

Projet 50h
Création d'un LiveDVD Linux

HABERER Thomas <haberer@turing.u-strasbg.fr>
HUMANN Alexandre <humann@turing.u-strasbg.fr>

Année universitaire 2005-2006

Table des matières

1	Présentation	2
2	Système de base	3
2.1	Copie du système de base	3
2.2	Copie de l'arbre <i>portage</i>	3
2.3	Intégration dans le système	3
3	Personnalisation du système	4
3.1	Portage, ebuild : comment ça marche?	4
3.2	Gestion des services, démons	5
3.3	Installation des applications basiques	5
3.4	Serveur X11	6
3.5	Gestionnaire de bureau	8
4	Noyau Linux	9
4.1	Notions essentielles	9
4.2	Paramétrage du noyau	9
4.3	Pilotes optionnels	11
4.4	Gestionnaire de démarrage	11
5	Création de l'image du LiveDVD	13
5.1	Sauvegarde et nettoyage	13
5.2	Création de l'image Squash	13
5.3	Configuration d'ISOLinux	13
5.4	Création de l'image ISO	14
6	Utilisation	15

1 Présentation

Dans le cadre du projet 50 heures, effectué en 1^{ère} année de Master Informatique, nous avons réalisé un LiveDVD. Le document présent retrace – de façon précise – les étapes de création du LiveDVD.

Choix de la distribution

Avant toute chose, la première question à se poser est le choix de la distribution Linux à utiliser. Dans le monde des LiveCD, la distribution Knoppix est sans doute la référence. Cependant, le choix s'est porté sur Gentoo Linux, pour plusieurs raisons :

- la base de l'installation de ce système est réalisé avec un LiveCD
- une documentation abondante est disponible pour cette distribution, notamment concernant la réalisation de LiveCD
- nous utilisons cette distribution au quotidien, nous sommes donc en terrain connu

Support : CD ou DVD ?

Dans la suite de ce document, nous considérerons que le support de stockage est le DVD. Cependant, il est tout à fait concevable de se limiter à un cédérom. La seule contrainte étant l'espace disponible sur le média.

Pré-requis

Outre une bonne connaissance des outils GNU/Linux, le seul véritable pré-requis est la disposition d'un espace disque conséquent. Il est souhaitable d'avoir environ 4Go d'espace disque libre. Cette espace étant pris pour le système en lui-même (celui qui sera gravé sur le support), mais aussi l'image SquashFS et l'image ISO.

Concernant ces images, il est nécessaire de disposer de :

- `squashfs-tools` : outils de création d'image compressée au format Squash 3.0
- `mkisofs` : outils de création d'image ISO9660 (image de cédérom)
- n'importe quel logiciel permettant de graver cette image (K3B, Nero...)

2 Système de base

La première étape consiste à créer l'environnement du futur système. Considérons que le répertoire de stockage est `/livecd` et que le répertoire contenant le système est `/livecd/source`.

2.1 Copie du système de base

Le système de base est disponible sous forme d'archive, nommé `stage3-i686-2006.0.tar.bz2`¹. Nous allons l'extraire dans le répertoire `source` :

```
/livecd/source # tar xjvfp /chemin/vers/stage3-i686-2006.0.tar.bz2
```

Remarque notez les paramètres de la commande `tar`, notamment le `p` qui indique que les fichiers extraits conserveront leur propriétaire et leurs droits tels qu'ils sont indiqués dans l'archive.

2.2 Copie de l'arbre *portage*

À l'instar de FreeBSD (*ports*), Gentoo utilise des fichiers (*ebuild*) décrivant comment compiler et installer un programme. Ceux-ci sont rangés dans des sous-répertoires formant des catégories d'applications (développement, système, X11, jeux...) : cette hiérarchie est nommée l'arbre *portage*.

L'archive précédente ne contient pas ces fichiers. Pour en disposer, nous avons deux solutions :

1. le système hôte est une distribution Gentoo, utilisons donc son arbre via la commande

```
/livecd/source # mount -o bind /usr/portage /livecd/source/usr/portage
```

2. téléchargement de l'archive `portage-latest.tar.bz2` et décompression dans `usr`

```
/livecd/source # wget ftp://.../portage-latest.tar.bz2 -P /tmp  
/livecd/source # tar xjvf /tmp/portage-latest.tar.bz2 -C /livecd/source/usr
```

