Construire un Live Debian

Encore un tuto sur ce sujet ?? yen a plein sur la toile ...

Oui... mais celui-là, c'est le bon : ce n'est pas une copie d'un wiki ou d'un article.

C'est une **méthode pas à pas** qui vous permettra de comprendre en construisant.

En quelques pages et quelques heures, vous serez à même de construire,

adapter et diffuser un liveCD.

J'ai découvert par tâtonnement, n'ayant aucune formation informatique. Ce tutoriel est là pour vous faire gagner du temps et vous éviter quelques erreurs décourageantes.

Ce tutoriel n'est pas un manuel complet de live-build.

Il demande l'utilisation du terminal et une certaine organisation,

mais est à la portée de tous si vous le suivez pas à pas.

L'exemple qui illustre ce tutoriel utilise les fichiers d'une HandyLinux-v1 sous Wheezy car son arborescence est assez simple, mais ce tutoriel est réalisé, testé et compatible avec **Debian Jessie**.

Bon'build !

arpinux 2014-2016 GPLv3

mise à jour: 11 déc. 2016

Dépendances et premiers tests

liveCD et live-build: une présentation plus que vulgarisée

Le **liveCD** installable est un **système embarqué** utilisable sur CDROM/DVD/USB et installable sur tout support de stockage (clé usb, carteSD, disque dur interne). Il se compose d'un *squashfs* qui contient l'intégralité du système, un *bootloader* pour pouvoir démarrer et accessoirement un *installeur* (ce sera la cas dans notre tuto).

Le bootloader démarre et lance la décompression du **squashfs** afin de le rendre utilisable. Dès lors, le CDROM inséré se comporte comme un système d'exploitation classique.

Live-build est un programme qui permet de créer un live à partir d'un système développé en **chroot** (le système dans le système chroot=*change root*).

Installation des outils

J'utilise live-build, l'outil de construction officiel des images Debian comme outil principal. Il est disponible dans les dépôts (*tree* n'est là que pour l'exemple) :

```
sudo apt-get install live-build live-manual live-tools tree
```

Une fois fait, vous pouvez créer votre dossier de construction (build) et lancer les premiers tests

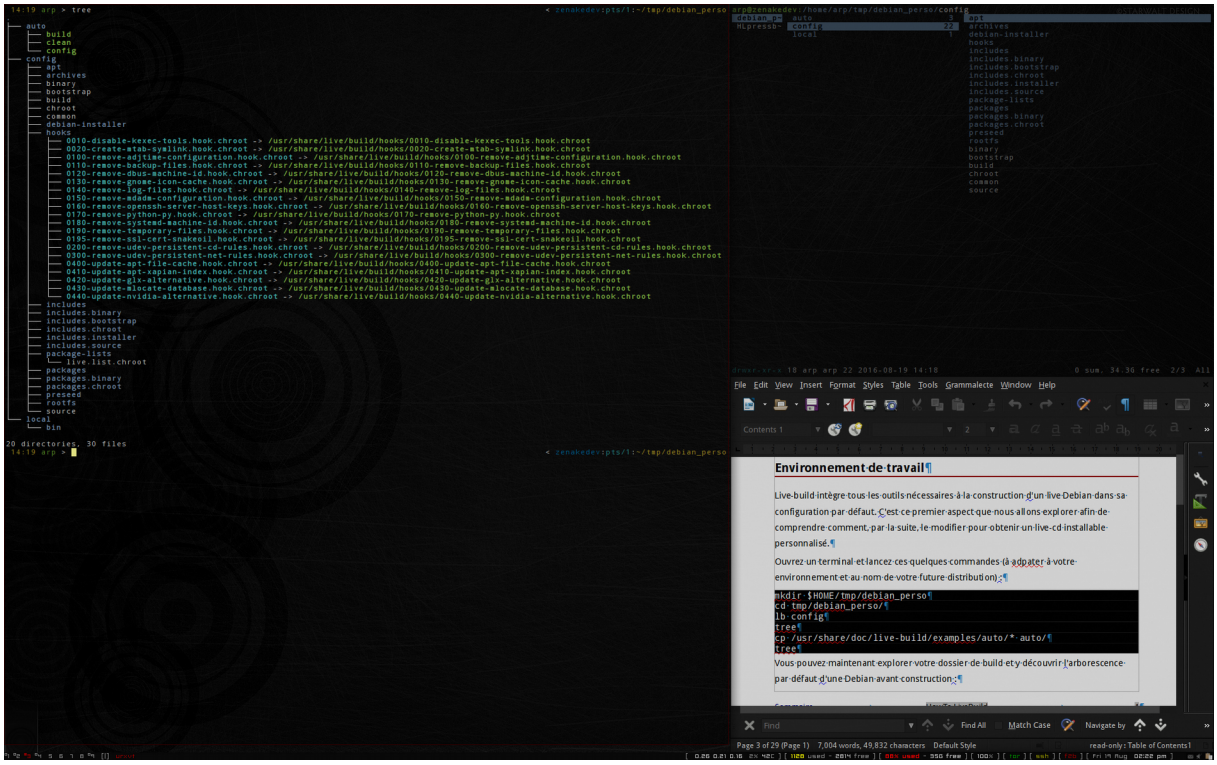
Environnement de travail

Live-build intègre tous les outils nécessaires à la construction d'un live Debian dans sa configuration par défaut. C'est ce premier aspect que nous allons explorer afin de comprendre comment, par la suite, le modifier pour obtenir un liveCD installable personnalisé.

Ouvrez un terminal et lancez ces quelques commandes (à adpater à votre environnement et au nom de votre future distribution) :

```
mkdir -p $HOME/tmp/debian_perso
cd tmp/debian_perso/
lb config
tree
cp /usr/share/doc/live-build/examples/auto/* auto/
tree
```

Vous pouvez maintenant explorer votre dossier de build et y découvrir l'arborescence par défaut d'une Debian avant construction :



Pour une liste complète des fonctions de chaque dossier, je vous renvoi au manuel de live-build. Je ne l'ai pas lu en entier. j'avoue que je préfère comprendre pour apprendre.

Mais comment comprendre quand on l'a jamais fait ? sans lire un manuel ?

bah en faisant 😉 .

Premier test à vide

Pour un premier test du live-build, je vous conseille vivement de construire un live Debian avec sa configuration par défaut. Cela vous permettra de voir tout de suite ce qui est modifiable et comment le faire.

