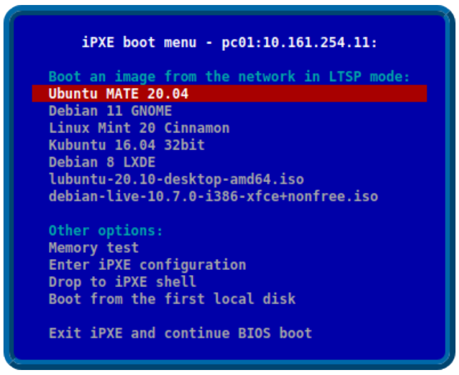
LTSP

Linux Terminal Server Project

Table of contents

1. About

Linux Terminal Server Project helps in netbooting LAN clients from a single template installation that resides in a virtual machine image or a chroot on the LTSP server, or the server root (/, chrootless). This way maintaining tens or hundreds of diskless clients is as easy as maintaining a single PC.



```
        iPXE boot menu - pc01:10.161.254.11:

  Boot an image from the network in LTSP mode:
  Ubuntu MATE 20.04
  Debian 11 GNOME
  Linux Mint 20 Cinnamon
  Kubuntu 16.04 32bit
  Debian 8 LXDE
  lubuntu-20.10-desktop-amd64.iso
  debian-live-10.7.0-i386-xfce+nonfree.iso

  Other options:
  Memory test
  Enter iPXE configuration
  Drop to iPXE shell
  Boot from the first local disk

  Exit iPXE and continue BIOS boot
```

LTSP has been redesigned and rewritten from scratch in 2019 by alkisg in order to support new technologies like systemd, updated desktop environments, Wayland, UEFI etc. Only the new version is actively developed, while the old one is now called LTSP5 and is in minimal maintenance mode.

LTSP automatically configures and uses the following tools:

- iPXE: network boot loader that shows the initial client boot menu and loads the kernel/initrd.
- dnsmasq or isc-dhcp-server: DHCP server that assigns IPs to the clients, or ProxyDHCP server when another DHCP server is used, e.g. a router.
- dnsmasq or tftpd-hpa: TFTP server that sends ipxe/kernel/initrd to the clients.
- dnsmasq: optional DNS server for DNS caching or blacklisting.
- mksquashfs: creates a compressed copy of the template image/chroot/VM to be used as the client root /.
- NFS or NBD: serve the squashfs image to the network.
- tmpfs and overlayfs: make the squashfs image temporarily writable for each client, similiarly to live CDs.
- NFS or SSHFS: access the user's /home directory which resides on the server.
- SSH or SSHFS or LDAP: authenticate users against the server.

What are you waiting for? Continue to the installation page!

2. Documentation

2.1 Documentation

Read the following pages in order to setup LTSP:

1. Preparation
2. Installation
3. Personal package archive
4. Netboot clients

For additional security like disabling the iPXE shell, you may also read the security wiki page.

2.2 Preparation

Before installing LTSP, make sure that it fits your needs and that your hardware is appropriate.

2.2.1 When to use

The main reason to use LTSP is if you want to maintain a single installation, instead of many. It saves administration time.

The older LTSP5 targetted thin clients with reduced hardware specifications; this is no longer the case. The LTSP clients should be able to run the distribution with their own CPU/RAM; except with LTSP they'll be using a network disk instead of a local disk, so they can be diskless.

2.2.2 Network

- The server ⟺ switch connection should be gigabit; i.e. gigabit server NIC, switch port and 5e+ cabling.
- The clients ⟺ switch connection may be 100 Mbps.
- It's best if the LTSP clients are connected to the same switch as the server.

As network performance is very important for LTSP, after it's up and running do a LAN benchmark with epoptes, to make sure the server can send and receive data with at least 800 Mbps bandwidth.

2.2.3 Server

Any recent PC will do with e.g. 4 GB RAM and 3000 cpubenchmark.net score. An SSD disk for both rootfs and home will probably make client network disk access a bit faster.

2.2.4 Clients

Consult your distribution for client specifications. 1 GB RAM with 500 cpubenchmark.net score could be the minimal specs, and 2 GB RAM / 2000+ score the recommended.

2.3 Installation

2.3.1 Installation

All the terminal commands in the documentation should be run as root, which means you should initially run `sudo -i` on Ubuntu or `su -` on Debian.

Server OS installation

The LTSP server can be headless, but it's usually better to install the operating system using a "desktop" .iso and not a "server" one. All desktop environments should work fine, but MATE receives the most testing. Any .deb-based distribution that uses systemd should work; i.e. from Ubuntu Xenial and Debian Jessie and onward. If you choose Ubuntu, you may also consider removing snap to avoid some issues.

Adding the LTSP PPA

The LTSP PPA is where stable upstream LTSP releases are published. It's mandatory for distributions before 2020 that have the older LTSP5, and optional but recommended to have in newer distributions. Follow the ppa page to add it to your sources, then continue reading here.

Installing LTSP server packages

The usual way to transform a normal installation into an LTSP server is to run:

```
apt install --install-recommends ltsp ltsp-binaries dnsmasq nfs-kernel-server openssh-server squashfs-tools ethtool net-tools epoptes
gpasswd -a administrator epoptes
```

Replace *administrator* with the administrator username. If you're not using the PPA, also replace `ltsp-binaries` with `ipxe`. Description of the aforementioned packages:

- ltsp: contains the LTSP code, it's common for both LTSP servers and LTSP clients.
- ltsp-binaries: contains iPXE and memtest binaries.
- dnsmasq: provides TFTP and optionally (proxy)DHCP and DNS services. Possible alternatives are isc-dhcp-server and tftpd-hpa, but only dnsmasq can do proxyDHCP, so it's the recommended default.
- nfs-kernel-server: exports the virtual client disk image over NFS.
- openssh-server: allows clients to authenticate and access /home via SSHFS.
- ethtool, net-tools: allow disabling Ethernet flow control, to improve LAN speed when the server is gigabit and some clients are 100 Mbps.
- epoptes: optional; allows client monitoring and remote control; the gpasswd command allows the sysadmin to run epoptes.

All those packages can also be displayed with `apt show ltsp | grep ^Suggests`.

Network configuration

There are two popular methods to configure LTSP networking. One is to avoid any configuration; this usually means that you have a single NIC on the LTSP server and an external DHCP server, for example a router, pfsense, or a Windows server. In this case, run the following command:

```
ltsp dnsmasq
```

Another method is to have a dual NIC LTSP server, where one NIC is connected to the normal network where the Internet is, and the other NIC is connected to a separate switch with just the LTSP clients. For this method to work automatically, assign a static IP of 192.168.67.1 to the internal NIC using Network Manager or whatever else your distribution has, and run:

```
ltsp dnsmasq --proxy-dhcp=0
```

You can read about more `ltsp dnsmasq` options, like --dns or --dns-servers, in its man page.

Maintaining a client image

LTSP supports three methods to maintain a client image. They are documented in the ltsp image man page. You can use either one or all of them. In short, they are:

- Chrootless (previously pnp): use the server root (/) as the template for the clients. It's the easiest method if it suits your needs, as you maintain only one operating system, not two (server and image).
- Raw virtual machine image: graphically maintain e.g. a VirtualBox VM.
- Chroot: manually maintain a chroot directory using console commands.

In the virtual machine and chroot cases, you're supposed to install the ltsp package to the image, by adding the LTSP PPA and running `apt install --install-recommends ltsp epoptes-client`, without specifying any other services. In the chrootless and virtual machine cases, if you're using separate partitions for some directories like /boot or /var, see the `ltsp image` man page EXAMPLES section for how to include them. When the image is ready, to export it in *squashfs* format and make it available to the clients over NFS, run the following commands.

For chrootless:

```
ltsp image /
```

Virtual machines need to be symlinked before running `ltsp image`:

```
ln -s "/home/user/VirtualBox VMs/debian/debian-flat.vmdk" /srv/ltsp/debian.img
ltsp image debian
```

For a chroot in /srv/ltsp/x86_32:

```
ltsp image x86_32
```

You need to run these commands every time you install new software or updates to your image and want to export its updated version.

Configuring iPXE

After you create your initial image, or if you ever create additional images, run the following command to generate an iPXE menu and copy the iPXE binaries in TFTP:

```
ltsp ipxe
```

In LTSP5, syslinux was used, but iPXE replaced it as it's much more powerful. You can read more about it in the ltsp ipxe man page.

NFS server configuration

To configure the LTSP server to serve the images or chroots over NFS, run:

```
ltsp nfs
```

For finetuning options, see the ltsp nfs man page.

Generate ltsp.img

A new procedure that wasn't there in LTSP5 is provided by the following command:

```
ltsp initrd
```

This compresses /usr/share/ltsp, /etc/ltsp, /etc/{passwd,group} and the server public SSH keys into /srv/tftp/ltsp/ltsp.img, which is transferred as an "additional initrd" to the clients when they boot. You can read about its benefits in its man page, for now keep in mind that you need to run `ltsp initrd` after each LTSP package update, or when you add new users, or if you create or modify /etc/ltsp/ltsp.conf, which replaced the LTSP 5 "lts.conf".

Questions

Questions? Start a discussion or come to the online chat room.

2.3.2 Raspberry Pi OS

Basic installation instructions for netbooting Raspberry Pi clients from a Raspberry Pi OS (formely Raspbian) chroot on an LTSP server.

