Functions and Files

This lesson flashes back to the standard operations and their syntaxes defined on functions and files.

WE'LL COVER THE FOLLOWING

📝 Useful code snippets for functions

Recovering to stop a panic terminating sequence:

```
func protect(g func()) {
  defer func() {
    log.Println("done") // Println executes normally even if there is a pa
nic
    if x := recover(); x != nil {
      log.Printf("run time panic: %v", x)
    }
  }()
  log.Println("start")
  g()
}
```

📝 Useful code snippets for file

Opening and reading a File

```
file, err := os.Open("input.dat")
if err!= nil {
  fmt.Printf("An error occurred on opening the inputfile\n" +
  "Does the file exist?\n" +
```

```
  "Have you got acces to it?\n")
  return

}
defer file.Close()
iReader := bufio.NewReader(file)
for {
  str, err := iReader.ReadString('\n')
  if err!= nil {
    return // error or EOF
  }
  fmt.Printf("The input was: %s", str)
}
```

Copying a file with a buffer

```
func cat(f *file.File) {
  const NBUF = 512
  var buf [NBUF]byte
  for {
    switch nr, er := f.Read(buf[:]); true {
    case nr < 0:
      fmt.Fprintf(os.Stderr, "cat: error reading from %s: %s\n", f.String
(),
      er.String())
      os.Exit(1)
    case nr == 0: // EOF
      return
    case nr > 0:
      if nw, ew := file.Stdout.Write(buf[0:nr]); nw != nr {
        fmt.Fprintf(os.Stderr, "cat: error writing from %s: %s\n", f.Strin
g(),
        ew.String())
      }
    }
  }
}
```

This pretty much summarizes functions and file handling. The next lesson deals with parallelism and networking.