<https://blog.csdn.net/u010155023/article/details/51984602>

本来想写下自己的东西，但看到有一篇文章写得很好，于是就不自己写了。

出处：https://typecodes.com/linux/linuxalarmknockfunc.html

**1. alarm函数**

**[cpp]** [view plain](https://blog.csdn.net/u010155023/article/details/51984602) [copy](https://blog.csdn.net/u010155023/article/details/51984602)

1. [1] 引用头文件：#include <unistd.h>;
3. [2] 函数标准式：unsigned **int** alarm(unsigned **int** seconds);
5. [3] 功能与作用：alarm()函数的主要功能是设置信号传送闹钟，即用来设置信号SIGALRM在经过参数seconds秒数后发送给目前的进程。如果未设置信号SIGALARM的处理函数，那么alarm()默认处理终止进程。
7. [4] 函数返回值：如果在seconds秒内再次调用了alarm函数设置了新的闹钟，则后面定时器的设置将覆盖前面的设置，即之前设置的秒数被新的闹钟时间取代；当参数seconds为0时，之前设置的定时器闹钟将被取消，并将剩下的时间返回。

**2. 测试**

**原文的测试环境是RedHat Linux5.3，本人在Ubuntu 14.04中再次进行了测试。测试结果与原作者一致。**

了解了alarm()函数的功能特性和返回值的特性后，我们就可以对其测试。测试方向有两个：其一，测试常规只单独存在一个闹钟函数alarm()的程序；其二，测试程序中包含多个alarm()闹钟函数。因此整理了下面几个程序，通过比较学习更有助于理解。测试环境是RedHat Linux5.3，GCC编译调试。

##### 2.1 alarm()测试1.1

**[cpp]** [view plain](https://blog.csdn.net/u010155023/article/details/51984602) [copy](https://blog.csdn.net/u010155023/article/details/51984602)

1. #include <unistd.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <signal.h>
6. **void** sig\_alarm()
7. {
8. exit(0);
9. }
10. **int** main(**int** argc, **char** \*argv[])
11. {
12. signal(SIGALRM, sig\_alarm);
13. alarm(10);
14. sleep(15);
15. printf("Hello World!\n");
16. **return** 0;
17. }

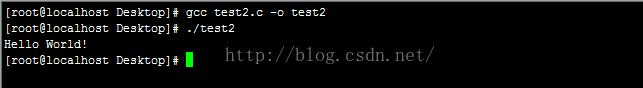


程序分析：在文件test1.c中，定义了一个时钟alarm(10)，它的作用是让信号SIGALRM在经过10秒后传送给目前main()所在进程；接着又定义了sleep(15)，它的作用是让执行挂起15秒的时间。所以当main()程序挂起10秒钟时，signal函数调用SIGALRM信号的处理函数sig\_alarm，并且sig\_alarm执行exit(0)使得程序直接退出。因此，printf("Hello World!\n")语句是没有被执行的。

##### 2.2 alarm()测试1.2

**[cpp]** [view plain](https://blog.csdn.net/u010155023/article/details/51984602) [copy](https://blog.csdn.net/u010155023/article/details/51984602)

1. #include <unistd.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <signal.h>
6. **void** sig\_alarm()
7. {
8. exit(0);
9. }
10. **int** main(**int** argc, **char** \*argv[])
11. {
12. signal(SIGALRM, sig\_alarm);
13. alarm(10);
14. sleep(5);
15. printf("Hello World!\n");
16. **return** 0;
17. }

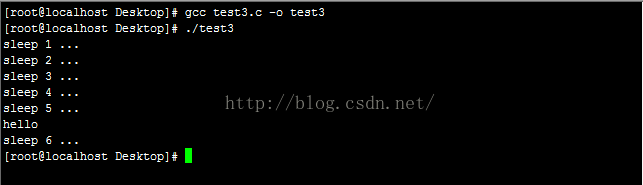


程序分析：与test1.c文件不同的是，在文件test2.c中延时函数为sleep(5)，即执行挂起5秒的时间。所以当main()程序挂起5秒钟时，由于还没到达设置的闹钟10秒，那么main就执行下面的printf("Hello World!\n")语句；紧接着又执行下面的return 0语句，从而直接退出程序。因此，整个test2.c文件输出的内容为：Hello World!。

##### 2.3 alarm()测试2

**[cpp]** [view plain](https://blog.csdn.net/u010155023/article/details/51984602) [copy](https://blog.csdn.net/u010155023/article/details/51984602)

