Funtoo Linux Installation Guide

(Reindirizzamento da **Funtoo Linux Installation**)

Nota:

Translators: Thank you for your help! Please use the section-by-section guide as the basis for your translations. Thanks again!

Introduction

```
# Welcome to Funtoo Linux!
```

Welcome to Funtoo Linux! This document was written to help you install Funtoo Linux on PC-compatible systems, while keeping distracting options regarding system configuration to a minimum.

If you've had previous experience installing Gentoo Linux then a lot of steps will be familiar, but you should still read through as there are a few differences. If you're new to installing a Gentoo-based Linux, or new to Linux entirely -- welcome! We have attempted to make these installation instructions understandable to new users as well.

Tip

We now offer direct deployment of Funtoo Linux in Amazon Web Services. This is a useful option for those who wish to take advantage of AWS or deploy Funtoo Linux automatically. A tutorial-style guide on how to use AWS with Funtoo is available.

Nota:

To familiarize yourself with the latest changes in Funtoo Linux, Release Notes for Funtoo Linux 1.3 are available.

Tip

We now offer the ability to read and browse the Install Guide section-by-section. Online users may find this more convenient.

Installation Overview

This is a basic overview of the Funtoo installation process:

1. Download and boot the live CD of your choice.
2. Prepare your disk.
3. MBR Partitioning.
4. GPT Partitioning.
5. Create and mount filesystems.
6. Setting the Date.
7. Install the Funtoo stage tarball of your choice.
8. Chroot into your new system.
9. Download the Portage tree.
10. Configure your system.

11. Introducing Portage.
12. Install a kernel.
13. Install a bootloader.
14. Configure the Network.
15. Complete final steps.
16. Profile Configuration.
17. All Done! Enjoy!

Download LiveCD

In order to install Funtoo Linux, you will first need to boot your computer using a Linux-based Live CD or USB stick. We recommend the Gentoo-based System Rescue CD as it contains lots of tools and utilities and supports both 32-bit and 64-bit systems. It can be burned to CD/DVD or installed on a USB stick. Download it here:

- Download from funtoo.org

Important

NO VIDEO: We have patched our download of System Rescue CD so that it should initialize video properly when booting from UEFI (See FL-2030.) If you are using the official, non-Funtoo System Rescue CD, at the GRUB menu, you may need to press `e` to edit the menu entry and add a GRUB boot line that reads `insmod all_video` and then boot. This bug has been reported upstream to System Rescue CD developers.

Nota:

If using an older version of System Rescue CD, be sure to select the `rescue64` kernel at the boot menu if you are installing a 64-bit system. By default, System Rescue CD used to boot in 32-bit mode though the latest version attempts to automatically detect 64-bit processors.

Network Access

Once you have booted System Rescue CD, see if you have Internet access. Internet access is required for installing Funtoo Linux:

```
# ping www.google.com

PING www.google.com (216.58.217.36) 56(84) bytes of data.

64 bytes from den03s10-in-f4.1e100.net (216.58.217.36): icmp_seq=1 ttl=57 time=30.1 ms
```

If the ping is successful (you see `64 bytes` messages as above,) then your Network is set up. Hit Control-C to stop the ping.

If you need to set up a WiFi connection for Internet access, then this can be accomplished using the `nmtui` command-line tool:

```
# nmtui
```

Remote Install

Alternatively, you can log into System Rescue CD over the network via SSH to perform the install from another computer, and this may be more convenient way to install Funtoo Linux.

If you'd like to complete the install remotely, here's how. First, you will need to ensure that System Rescue CD has a functioning network connection. Then, you will need to set a root password for System Rescue CD:

```
# passwd

New password: ********

Retype new password: ********

passwd: password updated successfully
```

Once you have typed in a password, you will now need to determine the IP address of System Rescue CD, and then you can use `ssh` to connect to it. To determine the IP address currently being used by System Rescue CD, type `ifconfig`:

```
# ifconfig
```

Alternatively, determining of an IP address is possible with iproute2 `ip` tool:

```
# ip addr show
```

One of the interfaces should have an IP address (listed as `inet addr:`) from your LAN. You can then connect remotely, from another system on your LAN, to System Rescue CD, and perform steps from the comfort of an existing OS. On your remote system, type the following, replacing `1.2.3.4` with the IP address of System Rescue CD. Connecting from an existing Linux or MacOS system would look something like this:

```
(remote system) $ ssh root@1.2.3.4

Password: **********
```

Nota:

If you'd like to connect remotely from an existing Microsoft Windows system, you'll need to download an SSH client for Windows, such as PuTTY.

After you've logged in via SSH, you're now connected remotely to System Rescue CD and can perform the installation steps.

Prepare Disk

In this section, you will need to choose a disk format to use for booting and partitioning -- either MBR or UEFI/GPT. If you are not familiar with the differences between these options, please review our Disk Formats page for an overview of each option and the trade-offs.

Generally, it's usually safe to pick the legacy MBR method for system disks under 2TB in size and most modern PC systems support MBR as well as UEFI booting.

But First...

