Gosh Documentation

Release 0.2.3

Kouhei Maeda

Mar 09, 2017

Contents

Contents:

CHAPTER 1

Gosh: interactive shell for golang

`Gosh` is the interactive Golang shell. The goal is to provide an easy-to-use interactive execution environment.

Documentation

http://gosh.readthedocs.org/

Features

- Interactive shell
- Enable to omit the main function
- Enable to omit package statement
- Enable to omit the import statement of standard library
- Enable to Import libraries of non-standard library
- Enable to re-declare function, type
- Ignoring duplicate import package
- Ignoring unused import package

Requirements

- Golang >= 1.2
- goimports command
  - We recommend that you install `goimports` to `$PATH` in advance.
  - Installing automatically if the command is not found in `$PATH` (>= v0.3.0).

  - **However, the time until the installation is complete in this case,** you will be waiting for the launch of “`Gosh`” process.

for documentation

- libpython2.7-dev
- libjpeg9-dev

Installation

Debian

Install the following packages

- golang
- golang-go.tools (recommended)

Set `GOPATH`:

```
$ install -d /path/to/gopath
$ export GOPATH=/path/to/gopath
```

If you install `goimports` in advance (recommended):

```
$ sudo apt-get install -y golang-go.tools
```

Install `Gosh` to `GOPATH`:

```
$ go get github.com/mkouhei/gosh
```

OS X

Install the follow packages with Homebrew.

- Go
- Mercurial (with Homebrew)

Set `GOPATH`:

```
$ install -d /path/to/gopath
$ export GOPATH=/path/to/gopath
```

If you install `goimports` in advance (recommend):

```
$ export PATH=${GOPATH}/bin:$PATH
$ go get golang.org/x/tools/cmd/goimports
```

Install the `Gosh`:

```
$ go get github.com/mkouhei/gosh
```

Basic usage

Examples:

```
$ $GOPATH/bin/gosh
>>> import "fmt"
>>> func main() {
>>> fmt.Println("hello")
>>> }
hello
>>>
```

or:

```
$ $GOPATH/bin/gosh
>>> func main() {
>>> fmt.Println("hello")
>>> }
hello
>>>
```

Note: Enabled to omit *import* statement for standard packages.

Enabled to import non-standard packages

Example of using non-standard package:

```
>>> import "net/http"
>>> import "example.org/somepkg"
>>> func main() {
>>> r, _ := http.Get("http://example.org/some")
>>> defer r.Body.Close()
>>> p, _ := somepkg.Reader(r.Body)
>>> fmt.Println(p)
>>> }
(print some payload)
```

Usage when omitting main function declarations

Example:

```
$ $GOPATH/bin/gosh
>>> i := 1
>>> i++
>>> fmt.Println(i)
2
>>>
```

Terminate `Gosh` to reset `main` declarations or declare `func main()` without body:

```
$ $GOSH/bin/gosh
>>> i := i
>>> fmt.Println(i)
```

```
1
>>> func main() {}
>>> fmt.Println(i)
[error] # command-line-arguments
./gosh_tmp.go:8: undefined: i
>>>
```

Limitations

- `fmt.Print*` are executed only once.

Known issues

Fail to evaluate when there are declared and not used valiables.:

```
$ $GOPATH/bin/gosh
>>> i := 1
>>> fmt.Println("hello")
>>>
```

Roadmap

- Tab completion
- Enable to omit `import` statement for global(system) installed packages

License

`Gosh` is licensed under GPLv3.

CHAPTER 2

User Manual

Run Gosh

Automatically installing `goimports` (>= v0.3.0)

Note: We recommend that you install `goimports` to `$PATH` in advance.

- Installing automatically if the command is not found in `$PATH`.
- However, the time until the installation is complete in this case, you will be waiting for the launch of "`Gosh`" process.