Toujours dans votre terminal et dans votre dossier de build, lancez la commande

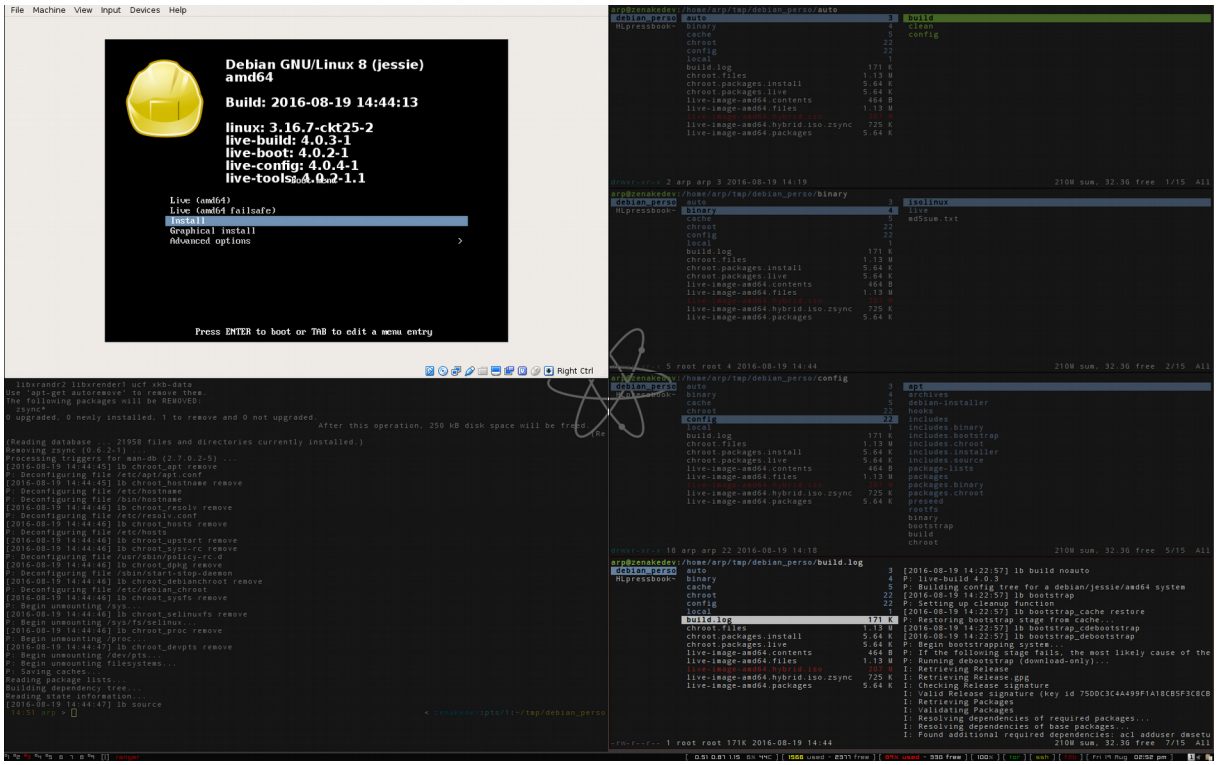
```
sudo lb build
```

Vous pouvez aller boire un thé tranquille, le processus va construire une arborescence système avec *debootstrap*, récupérer les paquets essentiels, installer et configurer les paquets, puis construire le *squashfs*. Selon votre machine et la qualité de votre réseau, cela peut prendre entre 20 minutes et 1heure. Tout le déroulement sera visible sur le terminal et dans le fichier de log créé dans debian_perso/build.log.

Oki, j'ai bu un thé, lu un magazine, réparé la chaise de jardin et .. ayest, c'est fini. Mais j'ai quoi en fait maintenant ?

Et bien vous venez de construire une Debian live netinstall i386 ou amd64 selon l'architecture de votre machine 😛 !

Si vous lancez l'image «live-image-amd64.hybrid.iso» (peut être 'i386' chez vous) dans une vbox, vous pouvez lancer un live sans server X ou installer une Debian.



Dossiers et fichiers présents dans votre dossier de build.

Maintenant que tout est là, on peut comprendre comment c'est gaulé le truc... :

Les dossiers du liveCD

- **auto/build** : le script de construction qui lit le script de config
- **auto/clean** : le script de nettoyage du dossier de build
- **auto/config** : le script principal définissant les options de constructions
- **binary/isolinux** : le dossier de configuration du bootloader syslinux. Vous comprendrez l'organisation des différents fichiers appelés par syslinux au moment du boot
- **binary/live** : le dossier du live proprement dit qui contient le *squashfs* et l'image de démarrage *initrd.img* ainsi que le *vmlinuz* appelé par le *bootloader*.
- **cache** : comme son nom l'indique, c'est le cache du live-build, avec les paquets nécessaires à la construction ainsi que le *bootstrap* de base
- **chroot** : le fameux sous-système dans lequel tout votre liveCD est construit. C'est ce chroot qui deviendra le *squashfs* compressé dans /binary/live
- **config** : le super dossier dans lequel toute la personnalisation va se jouer. Pour le moment, tous les dossiers sont vides (sauf le *hook* qui contient les liens des scripts

prévus par défaut pour live-build et le *package-lists* qui contient la liste minimale des paquets pour le live), mais vous pouvez déjà regarder les fichiers *binary*, *bootstrap*, *common* et *source* qui contiennent les options par défaut de la construction du live. Ces options sont listées dans "man lb_config" et seront modifiées par le script auto/config.

- **local** : jamais utilisé, non documenté dans le manuel.

Les fichiers importants du liveCD

- **build.log** : votre fichier de log qui vous raconte l'histoire de la construction
- **chroot.packages.install** : les paquets prévus pour être installés
- **chroot.packages.live** : les paquets présents sur le liveCD
- **live-image-amd64.contents** : le contenu du liveCD
- **live-image-amd64.files** : arborescence / droits / permissions du contenu du liveCD (un gros ls -l)
- **live-image-amd64.hybrid.iso** : votre liveCD installable en format hybrid pour une utilisation sur cd ou usb
- **live-image-amd64.packages** : les paquets installés dans le liveCD

Prenez le temps de consulter ces fichiers avec le "man lb_config" à côté pour explorer les différentes options proposées. Familiarisez-vous avec cette architecture et le contenu des dossiers du build.

Je vous laisse découvrir tout ça tranquille. Ne négligez pas cette étape qui vous permettra d'aborder la personnalisation beaucoup plus sereinement.

0ptions générales et choix des paquets

On va analyser le script des options principales auto/config ainsi que les différentes méthodes pour intégrer des paquets au liveCD.

En gros, si vous lancez un test à la fin de ce chapitre, le iveCD obtenu intégrera toutes les applications que vous désirez. Il restera la personnalisation et pour finir les derniers tests et l'utilisation finale.

En avant pour les options acceptées par live-build... ou *comment débuter la personnalisation de votre liveCD*

auto/config : le script de construction

Si vous avez suivi les étapes, ce script existe déjà. il est là: *debian_perso/auto/config*.

Ce script va contenir les arguments que vous pourriez passer à la commande *lb config* lors de la construction (oui, ces scripts sont là pour vous faciliter la vie, mais tout peut être passé en argument dans une looooooongue commande live-build.). Pour obtenir la liste des arguments acceptés et leurs options, direction le man (bah oui, il en faut un peu) et un "man lb_config" vous affichera les possibilités.

ayest ? vous avez lancé la commande ? yen a trop pour vous ? normal... c'est prévu pour couvrir toutes les possibilités et architectures (Debian quoi...) mais on va se concentrer sur ce qui va le plus visiblement modifier votre futur liveCD:

Les options principales

--apt-recommends true|false : considérer les paquets recommandés comme des dépendances. Cette option sur *false* permet d'alléger le système mais vous oblige à plus de vigilance dans le choix des paquets: c'est à vous de prévoir les paquets recommandés nécessaires et les ajouter à la liste.

--architectures ARCHITECTURE : i386 ou amd64. Par défaut, live-build construira une version avec la même architecture que celle de votre système.

--binary-images iso|iso-hybrid|netboot|tar|hdd : type d'image iso distribuée. Ce choix dépend de l'utilisation que vous désirez faire de votre liveCD. Pour une utilisation sur CD/DVD/USB, c'est *iso-hybrid* qu'il vous faut.

--bootloader grub|syslinux : le système de démarrage utilisé pour le liveCD. Par défaut, c'est syslinux.

--clean : permet de supprimer les dossiers vide (donc inutiles) du dossier de build

--debian-installer true|cdrom|netinst|netboot|businesscard|live|false : choisir si le live sera installable, et si oui, choisir le type de l'installeur. Pour installer le système personnalisé depuis le liveCD, il faut choisir *live*.