Prerequisites

The LTSP server should already be configured by following the installation page. If booting x86 clients is also required, do that part first as it's easier.

Client configuration

The client configuration is officially documented here. In short:

- To netboot Pi 2, format an SD card with the fat file system and put only bootcode.bin in it. This file can be found in /boot/bootcode.bin inside your Raspberry Pi OS image.
- For Pi 3B, boot from an SD card, run `echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt` and reboot.
- Pi 3B+ supports netbooting out of the box.
- For Pi 4 and Pi 400, boot from a fully updated SD card to get the latest firmware, then run `sudo raspi-config` and select `Advanced Options > Boot Order > Network Boot`.

Chroot preparation

Raspberry Pi OS is very optimized for Raspberries, so it's currently a better option than e.g. Ubuntu MATE or other distributions. The easiest way to generate a Raspberry Pi OS chroot isn't with the `debootstrap` command, but by downloading an image. You may also follow the Raspberry Pi OS installation guide or you may use `dd` to read the SD card from an existing Raspberry Pi installation; to keep the instructions shorter, we assume that in the end you have an uncompressed raspios.img on the LTSP server.

```
losetup -rP /dev/loop8 2020-12-02-raspios-buster-armhf-full.img
mount -o ro /dev/loop8p2 /mnt
time cp -a /mnt/. /srv/ltsp/raspios
umount /mnt
mount -o ro /dev/loop8p1 /mnt
cp -a /mnt/. /srv/ltsp/raspios/boot/
umount /mnt
losetup -d /dev/loop8
```

At this point, Raspberry Pi OS should be in `/srv/ltsp/raspios`. This chroot isn't ready for netbooting yet, the following commands are needed:

```
# Go to the chroot in order to use relative directories
cd /srv/ltsp/raspios
# Mask services that we don't want in netbooting
systemctl mask --root=. dhcpcd dphys-swapfile raspi-config resize2fs_once
# Remove SD card entries from fstab
echo 'proc            /proc           proc    defaults          0       0' >./etc/fstab
# Use an appropriate cmdline for NFS_RW netbooting
echo 'ip=dhcp root=/dev/nfs rw nfsroot=192.168.67.1:/srv/ltsp/raspios,vers=3,tcp,nolock console=serial0,115200 console=tty1 elevator=deadline fsck.repair=yes
rootwait quiet splash plymouth.ignore-serial-consoles modprobe.blacklist=bcm2835_v4l2' >./boot/cmdline.txt
```

Server preparation

In ltsp.conf, set RPI_IMAGE to the chroot name. This will be used by `ltsp kernel` to generate the appropriate symlinks from `/srv/tftp/*` to `/srv/ltsp/raspios/boot/*`.

```
[server]
RPI_IMAGE="raspios"
```

Then, run:

```
ltsp kernel raspios
ltsp initrd
ltsp nfs
```

NFS_RW netbooting

At this point we're ready to netboot a single client in NFS_RW mode. This means that whatever changes we do on that client, like installing new programs, are directly saved in /srv/ltsp/raspios.

First, export the chroot in NFS read-write mode:

```
echo '/srv/ltsp/raspios  *(rw,async,crossmnt,no_subtree_check,no_root_squash,insecure)' >/etc/exports.d/ltsp-raspios.exports
exportfs -ra
```

You may replace `*` with an IP to only allow access to a single client, or you may delete the /etc/exports.d/ltsp-raspios.exports file when you're done, so that there are no security issues.

Now boot a single client, add the LTSP PPA to your sources, and install the client-side packages:

```
apt install --install-recommends ltsp epoptes-client
```

LTSP mode netbooting

At this point our chroot contains the LTSP code and is ready to be netbooted. But it needs a different kernel cmdline than the NFS_RW mode, so run the following commands:

```
echo 'ip=dhcp root=/dev/nfs nfsroot=192.168.67.1:/srv/ltsp/raspios,vers=3,tcp,nolock init=/usr/share/ltsp/client/init/init ltsp.image=images/raspios.img
console=serial0,115200 console=tty1 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles modprobe.blacklist=bcm2835_v4l2'
>/srv/ltsp/raspios/boot/cmdline.txt

# Finally, create the squashfs image
ltsp image raspios --mksquashfs-params='-comp lzo'
```

That's it, now you should be able to netboot all your Raspberry Pi clients.

2.4 Personal Package Archive

Stable upstream LTSP releases are offered in .deb package format in the LTSP PPA. They should work in all .deb-based distributions that use systemd, i.e. from Debian Jessie 8 and Ubuntu Xenial 16.04 and upwards.

The LTSP PPA is mandatory for distributions before 2020 that have the older LTSP5, and optional but recommended to have in newer distributions. When clients netboot, `ltsp init` dynamically configures a lot of other packages, like systemd, network-manager, display managers, netplan etc. Some times normal distribution updates of said packages break the netboot process, and urgent LTSP updates provided by the PPA are required to fix it.

To install the repository in Ubuntu-based distributions, run as root:

```
add-apt-repository ppa:ltsp
apt update
```

To install the repository in Debian-based distributions, run as root:

```
wget https://ltsp.org/misc/ltsp-ubuntu-ppa-focal.list -O /etc/apt/sources.list.d/ltsp-ubuntu-ppa-focal.list
wget https://ltsp.org/misc/ltsp_ubuntu_ppa.gpg -O /etc/apt/trusted.gpg.d/ltsp_ubuntu_ppa.gpg
apt update
```

Note that normally Debian users should not be using PPAs. LTSP is an exception, as it only contains shell code (and a bit of python), it is interpreted (Architecture:all, no compiled binaries involved), and the .debs are tested on Debian too.

2.5 Netboot clients

To configure the LTSP clients to boot from the network with the PXE protocol, two methods are used, depending on whether the network card is onboard, in which case you should set appropriate BIOS/UEFI settings, or whether it's in a PCI or PCI-e slot, in which case you should install iPXE.

2.5.1 BIOS/UEFI

To enable network booting from your BIOS/UEFI settings, press Del or F2 while your computer boots, and choose one or more of the following options, as appropriate:

- Enable onboard NIC
- Enable LAN boot ROM
- Boot sequence: network first

LTSP supports both BIOS and UEFI network booting, but not IPv6 network booting yet.

2.5.2 iPXE

iPXE contains various network drivers and is able to netboot most clients. It can be installed by various forms:

- win32-loader.exe: if your clients have a local disk with Microsoft Windows installed in BIOS mode, win32-loader adds a "Boot from network" option to the Windows boot manager.
- grub-ipxe: if your clients have Ubuntu etc, running `apt install grub-ipxe` adds a "Boot from network" option to the Grub boot manager.
- ipxe.lkrn: can be used in other distributions that don't have the grub-ipxe package.
- ipxe.iso: iPXE in CD ROM format.
- ipxe.usb: iPXE in USB format. Just `dd` it to a USB stick.
- ipxe.dsk: iPXE in floppy disk format.
- ipxe.efi: can be copied in a local EFI partition if the clients are UEFI and for some reason the internal PXE stack isn't appropriate.

3. Man pages

3.1 Man pages

The LTSP man pages are the best source of information for LTSP. They are generated from markdown files in the LTSP source code. You may run `man ltsp applet` to read them from your terminal, or you may read them online here:

- ltsp: entry point to Linux Terminal Server Project applets
- ltsp.conf: configuration file for LTSP
- ltsp dnsmasq: configure dnsmasq for LTSP
- ltsp image: generate a squashfs image from an image source
- ltsp info: gather support information about the LTSP installation
- ltsp initrd: create the ltsp.img initrd add-on
- ltsp ipxe: install iPXE binaries and configuration in TFTP
- ltsp kernel: copy the kernel and initrd from an image to TFTP
- ltsp nfs: configure NFS exports for LTSP
- ltsp remoteapps: run applications on the LTSP server via ssh -X

3.2 ltsp

3.2.1 NAME

ltsp - entry point to Linux Terminal Server Project applets

3.2.2 SYNOPSIS

ltsp [**-b** *base-dir*] [**-h**] [**-m** *home-dir*] [**-o** *overwrite*] [**-t** *tftp-dir*] [**-V**] [*applet*] [*applet-options*]

3.2.3 DESCRIPTION

Run the specified LTSP *applet* with *applet-options*. To get help with applets and their options, run `**man ltsp** *applet*` or `**ltsp --help** *applet*`.

3.2.4 APPLETS

The following applets are currently defined:

- **dnsmasq**: configure dnsmasq for LTSP
- **image**: generate a squashfs image from an image source
- **info**: gather support information about the LTSP installation
- **initrd**: create the ltsp.img initrd add-on
- **ipxe**: install iPXE binaries and configuration in TFTP
- **kernel**: copy the kernel and initrd from an image to TFTP
- **nfs**: configure NFS exports for LTSP

LTSP clients also have some additional applets, like **initrd-bottom**, **init** and **login**, but they're not runnable by the user.

3.2.5 OPTIONS

LTSP directories can be configured by passing one or more of the following parameters, but it's recommended that an /etc/ltsp/ltsp.conf configuration file is created instead, so that you don't have to pass them in each ltsp command.

