi » est présent depuis les toutes premières versions d'Unix. Malgré une ergonomie parfois étrange, il reste très utilisé par les administrateurs, essentiellement pour des raisons pratiques : par exemple, il fonctionne avec n'importe quel terminal.
Mots clés	Vi

L'éditeur **vi** se trouve systématiquement sur toutes les versions de Linux et est surtout utile lorsque l'on intervient sur une machine à distance. Il fonctionne en deux modes différents, le mode édition et le mode commande. On bascule de l'un à l'autre par la commande `esc`.

Pour ouvrir un fichier avec l'éditeur vi :

```
vi mon_fichier
```

Pour se déplacer dans le texte :

- `l` : vers la droite ;

- `h` : vers la gauche ;
- `j` : vers le haut ;
- `k` : vers le bas ;
- `L` : se déplace sur la dernière ligne de la page courante ;
- `H` : se déplace sur la première ligne de la page courante ;
- `nG` : se déplace sur la ligne « n » du fichier courant.

Pour ajouter ou supprimer du texte :

- `i` : ajoute le texte à partir de la position du curseur ;
- `A` : ajoute le texte à partir de la fin de la ligne ;
- `O` : crée une nouvelle ligne ;
- `R` : remplace le texte ;
- `r` : remplace le caractère courant ;
- `dd` : supprime la ligne courante, 4dd supprime 4 lignes à partir de la ligne courante ;
- `x` : supprime le caractère courant, 5x supprime 5 caractères.

Pour rechercher du texte : `/texte_cherché`

Pour substituer du texte :

- `:ligne_début,ligne_fin/chaine1/chaine2/g` : substitue la chaine2 à la chaine1 entre les lignes `ligne_début` et `ligne_fin` sans demande de confirmation ;
- `:ligne_début,ligne_fin/chaine1/chaine2/c` : substitue la chaine2 à la chaine1 entre les lignes `ligne_début` et `ligne_fin` avec demande de confirmation.

Pour annuler la dernière commande : `u`.

Pour sauvegarder le fichier :

- `:wq (ou ZZ)` : sauvegarde le fichier et quitte ;
- `:w son_fichier` : sauvegarde dans le fichier son_fichier ;
- `:q!` : quitte sans sauvegarde ;
- `:w!` : force la sauvegarde lorsque le fichier est en lecture seule ;
- `:e nouveau_fichier` : sauvegarde le fichier courant et édite le fichier nouveau_fichier ;
- `:e!nouveau_fichier` : édite le fichier nouveau_fichier sans sauvegarder le fichier courant.

D. Tubes et les redirections

Objectifs	⇒ Comprendre le fonctionnement des tubes. ⇒ Comprendre le fonctionnement des redirections. ⇒ Connaître les commandes `tee` et `xargs`.
Points importants	Les tubes Unix permettent de combiner des commandes en les utilisant comme des « briques » indépendamment de leur provenance (*shell script*, programme …). C'est l'une des grandes forces d'Unix.
Mots clés	tee, xargs, |, >, >>, <, <<, echo, exec

a) Redirection

Linux fonctionne avec trois types de flux de données :

- l'entrée standard identifiée par le descripteur 0, par exemple le clavier ;
- la sortie standard identifiée par le descripteur 1, par exemple l'écran ou l'interpréteur de commande ;
- la sortie d'erreur standard identifiée par le descripteur 2, par exemple l'écran :

```
maitre@maestro maitre$echo 'ceci est un message de test'
ceci est un message de test
maitre@maestro maitre$
```

La commande `echo` permet d'effectuer un simple affichage sur la sortie standard.

Le mécanisme de redirection permet de changer la sortie, l'entrée ou la sortie d'erreur d'une commande donnée. Le caractère « > » est utilisé pour changer la sortie standard.

Syntaxe : `commande > chemin`.

```
maitre@maestro maitre$echo 'message test' > /home/maitre/fichier
maitre@maestro maitre$cat /home/maitre/fichier
message test
maitre@maestro maitre$
```

Si le fichier de redirection n'existe pas encore, « > » permet de le créer. S'il existe, son contenu sera écrasé par la sortie de la dernière commande. Les caractères « >> » permettent soit de créer un fichier inexistant soit de rajouter la sortie d'une commande au contenu d'un fichier existant (sans écrasement).

La redirection fonctionne dans les deux sens, le caractère « < » permet de spécifier une autre entrée que l'entrée standard, syntaxe :

```
commande < chemin
```

Les caractères « << » permettent de lire le fichier en entrée jusqu'à ce que la commande rencontre une certaine chaîne de caractères.

Dans l'exemple qui suit, les caractères saisis sur l'entrée standard seront pris en compte jusqu'à ce que la commande `cat` rencontre la chaîne « `FIN` ».

```
cat << FIN
```

Enfin, pour rediriger la sortie d'erreur vers un fichier, on utilise le descripteur 2 de la sortie standard :

```
cat /etc/passwd 2> fichier_erreur
```

Les caractères « >& » permettent de rediriger la sortie erreur et la sortie standard vers un fichier :

```
tail /etc/passwd > fichier_sortie 2>&1
```

ou

```
tail /etc/passwd >& fichier_sortie
```

Cette commande copie les 10 dernières lignes du fichier /etc/passwd dans le fichier fichier_sortie et y redirige également les éventuels messages d'erreur.

b) Les tubes

Le mécanisme de tube (*pipe*) permet de faire en sorte que la sortie d'une commande devienne l'entrée d'une autre. Les tubes utilisent le caractère « | », syntaxe : `commande | commande`.

```
sort /etc/passwd | head -6
```

Cette commande affiche les 6 premières lignes du fichier /etc/passwd une fois ce fichier trié par ordre alphabétique croissant.

