NAME

live−build − the Debian Live tool suite

SYNOPSIS

lb {**−h**|**−−help**|**−u**|**−−usage**|**−v**|**−−version**}

lb *COMMAND* [*OPTIONS*]

DESCRIPTION

live−build is a set of scripts to build live system images. The idea behind live−build is a tool suite that uses a configuration directory to completely automate and customize all aspects of building a Live image.

The *COMMAND* is a name of a live−build command (see below).

More documentation about how to use live−build is available in the individual manpages for each helper and in the manual at <*https://live-team.pages.debian.net/live-manual/*>.

OPTIONS

Shared live−build options

The following command line options are supported by all live−build programs.

-h, −−help
display help and exit.

-u, −−usage
show usage and exit.

-v, −−version
output version information and exit.

Common live−build options

The following command line options are supported by most live−build programs. See the man page of each program for a complete explanation of what each option does.

−−breakpoints
run with breakpoints.

−−color
enable color use in messages.

−−debug
show debug information.

−−force
force helper execution, even if stage file exists.

−−no-color
disable color use in messages.

−−quiet
be quiet.

−−verbose
be verbose.

LIVE−BUILD COMMANDS

We divide live−build into high level ("porcelain") commands, secondary major build stage ("porcelain") commands, and low level ("plumbing") commands.

Here is the complete list of all available live−build commands. See their man pages for additional documentation.

HIGH−LEVEL COMMANDS (PORCELAIN)

We separate the porcelain commands into the main commands and some ancillary user utilities.

Main porcelain commands

lb config(1)
: creates configuration for live−build

lb build(1)
: executes the build process (by executing all of the secondary level major build stages in sequence)

lb clean(1)
: cleans up system build directories

Ancillary Commands

lb(1)
: generic live−build script execution wrapper

SECONDARY−LEVEL BUILD COMMANDS (PORCELAIN)

The following are the commands that execute each major stage of the build process, in their necessary order of execution. Normally a user might just execute the higher level **lb build**(1) command rather than use these individually.

lb bootstrap(1)
: executes the first build stage, creating (bootstraping) a basic Debian root filesystem

lb chroot(1)
: executes the second build stage, building the live OS filesystem

lb installer(1)
: executes the third build stage, obtaining installer components (optional)

lb binary(1)
: executes the fourth build stage, generating the binary (live) image

lb source(1)
: executes the fifth build stage, generating a corresponding source image (optional)

LOW−LEVEL COMMANDS (PLUMBING) - BUILD STAGE COMPONENTS

The actual work of live−build is implemented in the low-level commands, called plumbing. They are not supposed to be used by end users, who should stick with porcelains as they ensure that all the different plumbing commands are executed in the right order. However, if you intend to reuse live−build commands in your own scripts, then the plumbings might be of interest for you.

Note that the interface (set of options and the semantics) to these low−level commands are meant to be a lot more stable than Porcelain level commands. The interface to Porcelain commands on the other hand are subject to change in order to improve the end user experience.

Bootstrap stage specific commands

lb bootstrap_archives(1)
: applies apt archive configuration

lb bootstrap_cache(1)
: in save mode, saves to cache a copy of the generated bootstrap directory, and in restore mode, restores from cache a previously generated copy

lb bootstrap_debootstrap(1)
: creates (bootstraps) a basic Debian root filesystem using debootstrap(8)

Chroot stage specific commands

Note: The following chroot_ prefixed commands are used in building the live OS filesystem. Another set of similarly prefixed files are listed separately (see further down).