-b, **--base-dir**=*/srv/ltsp*

> This is where the chroots, squashfs images and virtual machine symlinks are; so when you run `ltsp kernel img_name`, it will search either for a squashfs image named **/srv/ltsp/images/img_name.img**, or for a chroot named **/srv/ltsp/img_name**, if it's a directory that contains /proc. Additionally, `ltsp image img_name` will also search for a symlink to a VM disk named **/srv/ltsp/img_name.img**. $BASE_DIR is exported read-only by NFSv3, so do not put sensitive data there.

-h, **--help**

> Display a help message.

-m, **--home-dir**=*/home*

> The default method of making /home available to LTSP clients is SSHFS. In some cases security isn't an issue, and sysadmins prefer the insecure NFSv3 speed over SSHFS. $HOME_DIR is used by `ltsp nfs` to export the correct directory, if it's different to /home, and by LTSP clients to mount it.

-o, **--overwrite**[=*0|1*]

Overwrite existing files. Defaults to 1 as administrators are not supposed to manually edit LTSP autogenerated files, but maintain local content into separate files (e.g. /etc/exports.d/local.exports). If you manually maintain ltsp.ipxe, it might be a good idea to set OVERWRITE=0 in ltsp.conf.

-t, **--tftp-dir**=*/srv/tftp*

LTSP places the kernels, initrds and iPXE files in /srv/tftp/ltsp, to be retrieved by the clients via the TFTP protocol. The TFTP server of dnsmasq and tftpd-hpa are configured to use /srv/tftp as the TFTP root.

-V, **--version**

Display the version information.

3.2.6 FILES

/etc/ltsp/ltsp.conf

All the long options can also be specified as variables in the **ltsp.conf** configuration file in UPPER_CASE, using underscores instead of hyphens.

3.2.7 ENVIRONMENT

All the long options can also be specified as environment variables in UPPER_CASE, for example:

```
BASE_DIR=/opt/ltsp ltsp kernel ...
```

3.2.8 EXAMPLES

The following are the typical commands to install and maintain LTSP in chrootless mode:

```
# To install:
ltsp image /
ltsp dnsmasq
ltsp nfs
ltsp ipxe

# To update the exported image, after changes in the server software:
ltsp image /
```

The following are the typical commands to provide an additional x86_32 image, assuming one uses VirtualBox. If you specifically name it x86_32, then the ltsp.ipxe code automatically prefers it for 32bit clients:

```
ln -rs $HOME/VirtualBox\ VMs/x86_32/x86_32-flat.vmdk /srv/ltsp/x86_32.img
ltsp image x86_32
ltsp ipxe
```

3.3 ltsp.conf

3.3.1 NAME

ltsp.conf - client configuration file for LTSP

3.3.2 SYNOPSIS

The LTSP client configuration file is placed at `/etc/ltsp/ltsp.conf` and it loosely follows the .ini format. It is able to control various settings of the LTSP server and clients. After each ltsp.conf modification, the `ltsp initrd` command needs to be run so that it's included in the additional ltsp.img initrd that is sent when the clients boot.

3.3.3 CREATION

To create an initial ltsp.conf, run the following command:

```
install -m 0660 -g sudo /usr/share/ltsp/common/ltsp/ltsp.conf /etc/ltsp/ltsp.conf
```

The optional `-g sudo` parameter allows users in the sudo group to edit ltsp.conf with any editor (e.g. gedit) without running sudo.

3.3.4 SYNTAX

Open and view the /etc/ltsp/ltsp.conf file that you just created, so that it's easier to understand its syntax.

The configuration file is separated into sections:

- The special [server] section is evaluated only by the ltsp server.
- The special [common] section is evaluated by both the server and ltsp clients.
- In the special [clients] section, parameters for all clients can be defined. Most ltsp.conf parameters should be placed there.
- MAC address, IP address, or hostname sections can be used to apply settings to specific clients. Those support globs, for example [192.168.67.*].
- It's also possible to group parameters into named sections like [crt_monitor] in the example, and reference them from other sections with the INCLUDE= parameter.
- Advanced users may also use [applet/host] sections, for example [initrd-bottom/library*] would be evaluated by the `ltsp initrd-bottom` applet only for clients that have a hostname that starts with "library".

The ltsp.conf configuration file is internally transformed into a shell script, so all the shell syntax rules apply, except for the sections headers which are transformed into functions.

This means that you must not use spaces around the "=" sign, and that you may write comments using the "#" character.

The `ltsp initrd` command does a quick syntax check by running `sh -n /etc/ltsp/ltsp.conf` and aborts if it detects syntax errors.

3.3.5 PARAMETERS

The following parameters are currently defined; an example is given in each case.

ADD_IMAGE_EXCLUDES="*/etc/ltsp/add-image.excludes*"
OMIT_IMAGE_EXCLUDES="*home/**"

Add or omit items to the `ltsp image` exclusion list. Some files and directories shouldn't be included in the generated image. The initial list is defined in /usr/share/ltsp/server/image/image.excludes. It can be completely overridden by creating /etc/ltsp/image.excludes. ADD_IMAGE_EXCLUDES and OMIT_IMAGE_EXCLUDES can finetune the list by adding or removing lines to it. They can either be filenames or multiline text.

AUTOLOGIN="*user01*"

RELOGIN=*0|1*
GDM3_CONF="*WaylandEnable=false*"
LIGHTDM_CONF="*greeter-hide-users=true*"
SDDM_CONF="*/etc/ltsp/sddm.conf*"

Configure the display manager to log in this user automatically. If SSHFS is used, the PASSWORDS_x parameter (see below) must also be provided. AUTOLOGIN can be a simple username like "user01", or it can be a partial regular expression that transforms a hostname to a username. For example, AUTOLOGIN="pc/guest" means "automatically log in as guest01 in pc01, as guest02 in pc02 etc". Setting RELOGIN=0 will make AUTOLOGIN work only once. Finally, the *_CONF parameters can be either filenames or direct text, and provide a way to write additional content to the generated display manager configuration.

CRONTAB_x="*30 15 * * * root poweroff*"

Add a line in crontab. The example powers off the clients at 15:30.

CUPS_SERVER="*$SERVER*"

Set the CUPS server in the client /etc/cups/client.conf. Defaults to $SERVER. You're supposed to also enable printer sharing on the server by running `cupsctl _share_printers=1` or `system-config-printer` or by visiting http://localhost:631. Then all printers can be managed on the LTSP server. Other possible values are CUPS_SERVER="localhost", when a printer is connected to a client, or CUPS_SERVER="ignore", to skip CUPS server handling.

DEBUG_LOG=*0|1*

Write warnings and error messages to /run/ltsp/debug.log. Defaults to 0.

DEBUG_SHELL=*0|1*

Launch a debug shell when errors are detected. Defaults to 0.

DEFAULT_IMAGE="*x86_64*"
KERNEL_PARAMETERS="*nomodeset noapic*"
MENU_TIMEOUT="*5000*"

These parameters can be defined under [mac:address] sections in ltsp.conf, and they are used by `ltsp ipxe` to generate the iPXE menu. They control the default menu item, the additional kernel parameters and the menu timeout for each client. They can also be defined globally under [server].

DISABLE_SESSION_SERVICES="*evolution-addressbook-factory obex*"
DISABLE_SYSTEM_SERVICES="*anydesk teamviewerd*"
KEEP_SESSION_SERVICES="*at-spi-dbus-bus*"
KEEP_SYSTEM_SERVICES="*apparmor ssh*"
MASK_SESSION_SERVICES="*gnome-software-service update-notifier*"
MASK_SYSTEM_SERVICES="*apt-daily apt-daily-upgrade rsyslog*"

Space separated lists of services to disable, permit or mask on LTSP clients. They mostly correspond to `systemctl disable/mask [--user]` invocations. Setting these ltsp.conf parameters adds or omits items from the default lists that are defined in `/usr/share/ltsp/client/init/56-services.sh`. Disabled services can be started on demand by e.g. dbus or socket activation, while masked services need to be manually unmasked first. Currently, MASK_SESSION_SERVICES also deletes the non-systemd user services from /etc/xdg/autostart.

DNS_SERVER="*8.8.8.8 208.67.222.222*"

Specify the DNS servers for the clients.

FSTAB_x="*server:/home /home nfs defaults,nolock 0 0*"

All parameters that start with FSTAB_ are sorted and then their values are written to /etc/fstab at the client init phase.

HOSTNAME="*pc01*"

Specify the client hostname. Defaults to "ltsp%{IP}". HOSTNAME may contain the %{IP} pseudovariable, which is a sequence number calculated from the client IP and the subnet mask, or the %{MAC} pseudovariable, which is the MAC address without the colons.

HOSTS_x="*192.168.67.10 nfs-server*"

All parameters that start with HOSTS_ are sorted and then their values are written to /etc/hosts at the client init phase.

IMAGE_TO_RAM=*0|1*

Specifying this option under the [clients] section copies the rootfs image to RAM during boot. That makes clients less dependent on the server, but they must have sufficient memory to fit the image.

INCLUDE="*other-section*"

Include another section in this section.

LOCAL_SWAP=*0|1*

Activate local swap partitions. Defaults to 1.

