Slackware Live Edition

Préface

Bienvenue auprès de la distribution Slackware Live Edition! C'est une version de la Slackware-current (qui est sur le point de devenir Slackware 14.2 – estimation écrite fin juin 2016 –), qu'on peut utiliser depuis un DVD ou une clé USB. Il s'agit d'une image ISO conçue pour bien montrer ce qu'est la Slackware. Vous disposez de l'installation par défaut, sans personnalisation des paquets ni du noyau, mais avec toute la puissance Slackware. L'image ISO est créée à partir de zéro par les scripts “liveslak” en utilisant un site miroir de paquets Slackware.

Slackware Live Edition n'a pas à être installée sur le disque dur d'un ordinateur. Vous pouvez vous déplacer avec la version sur clé USB dans votre poche. Vous aurez un système d'exploitation Slackware préconfiguré, à jour et qui tourne tout de suite, partout où vous disposerez d'un ordinateur avec un port USB.

La version USB est “persistante” - c'est à dire que l' OS enregistre votre travail sur la clé USB. Les versions CD/DVD (et la clé USB si vous la configurez en conséquence) agissent sans la persistance de vos données, ce qui signifie que toutes les modifications apportées au système seront perdues au prochain démarrage. Pour protéger vos données privées sensibles en cas de perte de votre clé USB (ou si elle vous est volée) vous pouvez enrichir votre système d'exploitation “Live” et persistant sur clé USB d'un répertoire /home chiffré et/ou d'un fichier chiffré de persistance, à déverrouiller au démarrage par un mot de passe long que vous êtes seul à connaître.

Pourquoi encore une autre Slackware Live

Voici les raisons qui m'ont incité à créer la distribution Slackware Live :

1. Fournir une version 'Live' de la vraie Slackware; c'est à dire montrer Slackware telle qu'elle est, mais sans avoir à l'installer. Sans la cascade de messages émis par le noyau qui défilent sur l'écran au démarrage; pas de papiers peints personnalisés, etc. Conçue à l'intention de l'enseignement, de la découverte et pour démonstration.
2. L'objectif étant la slackware-current, l'avant-garde. Beaucoup de gens voudraient bien savoir à quoi ressemble la version de développement de Slackware mais hésitent à installer slackware-current par crainte qu'elle abîme quelque chose provoquant alors une perte de productivité.
3. Fournir un moyen de créer une image ISO avec simplement un site miroir des paquets Slackware comme source, entièrement dotée des scripts nécesaires et d'une structure déterministe.
4. Garder la possiblité de personnaliser son contenu. Par exemple fournir des versions dépouillées ou minimalistes de Slackware mais également permettre l'inclusion de paquets tiers.
5. Pouvoir choisir la création d'une clé USB bootable qui fera tourner Slackware Live (ce qui n'est pas la même chose que de dupliquer un fichier ISO hybride par 'dd' – disk duplicate – ou par 'cp' –copy– sur une clé USB seulement destinée, de façon classique à installer la distrib qui est dans l'ISO hybrid!)
6. KISS: Keep It Simple Stupid! (Voir la philosophie de la Slackware ;^) ! philosophy)

ISO variées

Les scripts “liveslak” peuvent accommoder la Slackware suivant différentes recettes:

1. une édition Live complète de la Slackware-current 64bit (fichier ISO de 2,6 Go);
2. une ISO allégée XFCE (700 Mo) avec le gestionnaire graphique de connexion XDM. Elle tient sur un CDROM ou sur une clé USB de 1 Go;
3. une image ISO (3,1 Go) de Slackware64-current qui présente Plasma 5 au lieu de KDE 4, et enrichie de plusieurs autres paquets provenant des dépôts d'alienBOB: vlc, libreoffice, calibre, qbittorrent, ffmpeg, chromium, openjdk, veracrypt.
4. une variante Mate(1.7 Go) où KDE 4 a été remplacé par Mate (un fork de Gnome 2);
5. une préparation de Cinnamon (un fork de Gnome 3 remplaçant KDE 4 de Slackware).
6. une variante *personnalisée* à laquelle vous pouvez donner le nom que vous voulez, sa propre liste de paquets et une configuration après installation sur mesure.

Télécharger les images ISO

Serveurs courants disponibles:

- Site principal: http://bear.alienbase.nl/mirrors/slackware-live/ (rsync://bear.alienbase.nl/mirrors/slackware-live/)
- Site de Darren: http://slackware.uk/people/alien-slacklive/ (rsync://slackware.uk/people/alien-slacklive/)
- Site de Willy: http://repo.ukdw.ac.id/slackware-live/
- Site de Ryan: https://seattleslack.ryanpcmcquen.org/mirrors/slackware-live/
- Site de Shasta: http://ftp.slackware.pl/pub/slackware-live/ (rsync://ftp.slackware.pl/slackware-live/)

Documentation de base

Utiliser l'image ISO

Les images ISO sont hybrides, vous pouvez donc ou bien les graver sur un DVD, ou employer 'dd' ou 'cp' pour copier l'ISO sur une clé USB. Les deux méthodes vous donneront un environnement live permettant de faire des changements qui semblent être “écrits sur un disque” tant que votre séance live n'est pas interrompue. Les changements seront en fait conservés dans un RAM disk, et donc un redémarrage remettra à zéro le système de l'OS live dans son état initial. Clairement, il n'y a pas persistance des données.

Slackware Live Edition reconnaît deux comptes d'utilisateur: “root” et “live”. Ils ont leurs mots de passe, et par défaut ceux-ci sont… vous avez deviné: “root” et “live”. Également par défaut, les ISOs démarreront en niveau 4, vous disposerez donc d'un gestionnaire de connexion graphique. Le chargeur d'amorçage vous permet de choisir une autre langue que l'anglais-US et/ou une disposition de clavier et (au démarrage d'un système UEFI) un fuseau horaire personnalisé.

Slackware Live Edition s'écarte aussi peu que possible de la façon de démarrer classique 'à la Slackware'. Une fois franchie l'étape initiale du Liveboot et que votre OS est en place, connectez vous

en tant qu'utilisateur “live”. À partir de maintenant et pour la suite, vous êtes dans un environnement Slackware standard.

Démarrer l'OS Live

Démarrage avec un BIOS

Slackware Live Edition utilise syslinux pour démarrer le noyau Linux sur les ordinateurs avec un BIOS. Pour être précis, la variante “isolinux” est installée dans l'image ISO et la variante “extlinux” est installée dans la partition Linux de la version Live USB.

Syslinux présente un menu graphique de démarrage sur un joli fond inspiré d'un thème Slackware où apparaissent les options:

- Start (SLACKWARE | PLASMA5 | XFCE | MATE) Live (suivant celle des ISOs sur laquelle vous démarrez)
- Non-US Keyboard selection
- Non-US Language selection
- Memory test with memtest86+

Vous pouvez choisir un plan de clavier correspondant à celui de votre ordinateur. Vous pouvez aussi démarrer Slackware dans une autre langue que l'anglais US. Si vous tenez à l'anglais US pour la langue de l'interface, vous voudrez sans doute changer le fuseau horaire car par défaut il correspond au temps universel UTC. Vous devez préciser à la main un fuseau horaire adapté en ajoutant “tz=VotrePartieduMonde/VotreVille” parce que le menu de démarrage de syslinux ne propose pas de choix des fuseaux horaires. Syslinux vous permet d'intervenir en ligne de commande en appuyant sur <TAB>. Appuyez sur <ENTER> pour démarrer après avoir fait les changements ou sur <echap> pour annuler votre commande et revenir au menu.

Démarrage avec UEFI

Sur les ordinateurs avec UEFI, Grub2 gère le démarrage et il présente un menu semblable à celui de Syslinux:

- Start (SLACKWARE | PLASMA5 | XFCE | MATE) Live (suivant sur laquelle des ISOs vous démarrez)
- Non-US Keyboard selection
- Non-US Language selection
- Non-US Timezone selection
- Memory test with memtest86+
- Help on boot parameters

On peut éditer un menu Grub avant le démarrage en appuyant sur la touche “e”. Après avoir fait vos modifications à la commande de démarrage, pressez <F10> pour démarrer. Pour annuler vos changements, appuyez sur <echap>.