Before doing anything to your disks, make sure you are partitioning the right one. Use the `lsblk` command to view a list of all block devices on your system, as well as partitions on these block devices:

```
# lsblk

NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
```

```
sda             8:0    0  1.8T  0 disk
├sda1            8:1    0  512M  0 part
├sda2            8:2    0    8G  0 part [SWAP]
└sda3            8:3    0  1.8T  0 part
  ├main-root 254:0    0  500G  0 lvm  /
  └main-data 254:1    0  1.3T  0 lvm  /home
```

Make sure you will not be overwriting any important data and that you have chosen the correct `/dev/sd?` device. Above, you can see that `sda` contains three partitions, `sda1`, `sda2` and `sda3`, and that `sda3` contains LVM volumes.

Once you've double-checked your target block device and made sure you'll be partitioning the correct disk, proceed to the next step.

MBR Partitioning

Legacy (BIOS/MBR) Method

Nota:

Use this method if you are booting using your BIOS, and if your System Rescue CD initial boot menu was light blue. If you're going to use the UEFI/GPT disk format, then please proceed to the next section.

First, it's a good idea to make sure that you've found the correct hard disk to partition. Try this command and verify that `/dev/sda` is the disk that you want to partition:

```
# fdisk -l /dev/sda



Disk /dev/sda: 640.1 GB, 640135028736 bytes, 1250263728 sectors

Units = sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk label type: gpt



#         Start          End    Size  Type            Name

 1         2048   1250263694  596.2G  Linux filesyste Linux filesystem
```

Now, it is recommended that you erase any existing MBR or GPT partition tables on the disk, which could confuse the system's BIOS at boot time. We accomplish this using `sgdisk`:

Warning

This will make any existing partitions inaccessible! You are **strongly** cautioned and advised to backup any critical data before proceeding.

```
# sgdisk --zap-all /dev/sda



Creating new GPT entries.

GPT data structures destroyed! You may now partition the disk using fdisk or

other utilities.
```

This output is also nothing to worry about, as the command still succeeded:

```
****************************************************************

Found invalid GPT and valid MBR; converting MBR to GPT format

in memory.

****************************************************************
```

Now we will use `fdisk` to create the MBR partition table and partitions:

```
# fdisk /dev/sda
```

Within `fdisk`, follow these steps:

Empty the partition table:

```
Command (m for help): o ↵
```

Create Partition 1 (boot):

```
Command (m for help): n ↵

Partition type (default p): ↵

Partition number (1-4, default 1): ↵

First sector: ↵

Last sector: +128M ↵
```

Create Partition 2 (swap):

```
Command (m for help): n ↵

Partition type (default p): ↵

Partition number (2-4, default 2): ↵

First sector: ↵
```

```
Last sector: +2G ↵

Command (m for help): t ↵

Partition number (1,2, default 2): ↵

Hex code (type L to list all codes): 82 ↵
```

Create the root partition:

```
Command (m for help): n ↵

Partition type (default p): ↵

Partition number (3,4, default 3): ↵

First sector: ↵

Last sector: ↵
```

Verify the partition table:

```
Command (m for help): p



Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors

Units: sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x82abc9a6



Device     Boot      Start       End     Blocks  Id System

/dev/sda1            2048    264191    131072  83 Linux

/dev/sda2          264192   4458495   2097152  82 Linux swap / Solaris

/dev/sda3         4458496 625142447 310341976  83 Linux
```

Write the partition table to disk:

```
Command (m for help): w
```

Your new MBR partition table will now be written to your system disk.

Nota:

You're done with partitioning! Now, jump over to Creating filesystems.

GPT Partitioning

UEFI/GPT Method

Nota:

Use this method if you are interested in booting using UEFI, and if your System Rescue CD initial boot menu was black and white. If it was light blue, this method will not work. Instead, use the instructions in the previous section, or reboot SystemRescueCD in UEFI mode first.

The `gdisk` commands to create a GPT partition table are as follows. Adapt sizes as necessary, although these defaults will work for most users. Start `gdisk`:

```
# gdisk /dev/sda
```

Within `gdisk`, follow these steps:

Create a new empty partition table (This *will* erase all data on the disk when saved):

```
Command: o ↵

This option deletes all partitions and creates a new protective MBR.

Proceed? (Y/N): y ↵
```

Create Partition 1 (boot):

```
Command: n ↵

Partition Number: 1 ↵

First sector: ↵

Last sector: +500M ↵

Hex Code: EF00 ↵
```

Create Partition 2 (swap):

```
Command: n ↵

Partition Number: 2 ↵

First sector: ↵

Last sector: +4G ↵

Hex Code: 8200 ↵
```

Create Partition 3 (root):

```
Command: n ↵

Partition Number: 3 ↵

First sector: ↵
```

```
Last sector: ↵ (for rest of disk)

Hex Code: ↵
```

Along the way, you can type " `p` " and hit Enter to view your current partition table. If you make a mistake, you can type " `d` " to delete an existing partition that you created. When you are satisfied with your partition setup, type " `w` " to write your configuration to disk:

Write Partition Table To Disk:

```
Command: w ↵

Do you want to proceed? (Y/N): Y ↵
```

The partition table will now be written to the disk and `gdisk` will close.

Now, your GPT/GUID partitions have been created, and will show up as the following *block devices* under Linux:

- `/dev/sda1`, which will be used to hold the `/boot` filesystem,
- `/dev/sda2`, which will be used for swap space, and
- `/dev/sda3`, which will hold your root filesystem.

Tip

You can verify that the block devices above were correctly created by running the command `lsblk`.

Creating Filesystems

Nota:

This section covers both BIOS *and* UEFI installs. Don't skip it!

