作业二: 尝试编写 *Shell* 脚本
实现 *Shell* 进程中断时的自定义行为

杨钧尹
信息与计算科学 3200103573

2022 年 6 月 28 日

1 情境介绍

在日常使用 *Linux* 并通过终端执行某些命令或脚本时，我们难免会遇到中断该进程的情况。譬如，利用脚本测试某简单的可执行程序时惊觉陷入了死循环，或因等待时间过长而临时取消。我们一般通过 $Ctrl + C$ 快捷键向系统发送中断信号 ($INT\ Signal$)。

然而很多时候，我们并不希望进程就此立即中止：某些脚本或许不希望在执行时被用户主动或意外打断，而某些脚本则在退出前期望执行一些类似于释放临时空间的额外的操作。

基于「*Beginning Linux Programming*」的学习，本文试图利用 *trap* 命令应对这些情况。简便起见，此处复现了 $Try\ It\ Out : Trapping\ Signals$ 中的简单范例。示例脚本通过精巧的设计说明了 *trap* 命令的运作模式。

2 实现过程

Shell 脚本参见附录，以下是该脚本的基本流程：

在最开始，我们使用 *trap* 命令，使得在后续接受到中断指令 INT 时，系统根目录的 *tmp* 文件夹中的所有日志文件将被删除。这些文件以 my_tmp_file 为前缀，记录了生成时的系统日期与时间信息。

紧接于此，我们使用 *echo* 与 *date* 命令，在执行生成日志文件操作时告知用户这一行为并提醒用户可以通过组合键打断本进程。

利用 *while* 循环，我们不断检查该目录下是否存在前述日志文件。如果存在，则告知用户；如果已不存在，则结束循环。其中，我们使用 *sleep* 命令在每次循环中设置了 1 秒钟的休眠期，使得循环以适合人类用户感官的方式进行。

在此之后，我们设置了第二个 *trap* 命令语句，使得第二次打断操作也能被拦截，并生成新的日志文件。同第一次一样，我们设置了检查日志文件存在性的循环。

实际运行时，在用户第一次打断进程前，脚本由已经在对应目录生成了日志文件，且处于文件存在的第一次循环之中。由于打断操作，文件被第一条 *trap* 语句删除，第一次循环跳出。此时用户会收到“文件已不复存在”的信息，但该脚本仍然正常进行中。

同理，由于再次生成了日志文件，在第二次循环中，直至用户打断进程，脚本才会结束。由于第二条 *trap* 语句没有删除日志文件等额外执行语句，因此打断在循环中被顺利结束。脚本末尾位于循环外的 *echo* 命令不可能被执行，即用户无法看到最后的信息。

总之，通过运行时终端中呈现的反馈和脚本所生成的日志文件的时间信息，可以明晰 *trap* 命令是如何运作的。

在脚本所在目录下输入命令

```
bash script
```

在两次“文件存在”的循环中分别按下 $Ctrl + C$ 快捷键，直至脚本结束，运行结果如下：

```
生成文件 /tmp/my_tmp_file_79050
使用组合键CTRL-C以打断进程
文件存在
文件存在
文件存在
文件存在
文件存在
^C文件已不复存在
生成下列文件 /tmp/my_tmp_file_79050
使用组合键CTRL-C以打断进程
文件存在
```

```
文件存在
文件存在
文件存在
^C
```

需要说明的是，由于脚本是自上而下解释的，我们必须在需要保护的内容之前指定好对应的 *trap* 命令。此外，记住课本 $P59$ 中的信号表格也很重要，中断信号 INT 只是 *trap* 可以理解的其中一种而已，我们可以利用它实现更加高阶的需求。

附录：脚本源代码

```
#!/bin/bash
#2022-06-28@Ralvine

trap 'rm -f /tmp/my_tmp_file_$$' INT
echo "生成文件" /tmp/my_tmp_file_$$
date > /tmp/my_tmp_file_$$

echo "使用组合键CTRL-C以打断进程"
while [ -f /tmp/my_tmp_file_$$ ]; do
echo "文件存在"
sleep 1
done
echo "文件已不复存在"

trap INT
echo "生成下列文件" /tmp/my_tmp_file_$$
date > /tmp/my_tmp_file_$$

echo "使用组合键CTRL-C以打断进程"
while [ -f /tmp/my_tmp_file_$$ ]; do
echo "文件存在"
sleep 1
```

```
done

echo "运行中不可能显示此条信息"
exit 0
```