--debian-installer-gui true|false : avctiver ou non l'installation en mode graphique (avec la bannière et les jolies fenêtres).

--distribution CODENAME : la distribution Debian sur laquelle se base votre dérivée. Pour nous, ce sera *jessie*.

--iso-application NAME : le nom de votre image iso (notez qu'elle sera identifiée sous *live-image-amd64.hybrid.iso*, mais montée en *NAME*).

--iso-volume NAME : pareil, le nom... j'ai jamais su lequel il fallait mettre.

--linux-flavours FLAVOUR|"FLAVOURS" : permet de choisir une autre branche que la votre. *586* ou *686-pae* pour un live en i386 par exemple.

--linux-package "PACKAGE" : le nom du kernel à installer. Ce nom sera associé à "flavours" (voir précédent) pour former le kernel par défaut sur le liveCD.

--memtest memtest86+|memtest86|none : intégrer une entrée memtest, et si oui, laquelle.

--system live|normal : vous pouvez aussi construire un système classique avec live-build.

--archive-areas ARCHIVE_AREA|"ARCHIVE_AREAS" : main, contrib, non-free ?

--security true|false : utilisation des dépôts security.

--source true|false : produire les sources complètes du liveCD (avec les sources des paquets utilisés pour la construction).

--updates true|false : intégrer les dépôts updates dans les sources du liveCD.

--backports true|false : intégrer les dépôts backports dans le liveCD.

--verbose : bavard, cool pour apprendre ou debugger.

--win32-loader true|false : intégrer un lanceur pour windows™.

Voilà en gros les options principales. La liste complète est plus longue et sera peut-être utile selon vos besoins, mais pour commencer, c'est largement suffisant.

Afin de soulager ce script, live-build permet la prise en compte automatique de certains fichiers si ils sont placés au bon endroit avec la bonne extension. C'est ce que nous allons voir dans la section «choix et intégration des paquets» (p.7) pour l'installation de paquets supplémentaires. Juste avant, un petit exemple avec le script de construction d'handylinux-686-pae

Exemple de script simple

```
#!/bin/sh
# build script - handylinux-686
# http://handylinux.org
################################
lb config noauto \
--mode "debian" \
--system "live" \
--architectures "i386" \
--distribution "jessie" \
--linux-flavours "686-pae" \
--archive-areas "main contrib non-free" \
--security "true" \
```

```
--updates "true" \
--backports "true" \
--binary-filesystem "fat32" \
--binary-images "iso-hybrid" \
--apt-indices "true" \
--apt-recommends "false" \
--apt-secure "true" \
--apt-source-archives "true" \
--linux-package "linux-image" \
--bootloader "syslinux" \
--debian-installer "live" \
--debian-installer-gui "true" \
--iso-application "handylinux" \
--iso-volume "handylinux" \
--memtest "none" \
--clean \
--debug \
--verbose \
--source "false" \
"${@}"
```

Choix et intégration des paquets

Comme vous pouvez le constater, le script principal du build ne mentionne aucun paquets hormis le kernel *linux-image*. Les paquets sont installés depuis une liste de *.deb directement ou depuis un script additionnel placé dans les *hooks*. Nous allons détailler chaque méthode.

Choix des paquets

Avant de lister les paquets, il faut les choisir. Comme vous avez pu le voir dans le test à vide, live-build ne colle pas grand chose par défaut... le système de base quoi. Du coup, tout est possible, dans la limite des 4G d'un DVD bien sûr (vous pouvez toujours construire un live plus gros pour l'utiliser en tant que sauvegarde, mais cette option sera évoquée en fin de tuto).

Modifications des paquets à installer depuis config/packages-lists

C'est **la méthode par défaut**. Il suffit de placer une liste de paquets séparés par un espace, dans un fichier *config/packages-list/debian_perso.list.chroot* pour qu'elle soit automatiquement prise en compte par live-build. Pratique non  ? On peut même y coller des conditions pour celles et ceux qui font du *cross-arch-building* (contruction de live i386 et amd64 depuis un système amd64). Les commentaires sont acceptés.

Voici la liste des paquets pour handylinux. Notez qu'elle est très détaillée car j'utilise l'option "--apt-recommends false" afin d'alléger le système, ce qui oblige à l'ajout de tout ce dont j'ai besoin :

```
## handylinux packages
##
## handylinux
handy-menu handytri slingscold-launcher redshift-config mpartage
slimconf xl-wallpaper
#if ARCHITECTURE i386
vmg
#endif
##basics
adduser gnupg user-setup eject desktop-base locales console-data
console-setup console-setup-linux console-common keyboard-
configuration iptables iproute net-tools
##xserver - window manager
xorg xfce4 xfdesktop4 slim
##terminal
xterm xfce4-terminal
##editor
leafpad evince
##print
cups hplip hplip-gui hpijs-ppds printer-driver-c2050 printer-
driver-c2esp printer-driver-cjet printer-driver-escpr
openprinting-ppds gutenprint-locales printer-driver-gutenprint
printer-driver-hpcups printer-driver-postscript-hp printer-driver-
m2300w printer-driver-min12xxw printer-driver-pnm2ppa printer-
driver-ptouch printer-driver-sag-gdi printer-driver-splix printer-
driver-foo2zjs printer-driver-hpijs printer-driver-pxljr system-
config-printer magicfilter djtools librecode0 recode lpr
##games
mahjongg aisleriot gbrainy gnome-sudoku
##network apps
chromium chromium-inspector chromium-l10n icedove icedove-l10n-fr
wpasupplicant wireless-tools network-manager-gnome pppoeconf
network-manager-openvpn-gnome network-manager-vpnc-gnome
flashplugin-nonfree icedtea-plugin
##graphics
shotwell simple-scan tumbler
##tools
x11-apps xpad xdotool gcalctool bleachbit htop iotop iftop
hardinfo gtk-redshift zenity bluez gnome-bluetooth blueman fuse-
utils fusesmb netatalk xcalib gnome-search-tool hddtemp lm-sensors
libunique-1.0 libgnome-menu2 gnome-font-viewer ntp rpl
##fonts
xfonts-terminus fonts-droid fonts-freefont-ttf fonts-liberation
xfonts-100dpi xfonts-75dpi xfonts-base xfonts-utils ttf-freefont
ttf-mscorefonts-installer
##theme
murrine-themes gtk2-engines gtk2-engines-pixbuf gtk3-engines-unico
gnome-themes-standard gnome-icon-theme-symbolic dmz-cursor-theme
##media
```

```
alsa-base alsa-utils alsa-tools vlc quodlibet quodlibet-plugins
radiotray asunder flac wavpack lame gstreamer0.10-ffmpeg
gstreamer0.10-plugins-bad gstreamer0.10-plugins-base oggconvert
dvd+rw-tools gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly
gstreamer0.10-x libphonon4 libdvdcss2 cheese gstreamer0.10-
fluendo-mp3 gstreamer0.10-alsa phonon-backend-vlc phonon-backend-
gstreamer libasound2-plugins
##archive
file-roller unrar dtrx p7zip-full bzip2 lzma zip
##power
upower xscreensaver acpi xscreensaver-data-extra xscreensaver-gl
xscreensaver-gl-extra
##filesystem
dosfstools ntfs-3g mtools ntfsprogs e2fsprogs gnome-disk-utility
gparted
##firmwares&drivers
firmware-b43-installer firmware-b43legacy-installer b43-fwcutter
linux-wlan-ng firmware-linux firmware-linux-free firmware-linux-
nonfree alsa-firmware-loaders intel-microcode iucode-tool
##admin
sudo gksu synaptic software-center update-notifier os-prober
gnome-system-tools live-tools lsb-release pm-utils gvfs gvfs-bin
gvfs-fuse gvfs-backends hal gnome-keyring libpam-gnome-keyring
policykit-1 policykit-1-gnome dbus dbus-x11 consolekit pam-dbus-
notify libpam-ck-connectoraccountsservice
##other
python python-dbus python-gtk2 python-notify python-lxml python-
xdg xdg-utils libnotify-bin usbutils libspeechd2 xsltproc libmtp9
libmtp-runtime aspell-fr myspell-fr manpages-fr manpages-fr-extra
##xfce
xfburn xfce4-appfinder xfce4-artwork xfce4-battery-plugin xfce4-
weather-plugin xfce4-cpufreq-plugin xfce4-cpugraph-plugin xfce4-
datetime-plugin xfce4-volumed xfce4-mixer xfce4-mount-plugin
xfce4-notifyd xfce4-panel xfce4-mailwatch-plugin xfce4-places-
plugin xfce4-power-manager xfce4-power-manager-plugins xfce4-
screenshooter xfce4-sensors-plugin xfce4-session xfce4-settings
xfce4-taskmanager xfce4-terminal xfce4-utils xfconf xfprint4 xfwm4
xfwm4-themes thunar thunar-archive-plugin thunar-data thunar-
volman
```