1. #include <unistd.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <signal.h>
6. **void** handler()
7. {
8. printf("hello\n");
9. }
11. **void** main()
12. {
13. **int** i;
14. signal(SIGALRM, handler);
15. alarm(5);
16. **for**(i = 1; i < 7; i++)
17. {
18. printf("sleep %d ...\n", i);
19. sleep(1);
20. }
21. }



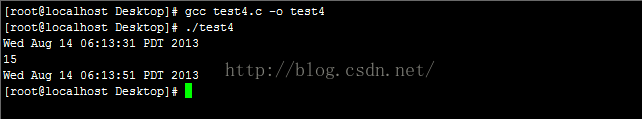
程序分析：在文件test3.c中，定义时钟alarm(5)，而main()函数中主要是一个for循环输出语句。当main函数执行到i=5时，for循环先执行printf("sleep %d ...\n", 5)语句输出"sleep 5 ..."，然后执行sleep(1)语句。此时已经到达闹钟时间5秒，因此会把信号SIGALRM传送给当前main()函数进程；接着调用SIGALRM信号的处理函数handler，从而输出"hello"，然后又返回到sleep(1)这个点；最后for循环执行i=6，输出"sleep 6"，最终延时1秒后结束整个程序。

以上三个程序都只包含一个alarm()闹钟函数，下面两个程序包含两个alarm()。并且为了更为真切的观察包含alarm()闹钟函数的程序的执行过程，程序通过调用系统打印输出当前时间，通过时间差来进一步理解。

##### 2.4 alarm()测试3.1

**[cpp]** [view plain](https://blog.csdn.net/u010155023/article/details/51984602) [copy](https://blog.csdn.net/u010155023/article/details/51984602)

1. #include <unistd.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <signal.h>
6. **static** **void** sig\_alrm(**int** signo);
7. **int** main(**void**)
8. {
9. signal(SIGALRM,sig\_alrm);
10. system("date");
11. alarm(20);
12. sleep(5);
13. printf("%d\n",alarm(15));
14. pause();
15. }
17. **static** **void** sig\_alrm(**int** signo){
18. system("date");
19. **return**;
20. }

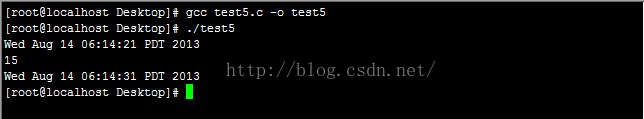


程序分析：在test4.c的main()函数中，先设置了一个闹钟函数alarm(20)，即在20秒时将SIGALRM信号传送送给当前进程；然后又定义了一个延时函数sleep(5)，接着又定义了一个闹钟函数alarm(15)，它的作用是清除前面设置的闹钟alarm(20)并返回剩余的时间20-5=15秒。所以，程序先执行system("date")语句输出当前时间；然后进程休眠5秒后，程序执行输出语句printf("%d\n",alarm(15))，由于alarm(15)先返回15秒，即打印输出15；接着程序执行pause()函数，使当前进程处于挂起状态，直到捕捉到一个信号；当再过15秒后，SIGALARM信号的处理函数sig\_alrm执行system("date")语句输出当前时间；最后pause终止进程。因此，整个程序执行的时间为5+15=20秒。

##### 2.5 alarm()测试3.2

**[cpp]** [view plain](https://blog.csdn.net/u010155023/article/details/51984602) [copy](https://blog.csdn.net/u010155023/article/details/51984602)

1. #include <unistd.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <signal.h>
6. **static** **void** sig\_alrm(**int** signo);
7. **int** main(**void**)
8. {
9. signal(SIGALRM,sig\_alrm);
10. system("date");
11. alarm(20);
12. sleep(5);
13. printf("%d\n",alarm(5));
14. pause();
15. }
17. **static** **void** sig\_alrm(**int** signo){
18. system("date");
19. **return**;
20. }



程序分析：与test4.c文件不同的是，在文件test5.c中闹钟函数为alarm(5)。因此，整个程序执行的时间为5+5=10秒。值得注意的是，alarm(0)表示清除之前设置的闹钟信号，并返回0。因为，如果这里把alarm(5)改成alarm(0)，那么整个程序执行的时间为5+0=5秒。

**最后**：需要注意的是，原作者在文章中进行了精确的时间计算，而程序运行的结果也与作者的计算一致，但即使如此，精确的结果也是不可信的和计算精确的结果也是不可行的。在某些条件下，我们实际花费和等待的时间很有可能比程序设定的时间要长，而且1秒对于现代的操作系统来说，实在是太长了。