Go优雅的退出程序

一、背景

为什么需要优雅退出程序呢？请你仔细设想一下，当我们的程序正在线上运行时，我们的程序需要进行更新，在更新之前需要停掉我们正在运行的这个程序，但是我们运行的程序中可能正在处理我们所设计的程序，例如处理http请求和缓存写入磁盘数据。此时如果我们强制结束程序，可能会造成一些数据丢失，这时我们就需要优雅地退出程序。

二、Linux信号机制(signal)

概念：

信号是发送给进程，或者是进程间通讯的一种手段，采用异步通讯方式。

信号表：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 便携式号码 | 默认操作 | 描述 |
| SIGHUP | 1 | 终止 | 挂断。 |
| SIGINT | 2 | 终止 | 终端中断信号。 |
| SIGQUIT | 3 | 终止（核心转储） | 终端退出信号。 |
| SIGABRT | 6 | 终止（核心转储） | 处理中止信号 |
| SIGKILL | 9 | 终止 | 杀死（不能被发现或忽略）。 |
| SIGALRM | 14 | 终止 | 闹钟 |
| SIGTERM | 15 | 终止 | 终止信号。 |

发起信号：

1、kill命令发出信号，详细请查看kill命令

2、在正在运行的进程的控制终端键入某些组合键会导致系统发送特定的信号，

Ctrl+c发出SIGINT信号

Ctrl+z发出SIGTSTP信号

Ctrl+\发出SIGQUIT信号

三、 Go对信号的处理

Go中对信号的支持：

Go中，os/signal包对信号的支持，

Notify () 接收指定信号，

Ignore () 忽略指定信号，

Reset ()  重置指定信号，

Stop () 停止接收信号，

func Ignore(sig ...os.Signal) 忽略一个、多个或全部（参数为空时，则忽略全部）信号

func Notify(c chan<- os.Signal, sig ...os.Signal) 需要指定监听的信号传入，如果sig为空，则监听全部信号

func Stop(c chan<- os.Signal) 停止监听所有信号

func Reset(sig ...os.Signal) 取消之前使用 Notify 对信号产生的效果；如果没有参数，则所有信号处理都被重置。

注意：

1、SIGKILL和SIGSTOP这两个信号是不会被应用程序所捕获。

2、kill pid命令默认 发送的信号为 SIGTREM kill -9 发送的信号为 SIGKILL

kill pid的作用是向进程号为pid的进程发送SIGTERM（这是kill默认发送的信号），**该信号是一个结束进程的信号且可以被应用程序捕获**。若应用程序没有捕获并响应该信号的逻辑代码，则该信号的默认动作是kill掉进程。这是终止指定进程的推荐做法。

**kill -9 pid则是向进程号为pid的进程发送SIGKILL**（该信号的编号为9），从本文上面的说明可知，**SIGKILL既不能被应用程序捕获，也不能被阻塞或忽略，其动作是立即结束指定进程**。

kill -9命令事实上，是发送给systemd进程的（pid为1的），systemd进程收到该信号后，直接终止指定的进程。

示例：

