DA TRADURRE


work in progress twenty-seven

Slackware Live Edition

Prefazione

Benvenuti nell' edizione live di Slackware! Questa e' una versione di Slackware 14.2 (e piu' recenti) che puo' essere utilizzata da un DVD o da una penna USB. E' un' immagine ISO che intende essere una vetrina di cio' che e' Slackware. Hai l' installer di default, nessun pacchetto o kernel customizzato, ma con tutte le potenzialita' della Slackware. La ISO e' creata da zero tramite gli scripts “liveslak”, utilizzando un mirror di pacchetti di Slackware.

La Slackware Live Edition non ha bisogno di essere installata sull' hard drive di un computer (*cosa che puoi comunque fare se lo vuoi: usando lo script* `setup2hd`). Puoi portare la versione USB con te tenendola in una tasca. Avrai un sistema Slackware preconfigurato pronto all' uso in un minuto ovunque tu possa mettere le mani su un computer con una porta USB.

La versione USB e' “persistente” - cio' significa che il sistema operativo memorizza le tue modifiche sulla chiavetta USB. La versione CD/DVD (e la penna USB se configurata appositamente) operano senza persistenza, che significa che tutti i cambiamenti che fai all' OS verranno persi al reboot.

Allo scopo di tenere protetti i tuoi dati in caso tu dovessi perdere l' USB (o in caso ti venisse rubata) puoi migliorare la tua USB persistente impostando una home directory criptata e/o un file di persistenza criptato, che sara' sbloccato al boot tramite una password che solo tu conosci.

Perche' ancora un' altra Slackware Live

I motivi che avevo per creare una Slackware Live Edition sono i seguenti:

1. Fornire una versione Live di Slackware appropriata; cioe' mostrare la Slackware cosi' com' e', ma senza doverla installare. Nessun occultamento dei messaggi del kernel che scorrono al boot; nessuno sfonodo customizzato, ecc... Indicata per scopi didattici, dimostrativi e di valutazione.
2. Il target dovrebbe essere la Slackware-current, la versione piu' all' avanguardia. Molte persone vogliono conoscere com'e' l' edizione di sviluppo di Slackware ma sono esitanti nell' installare la Slackware-current per paura che possa causare crash e perdite di produttivita'.
3. Fornire un modo per generare una Live ISO con solo un mirror di pacchetti di Slackware come fonte, completamente scriptato e deterministico.
4. Fornire tuttavia la possibilita' di customizzare il suo contenuto. Per esempio fornire versioni minimaliste o depotenziate di Slackware ma anche permettere l' inclusione di pacchetti di terze parti.

5. Dare l' opzione di crearsi da se' una USB bootabile con Slackware Live (che e' diverso da fare dd di una ISO ibrida su una penna USB!!)
6. KISS: Keep It Simple Stupid!

Varianti ISO

Gli scripts “liveslak” possono generare una varieta' di sfumature di Slackware:

1. una completa Live Edition a 64bit di Slackware-current (in una ISO da 4.0GB)
2. una ISO XFCE “smagrita” (700 MB) con XDM come login manager grafico. Ci sta su un CDROM o una penna USB da 1 GB;
3. una ISO di Slackware64-current (4.3 GB) che contiene Plasma 5 'ktown' al posto del KDE di Slackware.
4. una Workstation Audio Digitale (DAW) basata su un set di pacchetti di Slackware customizzato piu' un Plasma 5 di base, che contiene una ricca collezione di software per musicisti, producers e artisti dal vivo.
5. una variante con Mate (3.2 GB) dove KDE 4 e' stato rimpiazzato da Mate (un fork di Gnome 2);
6. una variante Cinnamon (un fork di Gnome 3 Shell che rimpiazza il KDE4 di Slackware);
7. una variante Dlackware, che e' Gnome3 + PAM + systemd sopra alla Slackware e con KDE4 rimosso.
8. una edizione StudioWare che contiene tutti i pacchetti software per editing audio, video e foto del progetto.
9. una variante *Custom* a cui puoi dare il tuo nome, la sua lista del pacchetti e una configurazione custom di post-installazione.

Download delle immagini ISO

I siti comuni per il download sono:

- Sito primario: https://download.liveslak.org/ (rsync://liveslak.org/liveslak/)
- https di Darren https://slackware.uk/liveslak/ (rsync://slackware.uk/liveslak)
- http di Willy http://repo.ukdw.ac.id/slackware-live/

Documentazione per l' Utente Finale

Utilizzare l' immagine ISO

Le immagini ISO sono ibride, che significa che potete masterizzarle su DVD, o usare 'dd' o 'cp' per copiare la ISO su una penna USB. Entrambi i metodi vi daranno un ambiente live che vi permettera' di fare cambiamenti e apparentemente “scriverli su disco”. I cambiamenti in realta' saranno tenuti su un RAM disk, cosicche' un reboot “resettera'” l' OS live al suo stato di default originario. In altr parole, non c'e' persistenza dei dati.

Slackware Live Edition ha due utenti: “root” e “live”. Entrambi hanno una password, e di default queste password sono... avete indovinato: “root” e “live”. Inoltre di default, la ISO fara' il boot a runlevel 4, cioe' avrete un login grafico. Il bootloader vi permettera' di scegliere una lingua non-US e/o

un layout di tastiera a (al boot di sistemi UEFI) un fuso orario customizzato.

Slackware Live Edition si discosta quanto meno possibile dal boot di una normale Slackware. Una volta che avrete passare l' iniziale stadio Liveboot e avviato il vero OS, vi loggherete come utente “live”. Da quel momento in poi vi troverete in un normale ambiente Slackware.

Avviare il Live OS

Boot da BIOS

Slackware Live Edition utilizza syslinux per fare il boot del kernel Linux su computers con BIOS. Per essere precisi, sull' immagine ISO e' installata la variante “isolinux” e la variante “extlinux” e' installata sulla partizione Linux della Live in versione USB.

Syslinux mostra un menu di login grafico con un simpatico sfondo a tema Slackware e diverse opzioni:

- Start (SLACKWARE | KTOWN | XFCE | MATE | DAW) Live (a seconda di quale ISO state avviando)
- Selezione della tastiera non-US
- Selezione di una lingua non-US
- Memory test con memtest86+

Potete scegliere una mappatura della tastiera che corrisponda a quella del vostro computer. Inoltre potete fare il boot di Slackware in un' altra lingua rispetto all' inglese US. Se rimanete con l' interfaccia in inglese US, probabilmente vorrete comunque cambiare il fuso orario perche' di default c'e' l' UTC. Dovete specificare un fuso orario manualmente aggiungendo “tz=VostraArea/VostraLocalita'” perche' il menu di boot di syslinux non offre una selezione dei fusi orari. Syslinux vi permette di modificare la riga di comando di boot premendo <TAB>. Premete <ENTER> per fare il boot dopo aver fatto le vostre modifiche or <ESC> per scartare quanto modificato e tornare al menu.

Boot da UEFI

Sui computer con UEFI, Grub2 gestisce il boot e mostrera' un menu simile (e con tema simile) al menu di Syslinux:

- Start (SLACKWARE | KTOWN | XFCE | MATE | DAW) Live (a seconda di quale ISO state avviando)
- Selezione della tastiera non-US
- Selezione di una lingua non-US
- Selezione di un fuso orario non-US
- Memory test con memtest86+
- Aiuto sui parametri di boot

Editare il menu di Grub prima di fare il boot e' possibile premendo il tasto “e”. Dopo aver fatto le vostre modifiche alla riga di comando di bot, premete <F10> per fare il boot. Per scartare le vostre modifiche, premete <ESC>.

Un' altra differenza tra Syslinux e il menu di Grub2: nel menu di Grub2 potete selezionare una tastiera non-US, lingua e fuso orario e tornerete al menu principale ogni volta. Dovrete comunque selezionare “Start Slackware Live” per far fare il boot al computer. Nel menu Syslinux, solo la selezione della

tastiera vi fara' tornare al menu principale. Qualsiasi selezione di *lingua* non-US, dall' altra parte, vi fara' immediatamente fare il boot nella Slackware Live; senza tornare al menu principale. Questa e' una limitazione di syslinux, che richiederebbe un gran numero di file di menu in piu' per creare un menu con piu' scelte. Grub2 supporta variabili che rendono facile modificare le caratteristiche delle voci del menu.

UEFI Secure Boot

Sui computer con il Secure Boot abilitato, possono essere necessari ulteriori accorgimenti per avviare un sistema operativo. Slackware, per esempio, non riesce ad avviarsi in un computer che abbia il Secure Boot abilitato. Nemmeno le ISO storiche basate su liveslak riescono a fare il boot. Dalla liveslak-1.5.0 in poi, il Secure Boot e' supportato per le immagini ISO a 64-bit.

Il Secure Boot richiede che il primo stadio del bootloader sia firmato con una chiave di cifratura che sia riconosciuta da Microsoft. Per i sistemi operativi basati su Linux, la soluzione ampiamente adottata e' di posizionare un piccolo bootloader prima del normale bootloader di Linux. Questo bootloader EFI e' chiamato 'shim'. Shim deve essere firmato crittograficamente da Microsoft affinche' possa avviare con successo un computer. Questo non e' un processo banale, Microsoft e' molto restrittiva sul processo di firma, in quanto il tuo bootloader firmato potra' sostanzialmente avviare qualsiasi cosa su un computer con Secure Boot abilitato, incluso del malware, se e' stato firmato dalla chiave della tua 'distro'. Cio' creerebbe una grossa falla di sicurezza e annullerebbe lo scopo del Secure Boot.

Anche firmare il tuo bootloader Grub e il tuo kernel diventa mandatorio, perche' 'shim' si rifiuta di caricare binari non firmati. Cio' complica ulteriormente il processo di aggiornamento a un nuovo kernel.