MULTISEAT=*0|1*
UDEV_SEAT_n_x="**/usb?/?-[2,4,6,8,10,12,14,16,18]/**"

MULTISEAT=1 tries to autodetect if an LTSP client has two graphics cards and to automatically split them along with the USB ports into two seats. Optional lines like `UDEV_SEAT_1_SOUND="*/sound/card1*"` can be used to finetune the udev rules that will be generated and placed in a file named /etc/udev/rules.d/72-ltsp-seats.rules.

NAT=*0|1*

Only use this under the [server] section. Normally, `ltsp service` runs when the server boots and detects if a server IP is 192.168.67.1, in which case it automatically enables IP forwarding for the clients to be able to access the Internet in dual NIC setups. But if there's a chance that the IP isn't set yet (e.g. disconnected network cable), setting NAT=1 enforces that.

OMIT_FUNCTIONS="*pam_main mask_services_main*"

A space separated list of function names that should be omitted. The functions specified here will not be executed when called. This option can be specified in any [section].

PASSWORDS_x="*teacher/cXdlcjEyMzQK [a-z][-0-9]*/MTIzNAo= guest[^:]*/*"

A space separated list of regular expressions that match usernames, followed by slash and base64-encoded passwords. At boot, `ltsp init` writes those passwords for the matching users in /etc/shadow, so that then pamltsp can pass them to SSH/SSHFS. The end result is that those users are able to login either in the console or the display manager by just pressing [Enter] at the password prompt.
Passwords are base64-encoded to prevent over-the-shoulder spying and to avoid the need for escaping special characters. To encode a password in base64, run `base64`, type a single password, and then Ctrl+D.
In the example above, the teacher account will automatically use "qwer1234" as the password, the a1-01, b1-02 etc students will use "1234", and the guest01 etc accounts will be able to use an empty password without even authenticating against the server; in this case, SSHFS can't be used, /home should be local or NFS.

POST_APPLET_x="*ln -s /etc/ltsp/xorg.conf /etc/X11/xorg.conf*"

All parameters that start with POST_ and then have an ltsp client applet name are sorted and their values are executed after the main function of that applet. See the ltsp(8) man page for the available applets. The usual place to run client initialization commands that don't need to daemonize is POST_INIT_x.

PRE_APPLET_x="*debug_shell*"

All parameters that start with PRE_ and then have an ltsp client applet name are sorted and their values are executed before the main function of that applet.

PWMERGE_SUR=, **PWMERGE_SGR=**, **PWMERGE_DGR=**, **PWMERGE_DUR=**

Normally, all the server users are listed on the client login screens and are permitted to log in. To exclude some of them, define one or more of those regular expressions. For more information, read /usr/share/ltsp/client/login/pwmerge. For example, if you name your clients pc01, pc02 etc, and your users a01, a02, b01, b02 etc, then the following line only shows/allows a01 and b01 to login to pc01: `PWMERGE_SUR=".*%{HOSTNAME#pc}"`

REMOTEAPPS="*users-admin mate-about-me*"

Register the specified applications as remoteapps, so that they're executed on the LTSP server via `ssh -X` instead of on the clients. For more information, see **ltsp-remoteapps(8)**.

RPI_IMAGE="*raspios*"

Select this LTSP image to boot Raspberry Pis from. This symlinks all $BASE_DIR/$RPI_IMAGE/boot/* files directly under $TFTP_DIR when `ltsp kernel $RPI_IMAGE` is called. See the Raspberry Pi OS documentation page for more information.

SEARCH_DOMAIN="*ioa.sch.gr*"

A search domain to add to resolv.conf and to /etc/hosts. Usually provided by DHCP.

SERVER="*192.168.67.1*"

The LTSP server is usually autodetected; it can be manually specified if there's need for it.

X_DRIVER="*vesa*"
X_HORIZSYNC="*28.0-87.0*"
X_MODELINE='"*1024x768_85.00" 94.50 1024 1096 1200 1376 768 771 775 809 -hsync +vsync*'
X_MODES='"*1024x768" "800x600" "640x480"*'
X_PREFERREDMODE="*1024x768*"
X_VERTREFRESH="*43.0-87.0*"
X_VIRTUAL="*800 600*"

If any of these parameters are set, the /usr/share/ltsp/client/init/xorg.conf template is installed to /etc/X11/xorg.conf, while applying the parameters. Read that template and consult xorg.conf(5) for more information. The most widely supported method to set a default resolution is X_MODES. If more parameters are required, create a custom xorg.conf as described in the EXAMPLES section.

3.3.6 EXAMPLES

To specify a hostname and a user to autologin in a client:

```
[3c:07:71:a2:02:e3]
HOSTNAME=pc01
AUTOLOGIN=user01
PASSWORDS_PC01="user01/cGFzczAxCg=="
```

The password above is "pass01" in base64 encoding. To calculate it, the `base64` command was run in a terminal:

```
base64
pass01
<press Ctrl+D at this point>
cGFzczAxCg==
```

If some clients need a custom xorg.conf file, create it in e.g. `/etc/ltsp/xorg-nvidia.conf`, and put the following in ltsp.conf to dynamically symlink it for those clients at boot:

```
[pc01]
INCLUDE=nvidia

[nvidia]
POST_INIT_LN_XORG="ln -sf ../ltsp/xorg-nvidia.conf /etc/X11/xorg.conf"
```

Since ltsp.conf is transformed into a shell script and sections into functions, it's possible to directly include code or to call sections at POST_APPLET_x hooks.

```
[clients]
# Allow local root logins by setting a password hash for the root user.
# The hash contains $, making it hard to escape in POST_INIT_x="sed ...".
# So put sed in a section and call it at POST_INIT like this:
POST_INIT_SET_ROOT_HASH="section_set_root_hash"

# This is the hash of "qwer1234"; cat /etc/shadow to see your hash.
[set_root_hash]
sed 's|^root:[^:]*:|root:$6$VRfFL349App5$BfxBbLE.tYInJfeqyGTv2lbk6KOza3L2AMpQz7bMuCdb3ZsJacl9Nra7F/Zm7WZJbnK5kvK74Ik9WO2qGietM0:|' -i /etc/shadow
```

3.4 ltsp-dnsmasq

3.4.1 NAME

ltsp dnsmasq - configure dnsmasq for LTSP

3.4.2 SYNOPSIS

ltsp [*ltsp-options*] **dnsmasq** [**-d** *dns*] [**-p** *proxy-dhcp*] [**-r** *real-dhcp*] [**-s** *dns-server*] [**-t** *tftp*]

3.4.3 DESCRIPTION

Install /etc/dnsmasq.d/ltsp-dnsmasq.conf, while adjusting the template with the provided parameters.

3.4.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

-d, **--dns**=*0|1*

Enable or disable the DNS service. Defaults to 0. Enabling the DNS service of dnsmasq allows caching of client requests, custom DNS results, blacklisting etc, and automatically disables DNSStubListener in systemd-resolved on the LTSP server.

-p, **--proxy-dhcp**=*0|1*

Enable or disable the proxy DHCP service. Defaults to 1. Proxy DHCP means that the LTSP server sends the boot filename, but it leaves the IP leasing to an external DHCP server, for example a router or pfsense or a Windows DHCP server. It's the easiest way to set up LTSP, as it only requires a single NIC with no static IP, no need to rewire switches etc.

-r, **--real-dhcp**=*0|1*

Enable or disable the real DHCP service. Defaults to 1. In dual NIC setups, you only need to configure the internal NIC to a static IP of 192.168.67.1; LTSP will try to autodetect everything else. The real DHCP service doesn't take effect if your IP isn't 192.168.67.x, so there's no need to disable it in single NIC setups unless you want to run isc-dhcp-server on the LTSP server.

-s, **--dns-server**="*space separated list*"

Set the DNS server DHCP option. Defaults to autodetection. Proxy DHCP clients don't receive DHCP options, so it's recommended to use the ltsp.conf DNS_SERVER parameter when autodetection isn't appropriate.

-t, **--tftp**=*0|1*

Enable or disable the TFTP service. Defaults to 1.

3.4.5 EXAMPLES

Create a default dnsmasq configuration, overwriting the old one:

```
ltsp dnsmasq
```

A dual NIC setup with the DNS service enabled:

```
ltsp dnsmasq -d1 -p0 --dns-server="0.0.0.0 8.8.8.8 208.67.222.222"
```

3.5 ltsp-image

3.5.1 NAME

ltsp image - generate a squashfs image from an image source

3.5.2 SYNOPSIS

ltsp [*ltsp-options*] **image** [**-b** *backup*] [**-c** *cleanup*] [**-i** *ionice*] [**-k** *kernel-initrd*] [**-m** *mksquashfs-params*] [**-r** *revert*] [*image*] ...

3.5.3 DESCRIPTION

Compress a virtual machine image or chroot directory into a squashfs image, to be used as the network root filesystem of LTSP clients. It's used in similar fashion to live CDs, i.e. all clients will boot from this single read only image and then use SSHFS or NFS to mount /home/username from the server.

3.5.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

-b, **--backup**=*0|1*

Backup /srv/ltsp/images/*image*.img to *image*.img.old. Defaults to 1.

-c, **--cleanup**=*0|1*

Create a writeable overlay on top of the image source and temporarily remove user accounts and sensitive data before calling mksquashfs. Defaults to 1.

-i, **--ionice**=*cmdline*

Set a prefix command to run mksquashfs with a lower priority, or specify "" to disable it completely. Defaults to `nice ionice -c3`.