Si vous utilisez des méta-paquets comme *gnome* ou *kde* avec les `--apt-recommends true`, la liste sera bien sûr beaucoup moins longue. Pour éviter les doublons, vous pouvez vérifier la liste des dépendances de vos paquets depuis le gestionnaire de paquets Synaptic en graphique ou depuis un terminal :

```
sudo apt-get update && sudo apt-get install apt-rdepends
apt-rdepends mon_paquet
```

Gestion des dépôts externes depuis config/archives

Pour pouvoir profiter de paquets distribués sur des dépôts externes, il faut ajouter le *sources.list* du-dit dépôt ainsi que sa clé d'authentification. Le *sources.list* additionnel

sera intégré dans le */etc/apt/sources.list.d/debian_perso.list* de votre liveCD. Vous pouvez choisir d'activer ces dépôts dans le live et/ou l'install. j'explique :

Pour activer un dépôt dans le live (pouvoir mettre à jour et installer depuis ces dépôts en session live), il suffit de placer les sources dans *debian_perso/config/archives/debian_perso.list.binary*.

Pour activer un dépôt dans le système installé (pouvoir l'utiliser après l'installation), il suffit de placer les sources dans *debian_perso/config/archives/debian_perso.list.chroot*.

Un exemple afin de profiter des dépôts **VideoLan** pour *libdvdcss2* (la librairie qui permet de lire les DVD du commerce) :

- placer *videolan.list.chroot* et *videolan.list.binary* dans *debian_perso/config/archives* et de même pour *videolan.key.binary* et *videolan.key.chroot*

```
22:15 arp > cd debian_perso/config/archives
22:15 arp > tree
.
├── videolan.key.binary
├── videolan.key.chroot
├── videolan.list.binary
└── videolan.list.chroot
```

les fichiers (identiques) videolan.list.chroot & videolan.list.binary

```
## Videolan ##
deb http://download.videolan.org/pub/debian/stable/ /
```

les fichiers *key* sont à récupérer sur le site du dépôt concerné. Pour videolan :

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.12 (GNU/Linux)
mQENBFIcm2kBCADL5XxrZ27r2z3qqV6r3FUfg90UvM58wvxryAaoXr+y+W0Joh/m7+
VtxvODBekTrACuGy7Vg1NzsFHNzEuAbPctpbZgffNSwxgrToFERenEbF14f7FfxDkF
3vLgyll5frNozE2csAaj19lTQLBTNZbhkEys0V844zQSDN9/UYp5BWWXRFt8xHtocG
ayloFPdVOcWA+B8gO6hBWyZa7QaMKVvzEiHyqzmxVINJ1k1P3PFqMuHGOFJsPY4ARR
ZzQ+JYfrtgoqtAh9LYMQAfKmn7F3nlQyUuocEsKuhS0kIDtvLxTdjkefZYcmQXWxPS
xUan6lqP3LGrkMdEmLrzwnFkujABEBAAG0MFZpZGVvTEFOIEFQVCBTaWduaW5nIEtl
eSA8dmlkZW9sYW5AdmlkZW9sYW4ub3JnPokBOAQTAQIAIgUCUhybaQIbAwYLCQgHAw
IGFQgCCQoLBBYCAwECHgECF4AACgkQa8peTbhCiNk7HQgAyy/HOyzk0UdrFv0bZgqZ
LxJcdB7ctcruYNkO0eZE09+sPjzBuX52sx6GuXElG/NUw/QoGUB5kDDKLyeddpyQgh
U7UhsIZoNdSb2UfVCqSosJ1G7dacRlVh5LFNF1ptYjXGzaNIp0zme8YtoQKVCO48YY
If+j94Q5AukTEd1vdPAUAm7e4Q6uYcvpyz1TIoZgnnO//fcoWPQrRHnxfsIB0GrqVk
Hncjt/U1qsxQh7hAJnKjwgnveUe7Q2ey54QId9BQWZHOxeIXpbhFGwBmHxBGAnDWmg
557YlsUI4ejiKy6DBsO8h91NL44cbN8H61Z+T3UHNWXycr/4PClWZEitkYhGBBARAg
AGBQJSHJxcAAoJEGFgnhjAr/EP+DAAmwfLrnFEktmlOv8YssNOBlCh8cEvAJ9qPJoC
bY4tWZtaC6aa4R2owLsSoohGBBARAgAGBQJSHJxpAAoJEHGAcTvljRrcaywAn3Gf8f
HXVdgpCrycr584Iie1+mEyAKCR8V9kt5DWF6VGUXbhV9INKSKCKbkBDQRSHJtpAQgA
w4TQFpFs9PthvzTj/fpa6rioz2gYIGxGTQHuC8LsOCEKuy4AFa8XF2d89NM7ApF/ix
U3QDKd7I6YydkW/yp3rTCbnI+xS649yxDd2RDIqVPsqjWuO8jEs8sQ/wOWMnDdCU95
Zwv0BU491u63UbRzamv5+kw7QGQqPe4VAnsiVxwnV1GOa9Ft0FmaD/m7KhFWAGkTF3
```

