2018-11-06 - golang-tools session - meeting notes

Previous session notes

2018-10-23 - golang-tools session - meeting notes

Recording

Apologies - recording malfunction this time around, which also means the notes below are a “best efforts” recollection of discussion. Please comment with any additions/corrections etc.

Attendees

- Paul Jolly
- Ian Cottrell
- bhcleek
- Ramya Rao
- bcmills
- Alan Donovan
- Rebecca Stambler
- Michael Matloob
- Jay Conrod
- Roger Peppe
- Daniel Marti

Agenda

- VSCode, Atom and vim-go trials with module-mode
- gocode, godef, goimports and other tools under the tracking umbrella issue https://github.com/golang/go/issues/24661
- go/packages
- go/analysis
- Alan's changes to vet
- cmd/go changes, including speed improvements
- LSP
- Quick overview of various modules-aware tools:
    - new cmd/go/internal packages
    - go-internal
    - gomodmerge
    - apidiff
    - goforward
    - gohack
    - mod
    - gobin
- AOB

Notes

- Thanks to Rebecca for setting up and moving us to Hangouts; easier for people to join, follow along. We will ensure the recording works for next time.
- VSCode, Atom and vim-go trials with module-mode
    - VSCode - Ramya
        - User numbers ~1100
        - No new features since last call
        - Issues

                - gorename is becoming a “must have”
                - Context is manifesting as more of an issue because, for example, following a godef that takes you to a file in the module cache, the working directory is now wrong
                - Ramya has Looped in Ian - need to solve context issue - **action Ian, Ramya**
                - Following up on Marwan’s question, Ramya has confirmed that setting GOFLAGS=-mod=vendor works just fine
    - Atom - Zak, Joe
        - Zak couldn’t make the call and so sent his apologies
        - Nothing new on the Atom side to report
        - Zak is going to try the module-aware goimports and integrate that soon
    - vim-go - Billie
        - Release 1.19 which has support for gocode module aware mode
        - Not clear how we’ll handle context yet
        - Most commands get executed in directory of buffer
        - Can potentially be looped in to VSCode discussions - **action Ian, Ramya, Billie**
- gocode, godef, goimports and other tools under the tracking umbrella issue https://github.com/golang/go/issues/24661
    - godef - Ian
        - Tested in old and new form with new testing framework
        - Speed ok
- go/packages
    - No specific updates following the breaking change to move to the file= pattern
    - Daniel asked about adding go list’s .Match to the go/packages output (or at least which loaded packages were explicit)
        - Michael responded
        - This would not supported on Blaze or others
        - There are certain patterns that all drivers must support - **action Ian/Michael to share details**
        - Something of a niche problem
        - Take offline into a golang-tools thread
- go/analysis
    - Daniel asked Alan a number of questions:
        - How should analyzers handle loading flags, such as -tags=list and -tests=bool?
            - Analyzers are called by a driver, and that’s the job of the driver. For example “go vet” runs an analyzer tool on exactly the packages that would be built by the corresponding “go build” or “go test” commands.
        - Should analyzers expose flags like -verbose or -debug?
            - You can if you like. I would hope that these are for maintainers not end users. There is no global verbosity control.
        - Why not just include packages.Package in analysis.Pass?
            - Because packages.Package is one particular representation of a Go package, and it’s coupled to a bunch of machinery for materializing it that I don’t want cmd/vet to depend on. The point of modular analysis is that the analysis tool can be invoked on a single package at a time, with results passed from one unit to the next through the file system, as in go vet, so you don’t need to load all the packages at once.
        - Would be good to have a simpler SSA example than analysis/passes/buildssa, or to have an API similar to ssautil.BuildPackage/Packages
            - buildssa is not an example of SSA, it’s the adaptor that makes the SSA representation easily available to other analysis passes. For an example of how to use buildssa, see https://github.com/golang/tools/blob/master/go/analysis/passes/nilness/nilness.go#L55. It’s basically a one-liner.
        - How can one construct a callgraph? E.g. go/callgraph/cha takes a *ssa.Program, but the buildssa example only builds a *ssa.Package. A working example would be great.

