文章编号:1000 - 582X(2004)12 - 0045 - 04

基于 Linux 的 IPC 测控系统中多任务调度的实现*

李志文,陈曾汉
(重庆大学 高电压与电工新技术教育部重点实验室,重庆 400030)

摘 要:IPC(工业 PC)系统通常需要持续长时间的工作,对运行其上的系统软件的稳定性要求非常高。Linux 具有运行稳定、源代码开放并且免费等诸多优点,因此采用 Linux 作为平台开发了 1 套 IPC 测控系统。多任务调度是系统软件中的关键部分。提出了一种利用 Linux 中的闹钟机制来实现 IPC 测控系统中多任务调度的简单方法。Linux 中的闹钟机制是由软件定时器和用来安装和处理闹钟信号的系统调用组合实现的。首先介绍了进程定时器、软中断信号、相关系统调用以及闹钟机制在多任务调度中的应用。随后给出了一个在 IPC 测控系统中应用的实例。该系统已经通过了长时间的运行测试,完全实现了预定功能,并且稳定性良好,证明了所介绍的调度方法的可行性。

关键词:IPC 测控系统;Linux;多任务调度;闹钟机制;进程定时器;信号
中图分类号:TP316 **文献标识码**:A

Linux 是一个开放源代码、协作开发的类 Unix 操作系统。它可以运行在大多数硬件平台上,并且对硬件的要求很低。它提供了广泛的网络支持、同其他系统交换所必须的许多特性和大量本身拥有以及从其他系统引入的应用程序。IPC(工业 PC)系统通常工作在条件恶劣的工业现场。相应地,对于运行其上的系统软件的稳定性也有很高的要求。由于 Linux 系统具有出众的稳定性,并且它是免费的,所以在工业测控领域已经得到了广泛的应用。从研究和应用的角度出发,在 Linux 环境下开发了一套 IPC 测控系统。对于一个多任务的测控系统,任务的调度无疑是其中最重要的部分。应用多进程或者多线程调度的算法实现上通常比较复杂。文中介绍了一种应用 Linux 中的闹钟机制来实现 IPC 测控系统多任务调度的简便方法。

1 闹钟机制的实现

Linux 中的"闹钟"是由两部分组合实现的,一部分是用于定时的软件定时器,另一部分是用来安装和处理闹钟信号的系统调用[1]。

1.1 Linux 中的进程定时器

操作系统为了能够准时调度任务,需要有一种能够保证调度准时进行的机制。这种机制是通过定时器来实现的。从硬件上来讲支持各种操作系统的微处理器必须包含一个可以周期性中断、可编程的间隔定时器。这个周期性中断被称为系统时钟滴答,它就像节拍器一样组织系统任务。从软件上来讲必须有一个软件上的定时器在硬件中断到来时处理任务调度[2]。

Linux 的定时器是一种定时服务机制,它所起的作用是在某个特定的时刻唤醒某个进程来完成一些工作。Linux 中的定时器分为内核定时器和进程定时器,文中所应用到的是进程定时器。

1.1.1 Linux 进程定时器的分类

Linux 的进程定时器,又称为间隔定时器。所谓"间隔定时器"(Interval Timer,简称 itimer)就是指定时器采用"间隔"值(interval)来作为计时方式,当定时器启动后,间隔值 interval 将不断减小。当 interval 值减到 0 时,就说该间隔定时器到期。间隔定时器实际上是通过一个不断减小的计数器来计时的。

由于进程执行时分为核心模式和用户模式,所以相应的进程执行时间也分为进程本身(即在用户模式下)的执行时间和系统代表进程(即通过系统调用在核心模式下)的执行时间。对于不同的时间,Linux 运行了不同的间隔定时器。这些定时器可以分为以下 3 种:

* 收稿日期:2004 - 08 - 11
作者简介:李志文(1978 -),男,内蒙古赤峰人,重庆大学硕士研究生,主要研究方向为工业 PC 机及测控系统。

1) 真实间隔定时器 ITIMER_REAL:这种间隔定时器在启动后,不管进程是否运行,每个时钟滴答(tick)都将其间隔计数器减当前值 1。当减到 0 值时,内核向进程发送 SIGALRM 信号。

2) 虚拟间隔定时器 ITIMER_VIRTUAL:也称为进程的用户态间隔定时器。当虚拟间隔定时器启动后,只有当进程在用户态下运行时,一次时钟滴答才能使间隔计数器当前值减 1。当减到 0 值时,内核向进程发送 SIGVTALRM 信号(虚拟闹钟信号)。

3) PROF 间隔定时器 ITIMER_PROF:当一个进程的 PROF 间隔定时器启动后,只要该进程处于运行中,不管是在用户态或核心态下执行,每个时钟滴答都使间隔计数器值减 1。当减到 0 值时,内核向进程发送 SIGPROF 信号。

Linux 允许同时启动多个定时器,Linux 将定时器工作所需的所有信息保存在进程的 task_struct 中。可以利用系统调用设置定时器、启动定时器、停止定时器或读出定时器的当前值[1,3-5]。

