

# 作业 2: Shell 脚本与信号

李志阳

混合班 3190105480

2022 年 6 月 28 日

## 1 Linux 的信号

我们常常使用 Ctrl+C 来终止程序的运行, 那么这个 Ctrl+C 是什么呢? 它工作的原理又是什么呢? 其实, 在 Linux 中, 这样的东西被称为**信号** (Signal).

信号是一种进程间通信的方式, 既然是通信, 也就是说会有一个程序负责发送信号, 另一个程序负责接收。这个发送的程序可以是任意运行中的程序, 包括终端。当按下 Ctrl+C 时, 就是由终端向我们的程序发送一个 SIGINT 信号, 在没有特定接收信号的动作的情况下, 程序接收到信号之后都会有默认的行为, 比如接收到 Ctrl+C 时, 默认为终止程序。类似的信号还包括 SIGKILL(用于杀死进程), SIGCHLD(用于标志子进程的结束) 等等.

下面是一张常见信号及其默认行为的列表:

信号名称	信号数值	默认动作	描述
SIGHUP	1	终止进程	终端连接结束时发出。终端连接断开, 会向当前终端连接会话关联的所有前台和后台进程组发送 SIGHUP 信号, 用于终止进程。
SIGINT	2	终止进程	程序终止 (interrupt) 信号, 通常是 Ctrl+C 发出。
SIGQUIT	3	终止进程	和 SIGINT 类似, 通常是 Ctrl+/发出。进程在收到 SIGQUIT 信号退出时会产生 core 文件, 在这个意义上类似于一个程序错误信号。

SIGFPE	8	终止进程，建立 CORE 文件	在发生致命的算术运算错误 (Floating-Point Exception) 时发出，不仅包括浮点运算错误，还包括溢出及除数为 0 等其它所有的算术错误。
SIGKILL	9	终止进程	用来立即结束程序的运行。本信号不能被阻塞，处理和忽略。
SIGSEGV	11	终止进程，建立 CORE 文件	段错误 (Segmentation Fault) 信号。进程试图访问非法内存地址，如往没有写权限的内存地址写数据时会触发段错误。
SIGALRM	14	终止进程	时钟定时信号，计时器到时会发出该信号。alarm() 函数使用该信号。
SIGTERM	15	终止进程	程序结束 (Terminate) 信号，与 SIGKILL 不同的是该信号可以被阻塞和处理。通常用来要求程序自己正常退出。Shell 命令 kill 缺省产生这个信号。
SIGCHLD	17	忽略信号	子进程结束时，父进程会收到这个信号

发送信号的方式也有很多，比如上面提到的使用 Ctrl+C 向运行中的进程发送 SIGINT 信号。常见的 kill 命令也可以用于发送信号 (In fact, kill 命令本来就是用来发信号的，而不是终止进程)。其用法为：

```
kill [-< 信号名称或编号 >][程序]
```

其中，程序可以是进程号，也可以是作业号。-s 后面的参数用来表示要发送的信号，使用时用上表中的信号名称去掉 SIG 表示，也可以由信号数值表示。比如：

```
kill -KILL 123456
```

表示向 PID 为 123456 的进程发送 SIGKILL 信号，同样的，这条命令也可以写为：

```
kill -9 123456
```

## 2 在 shell 中处理信号

### 2.1 trap 命令

在 shell 中处理信号，使用 trap 命令。trap 命令有如下两种用法：

- `trap 'commands' signal-list`  
或 `trap "commands" signal-list`

这种形式用于改变程序接收到信号的行为, 接收到`signal-list`中的信号 (同样用上表中的名字去掉 SIG 或序号表示) 时, 将执行的默认动作改为`commands`中的命令, 如:

```
trap "echo 'trapped';" INT
```

表示接收到 SIGINT 信号, 不再终止运行, 而是输出 `trapped`. 当 `commands` 为空时, 不执行任何命令.

- `trap signal-list`

这种情况表示恢复该信号的默认操作.