Une autre différence entre les menus de Syslinux et ceux de Grub2: dans Grub2 vous pouvez choisir un clavier non-US, une langue et/ou un fuseau horaire et vous reviendrez chaque fois au menu principal. Vous devrez encore lancer “Start SLACKWARE Live” pour démarrer l'ordinateur. Dans le menu Syslinux, seule la sélection de clavier vous ramène au menu principal. Toute sélection de

language non-US va par contre démarrer immédiatement votre Slackware Live; sans revenir au menu principal. C'est une limitation de syslinux qui nécessiterait un nombre exponentiellement plus élevé de fichiers pour pouvoir offrir une plus grande variété de choix dans son menu. Grub2 gère des variables grâce auxquelles on peut aisément modifier les caractéristiques des entrées de menu.

Transférer un fichier ISO dans une clé USB

Vous disposez d'un script qui vous permet de mettre le contenu d'une image ISO sur une clé USB, et qui adapte cette ISO en fonction des paramètres du script.

En utilisant ce script, ce que contenait la clé USB sera effacé et elle sera reformatée! Avant d'occasionner le moindre dégât irréparable, le script vous adressera une invite pour vous permettre d'estimer à ce moment là s'il est prudent de continuer.

Ce script, nommé 'iso2usb.sh', admet les paramètres suivants:

```
  -c|--crypt size|perc       Add a LUKS encrypted /home ; parameter is the
                             requested size of the container in kB, MB, GB,
                             or as a percentage of free space.
                             Examples: '-c 125M', '-c 1.3G', '-c 20%'.
  -f|--force                 Ignore most warnings (except the back-out).
  -h|--help                  This help.
  -i|--infile <filename>     Full path to the ISO image file.
  -o|--outdev <filename>     The device name of your USB drive.
  -p|--persistence <name>    Custom name of the 'persistence'
directory/file.
  -u|--unattended            Do not ask any questions.
  -v|--verbose               Show verbose messages.
  -w|--wait<number>          Add <number> seconds wait time to initialize
USB.
  -C|--cryptpersistfile size|perc
                             Use a LUKS-encrypted 'persistence' file instead
                             of a directory (for use on FAT filesystem).
  -P|--persistfile           Use an unencrypted 'persistence' file instead
                             of a directory (for use on FAT filesystem).
```

Exemples:

- Crée une version USB de Slackware Live, avec la clé USB reconnue par le système comme '/dev/sdX'. Note - la valeur du paramètre indiquant la sortie est l'adresse de périphérique de la clé, pas celle d'une de ses partitions!

./iso2usb.sh -i ~/download/slackware64-live-14.2.iso -o /dev/sdX

- Crée une 'Live' USB comme ci-dessus, mais cette fois en ajoutant un répertoire /home crypté de 750 Mo, et en même temps augmente le délai d'attente au démarrage jusqu'à 15 secondes (ce qui est utile pour les matériels USB lents qui sans cela échouent er le système Live):

./iso2usb.sh -i slackware64-live-current.iso -o /dev/sdX -c 750M -w 15

- Crée une 'Live' USB avec un /home crypté (en réservant 30% de l'espace libre de la clé pour

/home) et où les données persistantes seront conservées dans un fichier non crypté plutôt que dans un répertoire:

./iso2usb.sh -i slackware64-live-current.iso -o /dev/sdX -c 30% -P

- Crée une Live USB avec à la fois /home et les données persistantes cryptées (l'arborescence persistante aura une taille de 300 Mo):

./iso2usb.sh -i slackware64-live-current.iso -o /dev/sdX -c 30% -C 300M

Vous avez sans doute remarqué que le paramètre “-P” n'accepte pas d'indication de dimension. C'est parce que le fichier de stockage est créé en tant que fichier “peu dense” de dimension nulle au début et qui peut croître progressivement jusqu'à un maximum de 90% de l'espace initialement disponible dans la partition Linux de la clé USB.

Boot de l'OS Live par PXE

Slackware Live Edition peut booter en réseau en employant le protocole PXE dans un réseau NFS. Mettez le contenu de l'ISO par exemple dans un nouveau répertoire nommé `slackware-live` dans le répertoire `/tftproot` de votre serveur TFTP et exportez ce répertoire via NFS. Ensuite ajoutez des lignes comme celle-ci à votre fichier `pxelinux.cfg/default` (en considérant que votre serveur NFS a l'adresse IP `192.168.0.1`):

```
label liveslak
kernel slackware-live/boot/generic
append initrd=slackware-live/boot/initrd.img load_ramdisk=1 prompt_ramdisk=0
rw printk.time=0 kbd=us tz=Europe/Amsterdam locale=us_EN.utf8
nfsroot=192.168.0.1:/tftpboot/slackware-live hostname=pxelive
```

Comme le montre l'exemple ci-dessus on utilise un paramètre de boot `nfsroot` qui permet le démarrage en réseau. Le paramètre indique l'adresse IP de votre serveur NFS et le chemin d'accès vers le fichier exporté dans lequel vous avez extrait l'ISO de Slackware Live. Indication: pour avoir une liste des partages exportés par votre serveur NFS, lancez `showmount -e localhost` sur le serveur NFS.

En fait, on dispose de deux paramètres permettant de gérer correctement le démarrage en réseau. Un second paramètre de démarrage `nic` peut être employé pour définir les caractéristiques de la configuration réseau de l'environnement de votre Live, comme le nom de l'interface réseau, l'adresse IP fixe et ce type d'indications. Dans le cas d'un réseau où un serveur DHCP configure vos clients, le paramètre `nic` ne sera pas nécessaire car Slackware Live Edition trouvera d'elle-même tous les détails.

Syntaxe de ces deux paramètres:

```
nfsroot=ip.ad.dr.ess:/path/to/liveslak
nic=<driver>:<interface>:<dhcp|static>[:ipaddr:netmask[:gateway]]
```

Exemples d'utilisation des deux paramètres de démarrage en réseau:

```
nfsroot=192.168.1.1:/tftproot/slackware-live
```

```
nic=auto:eth0:dhcp
nic=auto:eth0:static:10.0.0.21:24:
nic=:eth1:static:192.168.1.6:255.255.255.248:192.168.1.1
```

Après avoir configuré votre environnement PXE (DHCP, TFTP et les serveurs NFS) correctement en utilisant les informations ci-dessus, démarrez l'un de vos ordinateurs aptes au PXE, interrompez le démarrage et choisissez “network boot” puis saisissez ou sélectionnez le nom approprié (dans l'exemple précédent, ce serait `liveslak`). Vous assisterez alors au chargement du noyau et du fichier initrd qui démarrent dans la foulée, et ainsi l'OS Live est lancé comme s'il fonctionnait depuis un support local.

Si votre serveur DHCP met trop de temps à répondre aux demandes du client, le client DHCP est hors-délai et votre OS Live ne peut pas démarrer parce que le système de fichiers Live monté par NFS n'est pas disponible. Dans ce cas vous pouvez essayer d'augmenter le délai d'attente avant que le client DHCP décide qu'il ne recevra plus d'adresse IP venant du serveur. Ajoutez le paramètre de boot `dhcpwait=30` (par exemple) afin que le client DHCP attende une réponse du serveur pendant une durée égale à 30 secondes. Il vous faut bien sûr choisir une valeur suffisamment grande qui conviendra au réglage de votre réseau.
Le délai d'attente par défaut pour DHCP dans l'OS Live est de 20 secondes.

Dans cette configuration la persistance des données n'est pas prise en charge; actuellement, l'overlayfs ne peut pas traiter NFS comme un espace où l'écriture soit possible dans le système 'live'.

Serveur PXE

Slackware Live Edition n'est pas uniquement capable de démarrer un client PXE; elle peut aussi faire fonctionner un serveur PXE entièrement par elle-même.

Ce qui donne?