2.3 Intégration dans le système

Maintenant que nous disposons d'un environnement de base, il va nous falloir le compléter avec les outils nécessaires, un noyau et ses pilotes, etc. La seconde étape consiste à utiliser la commande `chroot` pour changer notre environnement de base (la racine), en :

1. montant les pseudo-partitions `dev`, `proc`, `sys`

```
/livecd/source # mount -o bind /proc /livecd/source/proc  
/livecd/source # mount -o bind /dev /livecd/source/dev  
/livecd/source # mount -o bind /sys /livecd/source/sys
```

2. puis, changement de la racine

```
/livecd # chroot source /bin/bash
```

3. et enfin, mise à jour de l'environnement

```
/ # source /etc/profile  
/ # env-update
```

¹Une liste de miroir disposant de ce fichier sera donné à la fin du document

3 Personnalisation du système

3.1 Portage, ebuild : comment ça marche ?

Cette partie est une initiation aux commandes usuelles et propres à l'installation d'application sous Gentoo.

Outil emerge

La commande de base sous Gentoo est **emerge**. Elle permet d'installer un programme, en spécifiant en paramètre son nom.

```
# emerge zsh
```

À cela s'ajoute un ensemble de paramètres permettant de modifier l'action. Les incontournables étant :

- **-s** nom recherche *nom* parmi l'ensemble de ebuilds
- **-v** mode bavard, un complément d'informations est affiché (version, taille...)
- **-p** n'affiche que la liste des programmes à installer, puis quitte (pas d'installation)
- **-a** affiche la liste des programmes à installer et attends une confirmation avant d'installer
- **-u world** recherche les programmes installés et pouvant (devant) être mise à jour

Comme exemple, regardons quels paquets installer pour disposer de la (magnifique) commande **vim** :

```
# emerge -vp vim
```

```
These are the packages that I would merge, in order:
```

```
Calculating dependencies ...done!
```

```
[ebuild N   ] dev-util/ctags-5.5.4-r2  254 kB
[ebuild N   ] app-editors/vim-core-6.4 -acl -bash-completion -livedcd +nls 4,752 kB
[ebuild N   ] app-editors/vim-6.4     -acl -bash-completion -cscope +gpm -minimal +nls \
                                         +perl +python -ruby -vim-with-x 0 kB
[ebuild N   ] app-vim/gentoo-syntax-20051221 -ignore-glep31 18 kB
```

Variable USE

Dans la commande précédente, remarquons les mots greffés à côté du nom de l'application. Ce sont les fonctionnalités additionnelles que propose le programme. Celles-ci peuvent être activées (+) ou désactivées (-) via la variable **USE** et permettent de gérer plus finement les dépendances de chaque paquet (contrairement aux distributions classiques telles que Debian).

Le paramétrage de cette variable s'effectue dans le fichier *make.conf* et/ou via le fichier */etc/portage/package.use*.

Paramétrage via make.conf

Le fichier */etc/make.conf* permet de définir le comportement de la commande **emerge**, et donc de l'installation des applications. Le fichier copié dans le système de base est succinct mais suffisant pour débiter. Un exemple de configuration évoluée est disponible dans le fichier */etc/make.conf.example*. Citons quelques paramètres essentiels :

- **USE** définit les fonctionnalités additionnelles des applications
- **CFLAGS** les paramètres d'optimisations pour le compilateur (gcc)
- **GENTOO_MIRRORS** les miroirs disposant des sources
- **SYNC** le miroir de synchronisation de l'arbre portage

Un jeu de paramètres possibles (ceci ne constituant qu'un exemple et en aucun cas un fichier *make.conf* complet) :

```
USE="-alsa -arts -oss -esd -gnome -kde -xmms -spell -gpm gtk gtk2 \
      -qt ipv6 ssl -cups X xinetd hardened mysql"
CFLAGS="-march=i686 -O2 -pipe"
GENTOO_MIRRORS="ftp://ftp.proxad.net/mirrors/ftp.gentoo.org http://distfiles.gentoo.org"
SYNC="rsync://rsync.europe.gentoo.org/gentoo-portage"
```

Commandes additionnelles

esearch À l'instar de la commande **locate**, **esearch** permet de faire une recherche (très) rapide dans l'ensemble des *ebuild* disponibles. La fonctionnalité est équivalente à **emerge -s**, mais en moins longue. Elle repose sur une base, mise à jour via la commande **eupdatedb**.

equery Cette commande permet de recueillir des informations sur les programmes installés (ou non), telles que les dépendances, la liste des fichiers installés pour un programme, etc.

