LInux下几种定时器的比较和使用

在数据通信过程中，会遇到对数据发送时间的格式要求。所以要在应用中根据实际要求选择不同的定时器，就要考虑到几种应用定时器的特点。

定时器文章参考

一般而言有，

1、sleep，usleep和nanosleep

sleep()和nanosleep()都是使进程睡眠一段时间后被唤醒，但是二者的实现完全不同。
Linux中并没有提供系统调用sleep()，sleep()是在库函数中实现的，它是通过调用alarm()来设定报警时间，调用sigsuspend()将进程挂起在信号SIGALARM上，sleep()只能精确到秒级上。

nanosleep()则是Linux中的系统调用，它是使用定时器来实现的，该调用使调用进程睡眠，并往定时器队列上加入一个timer_list型定时器，time_list结构里包括唤醒时间以及唤醒后执行的函数，通过nanosleep()加入的定时器的执行函数仅仅完成唤醒当前进程的功能。系统通过一定的机制定时检查这些队列（比如通过系统调用陷入核心后，从核心返回用户态前，要检查当前进程的时间片是否已经耗尽，如果是则调用schedule()函数重新调度，该函数中就会检查定时器队列，另外慢中断返回前也会做此检查），如果定时时间已超过，则执行定时器指定的函数唤醒调用进程。当然，由于系统时间片可能丢失，所以nanosleep()精度也不是很高。

alarm()也是通过定时器实现的，但是其精度只精确到秒级，另外，它设置的定时器执行函数是在指定时间向当前进程发送SIGALRM信号。

2、使用信号量SIGALRM + alarm()

alarm方式虽然很好，但这种方式的精度能达到1秒，是无法低于1秒的精度。其中利用了*nix系统的信号量机制，首先注册信号量SIGALRM处理函数，调用alarm()，设置定时长度，代码如下：

```
//设置一个1s延时信号，再注册一个
#include <stdio.h>
#include <signal.h>

void timer(int sig)
{
        if(SIGALRM == sig)
        {
                printf("timer\n");
                alarm(1);       //重新继续定时1s
        }

        return ;
}

int main()
{
        signal(SIGALRM, timer); //注册安装信号

        alarm(1);       //触发定时器

        getchar();

        return 0;
}
```

3、使用RTC机制

RTC机制利用系统硬件提供的Real Time Clock机制，通过读取RTC硬件/dev/rtc，通过ioctl()设置RTC频率，这种方式比较方便，利用了系统硬件提供的RTC，精度可调，而且非常高代码如下：

```
#include <stdio.h>
#include <linux/rtc.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
        unsigned long i = 0;
        unsigned long data = 0;
        int retval = 0;
        int fd = open ("/dev/rtc", O_RDONLY);

        if(fd < 0)
        {
                perror("open");
                exit(errno);
        }

        /*Set the freq as 4Hz*/
        if(ioctl(fd, RTC_IRQP_SET, 1) < 0)
        {
                perror("ioctl(RTC_IRQP_SET)");
                close(fd);
                exit(errno);
        }
        /* Enable periodic interrupts */
        if(ioctl(fd, RTC_PIE_ON, 0) < 0)
        {
                perror("ioctl(RTC_PIE_ON)");
                close(fd);
                exit(errno);
        }

        for(i = 0; i < 100; i++)
        {
                if(read(fd, &data, sizeof(unsigned long)) < 0)
                {
                        perror("read");
                        close(fd);
                        exit(errno);

                }
                printf("timer\n");
        }
        /* Disable periodic interrupts */
        ioctl(fd, RTC_PIE_OFF, 0);
        close(fd);

        return 0;
}
```

该种方式要求系统有RTC设备，我们的1860有两个RTC，用的是电源管理模块的LC1160中的RTC，但是驱动中没有关于RTC_IRQP_SET控制字的支持，需要后期添加驱动实现。

4、使用select()

