Installing the Gentoo installation files

From Gentoo Wiki
< Handbook:AMD64 (/wiki/Special:MyLanguage/Handbook:AMD64) | Installation (/wiki/Special:MyLanguage/Handbook:AMD64/Installation)
Jump to:navigation Jump to:search

AMD64 Handbook (/wiki/Handbook:AMD64)
Installation (/wiki/Handbook:AMD64/Full/Installation)
About the installation (/wiki/Handbook:AMD64/Installation/About)
Choosing the media (/wiki/Handbook:AMD64/Installation/Media)
Configuring the network (/wiki/Handbook:AMD64/Installation/Networking)
Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks)
The stage file
Installing base system (/wiki/Handbook:AMD64/Installation/Base)
Configuring the kernel (/wiki/Handbook:AMD64/Installation/Kernel)
Configuring the system (/wiki/Handbook:AMD64/Installation/System)
Installing tools (/wiki/Handbook:AMD64/Installation/Tools)
Configuring the bootloader (/wiki/Handbook:AMD64/Installation/Bootloader)
Finalizing (/wiki/Handbook:AMD64/Installation/Finalizing)
Working with Gentoo (/wiki/Handbook:AMD64/Full/Working)
Portage introduction (/wiki/Handbook:AMD64/Working/Portage)
USE flags (/wiki/Handbook:AMD64/Working/USE)
Portage features (/wiki/Handbook:AMD64/Working/Features)
Initscript system (/wiki/Handbook:AMD64/Working/Initscripts)
Environment variables (/wiki/Handbook:AMD64/Working/EnvVar)
Working with Portage (/wiki/Handbook:AMD64/Full/Portage)
Files and directories (/wiki/Handbook:AMD64/Portage/Files)

Choosing a stage file

Tip
On supported architectures, it is recommended for users targeting a desktop (graphical) operating system environment to use a stage file with the term `desktop` within the name. These files include packages such as `sys-devel/llvm` (https://packages.gentoo.org/packages/sys-devel/llvm) and `dev-lang/rust-bin` (https://packages.gentoo.org/packages/dev-lang/rust-bin) and USE flag tuning which will greatly improve install time.

The stage file (/wiki/Stage_file) acts as the seed of a Gentoo install. Stage files are generated with Catalyst (/wiki/Catalyst) by the Release Engineering Team (/wiki/Project:RelEng). Stage files are based on specific profiles (/wiki/Profile_(Portage)), and contain an almost-complete system.

When choosing a stage file, it's important to pick one with profile targets corresponding to the desired system type.

Important
While it's possible to make major profile changes after an installation has been established, switching requires substantial effort and consideration, and is outside the scope of this installation manual. Switching init systems is difficult, but switching from `no-multilib` to `multilib` requires extensive Gentoo and low-level toolchain knowledge.

Tip
Most users should not need to use the 'advanced' tarballs options; they are for atypical or advanced software or hardware configurations.

OpenRC

OpenRC (/wiki/OpenRC) is a dependency-based init system (responsible for starting up system services once the kernel has booted) that maintains compatibility with the system provided init program, normally located in **/sbin/init**. It is Gentoo's native and original init system, but is also deployed by a few other Linux distributions and BSD systems.

OpenRC does not function as a replacement for the **/sbin/init** file by default and is 100% compatible with Gentoo init scripts. This means a solution can be found to run the dozens of daemons in the Gentoo ebuild repository.

systemd

systemd is a modern SysV-style init and rc replacement for Linux systems. It is used as the primary init system by a majority of Linux distributions. systemd is fully supported in Gentoo and works for its intended purpose. If something seems lacking in the Handbook for a systemd install path, review the systemd article (/wiki/Systemd) *before* asking for support.

Multilib (32 and 64-bit)

Note
Not every architecture has a multilib option. Many only run with native code. Multilib is most commonly applied to **amd64**.

The multilib profile uses 64-bit libraries when possible, and only falls back to the 32-bit versions when strictly necessary for compatibility. This is an excellent option for the majority of installations because it provides a great amount of flexibility for customization in the future.

Tip
Using `multilib` targets makes it easier to switch profiles later, compared to `no-multilib`

No-multilib (pure 64-bit)

Warning
Readers who are just starting out with Gentoo should *not* choose a no-multilib tarball unless it is absolutely necessary. Migrating from a `no-multilib` to a `multilib` system requires an extremely well-working knowledge of Gentoo and the lower-level toolchain (it may even cause our Toolchain developers (/wiki/Project:Toolchain) to shudder a little). It is not for the faint of heart and is beyond the scope of this guide.

Selecting a no-multilib tarball to be the base of the system provides a complete 64-bit operating system environment - free of 32-bit software. This effectively renders the ability to switch to `multilib` profiles burdensome, although still technically possible.