Before your newly-created partitions can be used, the block devices that were created in the previous step need to be initialized with filesystem *metadata*. This process is known as *creating a filesystem* on the block devices. After filesystems are created on the block devices, they can be mounted and used to store files.

Let's keep this simple. Are you using legacy MBR partitions? If so, let's create an ext2 filesystem on `/dev/sda1`:

```
# mkfs.ext2 /dev/sda1
```

If you're using GPT partitions for UEFI, you'll want to create a vfat filesystem on `/dev/sda1`, because this is what UEFI is able to read:

```
# mkfs.vfat -F 32 /dev/sda1
```

Now, let's create a swap partition. This partition will be used as disk-based virtual memory for your Funtoo Linux system.

You will not create a filesystem on your swap partition, since it is not used to store files. But it is necessary to initialize it using the `mkswap` command. Then we'll run the `swapon` command to make your newly-initialized swap space immediately active within the live CD environment, in case it is needed during the rest of the install process:

```
# mkswap /dev/sda2
```

```
# swapon /dev/sda2
```

Root Filesystem

Now, we need to create a root filesystem. This is where Funtoo Linux will live. We generally recommend ext4 or XFS root filesystems. Keep in mind that some filesystems will require additional filesystem tools to be `emerge`d prior to rebooting. Please consult the following table for more information:

Filesystem	Recommended as root file system?	Additional tools required to emerge
ext4	Yes	None
XFS	Yes	`sys-fs/xfsprogs`
zfs	No - advanced users only	`sys-fs/zfs`
btrfs	No - advanced users only	`sys-fs/btrfs-progs`

Important

We do not recommend users set up ZFS or BTRFS as their root filesystem. This is much more complex and usually not necessary. Instead, choose XFS or ext4. *We do support ZFS or BTRFS as non-root filesystems* and this is much, much easier to configure.

See ZFS and BTRFS after you are done setting up your Funtoo Linux system to configure ZFS or BTRFS for additional secondary storage.

If you're not sure, choose ext4. Here's how to create a root ext4 filesystem:

```
# mkfs.ext4 /dev/sda3
```

...and here's how to create an XFS root filesystem, if you prefer to use XFS instead of ext4:

```
# mkfs.xfs /dev/sda3
```

Your filesystems (and swap) have all now been initialized, so that that can be mounted (attached to your existing directory heirarchy) and used to store files. We are ready to begin installing Funtoo Linux on these brand-new filesystems.

Mounting Filesystems

Mount the newly-created filesystems as follows, creating `/mnt/funtoo` as the installation mount point:

```
# mkdir /mnt/funtoo

# mount /dev/sda3 /mnt/funtoo

# mkdir /mnt/funtoo/boot

# mount /dev/sda1 /mnt/funtoo/boot
```

Optionally, if you have a separate filesystem for `/home` or anything else, you can choose to create these filesystems now. Note that if you are wanting to use ZFS or BTRFS to create general storage for files that do not hold operating system files, then this is best to do later after Funtoo Linux is installed.

```
# mkdir /mnt/funtoo/home
```

```
# mount /dev/sda4 /mnt/funtoo/home
```

If you have `/tmp` or `/var/tmp` on a separate filesystem, be sure to change the permissions of the mount point to be globally-writeable after mounting, as follows:

```
# chmod 1777 /mnt/funtoo/tmp
```

Setting the Date

Important

If your system's date and time are too far off (typically by months or years,) then it may prevent Portage from properly downloading source tarballs. This is because some of our sources are downloaded via HTTPS, which use SSL certificates and are marked with an activation and expiration date. However, if your system time is relatively close to correct, you can probably skip this step for now.

Now is a good time to verify the date and time are correctly set to UTC. Use the `date` command to verify the date and time:

```
# date

Fri Jul 15 19:47:18 UTC 2011
```

If the date and/or time need to be corrected, do so using `date MMDDhhmmYYYY`, keeping in mind `hhmm` are in 24-hour format. The example below changes the date and time to "July 16th, 2011 @ 8:00PM" UTC:

```
# date 071620002011
```

```
Fri Jul 16 20:00:00 UTC 2011
```

Once you have set the system clock, it's a very good idea to copy the time to the hardware clock, so it persists across reboots:

```
# hwclock --systohc
```

Download and Extract Stage3

Now that filesystems are created and your hardware and system clock are set, the next step is downloading the initial Stage 3 tarball. The Stage 3 is a pre-compiled system used as a starting point to install Funtoo Linux.

To download the correct build of Funtoo Linux for your system, head over to the Subarches page. Subarches are builds of Funtoo Linux that are designed to run on a particular type of CPU, to offer the best possible performance. They also take advantage of the instruction sets available for each CPU.

Nota:

> We now have a database of all Intel 64-bit CPUs in existence, mapped to the ideal stage3 to download! To view this list, visit Funtoo CPU Database and search for your CPU. A subset of this list now also appears on 64-bit Intel subarch pages.

If you are using a virtualization technology to run Funtoo Linux, and your VM may migrate to different types of hardware, then it's recommended that you use a stage3 that is optimized for the oldest CPU instruction set that your VM will run on, or a generic image if it may run on both AMD and Intel processors.

For most subarches, you will have several stage3s available to choose from. This next section will help you understand which one to pick.

Which Build?

Pick `1.3-release-std`. This is release 1.3 of Funtoo Linux, our current release.

