<https://blog.csdn.net/macrohasdefined/article/details/12713567>

# 【C/C++】Linux下使用system()函数一定要谨慎

15人收藏此文章,我要收藏 发表于1年前(2012-04-15 00:35) , 已有**14760**次阅读 ，共[**3**](http://my.oschina.net/renhc/blog/53580#comments)个评论

曾经的曾经，被system()函数折磨过，之所以这样，是因为对system()函数了解不够深入。只是简单的知道用这个函数执行一个系统命令，这远远不够，它的返回值、它所执行命令的返回值以及命令执行失败原因如何定位，这才是重点。当初因为这个函数风险较多，故抛弃不用，改用其他的方法。这里先不说我用了什么方法，这里必须要搞懂system()函数，因为还是有很多人用了system()函数，有时你不得不面对它。

**先来看一下system()函数的简单介绍：**

|  |  |
| --- | --- |
| 1 | #include <stdlib.h> |
| 2 | int system(const char \*command); |

system() executes a command specified in command by calling /bin/sh -c command, and returns after the command has been completed. During execution of the command, SIGCHLD will be blocked, and SIGINT and SIGQUIT will be ignored.

system()函数调用/bin/sh来执行参数指定的命令，/bin/sh 一般是一个软连接，指向某个具体的shell，比如bash，-c选项是告诉shell从字符串command中读取命令；

在该command执行期间，SIGCHLD是被阻塞的，好比在说：hi，内核，这会不要给我送SIGCHLD信号，等我忙完再说；

在该command执行期间，SIGINT和SIGQUIT是被忽略的，意思是进程收到这两个信号后没有任何动作。

**再来看一下system()函数返回值：**

The value returned is -1 on error (e.g. fork(2) 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).

If the value of command is NULL, system() returns nonzero if the shell is available, and zero if not.

为了更好的理解system()函数返回值，需要了解其执行过程，实际上system()函数执行了三步操作：

1.fork一个子进程；

2.在子进程中调用exec函数去执行command；

3.在父进程中调用wait去等待子进程结束。

对于fork失败，system()函数返回-1。

如果exec执行成功，也即command顺利执行完毕，则返回command通过exit或return返回的值。

（注意，command顺利执行不代表执行成功，比如command："rm debuglog.txt"，不管文件存不存在该command都顺利执行了）

如果exec执行失败，也即command没有顺利执行，比如被信号中断，或者command命令根本不存在，system()函数返回127.

如果command为NULL，则system()函数返回非0值，一般为1.

**看一下system()函数的源码**

看完这些，我想肯定有人对system()函数返回值还是不清楚，看源码最清楚，下面给出一个system()函数的实现：

|  |  |
| --- | --- |
| 01 | int system(const char \* cmdstring) |
| 02 | { |

|  |  |
| --- | --- |
| 03 | pid\_t pid; |
| 04 | int status; |

|  |  |
| --- | --- |
| 05 |  |
| 06 | if(cmdstring == NULL) |

|  |  |
| --- | --- |
| 07 | { |
| 08 | return (1); //如果cmdstring为空，返回非零值，一般为1 |

|  |  |
| --- | --- |
| 09 | } |
| 10 |  |

|  |  |
| --- | --- |
| 11 | if((pid = fork())<0) |
| 12 | { |

|  |  |
| --- | --- |
| 13 | status = -1; //fork失败，返回-1 |
| 14 | } |

|  |  |
| --- | --- |
| 15 | else if(pid == 0) |
| 16 | { |

|  |  |
| --- | --- |
| 17 | execl("/bin/sh", "sh", "-c", cmdstring, (char \*)0); |
| 18 | \_exit(127); // exec执行失败返回127，注意exec只在失败时才返回现在的进程，成功的话现在的进程就不存在啦~~ |

|  |  |
| --- | --- |
| 19 | } |
| 20 | else //父进程 |

|  |  |
| --- | --- |
| 21 | { |
| 22 | while(waitpid(pid, &status, 0) < 0) |

|  |  |
| --- | --- |
| 23 | { |
| 24 | if(errno != EINTR) |

|  |  |
| --- | --- |
| 25 | { |
| 26 | status = -1; //如果waitpid被信号中断，则返回-1 |

|  |  |
| --- | --- |
| 27 | break; |
| 28 | } |

|  |  |
| --- | --- |
| 29 | } |
| 30 | } |

|  |  |
| --- | --- |
| 31 |  |
| 32 | return status; //如果waitpid成功，则返回子进程的返回状态 |

|  |  |
| --- | --- |
| 33 | } |

仔细看完这个system()函数的简单实现，那么该函数的返回值就清晰了吧，那么什么时候system()函数返回0呢？只在command命令返回0时。

**看一下该怎么监控system()函数执行状态**

这里给我出的做法：

|  |  |
| --- | --- |
| 01 | int status; |
| 02 | if(NULL == cmdstring) //如果cmdstring为空趁早闪退吧，尽管system()函数也能处理空指针 |

|  |  |
| --- | --- |
| 03 | { |
| 04 | return XXX; |

|  |  |
| --- | --- |
| 05 | } |
| 06 | status = system(cmdstring); |

|  |  |
| --- | --- |
| 07 | if(status < 0) |
| 08 | { |

|  |  |
| --- | --- |
| 09 | printf("cmd: %s\t error: %s", cmdstring, strerror(errno)); // 这里务必要把errno信息输出或记入Log |
| 10 | return XXX; |

|  |  |
| --- | --- |
| 11 | } |
| 12 |  |

|  |  |
| --- | --- |
| 13 | if(WIFEXITED(status)) |
| 14 | { |

|  |  |
| --- | --- |
| 15 | printf("normal termination, exit status = %d\n", WEXITSTATUS(status)); //取得cmdstring执行结果 |
| 16 | } |

|  |  |
| --- | --- |
| 17 | else if(WIFSIGNALED(status)) |
| 18 | { |

|  |  |
| --- | --- |
| 19 | printf("abnormal termination,signal number =%d\n", WTERMSIG(status)); //如果cmdstring被信号中断，取得信号值 |
| 20 | } |

|  |  |
| --- | --- |
| 21 | else if(WIFSTOPPED(status)) |
| 22 | { |

|  |  |
| --- | --- |
| 23 | printf("process stopped, signal number =%d\n", WSTOPSIG(status)); //如果cmdstring被信号暂停执行，取得信号值 |
| 24 | } |

到于取得子进程返回值的相关介绍可以参考另一篇文章：<http://my.oschina.net/renhc/blog/35116>

system()函数用起来很容易出错，返回值太多，而且返回值很容易跟command的返回值混淆。这里推荐使用popen()函数替代，关于popen()函数的简单使用也可以通过上面的链接查看。

popen()函数较于system()函数的优势在于使用简单，popen()函数只返回两个值：  
成功返回子进程的status，使用WIFEXITED相关宏就可以取得command的返回结果；  
失败返回-1，我们可以使用perro()函数或strerror()函数得到有用的错误信息。

这篇文章只涉及了system()函数的简单使用，还没有谈及SIGCHLD、SIGINT和SIGQUIT对system()函数的影响，事实上，之所以今天写这篇文章，是因为项目中因有人使用了system()函数而造成了很严重的事故。现像是system()函数执行时会产生一个错误：“No child processes”。

关于这个错误的分析，感兴趣的朋友可以看一下：<http://my.oschina.net/renhc/blog/54582>

2012-04-14 qdurenhongcai@163.com

转载请注明出处。