Mise à jour de l'arbre La commande **emerge** permet également de mettre à jour l'arbre portage, en exécutant **emerge --sync**. Cette commande peut échouer si la machine est derrière un pare-feu, l'alternative pour la synchronisation est la commande **emerge-webrsync**.

3.2 Gestion des services, démons

La gestion des services est simplifiée sous Gentoo. La notion de *runlevel* sous forme numérique (rc1, rc3, etc.) est remplacée par des niveaux de démarrage nommé :

- **boot** regroupe les services critiques démarrés après *init* (vérification des partitions système, chargement des modules, etc.)
- **default** regroupe les services de base, démarrés après **boot** et avant la phase de *login*
- **single** et **nonetwork**

La commande **rc-status** permet de consulter les différents niveau d'exécution (*runlevel*) et leurs services associés.

Démarrage et arrêt d'un service

Le démarrage et l'arrêt d'un service s'effectue avec la commande

```
# /etc/init.d/service action
```

où le terme **service** désigne le nom du service et **action** est :

- **status** pour connaître l'état du service
- **start** pour démarrer le service
- **stop** pour arrêter le service
- **restart** pour redémarrer le service
- **zap** pour réinitialiser l'état du service
- **reload** pour demander au service de recharger sa configuration sans s'arrêter (tous les services ne supportent pas cette option)

Pour démarrer le service **ssh**, par exemple, la commande sera

```
# /etc/init.d/sshd start
* Starting sshd ... [ ok ]
```

La commande **rc-status** permet de connaître l'état des services appartenant à un *runlevel* :

```
# rc-status
Runlevel: default
netmount           [ started ]
local              [ started ]
hdparm             [ started ]
syslog-ng          [ started ]
fcron              [ started ]
sshd               [ started ]
```

Ajout d'un service à un *runlevel*

C'est avec la commande **rc-update** que les services sont ajoutés ou retirés d'un *runlevel*

```
# rc-update action service runlevel
```

où le terme **action** est soit **add**, soit **del**, le terme **service** désignant le nom du service et **runlevel** désignant le nom du niveau d'exécution. Par exemple :

```
# rc-update add sshd default
* sshd added to runlevel default
* rc-update complete.
```

3.3 Installation des applications basiques

Avec la commande **emerge**, nous pouvons installer les outils essentiels à l'utilisation du LiveDVD.

Outils système

- `syslog-ng`, le gestionnaire de journaux système
- `exim`, l'agent de distribution de courrier électronique
- `fcron`, le planificateur de tâches
- `grub`, le gestionnaire de démarrage
- `mailx`, un outils permettant d'envoyer du courrier à partir d'une console

Gestion de périphériques

- `hotplug`, et `coldplug` ces outils chargent automatiquement les pilotes nécessaires au fonctionnement du matériel
- `pcmcia-cs`, le gestionnaire de cartes PCMCIA

Réseau et connectivité

- `dhcpcd`, un client DHCP
- `wireless-tools`, manipulations de cartes WIFI
- `prism54-firmware`, le *firmware* essentiel au bon fonctionnement des carte WIFI à base de chipset Prism

Pensons à ajouter les services au *runlevel default* pour les initialiser au démarrage du LiveDVD

```
# rc-update add syslog-ng default
# rc-update add fcron default
# rc-update add hotplug default
# rc-update add coldplug default
# rc-update add pcmcia default
```

3.4 Serveur X11

L'installation d'un serveur X11 n'est pas forcément nécessaire, l'ajout de ce type d'application dépendant des fonctionnalités à offrir avec le LiveDVD.

Sous Gentoo, le serveur X11 recommandé est celui d'*X.org*, au détriment du classique *XFree86*².

```
# emerge -va xorg-x11
```

Note La compilation de `xorg-x11` requiert 1 bonne heure, prenez votre mal en patience.

À propos de la configuration automatique

La configuration d'un serveur X11 est étroitement lié aux périphériques (carte graphique, écran, clavier, souris, etc.) de la machine. À l'heure où nous écrivons ce manuel, il n'existe pas de script permettant de générer automatiquement une configuration optimale. La distribution Knoppix dispose d'outils remplissant ce rôle – voir l'ebuild `ddcxinfo-knoppix` – mais leur utilisation n'a pas donné de résultats significatifs. Nous nous contenterons d'un fichier générique, supportant (normalement) un jeu de matériel typique.

Configuration générique

Le fichier de configuration de *X.org* est `/etc/X11/xorg.conf`. Il est identique à celui de *XFree86* et donc, il est possible de récupérer une configuration déjà opérationnelle.

