

Operating principle

To create a live Slackware system, two Slackware systems are needed:

- the first one, with **"GRUB"**, **"Slackware-Live"**, **"Unionfs-FUSE"** (if You don't plan to use AUFS) and **"SquashFS-Tools"** will be used to build the live system;
- the second one is the system to make live; this system must have a **kernel** (Slackware stock huge SMP kernel is recommended), **modules** and the **"MkinitRD"** utility; **"GRUB"**, **"Slackware-Live"** and **"SLI (Slackware-Live Installer)"** (optional GUI) are needed to install or copy the live system.

Once the system has been setup, the building of the live system needs three commands:

- **"build-slackware-live.sh --init ..."** to create the initrd and copy the kernel;
- **"build-slackware-live.sh --module ..."** to create a SquashFS module of the system;
- **"build-slackware-live.sh --iso ..."** to create an ISO image of the live-system or **"build-slackware-live.sh --usb ..."** to copy it on USB key.

Example

- Setup your system:
 - From a Slackware Linux system — the build system —, install wanted packages into the **"/mnt/system"** — the live system root — directory:

```
installpkg -root /mnt/system /mnt/cdrom/slackware/a/*.t?z
```

- Run setup (recommended) and add a user (if needed):

```
chroot /mnt/system #live system
pkgtool #setup
useradd -m -g users -s /bin/bash liveuser
usermod -G floppy,cdrom,netdev,plugdev,scanner,lp,audio,video,power liveuser
passwd liveuser
exit #chroot
```

- Build the live system:

```
build-slackware-live.sh --init /mnt/system /tmp/live
build-slackware-live.sh --module /mnt/system /tmp/live 0-slackware-live -xz
build-slackware-live.sh --iso /tmp/live /tmp/slackware-live.iso
```

*It is possible to have a single system that transforms itself into a live system: just use **"/"** for the system root directory parameter (instead of **"/mnt/system"** in the example above); be careful: all directories but **"/sys"**, **"/proc"**, **"/dev"** and **"/tmp"** are included in the live system; think to umount removable devices for example.*

System setup

Live system is just an ordinary Slackware

Install the live system like any other Slackware Linux (or derived distribution):

- make a regular install from the distribution install media, then, access this system by mounting its partition from the build system:

```
mkdir /mnt/system
mount /dev/installation_device /mnt/system
```

- or install packages into a subdirectory of the build system (the live system root directory) — with **"installpkg -root /mnt/system ..."** for example —.

Installing from a list of selected packages

The build system includes an option to install a list of selected packages into a subdirectory:

- usage: **"build-slackware-live.sh --add packages_dir root_dir pkg_list_file"**
- example: **"build-slackware-live.sh --add /mnt/cdrom /mnt/system packages-list.txt"**

packages-list.txt example

```
slackware/a/*
slackware/n/dhccpd
slackware/n/iputils
slackware/n/net-tools
slackware/n/network-scripts
postinstall=ln -sf ifconfig usr/bin/ifcfg
postinstall=echo "live.slackware.org" > etc/HOSTNAME
```

Sysprep

Running Sysprep is recommended before building the live system and is required if you want to speed up live system boot (see **"fastboot"** boot parameter); it consists on the following actions:

- run commands like **"mkfontdir"** (... — ordinary run by **"pkgtool"** during setup);
- run commands like **"fc-cache"**, **"update-pango-querymodules"** (... — run by stock Slackware **"rc.M"** startup script);
- run **"ldconfig"** and **"depmod"**;
- merge **"/etc/passwd"** and **"/etc/group"** system files (if the system is divided into multiple directories);
- set up root and users profile (from **"/etc/skel"**).
- usage: **"build-slackware-live.sh --sysprep dir_1(rw) dir_2(ro) dir_3(ro)..."**
only the first directory parameter is mandatory; if the system is divided into multiple directories (one per module), list all of them to recompose the whole system; setup changes will be written into the first one
- example: **"build-slackware-live.sh --sysprep /mnt/system-gui /mnt/system-core"**