Tubes et redirections peuvent êtres enchaînés indéfiniment sur une ligne de commande selon les résultats que l'on veut obtenir.

c) La commande tee

La commande `tee` duplique le flux de données en sortie : elle copie la sortie dans un fichier (simple redirection) et, en même temps, affiche le résultat sur la sortie standard, et permet donc de le renvoyer à une autre commande.

La commande de l'exemple ci-dessous affiche à l'écran les 6 premières lignes du fichier /etc/passwd et, en même temps, les redirige dans le fichier le_fichier.

```
ls -l /etc | tee le_fichier | wc -l
```

d) La commande xargs

La commande `xargs` permet de passer en argument d'une commande les flux reçus en entrée.

La commande de l'exemple ci-dessous prend la sortie de la commande `cat le_fichier` comme argument de la commande `ls`.

```
cat le_fichier | xargs ls
```

e) La commande exec

La commande `exec` redirige dans un fichier l'entrée et la sortie standard (`stdin` et `stdout`). Elle est généralement utilisée dans un script shell et permet par exemple de rediriger les sorties des commandes dans des fichiers différents au cours de l'exécution d'un script :

```
exec < le_fichier
```

E. Les caractères spéciaux

Objectifs	⇒ Connaître les caractères spéciaux (= qui ont une signification particulière) pour les différents « shell ». ⇒ Comprendre la signification des caractères spéciaux.
Points importants	Il est nécessaire de bien identifier ces caractères particuliers pour éviter des effets « étranges » lors de l'utilisation des commandes.
Mots clés	;, &&, ||, &>, $, [,], [[,]], {, }, `, ", ', ., ?, \

Certains caractères ont une signification particulière sur une ligne de commande :

- `;` : permet d'enchaîner plusieurs commandes sur une même ligne. Les commandes séparées par un « ; » sont exécutées séquentiellement, l'une après l'autre ;
- `&` : placé en fin de commande, permet de lancer cette commande en tâche de fond (*background*) ;
- `&&` : placé entre deux commandes, permet d'exécuter la deuxième commande si et seulement si la première s'est exécutée sans erreur :

```
commande_1 && commande_2
```

- `||` : placé entre deux commandes, permet d'exécuter la deuxième commande si et seulement si la première a renvoyé une erreur :

```
commande_1 || commande_2
```

- `&>` : redirige dans un fichier la sortie d'une commande, ainsi que les erreurs éventuelles :

```
commande &> le_fichier
```

- `$` : permet d'accéder au contenu d'une variable ;
- `\` : placé avant un caractère spécial, permet que celui-ci soit interprété comme un simple caractère ;
- " (guillemets) : placé de part et d'autre d'une chaîne de caractères, permet que tous les caractères spéciaux que celle-ci contient soient ignorés, à l'exception de « $ », « \ » et « ` » :

```
Un='a trop de travail'
Deux="Nicolas $Un"
echo $Deux
   -> Nicolas a trop de travail
```

- ' (apostrophe) : placée de part et d'autres d'une chaîne de caractères, permet que tous les caractères spéciaux que celle-ci contient soient ignorés.

```
Un='a très faim'
Deux='Nicolas $Un'
echo $Deux
   -> Nicolas $Un
```

- ` (apostrophe culbutée) placée de part et d'autre d'une commande, force l'exécution de cette commande sur la ligne de commande elle-même :

```
Un=`pwd`
echo $Un
   /home/nicolas/travail/
```

- `[[ ]]` ou `{ }` permettent d'effectuer une opération simple sur la ligne de commande (addition, soustraction, multiplication, division entière).

Les caractères suivants sont utilisés pour remplacer un ou plusieurs caractères (*wildcards*) :

- `*` : pour remplacer aucun, un ou plusieurs caractères ;
- `?` : pour remplacer un et un seul caractère ;
- `[a-z]` : pour remplacer un caractère ayant l'une des valeurs indiquées dans l'intervalle « a-z » ;
- `[^a-z]` : pour remplacer un caractère différent de toutes les valeurs indiquée dans l'intervalle « a-z ».

F. Les variables et les variables d'environnement

Objectifs	⇒ Comprendre les variables d'environnement. ⇒ Connaître les commandes relatives aux variables.
Points importants	Les variables Unix, en particulier celles dites d'environnement, modifient le comportement des commandes. Il est important de les connaître et d'être capable de modifier leur contenu.
Mots clés	env, set, unset, export, $!, $$, $?, /etc/profile, ~/.profile

Les interpréteurs de commande Linux distinguent deux types de variables : les **variables simples** et les **variables d'environnement**. Les variables d'environnement ont la particularité d'être connues de toutes les commandes lancées depuis l'interpréteur de commande courant.