Download the Stage3

Once you have found the stage3 that you would like to download, use `wget` to download the Stage 3 tarball you have chosen to use as the basis for your new Funtoo Linux system. It should be saved to the `/mnt/funtoo` directory as follows:

```
# cd /mnt/funtoo

# wget https://build.funtoo.org/1.3-release-std/x86-64bit/generic_64/stage3-latest.tar.xz
```

Note that 64-bit systems can run 32-bit or 64-bit stages, but 32-bit systems can only run 32-bit stages. Make sure that you select a Stage 3 build that is appropriate for your CPU. If you are not certain, it is a safe bet to choose the `generic_64` or `generic_32` stage. Consult the Subarches page for more information.

Once the stage is downloaded, extract the contents with the following command, substituting in the actual name of your Stage 3 tarball:

```
# tar xpf stage3-latest.tar.xz
```

Important

It is very important to use `tar's "p"` option when extracting the Stage 3 tarball - it tells `tar` to *preserve* any permissions and ownership that exists within the archive. Without this option, your Funtoo Linux filesystem permissions will be incorrect.

Chroot into Funtoo

To install Funtoo Linux, the `chroot` command is first used. The chroot command will "switch into" the new Funtoo Linux system, so the commands you execute after running "chroot" will run within your newly-extracted Funtoo Linux system.

Before chrooting, there are a few things that need to be done to set up the chroot environment. You will need to mount `/proc`, `/sys` and `/dev` inside your new system. Use the following commands to do so:

```
# cd /mnt/funtoo

# mount -t proc none proc

# mount --rbind /sys sys

# mount --rbind /dev dev
```

You'll also want to copy over `resolv.conf` in order to have proper resolution of Internet hostnames from inside the chroot:

```
# cp /etc/resolv.conf /mnt/funtoo/etc/
```

Now you can chroot into your new system. Use `env` before `chroot` to ensure that no environment settings from the installation media are pulled in to your new system:

```
# env -i HOME=/root TERM=$TERM /bin/chroot . bash -l
```

Nota:

For users of live CDs with 64-bit kernels installing 32-bit systems: Some software may use `uname -r` to check whether the system is 32 or 64-bit. You may want to append linux32 to the chroot command as a workaround, but it's generally not needed.

Important

If you receive the error "`chroot: failed to run command `/bin/bash': Exec format error`", it is most likely because you are running a 32-bit kernel and trying to execute 64-bit code. Make sure that you have selected the proper type of kernel when booting SystemRescueCD.

It's also a good idea to change the default command prompt while inside the chroot. This will avoid confusion if you have to change terminals. Use this command:

```
# export PS1="(chroot) $PS1"
```

Test internet name resolution from within the chroot:

```
# ping -c 5 google.com
```

If you can't ping, make sure `/etc/resolv.conf` doesn't contain things like `127.0.x.x` addresses, if it does, change the `127.0.x.x` entry to `8.8.8.8` -- Google's public dns address. Make sure to replace this with your dns of choice once the system is installed.

Congratulations! You are now chrooted inside a Funtoo Linux system. Now it's time to get Funtoo Linux properly configured so that Funtoo Linux will start successfully, without any manual assistance, when your system is restarted.

Download Portage Tree

Now it's time to install the Portage repository, which contains package scripts (ebuilds) that tell portage how to build and install thousands of different software packages. To create the Portage repository, simply run `ego sync` from within the chroot. This will automatically clone the portage tree from GitHub and all kits:

```
(chroot) # ego sync
```

Configuration Files

As is expected from a Linux distribution, Funtoo Linux has its share of configuration files. The one file you are absolutely required to edit in order to ensure that Funtoo Linux boots successfully is `/etc/fstab`. The others are optional.

Using Nano

The default editor included in the chroot environment is called `nano`. To edit one of the files below, run nano as follows:

```
(chroot) # nano -w /etc/fstab
```

When in the editor, you can use arrow keys to move the cursor, and common keys like backspace and delete will work as expected. To save the file, press Control-X, and answer `y` when prompted to save the modified buffer if you would like to save your changes.

Configuration Files

Here are a full list of files that you may want to edit, depending on your needs:

File	Do I need to change it?	Description
`/etc/fstab`	**YES - required**	Mount points for all filesystems to be used at boot time. This file must reflect your disk partition setup. We'll guide you through modifying this file below.
`/etc/localtime`	*Maybe - recommended*	Your timezone, which will default to UTC if not set. This should be a symbolic link to something located under /usr/share/zoneinfo (e.g. /usr/share/zoneinfo/America/Montreal)
`/etc/portage/make.conf`	*Maybe - recommended*	Parameters used by gcc (compiler), portage, and make. *Note that it is normal for this file to be empty in Funtoo Linux, as many settings have been migrated to our enhanced profile system.*
`/etc/conf.d/hostname`	*Maybe - recommended*	Used to set system hostname. Set the `hostname` variable to the fully-qualified (with dots, ie. `foo.funtoo.org`) name if you have one. Otherwise, set to the local system hostname (without dots, ie. `foo`). Defaults to `localhost` if not set.
`/etc/hosts`	*No*	You no longer need to manually set the hostname in this file. This file is automatically generated by `/etc/init.d/hostname`.
`/etc/conf.d/keymaps`	Optional	Keyboard mapping configuration file (for console pseudo-terminals). Set if you have a non-US keyboard. See Funtoo Linux Localization.
`/etc/conf.d/hwclock`	Optional	How the time of the battery-backed hardware clock of the system is interpreted (UTC or local time). Linux uses the battery-backed hardware clock to initialize the system clock when the system is booted.
`/etc/conf.d/modules`	Optional	Kernel modules to load automatically at system startup. Typically not required. See Additional Kernel Resources for more info.
`/etc/conf.d/consolefont`	Optional	Allows you to specify the default console font. To apply this font, enable the consolefont service by running rc-update add consolefont.