L' OS Slackware Live fa il boot su un computer con Secure Boot abilitato se creato con una liveslak-1.5.0 o superiore, e solamente per le immagini ISO a 64-bit. La distribuzione Slackware Linux non include uno 'shim' firmato da Microsoft, e quindi come aggirare il dilemma della necessita' di uno 'shim' firmato?

Per risolvere cio', la ISO di Slackware Live 'prende in prestito' uno 'shim' di terze parti. Il binario chiamato `bootx64.efi` nella directory `/EFI/BOOT/` e' stato estratto dal pacchetto 'shim' ufficialmente firmato di un' altra distro; di default Fedora, ma anche gli shim di Debian e OpenSUSE sono supportati dallo script `make_slackware_live.sh`. Questo binario 'shim' di terze parti e' stato firmato dalla 'Microsoft UEFI CA' che ti permettera' di fare il boot su ogni computer. Dobbiamo solo “dirgli” che e' OK caricare in memoria il grub di Slackware e il kernel.

Una distro 'shim' come Fedora contiene un certificato SSL della distribuzione embedded e 'shim' si fidera' della firma di ogni binario (grub, kernel, ecc...) che sia stato firmato usando quel certificato. Naturalmente, i binari 'shim' di terze parti non includono un certificato SSL di Slackware. Quindi, deve essere utilizzato un altro mezzo per stabilire la fiducia. Il Secure Boot riconosce il certificato SSL aggiunto nel database MOK (Machine Owner Key) del computer come valido. 'shim' si fida dei certificati SSL custom dei binari firmati, se sono presenti nel database MOK. E' compito dell' utente (Il proprietario della macchina - Machine Owner) aggiungere un certificato personalizzato a quel database.

Le immagini del Grub e del kernel della Slackware Live Edition sono firmate con un certificato SSL 'Alien BOB' e una chiave privata. Questo certificato SSL ha bisogno di essere aggiunto al database MOK del tuo computer con Secure Boot abilitato. Tutte le ISO liveslak usano questo specifico certificato piu' la sua chiave privata associata. La chiave privata naturalmente non dovra' mai essere

distribuita, ma una versione codificata DER del certificato pubblico e' distribuita come parte della ISO. Potrai trovarlo come `/EFI/BOOT/liveslak.der` dentro la ISO. Su una penan USB persistente creata dalla ISO, questo file sara' sulla seconda partizione (la ESP).

Aggiungere il certificato ''liveslak.der'' al database MOK

Ci sono due modi di aggiungere questo certificato.

- Quando fate il boot di una liveslak ISO con Scure Boot abilitato per la prima volta, 'shim' fallira' la validazione del certificato del Grub della liveslak. Avviera' quindi il 'MokManager' mostrandovi un simpatico schermo blu con una finestra che vi richiedera' di aggiungere una chiave pubblica (anche detta certificato SSL) da disco. Potete usare il selettore di file per navigare nella paritizione 'efi' fino all directory `./EFI/BOOT`. Selezionate la `liveslak.der` e confermate che questo e' il certificato corretto. Il computer a quel punto si riavviera', e dopo il reboot finirete automaticamente nel menu di boot del Grub senza bisogno di intervenire ulteriormente.
- Se avete gia' un altro sistema Linux su quel computer, potrete usare il programma `mokutil` per aggiungere la chiave prima di fare il boot della ISO liveslak:

  ```
  # mokutil --import liveslak.der
  ```

  . Questo comando programmera' una richiesta a shim, e la prima volta che farete il boot di una ISO liveslak il MokManager vi chiedera' conferma sull' aggiunta della chiave prestabilita. In altre parole, non avrete bisogno di fare l' aggiunta da disco.

Tenete conto che l' aggiunta di uan chiave MOK e' un' azione da fare una volta sola per la ISO ufficiale basata su liveslak. Tutte le liveslak che installerete in futuro saranno anch'esse firmate utilizzando questo certificato `liveslak.der`, e fintantoche' esso stara' nel database MOK del vostro computer, 'shim' carichera' il Grub e il kernel senza lamentarsi.

Tenete anche conto che potete crearvi il vostro certificato SSL piu' chiave privata e usarli per generareimmagini ISO customizzate della liveslak con supporto a Secure Boot. Tutto cio' che dovrete fare e' aggiungere la chiave pubblica (la versione codificata DER del vostro certificato SSL) nel database MOK del vostro computer. Il database MOK ha spazio per molteplici chiavi, per cui sia le vostre che quelle della liveslak (e altre) ci staranno.

Boot da un file ISO su disco

Se avete scaricato un file ISO liveslak e volete fare il boot di quella ISO direttamente dalla sua posizione sul disco del computer, potete usare il questo blocco di configurazione del Grub da aggiungere al vostro file `/boot/grub/grub.cfg.`:

```
menuentry "LIVESLAK ISO" --class gnu-linux --class os --class icon-linux {
  set iso='/data/ISOS/slackware64-live-xfce-current.iso'
  set bootparms='load_ramdisk=1 prompt_ramdisk=0 rw printk.time=0 kbd=us
tz=Europe/Amsterdam lang=nl'

  search -f $iso --set=root
  loopback loop $iso
```

```
  linux (loop)/boot/generic livemedia=scandev:$iso $bootparms
  initrd (loop)/boot/initrd.img
}
```

Questo esempio aggiungera' una voce 'LIVESLAK ISO' a quelle del menu della vostra macchina locale, attraverso cui potrete avviare una Live ISO con XFCE scaricata, preconfigurata per una tastiera US, lingua Olandese fuso orario di Amsterdam.

Trasferire il contenuto di una ISO su una penna USB

E' disponibile uno script che vi permettera' di trasferire il contenuto di una immagine ISO su una penna USB, facendo qualche modifica a secondo dei parametri dello script.

Lanciando questo scritp la penna USB sara' cancellata e riformattata (eccetto quando si usa l' opzione refresh '-r')! Prima di fare danni irreversibili, lo script vi mostrera' un prompt, e quel punto potrete valutare se sia sicuro continuare.

Questo script, chiamato 'iso2usb.sh' accetta i seguenti parametri:

```
  -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
                             (integer numbers only).
                             Examples: '-c 125M', '-c 2G', '-c 20%'.
  -d|--devices               List removable devices on this computer.
  -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.
                             If it does not exist yet, create it manually.
  -r|--refresh               Refresh the USB stick with the ISO content.
                             No formatting, do not touch user content.
  -s|--scan                  Scan for insertion of new USB device instead of
                             providing a devicename (using option '-o').
  -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).
                             Format for size/percentage is the same
                             as for the '-c' parameter.
  -P|--persistfile           Use an unencrypted 'persistence' file instead
                             of a directory (for use on FAT filesystem).
```

Esempi:

- Creare una versione USB di Slackware Live, in cui la penna USB e' conosciuta al sistema come '/dev/sdX'. Nota - il valore del parametro di output e' il nome di dispositivo della penna e non una delle sue partizioni!

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

- Creare una USB Live come all' esempio sopra, ma questa volta aggiungendo un filesystem /home criptato con 750MB di spazio, e al tempo stesso aumentare il tempo di attesa al boot a 15 secondi (utile per i media USB lenti che fallirebbero il boot del sistema altrimenti):

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

- Creare una USB Live con una /home criptata (allocando il 30% dello spazio libero della chiavetta per /home) e in cui i dati persistenti saranno contenuti in un file invece che in una directory:

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

- Creare una USB Live con sia la /home che i dati persistenti criptati (il filesystem di persistenza sara' di 300MB):

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

- Fare i refresh dei moduli di sistema su una USB Live utilizzando una ISO come sorgente. Lasciare che lo script faccia la scansione per rilevare una USB inserita invece di specificare il nome a riga di comando. Tenete conto che addons e moduli opzionali non saranno modificati da questa azione:

```
# ./iso2usb.sh -i slackware64-live-current.iso -r -s
```

Potreste aver notato che il parametro “-P” non accetta come argomento la dimensione. Questo perche' il contenitore cifrato e' creato come 'sparse' file che parte da una dimesione zero e aumenta dinamicamente fino ad un massimo del 90% dello spazio libero iniziale sulla partizione Linux della penna USB.

Utilizzare il sistema Live per installare Slackware sull' hard disk

Tutte le varianti di Slackware Live Edition contengono uno script “setup2hd”, una versione modificata del normale programma di setup di Slackware. Lo script “setup2hd” supporta la normale installazione da rete di Slackware. In aggiunta vi permette di installare sull' hard disk del vostro computer la release di Slackware su cui il Live OS e' basato. Dovete fare il boot dell' OS Live innanzitutto, e poi lanciare `setup2hd` da un terminale X nel vostro ambiente desktop grafico (runlevel 4), oppure da una console in runlevel 3. Il fatto che possiate lanciare “setup2hd” da un terminale grafico implica che durante l' installazione voi potrete continuare a navigare su internet, ascoltare musica, guardare video, leggere un e-book o qualunque altra cosa che vi faccia passare il tempo.

Lo script “setup2hd” ha alcune funzionalita' che mancano nell' originale 'setup' script di Slackware:

- Lancia fidsk/gdisk se per caso avete dimenticato di creare le partizioni Linux precedentemente;
- Permette di creare un normale account utente e impostare la sua password;
- Vi propone di impostare la password di root in una finestra grafica.

Aggiornare il kernel (e altro) su una penna USB

E' disponibile uno scritp che permette di modificare il contenuto di una penna USB con la Live.

Nello specifico, lo script e' in grado di:

- Aggiornare il kernel e i moduli, facendo un backup del vecchio kernel e dei vecchi moduli.
- Fare il restore del vecchio kernel e dei vecchi moduli se il nuovo kernel non dovesse funzionare.
- Aggiungere i moduli per il supporto del boot PXE da rete (in caso mancassero).
- Aumentare (o diminuire) il tempo di attesa al boot della USB.
- Rimpiazzare lo script di init della Live dentro l' immagine initrd con un nuovo script da voi fornito.
- Spostare i dati persistenti su un nuovo modulo squashfs negli 'addons', reinizializzando poi il nuovo spazio di persistenza. Il nome del nuovo modulo ha la data (/liveslak/addons/0099-slackware__customchanges-yymmddHHMMSS.sxz), cosi' che la stessa azione puo' essere ripetuta molte volte.