L'affectation de variables se fait comme suit : `variable="valeur"`.

```
variab="abcdef"
```

Il est important de noter qu'avec *bash*, il ne doit pas y avoir d'espace avant et après le signe « = ».

L'accès au contenu s'effectue en ajoutant le caractère `$` au début du nom de la variable ou par l'expression `${nom-de-la-variable}` :

```
maitre@maestro maitre$echo $variab
   abcdef
maitre@maestro maitre$echo ${variab}
   abcdef
maitre@maestro maitre$
```

Nous venons de définir une variable simple. Pour la transformer en variable d'environnement, on utilise la commande `export` :

```
export variab="abcdef"
```

La commande `env` seule, sans option ni argument, permet d'afficher toutes les variables d'environnement définies.

La commande `set` affiche la liste complète des variables définies (variables simples ou variables d'environnement).

La commande `unset` permet de détruire une variable.

```
unset variable
```

Voici certaines variables spéciales de `bash` :

- `$$` : le numéro de processus (PID) de l'interpréteur de commande ;
- `$!` : le numéro de processus (PID) de la dernière commande lancée en tâche de fond (c'est-à-dire avec l'opérateur « & ») ;
- `$?` : la valeur retournée par la dernière commande ;
- `$` : la liste des options avec lesquelles l'interpréteur de commande a été appelé.

Dans le cadre d'une commande lancée à partir d'un fichier script :

- `$#` : désigne le nombre de paramètres accompagnant l'appel du script ;
- `$@` et `$*` : désignent l'ensemble des paramètres ;
- `$1`, ..., `$9`, `${10}`, `${11}`, ... : désignent la valeur de chaque paramètre ;
- `$0` : désigne le nom (le chemin) du script.

Plusieurs variables sont définies au démarrage de l'interpréteur de commandes. La définition de ces variables peut se trouver dans les fichiers lus par l'interpréteur au démarrage comme le fichier /etc/profile pour tous les utilisateurs du système ou dans les fichiers personnels de l'utilisateur ~/.profile, ~/.bash_profile, etc.

G. Les processus

Objectifs	⇒ Comprendre la notion fondamentale de processus sous Unix. ⇒ Savoir gérer les processus.
Points importants	Tout programme lancé sous Unix est associé à un processus. Être capable de les identifier et les gérer est indispensable.

Mots clés	ps, kill, killall, pstree, nice, top, renice, nohup

Ce qui est désigné comme processus est une instance de programme s'exécutant à un instant donné ainsi que son contexte (ou environnement). Ce dernier terme désigne l'ensemble des ressources utilisées par le programme pour pouvoir se dérouler, comme par exemple la mémoire, les fichiers ouverts, les droits associés, la priorité...

Les processus sont identifiés par un numéro unique dans le système à un moment donné, le PID. C'est à l'aide de ce nombre que l'on peut désigner une instance de programme et interagir avec.

Les processus sont également caractérisés par un propriétaire. Il s'agit de l'utilisateur qui en a demandé l'exécution. En général, seul ce propriétaire pourra entreprendre des actions sur le processus.

Les processus sont créés par « clonage », ce qui permet la recopie d'une partie des informations citées ci-dessus. Le premier processus créé est le processus `init` qui est l'**ancêtre** de tous les autres.

La commande pstree affiche les processus sous forme d'arborescence, on peut donc visualiser qui est le processus parent d'un autre. L'option `-p` affiche le PID de chaque processus et l'option `-u` affiche le nom de l'utilisateur ayant lancé le processus.

```
pstree -pu
```

a) Obtenir des informations sur un processus

Pour chaque processus exécuté, le système d'exploitation stocke un certain nombre d'informations :

- numéro unique du processus ou **PID** (*Process Identification*) ;
- numéro du processus parent ou **PPID** (*Parent Process Identification*) ;
- numéro de l'utilisateur, ou **UID** (*User Identification*), ayant lancé le processus ;
- numéro du groupe, ou **GID** (*Group Identification*), ayant lancé le processus ;
- durée de traitement (*temps CPU*) et priorité du processus ;
- référence au répertoire de travail courant du processus ;
- table de référence des fichiers ouverts par le processus.

Pour retrouver quels sont les processus exécutés par le système, il suffit d'exécuter la commande `ps`.

Pour voir tous les processus exécutés par l'utilisateur :

```
ps -ux
```

Pour voir les processus exécutés par l'utilisateur dans le terminal courant :

```
ps T
```

Pour voir tous les processus du système :

```
ps aux
```

La commande `top` permet d'afficher des informations en continu sur l'activité du système. Elle permet surtout de suivre les ressources que les processus utilisent (quantité de RAM, pourcentage de CPU, durée d'exécution d'un processus depuis son démarrage).

L'option `-d` permet de spécifier des délais de rafraîchissement (en secondes) :

```
top -d
```

En cours d'utilisation de `top`, il est possible de gérer les processus (changer les priorités des processus, leur envoyer un signal ...) de manière interactive.

b) Arrêter ou envoyer un signal à un processus

La commande `kill` permet d'arrêter ou d'envoyer un signal à un processus, syntaxe : `kill [-Numéro-du-signal] PID`.