能精确到1us，目前精确定时的最流行方案。通过使用select()，来设置定时器；原理利用select()方法的第5个参数，第一个参数设置为0，三个文件描述符集都设置为NULL，第5个参数为时间结构体，代码如下：

```
#include <sys/time.h>
#include <sys/select.h>
#include <time.h>
#include <stdio.h>

/*seconds: the seconds; mseconds: the micro seconds*/
void setTimer(int seconds, int mseconds)
{
        struct timeval temp;

        temp.tv_sec = seconds;
        temp.tv_usec = mseconds;
```

非策略

< 2017年8月 >

日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

公告

Visitors
3,662 51
210 10
153 9
65 6
55 6
55 5
FLAG counter

昵称：青丘凤九
园龄：2年1个月
粉丝：14
关注：49
+加关注

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签
git(4)
git.branch.删除分支.分支重命名.创建分支(1)
git.reset(1)
gpio interrupts(1)
ini(1)
Linux 应用 定时器(1)
propertytree(1)
py追加(1)
QT(1)
ROS(1)
更多

随笔分类
Android(1)
ARM
C++ | 设计模式(1)
C++ | 算法
C++ | 语法(5)
Driver(3)
git | 本地指令(5)
Je Viens(1)
Python | just py(3)
Python | ROS
Reminider(2)
ROS | Code(1)
ROS | 基础(6)
STM32(1)
基础知识
人月神话(1)
随手记IT(8)
网络通讯基础(1)

随笔档案
2017年7月 (1)
2017年6月 (1)
2017年4月 (2)
2017年1月 (1)
2016年11月 (1)
2016年7月 (1)
2016年4月 (3)
2016年3月 (6)
2016年2月 (2)
2016年1月 (2)
2015年12月 (1)
2015年11月 (2)
2015年10月 (5)
2015年9月 (4)
2015年8月 (2)
2015年7月 (1)

文章分类
C++ | STL(1)

最新评论
1. Re:遇到bug怎么办
1. 全文搜索, 推荐用 grep -n -r "balabala" * 速度很快2. 版本保存, 推荐用git, 没外网的话本地也是可以的, 有事没事多开branch, 随时commit3. 功能实.....
--ferstar
2. Re:actionlib的身世之谜
大姐厉害
学习了
--海漩涡
3. Re:SPI 驱动分析
爱上了怎么办?
--ferstar
4. Re:Socket通讯
大神请收下我的膝盖~~
--ferstar
5. Re:actionlib的身世之谜
姐姐，算了，不要想了，可能根本实现不了
--番茄素

阅读排行榜
1. Subscribe的第四个参数用法(1231)
2. C++ | boost库 类的序列化(1113)
3. ROS的单线程Spinning和多线程Spinning(1089)
4. actionlib的身世之谜(998)
5. QT打开ROS工作空间时遇到的问题和解决方法(748)

评论排行榜
1. actionlib的身世之谜(14)

2. QT打开ROS工作空间时遇到的问题和解决方法(6)
3. git删除远程仓库的某次错误提交(3)
4. C++的vector学习abc(1)
5. 遇到bug怎么办(1)

推荐排行榜

1. actionlib的身世之谜(1)
2. git删除远程仓库的某次错误提交(1)
3. C++的vector学习abc(1)

```
        select(0, NULL, NULL, NULL, &temp);
        printf("timer\n");

        return ;
}

int main()
{
        int i;

        for(i = 0 ; i < 100; i++)
                setTimer(1, 0);

        return 0;
}
```

结果是，每隔1s打印一次，打印100次。

select定时器是阻塞的，在等待时间到来之前什么都不做。要定时可以考虑再开一个线程来做。

you lust for my life, darkness and light

分类: 随手记[],Reminider

标签: Linux 应用 定时器

« 上一篇：调试内核打印debugfs
» 下一篇：linux 下的read write 和fread fwrite

posted on 2017-04-16 23:19 青丘凤九 阅读(131) 评论(0) 编辑 收藏