Lo script e' pensato per essere usato dalla stessa USB da cui state facendo girare la Slackware Live, tuttavia cio' non e' mandatorio. Con l' eccezione dell' opzione '-p', che sposta la persistenza dei dati su un modulo squashfs, le sue funzionalita' possono essere usate su ogni computer Linux in cui potete inserire la chiave USB.

Prima di fare qualsiasi modifica, lo script vi mostrera' un prompt, da cui potrete valutare se sia sicuro procedere oltre.

Questo script, chiamato 'upslak.sh', accetta i seguenti parametri:

```
  -b|--nobackup              Do not try to backup original kernel and
modules.
  -d|--devices               List removable devices on this computer.
  -h|--help                  This help.
  -i|--init <filename>       Replacement init script.
  -k|--kernel <filename>     The kernel file (or package).
  -m|--kmoddir <name>        The kernel modules directory (or package).
  -n|--netsupport            Add network boot support if not yet present.
  -o|--outdev <filename>     The device name of your USB drive.
  -p|--persistence           Move persistent data into new Live module.
  -r|--restore               Restore previous kernel and modules.
  -s|--scan                  Scan for insertion of new USB device instead of
                             providing a devicename (using option '-o').
  -v|--verbose               Show verbose messages.
  -w|--wait<number>          Add <number> seconds wait time to initialize
USB.
```

Esempi:

- Avere un elenco di tutti i dispositivi rimovibili sul computer:

./upslak.sh -d

- Aggiornare il kernel e i moduli, fornendo due pacchetti come input e assumendo che la penna

USB sia riconosciuta come /dev/sdX:

```
# ./upslak.sh -o /dev/sdX -m kernel-modules-4.19.0-x86_64-1.txz -k kernel-generic-4.19.0-x86_64-1.txz
```

- Restorare il precedente kernel e i precedenti moduli dopo un aggiornamento andato male, e far si' che lo script faccia lo scan del vostro computer per rilevare la vostra USB:

```
# ./upslak.sh -s -r
```

- Rimpiazzare lo script di init della Live con l' ultimo template preso dal repository git:

```
# ./upslak.sh -o /dev/sdX -i liveslak/liveinit.tpl
```

Fare il boot della Live OS da PXE

La Slackware Live Edition puo' fare un boot dare rete utilizzando il protocollo PXE da un NFS esportato. Estraete il contenuto di una ISO su una nuova directory chiamata (per esempio) `slackware-live` sotto la directory `tftpboot` del vostro server TFTP ed esportatela via NFS. Aggiungete poi delle linee come questa al vostro file `pxelinux.cfg/default` (assumendo che il vostro server NFS abbia indirizzo 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
```

Come mostrato nell' esempio sopra, viene utilizzato un parametro di boot `nfsroot` per il boot da rete. Il valore del parametro definisce l' indirizzo IP del vostro server NFS e il percorso della directory NFS esportata in cui avete estratto la Live ISO Slackware. Suggerimento: per avere una lista delle share di rete esportate sul vostro server, lanciate `showmount -e localhost` sul server NFS.

In realta', sono due i parametri disponibili per supportare adeguatamente il boot da rete. un secondo parametro `nic` puo' essere utilizzato per definire le caratteristiche della configurazione di rete del tuo ambiente Live, come il nome dell' interfaccia di rete, indirizzo IP statico e cosi' via. Se sei su una rete dove un server DHCP configura i client, allora il parametro `nic` non sara' necessario, dato che la Slackware Live Edition recuperera' tutti i dettagli per suo conto.

Sintassi di questi due parametri:

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

Esempi d' uso dei due parametri di rete al boot:

```
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
```

Dopo che hai fatto il setup corretto del tuo ambiente PXE (DHCP, TFTP e sderver NFS) utilizzando le informazioni riportate sopra, fai il boot di uno dei tuoi computer che supportino il PXE, interrompi il boot e seleziona “boot da rete” e digita o seleziona la label corretta (nell' esempio sopra sarebbe `liveslak`). Vedrai il kernel e l' initrd venire scaricati e avviati, e successivamente l' OS Live partira' come se fosse stato lanciato in locale.

Se il tuo server DHCP ci impiega troppo a rispondere alla richiesta del client, il DHCP del client andra' in timeout e il boot della Live fallira' perche' il filesystem della Live montato in NFS non risultera' disponibile. In quel caso puoi provare ad aumentare il tempo di attesa prima che il client DHCP decida di non essere riuscito ad avere l' indirizzo IP dal server. Aggiungi il parametro di boot `dhcpwait=30` (valore d'esempio) dove il numero `30` sono i secondi entro i quali il client DHCP dovrebbe aspettare la risposta del server. Dovresti naturalmente scegliere un valore che sia abbastanza in base al setup della tua rete.
Il tempo di attesa di default per la Live OS e' di 20 secondi.

In questa configurazione non e' supportata la persistenza; allo stato attuale l' overlayfs non supporta NFS come uno strato scrivibile nel filesystem live.

Server PXE

Slackware Live Edition non e' solo capace di fare il boot come client PXE; e' anche capace di far girare un server PXE per suo conto.

Cosa significa?

Un esempio pratico potrebbe essere che voi portiate una penna USB con Slackware Live Edition ad un LAN party, utilizzarla per fare il boot di uno dei computers e da li' tutti gli altri computers nella LAN (cablata) saranno in grado di fare un boot da rete e lanciare la stessa Slackware Live Edition un paio di minuti piu' tardi. Il computer con la penna USB agira' come server PXE e tutti gli altri computers saranno i suoi client PXE, leggendo i dati di Slackware da quella penna USB. I clients acquisiranno il fuso orario del server, lingua e tastiera di default, ma cio' puo' essere sovrascritto. I clients PXE non avranno la 'persistenza'. Se il server ha accesso ad Internet, anche i clients lo avranno.

Come avviare un server PXE?

Quando fai il boot della Live OS puoi avviare uno script “pxeserver” dalla console in runlevel 3 o da un terminale X in runlevel 4. Lo script recuperera' tutte le informazioni richieste e se non e' in grado di risolvere qualcosa di suo ti fara' delle domande. Se non e' in grado di capire quale interfaccia di rete cablata debba usare, puoi aggiungere il nome della tua interfaccia (per esempio, `eth1`) come singolo parametro dello script quando lo lanci.

Il server PXE usa dnsmasq per offrire i DNS ai clients PXE. Il programma dnsmasq abilitera' il suo server DHCP interno se la tua LAN non dovesse avere un suo DHCP server. Dnsmasq inoltre avviera' un server TFTP a cui i clients PXE si connetteranno in modo da recuperare i files di boot (kernel e initrd). Lo script `pbxserver` inoltre avviera' un server NFS che sara' usato dall' initrd della Live per ottenere i moduli squashfs e avviare il Live OS. Se il tuo server PXE ha piu' di una interfaccia di rete, per esempio un' interfaccia wireless che e' connessa al mondo esterno e una interfaccia cablata connessa ad un altro computer che diventera' un client PXE (o in realta' connesso a uno switch insieme ad un gruppo di futuri clients PXE dietro esso), allora il server PXE abilitera' il forwarding dei pacchetti cosicche' i clients PXE saranno in grado di accedere al mondo esterno attravero l' interfaccia cablata e uscendo dall' altra interfaccia.

Se hai piu' di una interfaccia di rete e' importante sapere che dnsmasq si leghera' solamente all' interfaccia a cui vuoi che i clients PXE si connettano. In una situazione con piu' interfacce, in cui una seconda interfaccia e' connessa al mondo esterno (la tura rete locale), questo significa che il server DHCP/DNS avviato da dnsmasq non interferira' con un eventuale DHCP esistente sulla rete locale.

Una volta che il server PXE sta girando, lo script vi mostrera' il log dell' attivita' di dnsmasq in una finestra dialog, cosicche' voi possiate monitorare i clients PXE che si connettono.

Se il tuo server PXE ha RAM a sufficienza, e' fortemente raccomandato di avviare la Live OS da penna USB con il parametro `toram`. Quando piu' di una manciata di clients PXE cominciano a leggere i files del sistema operativo dal server PXE, la penna USB diverrebbe un collo di bottiglia. Far girare l' OS del server dalla RAM risolvera' il problema di un simile collo di bottiglia.

I parametri di boot spiegati

Premi <F2> nella scheramta di boot di syslinux per una panoramica della (maggior parte) dei parametri di boot. Quando avvi Grub, seleziona il menu “Help on boot parameters” invece. L' help di Grub e' fatto male, lo so, ma Grub non offre niente di meglio.

I seguenti parametri sono ricoosciuti dalla Slackware Live Edition. Per avviare coi parametri di default premi semplicemente ENTER.

Ambiente Desktop

0|1|2|3|4|5|6|S|s|single ⇒

```
Seleziona un runlevel da cui cominciare.
Il default e' 4 per il login grafico.
```

kbd=fr xkb=ch,fr ⇒

```
Esempio di customizzazione del layout della tastiera in X.
Il parametro xkb puo' essere impostato a "XkbLayout,XkbVariant,XkbOptions".
Il menu di boot configurera' alcuni di questi al posto vostro, ma potete
sempre modificare i valori naturalmente.
Notate che l' opzionale XkbOptions puo' consistere di diversi valori
separati da virgola
I valori per XkbLayout e XkbVariant non devono contenere virgole.
Potete impostare solo XkbVariant agguingendo qualcosa come "kbd=ch xkb=,fr"
```

livepw=“somestring” ⇒

```
Cambia la password per l' utente "live".
La password e' passata in chiaro.
```

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

```
Esempio della customizzazione di lingua,
```

```
tastiera e fuso orario.
```

rootpw=“somestring” ⇒

```
Cambia la password per l' utente "root".
La password e' passata in chiaro.
```

Personalizzazione del Software

load=nvidia ⇒