Les principaux signaux sont :

- `1` (`SIGHUP`) : le signal de numéro 1, `SIGHUP` (en anglais *hang up*), est envoyé par le processus « parent » à tous ses « enfants » lorsqu'il termine son activité. La plupart des démons redéfinissent ce signal en le transformant en « relire les fichiers de configuration » ;
- `2` (`SIGINT`) : signal d'interruption d'un processus. Il est équivalent à celui envoyé par la combinaison de touches <Ctrl> et <C> ;
- `9` (`SIGKILL`) : termine le processus via un appel noyau, donc sans sauvegarde ;
- `15` (`SIGTERM`) : signal par défaut de la commande `kill`. Il exécute le code de terminaison (s'il y arrive) et vide la mémoire. C'est donc le signal de fin d'un processus le plus propre.

Pour mémoire, Il existe 63 signaux différents (*tableau 6*).

Tableau 6. Signaux pour les processus

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL
5) SIGTRAP	6) SIGABRT	7) SIGBUS	8) SIGFPE
9) SIGKILL	10) SIGUSR1	11) SIGSEGV	12) SIGUSR2
13) SIGPIPE	14) SIGALRM	15) SIGTERM	17) SIGCHLD
18) SIGCONT	19) SIGSTOP	20) SIGTSTP	21) SIGTTIN
22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO
30) SIGPWR	31) SIGSYS	32) SIGRTMIN	33) SIGRTMIN+1
34) SIGRTMIN+2	35) SIGRTMIN+3	36) SIGRTMIN+4	37) SIGRTMIN+5
38) SIGRTMIN+6	39) SIGRTMIN+7	40) SIGRTMIN+8	41) SIGRTMIN+9
42) SIGRTMIN+10	43) SIGRTMIN+11	44) SIGRTMIN+12	45) SIGRTMIN+13
46) SIGRTMIN+14	47) SIGRTMIN+15	48) SIGRTMAX-15	49) SIGRTMAX-14
50) SIGRTMAX-13	51) SIGRTMAX-12	52) SIGRTMAX-11	53) SIGRTMAX-10
54) SIGRTMAX-9	55) SIGRTMAX-8	56) SIGRTMAX-7	57) SIGRTMAX-6
58) SIGRTMAX-5	59) SIGRTMAX-4	60) SIGRTMAX-3	61) SIGRTMAX-2
62) SIGRTMAX-1	63) SIGRTMAX		

Si l'on ne connaît pas le PID du processus auquel on veut envoyer un signal, on peut le désigner par son nom en utilisant la commande `killall`.

Syntaxe : `killall SIGNAL nom_du_processus`.

b) Priorité d'un processus

La **priorité d'un processus** peut être connue en examinant la colonne PRI du résultat des commandes `ps -l` ou `top`. Plus la priorité est grande, plus de temps CPU est offert au processus.

Le noyau gère la priorité moyennant une file d'attente de processus. Un utilisateur autorisé peut modifier la priorité des processus à travers une **valeur de gentillesse** ou **valeur nice** (colonne NI du résultat des commandes `ps -l` ou `top`).

La valeur nice varie de -20 à +19, plus elle est élevée, moins le processus est prioritaire.

La commande `nice` permet de modifier la priorité d'un processus au démarrage du processus. Syntaxe :

```
nice [-n Valeur] [Commande [Arguments ...]]
```

Comme premier paramètre, la commande `nice` attend l'indication de la valeur nice. L'utilisateur standard ne peut utiliser qu'une valeur nice positive, seul le super-utilisateur (*root*) peut utiliser des valeurs négatives.

Dans l'exemple suivant `mon_programme` est lancé avec une valeur nice égale à 19, c'est à dire avec une priorité la moins basse possible.

```
nice -n 19 mon_programme
```

La commande `renice` permet de modifier la priorité d'un ou plusieurs processus en cours d'exécution. Cette commande peut prendre comme paramètres des identifiants de processus, des identifiants de groupes de processus ou des noms d'utilisateurs. Utiliser `renice` sur un groupe de processus implique que tous les processus de ce groupe auront leur priorité modifiée. Lorsqu'un utilisateur est spécifié, tous les processus appartenant à celui-ci auront leur priorité modifiée.

Syntaxe :

```
renice [-n] nouvelle_valeur_nice [[-p] pid ] [-g] prgg ] [-u] utilisateur]
```

Pour obtenir le PID de « mon_processus » :

```
ps -l | grep mon_processus
```

Pour modifier la valeur nice :

```
renice -n 10 -p PID_de_mon_processus
```

c) Les processus et le shell

Si on lance un programme depuis un terminal, l'interpréteur de commande (shell) ne rend la main que quand le programme ainsi lancé est terminé. Il est possible d'interrompre ce programme et de reprendre la main en utilisant la combinaison de touches <Ctrl>-<Z>.

Ajouter le signe « & » à la fin de la ligne de commande permet de lancer le programme en arrière-plan – on parle aussi de tâche de fond – et de reprendre tout de suite la main : on revient directement au *prompt* du shell. Cette possibilité est surtout utile lorsqu'on travaille en mode « console », ou à distance, et qu'on ne dispose que d'une seule fenêtre ouverte.

