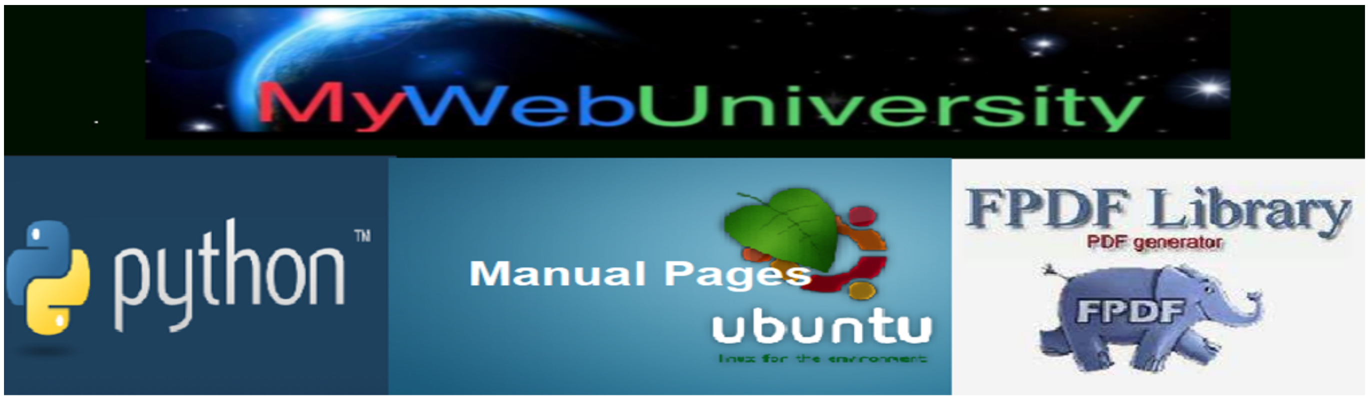

Rocky Enterprise Linux 9.2 Manual Pages on command 'go-build.1'

$ man go-build.1

```
GO-BUILD(1)                General Commands Manual                GO-BUILD(1)

NAME
       go-build - compile the packages named by the import paths

SYNOPSIS
       go build [-o output] [build flags] [packages]

DESCRIPTION
       Build  compiles the packages named by the import paths, along with their dependencies, but
       it does not install the results.

       If the arguments to build are a list of .go files from a single  directory,  build  treats
       them as a list of source files specifying a single package.

       When compiling packages, build ignores files that end in '_test.go'.

       When  compiling  a single main package, build writes the resulting executable to an output
       file named after the first source file ('go build ed.go rx.go' writes 'ed' or 'ed.exe') or
       the  source  code  directory  ('go build unix/sam' writes 'sam' or 'sam.exe').  The '.exe'
       suffix is added when writing a Windows executable.

       When compiling multiple packages or a single non-main package, build compiles the packages
       but discards the resulting object, serving only as a check that the packages can be built.

       The  -o  flag forces build to write the resulting executable or object to the named output
       file or directory, instead of the default behavior described in the last  two  paragraphs.
       If  the  named output is an existing directory or ends with a slash or backslash, then any
       resulting executables will be written to that directory.

       The -i flag installs the packages that are dependencies of the target.
       The -i flag is deprecated. Compiled packages are cached automatically.
```

OPTIONS

The build flags are shared by the build, clean, get, install, list, run, and test com?
mands:

-a force rebuilding of packages that are already up-to-date.

-n print the commands but do not run them.

-p n the number of programs, such as build commands or test binaries, that can be run in
parallel.

The default is GOMAXPROCS, normally the number of CPUs available.

-race enable data race detection.

Supported only on linux/amd64, freebsd/amd64, darwin/amd64, windows/amd64,
linux/ppc64le and linux/arm64 (only for 48-bit VMA).

-msan enable interoperation with memory sanitizer. Supported only on linux/amd64,
linux/arm64 and only with Clang/LLVM as the host C compiler. On linux/arm64, pie
build mode will be used.

-v print the names of packages as they are compiled.

-work print the name of the temporary work directory and do not delete it when exiting.

-x print the commands.

-asmflags '[pattern=]arg list'

arguments to pass on each go tool asm invocation.

-buildmode mode

build mode to use. See 'go help buildmode' for more.

