协程出现panic并导致程序退出，如何对野协程进行管理？

Golang 2021.10.23 22:53:33 1813 0

> Go 关键词为启动一个协程，以上代码如果是简单的逻辑没什么问题，如果复杂代码就会出现问题。如果在协程中出现比较复杂的逻辑代码，可能会出现 panic，但是在 Go 语言 main 中无法捕获子协程的 panic，因此导致整个程序退出，也无法捕获到错误。一般称呼以上 Go 代码为野协程。

我们常见的视频流接入协议包括RTSP协议、RTMP协议、GB28181协议三种，针对这三种协议，TSINGSEE青犀视频研发了不同的视频解决方案，其中EasyDSS是支持推流协议RTMP接入的平台，它与其他平台不同的点在于EasyDSS可同时支持视频直播和点播。

在EasyDSS的开发过程中，经常会启动协程来处理一些任务，协程和线程一样共享堆，不共享栈，协程由程序员在协程的代码里显示调度。但是在使用过程中发现部分协程会出现 panic 的情况导致整个程序退出，因此需要对协程进行管理。

一般在使用协程的情况下为以下代码：

```
go func() {
   fmt.Println("Hello World!")
}()
```

Go 关键词为启动一个协程，以上代码如果是简单的逻辑没什么问题，如果复杂代码就会出现问题。如果在协程中出现比较复杂的逻辑代码，可能会出现 panic，但是在 Go 语言 main 中无法捕获子协程的 panic，因此导致整个程序退出，也无法捕获到错误。一般称呼以上 Go 代码为野协程。

因此优化以上代码，捕获子线程中的panic：

```
go func() {
   defer func() {
      if err := recover(); err != nil {
         fmt.Println(fmt.Sprintf("panic %s\n", err))
         fmt.Println(fmt.Sprint(string(debug.Stack())))
      }
   }()

   fmt.Println("Hello World!")
}()
```

但是以上代码会导致大量的 defer func() {} 代码存在，因此进一步优化代码如下：

```
// 管理所有的 go routine
func Go(x func())  {
   go func() {
      defer func() {
         if err := recover(); err != nil {
            fmt.Println(fmt.Sprintf("panic %s\n", err))
            fmt.Println(fmt.Sprint(string(debug.Stack())))
         }
      }()
      x()
   }()
}
```

编写一个 Go 函数，该函数接收的输入为一个无参数函数x，然后函数内部启动一个 go 协程运行 x()，在 defer 中将 panic 信息捕获即可。在其他包中调用如下：

```
Go(func() {
   fmt.Println("Hello World!")
})
```

以上代码比较好的处理了所有的野协程，方便了我们对EasyDSS编译的进一步深入。