<https://blog.csdn.net/cheyo/article/details/6595955>

例：

**[cpp]** [view plain](https://blog.csdn.net/cheyo/article/details/6595955) [copy](https://blog.csdn.net/cheyo/article/details/6595955)

1. status = system("./test.sh");

1、先统一两个说法：

（1）system返回值：指调用system函数后的返回值，比如上例中status为system返回值

（2）shell返回值：指system所调用的shell命令的返回值，比如上例中，test.sh中返回的值为shell返回值。

2、如何正确判断test.sh是否正确执行？

仅判断status是否==0？或者仅判断status是否!=-1?

都错！

3、man中对于system的说明

RETURN VALUE  
       The value returned is -1 on error (e.g.  fork() failed), and the return  
       status  of  the command otherwise.  This latter return status is in the  
       format specified in wait(2).  Thus, the exit code of the  command  will  
       be  WEXITSTATUS(status).   In  case  /bin/sh could not be executed, the  
       exit status will be that of a command that does exit(127).

看得很晕吧？

system函数对返回值的处理，涉及3个阶段：

阶段1：创建子进程等准备工作。如果失败，返回-1。

阶段2：调用/bin/sh拉起shell脚本，如果拉起失败或者shell未正常执行结束（参见备注1），原因值被写入到status的低8~15比特位中。system的man中只说明了会写了127这个值，但实测发现还会写126等值。

阶段3：如果shell脚本正常执行结束，将shell返回值填到status的低8~15比特位中。

备注1：

只要能够调用到/bin/sh，并且执行shell过程中没有被其他信号异常中断，都算正常结束。

比如：不管shell脚本中返回什么原因值，是0还是非0，都算正常执行结束。即使shell脚本不存在或没有执行权限，也都算正常执行结束。

如果shell脚本执行过程中被强制kill掉等情况则算异常结束。

如何判断阶段2中，shell脚本是否正常执行结束呢？系统提供了宏：WIFEXITED(status)。如果WIFEXITED(status)为真，则说明正常结束。

如何取得阶段3中的shell返回值？你可以直接通过右移8bit来实现，但安全的做法是使用系统提供的宏：WEXITSTATUS(status)。

由于我们一般在shell脚本中会通过返回值判断本脚本是否正常执行，如果成功返回0，失败返回正数。

所以综上，判断一个system函数调用shell脚本是否正常结束的方法应该是如下3个条件同时成立：

（1）-1 != status

（2）WIFEXITED(status)为真

（3）0 == WEXITSTATUS(status)

注意：

根据以上分析，当shell脚本不存在、没有执行权限等场景下时，以上前2个条件仍会成立，此时WEXITSTATUS(status)为127，126等数值。

所以，我们在shell脚本中不能将127，126等数值定义为返回值，否则无法区分中是shell的返回值，还是调用shell脚本异常的原因值。shell脚本中的返回值最好多1开始递增。

判断shell脚本正常执行结束的健全代码如下：

**[cpp]** [view plain](https://blog.csdn.net/cheyo/article/details/6595955) [copy](https://blog.csdn.net/cheyo/article/details/6595955)

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <sys/wait.h>
4. #include <sys/types.h>
6. **int** main()
7. {
8. pid\_t status;

11. status = system("./test.sh");
13. **if** (-1 == status)
14. {
15. printf("system error!");
16. }
17. **else**
18. {
19. printf("exit status value = [0x%x]\n", status);
21. **if** (WIFEXITED(status))
22. {
23. **if** (0 == WEXITSTATUS(status))
24. {
25. printf("run shell script successfully.\n");
26. }
27. **else**
28. {
29. printf("run shell script fail, script exit code: %d\n", WEXITSTATUS(status));
30. }
31. }
32. **else**
33. {
34. printf("exit status = [%d]\n", WEXITSTATUS(status));
35. }
36. }
38. **return** 0;
39. }

WIFEXITED(stat\_val) Evaluates to a non-zero value if status  
was returned for a child process that  
terminated normally.  
  
WEXITSTATUS(stat\_val) If the value of WIFEXITED(stat\_val) is  
non-zero, this macro evaluates to the  
low-order 8 bits of the status argument  
that the child process passed to \_exit()  
or exit(), or the value the child  
process returned from main().