Downloading the stage file

Before downloading the *stage file*, the current directory should be set to the location of the mount used for the install:

```
root # cd /mnt/gentoo
```

Setting the date and time

Stage archives are generally obtained using HTTPS which requires relatively accurate system time. Clock skew can prevent downloads from working, and can cause unpredictable errors if the system time is adjusted by any considerable amount after installation.

The current date and time can be verified with **date**:

```
root # date
```

```
Mon Oct  3 13:16:22 PDT 2021
```

If the displayed date/time is more than few minutes off, it should be updated using one of the following methods.

Automatic

Using NTP (/wiki/NTP) to correct clock skew is typically easier and more reliable than manually setting the system clock.

chronyd, part of `net-misc/chrony` (https://packages.gentoo.org/packages/net-misc/chrony) can be used to update the system clock to UTC with:

```
root # chronyd -q
```

Important
Systems without a functioning Real-Time Clock (RTC) must sync the system clock at every system start, and on regular intervals thereafter. This is also beneficial for systems with a RTC, as the battery could fail, and clock skew can accumulate.

Warning
Standard NTP traffic not authenticated, it is important to verify time data obtained from the network.

Manual

When NTP access is unavailable, **`date`** can be used to manually set the system clock.

Note
UTC time is recommended for all Linux systems. Later, a system timezone is defined, which changes the offset when the date is displayed.

The following argument format is used to set the time: `MMDDhhmmYYYY` syntax (**M**onth, **D**ay, **h**our, **m**inute and **Y**ear).

For instance, to set the date to October 3rd, 13:16 in the year 2021, issue:

```
root # date 100313162021
```

Graphical browsers

Those using environments with fully graphical web browsers will have no problem copying a stage file URL from the main website's download section (https://www.gentoo.org/downloads/#other-arches). Simply select the appropriate tab, right click the link to the stage file, then **`Copy Link`** to copy the link to the clipboard, then paste the link to the **`wget`** utility on the command-line to download the stage file:

```
root # wget <PASTED_STAGE_FILE_URL>
```

Command-line browsers

More traditional readers or 'old timer' Gentoo users, working exclusively from command-line may prefer using **`links`** (`www-client/links` (https://packages.gentoo.org/packages/www-client/links)), a non-graphical, menu-driven browser. To download a stage, surf to the Gentoo mirror list like so:

```
root # links https://www.gentoo.org/downloads/mirrors/
```

To use an HTTP proxy with **`links`**, pass on the URL with the `-http-proxy` option:

```
root # links -http-proxy proxy.server.com:8080 https://www.gentoo.org/downloads/mirrors/
```

Next to **`links`** there is also the **`lynx`** (`www-client/lynx` (https://packages.gentoo.org/packages/www-client/lynx)) browser. Like **`links`** it is a non-graphical browser but it is not menu-driven.

```
root # lynx https://www.gentoo.org/downloads/mirrors/
```

If a proxy needs to be defined, export the *http_proxy* and/or *ftp_proxy* variables:

```
root # export http_proxy="http://proxy.server.com:port"
root # export ftp_proxy="http://proxy.server.com:port"
```

On the mirror list, select a mirror close by. Usually HTTP mirrors suffice, but other protocols are available as well. Move to the **`releases/amd64/autobuilds/`** directory. There all available stage files are displayed (they might be stored within subdirectories named after the individual sub-architectures). Select one and press `d` to download.

After the stage file download completes, it is possible to verify the integrity and validate the contents of the stage file. Those interested should proceed to the next section (/wiki/Handbook:AMD64/Installation/Stage#Verifying_and_validating).

Those not interested in verifying and validating the stage file can close the command-line browser by pressing `q` and can move directly to the Unpacking the stage file (/wiki/Handbook:AMD64/Installation/Stage#Unpacking_the_stage_file) section.

Verifying and validating

Note
Most stages are now explicitly suffixed (https://www.gentoo.org/news/2021/07/20/more-downloads.html) with the init system type (**`openrc`** or **`systemd`**), although some architectures may still be missing these for now.

Like with the minimal installation CDs, additional downloads to verify and validate the stage file are available. Although these steps may be skipped, these files are provided for users who care about the integrity of the file(s) they just downloaded. The extra files are available under the root of the mirrors directory. Browse to the appropriate location for the hardware architecture and the system profile and download the associated **`.CONTENTS.gz`**, **`.DIGESTS`**, and **`.sha256`** files.

```
root # wget https://distfiles.gentoo.org/releases/
```

- **`.CONTENTS.gz`** - A compressed file that contains a list of all files inside the stage file.
- **`.DIGESTS`** - Contains checksums of the stage file in using several cryptographic hash algorithms.
- **`.sha256`** - Contains a PGP signed SHA256 checksum of the stage file. This file may not be available for download for all stage files.