Dans l'exemple ci-dessous, le processus lancé en tâche de fond depuis le shell est l'éditeur emacs. La première valeur, entre crochets, est le **numéro de job**. La seconde valeur est le PID de emacs.

```
emacs&
[1] 7522
```


Si plusieurs shells sont utilisés simultanément – par exemple avec plusieurs fenêtres de console – chacun aura ses propres processus fils et ne pourra agir sur ceux des autres shells. Le numéro de job identifie un processus de façon univoque par rapport au shell à partir duquel il a été lancé, mais plusieurs processus lancés à partir de shells différents peuvent avoir le même numéro de job.

On peut alors utiliser la commande `jobs` qui affichera la liste des processus s'exécutant en arrière-plan.

On peut faire passer un processus en avant-plan (*foreground*) par la commande `fg` et le remettre en arrière-plan (*background*) par la commande `bg`.

```
fg 1
```

d) La commande nohup

`nohup` permet de lancer un processus qui pourra rester actif aussi longtemps que nécessaire, et ceci même après la déconnexion de l'utilisateur l'ayant initié.

Par défaut, quand un utilisateur se déconnecte, le shell est quitté et un signal SIGHUP (signal 1) est envoyé aux processus enfants de ce shell. L'action associée à ce signal est l'arrêt de ces processus.

`nohup` ignore le signal SIGHUP (d'où son nom) puis exécute le programme indiqué en ligne de commande, de sorte que celui-ci ignorera le signal SIGHUP.

Dans l'exemple qui suit, le script « mon_calcul_long » est lancé avec la `nohup` pour que, si l'utilisateur se déconnecte avant que son exécution soit terminée, elle ne soit pas interrompue. Notons qu'en plus d'utiliser la commande `nohup` pour ignorer le signal SGHUP, on tape « & » en fin de ligne de commande pour que le script s'exécute en tâche de fond.

```
nohup mon_calcul_long &
```

H. Exercices

1. **En bash, pour afficher la liste des dernières commandes entrées, vous utilisez la commande _____.**

2. **Dans le shell Bash, la commande `!!` a le même effet que**

   - ❑ <Ctrl>-<P> et <Entrée>

- ❑ <Ctrl>-<N> et <Entrée>
- ❑ <Ctrl>-<U> et <Entrée>
- ❑ <!>-<2>

3. Quelle touche permet d'aller en fin de ligne ?

- ❑ </>
- ❑ <$>
- ❑ <0>
- ❑ <9>

4. La commande ls -l /etc/ | grep 'd'

- ❑ affiche tous les répertoires du répertoire /etc/.
- ❑ affiche les fichiers du répertoire /etc/ qui contiennent « d ».
- ❑ affiche les répertoires du répertoire /etc/ ainsi que les fichiers dont le nom contient un « d ».

5. **La commande pstree permet** :

- ❑ d'afficher l'arborescence des processus par « famille ».
- ❑ d'afficher uniquement les processus d'un utilisateur.
- ❑ d'afficher l'occupation du processeur par les processus.

6. Supposons que vous ayez un programme en cours appelé « monprog ». Vous souhaitez diminuer l'utilisation du processeur de ce programme. Laquelle des commandes suivantes pouvez-vous utiliser?

- ❑ `renice +1 -p `ps -a | grep monprog | cut -b 1-6``
- ❑ `ps h -o pid -C monprog | xargs nice +1 -`
- ❑ `nice +1 monprog`
- ❑ `renice +1 -u `whoami` monprog`

Annexe 1. Exemple d'examen de certification

Voici un exemple d'énoncé d'examen LPIC 101, suivi des réponses.

Cet exemple est destiné à vous aider à évaluer vos connaissances, et également à vous préparer au style des questions posées lors de l'examen de certification. Il est en effet nécessaire de s'habituer à la formulation des questions, qui peut parfois paraître ambiguë, ainsi qu'aux questions à choix multiples qui sont courantes dans le monde anglo-saxon.

Lors de l'examen, vous disposez d'environ une minute par question.

Énoncé

1. **Comment est représenté le périphérique IDE esclave du deuxième contrôleur ?**
   - ❑ A hda.
   - ❑ B sdc.
   - ❑ C hdc1.
   - ❑ D hdd.

2. **La commande _____ permet de lister les périphériques USB présents sur la machine.**

3. **Comment peut-on afficher le contenu d'un fichier en sens inverse ?**
   - ❑ A `cat -r`.
   - ❑ B `tail`.

   - ❑ C `tac`.
   - ❑ D `sort -r`.

4. **Le choix du périphérique de démarrage de votre machine se fait**
   - ❑ A dans le BIOS.
   - ❑ B avec une option du chargeur de démarrage (*boot loader*).
   - ❑ C automatiquement.

5. **Quels droits sont positionnés par la commande `chmod 1777 un_repertoire` sur un répertoire (plusieurs réponses) ?**
   - ❑ A Elle permet à tous les utilisateurs d'accéder au répertoire.
   - ❑ B Elle interdit l'accès aux autres utilisateurs que le groupe.
   - ❑ C Elle permet à tous de créer des fichiers dans ce répertoire, mais pas de modifier ou d'effacer ceux des autres.

6. **La commande `locate`**
   - ❑ A fonctionne comme `find` mais avec des options différentes.
   - ❑ B est plus efficace que `find` car elle utilise un index.
   - ❑ C permet de changer la langue par défaut de Linux.

