<https://www.cnblogs.com/feixiao5566/p/6720573.html>

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](http://m.blog.csdn.net/article/details?id=7669296)

一般而言有，

1、[sleep，usleep和nanosleep](http://www.jb51.net/LINUXjishu/109403.html" \t "_blank)

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：

[复制代码](javascript:void(0);)

//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;

}

[复制代码](javascript:void(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：

[复制代码](javascript:void(0);)

#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;

}

[复制代码](javascript:void(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：

[复制代码](javascript:void(0);)

#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;

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;

}

[复制代码](javascript:void(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。