```
pV432gH93V8srPZD+CWZgld97E45TXXjzWK03NnUyIK+rx+LHJf4I8IFFP0F77FJDI
FRJ6BiYbXoKNDVk/nvy1jWg598dFi7jzw1t83EnAVgTvBR+bQqBpB69EtY2WUykABM
njMj9t0Y6eFwARAQABiQEfBBgBAgAJBQJSHJtpAhsMAAoJEGvKXk24QojZNMIH/3YE
92QZ22YYdl6TchMDswsBFXvsS2/W0xhvKvG5YuyCKoUKYCornK74YrsRURAhYAOvOf
rcWeq+wtmUmXrhNvZ2Qz370hS2EW6Db7FnMdu188LrVZS0AmC1ztu8/DlwRcRF4e05
CVgwUMngG2RiAuZN1ZYQiHJwU2444/Vm+ny/YD55I+RUmKFW69iAvWnJrz5QzohoeZ
efr48c3FU1glKYE3OtJK/T+/nOVzt+Kzjw8mmrtkZqck/ZVpA51ikcTRSpyLw5ioLp
zLFbB/Ew+Dx3CzFOZhG9UxBV08BOw9a300900hxPDoKBu210T3tBW9XWmFPohT2fsg
5OXZHvp/Q==pGjl
-----END PGP PUBLIC KEY BLOCK-----
```

Chaque liste et clé GPG additionnelle correctement nommée et placée dans *debian_perso/config/archives/* sera disponible:

- en live avec l'extension *.binary*
- sur le système installé avec *.chroot*.

Installation automatique de paquet locaux depuis config/packages

Un autre moyen d'ajouter des paquets à votre liveCD est de littéralement les "coller" dedans. En effet, tous les paquets de type debian (*.deb*) placés dans le dossier *debian_perso/config/packages* seront installés dans le liveCD.

Attention : cette procédure n'est pas effectuée par apt, il n'y a donc pas de prise en charge des dépendances : prenez soin de les ajouter dans le même dossier, ou de les ajouter dans la liste des paquets à installer (voir section précédente sur les *package-lists*)

Installation de paquets avec un script additionnel depuis config/hooks

L'arme ultime de la personnalisation par commande, le reste se faisant par l'ajout de fichiers ou dossiers. Donc, l'arme ultime : les scripts placés dans le dossier *debian_perso/config/hooks/* seront exécutés dans le chroot avant la compression en *squashfs*.

Cela vous permet de faire en gros ce que vous voulez dans le nouveau système fraîchement construit, bien pratique quand on lit pas tout le manuel et qu'il faut bidouiller un truc .

Je vous livre le script config/hooks/handylinux.chroot pour handylinux. Vous noterz que cette version (basée sur Debian Wheezy) utilise les backports pour obtenir une version plus récente de LibreOffice. Encore un exemple de tout ce qu'on peut faire simplement avec live-build :

```
#!/bin/sh
# activer zram
# installer libreoffice depuis les backports
# installer les paquets externes :
# skype, teamviewer, fonts-opendyslexic, minitube et whiskermenu
# prise en compte du 64 et ajout du multiarch
# prendre en compte mes "proposed-updates"
##########################################
#set -e
echo "INFO : begin arphooks"
echo ""
echo "INFO : enable zram"
update-rc.d zram defaults
echo ""
echo "INFO : install fonts-opendyslexic"
dpkg -i /usr/share/handylinux/fonts-opendyslexic_20130616-
1_all.deb
echo ""
echo "INFO : installation de libreoffice"
apt-get install -y -t wheezy-backports libreoffice libreoffice-gtk
echo ""
if [ $(uname -m) = "x86_64" ]; then
echo "INFO : amd64 detected : enable multiarch"
dpkg --add-architecture i386
apt-get update
echo "installation des paquets i386"
echo "INFO : install vmg"
apt-get install vmg
echo ""
echo "INFO : installing skype"
dpkg -i /usr/share/handylinux/skype-debian_4.2.0.13-1_i386.deb
apt-get -f install
echo ""
echo "INFO : installing teamviewer"
dpkg -i /usr/share/handylinux/teamviewer_linux_x64.deb
apt-get -f install
echo ""
echo "INFO : installing whiskermenu"
dpkg -i /usr/share/handylinux/xfce4-whiskermenu-plugin_1.3.2-
1_amd64.deb
echo ""
#echo "INFO : install minitube"
#dpkg -i /usr/share/handylinux/minitube_2.0-3_amd64.deb
#echo ""
else
echo "INFO : installing skype"
dpkg -i /usr/share/handylinux/skype-debian_4.2.0.13-1_i386.deb
echo ""
echo "INFO : installing teamviewer"
dpkg -i /usr/share/handylinux/teamviewer_linux.deb
echo ""
echo "INFO : installing whiskermenu"
dpkg -i /usr/share/handylinux/xfce4-whiskermenu-plugin_1.3.2-
1_i386.deb
```

```
echo ""
echo "INFO : install minitube"
dpkg -i /usr/share/handylinux/minitube_2.0-3_i386.deb
echo ""
fi
echo ""
echo "INFO : cleaning"
rm /usr/share/handylinux/skype-debian_4.2.0.13-1_i386.deb
rm /usr/share/handylinux/teamviewer_linux.deb
rm /usr/share/handylinux/teamviewer_linux_x64.deb
rm /usr/share/handylinux/fonts-opendyslexic_20130616-1_all.deb
rm /usr/share/handylinux/minitube_2.0-3_i386.deb
#rm /usr/share/handylinux/minitube_2.0-3_amd64.deb
rm /usr/share/handylinux/xfce4-whiskermenu-plugin_1.3.2-1_i386.deb
rm /usr/share/handylinux/xfce4-whiskermenu-plugin_1.3.2-
1_amd64.deb
echo ""
echo "INFO : endof HOOK
```

Note : les paquets utilisés dans ce script sont intégrés au live en dur directement dans le dossier *debian_perso/config/includes.chroot/usr/share/handylinux* parfaitement accessible puisqu'on est dans le chroot. Cet aspect sera développé avec la personnalisation du système et de l'utilisateur.

Voilà, du bash tout con en fait. Vous comprenez donc que vous pouvez tout faire dans votre système : installer des paquets, télécharger des archives, même installer git, choper des sources, et compiler direct depuis ce script !

Vous avez maintenant toutes les armes pour ajouter les paquets que vous désirez dans votre liveCD. Vous pouvez effectuer un test de suite.

- Si vous avez choisi d'intégrer un DE comme Gnome, Kde ou Xfce, ils intègrent des gestionnaires de connexion (Gdm, Slim ou Kdm) et tout le *staff X* vous permettant de tester votre liveCD.
- Si vous avez opté pour un système minimal sans gestionnaire de connexion, vous pourrez démarrer avec un "`startx`" (pour openbox ou fluxbox par ex) ou tester le live en console si votre système est vraiment minimal.

Test des applications intégrées

Après le test à vide, je suppose que vous êtes impatients de construire un système avec du X... Prenez le temps de bien vérifier les paquets et leurs dépendances, ainsi que la disponibilité des dépôts externes si besoin. Comment ? En copiant les sources.list @home et en testant avec un *apt-get update*.

Tout est prêt ? Vous aussi ? Alors avant de tester, il va falloir nettoyer votre dossier de build qui contient encore l'ancien liveCD. Cette opération est lancée avec un `lb clean` :

```
cd ~/tmp/debian_perso/
sudo lb clean
```

Et voilà votre dossier de build tout propre, hormis le cache contenant le *bootstrap* de base et les paquets déjà téléchargés. Ce n'est pas gênant : les dépôts vont être consultés et les mises à jour effectuées si nécessaires. Je vous conseille toutefois de vider le cache avec un `sudo rm -Rf cache` avant la construction du live définitif, histoire de partir sur une base propre. Vous devez le faire absolument si vous construisez des liveCD d'architectures différentes à la suite.

Assez parler... let's build !

Donc toujours dans votre dossier de build, on lance un

```
sudo lb build
```

et on va boire un thé. Votre *live-image-amd64.hybrid.iso* pourra être testée avec Virtualbox ou tout autre émulateur. Vous pouvez aussi tester *en vrai*. Dans ce cas, *dd* est ton ami.

dd est une commande qui permet (entre autres choses) de transférer une image ISO sur un support USB afin de reproduire un système de fichier identique à celui d'un CDROM, ce qui permet ensuite de démarrer sur cette clé usb comme sur un liveCD.

Test du liveCD en vrai

Pré-requis : une clé usb de taille suffisante sans données importantes (le processus va effacer toute votre clé) et un terminal. (dd fait partie des outils de base)

Identifier la clé USB à utiliser grâce à la commande (si ça répond rien, tentez -by-uuid) :

