

feisky

云计算、虚拟化与Linux技术笔记

博客园 :: 首页 :: 新随笔 :: 联系 :: 订阅 [XML](#) :: 管理

posts - 951, comments - 263, trackbacks - 1, articles - 1

 公告

下载我的Android博客软件：




昵称：feisky
园龄：5年4个月
粉丝：586
关注：10
[+加关注](#)

 搜索

找找看

谷歌搜索

 随笔分类(538)

[Android\(43\)](#)
[C/C++\(34\)](#)
[Linux\(145\)](#)
[Linux kernel\(32\)](#)
[Matlab\(5\)](#)
[NoSQL\(2\)](#)
[OpenStack\(3\)](#)
[Python\(16\)](#)
[QT \(26\)](#)
[Visual C++\(11\)](#)
[Windows\(4\)](#)
[Xen\(62\)](#)
[个人管理\(2\)](#)
[个人日记\(10\)](#)
[机器视觉\(19\)](#)
[嵌入式系统\(8\)](#)
[软件设计\(12\)](#)
[数据库\(3\)](#)
[算法结构\(5\)](#)
[图像处理\(11\)](#)
[网络\(45\)](#)
[虚拟化\(15\)](#)
[云计算\(19\)](#)
[职业规划\(6\)](#)

 我的链接

[Blogger](#)
[GitHub](#)
[Quora](#)
[twitter](#)
[wordpress](#)
[博客园](#)
[新浪微博](#)

Linux定时器的使用

Posted on 2010-03-20 17:40 feisky 阅读(26548) 评论(0) 编辑 收藏

使用定时器的目的无非是为了周期性的执行某一任务，或者是到了一个指定时间去执行某一个任务。要达到这一目的，一般有两个常见的比较有效的方法。一个是用linux内部的三个定时器，另一个是用sleep, usleep函数让进程睡眠一段时间，使用alarm定时发出一个信号，还有那就是用gettimeofday, difftime等自己来计算时间间隔，然后时间到了就执行某一任务，但是这种方法效率低，所以不常用。

alarm

alarm用在不需要经确定时的时候，返回之前剩余的秒数。

NAME

alarm - set an alarm clock for delivery of a signal

SYNOPSIS

```
#include <unistd.h>
unsigned int alarm(unsigned int seconds);
```

DESCRIPTION

alarm arranges for a **SIGALRM** signal to be delivered to the process in seconds seconds.
If seconds is zero, no new alarm is scheduled.
In any event any previously set alarm is cancelled.

测试程序：

1	cat timer.c
2	#include <stdio.h>
3	#include <unistd.h>
4	#include <sys/time.h>
5	#include <signal.h>
6	
7	void func()
8	{
9	printf("2 s reached.\n");
10	}
11	
12	int main()
13	{
14	signal(SIGALRM, func);
15	alarm(2);
16	while(1);
17	return 0;
18	}
19	

Linux内置的3个定时器

Linux为每个任务安排了3个内部定时器：

ITIMER_REAL：实时定时器，不管进程在何种模式下运行（甚至在进程被挂起时），它总在计数。定时到达，向进程发送SIGALRM信号。

ITIMER_VIRTUAL：这个不是实时定时器，当进程在用户模式（即程序执行时）计算进程执行的时间。定时到达后向该进程发送SIGVTALRM信号。

ITIMER_PROF：进程在用户模式（即程序执行时）和核心模式（即进程调度用时）均计数。定时到达产生SIGPROF信号。ITIMER_PROF记录的时间比ITIMER_VIRTUAL多了进程调度所花的时间。

定时器在初始化是，被赋予一个初始值，随时间递减，递减至0后发出信号，同时恢复初始值。在任务中，我们可以一种或者全部三种定时器，但同一时刻同一类型的定时器只能使用一个。

用到的函数有：

```
#include <sys/time.h>
int getitimer(int which, struct itimerval *value);
int setitimer(int which, struct itimerval*newvalue, struct itimerval* oldvalue);
strcut timeval
{
long tv_sec; /*秒*/
long tv_usec; /*微秒*/
};
struct itimerval
{
struct timeval it_interval; /*时间间隔*/
struct timeval it_value; /*当前时间计数*/
};
```

it_interval用来指定每隔多长时间执行任务， it_value用来保存当前时间离执行任务还有多长时间。比如说，你指定it_interval为2秒(微秒为0)，开始的时候我们把it_value的时间也设定为2秒（微秒为0），当过了一秒， it_value就减少一个为1，再过1秒，则it_value又减少1，变为0，这个时候发出信号（告诉用户时间到了，可以执行任务了），并且系统自动把it_value的时间重置为it_interval的值，即2秒，再重新计数。

为了帮助你理解这个问题，我们来看一个例子：

1	#include <stdio.h>
2	#include <signal.h>

```

3  #include <sys/time.h>
4
5  /*
6  ****
7  ** Function name: main()
8  ** Descriptions : Demo for timer.
9  ** Input       : NONE
10 ** Output      : NONE
11 ** Created by  : Chenxibing
12 ** Created Date : 2005-12-29
13 ****
14 ** Modified by :
15 ** Modified Date:
16 ****
17 ****
18 */
19 int limit = 10;
20 /* signal process */
21 void timeout_info(int signo)
22 {
23     if(limit == 0)
24     {
25         printf("Sorry, time limit reached.\n");
26         return;
27     }
28     printf("only %d seconds left.\n", limit--);
29 }
30
31 /* init sigaction */
32 void init_sigaction(void)
33 {
34     struct sigaction act;
35
36     act.sa_handler = timeout_info;
37     act.sa_flags = 0;
38     sigemptyset(&act.sa_mask);
39     sigaction(SIGPROF, &act, NULL);
40 }
41
42 /* init */
43 void init_time(void)
44 {
45     struct itimerval val;
46
47     val.it_value.tv_sec = 1;
48     val.it_value.tv_usec = 0;
49     val.it_interval = val.it_value;
50     setitimer(ITIMER_PROF, &val, NULL);
51 }
52
53
54 int main(void)
55 {
56     init_sigaction();
57     init_time();
58     printf("You have only 10 seconds for thinking.\n");
59
60     while(1);
61     return 0;
62 }
63