7. **Vous créez un nouveau fichier. Les droits sont « rw-r--r— ». La valeur de `umask` est (plusieurs réponses)**
   - ❑ A 666.
   - ❑ B 022.
   - ❑ C 123.
   - ❑ D 133.

8. **Quelle différence y a-t-il entre les systèmes de fichiers ext2 et ext3 (plusieurs réponses) ?**
   - ❑ A ext3 est un nouveau système, incompatible avec ext2.
   - ❑ B ext3 est une simple extension de ext2 comprenant la journalisation.
   - ❑ C on peut passer de ext2 à ext3 et inversement très facilement.

9. Le fichier de la commande `passwd` a le bit _____ positionné pour permettre à tous les utilisateurs de changer leur mot de passe (modifier le fichier /etc/shadow).

10. Quelle commande allez-vous utiliser afin d'afficher les messages du noyau lors du démarrage du système ?

- ❑ A `mess`.
- ❑ B `dmesg`.
- ❑ C `bootmsg`.
- ❑ D `lsmsg`.

11. Quelle commande allez-vous utiliser afin d'afficher toutes les variables ?

- ❑ A `set`.
- ❑ B `varlist`.
- ❑ C `vshow var`.
- ❑ D `ps`.

12. La commande `killall` permet de

- ❑ A tuer tous les processus d'un utilisateur.
- ❑ B tuer un processus sans connaître son PID.
- ❑ C envoyer tous les signaux standard à un processus.

13. La commande `paste` permet de

- ❑ A coller un fichier à la suite d'un autre.
- ❑ B joindre deux fichiers côte à côte simplement sans critères.
- ❑ C laisser un processus en mémoire.

14. Pour changer la date de modification d'un fichier vous pouvez utiliser la commande (plusieurs réponses)

- ❑ A `date`.
- ❑ B `timestamp`.
- ❑ C `touch`.
- ❑ D `vi`.

15. Quel fichier contient la liste des partitions qui seront montées au démarrage ?

- ❑ A /etc/inittab.
- ❑ B /etc/fstab.
- ❑ C /etc/mtab.

16. En bash, la commande `set`

- ❑ A affiche toutes les variables.
- ❑ B affecte une valeur à une variable.
- ❑ C affiche les variables exportées.

17. Pour changer de niveau d'exécution vers le niveau 2 vous

- ❑ A lancez la commande `reinit 2`.
- ❑ B modifiez la ligne `initdefault` du fichier /etc/inittab et redémarrez la machine (*reboot*).
- ❑ C lancez la commande `shutdown -n 2`.
- ❑ D aucune des réponses n'est valable.

18. Quel fichier indique l'action à effectuer lors de l'appui sur <ctrl>-<alt>-<del> ?

- ❑ A /etc/keyboard.
- ❑ B /etc/inittab.
- ❑ C /etc/passwd.

19. Pour rechercher un fichier texte d'un utilisateur sur votre disque, vous utilisez

- ❑ A `locate`.
- ❑ B `whereis`.
- ❑ C `who`.
- ❑ D `find`.

20. En bash, pour afficher la liste des dernières commandes entrées, vous utilisez la commande _____.

21. Que signifie wait dans le fichier /etc/inittab ?

- ❑ A Attendre la fin de la commande pour continuer.
- ❑ B Attendre une entrée au clavier.
- ❑ C Attendre quelques minutes.

22. Comment obtenir de l'information sur une commande et effectuer une recherche par mots-clefs ?

- ❑ A `whoami`.
- ❑ B `whatis`.
- ❑ C `apropos`.
- ❑ D `which`.

23. La priorité d'un processus peut être diminuée

- ❑ A par son propriétaire à tout moment.
- ❑ B par tous les utilisateurs.
- ❑ C par son propriétaire au moment du lancement seulement.

24. La commande kill -HUP 1664

- ❑ A détruit le processus 1664.
- ❑ B force le processus 1664 à relire ces fichiers de configuration si c'est un « démon ».
- ❑ C redémarre le processus 1664.

25. Comment faire la sauvegarde d'une arborescence en conservant les droits des fichiers ?

- ❑ A Utiliser `cp` avec l'option `-a`.
- ❑ B Utiliser `cp` avec l'option `-d`.
- ❑ C Utiliser `cp` avec l'option `-v`.

26. Un programme mon_programme doit envoyer toutes ses sorties dans un fichier et ne doit rien écrire sur la console. Quelle commande utilisez-vous ?

- ❑ A `mon_programme > lefichier 2>&1`.
- ❑ B `mon_programme > lefichier`.
- ❑ C `mon_programme 1> /dev/null 2> lefichier`.

27. Comment peut-on installer le paquetage RPM leprogramme-1.2.3.4.i386.rpm ? (plusieurs réponses)

- ❑ A `rpm -Uvh leprogramme-1.2.3.4.i386.rpm`.
- ❑ B `rpm -i leprogramme-1.2.3.4.i386.rpm`.
- ❑ C `rpm -u leprogramme-1.2.3.4.i386.rpm`.
- ❑ D `rpm -e leprogramme-1.2.3.4.i386.rpm`.

