Introduction

Sometimes an application needs to save internal state or perform some cleanup activity before it exits, or needs to be able to reload a configuration file or write a memory/cpu profile on demand. In UNIX-like operating systems, signals can accomplish these tasks.

Example

The following code demonstrates a program that waits for an interrupt signal and removes a temporary file when it occurs.

```go
package main

import (
    "io/ioutil"
    "os"
    "os/signal"
)

func main() {
    f, err := ioutil.TempFile("", "test")
    if err != nil {
        panic(err)
    }
    defer os.Remove(f.Name())
    defer f.Close()
    sig := make(chan os.Signal, 1)
    signal.Notify(sig, os.Interrupt)
    <-sig
}
```