Command vet

Vet examines Go source code and reports suspicious constructs, such as Printf calls whose arguments do not align with the format string. Vet uses heuristics that do not guarantee all reports are genuine problems, but it can find errors not caught by the compilers.

Vet is normally invoked through the go command. This command vets the package in the current directory:

```
go vet
```

whereas this one vets the packages whose path is provided:

```
go vet my/project/...
```

Use "go help packages" to see other ways of specifying which packages to vet.

Vet's exit code is non-zero for erroneous invocation of the tool or if a problem was reported, and 0 otherwise. Note that the tool does not check every possible problem and depends on unreliable heuristics, so it should be used as guidance only, not as a firm indicator of program correctness.

To list the available checks, run "go tool vet help":

```
asmdecl      report mismatches between assembly files
and Go declarations
assign       check for useless assignments
atomic       check for common mistakes using the
sync/atomic package
bools        check for common mistakes involving
boolean operators
buildtag     check that +build tags are well-formed and
correctly located
cgocall      detect some violations of the cgo pointer
passing rules
composites   check for unkeyed composite literals
copylocks    check for locks erroneously passed by
value
httpresponse check for mistakes using HTTP responses
loopclosure  check references to loop variables from
within nested functions
lostcancel   check cancel func returned by
context.WithCancel is called
nilfunc      check for useless comparisons between
functions and nil
printf       check consistency of Printf format strings
and arguments
shift        check for shifts that equal or exceed the
width of the integer
stdmethods   check signature of methods of well-known
interfaces
structtag    check that struct field tags conform to
reflect.StructTag.Get
tests        check for common mistaken usages of tests
and examples
unmarshal    report passing non-pointer or non-
interface values to unmarshal
unreachable  check for unreachable code
unsafeptr    check for invalid conversions of uintptr
to unsafe.Pointer
```

```
unusedresult check for unused results of calls to some
functions
```

For details and flags of a particular check, such as printf, run "go tool vet help printf".

By default, all checks are performed. If any flags are explicitly set to true, only those tests are run. Conversely, if any flag is explicitly set to false, only those tests are disabled. Thus -printf=true runs the printf check, and -printf=false runs all checks except the printf check.

For information on writing a new check, see golang.org/x/tools/go/analysis.

Core flags:

```
-c=N
        display offending line plus N lines of
surrounding context
-json
        emit analysis diagnostics (and errors) in JSON
format
```