MyShell操作手册

1.背景

MyShell是自行实现的一个Linux shell, 具备基本的shell功能

TestShell是用来做验证用的测试程序,源代码在Test Shell.c

实验环境:

- 0. 本地环境Window10
- 1. VSCode (工作在本地)
- 2. C语言
- 3. 使用VSCode远程连接华为云服务器
- 4. gcc version 7.4.0 (Ubuntu/Linaro 7.4.0-1ubuntu1~18.04.1)

2.操作步骤

首先编译MyShell和TestShell的源码正常情况下不会出现任何警告或错误

```
gcc -o MyShell MyShell.c -lpthread
gcc -o TestShell TestShell.c
```

得到两个可执行程序。

直接运行MyShell即可,会看到如下的提示符,这个时候可以输入指令。这里提醒一下,本程序没有对换行等操作进行处理,所以执行程序的时候,如果不输入指令直接按下回车键会导致程序异常终止。

```
1 | MyShell>
```

接下来开始测试

1. 首先测试常见的一些指令

2. 然后使用测试程序TestShell进行测试,测试程序的主要目的是为了测试后台运行情况。

这里提前说明: result.txt原本没有内容, TestShell的功能是每隔两秒钟向result.txt中追加一行"Hello", 在本文的环境下, 用VSCode打开result.txt可以看到其变化情况, 更方便观察实验效果。

首先是让其进行后台执行,使用 jobs 内置命令可以看出它在执行 而且状态为 2 即后台执行,等 result.txt中出现20行"Hello"后在用 jobs 命令查看,发现这个后台任务已经结束并已经从任务列表中删除

```
1 MyShell> ./TestShell &
  [5] (13354) 2 ./TestShell &
2
4 | MyShell> jobs
5
  PID JID status command
6 13354 5 2 ./TestShell &
7
8
   0 0 0
9
    0 0
           0
10
   0 0
           0
11
    0 0
           0
           0
12
    0 0
13
   0 0
    0 0
14
           0
15
   0 0
           0
16
    0 0
           0
           0
17
   0 0
   0 0
           0
18
    0 0
           0
19
   0 0
           0
20
21
    0 0
           0
   0 0 0
22
23 MyShell> jobs
  MyShell> PID JID status command
   0 0
25
           0
    0 0
26
           0
           0
   0 0
27
28
   0 0
           0
29
   0 0
           0
30
   0 0
           0
    0 0
           0
31
           0
   0 0
32
33
    0 0
           0
           0
34
    0 0
35
    0 0
    0 0
36
           0
   0 0
           0
37
    0 0
38
           0
    0 0
           0
39
40
41 | MyShell>
```

3. 接下来测试信号(Ctrl z),为了方便起见,先将result.txt中的内容清空。开始执行TestShell的瞬间按下Ctrl z,该任务就会被挂起到后台,状态为3即stopped,并且此时的result.txt没有任何变化(如果按下Ctrl z足够及时那么result.txt应该为空)

```
1 MyShell> ./TestShell
2 ^Z
```

```
Tips: child 13359 stopped by signal 20
 5
   MyShell> jobs
   PID JID status command
 7 13359 6 3 ./TestShell
 8
     0 0 0
0 0 0
 9
10
11
     0 0
             0
     0 0 0
0 0 0
12
13
14
     0 0
             0
             0
15
     0 0
16
     0 0
             0
    0 0 0
0 0 0
17
18
19
     0 0
             0
20
     0 0
             0
21
     0 0
             0
0 0 0
23 0 0 0
24 MyShell>
```

```
CSAPP > myshell > ≡ result.txt
问题 輸出 调试控制台 终端
MyShell> ./TestShell
^Z
   Tips: child 13359 stopped by signal 20
MyShell> jobs
 PID JID status command
          3 ./TestShell
13359
      6
   0 0
           0
   0 0
            0
   0 0
            0
   0 0
            0
   0 0
           0
   0 0
           0
   0 0
            0
   0 0
            0
   0 0
            0
   0 0
            0
           0
   0 0
           0
   0 0
            0
   0
     0
             0
   0
      0
             0
MyShell>
```