```
Carica e configura i drivers Nvidia se disponibili
nella ISO (di default non per le varianti SLACKWARE e XFCE).
```

load=mod1[,mod2[...]] ⇒

```
Carica uno o piu' moduli squashfs
dalla directory "/liveslak/optional".
Di default nessuno di questi moduli "opzionali" e' caricato al boot.
```

noload=mod1[,mod2[...]] ⇒

```
Previene il caricamento di uno o piu'
moduli squashfs dalla directory "/liveslak/addons".
Di default nessuno di questi moduli "addon" e' caricato al boot.
```

Boot da Rete

dhcpwait=<numseconds> ⇒

```
Tempo massimo di attesa per il client DHCP per configurare un' interfaccia
di rete
(default: 20 seconds).
```

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

```
definisce l' indirizzo IP
del server NFS, e il percorso del contenuto estratto
della Slackware Live Edition.
```

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

```
personalizzazione della scheda di rete, solitamente questo parametro
non e' necessario quando nella tua rete c'e' un server DHCP.
Specificare un driver se UDEV non dovesse rilevare l' interfaccia.
Specificare
l' interfaccia se la Slackware Live non dovesse riuscire. Se specificate
```

```
'static' dovrete inoltre specificare ipaddr e netmask. Il Gateway e'
opzionale
ma e' necessario per esempio per accedere ad Internet.
```

Relativi all' Hardware

localhd ⇒

```
inizializza RAID/LVM sull' hard disk locale.
```

tweaks=tweak1[,tweak2,[,...]] ⇒

```
Modifiche implementate:
nga - nessuna affascinante accelerazione 2D, evita errori
"EGL_MESA_drm_image required".
nsh - nessun sub-pixel hinting 'nuovo stile' in freetype.
tpb - abilita il TrackPoint scrolling mentre si tiene premuto tasto centrale
del mouse.
syn - avvia il syndaemon per un miglior supporto dei touchpads Synaptics.
ssh - avvia il server SSH (disabilitato di default).
```

nomodeset ⇒

```
Fa il boot senza il kernel modesetting, necessario su
aclune macchine.
```

rootdelay=10 ⇒

```
Aggiunge 10 secondi di ritardo per dare al kernel piu' tempo per
inizializzare l' USB. Provatelo se il boot dovesse fallire. il default e' 5.
```

swap ⇒

```
Permette alla Live OS di attivare tutte le partizioni di swap sull' hardware
locale. Di default nessuna swap viene toccata.
```

Modifiche ai Media

cfg=[skip|write] ⇒

```
Specificare 'skip' per saltare i file di configurazione basati su disco,
contenenti
parametri dell' OS; oppure specificare 'write' per scrivere i parametri
dell' OS correnti su disco.
```

domain=your_custom_domain ⇒

```
Specificare un nome di dominio customizzato. Il default e' 'example.net'.
```

hostname=your_custom_hostname[,qualifier] ⇒

```
Specificare un hostname personalizzato. Un qualificatore 'fisso' puo' essere
appeso
per proibire la modifica dell' hostname in caso di boot da rete.
```

livemedia=/dev/sdX ⇒

```
Dice allo script di init quale partizione contiene
la Slackware Live OS che volete avviare. Questo puo'
diventare necessario se avete un' altra copia di Slackware Live installata
su un' altra partizione. Accettati anche: UUID o LABEL.
```

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

```
Usate questo se volete caricare
la live OS da un file ISO su una partizione dell' hard disk locale.
```

livemedia=scandev:/path/to/live.iso ⇒

```
Usando questo se la liveslak dovrebbe
cercare su tutte le partizioni per localizzare il file ISO.
```

livemain=directoryname ⇒

```
Usate questo se avete copiato il contenuto della ISO
su una directory diversa da "liveslak".
```

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

```
Monta un container LUKS di nome "file1" sul mount point "/mountpoint1"
nel fs della Live. File multipli vanno separati da una virgola.
Specificate "luksvol=" per *prevenire* il mount di qualsiasi container LUKS,
inclusa una /home criptata.
```

nop ⇒

```
Nessuna persistenza, cioe' avvia l' installazione "vergine" in caso
La vostra directory "persistente" risultasse corrotta.
Se volete ignorare ogni dato persistente durante il boot,
inclusi i dati LUKS, specificate "nop luksvol=" .
```

nop=wipe ⇒

```
Eliminate tutti i dati dalla directory di persistenza o da container.
Utile in caso i vostri dati persistenti fossero corrotti.
```

persistence=name ⇒

```
Impostate questo se state usando un file/directory diversi
da "persistence" per contenere i dati persistenti.
```

persistence=/dev/sdX:/path/to/mypersistence persistence=scandev:/path/to/mypersistence ⇒

```
Usate questo se la directory di persistenza o il container non e' sulla
penna USB, ma su una partizione locale del disco fisso.
Utile per il boot da rete (PXE) in cui volete offire agli utenti la
persistenza.
```

toram ⇒

```
Copiate il sistema operativo dal media alla RAM prima di avviarlo.
Una volta terminato il boot potete rimuovere il media.
```

toram=all ⇒

```
Previene le scritture su disco dato che si suppone che funzioni in RAM;
equivalente al parametro "toram".
```

toram=core ⇒ Carica i moduli della Console nella RAM. Slackware con sola Console

```
si carica piu' velocemente, contiene 'setup2hd' e libera la vostra USB
cosicche' potete
sovrascriverla con un Live OS persistente.
```

toram=os ⇒

```
Carica i moduli OS nella RAM, ma scrive dati persistenti sulla USB.
```

Troubleshooting

blacklist=mod1[,mod2[...]] ⇒

```
Aggiunge uno o piu' moduli del kernel
alla blacklist per prevenirne il caricamento, in caso
dovessero causare problemi durante le operazioni.
```

debug ⇒

```
Durante l' init, fermati in "posti" strategici mentre assembli il filesystem
overlay e mostra le informazioni di mount.
```

debug=<number> ⇒

```
'2' abilita la scrittura verbosa delle operazioni;
'4' fa il dump in una shell di debug proprio prima di dello switch_root.
```

rescue ⇒