lb chroot_cache(1)
in save mode, saves to cache a copy of the chroot directory, and in restore mode, restores from cache a previously generated copy

lb chroot_firmware(1)
compiles a list of firmware packages to be installed in the live OS root filesystem

lb chroot_hacks(1)
executes local hacks against the live OS root filesystem, if any are provided

lb chroot_hooks(1)
executes local hooks against the live OS root filesystem, if any are provided

lb chroot_includes(1)
copies a set of local files from the config directory into the live OS root filesystem, if any are provided

lb chroot_install−packages(1)
installs into the live OS root filesystem any packages listed in local package lists

lb chroot_interactive(1)
pauses the build process and starts an interactive shell from the live OS root filesystem, providing an oportunity for manual modifications or testing; note that this is (currently) usually executed with several chroot prep modifications applied (see description of these further down)

lb chroot_linux−image(1)
compiles a list of kernel images to be installed in the live OS root filesystem

lb chroot_package−lists(1)
compiles a list of packages provided in the user´ local config to be installed in the live OS root filesystem

lb chroot_preseed(1)
installs pre-configured answers to certain install prompts into the live OS root filesystem

Installer stage specific commands

lb installer_debian−installer(1)
obtains and sets up Debian installer (d-i) components

lb installer_preseed(1)
installs pre-configured answers to certain install prompts

Binary stage specific commands

lb binary_checksums(1)
creates checksums (md5, sha1, and/or sha256) for live image content

lb binary_chroot(1)
duplicates the chroot directory, to place a copy of what would be the completed live OS root filesystem to one side, allowing the original to continue to be used in executing certain parts of the remainder of the build process

lb binary_disk(1)
creates disk information files to be added to live image

lb binary_grub_cfg(1)
creates the config for grub-pc and grub-efi, and also enables loopback support (which depends upon it) in the live image

lb binary_grub-efi(1)
installs grub-efi (grub2 for EFI) into live image to provide image boot capability. It relies upon **lb binary_grub_cfg** to create the config.

lb binary_grub-legacy(1)
installs grub into live image to provide image boot capability

lb binary_grub-pc(1)
installs grub-pc (grub2 for BIOS) into live image to provide image boot capability. It relies upon **lb binary_grub_cfg** to create the config.

lb binary_hdd(1)
compiles the final live image into an HDD image file

lb binary_hooks(1)
executes local hooks against the live image, if any are provided

lb binary_includes(1)
copies a set of local files from the config directory into the live image, if any are provided

lb binary_iso(1)
compiles the final live image into an ISO file

lb binary_linux−image(1)
copies the linux−image into the live image

lb binary_loadlin(1)
bundles a copy of loadlin into the live image

lb binary_manifest(1)
creates manifest of packages installed into live OS filesystem, and list of packages to be excluded by a persistence mechanism installing the live OS to disk

lb binary_memtest(1)
bundles a copy of memtest into the live image

lb binary_netboot(1)
compiles the final live image into a netboot tar archive

lb binary_onie(1)
installs onie into the live image

lb binary_package−lists(1)
processes local lists of packages to obtain and bundle into image (from which they could later be installed if not already)

lb binary_rootfs(1)
wraps up the completed live OS root filesystem into a virtual file system image

lb binary_syslinux(1)
installs syslinux into live image to provide image boot capability

lb binary_tar(1)
compiles the final live image into a tar archive

lb binary_win32−loader(1)
bundles a copy of win32−loader into the live image and creates an autorun.inf file

lb binary_zsync(1)
builds zsync control files

Source stage specific commands

lb source_checksums(1)
creates checksums (md5, sha1, and/or sha256) for source image content

lb source_debian(1)
downloads source packages for bundling into source image

lb source_disk(1)
creates disk information files to be added to source image

lb source_hdd(1)
compiles the final source image into an HDD image file

lb source_hooks(1)
executes local hooks against the source image, if any are provided

lb source_iso(1)
compiles the final source image into an ISO file

lb source_live(1)
copies live-build config into source

lb source_tar(1)
compiles the final source image into a tar archive

LOW−LEVEL COMMANDS (PLUMBING) - CHROOT PREP COMPONENTS

The notes above under the section regarding build-stage specific low-level plumbing commands also apply here.

The following chroot_ prefixed commands are used throughout the various primary stages of the build process to apply and remove modifications to a chroot root filesystem. Generally these are used to apply modification that setup the chroot for use (execution of programs within it) during the build process, and later to remove those modification, unmounting things that were mounted, and making the chroot suitable for use as the root filesystem of the live OS to be bundled into the live image.