4. 测试 bg 指令,这个指令是将后台挂起进程继续后台执行,可以看到程序已经开始后台执行,而且 状态也发生了改变。待TestShell执行完成,再次查看jobs,发现该任务已经被清理

```
1  MyShell> bg %6
2  [6] (13359) 2 ./TestShell
3
4  MyShell> jobs
5  PID JID status command
6  13359  6   2 ./TestShell
7
8  0  0  0
```

9	0	0	0		
10	0	0	0		
11	0	0	0		
12	0	0	0		
13	0	0	0		
14	0	0	0		
15	0	0	0		
16	0	0	0		
17	0	0	0		
18	0	0	0		
19	0	0	0		
20	0	0	0		
21	0	0	0		
22	0	0	0		
23	MyShell>				
24	MyShell>		JID	status	command
25	0	0	0		
26	0	0	0		
27	0	0	0		
28	0	0	0		
29	0	0	0		
30	0	0	0		
31	0	0	0		
32	0	0	0		
33	0	0	0		
34	0	0	0		
35	0	0	0		
36	0	0	0		
37	0	0	0		
38	0	0	0		
39	0	0	0		
40	0	0	0		
41	MyShell>				

```
CSAPP > myshell > 📱 result.txt
    Hello
 2 Hello
    Hello
 4 Hello
 5 Hello
 6 Hello
 7 Hello
 8 Hello
 9 Hello
问题 輸出 调试控制台 终端
  0 0
           0
MyShell> bg %6
[6] (13359) 2 ./TestShell
MyShell> jobs
 PID JID status command
13359 6
         2 ./TestShell
  0 0
          0
  0 0
           0
     0
           0
  0
  0
     0
           0
          0
  0 0
  0 0
           0
  0
     0
           0
    0
  0
           0
  0 0
          0
  0 0
           0
     0
  0
          0
  0 0
          0
  0 0
           0
  0 0
            0
      0
            0
  0
MyShell>
```

5. 测试 fg 指令,首先还是先清空result.txt,然后重新执行第三步操作,可以发现任务再次被挂起。 fg 指令是让后台挂起任务放到前台执行,由于shell只允许最多一个前台任务,所以这个时候其他 的指令都不能被执行。(最后一行的 jobs 命令不会被立刻执行,但是当TestShell执行完成后就会 执行)

```
MyShell> ./TestShell

// Z

Tips: child 13368 stopped by signal 20

MyShell> jobs
PID JID status command
```

7	13368	7	3	./TestShell
8				,
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	
15	0	0	0	
16	0	0	0	
17	0	0	0	
18	0	0	0	
19	0	0	0	
20	0	0	0	
21	0	0	0	
22	0	0	0	
23	0	0	0	
24	MyShell>	fg %7		
25	jobs			

```
1 Hello
 2 Hello
 3 Hello
 4 Hello
 5 Hello
 6 Hello
问题 輸出 调试控制台 终端
  Tips: child 13368 stopped by signal 20
MyShell> jobs
 PID JID status command
13368
        3 ./TestShell
         0
  0 0
  0 0
          0
  0
     0
          0
  0 0
          0
  0 0
          0
  0 0
          0
  0
    0
        0
0
          0
  0 0
  0 0
         0
  0 0
          0
        0
  0 0
  0 0
  0 0
         0
          0
  0
     0
          0
MyShell> fg %7
jobs
```

6. 最后测试 quit 命令,在上一步已经执行完的基础上执行,程序正常退出

```
1 | MyShell> quit
2 | root@----:~/Code/CSAPP/myshell#
```

3.结语

自我反思一下,整个MyShell.c源码有500行左右,明明只是信号控制一下能解决的问题,偏偏花了我好几天的时间,基本功还是不行。但是,从此之后我不再称自己是"菜鸡",要不然每天晚上敲代码的时候一想到"菜鸡"就饿的不行。

最后,程序只是简单的测试了一下常规的操作,还有一些问题没有解决,但是整体目标已经达到了,后面还有很多东西要学,不能长时间耽误在这个地方,所以就先这样吧,以后有时间了再折腾。

--by Panda