-k, **--kernel-initrd**=*glob-regex*

Pass this parameter to the `ltsp kernel` call after the squashfs creation. See ltsp-kernel(8) for more information.

-m, **--mksquashfs-params**="*params*"

Pass *$params* to the mksquashfs call unquoted; so *params* shouldn't contain spaces. See mksquashfs(1) for more information.

-r, **--revert**[=*0|1*]

Move /srv/ltsp/images/*image*.img.old to *image*.img and call `ltsp kernel image`. Useful when the clients won't boot with the new image.

3.5.5 IMAGE TYPES

There are three "image" types in LTSP, in the following locations. The /srv/ltsp path can be configured using `ltsp --base-dir=`:

/srv/ltsp/`img_name`.img

Source images are placed directly under /srv/ltsp and usually are symlinks to virtual machine raw disk files. They're only used by `ltsp image`.

/srv/ltsp/ `img_name`

Chroot directories can be used both as sources for `ltsp image` and as NFS root exports for the clients.

/srv/ltsp/images/ `img_name` **.img**

Exported images (usually squashfs) are placed under the images directory and the clients can netboot from them.

Images can be specified as simple names like `ltsp image img_name`, in which case the aforementioned locations are searched, or as or full paths like `ltsp image ~/VMs/vm.img`.

The supported image types result in the following three methods to use LTSP. You may use either one of the methods or even all of them at the same time.

3.5.6 CHROOTLESS

Chrootless LTSP, previously called "ltsp-pnp", is the recommended way to maintain LTSP **if** its restrictions are acceptable. In this mode, the server operating system itself is exported into a squashfs file and used for netbooting all the clients. You, the sysadmin, would use the typical GUI tools to manage the server, like software centers or update managers. Then whenever necessary, you'd run:

```
ltsp image /
```

This creates or updates /srv/ltsp/images/x86_64.img (the arch name comes from `uname -m`). Then, all the clients should be able to boot from x86_64.img and have a desktop environment identical to the server.

The big advantage of the chrootless mode is simplicity: there are no virtual machines or chroots involved. You'd maintain the server like any "home desktop PC", and have all clients be exact replicas, which is as simple as it gets.

The disadvantages are that the clients need to have the same architecture as the server (e.g. all x86_64), and that the server can't be a "full blown server" with LDAP and Apache and a lot of other services, without taking care to disable those services on the clients with the MASK_SYSTEM_SERVICES parameter of ltsp.conf. Note that MASK_SYSTEM_SERVICES already includes Apache and MySQL and a few other popular services that we don't want in LTSP clients, so it's not a problem if you install Apache on the LTSP server.

If for some reason you prefer a different name to `uname -m`, you may create a symlink:

```
ln -s / ~/amd64
```

...and run `ltsp image ~/amd64` instead.

3.5.7 VM IMAGES

If the chrootless case doesn't fit you, you may use VirtualBox, virt-manager, KVM, VMWare and similar tools to maintain one or more template images for the clients. As an example, let's suppose you create a VM in VirtualBox and call it "debian". At the disk creation dialog, select "VMDK" type and "Fixed size", not "Dynamically allocated". Proceed with installing Debian on it. In the partitioning step, make sure that the whole operating system goes in the first partition, without extra partitions for /boot etc. BIOS/MBR is easier, while if you have to use GPT/UEFI, put the EFI partition second. When you're done, close VirtualBox and symlink the VM disk so that LTSP finds it more easily:

```
ln -rs ~/VirtualBox\ VMs/debian/debian-flat.vmdk /srv/ltsp/debian.img
```

To export this image to the clients, after the initial creation or after updates etc, you'd run:

```
ltsp image debian
```

It's also possible to omit the symlink by running:

```
ltsp image ~/VirtualBox\ VMs/debian/debian-flat.vmdk
```

...but then the image name shown in the iPXE boot menu would be "debian-flat", which isn't pretty.

In summary, you may symlink raw VM disks in /srv/ltsp/img_name.img, and `ltsp image img_name` will allow LTSP clients to netboot from them. Please also see the DIRECT IMAGES section of ltsp-kernel(8) for an advanced method of allowing clients to netboot directly from a VM or .iso image without even running `ltsp image`, and the ADVANCED IMAGE SOURCES section of ltsp-ipxe(8) for extreme cases like telling the LTSP clients to boot from an .iso image inside a local disk partition!

3.5.8 CHROOTS

Chroot directories in /srv/ltsp/img_name are properly supported as image sources by LTSP, but their creation and maintenance are left to external tools like debootstrap, lxc etc. The `ltsp-build-client` LTSPv5 tool no longer exists. LTSP users are invited to create appropriate documentation in the community wiki. As a small example, you can use kvm to netboot a chroot and maintain it if you NFS-export /srv/ltsp/img_name in rw mode for your server IP, and then run

```
kvm -m 512 -kernel img_name/vmlinuz -initrd img_name/initrd.img \
    -append "rw root=/dev/nfs nfsroot=192.168.67.1:/srv/ltsp/img_name"
```

3.5.9 EXAMPLES

Use the server installation as a template to generate a client image (chrootless, previously called ltsp-pnp):

```
ltsp image /
```

Inform `ltsp image` that a chrootless installation uses separate /boot and /opt partitions:

```
ltsp image /,,/boot,subdir=boot,,/opt,subdir=opt
```

Compress the /srv/ltsp/x86_64 chroot or the /srv/ltsp/x86_64.img virtual machine image, whichever exists of those two, into /srv/ltsp/images/x86_64.img, while disabling ionice:

```
ltsp image --ionice="" x86_64
```

Specify an absolute path to a virtual machine image:

```
ltsp image /home/user/VirtualBox\ VMs/x86_32/x86_32-flat.vmdk
```

Revert to the the previous version of the "chrootless" image:

```
ltsp image -r /
```

3.6 ltsp-info

3.6.1 NAME

ltsp info - display troubleshooting information about ltsp server and images

3.6.2 SYNOPSIS

ltsp [*ltsp-options*] **info**

3.6.3 DESCRIPTION

Display the LTSP version and the available chroots, VMs and images. This applet is very minimal currently, it will be developed further in the future.

3.6.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

3.7 ltsp-initrd

3.7.1 NAME

ltsp initrd - create the ltsp.img initrd add-on

3.7.2 SYNOPSIS

ltsp [*ltsp-options*] **initrd**

3.7.3 DESCRIPTION

Create a secondary initrd in /srv/tftp/ltsp/ltsp.img, that contains the LTSP client code from /usr/share/ltsp/{client,common} and everything under /etc/ltsp, including the ltsp.conf settings file. Additionally it contains the server users and groups lists (passwd/group) and public SSH keys. LTSP clients receive this initrd in addition to their usual one.

This means that whenever you add new users or edit **ltsp.conf(5)**, you need to run `ltsp initrd` to update **ltsp.img**, and reboot the clients.

It also means that you can very easily put template xorg.conf or sshfs or other files in /etc/ltsp, and have them on the clients in seconds, without having to run `ltsp image`.

3.7.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

3.7.5 EXAMPLES

Most live CDs do not contain sshfs, so by default you can only use NFS home with them. But sshfs is a small binary without many dependencies, so you may usually provide it to the clients if you include it to ltsp.img:

```
mkdir -p /etc/ltsp/bin
cp /usr/bin/sshfs /etc/ltsp/bin/sshfs-$(uname -m)
ltsp initrd
```

You can even provide multiple sshfs versions for different architectures. LTSP contains code to automatically use those sshfs binaries if it can't find the /usr/bin/sshfs one.

3.8 ltsp-ipxe

3.8.1 NAME

ltsp ipxe - install iPXE binaries and configuration in TFTP

3.8.2 SYNOPSIS

ltsp [*ltsp-options*] **ipxe** [**-b** *binaries*]

3.8.3 DESCRIPTION

Generate the ltsp.ipxe configuration file and install the required iPXE binaries in /srv/tftp/ltsp: memtest.0, memtest.efi, snponly.efi and undionly.kpxe.

An ltsp-binaries package is available in the LTSP PPA that provides them; otherwise, some of them are automatically found in the ipxe/memtest86+ packages.

3.8.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

-b, **--binaries**[=*0*|*1*|""]

Reinstall the iPXE binaries in TFTP even if they already exist. Defaults to "", which means "only install the missing ones". Note that the --overwrite flag doesn't affect the binaries, they're only controlled by the --binaries flag.

3.8.5 ADVANCED IMAGE SOURCES

This section is for advanced LTSP sysadmins. Normally, image sources are simple names like "x86_64" or full paths like "../path/to/image". But the "img_src" parameters are much more flexible than that; specifically, they are series of mount sources:

```
img1,mount-options1,,img2,mount-options2,,...
```

...where img1 may be a simple name or full path relative to the current directory, and img2+ are full paths relative to the target directory.