Live system build commands

Setup kernel and initrd

- usage: **"build-slackware-live.sh --init root_dir live_dir [modules_list]"**
the **"modules_list"** defaults to **"squashfs:fuse:loop:isofs:nls_utf8:xhci-hcd:ehci-hcd:uhci-hcd:usb-storage"** (needed for Slackware stock huge kernel)
- example: **"build-slackware-live.sh --init /mnt/system /tmp/live"**

Create a SquashFS module for the system

- usage: **"build-slackware-live.sh --module root_dir live_dir module_file [-xz-gzip] [-optional]"**
with the **"-xz"** or **"-gzip"** option, it is possible to choose SquashFS compression method: the live system will be about 20 % smaller with xz, but about 20% faster with gzip — the provided SquashFS package uses gzip by default
with the **"-optional"** option, the module is stored in directory **"live_system_dir/boot/optional"** instead of **"live_system_dir/boot/modules"**
- example - for live system build (typically):
"build-slackware-live.sh --module /mnt/system /tmp/live 0-slackware-live"
- example - to store changes made on live USB while running:
"build-slackware-live.sh --module /live/changes /live/media 1-changes"

Using multiple modules

The system can be divided into multiple modules (example: "core", "gui", "tools",...) that are loaded in alphabetical order; if a file is present into several modules, the one taken from the last loaded module is used; naming example:

- 1-an_application
- 1-an_other_application
- ...
- 2-gui
- 3-core
- 4-2012-01-01-updates
- 4-2012-02-01-updates
- ...

System on USB device: manual partitionning (optional / discouraged)

Doing a manual partitionning is tricky, and is discouraged unless a Windows readable partition is needed.

For USB booting (live or installed system), the required partition scheme is an hybrid GPT (GUID Partition Table) / MBR (Master Boot Record) with the following partitions (use **"r"** then **"h"** gdisk commands to add partitions 1 to 3 to MBR):

- a GPT / EFI partition is needed to enable UEFI (Unified Extensible Firmware Interface) booting.
- a BIOS (Basic Input Output System) boot partition is required by GRUB on GPT disk / key for legacy BIOS (aka CSM (Compatibility Support Module)) booting.
- an install partition (for the live system) is also required ;-)

Example:

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	67583	32.0 MiB	EF00	EFI System
2				8300	Linux filesystem (installation partition)
3				0700	Microsoft basic data(optional: for Windows compatibility)
any	34	2047	1007.0 KiB	EF02	BIOS boot partition

Copy live system on USB device

Warning: if the destination is a whole disk or key (**"/dev/sdb"** for example), it is **automatically repartitionned** (see above) (and **all data on it are wiped**).

- usage: **"build-slackware-live.sh --usb live_dir device"**
- example - after initialization and module creation:
"build-slackware-live.sh --usb /tmp/live /dev/sdx2"
- example - from a running live system:
"build-slackware-live.sh --usb /live/media /dev/sdx"

Create a live CD/DVD ISO from live system

- usage: **"build-slackware-live.sh --iso live_dir iso_file_name"**
- example - after initialization and module creation:
"build-slackware-live.sh --iso /tmp/live /tmp/slackware-live.iso"
- example - from a running live system:
"build-slackware-live.sh --iso /live/media /mnt/sdy1/slackware-live.iso"

Hybrid ISO / USB

- Convert ISO: **"isohybrid /path/to/iso"**
- Copy ISO on key (warning, **wipes everything** on key): **"dd if=/path/to/iso of=/dev/sdx"**

Boot parameters

System language and keymap layout

- **"locale"**: system language; example: **"locale=fr_FR.UTF-8"**
- **"keymap"**: keymap layout; example: **"keymap=fr"**; the first two characters are used for Xorg keymap layout
- **"tz"**: timezone; example: **"tz=Europe/Paris"** (value must be a valid path from **"/usr/share/zoneinfo"**)
- **"hwc"**: hardware clock: **"UTC"** or **"localtime"**

Modules loading

- **"include=module1:module2:..."**: to load selected modules from **"/boot/optional"** directory (module names are the SquashFS file names)
- **"exclude=module1:module2:..."**: to specify the main modules (from **"/boot/modules"** directory) not to load; example: **"exclude=1-gui"**

Persistent home directory storage or system changes

- **"home=/path/to/directory|NFS_resource|UUID (Universal Unique Identifier)|label"**: activates persistent home directory;
- **"changes=/path/to/directory|NFS_resource|UUID (Universal Unique Identifier)|label"**: activates persistent system changes .