`profiles`	Optional	Some useful portage settings that may help speed up intial configuration.

If you're installing an English version of Funtoo Linux, you're in luck, as most of the configuration files can be used as-is. If you're installing for another locale, don't worry. We will walk you through the necessary configuration steps on the Funtoo Linux Localization page, and if needed, there's always plenty of friendly, helpful support available. (See Getting Help)

Let's go ahead and see what we have to do. Use `nano -w <name_of_file>` to edit files -- the " `-w` " argument disables word-wrapping, which is handy when editing configuration files. You can copy and paste from the examples.

Warning

It's important to edit your `/etc/fstab` file before you reboot! You will need to modify both the "fs" and "type" columns to match the settings for your partitions and filesystems that you created with `gdisk` or `fdisk`. Skipping this step may prevent Funtoo Linux from booting successfully.

/etc/fstab

`/etc/fstab` is used by the `mount` command which is run when your system boots. Lines in this file inform `mount` about filesystems to be mounted and how they should be mounted. In order for the system to boot properly, you must edit `/etc/fstab` and ensure that it reflects the partition configuration you used earlier in the install process. If you can't remember the partition configuration that you used earlier, the `lsblk` command may be of help to you:

```
(chroot) # nano -w /etc/fstab
```

`/etc/fstab` - An example fstab file

```
# The root filesystem should have a pass number of either 0 or 1.
# All other filesystems should have a pass number of 0 or greater than 1.
#
# NOTE: If your BOOT partition is ReiserFS, add the notail option to opts.
#
# See the manpage fstab(5) for more information.
#
# <fs>       <mountpoint>  <type>  <opts>          <dump/pass>

/dev/sda1    /boot         ext2    noauto,noatime 1 2
/dev/sda2    none          swap    sw             0 0
/dev/sda3    /             ext4    noatime        0 1
#/dev/cdrom  /mnt/cdrom    auto    noauto,ro      0 0
```

Nota:

If you're using UEFI to boot, change the `/dev/sda1` line so that it says `vfat` instead of `ext2`. Similarly, make sure that the `/dev/sda3` line specifies either `xfs` or `ext4`, depending on which filesystem you chose earlier on in the installation process when you created filesystems.

/etc/localtime

`/etc/localtime` is used to specify the timezone that your machine is in, and defaults to UTC. If you would like your Funtoo Linux system to use local time, you should replace `/etc/localtime` with a symbolic link to the timezone that you wish to use.

```
(chroot) # ln -sf /usr/share/zoneinfo/MST7MDT /etc/localtime
```

The above sets the timezone to Mountain Standard Time (with daylight savings). Type `ls /usr/share/zoneinfo` to list available timezones. There are also sub-directories containing timezones described by location.

/etc/portage/make.conf

`USE` flags define what functionality is enabled when packages are built. It is not recommended to add a lot of USE flags during installation; you should wait until you have a working, bootable system before changing your USE flags. A USE flag prefixed with a minus (" `-` ") sign tells Portage not to use the flag when compiling. A Funtoo guide to USE flags will be available in the future. For now, you can find out more information about USE flags in the Gentoo Handbook.

/etc/conf.d/hwclock

If you dual-boot with Windows, you'll need to edit this file and change the value of **clock** from **UTC** to **local**, because Windows will set your hardware clock to local time every time you boot Windows. Otherwise you normally wouldn't need to edit this file.

```
(chroot) # nano -w /etc/conf.d/hwclock
```

Localization

By default, Funtoo Linux is configured with Unicode (UTF-8) enabled, and for the US English locale and keyboard. If you would like to configure your system to use a non-English locale or keyboard, see Funtoo Linux Localization.

Introducing Portage

Portage, the Funtoo Linux package manager has a command called `emerge` which is used to build and install packages from source. It also takes care of installing all of the package's dependencies. You call emerge like this:

```
(chroot) # emerge packagename
```

When you install a package by specifying its name in the command-line, Portage records its name in the `/var/lib/portage/world` file. It does so because it assumes that, since you have installed it by name, you want to consider it part of your system and want to keep the package updated in the future. This is a handy feature, since when packages are being added to the `world` set, we can update our entire system by typing:

```
(chroot) # ego sync

(chroot) # emerge -auDN @world
```

This is the "official" way to update your Funtoo Linux system. Above, we first update our Portage tree using git to grab the latest ebuilds (scripts), and then run an emerge command to update the `world` set of packages. The options specified tell `emerge` to:

- **a** - show us what will be emerged, and **ask** us if we want to proceed

- **u** - **update** the packages we specify -- don't emerge them again if they are already emerged.
- **D** - Consider the entire dependency tree of packages when looking for updates. In other words, do a **deep** update.
- **N** - Update any packages that have changed (**new**) USE settings.

You should also consider passing `--with-bdeps=y` when emerging @world, at least once in a while. This will update build dependencies as well.

