Introduction

Some libraries—especially graphical frameworks and libraries like Cocoa, OpenGL, and libSDL—use thread-local state and can require functions to be called only from a specific OS thread, typically the 'main' thread. Go provides the `runtime.LockOSThread` function for this, but it's notoriously difficult to use correctly.

Solutions

Russ Cox presented a good solution for this problem in this thread.

```
package sdl

// Arrange that main.main runs on main thread.
func init() {
    runtime.LockOSThread()
}

// Main runs the main SDL service loop.
// The binary's main.main must call sdl.Main() to run this loop.
// Main does not return. If the binary needs to do other work, it
// must do it in separate goroutines.
func Main() {
    for f := range mainfunc {
        f()
    }
}

// queue of work to run in main thread.
var mainfunc = make(chan func())

// do runs f on the main thread.
func do(f func()) {
    done := make(chan bool, 1)
    mainfunc <- func() {
        f()
        done <- true
    }
    <-done
}
```

And then other functions you write in package sdl can be like

```
func Beep() {
    do(func() {
        // whatever must run in main thread
```

```
    })
}
```