Introduction

This document is about the policy for adding a new port to the main Go repository. By port we mean an operating system + architecture combination, such as linux/386.

The goal of this policy is to avoid the accumulation of incomplete or broken ports.

Requirements for a new port

Before any code relating to a port can be added to the main Go repository, the following must all be done:

- A proposal must be filed and accepted in which the Go team accepts overall responsibility for having the new port in the core Go tree. In general, each new port carries an upkeep cost separate from the direct maintenance. That cost varies by port, depending on how similar a new port is to existing ones. The cost must be balanced by an overall benefit in the form of potential new users or use cases for Go.
- At least one developer must be named (and agree) to maintain the port, by making required updates in a timely manner as architecture or operating system requirements change.
- A developer must be named (and agree) to maintain the builder, the machine trying each git revision and providing data for https://build.golang.org.
- The builder must already be running (and failing, because the code is not yet in the main repository).
- All CLs necessary to run all.bash successfully must have been sent for review. Typically this will be a handful of CLs split by the part of the tree they change.

Once those conditions are satisfied, the Go team can accept the port and begin to merge the CLs. Once the CLs are all submitted, all.bash must pass, so that the builder reports “ok” in the dashboard.

Any port started during a release cycle must be finished (all.bash passing, builder reporting “ok”) before the corresponding release freeze, or else the code will be removed at the freeze.

Other repositories

Although it is not part of the core repository, the x/sys repository should add support for the new port before the release happens because it is the official place to add new system calls.

First class ports

Some ports are considered "first class". The distinction is mostly about releases and distribution.

A first class port has these properties:

- Broken builds block releases
  - All contributors are effectively responsible for these ports (You break it, you fix it, or find someone who can.)
- Official binaries are provided
  - Requires Google's Go team to own the builder machine
- Installation is documented at https://go.dev/doc/install

Graduating a port to "first class" is at the discretion of the Go team at Google.

The current first class ports are:

- darwin/amd64
- darwin/arm64
- linux/386
- linux/amd64
- linux/arm
- linux/arm64
- windows/386
- windows/amd64

We distribute binaries for other GOOS/GOARCH pairs (other "ports"), but they are not "first class" by this definition.

All Linux first class ports are for systems using glibc only. Linux systems using other C libraries are not fully supported and are not treated as first class.

Removing a port

If a builder for a particular port starts failing, the code should be corrected as soon as possible; otherwise future regressions cannot be detected, and the amount of work required to bring the builder back to "ok" compounds.

Ultimately the job of making the port work again falls to the developer maintaining the port, although simple cases (such as +build lines in new code) can and should be fixed by others.

If a builder fails for more than two weeks, it is time to start looking for a more active maintainer for the port.

If a builder fails for more than four weeks or is failing at the time of a release freeze, and a new maintainer cannot be found, the port will be removed from the tree.

Due to backwards compatibility concerns, removing a formerly working port should be a last resort. Finding a new maintainer is always preferable.

Getting started

See https://groups.google.com/forum/#!topic/golang-dev/SRUK7yJVA0c for some discussion on how to go about writing a new port.

Comments and Questions

Comments or questions about the policy should be sent to golang-dev.