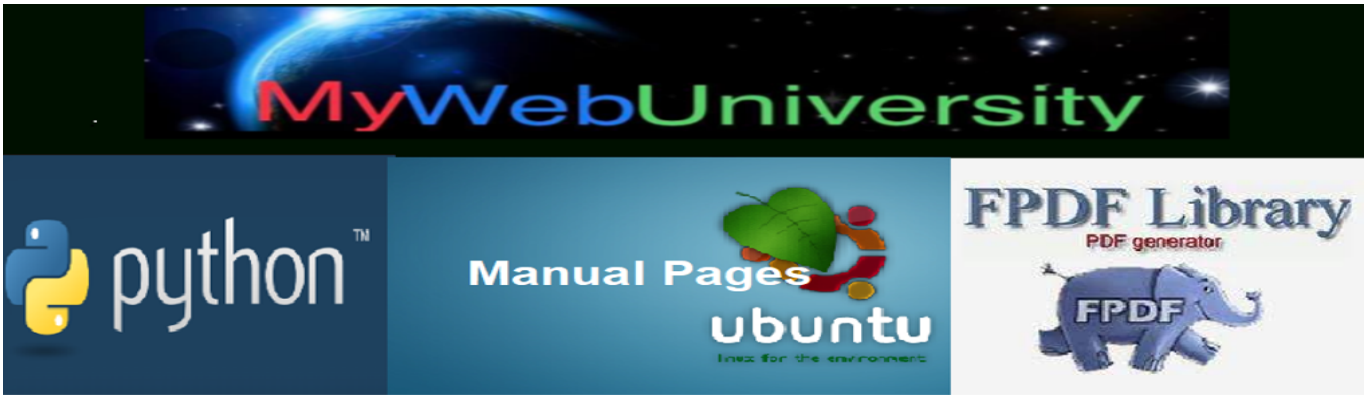

Rocky Enterprise Linux 9.2 Manual Pages on command 'go-path.7'

$ man go-path.7

GO-PATH(7) Miscellaneous Information Manual GO-PATH(7)

NAME

go - tool for managing Go source code

DESCRIPTION

The Go path is used to resolve import statements. It is implemented by and documented in the go/build package.

The GOPATH environment variable lists places to look for Go code. On Unix, the value is a colon-separated string. On Windows, the value is a semicolon-separated string. On Plan 9, the value is a list.

If the environment variable is unset, GOPATH defaults to a subdirectory named "go" in the user's home directory ($HOME/go on Unix, %USERPROFILE% on Windows), unless that directory holds a Go distribution. Run "go env GOPATH" to see the current GOPATH.

See https://golang.org/wiki/SettingGOPATH to set a custom GOPATH.

Each directory listed in GOPATH must have a prescribed structure:

The src directory holds source code. The path below src determines the import path or exe? cutable name.

The pkg directory holds installed package objects. As in the Go tree, each target operat? ing system and architecture pair has its own subdirectory of pkg (pkg/GOOS_GOARCH).

If DIR is a directory listed in the GOPATH, a package with source in DIR/src/foo/bar can be imported as "foo/bar" and has its compiled form installed to "DIR/pkg/GOOS_GOARCH/foo/bar.a".

The bin directory holds compiled commands. Each command is named for its source direc? tory, but only the final element, not the entire path. That is, the command with source in

DIR/src/foo/quux is installed into DIR/bin/quux, not DIR/bin/foo/quux. The "foo/" prefix is stripped so that you can add DIR/bin to your PATH to get at the installed commands. If the GOBIN environment variable is set, commands are installed to the directory it names instead of DIR/bin. GOBIN must be an absolute path.

Here's an example directory layout:

```
GOPATH=/home/user/gocode

/home/user/gocode/
    src/
        foo/
            bar/               (go code in package bar)
                x.go
            quux/              (go code in package main)
                y.go
    bin/
        quux                   (installed command)
    pkg/
        linux_amd64/
            foo/
                bar.a          (installed package object)
```

Go searches each directory listed in GOPATH to find source code, but new packages are al? ways downloaded into the first directory in the list.

See https://golang.org/doc/code.html for an example.

GOPATH and Modules

When using modules, GOPATH is no longer used for resolving imports. However, it is still used to store downloaded source code (in GOPATH/pkg/mod) and compiled commands (in GOPATH/bin).

Internal Directories

Code in or below a directory named "internal" is importable only by code in the directory tree rooted at the parent of "internal". Here's an extended version of the directory lay? out above:

```
/home/user/go/
    src/
        crash/
```

```
        bang/             (go code in package bang)
            b.go
        foo/              (go code in package foo)
            f.go
            bar/          (go code in package bar)
                x.go
            internal/
                baz/      (go code in package baz)
                    z.go
            quux/         (go code in package main)
                y.go
```

The code in z.go is imported as "foo/internal/baz", but that import statement can only ap? pear in source files in the subtree rooted at foo. The source files foo/f.go, foo/bar/x.go, and foo/quux/y.go can all import "foo/internal/baz", but the source file crash/bang/b.go cannot.

See https://golang.org/s/go14internal for details.

Vendor Directories

Go 1.6 includes support for using local copies of external dependencies to satisfy imports of those dependencies, often referred to as vendoring.

Code below a directory named "vendor" is importable only by code in the directory tree rooted at the parent of "vendor", and only using an import path that omits the prefix up to and including the vendor element.

Here's the example from the previous section, but with the "internal" directory renamed to "vendor" and a new foo/vendor/crash/bang directory added:

```
    /home/user/go/
        src/
            crash/
                bang/             (go code in package bang)
                    b.go
            foo/              (go code in package foo)
                f.go
                bar/          (go code in package bar)
                    x.go
```

```
vendor/
    crash/
        bang/      (go code in package bang)
            b.go
    baz/           (go code in package baz)
        z.go
quux/              (go code in package main)
    y.go
```

The same visibility rules apply as for internal, but the code in z.go is imported as "baz", not as "foo/vendor/baz".

Code in vendor directories deeper in the source tree shadows code in higher directories. Within the subtree rooted at foo, an import of "crash/bang" resolves to "foo/vendor/crash/bang", not the top-level "crash/bang".

Code in vendor directories is not subject to import path checking (see 'go help importpath').

When 'go get' checks out or updates a git repository, it now also updates submodules.

Vendor directories do not affect the placement of new repositories being checked out for the first time by 'go get': those are always placed in the main GOPATH, never in a vendor subtree.

See https://golang.org/s/go15vendor for details.

AUTHOR

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