On peut voir un exemple pratique si à l'occasion d'une réunion “réseau local câblé”, vous venez muni d'une clé USB portant Slackware Live Edition avec laquelle vous démarrez un ordinateur puis tous les autres sur le réseau (câblé), réussissant ainsi à démarrer un réseau dans lequel tourne la même Slackware Live Edition en moins de trois minutes. L'ordinateur avec la clé USB sera le serveur PXE et tous les autres ordinateurs seront ses clients PXE, lisant et utilisant les données Slackware depuis cette clé USB. Les clients hériteront du fuseau horaire du serveur, de la langue et de la configuration de clavier par défaut, ces caractéristiques pouvant être modifiées . Les clients PXE ne disposeront pas de la 'persistance'. Si le serveur a accès à l'Internet, les clients y accéderont eux aussi.

Comment démarrer le serveur PXE?

Quand vous démarrez l'OS Live vous pouvez lancer le script “pxeserver” depuis la console en niveau d'exécution 3 ou depuis un terminal X en niveau 4. Le script réunira toutes les informations nécessaires et s'il ne parvient pas à trouver quelque chose tout seul, il vous le demandera. S'il ne trouve pas l'interface du réseau connecté qu'il doit utiliser, vous pouvez ajouter le nom de votre interface (par exemple, eth1) en tant que simple paramètre du script que vous lancez.

Le serveur PXE utilise dnsmasq pour offrir DNS aux clients PXE. Le programme dnsmasq activera ses fonctions internes de serveur DHCP si votre LAN n'a pas son propre serveur DHCP. Dnsmasq lancera également un serveur TFTP auquel les clients PXE se connecteront pour obtenir les fichiers de démarrage (le noyau et initrd). Le script `pxeserver` démarre aussi un serveur NFS qui permettra à

l'initrd Live d'obtenir les modules squashfs et démarrer l'OS Live. Si votre serveur PXE a plusieurs interfaces réseau, par exemple une interface sans fil reliée au monde extérieur et une interface par câble connectée à un autre ordinateur qui deviendra un client PXE (ou en réalité, connectée à un répartiteur avec toute une série de clients potentiels PXE derrière cela) alors le serveur PXE coordonnera la transmission de paquets afin que les clients PXE puissent accéder au monde extérieur en passant par l'interface câblée pour sortir grâce à l'autre interface.

Si vous avez plusieurs interfaces réseau, il est bon de savoir que dnsmasq n'activera un lien qu'avec la seule interface à laquelle vous voulez que les clients se connectent. Dans une configuration à plusieurs cartes réseau (NIC) où une seconde carte est connectée à l'extérieur (votre réseau local), cela signifie que le serveure DHCP/DNS démarré par dnsmasq ne va pas interférer avec un serveur DHCP existant dans votre réseau local.

Quand le serveur PXE fonctionne, le script va vous montrer le rapport d'activité de dnsmasq dans une fenêtre de dialogue pour vous permettre de gérer les clients PXE qui se connectent.

Si l'ordinateur qui est votre serveur PXE a suffisamment de mémoire RAM, il est vivement recommandé de démarrer l'OS Live du serveur, lancé depuis la clé USB, avec le paramètre `'toram'`. Quand on ne peut plus compter sur les doigts de la main les clients PXE qui doivent lire les fichiers de l'OS depuis le serveur PXE, la clé USB devient un goulot d'étranglement. En faisant tourner l'OS du serveur en RAM on évite ce goulot d'étranglement.

Les paramètres de démarrage expliqués

Appuyez sur <F2> dans l'écran de démarrage de syslinux pour une vue générale de (la plupart) des paramètres de démarrage. Dans l'écran de démarrage de Grub, choisissez l'article de menu “Help on boot parameters” au lieu de démarrer l' OS tout de suite. L'aide de Grub est horrible, je sais, mais Grub n'a pas mieux à offrir.

Les paramètres suivants sont reconnus par Slackware Live Edition. Pour démarrer avec les valeurs par défaut, appuyez simplement sur ENTER.

Environnement du bureau

0|1|2|3|4|5|6|S|s|single ⇒

```
Select a runlevel to start with.
The default is 4 for graphical login.
```

kbd=fr xkb=ch,fr ⇒

```
Example of custom X keyboard layout.
The parameter xkb can be set to "XkbLayout,XkbVariant,XkbOptions".
The boot menus will configure some of these for you but you can
of course always modify the values.
Note that the optional XkbOptions can be several comma-separated values.
The XkbLayout and XkbVariant values must not contain commas.
You can set just the XkbVariant by adding something like "kbd=ch xkb=,fr"
```

locale=nl_NL kbd=nl tz=Europe/Amsterdam ⇒

```
Example of language,
keyboard and/or timezone customization.
```

rootpw="somestring" ⇒

```
Change the password for user "root".
The password is passed as a cleartext string.
```

Logiciels particuliers

load=nvidia ⇒

```
Load and configure Nvidia drivers if available
in the ISO (not for SLACKWARE and XFCE variants by default).
```

load=mod1[,mod2[...]] ⇒

```
Load one or more squashfs modules
from the directory "/liveslak/optional".
By default none of these "optional" modules are loaded on boot.
```

noload=mod1[,mod2[...]] ⇒

```
Prevent loading of one or more
squashfs modules from the directory "/liveslak/addons".
By default all these "addon" modules are loaded on boot.
```

Démarrage par le réseau

dhcpwait=<numseconds> ⇒

```
Maximum wait time for the DHCP client to configure a network interface
(default: 20 seconds).
```

nfsroot=ip.ad.dr.ess:/path/to/liveslak ⇒

```
defines the IP address
of the NFS server, and the path to the extracted content
of Slackware Live Edition.
```

nic=<driver>:<interface>:<dhcp|static>[:ipaddr:netmask[:gateway]] ⇒

```
network device customization, usually this parameter is
not needed when your network runs a DHCP server.
Specify a driver if UDEV does not detect the device. Specify the
```

```
interface if Slackware Live can not figure it out. If you specify
'static' you need to also specify ipaddr and netmask. The gateway
is optional but needed to access the internet for instance.
```

Par rapport au matériel

localhd ⇒

```
initialize RAID/LVM on local hard drives.
```

tweaks=tweak1[,tweak2,[,...]] ⇒

```
Implemented tweaks:
nga - no glamor 2D acceleration, avoids error "EGL_MESA_drm_image required".
tpb - enable TrackPoint scrolling while holding down middle mouse button.
syn - start the syndaemon for better support of Synaptics touchpads.
ssh - start the SSH server (disabled by default).
```

nomodeset ⇒

```
Boot without kernel mode setting, needed with
some machines.
```

rootdelay=10 ⇒

```
Add 10 second delay to give the kernel more
time to initialize USB. Try this if booting fails. Default is 5.
```

swap ⇒

```
Allow the Live OS to activate all swap partitions on
the local hardware. By default, no swap is touched.
```

Réglages fins des supports

hostname=your_custom_hostname[,qualifier] ⇒

```
Specify a custom hostname.  A qualifier 'fixed' can be appended
to prohibit hostname modification in  case of network boot.
```

livemedia=/dev/sdX ⇒

```
Tell the init script which partition
contains the Slackware Live OS you want to boot. This can
become necessary if you have another copy of Slackware Live
installed in another partition.  Also accepted: UUID or LABEL.
```

livemedia=/dev/sdX:/path/to/live.iso ⇒

```
Use this if you want to
load the live OS from an ISO file on a local harddisk partition.
```

livemain=directoryname ⇒

```
Use this if you copied the content
of the ISO to a different directory than "liveslak".
```

luksvol=file1[:/mountpoint1][,file1[:/mountpoint2],...] ⇒

```
Mount LUKS container "file1" at mount point "/mountpoint1"
in the Live fs.  Multiple files must be separated by a comma.
Specify "luksvol=" to *prevent* mounting any LUKS container,
including an encrypted /home .
```

nop ⇒

```
No persistence, i.e. boot the virgin installation in
case your "persistence" directory got corrupted.
If you want to ignore any persistent data during boot,
including LUKS data, specify "nop luksvol=" .
```

persistence=name ⇒

```
Use this if you are using a different
directory/file than "persistence" for storing persistent data.
```

toram ⇒

```
copy the OS from the media to to RAM before running it.
You can remove the boot media after booting.
```