Clavier Le support du clavier est (théoriquement) universel, la configuration peut donc se contenter de

```
Section "InputDevice"
    Identifier      "Keyboard0"
    Driver          "kbd"
    Option         "CoreKeyboard"
    Option         "XkbRules" "xorg"
    Option         "XkbModel" "pc105"
    Option         "XkbLayout" "fr"
EndSection
```

²Ce choix est motivé par un récent changement de licence d'*XFree86*, plus restrictive.

Souris Le support de la souris est plus délicat, car différents ports peuvent être rencontrés : série, PS/2 et USB. Le premier n'existe quasiment plus pour les souris, mais les deux autres sont très courants. Heureusement, il est possible de configurer plusieurs souris pour un serveur X11.

```
Section "InputDevice"
    Identifieur      "Serial Mouse"
    Driver           "mouse"
    Option           "Protocol" "Microsoft"
    Option           "Device"   "/dev/ttyS0"
    Option           "Emulate3Buttons" "true"
    Option           "Emulate3Timeout" "70"
    Option           "SendCoreEvents" "true"
EndSection
```

```
Section "InputDevice"
    Identifieur      "PS/2 Mouse"
    Driver           "mouse"
    Option           "Protocol" "IMPS/2"
    Option           "Device"   "/dev/psaux"
    Option           "Emulate3Buttons" "true"
    Option           "Emulate3Timeout" "70"
    Option           "SendCoreEvents" "true"
    Option           "ZAxisMapping" "4 5"
EndSection
```

```
Section "InputDevice"
    Identifieur      "USB Mouse"
    Driver           "mouse"
    Option           "Device"   "/dev/input/mice"
    Option           "SendCoreEvents" "true"
    Option           "Protocol" "IMPS/2"
    Option           "ZAxisMapping" "4 5"
EndSection
```

Carte graphique Le support de la carte graphique – et plus particulièrement son *chipset* – n'est pas automatique. Cependant, les cartes récentes sont compatible avec la norme VESA.

```
Section "Device"
    Identifieur      "Card0"
    Driver           "vesa"
    VendorName       "All"
    BoardName        "All"
    BusID            "PCI:1:0:0"
EndSection
```

Deux points à retenir :

- le paramètre `BusID` est optionnel et si il est absent de la configuration, le serveur X11 se chargera de retrouver le bus *PCI* ou *AGP* correspondant à la carte
- le paramètre `Driver` pourra être modifié par la suite, les valeurs possibles étant
 - `vesa` pour un support universel
 - `nv` ou `nvidia` pour les *chipset* Nvidia
 - `radeon` pour les *chipset* ATI Radeon
 - `r128` pour les *chipset* ATI Rage

Pour obtenir une liste de pilotes disponibles, consultez le fichier `/usr/X11R6/lib/X11/Cards`.

Moniteur La configuration d'un moniteur est délicate, puisqu'il n'est pas possible (à notre connaissance) de déterminer ses fréquences de rafraîchissement et ses résolutions. Là aussi, une configuration typique fera l'affaire.

```
Section "Monitor"
    Identifieur      "Monitor0"
    HorizSync        28.0 - 96.0
    VertRefresh      50.0 - 75.0
```

```

EndSection

Section "Screen"
    Identifier      "Screen0"
    Device          "Card0"
    Monitor         "Monitor0"
    DefaultColorDepth 16
    SubSection "Display"
        Depth      8
        Modes "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth      15
        Modes "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth      16
        Modes "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth      24
        Modes "1024x768" "800x600" "640x480"
    EndSubSection
EndSection

```

Les paramètres pourront être affinés par la suite, notamment la résolution de l'écran.

Configuration globale La dernière étape consiste à assembler tous les périphériques dans la section `ServerLayout`

```

Section "ServerLayout"
    Identifier      "X.Org Configured"
    Screen 0       "Screen0" 0 0
    InputDevice    "Keyboard0" "CoreKeyboard"
    InputDevice    "USB Mouse"
    InputDevice    "PS/2 Mouse"
EndSection

```

Nous ajouterons également un l'option `AllowMouseOpenFail` qui implique que le serveur X11 démarrera, même si il ne trouve pas une des souris que nous avons configuré (si il n'y a pas de souris USB par exemple).

```

Section "ServerFlags"
    Option          "AllowMouseOpenFail" "True"
EndSection

```

3.5 Gestionnaire de bureau

Il est recommandé d'installer un gestionnaire de bureau (*Desktop Environment*), facilitant l'utilisation du système. Dans le cadre d'un LiveDVD, nous avons opter pour *XFCE*, environnement basé sur *Gtk2*. Il est léger et fonctionnel (il n'occupe que 30Mo d'espace), constituant une bonne base de départ. D'autres outils X11 peuvent être ajoutés, en fonction des besoins.