Misc

- **"runlevel=[1-5]"**: overrides default startup runlevel (cf **"/etc/inittab"**)
on runlevel 5, the user with uid 1000 will be automatically logged in and X session started (if installed); if this user doesn't exists, root user is used instead
- **"copy2ram=yes"**: enables running the live system from memory
- **"useswap=yes"**: enables swap on detected swap partitions
- **"rootpw=password"**: defines a root password
- **"fastboot=yes"**: skip **"ldconfig"**, **"depmod"**, **"fc-cache"**, **"update-mime-database"**,
"gtk-update-icon-cache", **"update-gtk-immodules"**, **"update-gtk-immodules"** and
"update-pango-querymodules"

Live system installation

The live system can be installed into a hard disk partition; the result is the same as a clean installation.

Install live system

- usage: **"build-slackware-live.sh --install root_dir device [-auto]"**
*the **"-auto"** option enables GRUB installation*
- example - from a running live system (typically):
"build-slackware-live.sh --install /live/system /dev/sdx2 -auto"
- example - system cloning:
"build-slackware-live.sh --install /mnt/system /dev/sdx2"

Live system usage

Firefox cache and persistent home directory problem

When using Firefox with persistent home directory feature enabled on an USB flash disk, disk cache should be disabled to avoid continuous writings. This can be done by pointing your Web browser to **"about:config"** (or by editing **".mozilla/firefox/PROFILE_NAME/prefs.js"**) and set **"browser.cache.disk.enable"** property to **"false"**.

Live Root Over NFS setup

The live system can be booted over the network. Both SysLinux and GRUB2 are supported for PXE booting.

Prerequisites

- DHCP server ("**n/dhcp**")
- TFTP server ("**n/inetd**", "**n/tftp-hpa**")
- NFS server ("**n/portmap**" + "**n/nfs-utils**")
- Patched GRUB2 (see attached patch and grub package)

DHCP setup

- Create a "**/etc/dhcpd.conf**" file with a content like:

```
ddns-update-style none;
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.1 192.168.1.100;
    option routers 192.168.1.254;
    option domain-name-servers 192.168.1.254;
    filename "/boot/grub/i386-pc/core.0";
    next-server 192.168.1.253;
}
```

- Start the server ("**dhcpd**")

TFTP setup

- Populate the "**/tftpboot**" directory with the following commands:

```
grub-mknetdir --net-directory=/tftpboot
cp /path/to/generic-kernel /tftpboot/boot/vmlinuz
cp /path/to/slackware-live/initrd.gz /tftpboot/boot/
```

- Add a "**/tftpboot/boot/grub/grub.cfg**" file (ip string has the following format: "**ip:pxe_server:gateway:netmask**"):

```
menuentry "Slackware-Live" {
    linux /boot/vmlinuz nfsroot=192.168.1.253:/srv/linomad max_loop=255
    locale=fr_FR.UTF-8 keymap=fr tz=Europe/Paris hwc=localtime
    ip=$net_pxe_ip:192.168.1.253:192.168.1.254:255.255.255.0
    initrd /boot/initrd.gz
}
```

- Enable tftp (edit "**/etc/inetd.conf**") and start inetd service

NFS setup

- In the "**/etc/exports**" file, add a line like:

```
/live/media 192.168.1.0/255.255.255.0(ro,no root squash,async,no subtree check)
```

- Start the portmap and nfs service (or reload configuration with "**exportfs -r**")

Grub patch and package

- grub-2.00-i486-2.txz.zip (IMG/zip/grub-2.00-i486-2.txz.zip) (2014-09-18)
- grub-tftp.patch.zip (IMG/zip/grub-tftp.patch.zip) (2014-09-18)