                - The premise of modular analysis is that you get to see exactly one package at a time, just like a compiler. A callgraph requires stitching together information from every package---it's analogous to a linker. Therefore you cannot build a whole-program call graph in the analysis framwork, by design. (We could add an analogue of link-time whole program analysis to the framework, but we haven't sketched it out yet.) You can build a callgraph of an ssa.Program containing a single package, but it's inherently going to be very limited; you would have to use the "static" algorithm but not cha, rta or pta.
- Vet changes
    - Daniel had a number of questions regarding the reorganisation
        - Where should people be making changes now?
            - To cmd/vet, but I will need to apply the same change to x/tools/go/analysis until the switch, which I hope to make any day now.
        - Are changes to cmd/vet being ported to the x/tools repo?
            - Yes.
        - Vendor vet-lite opt-in
            - I discussed with Russ and we're going to make a hard switch instead of a gradual transition. The only obstacle right now is cmd/vet/all.
        - One-line to opt in to new behaviour
            - Moot, see above.
        - Nilness package
            - I don't remember the question. :) The nilness package is an SSA-based analysis for problems such as if p==nil { … use(*p) }. Because it uses SSA I don't plan to add it to the core set of vet analyzers.
- cmd/go changes, including speed improvements
    - No specific updates on the speed front
    - Bryan is in the process of landing some significant changes to cmd/go to make all go operations safe for concurrent execution using file locking
    - Bryan and Russ have also triaged the open modules issues to decide what should make the cut for Go 1.12
- LSP
    - No specific update
    - Rebecca continues to work on a first-cut of the LSP
    - The current version is currently working "out of the box" (with some config) in VSCode
    - It is therefore anticipated that it should also work "out of the box" with any other LSP-compatible editor/plugin (Atom, Vim + LSP plugin)
    - Goto definition support also being integrated
- AOB
    - Follow up on https://github.com/golang/go/issues/24661#issuecomment-436008945
    - The question being debated is how to "release" goimports and other golang.org/x/tools commands
    - We short-circuited this conversation to conclude the proposed method of releasing to a GitHub repo is the most straightforward: hence https://github.com/heschik/goimports was born
- Tools/packages roundup
    - The idea behind this new section is to introduce modules-aware tools, interesting packages, etc
    - Please send your favourite tools, packages etc. Self promotion is absolutely encouraged!
    - New cmd/go/internal packages - Bryan
        - A number of interesting packages:
            - lockedfile - package lockedfile creates and manipulates files whose contents should only change atomically.
            - https://go-review.googlesource.com/c/go/+/145178
            - renameio - renameio.WriteFile writes files atomically by renaming temporary files
            - https://go-review.googlesource.com/c/go/+/146377
    - New API to expose module information at runtime - Hana
        - runtime/debug: add API to read module info in binary
        - https://go-review.googlesource.com/c/go/+/144220
    - github.com/rogpeppe/go-internal

        - Selected cmd/go/internal packages factored out from the standard library
            - testscript - package testscript provides support for defining filesystem-based tests by creating scripts in a directory.
            - txtar - package txtar implements a trivial text-based file archive format.
            - semver - semver implements comparison of semantic version strings
    - github.com/rogpeppe/gomodmerge
        - gomodmerge merges dependency requirements from non-module build systems such as dep into a go.mod file
    - apidiff - Jonathan Amsterdam
        - apidiff determines whether two versions of the same package are compatible.
        - https://go-review.googlesource.com/c/exp/+/143597
        - Jonathan will join us on the next call to provide a more comprehensive introduction
    - goforward
        - The goforward command adds forwarding declarations from one Go package to another
        - https://go-review.googlesource.com/c/tools/+/137076
    - github.com/rogpeppe/gohack
        - The gohack command automates checking out a mutable copy of a module dependency and adding the relevant replace statement to the go.mod file
        - Example of how to use it
    - github.com/marwan-at-work/mod/cmd/mod
        - mod upgrades/downgrades semantic import paths in Go modules
        - Example of how to use it
    - github.com/myitcv/gobin
        - The gobin command installs/runs main packages
        - Example of how to use it

Next call

27 Nov 2018 16:30 UTC

Current agenda:

- AppEngine vs go/packages - Chris Broadfoot
- apidiff: an introduction - Jonathan Amsterdam
- Athens: an update - Aaron Schlesinger

in addition to the regular current topics:

- updates on editor module-mode "betas" from VSCode, Atom and Vim
- any updates on conversion of tools that fall under the tools umbrella issue (https://github.com/golang/go/issues/24661)
- go/packages updates
- go/analysis updates
- cmd/go updates
- LSP updates
- AOB