Dépannage

blacklist=mod1[,mod2[...]] ⇒

```
Add one or more kernel modules
to the kernel blacklist to prevent them from loading, in case
they cause issues during operation.
```

debug ⇒

```
During init, pause at strategic locations while
assembling the overlay filesystem and show mount information.
```

rescue ⇒

```
After initialization, you will be dropped in a
rescue shell to perform lowlevel maintenance.
```

Structure de l'ISO

L' ISO Live contient trois répertoires à la racine de son arborescence de fichiers:

- EFI/
- boot/
- liveslak/

La variante USB avec persistance des données peut avoir un répertoire supplémentaire à sa racine:

- persistence/

- Le répertoire EFI/ contient la configuration de Grub employée quand vous démarrez l'OS Live sur un ordinateur capable de UEFI.
- Le répertoire boot/ contient la configuration de syslinux employée pour le démarrage de Live OS sur un ordinateur qui a un BIOS. Ce répertoire contient aussi le noyau et les fichiers de l'initrd qui servent véritablement au démarrage de l'OS.
- Le répertoire liveslak/ contient tous les modules squashfs qui constituent le système de fichiers de l'OS Live, ainsi que les fichiers qui sont copiés directement à la racine de l'arborescence Live. Il comporte quatre sous-répertoires:
  - addons/ - les modules réunis dans ce répertoire seront toujours ajoutés à l'arborescence Live sauf si vous l'empêchez par le paramètre de démarrage “noload=” ;
  - optional/ - les modules de ce répertoire ne seront pas ajoutés à l'arborescence de l'OS Live sauf si vous en forcez l'ajout avec le paramètre de démarrage “load=” ;
  - system/ - ce répertoire contient tous les modules qui ont été créés par le script “make_slackware_live.sh”. Tous ces modules sont numérotés et l'init script Live utilisera cette numérotation pour réorganiser l'arborescence Live dans exactement le même ordre qu'à leur création initiale.
  - rootcopy/ - par défaut ce répertoire est vide. Tout ce que vous y ajouterez (vous, utilisateur de l'ISO) sera copié très fidèlement -verbatim- par le script d'initialisation ('init script') dans l'arborescence au démarrage de l'OS Live. Vous pouvez par exemple utiliser cette fonctionnalité si vous ne voulez pas créer un module squashfs à part pour les fichiers de configuration de vos applications.

Documentation pour les développeurs

Scripts et outils

make_slackware_live.sh

Le premier script:

Le script “make_slackware_live.sh” produit en sortie un fichier ISO qui contient l'OS Live. Grâce au noyau Linux 4.x et au paquet squashfs-tools de Slackware, la création de l'ISO de Slackware Live ne

nécessite **pas** de (re)compilation du contenu de la Slackware ni l'installation de paquets fournis par des tiers.

On peut subdiviser le déroulement interne du script en plusieurs étapes distinctes. Les étapes de la création de l'ISO Live de la Slackware complète sont les suivantes:

Installation des paquets Slackware

Première étape:

- Le script lit une série de paquets pour la variante Live et installe tous les paquets de cette série dans les sous-répertoires d'une arborescence provisoire.
- Chaque ensemble de paquets Slackware (a, ap, d, … , y) ou liste de paquets (min, xbase, xapbase, …) est installé dans un répertoire 'root' à part.
- Chacun de ces répertoires root est “squashed”, c'est à dire transformé (par squashfs) en un module squashfs distinct. Un tel module est un seul fichier archive contenant sous forme compressée la structure du répertoire des paquets installés.
- Ces fichiers de modules sont ensuite montés en boucle puis groupés dans une seule arborescence de fichiers en lecture seule en employant un “montage par recouvrement”. L'overlayfs est relativement nouveau; les premières distributions Live ont utilisé aufs et unionfs pour parvenir à une fonctionnalité comparable, mais on ne pouvait trouver ces systèmes de fichiers dans aucune livraison standard des sources du noyau et par conséquent il fallait compiler des noyaux adaptés pour ce type de distribs Live.
- C'est cet “assemblage par recouvrement” qui est le système de fichiers démarré en tant que système de fichiers de l'OS Live.
- Par dessus ces séries d'arborescences en lecture seule et superposées, un système de fichiers inscriptible est ajouté par le script “make_slackware_live.sh”. Ce répertoire dans lequel on peut écrire sert à garder tous les réglages particuliers faits à notre distrib Live qui doivent s'appliquer quand la Slackware Live démarre. Voir la section suivante pour plus de précisions.
- Vous noterez que lors des démarrages suivants, un autre espace inscriptible sera utilisé pour recevoir les opérations d'écritures effectuées par l'OS. Vous percevrez l'OS Live comme un 'véritable' OS qui peut écrire et supprimer des fichiers. Ce système de fichiers inscriptible sera:
    - une arborescence de fichiers en RAM quand on active l'OS Live sans la persistance.
    - un répertoire ou un fichier conteneur monté en boucle dans votre clé USB, si vous utilisez une clé USB avec l'option persistance paramétrée.

Réglages d'origine par défaut, bien utiles dans la structure Live

Deuxième étape:

- Quelques éléments d'initialisation du système de fichiers sont traités quand la pile de modules a été assemblée:
    - les comptes d'utilisateur 'root' et 'live' sont créés,
    - un environnement initial pour ces comptes est configuré,
    - l'environnement du bureau est configuré pour la première utilisation,
    - les scripts de liveslak “makemod” et “iso2usb.sh” sont copiés vers “/usr/local/sbin/” dans l'ISO, pour votre commodité,
    - si le système Live contient un noyau complet (cas des différentes ISO fournies, sauf XFCE) alors le script “setup2hd” et les fichiers de l'installateur Slackware sont copiés

respectivement dans “/usr/local/sbin” et “/usr/share/liveslak”,
    - slackpkg est configuré,
    - une base de données 'locate' est créée,
    - etc...
- Toutes ces modifications sont écrites dans le système de fichiers inscriptible créé dans la section précédente. Ce système de fichiers sera également enregistré dans l'ISO en tant que module squashfs et au démarrage de l'OS Live, il sera monté en lecture seule exactement comme tous les autres modules. Il aura le nom “0099-slackware_zzzconf-current-x86_64.sxz” ou de manière plus générique “0099-slackware_zzzconf-${SLACKVERSION}-${ARCH}.sxz”

Configuration de l'étape de démarrage de l'OS Live

Troisième étape:

- un répertoire initrd est produit, qui contient un script “init” modifié (les autres distribs Live l'appellent “script linuxrc”) lequel réassemble le système de fichiers de la couche supérieure au moment du démarrage et configure l'OS Live suivant les paramètres de démarrage (langue, clavier, fuseau horaire, niveau d'exécution,...)
- la référence à un noyau générique slackware plus l'initrd indiqué ci-dessus sont ajoutés aux configurations de syslinux (pour les ordinateurs avec BIOS) et de grub2 (pour les machines employant UEFI).

Création de l'image ISO

Quatrième étape:

- un fichier ISO amorçable est créé au moyen de mkisofs.
- la commande “isohybrid” est activée sur l'ISO pour vous permettre d'effectuer “dd” ou “cp” de cette ISO vers une clé USB créant ainsi un support USB capable de démarrer (bootable).

C'est fait! Vous avez le fichier ISO et sa somme de contrôle MD5 dans le répertoire /tmp.

iso2usb.sh

Le second script:

L'emploi du script “iso2usb.sh” en phase d'exécution est expliqué en détail dans un paragraphe précédent “Transférer un fichier ISO dans une clé USB”.

Cette section explique comment le script modifie l'ISO pour la perfectionner par la fonctionnalité USB.

