Building Windows Go programs on Linux

See here for available `GOOS` and `GOARCH` values.

Go version >= 1.5

Since Go version 1.5 cross-compiling of pure Go executables has become very easy. Try it out with the code below. More can be found at this blog post by Dave Cheney.

```
$ cat hello.go
package main

import "fmt"

func main() {
        fmt.Printf("Hello\n")
}
$ GOOS=windows GOARCH=386 go build -o hello.exe hello.go
```

In cmd.exe instead of PowerShell:

```
$ set GOOS=windows
$ set GOARCH=386
$ go build -o hello.exe hello.go
```

You can now run `hello.exe` on a Windows machine near you.

Note that the command above will silently rebuild most of standard library, and for this reason will be quite slow. To speed-up the process, you can install all the windows-amd64 standard packages on your system with

```
GOOS=windows GOARCH=amd64 go install
```

Note also that `cgo` is disabled when cross-compiling, so any file that mentions `import "C"` will be silently ignored (See https://github.com/golang/go/issues/24068). In order to use cgo, or any of the build modes `c-archive`, `c-shared`, `shared`, `plugin`, you need to have a C cross-compiler.

Older Go version (<1.5)

I use linux/386, but, I suspect, this procedure will apply to other host platforms as well.

Preparation (if needed):

```
sudo apt-get install gcc
export go env GOROOT
```

First step is to build host version of go:

```
cd $GOROOT/src
sudo -E GOOS=windows GOARCH=386 PATH=$PATH ./make.bash
```

Next you need to build the rest of go compilers and linkers. I have small program to do that:

```
$ cat ~/bin/buildcmd
#!/bin/sh
set -e
for arch in 8 6; do
    for cmd in a c g l; do
        go tool dist install -v cmd/$arch$cmd
    done
done
exit 0
```

Last step is to build Windows versions of standard commands and libraries. I have a small script for that too:

```
$ cat ~/bin/buildpkg
#!/bin/sh
if [ -z "$1" ]; then
    echo 'GOOS is not specified' 1>&2
    exit 2
else
    export GOOS=$1
    if [ "$GOOS" = "windows" ]; then
        export CGO_ENABLED=0
    fi
fi
shift
if [ -n "$1" ]; then
    export GOARCH=$1
fi
cd $GOROOT/src
go tool dist install -v pkg/runtime
go install -v -a std
```

I run it like that:

```
$ ~/bin/buildpkg windows 386
```

to build Windows/386 version of Go commands and packages. You can probably see from my script that I exclude building of any cgo related parts — these will not work for me, since I do not have correspondent gcc cross-compiling tools installed. So I just skip those.

Now we're ready to build our Windows executable:

```
$ cat hello.go
```

```
package main

import "fmt"

func main() {
        fmt.Printf("Hello\n")
}
$ GOOS=windows GOARCH=386 go build -o hello.exe hello.go
```

We just need to find a Windows computer to run our `hello.exe`.