Of course, sometimes we want to install a package but not add it to the `world` file. This is often done because you only want the package installed temporarily or because you know the package in question is a dependency of another package. If this behavior is desired, you call emerge like this:

```
(chroot) # emerge -1 packagename
```

Advanced users may be interested in the Emerge wiki page.

Updating World

Certain packages in the Funtoo stage3 tarball are compiled with the bindist USE flag enabled by default. (The bindist flag controls enabling or disabling of options for proprietary and/or patented parts of code which is not allowed to be distributed in images due to licensing issues). You may notice a dependency resolution problem with bindist USE flags during updating packages after initial system setup. To avoid potential problems, **update the system before first boot or any other package installation** as shown below:

```
(chroot) # ego sync
```

```
(chroot) # emerge -auDN @world
```

Important

Make sure you read any post emerge messages and follow their instructions. This is especially true if you have upgraded perl or python.

Important

If you choose different file systems then the ones in this guide, like JFS, XFS, ZFS or Btrfs, make sure the kernel has the tools to check them. For JFS the package is jsfutils, similar packages exist for all file systems.

Prepare Disk

Funtoo Linux stage3's include a pre-built `debian-sources-lts` kernel to make installation faster and easier. To see what kernel version is pre-installed, type:

```
(chroot) # emerge -s debian-sources-lts

Searching...

[ Results for search key : debian-sources-lts ]

[ Applications found : 1 ]
```

```
*  sys-kernel/debian-sources

      Latest version available: 4.9.130

      Latest version installed: 4.9.130

      Size of files: 95,022 kB

      Homepage:      https://packages.debian.org/stable/kernel/linux-image-4.9.0-6-amd64

      Description:   Debian Sources (and optional binary kernel)

      License:       GPL-2
```

Important

At this point it is wise to emerge the latest `sys-kernel/linux-firmware` package, because various drivers rely on firmware blobs and instructions. Hardware like Wi-Fi cards, graphic cards, network cards, and others will not work properly or at all if firmware not present.

Nota:

NVIDIA card users: the `binary` USE flag installs the Nouveau drivers which cannot be loaded at the same time as the proprietary drivers, and cannot be unloaded at runtime because of KMS. You need to blacklist it under `/etc/modprobe.d/`. If you're planning to use NVIDIA's drivers, see the NVIDIA Linux Display Drivers page for blacklisting information.

Bootloader

These install instructions show you how to use GRUB to boot using BIOS (legacy) or UEFI.

ego boot update (`ego boot`) is installed by default, but GRUB is not, as it is not required for all Funtoo Linux systems (such as containers, for example.) But for booting on bare metal, it is the recommended and best-supported boot loader, so you will need to emerge it:

```
(chroot) # emerge -av grub
```

Next, edit `/etc/boot.conf` using `nano` and specify " `Funtoo Linux genkernel` " as the `default` setting at the top of the file, replacing `"Funtoo Linux"` . Also, if you're not using memtest86+ remove the entry in boot.conf to avoid errors.

`/etc/boot.conf` should now look like this:

`/etc/boot.conf`

```
boot {
        generate grub
        default "Funtoo Linux"
        timeout 3
}

"Funtoo Linux" {
        kernel bzImage[-v]
}
```

```
"Funtoo Linux genkernel" {
        kernel kernel[-v]
        initrd initramfs[-v]
        params += real_root=auto rootfstype=auto
}
```

If you are booting a custom or non-default kernel, please read `man boot.conf` for information on the various options available to you.

Old School (BIOS) MBR

When using "old school" BIOS booting, run the following command to install GRUB to your MBR, and generate the `/boot/grub/grub.cfg` configuration file that GRUB will use for booting:

```
(chroot) # grub-install --target=i386-pc --no-floppy /dev/sda

(chroot) # ego boot update
```

New School (UEFI) Boot Entry

If you're using "new school" UEFI booting, run of the following sets of commands, depending on whether you are installing a 64-bit or 32-bit system. This will add GRUB as a UEFI boot entry.

For x86-64bit systems:

```
(chroot) # mount -o remount,rw /sys/firmware/efi/efivars

(chroot) # grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id="Funtoo Linux [GRUB]" --recheck /dev/sda
```

```
(chroot) # ego boot update
```

For x86-32bit systems:

```
(chroot) # mount -o remount,rw /sys/firmware/efi/efivars

(chroot) # grub-install --target=i386-efi --efi-directory=/boot --bootloader-id="Funtoo Linux [GRUB]" --recheck /dev/sda

(chroot) # ego boot update
```

First Boot, and in the future...

OK -- you are almost ready to boot!

You only need to run `grub-install` when you first install Funtoo Linux, but you need to re-run `ego boot update` every time you modify your `/etc/boot.conf` file or add new kernels to your system. This will regenerate `/boot/grub/grub.cfg` so that you will have new kernels available in your GRUB boot menu, the next time you reboot.

Network

It's important to ensure that you will be able to connect to your local-area network after you reboot into Funtoo Linux. There are three approaches you can use for configuring your network: NetworkManager, dhcpcd, and the Funtoo Linux Networking scripts. Here's how to choose which one to use based on the type of network you want to set up.