Montage d'un système de fichiers dans un conteneur crypté

Le script créera un fichier de dimension suffisante à la racine de la partition Live en faisant appel à la commande 'dd'. La commande 'cryptsetup luksCreate' va initialiser le cryptage, et donc le script vous demande la confirmation: “are you sure, type uppercase YES” après laquelle il faudra saisir trois fois une phrase de contrôle (deux pour l'initialisation, et une pour ouvrir le conteneur). Si le conteneur est

utilisé pour un /home crypté, son nom de fichier sera “slhome.img”. Le script copiera le contenu du /home de l'ISO vers l'arborescence du conteneur. C'est de là que ce contenu de /home sera ensuite monté en tête du /home de l'ISO (masquant de ce fait le répertoire /home existant). L'OS Live est organisé pour décrypter le conteneur et monter le système de fichiers. Ceci est obtenu en éditant le fichier “/luksdev” dans l'initrd et en y ajoutant la ligne: “/slhome.img”. Le script iso2usb.sh a seulement pour rôle de créer et configurer un /home crypté, mais vous pouvez créer des conteneurs cryptés complémentaires vous-même et les monter à d'autres endroits de l'ISO. Pour obtenir cela, il vous faut éditer le fichier “/luksdev” et ajouter la ligne: “/votre/conteneur.img:/votre/pointdemontage” c.à d. sur une seule ligne, les chemins d'accès au conteneur et au répertoire cible pour votre montage, séparés par 'deux-points'. Le script d'init Live créera le point de montage s'il n'y en a pas.

Utilisation d'un fichier conteneur pour stocker des données permanentes

Un second type de conteneur crypté existe, qui peut servir à stocker vos données permanentes. Le script d'init de l'OS Live vérifie ce qui permettra d'activer la persistance dans l'ordre suivant:

1. le système de fichiers du support USB est-il inscriptible? Si oui,
    1. y a-t-il un répertoire /persistence? Si c'est le cas, on l'utilise; si non,
    2. y a-t-il un fichier /persistence.img? Si oui, monter le fichier et si c'est un conteneur crypté, demander une phrase de contrôle pendant le démarrage.

Ajouter un délai d'attente pour périphérique USB

Pour les supports USB lents, le délai d'attente au démarrage de 5 secondes par défaut est quelquefois insuffisant pour permettre au noyau de détecter les partitions sur votre matériel USB. À titre d'option le script peut augmenter le délai d'attente. Ce qui se fait en éditant le fichier “wait-for-root” dans l'initrd et en modifiant la valeur enregistrée ici (par défaut le script “make_slackware_live.sh” y a inscrit “5”).

makemod

Le troisième script:

Le script “makemod” vous permet de créer facilement un module Slackware Live avec un paquet Slackware ou avec un répertoire complet comme paramètre en entrée.

Usage:

```
# makemod <nomdupaquet|répertoire> nomdumodule.sxz
```

- Le premier paramètre est ou bien le chemin complet vers un paquet Slackware , ou bien un répertoire.
    - Si un nom de paquet est donné en premier paramètre, ce paquet sera installé dans un répertoire temporaire en utilisant l'utilitaire “installpkg” de Slackware. Le contenu du répertoire temporaire sera mis au format 'squash' par le programme “squashfs”.
    - Si c'est un nom de répertoire qui est donné, son contenu deviendra un module au format squash par l'action du programme “squashfs”.
- Le second paramètre est le chemin d'accès complet du module créé en sortie.

Vous pouvez copier le module que vous venez de créer (en observant les conventions de nommage pour un module Live de Slackware, voir le paragraphe “format des modules Live Slackware”) plaçant la copie ou bien dans le répertoire optionnel (optional/) ou dans le répertoire addon/ de votre OS Live. Si vous le copiez dans les emplacements optional/ ou addon/ des sources de liveslak, alors “make_slackware_live.sh” utilisera le module au moment de la création de l'image ISO.

setup2hd

Le quatrième script:

Le script “setup2hd” vous permet d'installer votre OS Live sur le disque dur de votre ordinateur local. Ce “setup2hd” est l'installateur Slackware modifié, vous retrouverez donc vos repères dans la procédure. Il n'y a pas de choix des'SOURCES' à faire car le script sait où trouver les modules squashfs. Une fois que vous avez choisi les partitions cibles, chaque module actif de la version de votre OS Live (SLACKWARE, PLASMA5, MATE, ...) est extrait vers le disque dur. Quand l'extraction est terminée, le script récapitule et détermine combien de modules ont été extraits. Il donne également un exemple de commande pour extraire à la main les modules qui seraient restés inactifs ou désactivés. L'étape finale de l'installation est encore l'installateur Slackware d'origine qui donne le coup d'envoi aux scripts de configuration Slackware.

pxeserver

Le cinquième script:

Le script `pxeserver` intervient comme ceci:

- Il a besoin d'un réseau câblé; le démarrage PXE par wifi est impossible.
- Le script pxeserver recherche une interface câblée; vous pouvez lui donner en paramètre un nom précis d'interface (optionnel).
- Si plusieurs interfaces câblées sont détectées, un dialogue demande à l'utilisateur d'indiquer laquelle il choisit.
- Une vérification est faite pour voir s'il y a une configuration DHCP de cette interface câblée;
  - Si la configuration de DHCP est trouvée alors pxeserver ne lancera pas son propre serveur DHCP et il utilisera à la place le serveur DHCP de votre réseau local (LAN).
  - Si aucun DHCP config n'est trouvé, le script demandera la permission de démarrer son propre serveur DHCP interne. De plus l'utilisateur sera invité à donner une adresse IP pour l'interface réseau et une plage de numéros IP pour le serveur interne DHCP.
- Le script démarrera ensuite le serveur PXE, constitué de:
  - dnsmasq fournissant DNS, DHCP et BOOTP;
  - les démons NFS et RPC;
- Le script détectera si vous avez une connexion vers un réseau externe sur une autre interface et il activera la redirection de paquets IP pour que les clients PXE aient accès au réseau:
- L'OS Live démarré par pxelinux est configuré avec des paramètres de démarrage complémentaires:

```
nfsroot=<server_ip_address>:/mnt/livemedia
luksvol=
nop
hostname=<distroname>
```

```
tz=<server_timezone>
locale=<server_locale>
kbd=<server_kbd_layout>
```

Montrant que la configuration de l'OS Live en relation avec le matériel où le serveur PXE tourne, détermine grandement la configuration des clients PXE.

- Notez qu'à l'occasion d'un démarrage par le réseau le nom d'hôte de l'OS Live reçoit en suffixe l'adresse MAC de chaque ordinateur pour obtenir que le nom d'hôte de chaque machine démarrée par le réseau soit unique.

Créer une image ISO Live à partir de zéro

La création d'une image ISO de la 'Slackware Live Edition' nécessite que vous utilisiez Slackware 14.2 (64-bit). Les versions plus anciennes de Slackware ont un noyau trop ancien pour permettre à la liveslack d'employer la fonctionnalité “overlayfs” récemment intégrée au noyau, et il leur manque les outils squashfs. De plus, on ne peut créer une version Live de la Slackwareque que pour Slackware 14.2 ou plus récente.

Il vous faut aussi l'ensemble des scripts “liveslak” téléchargeables depuis l'un ou l'autre des liens indiqués au bas de cette page.

Liveslak est une arborescence qui contient des scripts, des bitmaps et des fichiers de configuration. Seulement 5 scripts seront utilisés par vous, l'utilisateur. TCes scripts (“make_slackware_live.sh”, “iso2usb.sh”, “makemod”, “setup2hd” et “pxeserver”) sont expliqués plus en détail dans la section “Scripts et outils” ci-dessus. Pour créer une image ISO Live à partir de zéro, il vous suffit d'utiliser le script “make_slackware_live.sh”.

Structure des sources de Liveslak

Le répertoire principal 'liveslak' contient les sous-répertoires suivants:

- EFI/ - contient la structure permettant de démarrer sur les ordinateurs avec UEFI. Plusieurs fichiers de configuration d'UEFI sont créés dynamiquement par le script “make_slackware_live.sh”.
- README.txt - la présente documentation.
- addons/ - les modules squashfs placés dans ce répertoire seront chargés dans l'arborescence Live au démarrage de l'OS.
- graphics/ - les modules squashfs pour la gestion par du code propriétaire de certaines cartes (Nvidia) peuvent être placés ici. Le(s) module(s) seront copié(s) dans addons/ par le script “make_slackware_live.sh” pour chacun des environnements de bureau (sauf la 'pure' Slackware) susceptibles de bénéficier de l'emploi du pilote propriétaire.
- local64/, local/ - ces répertoires peuvent contenir des paquets Slackware considérés 'local' c. à d. ne provenant d'aucun dépôt. Le(s) paquet(s) ser(a- ont) converti(s) en module(s) par le script “make_slackware_live.sh”, copié(s) vers “addons/” dans l'ISO et chargé(s) dans le système de fichiers Live au démarrage de l'OS.
- optional/ - les modules squashfs placés dans ce répertoire ne seront pas automatiquement chargés dans l'arborescence Live au démarrage de l'OS. Pour installer chacun de ces modules, vous devez passer “load=[nom_module]” comme paramètre de démarrage.