Let's see an advanced example: suppose that your clients came with Windows, and that you copied a live CD into C:\ltsp\ubuntu.iso, and you want your LTSP clients to use that for speed. First, disable Windows fast boot and hibernation, so that Linux is able to mount its partition. Then create the following "method" in ltsp.ipxe:

```
:local_image
# The "local_image" method boots C:\ltsp\ubuntu.iso
set cmdline_method root=/dev/sda1 ltsp.image=ltsp/ubuntu.iso,fstype=iso9660,loop,ro,,casper/filesystem.squashfs,squashfs,loop,ro loop.max_part=9
goto ltsp
```

Explanation:

- The root=/dev/sda1 parameter tells the initramfs to mount /dev/sda1 into /root.
- Then the LTSP code will look under /root/ltsp/ and mount ubuntu.iso using the loop,ro options over /root again.
- Then the LTSP code will look under /root/casper/ and mount filesystem.squashfs over /root again. This casper/filesystem.squashfs path is where the live filesystem exists inside the Ubuntu live CDs.

So while this long line gives a good example on using advanced image sources, the LTSP code is actually smart enough to autodetect Ubuntu live CDs and filesystem types, so one could simplify it to:

```
:local_image
# The "local_image" method boots C:\ltsp\${img}.img
```

```
set cmdline_method root=/dev/sda1 ltsp.image=ltsp/${img}.img loop.max_part=9
goto ltsp
```

The ${img} parameter is the name of the menu; it would be "ubuntu" if you copied ubuntu.iso in /srv/ltsp/images/ubuntu.img and ran `ltsp ipxe`.

3.8.6 EXAMPLES

Initial use:

```
ltsp ipxe
```

Regenerate ltsp.ipxe and reinstall the binaries:

```
ltsp ipxe -b
```

Copy the binaries from a USB stick before running ltsp ipxe:

```
mkdir -p /srv/tftp/ltsp
cd /media/administrator/usb-stick
cp {memtest.0,memtest.efi,snponly.efi,undionly.kpxe} /srv/tftp/ltsp
ltsp ipxe
```

3.9 ltsp-kernel

3.9.1 NAME

ltsp kernel - copy the kernel and initrd from an image to TFTP

3.9.2 SYNOPSIS

ltsp [*ltsp-options*] **kernel** [**-k** *kernel-initrd*] [*image*] ...

3.9.3 DESCRIPTION

Copy vmlinuz and initrd.img from an image or chroot to TFTP. If *image* is unspecified, process all of them. For simplicity, only chroot directories and raw images are supported, either full filesystems (squashfs, ext4) or full disks (flat VMs). They may be sparse to preserve space. Don't use a separate /boot nor LVM in disk images. The targets will always be named vmlinuz and initrd.img to simplify ltsp.ipxe.

3.9.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

-k, **--kernel-initrd**=*glob-regex*

Specify a kernel glob and an initrd regex to locate them inside the *image*; try to autodetect if undefined. See the EXAMPLES section below.

3.9.5 DIRECT IMAGES

This section is for advanced LTSP sysadmins. Let's suppose that you want to test if your users would prefer Xubuntu to your existing Ubuntu MATE. First, move and rename your Xubuntu CD to this location, without using symlinks, and then update kernels and ipxe:

```
mv xubuntu-18.04-desktop-amd64.iso /srv/ltsp/images/xubuntu-18.04.img
ltsp kernel xubuntu-18.04
ltsp ipxe
```

If you reboot your clients, they'll now have the option to boot with the Xubuntu live CD in LTSP mode! This is like booting with the live CD, except that all the users and their homes are available! So the users can normally login and work for days or weeks in the new environment, before you decide that they like Xubuntu and that you want to move from using the live CD to maintaining a Xubuntu image using a virtual machine.

You can also do this with virtual machine images! For example:

```
mv ~/VirtualBox\ VMs/debian/debian-flat.vmdk /srv/ltsp/images/debian-vm.img
ln -rs /srv/ltsp/images/debian-vm.img ~/VirtualBox\ VMs/debian/debian-flat.vmdk
ltsp kernel debian-vm
ltsp ipxe
```

These commands move your "debian" VM to the LTSP images directory, symlink it back to where VirtualBox expects it, and update the kernels and ipxe. After these, you'll be able to boot directly from the "debian-vm" iPXE menu item without having to run `ltsp image`! It's the fastest way to test image changes without waiting 10 minutes for `ltsp image` each time.

Some advanced users may think of using the opposite symlink instead:

```
ln -rs ~/VirtualBox\ VMs/debian/debian-flat.vmdk /srv/ltsp/images/debian-vm.img
```

Unfortunately NFS doesn't follow symlinks outside of the exported directories, so the clients wouldn't be able to boot in this case. Advanced users may use bind mounts though, e.g.:

```
mount --bind ~/VirtualBox\ VMs/debian/debian-flat.vmdk /srv/ltsp/images/debian-vm.img
```

3.9.6 EXAMPLES

Typical use:

```
ltsp kernel x86_64
```

Passing a glob to locate the kernel and a regex to locate the initrd in a Debian live CD:

```
ltsp kernel --kernel-initrd="live/vmlinuz-* s|vmlinuz|initrd.img|"
```

3.10 ltsp-nfs

3.10.1 NAME

ltsp nfs - configure NFS exports for LTSP

3.10.2 SYNOPSIS

ltsp [*ltsp-options*] **nfs** [**-h** *nfs-home*] [**t** *nfs-tftp*]

3.10.3 DESCRIPTION

Install /etc/exports.d/ltsp-nfs.conf in order to export /srv/ltsp ($BASE_DIR), /srv/tftp/ltsp ($TFTP_DIR) and optionally /home ($HOME_DIR).

3.10.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

-h, **--nfs-home**=*0|1*

Export /home over NFS3. Defaults to 0. Note that NFS3 is insecure for home, so by default SSHFS is used. To specify a different directory, set $HOME_DIR in /etc/ltsp/ltsp.conf.

-t, **--nfs-tftp**=*0|1*

Export /srv/tftp/ltsp over NFS3. Defaults to 1. To specify a different directory, set $TFTP_DIR in /etc/ltsp/ltsp.conf.

3.10.5 EXAMPLES

To export /home over NFS (insecure), use the following ltsp.conf parameters:

```
[server]
NFS_HOME=1

[clients]
FSTAB_HOME="server:/home /home nfs defaults,nolock 0 0"
```

And run these commands on the server:

```
ltsp initrd  # This is needed whenever ltsp.conf is modified
ltsp nfs
```

To export only some user homes over NFS while the rest still use SSHFS, use these lines in ltsp.conf instead:

```
[server]
NFS_HOME=1
HOME_DIR=/home/nfs

[clients]
FSTAB_HOME="server:/home/nfs /home nfs defaults,nolock 0 0"
```

Then run the following commands on the server, to move some home directories under /home/nfs and to create appropriate symlinks in case the users ever need to SSH to the server. Note that the NFS server doesn't follow symlinks outside of an export:

```
mkdir /home/nfs
for u in guest01 guest02; do
    mv "/home/$u" /home/nfs/
    ln -s "nfs/$u" "/home/$u"
done

ltsp initrd
ltsp nfs
```

3.11 ltsp-remoteapps

3.11.1 NAME

ltsp remoteapps - run applications on the LTSP server via ssh -X

3.11.2 SYNOPSIS

ltsp [*ltsp-options*] **remoteapps** *application* [*parameters*] ...
ltsp [*ltsp-options*] **remoteapps** [**-r** *register*] *application* ...

3.11.3 DESCRIPTION

Setup passwordless SSH, by ensuring that ~/.ssh/authorized_keys contains one of the public user's SSH keys. Then execute `ssh -X server app params`. If the user has no SSH keys, a new one is generated. For `ltsp remoteapps app` to work, /home/username should already be mounted via NFS (on LTSP client boot) or SSHFS (on LTSP user login).

3.11.4 OPTIONS

See the **ltsp(8)** man page for *ltsp-options*.

-r, **--register**

Register the specified applications as remoteapps, by creating symlinks in /usr/local/bin/*application* for each one of them. Since `/usr/local/bin` usually comes before `/usr/bin` in `$PATH`, the remoteapps should be invoked even if the application is selected from the system menu.

3.11.5 EXAMPLES

The following ltsp.conf parameter can be used to register the MATE applications `users-admin` (Menu ▸ System ▸ Administration ▸ Users and Groups) and `mate-about-me` (Menu ▸ System ▸ Preferences ▸ Personal ▸ About Me) as remoteapps:

```
[clients]
REMOTEAPPS="users-admin mate-about-me"
```

That way, LTSP users are able to change their passwords or display names. The password change takes effect immediately, while for the new display name to appear in the LTSP client, the sysadmin must run `ltsp initrd` and the client needs to be rebooted.

4. Guides

4.1 Guides

LTSP follows the Diátaxis framework, which identifies four documentation models:

- Tutorials (learning-oriented): ltsp.org/docs is meant to be read like a book.
- How-to guides (task-oriented): ltsp.org/guides can be searched for guidance on specific tasks.
- Explanation (understanding-oriented): github.com/ltsp/ltsp/discussions is available for asking and providing clarifications.
- Reference (information-oriented): ltsp.org/man contains the man pages.

This means that the guides section isn't meant to be read sequentially. Use the search box or the following table of contents to locate the one you're interested in.

Documentation contributors and pull requests are most welcome, please contact us in the online chat room for details.

4.1.1 Available guides

4.2 Chat room

LTSP currently has two online chat rooms with mirrored content:

- Matrix room: the main room; it requires signing up for a free matrix account. It also offers logging, voice/video chat and an android application.
- Libera.chat IRC channel: an IRC channel, bridged with the Matrix room. It allows unregistered users; logs are available at https://irclogs.ltsp.org.