Cryptographic tools and utilities such as **`openssl`**, **`sha256sum`**, or **`sha512sum`** can be used to compare the output with the checksums provided by the **`.DIGESTS`** file.

To verify the SHA512 checksum with **`openssl`**:

```
root # openssl dgst -r -sha512 stage3-amd64-<release>-<init>.tar.xz
```

`dgst` instructs the **openssl** command to use the Message Digest sub-command, `-r` prints the digest output in coreutils format, and `-sha512` selects the SHA512 digest.

To verify the BLAKE2B512 checksum with **openssl**:

```
root # openssl dgst -r -blake2b512 stage3-amd64-<release>-<init>.tar.xz
```

Compare the output(s) of the checksum commands with the hash and filename paired values contained within the **.DIGESTS** file. The paired values need to match the output of the checksum commands, otherwise the downloaded file is corrupt, and should be discarded and re-downloaded.

To verify the SHA256 hash from an associated **.sha256** file using the **sha256sum** utility:

```
root # sha256sum --check stage3-amd64-<release>-<init>.tar.xz.sha256
```

The `--check` option instructs **sha256sum** to read a list of expected files and associated hashes, and then print an associated "OK" for each file that calculates correctly or a "FAILED" for files that do not.

Just like with the ISO file, the cryptographic signature of the **tar.xz** file can be verified using **gpg** to ensure no tampering has been performed on the tarball.

For official Gentoo live images, the `sec-keys/openpgp-keys-gentoo-release` (https://packages.gentoo.org/packages/sec-keys/openpgp-keys-gentoo-release) package provides PGP signing keys for automated releases. The keys must first be imported into the user's session in order to be used for verification:

```
root # gpg --import /usr/share/openpgp-keys/gentoo-release.asc
```

For all non-official live images which offer **gpg** and **wget** in the live environment, a bundle containing Gentoo keys can be fetched and imported:

```
root # wget -O - https://qa-reports.gentoo.org/output/service-keys.gpg | gpg --import
```

Verify the signature of the tarball and, optionally, associated checksum files:

```
root # gpg --verify stage3-amd64-<release>-<init>.tar.xz.asc
root # gpg --verify stage3-amd64-<release>-<init>.tar.xz.DIGESTS
root # gpg --verify stage3-amd64-<release>-<init>.tar.xz.sha256
```

If verification succeeds, "Good signature from" will be in the output of the previous command(s).

The fingerprints of the OpenPGP keys used for signing release media can be found on the release media signatures page (https://www.gentoo.org/downloads/signatures/).

Installing a stage file

Once the *stage file* has been downloaded and verified, it can be extracted using **tar**:

```
root # tar xpvf stage3-*.tar.xz --xattrs-include='*.*' --numeric-owner -C /mnt/gentoo
```

Before extracting verify the options:

- `x` **e**x**tract, instructs **tar** to extract the contents of the archive.
- `p` **p**reserve permissions.
- `v` **v**erbose output.
- `f` **f**ile, provides **tar** with the name of the input archive.
- `--xattrs-include='*.*'` Preserves extended attributes in all namespaces stored in the archive.
- `--numeric-owner` Ensure that the user and group IDs of files being extracted from the tarball remain the same as Gentoo's release engineering team intended (even if adventurous users are not using official Gentoo live environments for the installation process).
- `-C /mnt/gentoo` Extract files to the root partition regardless of the current directory.

Now that the stage file is unpacked, proceed with Configuring compile options (/wiki/Handbook:AMD64/Installation/Stage#Configuring_compile_options).

Configuring compile options

Introduction

To optimize the system, it is possible to set variables which impact the behavior of Portage, Gentoo's officially supported package manager. All those variables can be set as environment variables (using **export**) but setting via **export** is not permanent.

Note
Technically variables can be exported via the shell's (/wiki/Shell) profile or rc files, however that is not best practice for basic system administration.

Portage reads in the **make.conf (/wiki/Make.conf)** file when it runs, which will change runtime behavior depending on the values saved in the file. **make.conf** can be considered the primary configuration file for Portage, so treat its content carefully.

Tip
A commented listing of all possible variables can be found in **/mnt/gentoo/usr/share/portage/config/make.conf.example**.

Additional documentation on **make.conf** can be found by running **man 5 make.conf**.

For a successful Gentoo installation only the variables that are mentioned below need to be set.

Fire up an editor (in this guide we use **nano**) to alter the optimization variables we will discuss hereafter.

```
root # nano /mnt/gentoo/etc/portage/make.conf
```

From the **make.conf.example** file it is obvious how the file should be structured: commented lines start with `#`, other lines define variables using the `VARIABLE="value"` syntax. Several of those variables are discussed in the next section.