28. Quels répertoires doivent appartenir au système de fichier root (plusieurs réponses) ?

- ❑ A /etc
- ❑ B /home
- ❑ C /lib
- ❑ D /usr
- ❑ E /root

29. Quels droits devrait avoir le fichier /usr/bin/passwd ?

- ❑ A 4511.
- ❑ B 6400.
- ❑ C 4222.

30. Comment peut-on afficher les lignes 10 à 20 d'un fichier de 30 lignes ?

- ❑ A `head -n 10-20 lefichier`.
- ❑ B `tail -n 10-20 lefichier`.
- ❑ C `head -n 20 lefichier | tail`.
- ❑ D `ns -n 10-20 lefichier`.

31. Comment peut-on afficher le contenu d'un fichier avec les lignes numérotées en sens inverse ?

- ❑ A `cat lefichier | nl | sort -r`.
- ❑ B `cat lefichier | tac -l`.
- ❑ C `cat lefichier | nl | tac`.

32. Quelle commande montre l'occupation d'un répertoire '/usr/XXX' ?

- ❑ A `du -s /usr/XXX`.
- ❑ B `df -h /usr/XXX`.
- ❑ C `ls -iR /usr/XXX`.

33. Quelle commande permet de voir toutes les partitions montées ?

- ❑ A `mount -a`.
- ❑ B `df`.
- ❑ C `show mount`.
- ❑ D `du -m`.

34. Quelle commande permet de voir l'espace restant sur une partition montée ?

- ❑ A `top`.
- ❑ B `free`.
- ❑ C `df`.
- ❑ D `fsstat`.

35. Quelle commande permet de transformer un « i » en « l » dans un fichier (plusieurs réponses) ?

- ❑ A `sed`.
- ❑ B `vi`.
- ❑ C `paste`.
- ❑ D `cat`.

36. Vous effectuez la commande `Ma_Var=''Coucou''`, que produit la commande `echo ''Ma_Var, ça va ????''`

- ❑ A Coucou, ça va ????
- ❑ B Ma_Var, ça va ????
- ❑ C erreur de syntaxe.

37. Quelle commande permet d'afficher les dernières lignes d'un fichier avec un rafraîchissement permanent ?

- ❑ A `cat -f`.
- ❑ B `tail -n 1`.
- ❑ C `tac -l`.
- ❑ D `tail -f`.

38. La commande `kill 9 1664`

- ❑ A détruit le processus 1664.
- ❑ B met en pause le processus 1664.
- ❑ C force le processus 1664 à passer en priorité 9.
- ❑ D aucune de ces réponses n'est valable.

39. **La commande `ls -l /etc/ | grep 'd'`**

- ❑ A affiche tous les répertoires du répertoire /etc/.
- ❑ B affiche les fichiers qui contiennent 'd' du répertoire /etc/.
- ❑ C affiche les répertoires du répertoire /etc/ ainsi que les fichiers dont le nom contient un 'd'.

40. La commande head monfichier | wc –l

- ❑ A affiche toujours 10.
- ❑ B affiche 5.
- ❑ C aucune de ces réponses n'est valable.

41. Que se passe-t-il lorsque vous utilisez la commande `tail fichier1 fichier2 fichier3` ?

- ❑ A elle affiche la fin du fichier1.

- ❑ B elle affiche la fin de tous les fichiers séparés par « ==> » et « <== ».
- ❑ C elle produit une erreur de syntaxe.

42. Pour formater une partition, vous utilisez

- ❑ A `disk druid`.
- ❑ B `fdisk`.
- ❑ C `mkfs`.
- ❑ D `fsck`.

43. Pour ajouter et utiliser une partition, vous devez exécuter quelles commandes (plusieurs réponses) ?

- ❑ A `df`.
- ❑ B `mount`.
- ❑ C `fdisk`.
- ❑ D `mkfs`.

44. Quelle variable contient les chemins d'accès aux librairies du système ?

- ❑ A PATH.
- ❑ B LD_LIBRARY_PATH.
- ❑ C LIBPATH.
- ❑ D USRLIB.

45. Quelle commande est lancée au démarrage du système sur toutes les partitions contenues dans `/etc/fstab` ?

- ❑ A `mkfs`.
- ❑ B `fsck`.
- ❑ C `chksum`.
- ❑ D `verify`.

46. Linux utilise une zone spécifique lorsque la mémoire est saturée, cet espace s'appelle le _____.

47. Quelle commande permet de trouver l'emplacement d'un programme, les pages de manuel et le code source éventuel ?

- ❑ A `search`.
- ❑ B `find`.
- ❑ C `whereis`.
- ❑ D `apropos`.

48. Quelles commandes permettent de tester un système de fichiers formaté en `ext3` (plusieurs réponses) ?

- ❑ A `fsck.ext3`.
- ❑ B `fsck -t ext3`.
- ❑ C `tune2fs -j`.
- ❑ D `e3fsck`.

49. La commande `ln -s` permet

- ❑ A de créer un lien symbolique.
- ❑ B de mettre à jour les librairies du système.
- ❑ C de faire une sauvegarde d'un fichier.

50. Quelle commande VI faut-il entrer afin d'enregistrer un fichier et quitter l'éditeur vi ?

- ❑ A `:x`
- ❑ B `:wq`
- ❑ C `:re`
- ❑ D `:we`

51. À quoi sert la commande `updatedb` ?