```
Dopo l' inizializzazione, sarai riportato a una shell di rescue
per fare manutenzione a basso livello.
```

Layout della ISO

La ISO Live contiene tre directory nella root del suo filesystem:

- EFI/
- boot/
- liveslak/

La variante USB con persistenza puo' avere una directory aggiuntiva nella root:

- persistence/

- La directory EFI/ contiene la configurazione del Grub usata quanto fai il boot del Live OS su un computer con UEFI.
- La directory boot/ contiene la configurazione del syslinux usata quando fai il boot del Live OS su un computer con BIOS. Questa directory contiene inoltre il kernel e i files dell' initrd che sono usati in realta' per avviare il sistema operativo.
- La directory liveslak/ contiene tutti i moduli squashfs che sono usati per assemblare il filesystem della Live OS, allo stesso modo dei files che sono copiati direttamente nella root del filesystem della Live. Contiene le seguenti sottodirectory:
  - addons/ - i moduli che sono contenuti in questa directory saranno sempre aggiunti al filesystem Live a meno che voi non li escludiate con un parametro “noload=” al boot;
  - optional/ - i moduli che sono contenuti in questa directory non saranno aggiunti al filesystem Live a meno che voi non ne forziate il caricamento con il parametro “load=” al boot;
  - system/ - questa directory contiene tutti i moduli che sono stai creati dallo script “make_slackware_live.sh”. Tutti questi moduli sono numerati e lo script di init della Live usera' quella numerazione per assemblare il filesystem Live nello stesso ordine di come essi sono stati creati inizialmente.
  - rootcopy/ - questa directory e' vuota di default. Qualunque cosa (l' utente della ISO) aggiungiate a questa directory sara' copiato testualmente nel filesystem dallo script di init quando la Live OS fa il boot. Potete usare questa feature per esempio se non volete creare un modulo squashfs separato per i file di configurazione della vostra applicazione.

Documentazione per lo Sviluppatore

Scripts e tools

make_slackware_live.sh

Il primo script:

Lo script “make_slackware_live.sh” crea un file ISO che come output contiene la Live OS. Grazie al

kernel Linux 4.x e al pacchetto squashfs-tools in Slackware, il processo di creazione di una Live ISO di Slackware **non** richiede (ri)compilazione del contenuto di Slackware o l' installazione di pacchetti di terze parti.

Il funzionamento interno dello script puo' essere suddiviso in diversi passaggi distinti. Per la completa ISO di Slackware i passaggi del processo sono i seguenti:

Installare i pacchetti di Slackware

Primo stage:

- Lo script legge una sequenza di pacchetti per la variante della Live e installa tutti i pacchetti in questa sequenza nello sottodirectory di un albero di directory temporaneo.
- Ogni set di pacchetti di Slackware (a, ap, d, … , y) o lista di pacchetti (min, noxbase, x_base, xapbase, …) viene installato in una directory 'root' separata.
- Ognuna di queste directory e' “squashata” (usando squashfs) in un modulo squashfs separato. Tale modulo e' un singolo file di archivio contenente la struttura compressa delle directory dei pacchetti installati.
- Questi files dei moduli sono successivamente montati in loop e poi messi insieme in una singola struttuta a directory a sola lettura usando un “overlay mount”. L' overlayfs e' relativamente nuovo; Le distro Live precedenti usavano aufs e unionfs per ottenere simili funzionalita', ma essi non erano parte di nessun sorgente stock del kernel e quindi i kernels custom dovevano essere compilati per tali distro Live.
- Questo “overlay assemblato” e' il filesystem che sara' avviato come filesystem della Live OS.
- In 'cima' a questa serie di filesystems overlay in sola lettura, lo script “make_slackware_live.sh” aggiunge un filesystem scrivibile. Questo filesystem scrivibile e' usato per contenere tutte le customizzazioni alla nostra distro, che vogliamo vengano applicate quando la Slackware Live si avvia. Vedere la prossima sezione per maggiori dettagli.
- Notate che quando fate il boot della Live OS successivamente, un altro overlay scrivibile sara' usato per recepire ogni operazione di scrittura eseguita dall' OS. Voi vedrete la Live Os come un vero sistema operativo che puo' scrivere e cancellare files. Questo filesystem scrivibile sara':
  - un filesystem basato sulla RAM quando la Live OS sara' eseguita senza persistenza.
  - una directory o un container file montato in loop sulla vostra penna USB, se state usando una USB con persistenza.

Configurare il filesystem della Live con utili defaults out-of-the-box

Secondo stage:

- L'inizializzazione di alcuni filesystem e' fatta quando l' overlay e' stato assemblato:
  - gli utenti 'root' e 'live' vengono creati,
  - un ambiente iniziale per gli account viene configurato,
  - l' ambiente desktop e' pre-configuurato per il primo uso,
  - gli scripts della liveslak “makemod”, “iso2usb” e “upslak.sh” sono copiati in “/usr/local/sbin/” nella ISO per vostra comodita',
  - lo script “setup2hd” e i files dell' installer della Slackware sono copiati rispettivamente in “/usr/local/sbin” e “/usr/share/liveslak”.
  - Viene configurato slackpkg,
  - viene creato un database locale,

    - ecc…
- Tutte queste modifiche sono scritte nel filesystem scrivibile che e' stato creato nella precedente sezione. Questo filesystem sara' anche contenuto della ISO come modulo squashfs e, quando il Live OS si avviera', sara' montato in sola lettura come tutti gli altri moduli. Il suo nome sara' “0099-slackware_zzzconf-current-x86_64.sxz” o, piu' genericamente, “0099slackware_zzzconf-${SLACKVERSION}-${ARCH}.sxz”

Configurare la fase di boot della Live OS

Fase tre:

- viene generato un initrd, contenente uno script di “init” modificato (altre distro Live lo chiamano “linuxrc script”) che ri-assembla il filesystem di overlay al boot e configura la Live OS a seconda dei parametri di boot (lingua, tastiera, fuso orario, runlevel, …)
- un kernel gnerico slackware e l' initrd di cui sopra vengono aggiunti a una configurazione syslinux (per computers con BIOS) e a grub2 (per computers con UEFI).

Creazione dell' immagine ISO

Faser quattro:

- una ISO bootabile viene creata usando mkisofs.
- il comando “isohybrid” viene lanciato sulla ISO cosicche' voi potrete copiare la ISO con “dd” o “cp” su una penna USB e di conseguenza creare un media USB bootabile.

Fatto! Potete trovare il file ISO e il suo checksum MD5 nella directory /tmp.

iso2usb.sh

Il secondo script:

L' utilizzo a runtime dello script “iso2usb.sh” e' spiegato in dettaglio in un precedente paragrafo “Trsferire il contenuto della ISO su una penna USB”.

Questa sezione spiega come lo script modifichi la ISO per le funzionalita' aggiuntive della USB.

Layout della chiavetta USB

Lo script “iso2usb.sh” cancella e ri-partiziona la penna USB a meno che non vengano usati i parametri “-r” o *refresh*. Vedi la sezione “Trasferire il contenuto di una ISO su una penna USB” per una spiegazione di tutte le impostazioni da linea di comando.
Lo script creera' 3 partizioni:

- Prima partizione: una piccola (1 MB di dimensione) partizione FAT che non e' usata dalla Slackware Live Edition. Puo' essere usata da un bootloader alternativo se si necessita. Potete anche metterci il vostro keyfile LUKS per sbloccare un computer con Slackware Linux criptato con LUKS (vedete il file README_CRYPT.TXT sul vostro DVD Slackware per maggiori

informazioni sui LUKS keyfile).
- Seconda partizione: una partizione VFAT di 100MB contenente il kernel, l' initrd e tutte le altre cose necessarie a syslinux e grub2 per avviare Slackware Live Edition.
- Terza partizione: una partizione Linux che occupa tutto il resto dello spazio. Conterra' i moduli liveslak, lo spazio di archiviazione persistente e, opzionalmente, la vostra directory home criptata. Potete usare il resto dello spazio libero di questo filesystem Linux *ext4* per metterci tutto quello che volete.

Notate che questo script e' l' unico metodo supportato per trasferire il contenuto della ISO liveslak su una penna USB e di rendere quella stessa USB un Live OS con persistenza. Diversi tool di terze parti (come mulibootusb, rufus, unetbootin), che dichiarano di essere in grado di mischiare diverse Live OS su una singola penna USB e renderle funzionanti in un setup multi-boot, attualmente non supportano liveslak.

Montare un filesystem in un container criptato

Lo script creera' un file di dimensione richiesta nella root della partizione Live usando il comando 'dd'. Il comando 'cryptsetup luksCreate' inizializzera' la cifratura, che fara' in modo che lo script vi chieda “se siete sicuri, digitate YES in maiuscolo”, dopodiche' dovrete inserire una passphrase tre volte (due per l' inizializzazione, una per l' apertura del container). Se il container e' usato per una /home criptata, il nome del suo file sara' “slhome.img”. Lo script copiera' il contenuto esistente della /home della ISO nel filesystem del container che sara' montato piu' tardi sulla /home della ISO (coprendo quindi la /home esistente). Il Live OS e' istruito per decriptare il container e montare il filesystem. Questo e' fatto editando il file “/luksdev” nell' initrd e aggiungendo una linea: “/slhome.img”. Lo script iso2usb.sh supporta solamente la creazione e configurazione di una /home criptata, ma potete creare dei containers criptati aggiuntivi da voi e montarli in altri posti del filesystem della ISO. Affinche' tutto cio' funzioni, dovete editare il file “/luksdev” e aggiungere una linea “/vostro/container.img:/vostro/mountpoint”, cioe'il path del container e la directory target per il mount su una singola linea, separati dai due punti. Lo scritp di init Live creera' un mount point, nel caso sia mancante.

Utilizzare un file container per immagazzinare dati persistenti

Esiste un secondo tipo di container cifrato, che puo' essere usato per immagazzinare i vostri dati persistenti. Lo script di init della Live controllera' se sia necessario abilitare la persistenza in questo ordine:

1. il filesystem USB e' scrivibile? Se si',
    1. esiste una directory /persistence? Se si', usala; altrimenti,
    2. esiste un file /persistence.img? Se si', montalo e, nel caso sia un container cifrato, chiedi la password durante il boot.

Aggiungere il tempo di atteda all' USB

Per media USB lenti, il tempo di attesa di default di 5 secondi durante il boot talvolta e' insufficiente per permettere al kernel di rilevare le partizioni sul vostro dispositivo USB. Lo script puo' opzionalmente aggiungere piu' tempo di attesa. Cio' viene fatto editando il file “wait-for-root” nell'

initrd e aggiornando il valore che e' vi e' contenuto (di default “5” e' scritto la' dallo script “make_slackware_live.sh”).

makemod

Il terzo script:

Lo script “makemod” vi permette di creare un modulo Slackware Live facilmente, con un pacchetto Slackware o un albero di directory come parametro di input.

Utilizzo:

```
# makemod <nome_pacchetto|directory> nome_modulo.sxz
```

- Il primo parametro e' il percorso completo di un pacchetto Slackware, oppure una directory.
  - Se il nome di un pacchetto viene passato come primo parametro, sara' installato in una directory temporanea utilizzando l' “installpkg” di Slackware. Il contenuto della directory temporanea sara' squashato in un modulo dal programma “squashfs”.
  - Se viene passato il nome di una directory, il suo contenuto sara' squashato in un modulo dal programma “squashfs”.
- Il secondo parametro e' il nome del percorso completo del modulo di output che sara' creato.

Potete copiare il modulo che avete appena creato (tenendo a mente le convenzioni sui nomi dei file per i moduli della Slackware Live, vedete il paragrafo “Formato dei moduli della Slackware Live”) nella directory optional/ oppure addon/ del vostro Live OS. Se lo copiate nella directory optional/ o addon/ dei sorgenti della liveslak, allora “make_slackware_live.sh” usera' il modulo quando creera' l' immagine ISO.

setup2hd

Il quarto script:

Lo script “setup2hd” e' un installer di Slackware modificato, in modo da farvi trovare a vostro agio con il processo. La sezione 'SOURCE' offre due tipi di scelta: una normale installazione di Slackware da rete, utilizzando un server NFS, HTTP, FTP o Samba, oppure l' installazione della live che state utilzzando. Lo script sa dove trovare i moduli squashfs, cosicche' la selezione “Install Live OS” non vi chidera' di inserire ulteriori inputs.

- L' installazione di rete di Slackware e' identica a quella del medium ufficiale di Slackware.
- Se scegliete di installare la Live OS, allora dopo che selezionerete la/e partizione/i oggetto, ogni modulo attivo della specifica variante della Live OS (SLACKWARE, KTOWN, MATE, ...) sara' estratto sul disco fisso.

Dopo che l' estrazione sara' completata, lo script fara' il sommario di quanti moduli sono stati estratti. Mostrera' inoltre un comando di esempio per estrarre ogni modulo inattivo o disabilitato rimanente a mano. Lo step finale nell' installazione e' ancora l' installer stock di Slackware, il quale fara' partire gli scripts di configurazione di Slackware.

pxeserver

Il quinto script:

Lo script `pxeserver` funziona nel seguente modo:

- Ha bisogno di una rete cablata; il boot PXE da wireless e' impossibile.
- Lo script pxeserver provera' a trovare un' interfaccia di rete cablata; potete esplicitamente passargli un nome di interfaccia come parametro allo script (opzionale).
- Se vengono rilevate interfacce di rete cablate multiple, una finestra dialog chiedera' all' utente di scegliere quella giusta.
- Per l' interfaccia scelta verra' fatto un check della configurazione con DHCP;
  - Se verra' trovata una configurazione DHCP, allora il pxeserver non attivera' il suo DHCP, affidandosi invece al DHCP server della vostra rete LAN.
  - Se non verra' trovata nessuna configurazione DHCP, allora lo script vi chiedera' il permesso di avviare il suo DHCP server interno. Inoltre all' utente sar'a chiesto di configurare un indirizzo IP per l' interfaccia di rete e le proprieta' della classe IP per il server DHCP interno.
- Lo script a quel punto avviera' il server PXE, costituito da:
  - dnsmasq che fornira' i DNS, DHCP e BOOTP;
  - demoni NFS e RPC;
- Lo script rilevera' se avete una connessione di rete in uscita su un' altra interfaccia e abilitera' l' inoltro dei pacchetti IP se necessario, cosicche' i clients PXE avranno anch' essi accesso alla rete.
- La Live OS avviata via pxelinux e' configurata con parametri di boot aggiuntivi:

```
nfsroot=<server_ip_address>:/mnt/livemedia
luksvol=
nop
hostname=<distroname>
tz=<server_timezone>
locale=<server_locale>
kbd=<server_kbd_layout>
```

  Che mostrano che la configurazone della Live OS dove il server PXE gira e' largamente determinante per la configurazione dei clients PXE.

- Notate che quando fate il boot da rete, l' hostname della Live OS avra' come suffisso l' indirizzo MAC della macchina, in modo da rendere unico l' hostname di ogni computer che faccia il boot da rete.

upslak.sh

Il sesto script:

L' utilizzo a runtime dello script “upslak.sh” e' spiegato nel dettaglio in un paragrafo precedente “Aggiornare il kernel (e altro) su una penna USB”.

Questa sezione spiega come lo script modifica il contenuto della penna USB Live.

Quando lo script e' avviato, fara' alcuni controlli di di integrita' e poi estrarra' il contenuto dell' immagine initrd. Alcune caratteristiche dell' initrd saranno registrate:

- sara' verficata l' esistenza di moduli del kernel precedentemente backuppati,
- saranno recuperate le variabili dei template, coi loro valori dallo script di init,
- sara' verificato il tempo di attesa dell' USB corrente.

A seconda dei parametri passati allo script, esso eseguira una o piu' delle seguenti azioni:

Aggiornare il kernel e i moduli

Potete fornire un nuovo kernel e i suoi moduli in due modi. L' opzione '-k' accetta un file immagine del kernel oppure un pacchetto di Slackware contenente un kernel. L' opzione '-m' accetta un albero di directory dei moduli sotto “/lib/modules”, oppure un pacchetto Slackware contenente moduli del kernel. Se c'e' sufficiente sapzione sulle partizioni Linux e EFI, lo script fara' un backup del kernel e dei moduli correnti rinominando la directory che li contiene con un suffisso “.prev”. Spazio sufficiente significa che devono rimanere almeno 10 MB di spazio libero sulla/e partizione/i dopo aver fatto il backup e installato il nuovo kernel piu' i moduli. Se lo spazio e' un problema, potete saltare il backup fornendo il parametro '-b' allo script (come possibile scelta insicura).

Restorare kernel e moduli backuppati

Se e' stato fatto un backup del kernel e dei moduli, lo script upslak.sh e' in grado di restorarli usando l' opzione '-r', rimuovendo quindi quelli che li avevano rimpiazzati. Questo torna utile quando il kernel nuovo si rivela non funzionante.

Aggiungere i moduli per il supporto di rete

Questo noromalmente non dovrebbe servire. Di default, tutte le immagini ISO liveslak hanno il supporto per la rete integrato. Ma le immagini ISO Live customizzate possono essere fornite senza supporto inizialmente. Se volete che la vostra Live OS sia bootabile da PXE avrete bisogno il supporto della rete nel kernel. Utilizzate l' opzione '-n'.

Aumentare (o diminuire) il tempo di attesa della USB

Similmente alla funzionalita' dello script “iso2usb.sh”, lo script “upslak.sh” e' in grado di alterare il tempo di attesa dell' USB al boot, utilizzando l' opzione '-w'.

Rimpiazzare lo script di init della liveslak

Lo script di init dentro l' immagine initrd e' il nucleo della liveslak. Quello script di init prepara il filesystem della Live e configura diversi paramentri del runtime dell' OS. Se avete fatto delle modifiche a questo script di init potete facilmente rimpiazzare l' init script di default con il vostro, utilizzando l' opzione '-i'. Lo script “upslak.sh” e' abbastanza smart da riconoscere un template

liveslak come input. L' estensione “.tpl” di alcuni file della liveslak significa che questi sono templates. Non sono utilizzabili cosi' come sono, perche' contengono delle stringhe segnaposto come “@VERSION@” o “@DISTRO@” che necessitano prima di essere espanse con valori reali. Lo script “upslack.sh” si prendera' cura di queste sostituzioni.

Contenere i dati persistenti in un nuovo modulo squashfs

I dati persistenti si accumuleranno lungo il tempo sulla penna USB. Questo e' perfettamente normale, e potete cancellarli al boot se volete. Ma talvolta potreste voler prendere i pacchetti installati nello storage persistente e creare da essi un nuovo modulo squashfs. Lo script “upslak.sh” e' in grado di di spostare i vostri dati persistenti in un nuovo modulo squashfs, usando l' opzione '-p'. Il nuovo modulo sara' creato nella directory “/liveslak/addons/”, cosicche' esso sara' caricato nella Live OS ogni volta che la Live USB si avviera'. Dopo la creazione del nuovo modulo, lo storage di persistenza sara' reinizializzato (cioe' il suo contenuto sara' cancellato al successivo boot). Il nome del nuovo modulo avra' un timestamp (/liveslak/addons/0099-slackware_customchanges-yyyymmddHHMMSS.sxz dove yyyymmddHHMMSS e' il timestamp) cosicche' questa azione possa essere ripetuta tutte le volte che volete.

Creare una ISO Live da zero

Creare un' immagine ISO di Slackware Live Edition richiede che voi stiate usando Slackware 14.2 o piu' recenti (64-bit). Le release piu' vecchie di Slackware hanno un kernel che e' troppo vecchio per supportare l' uso che la liveslak fa della funzionalita' del kernel “overlayfs”, ed e' mancante dei tools per lo squashfs. Allo stesso modo, una Slackware Live Edition puo' essere creata solamente da una Slackware 14.2 o piu' recenti.

Avrete anche bisogno della collezione di script “liveslak” che puo' essere scaricata da un qualsiasi link in fondo a questa pagina.

Liveslak e' un albero di directory che contiene gli scripts, i bitmaps e i file di configurazione. Solamente 6 scripts sono pensati per essere lanciati da voi, gli utenti. Questi scripts (“make_slackware_live.sh”, “iso2usb.sh”, “makemod”, “setup2hd”, “pxeserver” e “upslak.sh”) sono spiegati in maggior dettaglio nella sezione “Scripts e tools” qui sopra. Quando si crea una ISO Live da zero, avrete bisogno solo di lanciare lo script “make_slackware_live.sh”.

Layout dei sorgenti della Liveslak

La radice della directory 'liveslak' contiene le seguenti sottodirectory:

- EFI/ - contiene la struttura per il suppoorto al boot sui computers UEFI. Alcune dei file di configurazione UEFI sono dinamicamente generati dallo script “make_slackware_live.sh”.
- README.txt - documentazione.
- addons/ - i moduli messi in questa directory saranno caricati nel filesystem Live al boot dell' OS.
- contrib/ - script aggiuntivi che non sono usati direttamente per la creazione e l' uso della ISO Live.
- graphics/ - i moduli squashfs per il supporto di GPU propirietarie (Nvidia) possono essere messi qui. I moduli saranno copiati nella directory addons/ dallo script “make_slackware_live.sh” per

ogni ambiente Desktop Live (eccetto il puro Slackware) che possa beneficiare dal supporto driver proprietario.
- local64/ , local/ - queste directory possono contenere pacchetti Slackware considerati 'locali', cioe' non appartenenti a nessun repository. I pacchetti saranno convertiti in moduli squashfs dallo script “make_slackware_live.sh”, copiati nella subdirectory “addons/” nella ISO e caricati nel filesystem Live quando l' OS fa il boot.
- media/ - scripts e immagini che sono specifiche di una variante Live.
- optional/ - moduli squashfs messi in questa directory non saranno caricati automaticamente nel filesystem Live quando l' OS fa il boot. Dovrete passare il parametro di boot “load=[mod]” per caricare uno di questi.
- patches/ - patches per gli scripts di Slackware che necessitano di essere modificati per essere lanciati in un OS Live.
- pkglists/ - files di definizione dei repository di terze parti (*.conf) e la lista dei pacchetti usata da questi repository (*.lst) va messa in questa directory.
- setup2hd/ - templates degli script usati dall' installatore su disco `setup2hd`.
- skel/ - contiene i tarball compressi (i cui nomi dei files devono matchare la wildcard “skel*.txz”). Questi files saranno estratti nella directory “/etc/skel” nel filesystem Live.
- syslinux/ - contiene la struttura per il supporto al boot sui conputer con BIOS. Alcuni dei suoi files sono generati dinamicamente dallo script “make_slackware_live.sh”.
- xdm/ - tema grafico Slackware per il gestore delle sessioni grafiche XDM per quelle varianti ISO che non hanno GDM, KDM o SDDM.

La radice della directory 'liveslak' contiene i seguenti files:

- blueSW-128px.png , blueSW-64px.png - questi sono i bitmaps del logo “Blue S” di Slackware, usato per l' icona dello user “live” e nel tema di XDM.
- grub.tpl - il template file che e' usato per generare il menu grub per il boot UEFI.
- iso2usb.sh - questo script crea una versione USB bootabile con persistenza da una ISO Live Slackware
- languages - questo file contiene le configurazioni di input per il supporto linguistico. Una lingua per riga contiene i seguenti campi: “code:name:kbd:tz:locale:xkb”. Esempio: “nl:nederlands:nl:Europe/Amsterdam:nl_NL.utf8:”
    - code = 2 - lettera codice della lingua
    - name = nome descrittivo della lingua
    - kbd = nome dalla mappatura della tastiera della console per la lingua
    - tz = fuso orario per il paese nativo della lingua
    - locale = il locale usato nel paese
    - xkb = customizzazione opzionale della tastiera per X per il linguaggio

########## DA QUI ##########

The toplevel 'liveslak' directory contains the following files:

- blueSW-128px.png , blueSW-64px.png - these are bitmaps of the Slackware “Blue S” logo, used for the “live” user icon and in the XDM theme.
- grub.tpl - the template file which is used to generate the grub menu for UEFI boot.
- iso2usb.sh - this script creates a bootable USB version wih persistence from a Slackware Live ISO.
- languages - this file contains the input configuration for language support. One language per line contains the following fields: “code:name:kbd:tz:locale:xkb”. Example: “nl:nederlands:nl:Europe/Amsterdam:nl_NL.utf8:”

    - code = 2-letter language code
    - name = descriptive name of the language
    - kbd = name of the console keyboard mapping for this language
    - tz = timezone for the language's native country
    - locale = the locale used in the country
    - xkb = optional custom X keyboard variant for the language
- liveinit.tpl - this is the template for the “init” script which is copied into the initrd image for the Live OS. Together with the Slackware generic kernel, the initrd is what boots the computer. The “init” script assembles the Live filesystem from its squashfs modules.
- make_slackware_live.conf - the configuration file for the “make_slackware_live.sh” script. You can define defaults for many script parameters here so that you do not have to edit the script itself.
- make_slackware_live.sh - the script that generates the Live ISO.
- makemod - this script creates a squashfs module out of a Slackware package (or out of a directory tree).
- menu.tpl - template which is used to generate the syslinux boot menu for BIOS computers.
- pxeserver.tpl - template to generate the script that starts a PXE server allowing other computers to boot Slackware Live over the network.
- setup2hd.tpl - template to generate the script you use to install your Slackware Live to a harddisk.
- setup2hd.local.tpl - here a developer of a custom Live OS can override the default post-installation routine by (re-)defining the function “live_post_install()” in the `setup2hd` script.
- upslak.sh - a script which allows you to tweak the content of a USB Live stick.

Generate the ISO

The liveslak's “make_slackware_live.sh” script accepts optional parameters to tweak the process of Live OS generation:

```
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.
 -c comp             Squashfs compression (default: xz).
                     Can be any of 'gzip lzma lzo xz zstd'.
 -d desktoptype      SLACKWARE (full Slack), LEAN (basic Plasma5/XFCE),
                     DAW (Digital Audio Workstation), XFCE (basic XFCE,
                     stripped), KTOWN (ktown Plasma5 replacement), MATE
                     (Gnome2 fork replaces KDE), CINNAMON (fork of Gnome3
Shell
                     replaces KDE), DLACK (Gnome3 replaces KDE).
 -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.
 -l <localization>   Enable a different default localization
                     (script-default is 'en').
 -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 <none|doc|mandoc|bloat>
                    Trim the ISO (remove man and/or doc and/or bloat).
 -v                 Show debug/error output.
 -z version         Define your Slackware version (default: current).
 -C                 Add RAM-based Console OS to boot menu.
 -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).
 -S privkey:cert    Enable SecureBoot support and sign binaries
                    using the full path to colon-separated
                    private key and certificate files.
 -X                 Use xorriso instead of mkisofs/isohybrid.
```

The script uses package repositories to create a Live ISO. The packages will be installed into a temporary directory.

In order to create a Live ISO for any of these variants, the package repositories that are required must be available as a local directory (this can be a network-mounted directory). If you have not mirrored them locally, then all packages of the Slackware repository as well as those you require from a 3rd party repository will be downloaded from a remote server as long as a rsync URL for the repository is configured in ./pkglists/*.conf.

When all pre-reqs are met, you issue a single command to generate the ISO. The following example will create a pure Slackware Live Edition:

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

Another example which creates a MATE variant, configuring runlevel '3' as default and specifying a custom path for the Slackware package repository root (note that the script will look for a subdirectory “slackware64-current” below this directory if you are generating this ISO for slackware64-current):

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

An example on how to create a DAW Live ISO which supports UEFI SecureBoot (since liveslak 1.5.0 and only for 64-bit), is compressed using 'zstd' instead of the default 'xz' and is generated using xorriso instead of mkisofs. You need to provide the full path to a SSL private key and certificate file:

```
# ./make_slackware_live.sh -d DAW -c zstd -X -S
/root/liveslak.key:/root/liveslak.pem
```

If you want to know what package sets are included in any of these Desktop Environments, run the following command:

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

for MATE, you will find:

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

```
local:slackpkg+"
```

Which means that most of the Slackware package series (excepting kde and kdei) will be installed from their tagfiles, and on top of that two package lists are installed from the pkglists/ subdirectory: slackextra and mate. Lastly, “slackpkg+” will be installed from a local directory.

Using the Customization Features of the Live OS

Master configuration file

You can create your own custom Live OS by changing its characteristics in the configuration file “`make_slackware_live.conf`”. Among the things you can change are:

- The name of the Desktop variant (the script itself knows “*SLACKWARE*”, “*KTOWN*”, “*DAW*”, “*XFCE*”, “*MATE*”, “*CINNAMON*”, “*STUDIOWARE*” and “*DLACK*”),
- The list(s) of packages used for your custom distribution,
- The full name of the user (by default that is “*Slackware Live User*”),
- The name of the useraccount (by default that is “*live*”),
- The name of the distribution (by default that is “*slackware*”),
- And finally you can define a function “`custom_config()`” where you can add all your costom post-installation steps that are not covered in the “`make_slackware_live.sh`” script itself.

This is the section in `make_slackware_live.conf` which deals with these customizations. Two variables are required if you want to create your own custom Live OS: “**`LIVEDE`**” and “**`SEQ_CUSTOM`**”, the rest is optional (but useful nevertheless):

```
# 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,noxbase,x_base,xapbase,xfcebase,cinelerra"

# OPTIONAL:
# Use something else than the name "min",
# for the package list containing the generic kernel:
#MINLIST="min"