- pkglists/ - les fichiers de définition venant de dépôts tiers (*.conf)et les listes de paquets à utiliser issus de ces dépôts (*.lst) doivent être mis dans ce répertoire.
- skel/ - contient des archives compressées (dont les noms doivent correspondre au motif de recherche “skel*.txz”). Ces fichiers seront extraits vers le répertoire “/etc/skel” de l'arborescence Live.
- syslinux/ - contient la structure de gestion du démarrage pour les ordinateurs qui utilisent un BIOS. Plusieurs fichiers de syslinux/ sont générés dynamiquement par le script “make_slackware_live.sh”.
- xdm/ - le thème graphique Slackware pour le gestionnaire graphique de session XDM destiné aux variantes ISO qui ne font pas appel à GDM, KDM ou SDDM.

Le répertoire principal 'liveslak' contient les fichiers suivants:

- blueSW-128px.png , blueSW-64px.png - ce sont les bitmaps du logo Slackware “Blue S”, pour illustrer l’icône utilisateur “live” dans le décor graphique de XDM.
- grub.tpl - le modèle de fichier servant à produire le menu grub pour démarrer avec UEFI.
- iso2usb.sh - ce script crée une variante capable de démarrer depuis un support USB et apte à conserver les données utilisateur à partir d'une ISO Live Slackware.
- languages - ce fichier contient le choix de configurations à adopter pour la gestion de la langue. Une langue par ligne formée des champs suivants: “code:name:kbd:tz:locale:xkb”. Exemple: “nl:nederlands:nl:Europe/Amsterdam:nl_NL.utf8:”
  - code = code de la langue en 2 lettres
  - name = désignation de la langue
  - kbd = nom du plan de clavier en mode console pour cette langue
  - tz = fuseau horaire du pays berceau de la langue
  - locale = locale employée dans ce pays
  - xkb = variante personnalisée optionnelle du clavier dans l'interface graphique en relation avec la langue
- liveinit - c'est le script “init” copié dans l'image initrd pour l'OS Live. L'initrd est avec le noyau générique Slackware, ce qui fait démarrer l'ordinateur. Le script “init” bâtit le système de fichiers Live à partir des modules squashfs de l'arborescence ISO Live.
- make_slackware_live.conf - le fichier de configuration pour le script “make_slackware_live.sh”. C'est là que vous pouvez déterminer bon nombre de paramètres par défaut, ce qui vous évitera de devoir éditer le script lui-même.
- make_slackware_live.sh - le script qui crée l'ISO Live.
- makemod - ce script crée un module squashfs à partir d'un paquet Slackware (ou à partir d'une arborescence).
- menu.tpl - modèle employé pour établir le menu de démarrage syslinux pour les ordinateurs avec BIOS.
- pxeserver - le script qui lance un serveur PXE grâce auquel les autres ordinateurs pourront démarrer Slackware OS Live en réseau.
- setup2hd - le script qui permet d'installer votre Slackware Live sur un disque dur.
- setup2hd.local - dans ce script, le développeur d'un OS Live personnalisé peut modifier les modalités post-installation par défaut en (re-)définissant la fonction “live_post_install()” dans le script `setup2hd`.

Bâtir l'ISO

Dans le système de fichiers liveslack, le script “make_slackware_live.sh” accepte des paramètres en option pour régler finement la création de l'OS Live:

```
The script's parameters are:
 -h                  This help.
 -a arch             Machine architecture (default: x86_64).
                     Use i586 for a 32bit ISO, x86_64 for 64bit.
 -d desktoptype      SLACKWARE (full Slack), KDE4 (basic KDE4),
                     XFCE (basic XFCE), PLASMA5 (full Plasma5 replaces KDE4),
                     MATE (Gnome2 fork replaces KDE4), CINNAMON (fork of
Gnome3
                     Shell replaces KDE4).
 -e                  Use ISO boot-load-size of 32 for computers
                     where the ISO won't boot otherwise (default: 4).
 -f                  Forced re-generation of all squashfs modules,
                     custom configurations and new initrd.img.
 -m pkglst[,pkglst] Add modules defined by pkglists/<pkglst>,...
 -r series[,series] Refresh only one or a few package series.
 -s slackrepo_dir   Directory containing Slackware repository.
 -t <doc|mandoc>    Trim the ISO for size (remove man and/or doc).
 -v                  Show debug/error output.
 -z version         Define your Slackware version (default: current).
 -G                  Generate ISO file from existing directory tree
 -H <hostname>      Hostname of the Live OS (default: darkstar).
 -M                  Add multilib (x86_64 only).
 -O <outfile>       Custom filename for the ISO.
 -R <runlevel>      Runlevel to boot into (default: 4).
 -X                  Use xorriso instead of mkisofs/isohybrid.
```

Le script fait appel à des dépôts de paquets pour créer une ISO Live. Les paquets seront installés dans un répertoire temporaire.

Pour qu'il soit possible de créer une ISO Live de chacune de ces variantes, les dépôts des paquets nécessaires doivent être accessibles localement (cela peut être dans un répertoire monté en réseau). Un miroir local du dépôt Slackware est obligatoire. Les différents paquets disponibles dans un dépôt tiers seront téléchargés à partir d'un serveur distant dans la mesure où une URL rsync pour le dépôt est configurée dans ./pkglists/*.conf.

Quand tous les prérequis seront satisfaits, vous n'aurez à lancer qu'une seule commande pour créer l'ISO. L'exemple suivant permet de créer une 'Live Edition' de la Slackware non modifiée:

```
# ./make_slackware_live.sh
```

Un autre exemple qui crée une variante MATE, et configure le niveau d'exécution à '3' par défaut, définissant aussi un chemin d'accès précis vers la racine du dépôt de paquets Slackware (notez que le script recherchera un sous-répertoire “slackware64-current” dans cette adresse du dépôt si vous créez l'ISO pour slackware64-current):

```
# ./make_slackware_live.sh -d MATE -R 3 -s ~ftp/pub/Slackware
```

Si vous voulez savoir quels groupes de paquets font partie de l'un ou l'autre de ces Environnements de Bureau, lancez la commande suivante:

```
# grep ^SEQ_ make_slackware_live.sh
```

pour MATE, vous obtiendrez:

```
SEQ_MSB="tagfile:a,ap,d,e,f,k,l,n,t,tcl,x,xap,xfce,y pkglist:slackextra,mate
local:slackpkg+"
```

ce qui montre que la plupart des séries de paquets Slackware (sauf kde et kdei) seront installées suivant leurs fichiers d'index, et par dessus le tout, deux listes de paquets seront installées à partir du sous-répertoire pkglists/ , listes qui sont slackextra et mate. Enfin, “slackpkg+” sera installé depuis un répertoire local.

Possibilités de personnalisation de l'OS Live

Vous pouvez créer votre propre OS Live en changeant ses caractéristiques dans le fichier de configuration “`make_slackware_live.conf`”. Parmi ce que vous pouvez changer il y a:

- Le nom de la version de gestionnaire de Bureau (le script lui-même prévoit “*SLACKWARE*”, “*PLASMA5*”, “*XFCE*”, “*MATE*” et “*CINNAMON*”),
- La ou les liste(s) des paquets employés pour votre distribution personnalisée,
- Le nom du compte utilisateur (par défaut c'est “*live*”),
- Le nom de la distribution (par défaut c'est “*slackware*”),
- Et enfin vous pouvez définir une fonction “`custom_config()`” dans laquelle vous pourrez ajouter des éléments personnalisés de post-installation non intégrés au script “`make_slackware_live.sh`” lui-même.