```
ls -l /dev/disk/by-label
```

qui vous donnera un résultat du type :

```
lrwxrwxrwx 1 root root 10 Oct 14 18:56 B4G -> ../../sdb1
lrwxrwxrwx 1 root root 10 Oct 13 18:32 debian -> ../../sda1
lrwxrwxrwx 1 root root 9 Oct 14 18:56 handylinux -> ../../sdb
lrwxrwxrwx 1 root root 9 Oct 14 18:56 U3\x20System -> ../../sr1
```

Ici, notre clé est identifiée comme **sdb** et contient la partition **sdb1**. Notez bien ce **sdb** pour ne pas, par erreur, effacer une partition de votre disque dur interne (*sda*).

Transférer l'ISO à tester depuis le dossier de build:

```
cd tmp/debian_perso/
sudo dd if=live-image-amd64.hybrid.iso of=/dev/sdb bs=4M && sync
```

Le temps de transfert sur votre clé USB dépend de la taille de ISO et du taux de transfert de votre port USB. Cette opération peut durer 10 à 15 minutes sans aucun signe d'activité de votre terminal. Le terminal vous rendra la main une fois le transfert terminé. Vous pourrez alors redémarrer pour utiliser votre debian_perso en live 😉.

Ne soyez pas trop exigeant sur le résultat, toutes les configurations des applications sont encore *par défaut*.

Vous pouvez cependant vérifier le bon fonctionnement des applications intégrées, les dépendances manquantes (à ajouter dans le *package-lists*), choisir de nouvelles applications etc.

Pendant le test, notez tout ce que vous faites pour améliorer votre système:

- les paquets installés dans *config/package-lists/debian_perso.list.chroot*,
- les modifications système dans vos *debian_perso/config/hooks/*
- et tout ce que vous voulez garder (réglages du navigateur, préférences personnelles, réglages par défaut) sont à archiver et à copier ailleurs pour être sûr de les retrouver.

Mon conseil : configurez votre système en session live, puis faites une archive de votre /home sur une deuxième clé ou envoyez-la sur un serveur (attention de ne pas vous identifier avec un mot de passe lors de votre session de navigateur en session live : le mot de passe sera distribué à tout le monde...). Vous pourrez vous servir de tout ça pour la phase de personnalisation de l'utilisateur.

Voilà, vous devriez disposer d'un système exploitable intégrant vos applications favorites, depuis les dépôts Debian ou depuis d'autres sources. La configuration des applications votre liveCD est semblable à celle qu'on trouve dans un liveCD Debian officiel... c'est du fonctionnel...

Personnalisation système et utilisateur

Nous avons vu que la construction d'un liveCD s'effectuait dans un *chroot*, un sous-système dans votre système. Nous avons vu que vous pouviez agir sur ce *chroot* grâce aux options du live-build, mais également depuis les scripts *hooks* qui modifient le *chroot* de l'intérieur avant sa compression dans le *squashfs*.

Nous allons voir comment remplacer ou ajouter des fichiers dans le chroot avant même

le processus de construction.

config/includes.chroot : modification directe du système

Si vous avez fait le test à vide lors du premier chapitre, vous avez remarqué ce dossier vide dans votre environnement de build. Ce dossier sera copié dans le *chroot* avant l'exécution des scripts *hooks*.

> *Euh .. oui .. et ?*

Et bien cela signifie que tout ce qui sera dans ce dossier *config/includes.chroot* se retrouvera dans le *chroot*, donc dans le *squashfs*, donc dans le **liveCD**.

> *oki. cool... comment ça fonctionne ?*

Tout simple :

- vous voulez votre theme_perso dispo dans le live? Il suffit de créer un dossier *debian_perso/config/includes.chroot/usr/share/themes* et d'y coller votre dossier theme_perso.
- vous voulez que votre petit script_perso.sh soit dispo dans le live? Il suffit de créer un dossier *debian_perso/config/includes.chroot/usr/local/bin* et d'y coller votre script_perso.sh.

Vous avez compris le principe ? excellent non ?

L'arborescence créée dans *config.includes.chroot* se retrouvera dans le liveCD, à la place des fichiers originaux. Car c'est ça le bonus ! Vous pouvez ajouter des thèmes, des icônes, des scripts, mais surtout, vous pouvez remplacer des fichiers existants afin de modifier le comportement du système par défaut.

Quelques exemples

- Vous utilisez slim comme gestionnaire de connexion et vous voulez changer le thème utilisé ? Il suffit de placer un */etc/slim.conf* dans *config/includes.chroot* avec le thème que vous voulez. Vous prendrez soin de placer le thème correspondant dans *config/includes.chroot/usr/share/slim/themes/* .
- Vous avez placé un script dans *config/includes.chroot/usr/local/bin/* et voudriez le voir apparaître dans les menus système ? Il suffit de placer un **.desktop* adéquat dans *config/includes.chroot/usr/share/applications/* afin que votre script soit reconnu par les menus classiques.
- Vous désirez ajouter automatiquement l'utilisateur à certains groupes ? Il suffit de copier votre fichier */etc/adduser.conf* dans

debian_perso/config/includes.chroot/etc/adduser.conf et de modifier les lignes 77 et 81 concernant les groupes par défaut des utilisateurs créés sur le système.

Ce qui ne fonctionne pas :

- Vous ne pouvez pas coller un */etc/defaut/grub* modifié, il sera écrasé lors de l'installation
- Le gestionnaire de démarrage (syslinux dans notre exemple) ne fait pas partie du *squashfs*, la modification du comportement au lancement se fera dans un autre dossier (les dernières modifications expliquées plus tard)
- La modification de .desktop existant est inutile car le .desktop sera écrasé à la prochaine mise à jour du paquet correspondant. Si vous désirez changer l'icône d'une application pour un lanceur par ex, il faudra le faire depuis le dossier utilisateur (chapitre suivant)

Ne pas oublier :

Cette méthode de modification n'est pas prise en charge par le système lui-même : vous devez donc prévoir toutes les dépendances nécessaires si vous ajoutez des scripts par ex.

Ni apt, ni dpkg n'auront trace de vos modifications si vous ajoutez un paquet à la main en copiant les fichiers exécutables : vous ne pourrez pas le mettre à jour depuis apt.

Ne recopiez pas vos préférences système dans *config/includes.chroot* : vérifiez tout ce que vous ajoutez. Certaines configurations sont propres à votre système et ne fonctionneront pas forcément en live ou sur un autre ordinateur.

Voilà pour les modifications du système. Bien...

je peux ajouter des trucs dans config/includes.chroot/home alors ?

Et non. c'est ailleurs que ça se passe. Le home reste vide dans le build. Il ne sera peuplé que lors du lancement du live et à l'installation.

config/includes.chroot/etc/skel : modification de l'utilisateur

Alors c'est là que ça se passe pour modifier l'utilisateur ?

Oui... /etc/skel aka skeleton. ...

C'est maintenant que vous allez vous servir de l'archive de votre /home préparée lors du test avec les applications. Vous pouvez aussi recopier vos dossiers et fichiers personnels (~/.config/*, ~/.bashrc, ~/.conkyrc, ~/.mozilla/*, etc) afin de profiter/partager vos réglages et marques pages.

Attention : la copie de vos dossiers personnels copie aussi vos cookies/mot de passe etc. Si vous souhaitez partager votre ISO, je vous conseille de travailler sur un dossier vierge.

Comment faire ?? Simple :

- vous commencez par sauvegarder votre dossier personnel ~/.mozilla par ex (pour firefox) dans ~/.mozilla_backup.
- à votre prochaine ouverture de Firefox, vous aurez la config par défaut. Installez les plugins que vous voulez, réglez les marques-pages, la disposition de la barre d'outils etc, puis effacer votre historique de navigation, votre cache.
- recopier le dossier neuf ~/.mozilla dans votre debian_perso/config/includes.chroot/etc/skel/.mozilla.
- il ne vous reste plus qu'à effacer votre ~/.mozilla et renommer votre ~/.mozilla_backup en ~/.mozilla pour retrouver vos préférences et vos réglages.

Prenez soin de refermer Firefox avant de restaurer votre dossier. En revanche, si vous souhaitez utiliser live-build comme outil de sauvegarde perso, allez-y, copiez tout ! (cette option sera détaillée plus bas).

Vous pouvez également déterminer les dossiers par défaut utilisés sur votre système. j'explique : les DE et WM aux nomes *freedesktop* acceptent les dossiers nominaux (XDG_DOCUMENTS_DIR = ~/Documents, XDG_PICTURES_DIR = ~/Images etc) et ces dossiers sont déclarés dans le fichier ~/.config/user-dirs.dirs. Ce fichier vous permet donc de déterminer quels dossiers vont être utilisés par défaut et ainsi les placer dans votre /etc/skel.

Exemple :