# 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:
#}
```

Custom background images

The Plasma5 based Live variants allow customization of the background image used for the login greeter, the desktop wallpaper and the lock screen. The image you want to use for this purpose, must have a 16:9 aspect ratio and its dimensions should at least be 1920×1080 pixels. You must store the custom image inside the liveslak source tree: in the subdirectory `./media/<variant>/bg/` where "<variant>" is the lower-case name of the Live variant (variant 'KTOWN' equals directory 'ktown', 'DAW' becomes 'daw', etc).

The "make_slackware_live.sh" script will look there for a file named either "background.jpg" or "background.png". If you want, that file can be a symlink to the actual bitmap file. The image will be converted into a set of wallpaper images of different aspect ratios and sizes. The different aspect ratios like 16:9, 16:10 and 4:3 will be achieved by cropping the images if needed, to avoid distortion. The image set will be installed as a Plasma5 wallpaper called "Slackware Live", and configured to be the default Live OS background.

Internals of Slackware Live Edition

Overlayfs, squashfs

Overlayfs and squashfs are doing the real magic here. As explained earlier, the squashfs program takes a directory structure and complresses this into a single archive file. It does this in a special way, more like how mkisofs does this than how tar creates an archive. The resulting module can be loop-mounted to access the filesystem inside.

When you have several of these loop-mounted squashfs modules, each containing a fraction of the filesystem of the OS, you are going to stack these fractional filesystems on top of each other and thus

assemble the complete filesystem (much like Tintin did in The Secret of the Unicorn when he overlayed several translucent pieces of parchment and looked through them to see the complete picture). Overlayfs is the driver that performs this 'overlaying' of the fractional filesystems. Overlayfs can work with many read-only fractional filesystems and exactly one writable filesystem. The writable filesystem is what gives your Live OS the appearance that it is writing to a hard drive - the writes to the overlay filesystem are however done to RAM (in that case you have a real Live OS) or to a 'persistence' filesystem which is stored somewhere on a writable medium (a USB stick with 'persistence' is an example of this case).

The initrd and its init script

The initrd used for the Slackware Live Edition is a standard Slackware initrd created with Slackware's “mkinitrd” command, with just one modification: its “init” script. The correct name for an 'initrd' nowadays is 'initramfs' by the way, short for “initial ram filesystem” because it contains the initial file system that gets loaded into kernel memory when your computer boots. The init script of an initrd is what prepares the root filesystem even before the actual OS starts. When the OS does start, it finds a root filesystem all ready to use. An example case is of course the LUKS-encrypted root filesystem. Another is a root filesystem stored on logical volumes (LVM). Some advance work is required to make the root filesystem accessible and start the real “init” program of the OS (PID 1). Just like the previous examples, you need a script in an initrd to assemble the root filesystem of a Live OS. Slackware's initrd does not support a Live environment, so the stock init script was expanded for use in liveslak.

What does the 'liveslak' init script do?

- It parses any boot parameters you entered (or were passed by syslinux/grub) and tries to make sense of them.
- It does some initialization just like the Slackware init (start udev, wait a bit for USB media to settle, load kernel modules, load custom keyboard mapping, initialize RAID etc) before it gets to the Slackware Live section.
- A RAM based filesystem is created which forms the base of all the rest.
- Using tools like 'blkid' and 'mount', the init script tries to locate the Live media. It uses blkid to search for the Live media's default filesystem label. If that fails (because you changed the label) it will use blkid to find all filesystem partitions available and mount them mount these one by one until the Live partition is found.
- With the Live media located, the next step is to loop-mount the squashfs modules one by one and add them to the overlay filesystem in the correct order. If you specified the 'toram' boot parameter, then a module will first be copied into RAM before loop-mounting it. This allows you to remove the boot media after booting since the complete OS will run from RAM.
- Modules will be loaded in order:
    - first the system modules (core modules in the system/ subdirectory)
    - then the addon modules (in the addon/ directory). If you do not want an addon to be loaded, you can specify “noload=modulename” on the syslinux/grub boot commandline
    - last, the optional modules (in the optional/ subdirectory). By default an optional module is not loaded unless you force it by adding “load=modulename” to the boot commandline.
- Next, persistence will be configured if the Live OS was booted from a writable media such as a USB stick. The init script will first look for a directory in the root of the Live partition of the USB stick called “persistence” and use that to store persistent changes to the Live filesystem. If that directory does not exist but a file “persistence.img” is found, then that file will be loop-mounted and persistent changes to the Live filesystem will be written to this container file. The “persistence.img” container can be LUKS-encrypted in which case the init script will ask you for

a passphrase to unlock it before mounting.
- The overlay filesystem is then finalized by adding the writable toplevel directory structure (either persistent or volatile).
- The complete RAM filesystem which underpins the overlay is made available to the user of the Live OS as “/mnt/live”
- The filesystem of the Live media is made available to the user of the Live OS as “/mnt/livemedia”. If the media is a USB stick then you will have write access to “/mnt/livemedia”.
- With the root filesystem assembled, the Live OS is configured before it actually boots:
    - if a OS-specific configuration file (by default `/liveslak/slackware_os.cfg`) exists, its contents will be parsed. Values of the variables defined in this file overrule any default *liveslak* or boot command-line values.
    - if you specified “swap” on the boot commandline, any available swap partition will be added to “/etc/fstab” in the Live OS.
    - if you specified a custom keyboard layout for the console (and optionally another for X) by using the “kbd” and “xkb” boot parameters then these will be confifured in “/etc/rc.d/rc.keymap” and “/etc/X11/xorg.conf.d/30-keyboard.conf” in the Live OS.
    - Same for any custom locale which was specified with the “locale” parameter, this will get added to “/etc/profile.d/lang.sh”.
    - If timezone and hardware clock were specified in the “tz” parameter, these will be configured in “/etc/localtime” and “/etc/hardwareclock”.
    - The boot parameters “livepw” and “rootpw” allow you to specify custom passwords for the 'live' and 'root' users; the defaults for these two are simply 'live' and 'root'. This is achieved by running the “chpasswd” command in the chrooted overlay so that a plain text password can be given as input.
    - The “hostname” boot parameter can be used to change the Live OS' hostname from its default “darkstar”. Configuration is written to “/etc/HOSTNAME” and “/etc/NetworkManager/NetworkManager.conf”.
    - If the “blacklist” boot parameter was specified, then the kernel modules mentioned as argument(s) will be added to a modprobe blacklist file “/etc/modprobe.d/BLACKLIST-live.conf”.
    - The “/var/lib/alsa/asound.state” file in the Live OS is removed to allow correct sound configuration on any computer where the Live media is booted.
    - The complete content of the /liveslak/rootcopy directory on the Live partition (may be empty) is copied to the filesystem root of the Live OS, potentially 'overwriting' files in the Live OS. Use the /liveslak/rootcopy to add customization to your Live OS when you run it off a USB stick.
    - And lastly but very importantly, any LUKS-encrypted container files are unlocked (init will ask you for the passphrase) and the filesystem(s) contained therein will be mounted in the Live OS. Currently, a LUKS-encrypted /home is supported. The files “/etc/fstab” and “/etc/crypttab” will be updated so that the Live OS will do the mounting and unmounting.
    - The init script will end by telling the kernel to swith to our new root filesystem (the overlay) and start the Slackware init program (PID 1, /sbin/init).
- From this moment onward, you are booting a 'normal' Slackware system and the fact that this is actually running in RAM and not from your local harddisk is not noticeable.

OS configuration file for persistent media

If present, the liveslak init will load a OS config file from a persistent Live medium such as a USB stick. In the case of *Slackware Live Edition* this file is called `/liveslak/slackware_os.cfg` - i.e. is

placed in the "liveslak" directory of your USB drive. For custom non-Slackware Live OS-es based on liveslak, the filename may be different.
This file contains one or more "VARIABLE=value" lines, where VARIABLE is one of the following variables that are used in the live init script:

- BLACKLIST, KEYMAP, LIVE_HOSTNAME, LOAD, LOCALE, LUKSVOL, NOLOAD, RUNLEVEL, TWEAKS, TZ, XKB.

Values for the variables defined in this configuration file override the values already set via liveslak's own defaults or via boot-up command-line parameters.

When booting your persistent *Slackware Live Edition*, the optional boot-time parameter "cfg" deals with this OS configuration file. The "cfg" parameter understands two possible argument values:

- "cfg=write" will (over)write the OS configuration file to your USB drive, using the values for all of the above variables that are valid for that particular boot. So if your timezone is "PST" then one of the lines in that file will read "TZ=PST".
- "cfg=skip" will skip processing of an existing "/liveslak/slackware_os.cfg" file.

The OS configuration file is not present by default. You either create it at boot-time using "cfg=write" (which is a persistent change) or you create it manually using an ASCII text editor, after mounting the USB partition on a computer. As an example, here is the content of "/liveslak/slackware_os.cfg" on my own USB stick:

```
KEYMAP=nl
LIVE_HOSTNAME=zelazny
LOCALE=nl_NL.utf8
TWEAKS=tpb,syn
TZ=Europe/Amsterdam
```

Slackware Live module format

A Slackware Live module contains a directory tree, which has been 'squashed' into a compressed archive file by the program "squashfs". The compression algorithm used is "xz" which explains the choice of the module's file extension ".sxz" meaning "squashed with xz".

Slackware Live Edition expects its modules to adhere to a particularly loose filename convention:

- The filename format is "NNNN-modname-*.sxz", where 'N' is a digit and 'modname' must not contain a dash '-'.
- The "modname" part is what you must specify in the boot parameters "load" and "noload".
- Anything may be part of the '*' but most commonly used is "${VERSION}-${ARCH}". The core modules in Slackware Live use the Slackware release as ${VERSION} and the Slackware architecture as ${ARCH}. For the modules in addons/ and optional/ subdirectories, ${VERSION} would commonly be the version of the program that is being made available in the module.
- The four digits of a modulename have a meaning. Some ranges are claimed by the core OS, so please do not use them. Their prefixes are based on the package source:

```
    0000 = contains the Slackware /boot directory
    0010-0019 = packages installed from a Slackware tagfile (a,ap,d,
... , y series)
```

```
    0020-0029 = packages installed from a package list as found in the
./pkglists subdirectory of the liveslak sources (min, noxbase, x_base,
xapbase, xfcebase etc)
    0030-0039 = a 'local' package, i.e. a package found in subdirectory
./local or ./local64 (depending on architecture)
    0099 = liveslak configuration module (contaning all the
customizations that change the installed packages into a usable Live
OS)
```

- Other ranges are free to be used. Note that order in which the filesystem of the Live OS is assembled by overlaying the squashed directory trees depends on the module numbering. So if you have modules that you want to have loaded in specific order, just ensure that their filenames have ascending numbers.

1. Example of a core module: 0099-slackware_zzzconf-14.2-x86_64.sxz
2. Example of an optional module: 0060-nvidia-352.79_4.4.1-current-x86_64.sxz

Other Slackware based Live distros

Naturally, there have been many who went before me, and since I started as a n00b in Linux Live land, I have learnt a lot about how a Live distro works from playing with these other Slackware-based Live distros. Allow me to name them, and show respect:

SLAX

Website: https://www.slax.org/

SLAX was the original Live variant of Slackware. The linux-live scripts which are used to create a SLAX ISO were generalized so that they can create a Live version of any OS that is already installed to a harddrive. SLAX development stalled a couple of years ago.
In 2017 a new release of SLAX became available, however Slackware is no longer its parent OS. New SLAX releases are based on Debian.

The Live functionality of SLAX is based on aufs and unionfs which requires a custom-built kernel with aufs support compiled-in. It is small and has its boot scripts tweaked for startup speed.

Porteus

Website: http://www.porteus.org/

Porteus was created as a fork of SLAX by the SLAX community when the development of SLAX seemed to have ended. Porteus has an active user community where it's “all about the modules”. The use of aufs instead of overlayfs allows Porteus (like SLAX) to add and remove squashfs modules in the running Live system on the fly, which sparked the development of a lot of community modules. It looks like the next generation of Porteus will be based on Arch Linux instead of Slackware: this has to do with the original Porteus developer leaving the team.

Salix Live

Website: http://www.salixos.org/download.html

Salix is a distribution based on Slackware with its own philosophy of “one tool per task” reducing the number of packages a lot, compared to its parent Slackware distro. Salix has implemented dependency checking in its package management tool. Live editions of Salix are available in several editions, each built around and focused on a different Desktop Environment or Window Manager. Live editions are available for XFCE and MATE.

Slackel

Website: http://www.slackel.gr/

Slackel is a Greek distro based on both Slackware and Salix. It comes in three flavors, each of which has a Live variant: KDE4, Openbox and Fluxbox. The Live scripts are a modification of Salix-Live.

SlackEX

Website: http://slackex.exton.net/

A website offering Live versions based on many regular Linux distributions. The SlackEX version is loosely based on Slackware with a custom kernel and some tools that are not part of Slackware itself. I was unable to find the sources for this live distro. Its creator stopped SlackEX development in December 2017.

Liveslak Sources

Slackware Live Edition is created by the 'liveslak' scripts developed and maintained by Eric Hameleers aka Alien BOB alien@slackware.com.

- Git repository: git://git.liveslak.org/liveslak.git
- Git repository (browsable): http://git.liveslak.org/liveslak/
- Download mirror: http://www.slackware.com/~alien/liveslak/

Sources

- Original source: https://git.slackware.nl/liveslak/tree/README.txt
- Project landing page: https://liveslak.org/
- ISO downloads: https://download.liveslak.org/

- Originally written by Eric Hameleers

slackware, live, overlayfs, squashfs, author alienbob