4 Noyau Linux

Étape cruciale dans la réalisation d'un LiveDVD, la compilation et l'installation du noyau (*kernel*) Linux.

4.1 Notions essentielles

La phase de démarrage (*boot*) du LiveDVD repose sur plusieurs étapes :

1. initialisation du noyau
2. chargement de l'image *initial ramdisk*
3. pré-paramétrage du système
4. lancement du démon *init*

La particularité d'un LiveDVD vient du fait que le système n'est pas présent sur un disque dur classique, mais dans une image SquashFS compressée, stockée sur le DVD – qui est un support en lecture seule. L'étape cruciale du démarrage est le chargement de cette image via un périphérique *loop* (3.), qui devient la racine du système. Cette action est réalisée par un ensemble de scripts stockés dans le fichier *initrd* (ou *initramfs*), chargée par le gestionnaire de démarrage en même temps que le noyau.

Sous Gentoo, il existe un outils nommé `[genkernel]` qui compile et installe le noyau automatiquement. De plus, il sait générer des fichiers *initrd* embarquant tous les scripts nécessaires au démarrage d'un LiveDVD. Cet outils va considérablement nous simplifier la vie.

4.2 Paramétrage du noyau

Installation L'installation du noyau et de l'outil `genkernel` se fait via la commande `emerge`

```
# emerge -va gentoo-sources genkernel
```

```
These are the packages that I would merge, in order:
```

```
Calculating dependencies ...done!  
[ebuild N ] sys-kernel/gentoo-sources-2.6.16  
[ebuild N ] sys-kernel/genkernel-3.3.11d
```

Divers noyaux sont disponibles sous Gentoo, pour en obtenir la liste : `emerge -s sources` ou `esearch sources`.

Paramétrage basique

Les sources du noyau sont installées dans le répertoire `/usr/src`. La première étape de configuration consiste à définir tous les pilotes en tant que modules

```
# cd /usr/src/linux  
# make make allmodconfig
```

Paramétrage manuelle

Nous allons maintenant affiner la liste des pilotes à compiler dans le noyau, sachant qu'il est nécessaire d'avoir le support pour :

- l'*initial ramdisk* et les fichiers SquashFS
- les contrôleur IDE et /ou SATA (voir SCSI)
- les cédéroms au format ISO 9660

Nous obtenons une configuration semblable à celle-là :

```
Device Drivers --->  
  Block devices --->  
    <*> Loopback device support  
    <*> RAM disk support  
    (16) Default number of RAM disks  
    (9600) Default RAM disk size (kbytes)  
    [*] Initial RAM disk (initrd) support  
ATA/ATAPI/MFM/RLL support --->  
  <*> ATA/ATAPI/MFM/RLL support  
  <M> Include IDE/ATA-2 DISK support  
  [*] Use multi-mode by default
```

```

<*> Include IDE/ATAPI CDROM support
<*> SCSI emulation support
<*> generic/default IDE chipset support
[*] CMD640 chipset bugfix/support
[*] CMD640 enhanced support
[*] PNP EIDE support
[*] PCI IDE chipset support
[*] Sharing PCI IDE interrupts support
<*> Generic PCI IDE Chipset Support
<*> RZ1000 chipset bugfix/support
[*] Generic PCI bus-master DMA support
[*] Force enable legacy 2.0.X HOSTS to use DMA
[*] Use PCI DMA by default when available
[*] AEC62XX chipset support
[*] ALI M15x3 chipset support
[*] AMD and nVidia IDE support
[*] ATI IXP chipset IDE support
[*] CMD64{3|6|8|9} chipset support
[*] Compaq Triflex IDE support
[*] CY82C693 chipset support
[*] Cyrix/National Semiconductor CS5530 MediaGX chipset support
[*] AMD CS5535 chipset support
[*] HPT34X chipset support
[*] HPT36X/37X chipset support
[*] National SCx200 chipset support
[*] Intel PIIxN chipsets support
[*] IT821X IDE support
[*] NS87415 chipset support
[*] PROMISE PDC202{46|62|65|67} support
[*] ServerWorks OSB4/CSB5/CSB6 chipsets support
[*] Silicon Image chipset support
[*] SiS5513 chipset support
[*] SLC90E66 chipset support
[*] Tekram TRM290 chipset support
[*] VIA82CXXX chipset support
File systems --->
  CD-ROM/DVD Filesystems --->
    <*> ISO 9660 CDROM file system support
    [*] Microsoft Joliet CDROM extensions
    [*] Transparent decompression extension

```

Important Il est fortement conseillé de configurer le support pour les disques durs en module, d'où l'option <M> pour le pilote Include IDE/ATA-2 DISK support.