- ❑ A À reconstruire la liste des librairies du système.
- ❑ B À mettre à jour l'index pour la commande `locate`.
- ❑ C À mettre à jour la liste des utilisateurs.

52. À quoi sert la commande `ldconfig` ?

- ❑ A À prendre en compte les nouvelles librairies dont le chemin est contenu dans le fichier /etc/ld.so.conf.
- ❑ B À configurer les chemins des librairies du système.
- ❑ C À préparer la compilation d'un programme.

53. Ajouter _____ au début d'une ligne de commande afin de modifier la priorité d'exécution du processus qui en résulte.

54. Lister les fichiers du paquetage gdm installé :

- ❑ A `dpkg -s gdm`
- ❑ B `dpkg -l gdm`
- ❑ C `dpkg -L gdm`
- ❑ D `dpkg -q gdm`

55. Quelle commande permet d'effacer tous les fichiers et les sous-répertoires de l'arborescence /home/nicolas/truc/ ?

- ❑ A `rm -r /home/nicolas/`.
- ❑ B `rmdir /home/nicolas/truc`.
- ❑ C `rm -r /home/nicolas/truc/*`.

56. La commande `pstree` permet

- ❑ A d'afficher l'arborescence des processus par « famille ».
- ❑ B d'afficher uniquement les processus d'un utilisateur.
- ❑ C d'afficher l'occupation du processeur par les processus.

57. Quel utilitaire permet de sélectionner les colonnes d'un fichier ?

- ❑ A `col`.
- ❑ B `join`.
- ❑ C `cut`.
- ❑ D `tail`.

58. Vous lancez les commandes suivantes en tant qu'utilisateur non privilégié :

```
$ nice -n 10 leprogramme1.
$ nice -n -1 leprogramme2.
$ nice -n 16 leprogramme3.
```

Lequel aura la priorité la plus élevée ?

- ❑ A leprogramme1.
- ❑ B leprogramme2.
- ❑ C leprogramme3.

59. Vous voulez connaître le nombre de fichiers contenus dans le répertoire /bin, vous entrez la commande _____.

60. La commande `top` vous permet de (plusieurs réponses)

- ❑ A changer la priorité d'un processus.
- ❑ B envoyer un signal à un processus.
- ❑ C afficher le début d'un fichier.
- ❑ D aucune de ces réponses n'est valable.

Réponses examen LPI 101

1. D
2. lsusb
3. C
4. A
5. A, C
6. B
7. B, D
8. B, C
9. SETUID ou 04000
10. B
11. A
12. B

13. B

14. C, D (D un peu limite : pas de changement de date si on n'enregistre pas)

15. B

16. A

17. B

18. B

19. D (A est aussi valable avec locate si updatedb n'est pas lancé en nobody, mais ce n'est effectivement pas valable avec slocate)

20. history

21. A

22. B, C

23. A

24. B (mais ça dépend des services)

25. A

26. A

27. A, B

28. A,C

29. A

30. C

31. C

32. A

33. B (ne montre que les systèmes de fichiers standard, sinon utiliser mount sans argument)

34. C

35. A, B

36. B

37. D

38. D

39. C

40. A (à condition que le fichier contienne au moins 10 lignes !!! donc plutôt C)

41. B

42. C

43. B, C, D

44. B

45. B

46. swap

47. C

48. A, B

49. A

50. A, B

51. B

52. A

53. nice

54. C

55. C (correct uniquement s'il n'y a pas de fichiers cachés dans le sous-dossier truc)

56. A (B aussi valable si on fournit un nom d'utilisateur en option)

57. C

58. A

59. ls /bin | wc -l

60. A, B

Table des figures et tableaux

Index des mots-clés

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Les auteurs

Zied Bouziri (Tunisie) : enseignant depuis 2003 au département Informatique de l'Institut supérieur des études technologiques de Charguia (ISET, Tunis), option réseaux informatiques. Il est ingénieur en informatique, diplômé de l'École nationale des sciences de l'informatique de Tunis (ENSI). Entre 1999 et 2003, il a été ingénieur conception et développement au département recherche et développement chez Alcatel.

Niry H. Andriambelo (Madagascar) : membre fondatrice de l'Association malagasy des utilisateurs de logiciels libres et formatrice de formateurs GNU/Linux, Niry Andriambelo est ingénieur informatique diplômé de l'Université d'Antananarivo et, depuis 2004, coordinatrice pour les systèmes et réseaux universitaires francophones dans l'océan Indien.

Andrei Boyanov (Bulgarie) : directeur d'Active Solutions et membre de la commission technique et développement de l'Institut professionnel Linux. Andrei Boyanov est ingénieur en informatique issu de l'Université technique de Sofia et formateur des personnels d'encadrement de l'enseignement supérieur dans le domaine des systèmes et des réseaux Linux.

Nicolas Larrousse (France) : concepteur de programmes Transfer dans les systèmes et réseaux sous Linux ainsi que de formations à la certification LPI. Nicolas Larrousse est ingénieur informatique. Il enseigne les systèmes d'information à l'Université de Versailles et exerce, depuis 1992, au Centre national de la recherche scientifique (CNRS), à Paris.

Véronique Pierre (France) : consultante indépendante en édition scientifique multimédia.