emerge --with-test-deps [y | n]

For packages matched by arguments, this option will pull in dependencies that are conditional on the "test" USE flag, even if "test" is not enabled in FEATURES for the matched packages. (see make.conf(5) for more information about FEATURES settings).

Example:

```
emerge --onlydeps --with-test-deps foo
```

@profile package set

@world = @system + @profile + @selected

The motivation to have @profile separate from @system is that @system packages may have incomplete dependency specifications (due to long-standing Gentoo policy), and incomplete dependency specifications have deleterious effects on the ability of emerge to parallelize builds. So, unlike @system, packages included in @profile do not hurt emerge's ability to parallelize.

Profile default EAPI

If the eapi file does not exist, then the EAPI defaults to 0 unless the default has been overridden by a profile_eapi_when_unspecified setting inside metadata/layout.conf of the containing repository.

emerge --sync-submodule <glsa|news|profiles>

Restrict sync to the specified submodule(s). This option may be specified multiple times, in order to sync multiple submodules. Currently, this option has no effect for sync protocols other than rsync.

emerge --changed-deps [y | n]

Tells emerge to replace installed packages for which the corresponding ebuild dependencies have changed since the packages were built. This option also implies the --selective option. Behavior with respect to changed build-time dependencies is controlled by the --with-bdeps option.

Similar to the emerge --newuse option.

emerge --binpkg-changed-deps [y | n]

Tells emerge to ignore binary packages for which the corresponding ebuild dependencies have changed since the packages were built. In order to help avoid issues with resolving inconsistent dependencies, this option is automatically enabled unless the --usepkgonly option is enabled. Behavior with respect to changed build-time dependencies is controlled by the --with-bdeps option.

Similar to the emerge --binpkg-respect-use option.

Binary packages - multiple compression types

- bzip2
- gzip
- lz4
- lzip
- lzop
- xz

Only decompression is supported now, for forward compatibility.

Binary package soname dependencies

PROVIDES
Contains information about the sonames that a package provides, which is automatically generated from the files that it installs. The sonames may have been filtered by the PROVIDES_EXCLUDE ebuild variable. A multilib category, followed by a colon, always precedes a list of one or more sonames.

Example:
x86_32: libcom_err.so.2 libss.so.2 x86_64: libcom_err.so.2 libss.so.2

REQUIRES
Contains information about the sonames that a package requires, which is automatically generated from the files that it installs. The sonames may have been filtered by the REQUIRES_EXCLUDE ebuild variable. Any sonames that a package provides are automatically excluded from REQUIRES. A multilib category, followed by a colon, always preceeds a list of one or more sonames.

Example:
x86_32: ld-linux.so.2 libc.so.6 x86_64: ld-linux-x86-64.so

Soname dependency resolution is disabled by default

Soname dependency resolution is disabled by default, since it will not work correctly unless all available installed and binary packages have been built by a version of portage which generates REQUIRES and PROVIDES metadata.

Soname dependency resolution is enabled when --ignore-soname-deps=n is specified, and one of the following is true:

- --usepkgonly option is enabled
- removal actions (--depclean and --prune)

Soname dependencies are automatically ignored for dependency calculations that can pull unbuilt ebuilds into the dependency graph, since unbuilt ebuilds do not have any soname dependency metadata, making it impossible to determine whether an unresolved soname dependency can be satisfied. Therefore, --usepkgonly must be used in order to enable soname dependency resolution when installing packages.

A new soname.provided file is supported for profiles, making it possible to selectively ignore soname dependencies (see the portage(5) man page).

FEATURES=binpkg-multi-instance

Enable support for multiple binary package instances per ebuild. Having multiple instances is useful for a number of purposes, such as retaining builds that were built with different USE flags or linked against different versions of libraries. The location of any particular package within PKGDIR can be expressed as follows:

    ${PKGDIR}/${CATEGORY}/${PN}/${PF}-${BUILD_ID}.xpak

The build-id starts at 1 for the first build of a particular ebuild, and is incremented by 1 for each new build. It is possible to share a writable PKGDIR over NFS, and locking ensures that each package added to PKGDIR will have a unique build-id. It is not necessary to migrate an existing PKGDIR to the new layout, since portage is capable of working with a mixed PKGDIR layout, where packages using the old layout are allowed to remain in place.

The new PKGDIR layout is backward-compatible with binhost clients running older portage, since the file format is identical, the per-package PATH attribute in the 'Packages' index directs them to download the file from the correct URI, and they automatically use BUILD_TIME metadata to select the latest builds.