Compilation et installation

La configuration du noyau étant faite, nous pouvons passer à la compilation. C'est à ce stade que genkernel intervient.

```

# genkernel all --no-menuconfig --no-bootsplash --no-clean
* Gentoo Linux Genkernel; Version 3.3.11d
* Running with options: all --no-menuconfig --no-bootsplash --no-clean

* Linux Kernel 2.6.16-gentoo for x86...
* config: --no-clean is enabled; leaving the .config alone.
* >> Compiling 2.6.16-gentoo bzImage...
* >> Compiling 2.6.16-gentoo modules...
* >> Installing...
* >> Copying to bincache...
patching file coreutils/Config.in
patching file coreutils/Makefile.in

```

```

patching file include/applets.h
* busybox: >> Configuring...
* busybox: >> Compiling...
* busybox: >> Copying to cache...
* initramfs: >> Initializing...
*       >> Creating base_layout cpio archive...
*       >> Creating auxiliary cpio archive...
*       >> Creating busybox cpio archive...
*       >> Creating udev cpio archive...
*       >> Creating insmod cpio archive...
*       >> Creating modules cpio archive...

* Merging
*   initramfs-base-layout.cpio.gz
*   initramfs-aux.cpio.gz
*   initramfs-busybox-1.00-rt-mdstart.plasmaroo.cpio.gz
*   initramfs-insmod-0.9.15-pre4.cpio.gz
*   initramfs-udev-077.cpio.gz
*   initramfs-modules-2.6.16-gentoo.cpio.gz
*
* Kernel compiled successfully!

```

Les paramètres de la commande sont explicites :

- `all` indique qu'il faut générer le noyau, les modules et l'image `initrd`
- `-no-menuconfig` indique que nous ne souhaitons plus modifier la configuration du noyau
- `-no-bootsplash` indique qu'il ne faut pas intégrer le support pour le *bootsplash* (image affiché en plein écran durant le démarrage du noyau)
- `-no-clean` indique qu'il ne faut pas nettoyer les sources, c'est-à-dire retirer les fichiers issus de la compilation ³

Après exécution de cette commande, le nouveau noyau est compilé et installé dans le répertoire `/boot`.

Remarque Il est possible que des messages d'avertissement soient affichés durant la compilation.

4.3 Pilotes optionnels

À ce stade – et si tout s'est bien passé – nous disposons d'un système Linux opérationnel. Certains périphériques n'ont pas de pilotes directement intégrés dans le noyau Linux. L'arbre *portage* fournit quelques-uns de ces pilotes.

Pilotes Nvidia Les cartes graphiques à base de chipset Nvidia sont très courantes. Si nous souhaitons disposer d'un serveur X11 par la suite, il est fortement conseillé d'installer les pilotes adéquats.

```
# emerge -va nvidia-kernel nvidia-glx
```

Pilotes Wifi Certaines cartes réseaux Wifi disposent de chipset Atheros, ACX (Texas Instrument) ou même Intel (essentiellement les portable Centrino). Il faut également prévoir leur utilisation.

```
# emerge -va madwifi-ng madwifi-ng-tools acx ipw2100 ipw2200
```

À cela s'ajoute également le support du WPA.

```
# emerge -va wpa_supplicant
```

4.4 Gestionnaire de démarrage

Afin de démarrer (*boot*) avec notre noyau, il nous faut un gestionnaire de démarrage (*boot loader*). Deux choix sont possibles :

1. `grub` très populaire, il est capable de démarrer à partir d'un cédérom (contrairement à *lilo*)
2. `isolinux` est le standard en matière de LiveCD, offrant une interface plus conviviale

³La configuration étant une opération délicate, nous garderons les objets liés à la compilation, au cas où il faudrait reconfigurer et recompiler le noyau plus tard.