Voici la section de `make_slackware_live.conf` concernée pa ces personnalisations. Deux variables sont nécessaires si vous voulez créer votre propre OS Live selon vos goûts: “**`LIVEDE`**” et “**`SEQ_CUSTOM`**”, le reste est optionnel (mais bien utile cependant):

```
# REQUIRED:
# Define a new name for your own variant of Slackware Live Edition:
#LIVEDE="CINELERRA"

# REQUIRED:
# Define your own custom package sequence for a custom Live ISO.
# In this example you would need to create two files
"pkglists/cinelerra.conf"
# and "pkglists/cinelerra.lst" defining the package location and package
list
# respectively):
#SEQ_CUSTOM="min,xbase,xapbase,xfcebase,cinelerra"

# OPTIONAL:
# Your custom distro name (will reflect in boot screen & filenames):
#DISTRO="cinelerra"

#OPTIONAL:
# Name of the 'live' user account in the Live image:
```

```
#LIVEUID="live"

# OPTIONAL:
# Marker used for finding the Slackware Live files:
#MARKER="CINELERRA"

# OPTIONAL:
# The filesystem label of the ISO:
#MEDIALABEL="CINELERRA"

# OPTIONAL:
# The ISO main directory:
#LIVEMAIN="cinelerra"

# OPTIONAL:
# Add your own Live OS customizations to the function custom_config() :
#custom_config() {
#  # Add your own stuff here which is not covered in the main script:
#}
```

Les rouages de Slackware 'Live Edition'

Overlayfs, squashfs

Overlayfs et squashfs font véritablement de la magie là. Comme expliqué précédemment, le programme squashfs traite une arborescence complète et en fait un unique fichier d'archive. Il le fait d'une manière spéciale, d'avantage à la façon de mkisofs que comme lors de la création d'une archive par tar. Le module ainsi produit peut être monté en boucle pour permettre l'accès au système de fichiers qu'il contient. Quand vous avez plusieurs de ces modules squashfs montés en boucle, chacun contenant une part du système de fichiers de l'OS, vous allez empiler ces systèmes de fichiers modulaires et reconstituer ainsi le système complet (tout à fait comme Tintin fit dans Le Secret de la Licorne quand il superposa plusieurs morceaux de parchemin translucides et les regarda par transparence pour voir l'image complète). Overlayfs est l’instrument 'guide de superposition' des systèmes de fichiers partiels. Overlayfs peut travailler avec un grand nombre d'éléments de systèmes de fichiers accessibles en lecture seule et exactement un système de fichiers inscriptible. Le système inscriptible permet que votre OS Live semble écrire sur un disque dur - les écritures vers le système de fichiers supérieur sont effectuées de fait en RAM (ce qui se produit quand vous avez un vrai OS Live) ou dans une arborescence de 'persistance' qui est conservée quelque part sur un support inscriptible (une clé USB avec 'persistance' en est un exemple).

L'initrd et son script init

L'initrd employé pour la Slackware 'Live Edition' est un initrd Slackware standard créé par la commande Slackware “mkinitrd”, avec une unique modification: son script “init”. À l'heure actuelle la bonne dénomination d'un 'initrd' est 'initramfs', abréviation de “initial ram filesystem” parce qu'il contient le système de fichiers initial chargé dans la zone mémoire du noyau au moment du démarrage de votre ordinateur. C'est le script init d'un initrd qui prépare le système de fichiers racine avant même que l'OS définitif soit actif. Quand l'OS démarre, il trouve un système de fichiers racine

tout prêt à l'emploi. Un cas exemplaire est bien sûr le système de fichiers racine LUKS codé. Un autre exemple est un système de fichiers racine enregistré dans des volumes logiques (LVM). Il faut un travail considérable pour rendre le système de fichiers racine accessible et démarrer le véritable programme “init” de l'OS (PID 1). Ces exemples montrent qu'il faut un script dans un initrd pour bâtir le système de fichiers racine d'un OS Live. Comme l'initrd de Slackware ne gère pas un environnement Live, le script init d'origine a été complété afin d'être efficace dans liveslak.

Que fait le script 'init' version 'liveslak' ?

- Il étudie les paramètres de démarrage que vous avez entrés (ou ceux transmis par syslinux/grub) et essaie de les utiliser de façon logique.
- Il procède à l’initialisation tout comme l'init de Slackware (lancer udev, attendre un peu que les supports physiques USB soient établis, charger les modules du noyau, charger le plan de clavier adapté, initialiser le RAID etc.) avant de passer à ce qui concerne spécifiquement la Slackware Live.
- Un système de fichiers établi en RAM est créé, qui constitue la base de tout le reste.
- En faisant appel à des outils comme'blkid' et 'mount', le script init tente de localiser le support physique Live. Il se sert de blkid pour chercher l'étiquette du système de fichiers par défaut sur le support Live. Si celà échoue (parce que vous avez changé le nom –i.e l'étiquette–) grâce à blkid il va répertorier toutes les partitions système disponibles et les monter une par une jusqu'à ce que la partition Live soit trouvée.
- Quand le support Live est déterminé, l'étape suivante est de monter en boucle les modules squashfs un par un et les ajouter à la couche supérieure de l'arborescence dans le bon ordre. Si vous avez spécifié le paramètre de démarrage 'toram', alors le module sera d’abord copié en RAM avant d'être monté en boucle. Cela vous permet de retirer le support physique qui a permis le démarrage puisque l'OS tournera entièrement en RAM.
- Ordre de chargement des modules:
  - d'abord les modules du système (modules de base dans le sous-répertoire system/)
  - puis les modules complémentaires (dans le répertoire addon/). Si vous ne voulez pas qu'un module complémentaire soit chargé, vous pouvez spécifier “noload=nomdumodule” dans la ligne de commande du démarrage de syslinux/grub.
  - enfin, les modules optionnels (dans le sous-répertoire optional/). Par défaut un module optionnel n'est pas chargé sauf si vous l'imposez en ajoutant “load=nomdumodule” à la ligne de commande du démarrage.
- Ensuite, la persistance pourra être configurée si l'OS a été démarré depuis un support inscriptible tel qu'une clé USB. Le script init cherchera d'abord un répertoire nommé “persistence” à la racine de la partition Live de la clé USB afin de l'utiliser pour stocker les changements à conserver dans le système de fichiers Live. Si ce répertoire n'existe pas et si un fichier “persistence.img” est trouvé, alors c'est ce fichier qui sera monté en boucle et les changements permanents apportés à l'arborescence seront écrits dans ce fichier conteneur. Le conteneur “persistence.img” peut être crypté par LUKS auquel cas le script init vous demandera une phrase secrète pour le déverrouiller avant de le monter.
- La couche supérieure du système de fichiers est alors complétée par l'ajout de la structure de répertoires et fichiers inscriptibles (soit persistants soit volatiles).
- Le système de fichiers global en RAM qui est la base de la couche supérieure est rendu accessible à l'utilisateur de l'OS Live en tant que “/mnt/live”
- Le système de fichiers du support Live est rendu accessible pour l'utilisateur en tant que “/mnt/livemedia”. Si le support est une clé USB vous aurez accès en écriture sur “/mnt/livemedia”.
- Une fois le système de fichiers racine assemblé, l'OS Live est configuré avant même de démarrer:

    - Si vous avez indiqué “swap” en ligne de commande au démarrage, toute partition swap disponible sera ajoutée à “/etc/fstab” dans l'OS Live.
    - Si vous avez indiqué un plan de clavier spécifique pour la console (et peut-être un autre pour X) en employant les paramètres de démarrage “kbd” et “xkb”, alors ils seront configurés dans “/etc/rc.d/rc.keymap” et “/etc/X11/xorg.conf.d/30-keyboard.conf” de l'OS Live.
    - De même pour toute locale spécifique déterminée par le paramètre“locale” , elle sera ajoutée à “/etc/profile.d/lang.sh”.
    - Si le fuseau horaire et le réglage de l'horloge matérielle ont été précisés par le paramètre “tz”, ils seront configurés dans “/etc/localtime” et “/etc/hardwareclock”.
    - Les paramètres de démarrage “livepw” et “rootpw” vous permettent de définir vos mots de passe pour les utilisateurs 'live' et 'root'; par défaut les deux sont simplement 'live' et 'root'. On peut le faire par appel de la commande “chpasswd” dans la couche logicielle isolée (chrooted) afin de pouvoir y saisir un mot de passe en clair.
    - Grâce au paramètre de démarrage “hostname” on peut changer le nom d'hôte dans l'OS Live, qui est “darkstar” par défaut. La configuration est enregistrée dans “/etc/HOSTNAME” et “/etc/NetworkManager/NetworkManager.conf”.
    - Si le paramètre de démarrage “blacklist” a été renseigné, alors les modules du noyau donnés en argument(s) seront ajoutés au fichier qui est la liste noire de modprobe “/etc/modprobe.d/BLACKLIST-live.conf”.
    - Le fichier “/var/lib/alsa/asound.state” de l'OS Live a été enlevé pour permettre de configurer correctement le son sur tout ordinateur utilisé pour démarrer depuis le support physique Live.
    - Tout le contenu du répertoire /liveslak/rootcopy de la partition Live (qui peut être vide) est copié vers la racine de l'arborescence de l'OS Live, en 'écrasant' potentiellement des fichiers dans l'OS Live. Faites appel à /liveslak/rootcopy pour ajouter des personnalisations à votre OS Live quand vous l'utilisez à partir d'une clé USB.
    - Et pour finir, mais c'est très important, tous les fichiers conteneurs cryptés par LUKS sont déverrouillés (init va vous demander la phrase clé) et le(s) système(s) de fichiers qu'ils contiennent est/sont monté(s) dans l'OS Live. Actuellement, un /home crypté par LUKS est géré. Les fichiers “/etc/fstab” et “/etc/crypttab” seront mis à jour afin que l'OS Live fasse le montage et le démontage.
    - Le script init termine en indiquant au noyau de basculer maintenant sur notre nouveau système de fichiers racine (la surcouche logicielle) et de lancer le programme init de Slackware(PID 1, /sbin/init).
- À partir de ce moment et pour la suite, vous démarrez un système Slackware 'normal' et on ne remarque pas qu'il fonctionne de fait en RAM et non pas depuis votre disque dur local.

Format de module Live Slackware

Un module Live Slackware contient une arborescence de répertoires qui a été 'squashed' en un fichier d’archive compressé par le programme “squashfs”. L'algorithme de compression employé est “xz” ce qui explique le choix de l'extension du fichier module: “.sxz” qui signifie “squashed avec xz”.

La Slackware 'Live Edition' est prévue pour que ses modules se conforment à une norme de nommage des fichiers particulièrement souple:

- Le format de nom de fichier est “NNNN-nom_module-*.sxz”, où 'N' est un chiffre et 'nom_module' ne doit pas contenir de tiret '-'.
- La partie “nom_module” est ce que vous devez indiquer dans les paramètres de démarrage

"load" et "noload".
- N'importe quoi peut occuper la place de '*' mais le plus fréquent est "${VERSION}-${ARCH}". Les modules de base de la Slackware Live utilisent le numéro de version Slackware pour ${VERSION} et l'architecture de la Slackware pour ${ARCH}. Pour les modules des sous-répertoires addons/ et optional/, ${VERSION} sera couramment la version du programme qui est disponible dans le module.
- Les quatre chiffres d'un nom_module ont une signification. Certains intervalles sont réservés à l'OS, veuillez donc ne pas les utiliser. Leurs préfixes sont déterminés suivant l'origine du paquet:

```
    0000 = contient le répertoire Slackware /boot
    0010-0019 = paquets installés suivant un fichier d'onglets des
groupes de logiciels Slackware (a,ap,d, ... , séries des jeux en
console y)
    0020-0029 = paquets installés d'après une liste comme on en trouve
dans le sous-répertoire ./pkglists des sources de liveslak  (min,
xbase, xapbase, xfcebase etc)
    0030-0039 = un paquet 'local', i.e. un paquet situé dans ./local ou
./local64 (suivant l'architecture)
    0099 = un module de configuration liveslak (contenant tous les
réglages fins grâce auxquels des paquets installés deviennent un OS
Live utilisable
```

- On reste libre d'utiliser d'autres intervalles. Notez que l'ordre d'empilement des arborescences traitées par squashfs lors de la constitution du système de fichiers de l'OS Live dépend de la numérotation des modules. Donc s'il y a des modules que vous voulez charger dans un ordre spécifique il vous suffit de veiller à ce que les nombres dans leurs noms de fichier soient en ordre croisssant.

1. Exemple d'un module de base: 0099-slackware_zzzconf-14.2-x86_64.sxz
2. Exemple d'un module optionnel: 0060-nvidia-352.79_4.4.1-current-x86_64.sxz

Autres distributions Live basées sur Slackware

Naturellement, il y a eu bien des prédécesseurs avant moi, et comme lors de mes premiers pas sur le terrain de Linux Live j'étais bien ignorant, j'ai beaucoup appris sur le fonctionnement d'une distrib Live en m'exerçant avec ces autres distributions que sont les 'Live' fondées sur Slackware. Permettez moi de les présenter, avec respect:

SLAX

Le site: https://www.slax.org/

SLAX fut la première variante Live de Slackware. Les scripts linux-live qui permettent la création d'une image ISO de SLAX ont été généralisés de sorte qu'ils peuvent créer une version Live de tout OS déjà installé sur un disque dur. Le développement de SLAX était en panne depuis environ deux ans mais son créateur semble redevenir actif depuis peu.

La fonctionnalité Live de SLAX se base sur aufs et unionfs, ce qui nécessite un noyau spécifiquement

compilé pour gérer la gestion du format aufs. Slax est peu volumineuse et ses scripts de démarrage sont finement réglés en vue de la rapidité de lancement.

Porteus

Le site: http://ww.porteus.org/

Porteus a été créée à partir de SLAX par la communauté SLAX quand le développement de SLAX semblait au point mort. Porteus a une communauté d'utilisateurs active au sein de laquelle c'est «tout, autour des modules». L'emploi d'aufs au lieu d'overlayfs permet à Porteus (comme à SLAX) d'ajouter et d'enlever des modules squashfs au système Live en cours de fonctionnement, à la volée, ce qui a provoqué le jaillissement d'un grand nombre de modules développés au sein de la communauté. Il semble que la prochaine génération de Porteus sera basée sur Arch Linux plutôt que sur Slackware: ceci est lié au départ du premier développeur de Porteus.

Salix Live

Le site: http://www.salixos.org/download.html

Salix est une distribution basée sur Slackware, avec sa propre philosophie de «un outil par tâche» qui réduit notablement le nombre de paquets, par comparaison avec sa parente Slackware. Salix a inclus le traitement des dépendances dans son outil de gestion des paquets. Des éditions Live de Salix sont disponibles pour différents environnements, chacune étant construite autour et centrée sur un Environnement de Bureau ou un Gestionnaire de Fenêtres particulier. Des versions Salix-Live sont disponibles pour XFCE ainsi que pour MATE.

Slackel

Le site: http://www.slackel.gr/

Slackel est une distribution Grecque basée à la fois sur Slackware et Salix. Elle est fournie sous trois habillages, pour chacun desquels il y a une version Live: KDE4, Openbox et Fluxbox. Les scripts Live sont ceux de Salix-Live modifiés.

SlackEX

Le site: http://slackex.exton.net/

Site où on voit beaucoup de distribs Linux classiques en version Live. La version SlackEX est fondée sur Slackware de plus ou moins loin, avec un noyau spécifique et plusieurs outils qui ne viennent pas de Slackware elle-même. Je n'ai pas trouvé les sources pour cette distrib Live.

Sources de Liveslak

'Slackware Live Edition' est créée par les scripts 'liveslak' développés et maintenus par Eric Hameleers pseudo Alien BOB alien@slackware.com.

- Dépôt Git: git://bear.alienbase.nl/liveslak.git
- Dépôt Git: (on peut y “fureter”): http://bear.alienbase.nl/cgit/liveslak/
- Serveur miroir de téléchargement: http://www.slackware.com/~alien/liveslak/

Sources

- Original source: http://bear.alienbase.nl/cgit/liveslak/tree/README.txt
- Originally written by Eric Hameleers
- Traduit de l'anglais par — *P-M Averseng*
  slackware, live, author alienbob, translator pierreaverseng