```
18:35 arp > cat .config/user-dirs.dirs
XDG_DESKTOP_DIR="$HOME/tmp"
XDG_DOWNLOAD_DIR="$HOME/downloads"
XDG_TEMPLATES_DIR="$HOME/tmp"
XDG_PUBLICSHARE_DIR="$HOME/tmp"
XDG_DOCUMENTS_DIR="$HOME/docs"
XDG_MUSIC_DIR="$HOME/zik"
XDG_PICTURES_DIR="$HOME/pics"
XDG_VIDEOS_DIR="$HOME/videos"
18:36 arp > tree -L 1
.
├── docs
├── downloads
├── pics
├── tmp
├── videos
└── zik
```

Pour afficher mon thème et mes icones que j'ai placé dans le dossier *config/includes.chroot/usr/share/themes* et */usr/share/icons*?

- soit vous utilisez un DE qui aura intégré les préférences dans sa config (.config/xfce4 pour xfce par ex),
- soit vous avez choisi un système minimal, et c'est le gtkrc qui gère ça. avec gtk3, yen a deux maintenant... donc il faut placer un gtkrc dans */etc/skel/.gtkrc-2.0* et un settings.ini dans */etc/skel/.config/gtk-3.0/settings.ini.*

Et mon menu spécial xml openbox que j'ai mis des plombes à faire ?

Il va aller dans config/includes.chroot/etc/skel/.config/openbox/menu.xml

...

Voilà, comme pour les modifications système, un simple processus de copie au bon endroit intégrera vos préférences dans le liveCD. Et comme pour les modifications système, prenez soin de vérifier et re-vérifier les dossiers et fichiers que vous ajoutez, surtout si vous comptez partager votre distribution.

Tests, debug et derniers changements du chroot

Vous avez désormais un dossier de build prêt à l'emploi pour construire une Debian personnalisée.

Alors un petit nettoyage, et on envoi le build

```
cd tmp/debian_perso/
sudo lb clean
sudo lb build
```

Pour la dernière phase de test, je vous conseille d'installer votre distro sur une machine virtuelle afin de pouvoir noter vos dernières commandes (qui seront à répercuter dans le dossier de build) et pouvoir profiter des modifications lors d'un reboot.

Conseils pour un bon debug :

- Testez chaque application en condition réelle : ne faites pas qu'ouvrir votre navigateur, naviguez avec. N'ouvrez pas simplement gimp, éditez une image, enregistrez-la et imprimez-la.
- Lancez les applications depuis un terminal afin d'observer les erreurs éventuelles et noter les modifications qui s'imposent (ajout d'un gtk-engine pour le rendu graphique, d'une librairie manquante...)

- Lancez un update/upgrade et vérifiez les paquets recommandés non-installés pour détecter d'éventuels petits outils manquants

Côté préférences utilisateur : je vous conseille de peaufiner vos réglages personnels, puis d'archiver tout ça pour la dernière étape et la construction finale.

Modification du build > build > tests > répercution des modifications > nettoyage > build ... ce processus devra être répété pour parvenir à la distro de vos rêves 😜.

Finalisation de l'image iso

Vous avez un liveCD exploitable et personnalisé, il reste à peaufiner l'environnement de démarrage.

config/includes.binary/isolinux : modifier le menu du liveCD

Le fond d'écran et le texte qui s'affichent au lancement du live (par défaut , c'est l'écran noir avec les infos de construction et le joli casque de chantier jaune) se configurent dans le dossier *debian_perso/config/includes.binary/isolinux*. Vous pouvez laisser le texte si vous voulez, mais pour le fond d'écran ,il se nomme *splash.png* et doit faire 640x480 en PNG. Pour les fichers texte, il suffit de recopier et modifier les fichiers originaux. (pour trouver les fichiers originaux, il faut monter le dernier liveCD testé, puis direction le dossier isolinux à la racine de votre liveCD)

```
20:29 arp > pwd
/home/arp/tmp/debian_perso/config/includes.binary/isolinux
20:29 arp > tree
.
├── install.cfg
├── isolinux.cfg
├── live.cfg
├── menu.cfg
├── splash.png
└── stdmenu.cfg
```

Détail des fichiers textes du menu

le fichier install.cfg :

```
label installgui
menu label ^Installer debian_perso sur votre ordinateur
linux /install/gtk/vmlinuzinitrd /install/gtk/initrd.gz
append video=vesa:ywrap,mtrr vga=788 -- quiet
file=/cdrom/install/preseed.cfg locale=fr_FR.UTF-8 keymap=fr
```

ici l'installeur graphique va démarrer en français avec un clavier fr directement. vous remarquez aussi l'appel au fichier preseed que nous détaillerons plus loin

le fichier live.cfg

```
label live
menu label ^Lancer debian_perso sans risques
linux /live/vmlinuz
initrd /live/initrd.img
append boot=live config quiet splash username=humain
hostname=debian_perso keyboard-layouts=fr locales=fr_FR.UTF-8
```

ici; la session live se lancera en français, le nom de l'utilsateur sera "humain" et le nom d'hôte "debian_perso"

le fichier menu.cfg

```
menu hshift 0
menu width 82
menu title Tester ou Installer debian_perso
include stdmenu.cfg
include live.cfg
include install.cfg
menu clear
```

Je vous laisse modifier les autres comme vous désirez si nécessaire... bah oui, pas de copier/coller .. faut bosser un peu ! 

config/includes.debian-installer/usr/share/graphics et themes : modifier le thème de l'installeur graphique

C'est dans ces dossiers que se trouvent les fichiers utilisés par l'installeur graphique. à modifier si vous voulez... Le thème, c'est pas obligé, mais la bannière au moins, située dans *config/includes.debian-installer/usr/share/graphics/logo_debian.png* (une bannière en png de 800x75 pixels)

> *Et à part le look du truc et la langue, on peut changer quoi alors ??*

Et bien il nous reste deux fichiers à explorer avant de finaliser l'iso.

config/debian_installer/preseed.cfg : la préconfiguration de l'installeur

Ce fichier va vous permettre de supplanter les étapes de l'installeur Debian et ainsi accélérer le processus ou définir vos préférences. L'autre atout de ce fichier *preseed* est qu'il peut lancer un script de post-installation préalablement placé dans votre *chroot* lors

du build, afin de procéder aux derniers ajustements avant le reboot et le premier démarrage du système installé.

Un exemple commenté de fichier preseed :