Grub

L'installation de grub est simple.

```
# emerge grub
```

Le paramétrage de grub s'effectue via le fichier `/boot/grub/menu.lst`. Avec un éditeur (`nano menu.lst`), modifions la configuration comme ceci :

```
default 0
timeout 10
title=Demarrage LiveCD
root (cd)
kernel (cd)/boot/kernel-genkernel-x86-2.6.15-gentoo-r1 root=/dev/ram0 init=/linuxrc \
        looptype=squashfs loop=/livecd.squashfs udev nodevfs cdroot dodmraid
initrd (cd)/boot/initramfs-genkernel-x86-2.6.15-gentoo-r1
```

ISOLinux

Les fichiers d'`textsfisolinux` s'intègrent en-dehors du système, ceux-ci devant être placé à la racine du média. Si vous souhaitez utiliser `textsfisolinux` :

```
# emerge syslinux
```

L'étape de configuration sera décrite plus loin dans ce document.

5 Création de l'image du LiveDVD

5.1 Sauvegarde et nettoyage

Le système est prêt. Nous pouvons quitter l'environnement *chroot* pour retrouver notre système hôte : `exit`.

Copie du système

Par mesure de précaution, nous ne créerons pas l'image SquashFS à partir du système de base, mais à partir d'une copie. En effet, nous allons supprimer les fichiers inutiles à notre LiveDVD (pour économiser de l'espace). Pour recopier le système vers le répertoire `final` :

```
/livecd # mkdir final
/livecd # cp -a source/boot final/
/livecd # mkdir -p /livecd/final/source
/livecd # cp -p -R -P -d /livecd/source/ /livecd/final
```

Arbre portage

L'arbre *portage* n'aura aucune utilité sur notre LiveDVD. Pour l'effacer :

```
/livecd # rm -rf final/usr/portage
```

Fichiers temporaires

Il est judicieux de supprimer les fichiers temporaires créés par les différents programmes utilisés.

```
/livecd # cd final/source
/livecd/final/source # rm -rf var/tmp/* var/run/* var/lock/* var/cache/* tmp/*
/livecd/final/source # rm -f etc/mtab
/livecd/final/source # touch etc/mtab
/livecd/final/source # rm -rf var/log var/lib var/db
/livecd/final/source # mkdir -p var/log var/lib var/lib/dhcp
```

5.2 Création de l'image Squash

Maintenant que notre système est propre, nous pouvons créer l'image au format SquashFS.

```
/livecd/final # mksquashfs source/ /livecd/final/livecd.squashfs
/livecd/final # touch /livecd/final/livecd
```

Cette étape prend du temps, donc c'est le moment ou jamais d'aller à la machine à café. Lorsque l'image est créée, nous pouvons supprimer la copie :

```
/livecd/final # rm -rf final/source
```

Dans le répertoire `/livecd/final`, nous avons le répertoire `boot` l'image compressée `livecd.squashfs` et le fichier `livecd`.

5.3 Configuration d'ISOLinux

Si `grub` a été privilégié à `isolinux`, vous pouvez ignorer cette section.

Le fichier nécessaire est recopié à partir du système du LiveDVD :

```
/livecd/final # mkdir isolinux
/livecd/final # cp ../source/usr/lib/syslinux/isolinux.bin isolinux/
```

La configuration se fait via la fichier `isolinux.cfg` :

```
prompt 1
default 0
label 0
kernel /boot/kernel-2.6.16-gentoo
append root=/dev/ram0 initrd=/boot/initramfs-2.6.16-gentoo init=/linuxrc \
        looptype=squashfs loop=/livecd.squashfs udev nodevfs cdroot dodmraid
```

Remarque Notons qu'il est indispensable de copier ces deux fichiers dans un répertoire nommé isolinux et placé à la racine du média.

5.4 Création de l'image ISO

Tous les fichiers nécessaires au LiveDVD sont prêts. Il est tout à fait possible de rajouter d'autres fichiers et répertoires dans `/livecd/final`. Ils seront intégrés à l'image ISO et disponible à la racine du média.

Pour créer l'image ISO, utilisons la commande `mkisofs` :

– pour utiliser `grub`

```
/livecd # mkisofs -R -b boot/grub/stage2_eltorito -no-emul-boot -boot-load-size 4 \  
-boot-info-table -iso-level 4 -hide-rr-moved -c boot.catalog \  
-o livedvd.iso final/
```

– pour utiliser `isolinux`

```
/livecd # mkisofs -R -b isolinux/isolinux.bin -no-emul-boot -boot-load-size 4 \  
-boot-info-table -iso-level 4 -hide-rr-moved -c boot.catalog \  
-o livedvd.iso final/
```

À la fin de cette commande, l'image `/livecd/livedvd.iso` est prête. Il ne reste plus qu'à la graver avec le programme adéquate.