## 2.2 try-out

书上提供如下例子 (P60):

```
trap 'rm -f /tmp/my_tmp_file_$$' INT
echo creating file /tmp/my_tmp_file_$$
date > /tmp/my_tmp_file_$$
echo "press interrupt (CTRL-C) to interrupt ...."
while [ -f /tmp/my_tmp_file_$$ ]; do
    echo File exists
    sleep 1
done
echo The file no longer exists
trap INT
echo creating file /tmp/my_tmp_file_$$
date > /tmp/my_tmp_file_$$
echo "press interrupt (control-C) to interrupt ...."
while [ -f /tmp/my_tmp_file_$$ ]; do
    echo File exists
    sleep 1
done
echo we never get here
exit 0
```

这段代码中, 先使用当前 PID 确定一个文件名, 将日期写入这个文件中, 然后不断循环输出 File exists.

当按下 Ctrl+C 时, 程序接收到信号 SIGINT. 因为在第一行使用 `trap` 将 SIGINT 的动作变更为了删除这一文件, 因此这时程序将执行删除而非退出. 这时循环条件已不满足 (因为循环条件是文件存在时继续), 脚本退出循环.

继续执行 `echo The file no longer exists`. 而后将 SIGINT 的处理恢复默认并重新创建文件, 然后进入新的循环, 不断输出 File exists. 在这时按下 Ctrl+C, 则会执行默认动作退出程序, 否则会一直执行这一循环. 因此后面的 echo 是不可到达的.

这里是一张执行截图:

```
creating file /tmp/my_tmp_file_258
press interrupt (CTRL-C) to interrupt ....
File exists
File exists
File exists
^CThe file no longer exists
creating file /tmp/my_tmp_file_258
press interrupt (control-C) to interrupt ....
File exists
File exists
File exists
^C
```

图 1: 运行结果 1

在 `^C` 处按下了 Ctrl+C, 可以看到第一次按下时并没有终止执行. 第二次按下时才退出. 运行结束时查看 /tmp, 发现文件仍然存在:

```
misaka@Misakalzy:/mnt/d/Desktop/Micheer_Saan/MathSoft$ ls /tmp
my_tmp_file_219  my_tmp_file_258  tmp.Ho9YrdIY2v  tmp05s8lc5q
mv tmp file 226  tmp.DRp3771lmr  tmp.m03WhXSE88  vscode-typscript1000
```

图 2: 查看/tmp

于是, 产生了一个有趣的想法, 在按下第一次 Ctrl+C 后手动删掉文件会如何? 按理说应该会循环条件不满足退出, 下面进行验证:

```
^CThe file no longer exists
creating file /tmp/my_tmp_file_339
press interrupt (control-C) to interrupt ....
File exists
File exists
File exists
We never get here
```

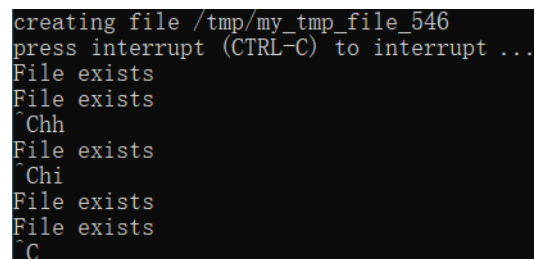
图 3: 运行结果 2

确实输出了 We never get here.

更进一步, 我们可以尝试 trap 命令嵌套使用, 比如:

```
trap 'echo "hh" && trap "echo hi && trap INT" INT' INT
echo creating file /tmp/my_tmp_file_$$
date > /tmp/my_tmp_file_$$
echo "press interrupt (CTRL-C) to interrupt ...."
while [ -f /tmp/my_tmp_file_$$ ]; do
    echo File exists
    sleep 1
done
echo Strange
exit 0
```

第一次 trap 时, 将 INT 的操作改为了输出 hh 并执行第二个 trap, 第二次 trap 时, 将 INT 的操作改为输出 hi 并恢复默认操作. 因此我们第一次 Ctrl+C 时, 输出 hh, 第二次输出 hi, 第三次退出. 验证:



```
creating file /tmp/my_tmp_file_546
press interrupt (CTRL-C) to interrupt ...
File exists
File exists
^Chh
File exists
^Chi
File exists
File exists
^C
```

图 4: 运行结果 3