Wi-Fi

For laptop/mobile systems where you will be using Wi-Fi, roaming, and connecting to various networks, NetworkManager is strongly recommended. Since Wi-Fi cards require firmware to operate, it is also recommended that you emerge the linux-firmware ebuild if you have not done so already:

```
(chroot) # emerge linux-firmware networkmanager

(chroot) # rc-update add NetworkManager default
```

The above command will ensure that NetworkManager starts after you boot into Funtoo Linux. Once you've completed these installation steps and have booted into Funtoo Linux, you can use the `nmtui` command (which has an easy-to-use console-based interface) to configure NetworkManager so that it will connect (and automatically reconnect, after reboot) to a Wi-Fi access point:

```
# nmtui
```

For more information about NetworkManager, see the NetworkManager package page.

Desktop (Wired DHCP)

For a home desktop or workstation with wired Ethernet that will use DHCP, the simplest and most effective option to enable network connectivity is to simply add `dhcpcd` to the default runlevel:

```
(chroot) # rc-update add dhcpcd default
```

When you reboot, `dhcpcd` will run in the background and manage all network interfaces and use DHCP to acquire network addresses from a DHCP server.

If your upstream DHCP server is dnsmasq, it can be configured to assign addresses via mac address to make servers on DHCP feasible.

Server (Static IP)

For servers, the Funtoo Linux Networking scripts are recommended. They are optimized for static configurations and things like virtual ethernet bridging for virtualization setups. See Funtoo Linux Networking for information on how to use Funtoo Linux's template-based network configuration system.

Hostname

By default Funtoo uses "localhost" as hostname. Although the system will work perfectly fine using this name, some ebuilds refuse to install when detecting localhost as hostname. It also may create confusion if several systems use the same hostname. Therefore, it is advised to change it to a more meaningful name. The hostname itself is arbitrary, meaning you can choose almost any combination of characters, as long as it makes sense to the system administrator. To change the hostname, edit

```
(chroot) # nano /etc/conf.d/hostname
```

Look for the line starting with hostname and change the entry between the quotes. Save the file, on the next boot Funtoo will use the new hostname.

Warning

Do not use special characters in the hostname, as the shell may interpret these, leading to unpredictable results. Use the Latin alphabet: a-z, A-Z, 0-9

Finishing Up

Set your root password

It's imperative that you set your root password before rebooting so that you can log in.

```
(chroot) # passwd
```

Restart your system

Now is the time to leave chroot, to unmount Funtoo Linux partitions and files and to restart your computer. When you restart, the GRUB boot loader will start, load the Linux kernel and initramfs, and your system will begin booting.

Leave the chroot, change directory to /mnt, unmount your Funtoo partitions, and reboot.

```
(chroot) # exit

# cd /mnt

# umount -lR funtoo

# reboot
```

Nota:

System Rescue CD will gracefully unmount your new Funtoo filesystems as part of its normal shutdown sequence.

You should now see your system reboot, the GRUB boot loader appear for a few seconds, and then see the Linux kernel and initramfs loading. After this, you should see Funtoo Linux itself start to boot, and you should be greeted with a `login:` prompt. Funtoo Linux has been successfully installed!