```

对于ITIMER_VIRTUAL和ITIMER_PROF的使用方法类似，当你在setitimer里面设置的定时器为ITIMER_VIRTUAL的时候，你把sigaction里面的SIGALRM改为SIGVTALARM，同理，ITIMER_PROF对应SIGPROF。

不过，你可能会注意到，当你用ITIMER_VIRTUAL和ITIMER_PROF的时候，你拿一个秒表，你会发现程序输出字符串的时间间隔会不止2秒，甚至5-6秒才会输出一个，至于为什么，自己好好琢磨一下^_^

sleep

下面我们来看看用sleep以及usleep怎么实现定时执行任务。

```

1. #include <signal.h>
2. #include <unistd.h>
3. #include <string.h>
4. #include <stdio.h>
5.
6. static char msg[] = "I received a msg.\n";
7. int len;
8. void show_msg(int signo)
9. {
10.     write(STDERR_FILENO, msg, len);
11. }
12. int main()
13. {
14.     struct sigaction act;
15.     union sigval tsval;
16.
17.     act.sa_handler = show_msg;
18.     act.sa_flags = 0;
19.     sigemptyset(&act.sa_mask);
20.     sigaction(50, &act, NULL);
21.
22.     len = strlen(msg);
23.     while ( 1 )
24.     {
25.         sleep(2); /*睡眠2秒*/

```

```
26. /*向主进程发送信号，实际上是自己给自己发信号*/
27. sigqueue(getpid(), 50, tsval);
28. }
29. return 0;
30. }
```

看到了吧，这个要比上面的简单多了，而且你用秒表测一下，时间很准，指定2秒到了就给你输出一个字符串。所以，如果你只做一般的定时，到了时间去执行一个任务，这种方法是最简单的。

时间差

下面我们来看看，通过自己计算时间差的方法来定时：

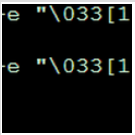


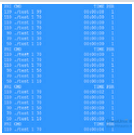
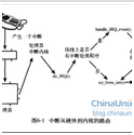


```
1. #include <signal.h>
2. #include <unistd.h>
3. #include <string.h>
4. #include <stdio.h>
5. #include <time.h>
6.
7. static char msg[] = "I received a msg.\n";
8. int len;
9. static time_t lasttime;
10. void show_msg(int signo)
11. {
12. write(STDERR_FILENO, msg, len);
13. }
14. int main()
15. {
16. struct sigaction act;
17. union sigval tsval;
18.
19. act.sa_handler = show_msg;
20. act.sa_flags = 0;
21. sigemptyset(&act.sa_mask);
22. sigaction(50, &act, NULL);
23.
24. len = strlen(msg);
25. time(&lasttime);
26. while ( 1 )
27. {
28. time_t nowtime;
29. /*获取当前时间*/
30. time(&nowtime);
31. /*和上一次的时间做比较，如果大于等于2秒，则立刻发送信号*/
32. if (nowtime - lasttime >= 2)
33. {
34. /*向主进程发送信号，实际上是自己给自己发信号*/
35. sigqueue(getpid(), 50, tsval);
36. lasttime = nowtime;
37. }
38. }
39. return 0;
40. }
```

这个和上面不同之处在于，是自己手工计算时间差的，如果你想更精确的计算时间差，你可以把 time 函数换成gettimeofday，这个可以精确到微妙。

上面介绍的几种定时方法各有千秋，在计时效率上、方法上和时间的精确度上也各有不同，采用哪种方法，就看你程序的需要。

参考：http://www.360doc.com/content/09/0415/22/26398_3145658.shtml
http://blog.chinaunix.net/u2/60434/showart_471561.html

无觅猜您也喜欢：

					
Linux彩色输出	Linux定时器的使用	转一篇Linux可用内存的统计方法	[转]linux进程调度之总章：一些片汤话	[转]linux进程调度之FIFO和RR调度策略	Linux Kernel Development——中断
					
[转]如何判断 Linux	Linux Kernel				

是否运行在虚拟机上

Development——
虚拟文件系统

无关联推荐[?]

分类: [Linux](#)

绿色通道:

好文要顶

关注我

收藏该文

与我联系

feisky
关注 - 10
粉丝 - 586
[+加关注](#)

3

0

(请您对文章做出评价)

« 上一篇: [使用文本文件\(.txt\)进行数据存取的技巧总结](#)

» 下一篇: [每天读一遍，不久你就会变](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

[博客园首页](#) [博文](#) [新闻](#) [闪存](#) [程序员招聘](#) [知识库](#)



- 最新IT新闻：
- 微软宣布 Zoo Tycoon Friends 今夏登陆 Windows 8.1 和 WP8
 - 程序员保护好自己颈椎
 - 9158傅政军：从屌丝程序员到新时代娱乐缔造者
 - 壁纸包又来了：Win7官方主题《春暖花开》
 - 9158自揭面纱：最大的视频秀场，是怎么运行的？
- » 更多新闻...
- 最新知识库文章：
- 开源软件许可协议简介
 - 程序员的自我修养(2)——计算机网络
 - 你是否中了工程师文化的毒？
 - 不安分的工程师
 - 流量劫持——浮层登录框的隐患
- » 更多知识库文章...