6 Utilisation

Quand l'image ISO est finalisée, vous pouvez la graver et l'utiliser pour démarrer un ordinateur. Cependant, quelques remarques sont à prendre en considération pour éviter toutes mauvaises surprises.

Horloge Il est plus que probable que le système ne soit pas à l'heure. Pour remédier à ceci, il est possible d'utiliser le client *NTP* si vous disposez d'une connexion à l'Internet.

```
# ntpdate -u pool.ntp.org
```

```
17 May 16:58:59 ntpdate[6027]: adjust time server 193.40.133.142 offset 0.000616 sec
```

L'horloge sera mise à jour en fonction du (des) serveur(s) d'horloge spécifié(s). Il est également possible d'utiliser la commande *date*.

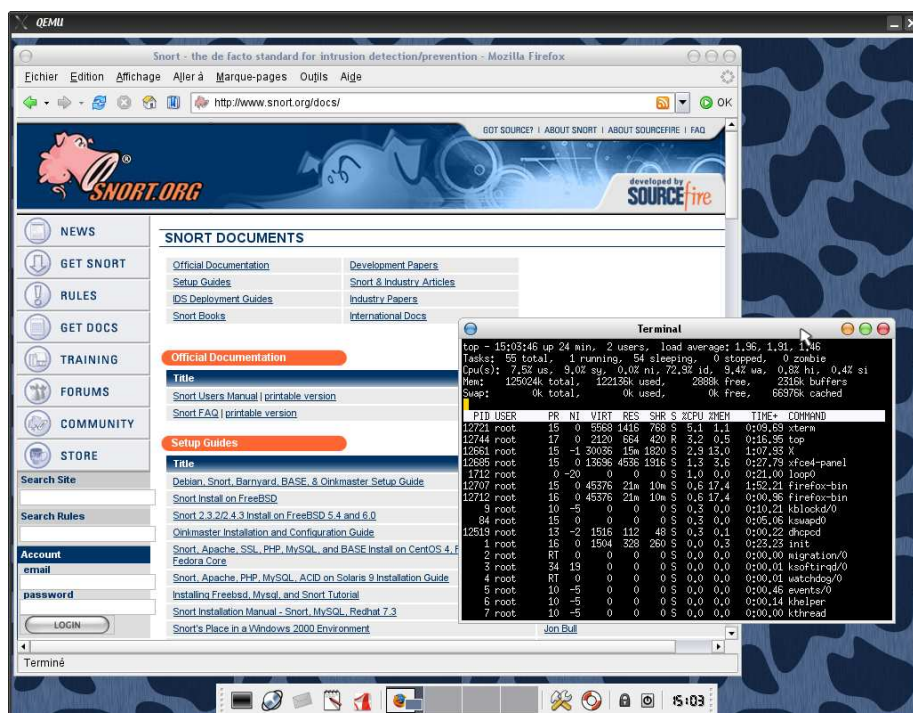


FIG. 1 – Session graphique avec XFCE4 à partir du LiveDVD

Références

- [1] *Gentoo Linux Genkernel Guide*
<http://www.gentoo.org/doc/en/genkernel.xml>
- [2] *Gentoo 2006.0 Handbook*
<http://www.gentoo.org/doc/en/handbook/2006.0/index.xml>
- [3] *A Portage Introduction*
<http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=2&chap=1>
- [4] *Gentoo-Wiki, HOWTO build a LiveCD from scratch*
http://gentoo-wiki.com/HOWTO_build_a_LiveCD_from_scratch
- [5] *Gentoo-Wiki, Gentoo Linux LiveCD*
http://gentoo-wiki.com/TIP_Some_LiveCD_related_tips
- [6] *Gentoo Forums, Gentoo Linux Livecd*
<http://forums.gentoo.org/viewtopic-t-410389.html>
- [7] *ISOLINUX, SYSLinux for CD-ROMs*
<http://syslinux.zytor.com/iso.php>
- [8] *ISOLINUX, Howto for newbies*
<http://stud3.tuwien.ac.at/~e0227529/ILpart1.htm>
- [9] *Miroirs Gentoo Linux*
<http://www.gentoo.org/main/en/mirrors.xml>
<ftp://mir1.ovh.net>
<ftp://ftp.free.fr/mirrors/ftp.gentoo.org>