The same rooms can also be used for Epoptes support.

Note that the Freenode IRC channel has been deprecated since the move to libera.chat and it's not used anymore.

4.3 Coding

4.3.1 Documentation

Man pages for ltsp applets are maintained in markdown format in `$SRC/docs`. At build time, `ronn` is used to convert them to man format.

The short usage, `ltsp --help $APPLET`, comes from `man ltsp $APPLET | grep ...` and is implemented in the `usage()` function.

4.3.2 PEP8

Since shell doesn't have something like PEP8, let's apply whatever makes sense from PEP8. For example, 4 spaces indentation.

TODO: heredocs <<- use tabs, and also see: https://www.kernel.org/doc/html/v4.10/process/coding-style.html#indentation.

Other remarks:

- Don't use periods for single sentence comments or commit messages. Do use periods for multiple sentences. Start with a capital letter, like in online chat rooms.
- Don't use quotes for numeric constants, e.g. `NAT=0`.
- Use quotes for strings, e.g. `HOSTNAME="pc01"`.
- Prefer "=" over "-eq" when a variable might not be numeric even by mistake, e.g. `if [ "$NAT" = 0 ]`.

4.3.3 Symlinks

If possible, put everything in /usr/share/ltsp and symlink binaries etc to appropriate places in FHS.

4.3.4 Shell script coding

Shell scripts should be able to run on `bash`, `dash`, and `busybox ash`, and they should produce no output when checked with shellcheck like `shellcheck -e SC2039 script.sh`. Occasionally, checkbashisms can also be used, but ignore its warnings about "local" and "command -v"; we want these.

Shell script extensions

Use the .sh extension for all shell scripts. Symlink the *public* binaries in /usr/[s]bin without extensions.

Variable case and visibility

Almost all the code should be inside functions. Most of the variables should be local and lowercase. Global variables that can be defined in the command line or in configuration files should be UPPERCASE, while global variables for internal use should start with an _UNDERSCORE. When subprocesses are spawned, make sure they only have the environment variables they need.

Functions

Applets start with their `$applet_cmdline()` and `$applet_main()` functions for readability. The rest of the functions should be alphabetically sorted. Complex functions should be preceded by a comment.

Script directories

Most scripts are organized in directories, and numbered as follows:

- [0-9][09]|[09][0-9]: site and local admins
- [1-8][18]|[18][1-8]: distributions, their derivatives and third part programs (sch-scripts, epoptes)
- [2-7][2-7]: upstream

`sort -V` is used for ordering, so 44-upstream~distro could be a distro override that runs exactly before 44-upstream, while 44-upstream-distro would run afterwards. Extra numbering can be appended when necessary, e.g. 42-0-config-first, 42-1-config-second. Distro or local scripts with the same name in /etc/ltsp/$applet/ completely override the respective upstream scripts.

Each NN-script.sh is expected to have a main function derived from the script name: `main_$script()`. All scripts are first sourced before their main functions are executed. This allows for shell function overriding, e.g. an 8x-distro `install_package()` function would override a 5x-upstream `install_package()` function.

One of the scripts, ideally the 55-applet.sh, is supposed to provide a function named `$applet_cmdline()`. This should call getopts to parse the command line and convert it to "$@" with `eval set`, and then call the `run_main_functions() "$@"` function, so that all main functions get the correct parameters.

4.4 ISC DHCP server

Dnsmasq is the recommended default for LTSP, as it's the only one that supports the proxyDHCP protocol. In some cases though, sysadmins might need isc-dhcp-server. The following template dhcpd.conf works out of the box for dual NIC LTSP setups:

```
# This file is part of LTSP, https://ltsp.org
# Copyright 2019 the LTSP team, see AUTHORS
# SPDX-License-Identifier: GPL-3.0-or-later

# Configure isc-dhcp-server for LTSP
# Documentation=man:ltsp(8)

authoritative;
# option domain-name "example.org";
option domain-name-servers 192.168.67.1, 8.8.8.8, 208.67.222.222;
option space ipxe;
option ipxe-encap-opts code 175 = encapsulate ipxe;
option ipxe.menu code 39 = unsigned integer 8;
option ipxe.no-pxedhcp code 176 = unsigned integer 8;
option arch code 93 = unsigned integer 16;

# This is the LTSP subnet declaration
subnet 192.168.67.0 netmask 255.255.255.0 {
  range 192.168.67.20 192.168.67.250;
  option ipxe.no-pxedhcp 1;
  option routers 192.168.67.1;
  # On single-NIC setups, usually routers != next-server (=TFTP server)
  # option next-server 192.168.67.1
  if exists ipxe.menu {
    filename "ltsp/ltsp.ipxe";
  } elsif option arch = 00:00 {
    filename "ltsp/undionly.kpxe";
  } elsif option arch = 00:07 {
    filename "ltsp/snponly.efi";
  } elsif option arch = 00:09 {
    filename "ltsp/snponly.efi";
  } else {
    filename "ltsp/unmatched-client";
  }
}

# Example for a host with static IP address and default iPXE menu entry
# host pc01 {
#   hardware ethernet 3c:07:71:a2:02:e3;
#   fixed-address 192.168.67.7;
#   option host-name "pc01";
#   option root-path "ipxe-menu-item";
# }
```

Put it in /etc/dhcpd/dhcpd.conf. If you have Ubuntu and LTSP5, delete /etc/ltsp/dhcpd.conf as it interferes. Then if you also have dnsmasq installed, disable its DHCP service. Finally, restart isc-dhcp-server:

```
ltsp dnsmasq --proxy-dhcp=0 --real-dhcp=0
systemctl restart isc-dhcp-server
```

If isc-dhcp-server runs somewhere else and not on the LTSP server, you'll probably need to modify the subnet range, the routers, and uncomment next-server and set it to the LTSP server IP. **Note** that in this case, it's easier to configure your external DHCP server **not** to offer a boot filename, and just run `ltsp dnsmasq` to use its default proxyDHCP mode.

4.5 Markdown

The LTSP project uses GitHub Flavored Markdown, with a few differences noted below.

4.5.1 Inline elements

Markdown uses two trailing spaces at the end of a line to indicate a `<br />`.

Text formatting

Element	HTML	When to use
`*asterisk1*`	em	*Emphasis*
`**asterisk2**`	strong	**Strong importance**
`***asterisk3***`	em>strong[1]	For ▸ menus and GUI labels
`` `backtick` ``	code	`commands, filenames, terminal stuff`
``*`a1>b`*``	em>code	*`Smart roles`* [2]
``**`a2>b`**``	strong>code	Keystrokes: **`Ctrl`**+**`Alt`**+**`Del`**

NOTES

1. Pymarkdown renders `***` as `strong>em`, while markdown-it as `em>strong`. We cover both cases in our CSS.
2. Smart roles are not implemented nor used yet. For example, *`page`* could automatically link to page.md and show a tooltip with its title.
3. In vscode-markdown-extended, underscore is rendered as u (underline) instead of em. Also some other parsers have issues with underscores. Avoid them, e.g. don't use `_text_` for emphasis.

Images

A normal image is `![alt](image.png)` and a link is `[text](url)`.

For right-aligned, shrinked, clickable images, use this syntax: `[![](image.png)](image.png)`

4.5.2 Block elements

Code blocks

Use fenced code blocks for multiline shell code:

~~~
```shell
code
```
~~~

Blank lines are not necessary. Do not use four spaces to indicate code blocks as some markdown implementations get confused e.g. in list > code, then code.

Lists

In the original markdown and in pymarkdown, indentation is always four spaces. This is necessary when we need to nest lists, admonitions etc, so we may use it always. Typing `Tab` after the dash is usually enough.

1. For numbered lists, `1.␣␣actual numbers` with two spaces

- For bulleted lists, `-␣␣␣dashes` with three spaces

Admonitions

For now the pymarkdown-extras syntax is used, which is also supported by vscode-markdown-extended:

Optional tip title

Let's hope pymarkdown gets support for general purpose blocks similar to pandoc's fenced divs!

4.5.3 Markdown research notes

Classes

It's possible to add classes to any inline element by appending `{.class}` to its right, while in block elements it goes underneath. Kramdown uses `{:.class}` instead, and it's supported by pymarkdown, but not by vscode-markdown-extended.

Admonitions

> [!NOTE]
> Blockquotes are used for admonitions by the Microsoft Docs Authoring Pack

> 📝 *NOTE*
> The same thing could be achieved by standard markdown and classes. Unfortunately, pymarkdown can only add the class to either the first or the last blockquote line, not to all of it.

4.6 Netconsole

Linux logs important messages, like out-of-memory conditions or module problems or kernel crashes, to the kernel ring buffer, which we can normally inspect with the `dmesg` command. They also show up on the screen if Linux is running in text mode, but not if it's running in graphics mode.

This means that if an LTSP client crashes, we frequently can't even see why it crashed. To remedy this, one can enable the netconsole module, and forward the client kernel messages to the server.