-compiler name

name of compiler to use, as in runtime.Compiler (gccgo or gc)

-gccgoflags 'arg list'

arguments to pass on each gccgo compiler/linker invocation

-gcflags 'arg list'

arguments to pass on each go tool compile invocation.

-installsuffix suffix

a suffix to use in the name of the package installation directory, in order to keep
output separate from default builds. If using the -race flag, the install suffix
is automatically set to race or, if set explicitly, has _race appended to it. Like?
wise for the -msan flag. Using a -buildmode option that requires non-default com?
pile flags has a similar effect.

-ldflags 'flag list'

arguments to pass on each go tool link invocation.

-linkshared

build code that will be linked against shared libraries previously created with -buildmode=shared.

-mod mode

module download mode to use: readonly, vendor, or mod. By default, if a vendor di? rectory is present and the go version in go.mod is 1.14 or higher, the go command acts as if -mod=vendor were set. Otherwise, the go command acts as if -mod=read? only were set. See https://golang.org/ref/mod#build-commands for details.

-modcacherw

leave newly-created directories in the module cache read-write instead of making them read-only.

-modfile file

in module aware mode, read (and possibly write) an alternate go.mod file instead of the one in the module root directory. A file named "go.mod" must still be present in order to determine the module root directory, but it is not accessed. When -mod? file is specified, an alternate go.sum file is also used: its path is derived from the -modfile flag by trimming the ".mod" extension and appending ".sum".

-overlay file

read a JSON config file that provides an overlay for build operations. The file is a JSON struct with a single field, named 'Replace', that maps each disk file path (a string) to its backing file path, so that a build will run as if the disk file path exists with the contents given by the backing file paths, or as if the disk file path does not exist if its backing file path is empty. Support for the -over? lay flag has some limitations: importantly, cgo files included from outside the in? clude path must be in the same directory as the Go package they are included from, and overlays will not appear when binaries and tests are run through go run and go test respectively.

-pkgdir dir

install and load all packages from dir instead of the usual locations. For exam? ple, when building with a non-standard configuration, use -pkgdir to keep generated packages in a separate location.

-tags tag,list

a comma-separated list of build tags to consider satisfied during the build. For more information about build tags, see the description of build constraints in the documentation for the go/build package. (Earlier versions of Go used a space-sepa? rated list, and that form is deprecated but still recognized.)

-trimpath

remove all file system paths from the resulting executable. Instead of absolute file system paths, the recorded file names will begin with either "go" (for the standard library), or a module path@version (when using modules), or a plain import path (when using GOPATH).

-toolexec 'cmd args'

a program to use to invoke toolchain programs like vet and asm. For example, in? stead of running asm, the go command will run ?cmd args /path/to/asm <arguments for asm>?.

The TOOLEXEC_IMPORTPATH environment variable will be set, matching ?go list -f {{.ImportPath}}? for the package being built.

The -asmflags, -gccgoflags, -gcflags, and -ldflags flags accept a space-separated list of arguments to pass to an underlying tool during the build. To embed spaces in an element in the list, surround it with either single or double quotes. The argument list may be pre? ceded by a package pattern and an equal sign, which restricts the use of that argument list to the building of packages matching that pattern (see 'go help packages' for a de? scription of package patterns). Without a pattern, the argument list applies only to the packages named on the command line. The flags may be repeated with different patterns in order to specify different arguments for different sets of packages. If a package matches patterns given in multiple flags, the latest match on the command line wins. For example, 'go build -gcflags=-S fmt' prints the disassembly only for package fmt, while 'go build -gcflags=all=-S fmt' prints the disassembly for fmt and all its dependencies.

For more about specifying packages, see go-packages(7).

For more about where packages and binaries are installed, see go-gopath(1).

For more about calling between Go and C/C++, run 'go help c'.

Note: Build adheres to certain conventions such as those described by 'go help gopath'. Not all projects can follow these conventions, however. Installations that have their own conventions or that use a separate software build system may choose to use lower-level in?

vocations such as 'go tool compile' and 'go tool link' to avoid some of the overheads and design decisions of the build tool.

SEE ALSO

go-install(1), go-get(1), go-clean(1).

AUTHOR

This manual page was written by Michael Stapelberg <stapelberg@debian.org> and is main?tained by the Debian Go Compiler Team <team+go-compiler@tracker.debian.org> based on the output of 'go help build' for the Debian project (and may be used by others).