Note that the **lb chroot_prep**(1) command can be used to run these components in bulk.

lb chroot_prep(1)
a helper to run the below components in bulk. The first parameter it takes is the execution mode - install or remove - to pass along. The second parameter is the set of helpers to run (they can be space or comma separated; remember to quote if space separated). Following this one or more of the special parameters 'mode-archives-chroot', 'mode-archives-binary', 'mode-archives-source' and 'mode-apt-install-binary' can optionally be used, to select the 'pass' parameter for **lb chroot_archives**(1) in the case of the first three (required if 'archives' is one of the helpers to be run), and to run **lb chroot_apt**(1) in 'install-binary' mode in the last case. Any remaining parameters (i.e. options like --force) are passed along to all scripts run. The second parameter can be simply 'all' in which case a default set of all components are used, or 'all-except-archives' which differs in skipping **lb chroot_archives**(1). Components can be specified without their filename 'chroot_' prefix for brevity. In remove mode the list of components are run in reverse order, so no need to provide them in reverse order yourself.

lb chroot_apt(1)
manages apt configuration; in apply mode it applies configuration for use during build process, and in remove mode removes that configuration

lb chroot_archives(1)
manages apt archive source lists; in apply mode it applies source list configurations suitable for use of the chroot in the build process, and in remove mode replaces that with a configuration suitable for the final live OS

lb chroot_debianchroot(1)
manages a /etc/debian_chroot file

lb chroot_devpts(1)
manages mounting of /dev/pts

lb chroot_dpkg(1)
manages dpkg; in apply mode disabling things like the start-stop-daemon, and in remove mode enabling them again

lb chroot_hostname(1)
manages the hostname configuration

lb chroot_hosts(1)
manages the /etc/hosts file

lb chroot_proc(1)
manages mounting of /proc

lb chroot_resolv(1)
manages configuration of the /etc/resolv.conf file

lb chroot_selinuxfs(1)
manages mounting of /sys/fs/selinux

lb chroot_sysfs(1)
manages mounting of /sys

lb chroot_sysv−rc(1)
manages the /usr/sbin/policy−rc.d file

lb chroot_tmpfs(1)
manages configuration of dpkg to use a tmpfs filesystem

CONFIG FILES

Many live−build commands make use of files in the *config/* directory to control what they do. Besides the common *config/common*, which is used by all live−build commands, some additional files can be used to configure the behavior of specific live−build commands. These files are typically named config/stage (where "stage" of course, is replaced with the name of the stage that they belong to).

Note that live−build will respect environment variables which are present in the context of the shell it is running. If variables can be read from config files, then they override environment variables, and if command line options are used, they override values from config files. If no value for a given variable can be found and thus is unset, live−build will automatically set it to the default value.

In some rare cases, you may want to have different versions of these files for different architectures or distributions. If files named config/stage.arch and config/stage.dist exist, where "arch" is the same as the output of "dpkg −−print−architecture" and "dist" is the same as the codename of the target distribution, then they will be used in preference to other, more general files.

All config files are shell scripts which are sourced by a live−build program. That means they have to follow the normal shell syntax. You can also put comments in these files; lines beginning with "#" are ignored.

FILES

/etc/live/build.conf

/etc/live/build/*

SEE ALSO

live−boot(7)

live−config(7)

This program is a part of live−build.

HOMEPAGE

More information about live−build and the Debian Live project can be found on the homepage at <*https://wiki.debian.org/DebianLive*>.

BUGS

Bugs can be reported by submitting a bug report for the live−build package in the Bug Tracking System at <*http://bugs.debian.org/*> or by writing a mail to the Debian Live mailing list at <*debian-live@lists.debian.org*>.

AUTHOR

live−build was originally written by Daniel Baumann <*mail@daniel-baumann.ch*>. Since 2016 development has been continued by the Debian Live team.