To enable netconsole:

- First run `ip l` on a client to see its Ethernet adapter name, for example `enp0s3`.
- Then add this parameter to the client kernel cmdline: `netconsole=@${ip}/enp0s3,@${srv}/`
  If you're not using iPXE, manually replace `${ip}` and `${srv}` with the IPs of the client and the server.
- You may also add `loglevel=5` to see more messages; the default is 4, and the valid values are 0-8.
- If you use loglevel, you may also need `POST_INIT_NETCONSOLE="rm -f /etc/sysctl.d/10-console-messages.conf"` in ltsp.conf, otherwise that file resets the `/proc/sys/kernel/printk` contents to `4 4 1 7`.

When you want to inspect the client messages, run one of the following commands on the server:

- `socat UDP-LISTEN:6666,fork -` (safer)
- `nc -l -u 6666`

Note: this was tested on initramfs-tools (Debian-based), but not on dracut (Fedora-based).

4.7 Proposed PPA

See also: LTSP PPA documentation.

LTSP releases are normally published in the LTSP proposed PPA and after a few days, if noone complains about regressions, they are copied to the LTSP PPA. That means that most LTSP administrators should have the proposed PPA in some test installation or in their personal computers etc, to be able to test new releases and file issues before they reach their production setups.

To install the repository in Ubuntu-based distributions, run as root:

```
add-apt-repository ppa:ltsp/proposed
apt update
```

To install the repository in Debian-based distributions, run as root:

```
wget https://ltsp.org/misc/ltsp-ubuntu-proposed-focal.list -O /etc/apt/sources.list.d/ltsp-ubuntu-proposed-focal.list
wget https://ltsp.org/misc/ltsp_ubuntu_ppa.gpg -O /etc/apt/trusted.gpg.d/ltsp_ubuntu_ppa.gpg
apt update
```

4.8 PXELinux

iPXE is the recommended default for LTSP, as it has great support for UEFI and HTTP, it provides a scripting language with variables, conditional instructions, dynamic menus etc. It even includes drivers for many popular network cards, and it's required for netbooting when the NIC isn't onboard and there's no BIOS "driver" for it.

For onboard NICs, iPXE tries to implement the aforementioned functionality using /srv/tftp/ltsp/undionly.kpxe. This is a wrapper driver that uses the underlying BIOS PXE stack, but in certain cases the PXE stack may be buggy and cause iPXE to either fail with e.g. "Network unreachable" or even to completely hang.

One workaround is to execute the following command on the LTSP server in order to replace undionly.kpxe with ipxe.pxe:

```
sudo wget https://boot.ipxe.org/ipxe.pxe -O /srv/tftp/ltsp/undionly.kpxe
```

While undionly.kpxe only contains the wrapper driver, ipxe.pxe contains all the iPXE drivers. If a native driver exists, it'll be used and it will probably work fine.

Thus, the rest of this page refers to the following scenario:

- The NIC is onboard (it has a PXE stack)
- You're using BIOS, not UEFI
- Neither undionly.kpxe nor ipxe.pxe worked

If these conditions are met, PXELinux might help, as it's simpler and doesn't exploit all the functionality of the (possibly buggy) PXE stack. To install PXELinux on your LTSP server, run:

```
sudo -i
cd /srv/tftp/ltsp
apt install pxelinux syslinux
ln -s /usr/lib/PXELINUX/pxelinux.0 pxelinux.0
ln -s /usr/lib/syslinux/modules/bios isolinux
mkdir -p pxelinux.cfg
wget https://ltsp.org/guides/pxelinux.txt -O pxelinux.cfg/default
```

The next step is to configure dnsmasq to point one or more clients to use PXELinux instead of iPXE. Create `/etc/dnsmasq.d/local.conf` with the following content:

```
# Documentation=https://ltsp.org/guides/pxelinux/
dhcp-mac=set:pxelinux,52:54:61:67:00:01
pxe-service=tag:pxelinux,X86PC,"pxelinux.0",ltsp/pxelinux.0
dhcp-boot=tag:pxelinux,ltsp/pxelinux.0
```

In the `dhcp-mac=` line, it's possible to use wildcards, e.g. `dhcp-mac=set:pxelinux,08:00:27:*:*:*` would match all VirtualBox clients. Run the following command to restart dnsmasq:

```
sudo systemctl restart dnsmasq
```

Then restart the problematic LTSP clients. If they boot successfully with PXELinux, you may optionally fine-tune its menu by editing its configuration file, `/srv/tftp/ltsp/pxelinux.cfg/default`.

4.9 Specifications

When asking for LTSP advice, it's a good idea to share as much information as possible. Here's an example:

- 1 server: i7-8750H with 12403 cpubenchmark score and 8 GB RAM
- 10 clients: i3-6100 with 5483 cpubenchmark score and 4 GB RAM
- 2 switches: one 8 port gigabit and one 8 port 100 Mbps
- Software: LTSP version 20.04, distribution Ubuntu 20.04.1, flavour MATE, kernel 5.4
- Use case: running educational applications and browsing

So, saying "I have a quad core CPU" isn't appropriate feedback, as it may have 1000 or 10000 score. It's best to mention the exact CPU, its cpubenchmark.net score, and how much RAM it has.

4.10 Snap

There's some controversy regarding snap, Ubuntu's new packaging format. LTSP does its best to support it, but occasionally some issues arise with it that can't be solved from the LTSP side, for example wasting RAM in live sessions or not working with NFS home.

If you decide to remove snap, you may open a terminal, type `sudo -i` to become root, and then copy/paste all the following code:

```
re() {
    if ! "$@"; then
        echo "Command failed: $*" >&2
        exit 1
    fi
}

remove_snap() {
    local packages

    if [ -x /usr/bin/mate-session ]; then
        packages=$(dpkg-query -W -f='${Package} ${Version}\n' arctica-greeter-guest-session ayatana-indicator-application evolution-common indicator-
application mate-hud 2>/dev/null | awk '$2 { print $1 }') || true
        if [ -n "$packages" ]; then
            echo "Removing some MATE cruft: $packages"
            re apt-get purge --yes --auto-remove $packages
        fi
    fi
    if [ -x /usr/bin/snap ]; then
        if [ -f /var/lib/snapd/desktop/applications/firefox_firefox.desktop ] &&
            [ ! -L /var/lib/snapd/desktop/applications/firefox_firefox.desktop ]; then
            echo "Replacing Firefox snap with deb using MozillaTeam PPA"
            # Remove firefox before snapd to work around LP: #1998710
            snap remove firefox 2>/dev/null || true
            re add-apt-repository --yes ppa:mozillateam/ppa
            echo 'Package: *
Pin: release o=LP-PPA-mozillateam
Pin-Priority: 1001' >/etc/apt/preferences.d/60mozillateam-ppa
            # If you need more locales e.g. firefox-locale-el add them in this line
            re apt-get install --yes firefox firefox-locale-en
            if [ -f /usr/share/mate/applications/firefox.desktop ]; then
                re dpkg-divert --package sch-scripts --divert \
                    /usr/share/mate/applications/firefox-desktop.diverted \
                    --rename /usr/share/mate/applications/firefox.desktop
            fi
        fi
        # Remove snapd, THEN provide a symlink to deb firefox for panels etc
        re apt-get purge --yes --auto-remove snapd
        if [ ! -e /var/lib/snapd/desktop/applications/firefox_firefox.desktop ]; then
            re mkdir -p /var/lib/snapd/desktop/applications
            re ln -s /usr/share/applications/firefox.desktop /var/lib/snapd/desktop/applications/firefox_firefox.desktop
        fi
    fi
}

remove_snap
```

4.11 Versioning

Taking into account the Debian policy, LTSP uses the following calver versioning scheme:

- 20.04: a typical upstream release in April 2020
- 20.04.2: additional releases within the same month add a counter
- 20.04.10: rarely, maintenance *upstream* releases for older LTSP versions may be needed when grave bugs are backported or security issues are discovered; since these will be kept in separate repositories, they can just bump that month's counter, or use "+"

Distribution package versioning should be separated with a hyphen "-" or tilde "~":

- 20.04-3: the third Debian packaging of the 20.04 release
- 20.04-3ubuntu1: an Ubuntu-patched version of the aforementioned Debian package
- 20.04-1~ubuntu20.04.1: an LTSP stable PPA build; debian/changelog contains the released version
- 20.04-1+ubuntu20.04.1: an LTSP proposed PPA build; debian/changelog contains the older version

Note: the YYYY.MM scheme was also considered, but it's longer, and YY.MM can easily be upgraded to YYYY.MM or to YYMM, but not the opposite.

5. Support

For LTSP support, the following methods are available:

5.1 Upstream

- PDF version of this site: for offline reading
- Github page: code repositories, pull requests etc
- Bug tracker: reporting bugs in the LTSP code

5.2 Community

- Chat room: real time discussions; developers also hang out there
- Discussions: asynchronous discussions; developers also participate there
- Wiki: guides maintained by the community
- Old LTSP5 wiki/resources: archived; not supported anymore

5.3 Commercial

- Alkis Georgopoulos: the lead LTSP developer is available for professional support, to install, configure, troubleshoot, maintain or extend LTSP.
- If you're offering commercial LTSP support and want to add your link here, contact the LTSP team.

5.4 Bounties

You may offer a bounty for implementing a new feature that you'd like to see in LTSP. Just file an issue and mention that you're interested in sponsoring its implementation.

5.5 Donations

LTSP is accepting donations via paypal. You may optionally include your nickname to get free LTSP support in the chat room.

Thank you for supporting LTSP development!