Check version

```
$GOPATH/bin/gosh -version
version: v0.x.x
```

Basic mode

```
$GOPATH/bin/gosh
go version go1.3.3 linux/amd64

Gosh v0.x.x
Copyright (C) 2014,2015 Kouhei Maeda
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software, and you are welcome to redistribute it.
There is NO WARRANTY, to the extent permitted by law.
>>>
```

Debug mode

```
$GOPATH/bin/gosh -d
```

Sudo mode (>= v0.3.0)

Warning: The `gosh` runs `sudo -E go run $tmppath/gosh-?????????/gosh_tmp.go` in this mode. Be careful your code, don't run the dangerous code. There is no warranty for this free software. See also the GPLv3 LICENSE.

You can use sudo mode `-s` option when Your code requires the privilege, for example using ICMP echo request.:

```
$GOPATH/bin/gosh -s="sudopassword"
```

Terminate Gosh

Enter Ctrl+D:

```
>>> [gosh] terminated
$
```

Execute main function

Go syntax validly:

```
>>> package main
>>> import "fmt"
>>> func main() {
>>> fmt.Println("hello")
>>> }
hello
>>>
```

Omit `package`, `import` statement, `main func`

Gosh supports omitting as follows;

- "`package`" statement
- "`import`" statement for standard libraries
- "`func main`" signature

So users give the same results with the following.:

```
>>> fmt.Println("hello")
hello
>>>
```

`fmt.Print*` are executed only once

```
>>> i:=1
>>> for i < 3 {
>>> fmt.Println(i)
>>> i++
>>> }
```

This omit `main func` is equivalent to the main func not following omitted.:

```
>>> func main() {
>>> i:=1
>>> for i < 3 {
>>> fmt.Println(i)
>>> i++
>>> }
>>> }
```

But, `fmt.Print*` are executed only once.:

```
>>> fmt.Println(1)
1
>>> fmt.Println(2)
2
```

This `fmt.Print*` are removed main body after executing main function.

Reset declaration of main func

Execute follow command.:

```
>>> func main() {}
```

For example, test function(),:

```
>>> func test() {
>>> fmt.Println("hello")
>>> }
```

Execute test() twice,:

```
>>> test()
hello
>>> test()
hello
hello
```

This is equivalent to the main func not following omitted.:

```
>>> func main() {
>>> test()
>>> test()
>>> }
```

So, print “hello” once after reset main.:

```
>>> test()
hello
>>> func main() {}
>>> test()
hello
```

Import packages

Gosh supports imports 3rd party libraryies. Gosh enter the `import "package"`, Gosh executes `go get` and installs the package into the `$GOPATH` of Gosh process.

For example of using the some package.:

```
>>> import "example.org/somepkg"
>>> resp, _ := http.Get("http://example.org/some")
>>> defer resp.Body.Close()
>>> payload, _ := somepkg.Reader(resp.Body)
>>> fmt.Println(payload)
(print some payload)
```

Users are able to omit import "`net/http`" package that is Go standard library.

If users import the same package, Gosh ignores duplicate import, adn treats as import of only once.

Declaration of type

Gosh supoorts declaration of type.:

```
>>> type foo struct {
>>> msg string
>>> cnt int
>>> }
>>> f := foo{"hello", 0}
>>> for f.cnt < 3 {
>>> fmt.Println(f.msg)
>>> f.cnt++
>>> }
hello
hello
hello
>>>
```

Gosh supports re-declarations of type. (>= v0.3.0)

Declaration of function

Gosh supports declaration of function.:

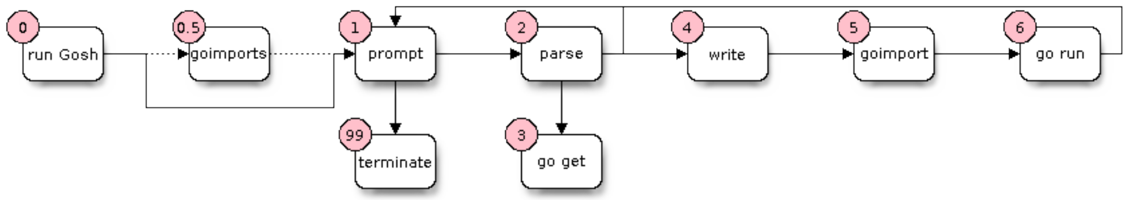
```
>>> func test(msg string) bool {
>>> if strings.HasPrefix(msg, "Hello") {
>>> return true
>>> }
>>> return false
>>> }
>>> fmt.Println(test("helo"))
false
>>> fmt.Println(test("hello"))
false
>>> fmt.Println(test("Hello"))
true
```

Gosh supports re-declarations of function.:

```
>>> func bar() {
>>> fmt.Println("hello")
>>> }
>>> bar()
hello
>>> func bar() {
>>> fmt.Println("bye")
>>> }
>>> bar()
bye
bye
```

CHAPTER 3

Internals



No	Name	Description
0	run Gosh	User executeted `$GOPATH/bin/gosh`
1	prompt	User is able to input commands
2	parse	Gosh parse from inputted command
3	go get	Gosh executes `go get example.org/sompkg` when user inputted `import "example.org/somepkg"`
4	write	Gosh write `$tmpdir/gosh-?????????/gosh_tmp.go` when user inputted `main func` (include omitted main)
5	goimport	Gosh executes `goimport $tmpdir/gosh-?????????/gosh_tmp.go`
6	go run	Gosh executes `go run $tmpdir/gosh-?????????/gosh_tmp.go`
99	terminate	User inputted `Ctrl+D`
0.5	goimports	Gosh executes `go get code.google.com/p/go.tools/cmd/goimport` if `goimports` command is not found

CHAPTER 4

History

0.3.0 (not-release)

- Re-declaration of type.
- Added user documentation.
- Automatically installing `goimports`.
- Added `sudo -E go run` option. (`-s=sudopassword`)

0.2.3 (2015-01-18)

- Fixed not running multiple gosh processes.

0.2.2 (2015-01-14)

- Fixed input unnecessary “Enter”.

0.2.1 (2015-01-13)

- Fixed declared function is executing immediately after func main.

0.2.0 (2015-01-08)

- Enable to omit the main function.
- Enable to re-declare function.

- refactoring of parser import, type, function declaration with go/scanner, go/token instead of regexp.
- Added parser of typeDecl.
- Fixed some bugs.
- Added to execute go get for goimports in Makefile.
- Applied golint, go vet.

0.1.7 (2014-11-28)

- Supported struct method and pointer parameters and results of function.
- Supported type of function.
- Appended func parser.
- Fixes allowing blanks of the begening of ImportDecl.
- Fixed Installation syntax of README

0.1.6 (2014-11-23)

- Supported patterns of ImportDecl supported by `go run`, for example, `[ . | PackageName ] "importPath"` syntax.
- Supported patterns of PackageClause supported by `go run`.

0.1.5 (2014-11-16)

- Unsupported Go 1.1.
- Added goVersion(), printing license.
- Appended GPLv3 copying permission statement.
- Appended printFlag argument to runCmd().

0.1.4 (2014-11-15)

- Fixed not work go run when noexistent package in parser.importPkgs.
- Changed log.Printf instead of log.Fatalf when error case at logger().
- Changed appending message string to returns of runCmd().

0.1.3 (2014-11-13)

- Fixed runtime error occurs when invalid import statement.
- Fixes issue infinite loop of go get.
- Cleanup all working directories on boot.

- Cleard parser.body when non-declaration statement.

0.1.2 (2014-11-12)

- Changed to print error of runCmd.
- Suppressed “go install: no install location”.
- Fixed lacking newline when writing.

0.1.1 (2014-11-10)

- Fixed deadlock occurs when typing `Ctrl+D` immediately after gosh start.
- Fixed fail override tmp code file.

0.1.0 (2014-11-09)

- First release

CHAPTER 5

Indices and tables

- genindex
- modindex
- search