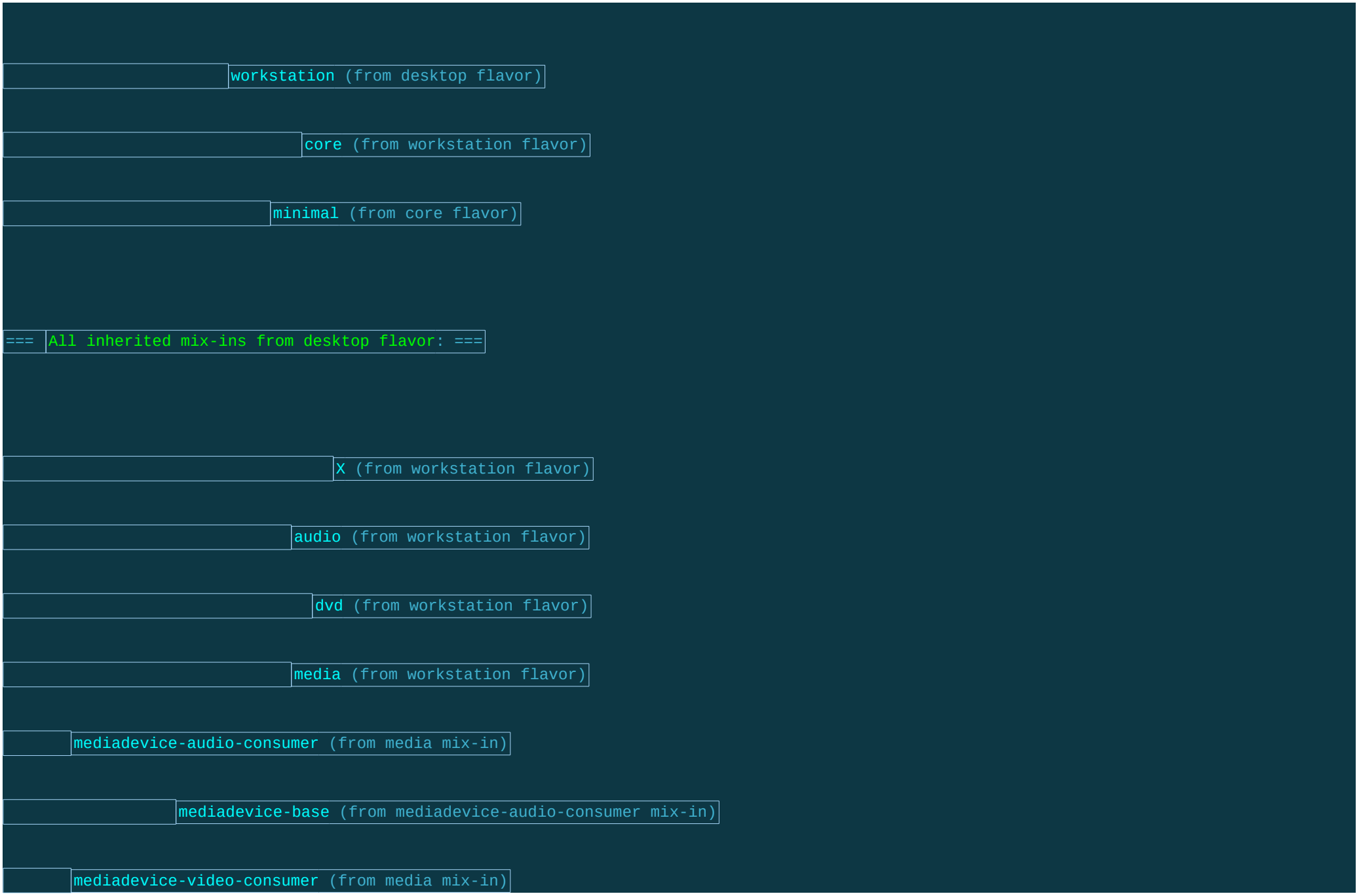
Profiles

Once you have rebooted into Funtoo Linux, you can further customize your system to your needs by using Funtoo Profiles. A quick introduction to profiles is included below -- consult the Funtoo Profiles page for more detailed information. There are five basic profile types: arch, build, subarch, flavors and mix-ins:

Sub-Profile Type	Description
`arch`	Typically `x86-32bit` or `x86-64bit`, this defines the processor type and support of your system. This is defined when your stage was built and should not be changed.
`build`	Defines whether your system is a `current`, `stable` or `experimental` build. At the moment, all Funtoo Linux builds use the `funtoo-current` build profile.
`subarch`	Defines CPU optimizations for your system. The subarch is set at the time the stage3 is built, but can be changed later to better settings if necessary. Be sure to pick a setting that is compatible with your CPU.
`flavor`	Defines the general type of system, such as `server` or `desktop`, and will set default USE flags appropriate for your needs.
`mix-ins`	Defines various optional settings that you may be interested in enabling.

One arch, build and flavor must be set for each Funtoo Linux system, while mix-ins are optional and you can enable more than one if desired. Often, flavors and mix-ins inherit settings from other sub-profiles. Use `epro show` to view your current profile settings, in addition to any inheritance information:

```
(chroot) # epro show


=== Enabled Profiles: ===


        arch:  x86-64bit

       build:  current

     subarch:  intel64-haswell

      flavor:  desktop

     mix-ins:  gnome


=== All inherited flavors from desktop flavor: ===
```


workstation (from desktop flavor)
core (from workstation flavor)
minimal (from core flavor)
=== All inherited mix-ins from desktop flavor: ===
X (from workstation flavor)
audio (from workstation flavor)
dvd (from workstation flavor)
media (from workstation flavor)
mediadevice-audio-consumer (from media mix-in)
mediadevice-base (from mediadevice-audio-consumer mix-in)
mediadevice-video-consumer (from media mix-in)

```
                mediadevice-base (from mediadevice-video-consumer mix-in)

        mediaformat-audio-common (from media mix-in)

           mediaformat-gfx-common (from media mix-in)

        mediaformat-video-common (from media mix-in)

                  console-extras (from workstation flavor)

                              print (from desktop flavor)
```

Here are some basic examples of `epro` usage:

Description	Command
View available profiles. Enabled profiles will be highlighted in cyan. Directly enabled profiles will be in bold and have a `*` appended.	`epro list`
Change the system flavor.	`epro flavor desktop`
Add a mix-in.	`epro mix-in +gnome`

All Done!

If you are brand new to Funtoo Linux and Gentoo Linux, please check out Funtoo Linux First Steps, which will help get you acquainted with your new system. You may also be interested in the following resources:

- ZFS - a quick and easy HOWTO on how to get ZFS set up under Funtoo Linux.
- BTRFS - a simple guide for setting up BTRFS on your new Funtoo Linux system.
- official documentation, which includes all docs that we officially maintain for installation and operation of Funtoo Linux.

We also have a number of pages dedicated to setting up your system. See First Steps for a list of these pages.

If your system did not boot correctly, see Installation Troubleshooting for steps you can take to resolve the problem.

Categorie:

- HOWTO
- Install
- Official Documentation

Menu di navigazione

- Cronologia

Vai

- FAQ
- scarica
- Install
- Funtoo Upgrade Docs
- Funtoo Release Notes
- Report a Bug
- Ebuilds
- Wiki Editing Guidelines
- Development Guide
- Forums
- Server Status
- Sponsors

Categorie

- Articles
- HOWTOs
- Tutorials
- Networking
- Portage

Links

- Ultime modifiche
- MediaWiki Help

Strumenti

- Puntano qui
- Modifiche correlate
- Pagine speciali
- Versione stampabile
- Link permanente
- Informazioni pagina
- Valori pagina
- Questa pagina è stata modificata per l'ultima volta il nov 22, 2018 alle 01:06.