```
############
## Global ##############
# fr
d-i debian-installer/language string fr
#d-i debian-installer/country string FR
#d-i debian-installer/locale string fr_FR.UTF-8
#d-i keymap select fr
# suppression de la config réseau lors de l'installation
d-i netcfg/enable boolean false
# horloge matérielle sur UTC :
#d-i clock-setup/utc boolean true
# Vous pouvez mettre toute valeur acceptée pour $TZ.
# Voyez ce que contient /usr/share/zoneinfo/ pour les valeurs
possibles.
#d-i time/zone string Europe/Paris
#########################
## gestion des comptes ##
#########################
# Ne pas créer de compte root (l'utilisateur ordinaire utilisera
sudo).
d-i passwd/root-login boolean false
# Le compte sera ajouté à certains groupes.
d-i passwd/user-default-groups string audio cdrom video sudo
netdev plugdev fuse users lp lpadmin
# apt - no mirror
d-i apt-setup/use_mirror boolean false
# tout dans une seule partition
d-i partman-auto/choose_recipe select atomic
################################
## Commandes lors de l'install ##
################################
# La préconfiguration de l'installation n'est pas sécurisée. Rien
dans
# l'installateur ne vérifie que des dépassements de tampons ou des
# exploitations quelconques des valeurs données dans ce fichier
n'ont pas lieu.
# N'utilisez que les fichiers dont vous connaissez la provenance !
# Pour tout dire, et parce que c'est en général utile, voici un
# moyen d'exécuter automatiquement une commande dans
l'installateur.
# La première commande est exécutée aussi tôt que possible, juste
après
# la lecture du fichier.
#d-i preseed/early_command string anna-install some-udeb
# Cette commande est exécutée juste avant que le partitionneur ne
commence. Il peut être
# utile de préconfigurer le partitionneur en fonction de l'état
des disques
```

```
# (qui ne sont peut-être pas visibles quand preseed/early_command
est exécutée).
#d-i partman/early_command \
#
string debconf-set partman-auto/disk "$(list-devices disk | head
-n1)"
# Cette commande est exécutée juste avant que l'installation ne se
termine,
# mais quand le répertoire /target est encore utilisable. Vous
pouvez exécuter
# chroot /target et utiliser cet environnement directement ; ou
bien vous
# pouvez utiliser les commandes apt-install et in-target pour
installer des
# paquets et lancer des commandes dans le répertoire target.
#d-i preseed/late_command string apt-install zsh; in-target chsh
-s /bin/zsh
# Grub
# Auto install grub if this is the only one system.
d-i grub-installer/only_debian boolean true
# suppression de la fenêtre de fin d'install
#d-i finish-install/reboot_in_progress note
# sources.list
d-i preseed/late_command string \
in-target /usr/local/bin/debian_perso_preseed ;\
in-target rm -f /usr/local/bin/debian_perso_preseed ;
```

Vous trouverez aisément des exemples de fichiers preseed sur la toile.

À propos des 3 dernières lignes du script : l'option *late_command* permet de lancer un script ou un commande à la fin de l'installation, une fois les utilisateurs en place, juste avant le redémarrage.

Encore une façon de personnaliser votre environnement. Je place un script dans *config/includes.chrotot/usr/local/bin/debian_perso_preseed* afin de modifier le *sources.list* Debian et modifier grub.

le fichier en question :

```
#!/bin/bash

# set handylinux sources.list
mv -f /usr/share/handylinux/sources.list /etc/apt/sources.list

# ajouter un background à GRUB
echo "GRUB_BACKGROUND=\"/usr/share/images/grub/handylinux.tga\""
>> /etc/default/grub
update-grub

exit 0
```

Ce script est utilisé simplement ici, mais rien ne vous empêche d'installer des paquets, faire un update, ou toute autre opération d'administration.

Notez que cette phase est "masquée" à l'utilisateur et peut le perturber si elle dure trop longtemps car plus rien ne bouge à l'écran pendant l'exécution du script *preseed*.

Et voilà, on a fait le tour des modifications simples de votre liveCD. Vous avez pu constater que plusieurs outils et procédures mènent au même résultat, c'est donc à vous de choisir la meilleure selon votre situation. Et donc non, c'est pas si simple dès lors qu'on modifie de plus en plus.

Il ne vous reste plus qu'à retourner dans votre dossier de build, re-vérifier vos dernières modifications puis lancer la formule magique

```
sudo lb clean && sudo lb build
```

et hop.. vous avez votre dérivée Debian personnalisée prête à être distribuée. Quelques mots à ce sujet...

Distribuer votre liveCD : ligne de conduite pour les dérivées Debian

Vous êtes fier de votre réalisation ? Vous voulez la partager, en faire profiter les autres ? Cool ! Vous pouvez faire héberger votre iso sur des sites dédiés aux projets open-source comme sourceforge.net, ou prendre un serveur mutualisé ou dédié afin de monter votre site pour votre OS avec forum, doc et blog associés. Une petite chose toutefois : vous venez de construire une dérivée Debian. Ce n'est pas n'importe quoi, vous utilisez le travail de toute une communauté, alors une petite ligne sérieuse :

Debian est une distribution GNU/Linux, mais pas que... Debian, c'est aussi une ligne de conduite, une charte, et certaines recommandations quant à la distribution de dérivée. Cette ligne de conduite est précisée dans les Debian Derivatives Guidelines (https://wiki.debian.org/Derivatives/Guidelines). je vous conseille vivement de consulter ce document afin de respecter un maximum la distribution grâce à laquelle vous avez pu construire la votre. Merci beaucoup 😊.

Live-build comme système de sauvegarde

Live-build peut être utilisé comme système de sauvegarde afin de pouvoir effectuer une réinstallation rapide de votre environnement personnel.

Pour ceci, il suffit de récupérer la liste de vos paquets pour les coller dans le live, et de prendre votre /home pour le coller dans le /etc/skel du live.

L'avantage est que votre système pourra s'installer facilement sur un autre ordinateur.

L'inconvénient est qu'il faut effectuer une sauvegarde externe de vos données si il y en a trop (vidéos/images/docs/zik).

Comment on fait le truc du live + données ?

On récupère la liste des paquets actuellement installés sur son système grâce à dpkg

```
dpkg --list | grep "ii" | awk '{print $2}' >
debian_perso/config/package-lists/my_system.list.chroot
```

Prenez soin de vérifier la liste complète pour éliminer les éventuels paquets externes que vous auriez compilé (et non présent dans les dépôts utilisés). De toute façon, si un paquet listé n'est pas dispo, le log du live-build vous le dira.

Pour la sauvegarde de votre /home, ne copiez que les données de configuration (le reste prend trop de place) dans le liveCD (debian_perso/config/includes.chroot/etc/skel), puis sauvegardez vos données régulièrement sur un disque dur externe. En cas de plantage ou de ré-installation sur un nouvel ordinateur, il vous suffira de lancer votre liveCD en mode installation, puis de rapatrier vos données une fois l'installation terminée. Vous retrouverez votre environnement, à jour, tout neuf... pratique hein ?!

Conclusion

log log log ... c'est le maître-mot de ce processus de construction. Pourquoi ? Parce que vous aurez de la chance si tout se passe bien du premier coup !

Mon conseil ? Lisez les logs debian_perso/build.log. Si tu veux des exemples, vas jeter un œil sur mes sources => arpinux@gitlab: https://git.framasoft.org/u/arpinux/groups

Bis@+

arp

Sommaire