CFLAGS and CXXFLAGS

The *CFLAGS* and *CXXFLAGS* variables define the optimization flags for GCC C and C++ compilers respectively. Although those are defined generally here, for maximum performance one would need to optimize these flags for each program separately. The reason for this is because every program is different. However, this is not manageable, hence the definition of these flags in the **make.conf** file.

In **make.conf** one should define the optimization flags that will make the system the most responsive generally. Don't place experimental settings in this variable; too much optimization can make programs misbehave (crash, or even worse, malfunction).

The Handbook will not explain all possible optimization options. To understand them all, read the GNU Online Manual(s) (https://gcc.gnu.org/onlinedocs/) or the gcc info page (**info gcc**). The **make.conf.example** file itself also contains lots of examples and information; don't forget to read it too.

A first setting is the `-march=` or `-mtune=` flag, which specifies the name of the target architecture. Possible options are described in the **make.conf.example** file (as comments). A commonly used value is *native* as that tells the compiler to select the target architecture of the current system (the one users are installing Gentoo on).

A second one is the `-O` flag (that is a capital O, not a zero), which specifies the gcc optimization class flag. Possible classes are s (for size-optimized), 0 (zero - for no optimizations), 1, 2 or even 3 for more speed-optimization flags (every class has the same flags as the one before, plus some extras). `-O2` is the recommended default. `-O3` is known to cause problems when used system-wide, so we recommend to stick to `-O2`.

Another popular optimization flag is `-pipe` (use pipes rather than temporary files for communication between the various stages of compilation). It has no impact on the generated code, but uses more memory. On systems with low memory, gcc might get killed. In that case, do not use this flag.

Using `-fomit-frame-pointer` (which doesn't keep the frame pointer in a register for functions that don't need one) might have serious repercussions on the debugging of applications.

When the *CFLAGS* and *CXXFLAGS* variables are defined, combine the several optimization flags in one string. The default values contained in the stage file archive should be good enough. The following one is just an example:

CODE **Example *CFLAGS* and *CXXFLAGS* variables**

```
# Compiler flags to set for all languages
COMMON_FLAGS="-march=native -O2 -pipe"
# Use the same settings for both variables
CFLAGS="${COMMON_FLAGS}"
CXXFLAGS="${COMMON_FLAGS}"
```

Tip
Although the GCC optimization (/wiki/GCC_optimization) article has more information on how the various compilation options can affect a system, the Safe CFLAGS (/wiki/Safe_CFLAGS) article may be a more practical place for beginners to start optimizing their systems.

MAKEOPTS

The *MAKEOPTS* variable defines how many parallel compilations should occur when installing a package. As of Portage version 3.0.31[1], if left undefined, Portage's default behavior is to set the *MAKEOPTS* jobs value to the same number of threads returned by **nproc**.

Further, as of Portage 3.0.53[2], if left undefined, Portage's default behavior is to set the *MAKEOPTS* load-average value to the same number of threads returned by **nproc**.

A good choice is the smaller of: the number of threads the CPU has, or the total amount of system RAM divided by 2 GiB.

Warning
Using a large number of jobs can significantly impact memory consumption. A good recommendation is to have at least 2 GiB of RAM for every job specified (so, e.g. `-j6` requires *at least* 12 GiB). To avoid running out of memory, lower the number of jobs to fit the available memory.

Tip
When using parallel emerges (`--jobs`), the effective number of jobs run can grow exponentially (up to make jobs multiplied by emerge jobs). This can be worked around by running a localhost-only distcc configuration that will limit the number of compiler instances per host.

FILE **/etc/portage/make.conf** **Example *MAKEOPTS* declaration**

```
# If left undefined, Portage's default behavior is to:
# - set the MAKEOPTS jobs value to the same number of threads returned by `nproc`
# - set the MAKEOPTS load-average value slightly above the number of threads returned by `nproc`, due to i
t being a damped value
# Please replace '4' as appropriate for the system (min(RAM/2GB, threads), or leave it unset.
MAKEOPTS="-j4 -l5"
```

Search for *MAKEOPTS in* ***man 5 make.conf*** *for more details.*

Ready, set, go!

Update the **/mnt/gentoo/etc/portage/make.conf** file to match personal preference and save (nano users would press `Ctrl`+`o` to write the change and then `Ctrl`+`x` to quit).

References

1. https://gitweb.gentoo.org/proj/portage.git/commit/?id=5d2af567772bb12b073f1671daea6263055cbdc2 (https://gitweb.gentoo.org/proj/portage.git/commit/?id=5d2af567772bb12b073f1671daea6263055cbdc2)
2. https://gitweb.gentoo.org/proj/portage.git/commit/?id=de7be7f45ee54e3f789def46542919550687d15e (https://gitweb.gentoo.org/proj/portage.git/commit/?id=de7be7f45ee54e3f789def46542919550687d15e)

← Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks) | Home (/wiki/Handbook:AMD64) | Installing base system → (/wiki/Handbook:AMD6