1.1.2 进程定时器的设置

在 Linux 中,通过系统调用 setitimer( ) 或者 alarm( ) 来对进程定时器进行设置。alarm( ) 的定时精度为秒级。setitimer( ) 比 alarm( ) 功能强大并且精度更高,它不仅设置进程定时器的值,而且还返回该进程定时器的原有信息。setitimer( ) 系统调用形式如下:

```
int setitimer ( int which, struct itimerval * new, struct itimerval * old)
```

其中,参数 which 指定设置进程的哪一个间隔定时器,其值可以是 ITIMER_REAL、ITIMER_VIRTUAL 和 ITIMER_PROF 三者之一;输入参数 new 指向用户空间的一个 itimerval 结构,含有待设置的新值;输出参数 old 指向用户空间的一个 itimerval 结构,用于接收间隔定时器的原有信息[6]。

其中用到的数据结构简介如下:

数据结构 timeval 是系统时钟结构体,定义在 include/linux/time.h 中。

```
struct timeval {
    time_t    tv_sec ;
    suseconds_t    tv_usec ;
}
```

其中 tv_sec 是从 1970 年 1 月 1 日开始计算的秒数,tv_usec 是当前秒内的微秒数。

数据结构 itimerval。在内核中间隔定时器的间隔计数器是以时钟滴答为单位,但是让用户以时钟滴答为单位来指定间隔定时器计数器的初值是不方便的,所以 Linux 定义了数据结构 itimerval 来让用户以秒为单位指定间隔定时器的时间间隔值。这个结构也定义在 include/linux/time.h 中。

```
struct itimerval {
    struct timeval it_interval;
    struct timeval it_value;
}
```

其中,it_interval 成员表示间隔计数器的初始值,也就是定时器时间间隔。而 it_value 成员表示间隔计数器的当前值。这两个成员都是 timeval 结构类型的变量,因此其精度可以达到微秒级[1]。

1.2 信号安装函数

进程之间可以通过系统调用相互发送软中断信号,内核也可以从内部发送软中断信号。软中断信号简称信号(signal)。信号是在软件层次上对中断机制的一种模拟,在原理上,一个进程收到一个信号与处理器收到一个中断请求可以说是一样的。同中断(interrupt)和异常(exception)相比较,信号对用户态的进程是可见的,可以被用户态进程捕获。

如果进程要处理某一信号,那么就要在进程中安装该信号。安装信号主要用来确定信号值及进程针对该信号值的动作之间的映射关系,即进程将要处理哪个信号;该信号被传递给进程时,将执行何种操作。在调用系统调用 setitimer( ) 设置定时器之前,必须先调用信号安装函数 signal( ) 或者 sigaction( ) 来接收由间隔定时器到期时所产生的警报信号,而且有时还需要建立一个恰当的信号处理函数(handler),使进程在收到信号后,执行这个处理函数所设定的操作。

signal( ) 函数提供了一个对特殊信号做出响应的简单的接口,和这个函数有关的宏声名在头文件 signal.h中。

```
sighandler_t signal ( int signum, sighandler_t action)
```

第 1 个参数 signum 指定一个要控制的信号名。系统调用 setitimer( ) 所产生的信号就是 SIGALRM、SIGVTALRM 或者 SIGPROF 三者之一。第 2 个参数 action 指定用对信号 signum 的处理。action 的值可以是常数 SIG_IGN(忽略此信号)或者常数 SIG_DFL(接到信号后的系统默认动作)或者当接到此信号后要调用的函数(即 handler)的地址。如果 signal( ) 调用成功,返回最后一次为安装信号 signum 而调用 signal( ) 时的 action 的值;失败则返回 SIG_ERR[1,6]。

信号处理函数用一个用来确定信号名的整数作为参数,返回 void。可以定义如下:

```
void handler ( int signum ) {
    ...
```

}

在信号处理函数里，用户可以执行自己定义的一些操作。

2 应用实例

文中给出的实例是应用上述机制开发的一个小型的多任务测控系统，利用工业 PC 机进行模拟量、开关量信号采集、处理与直流电动机和步进电动机的控制。该系统中包括 3 个定时任务：0 点制表、8 点制表和 16 点制表；4 个周期任务：每 10 ms 对开关量采样一次，每 1 s 对模拟量采样一次，每 250 ms 测量直流电动机转速、按设定值调整转速并在屏幕上显示速度和偏差，每 500 ms 对温度测控子系统的温度值采样一次并控制步进电动机按采样结果正转或反转若干步；7 个键盘任务，用于显示画面切换；北京时间的动态显示任务。

在多任务的操作系统中，各个任务都根据一定条件在几个状态之间转换，这几个状态分别为就绪状态、运行状态、阻塞状态和睡眠状态。为描述任务的状态，必须为每个任务设立一个状态标志变量，形成状态队列。状态标志变量至关重要，因为它是任务调度能“感知”任务存在的唯一标志，它与任务一一对应。任务管理并不关心各任务程序的具体内容和功能，它只是面对状态队列，通过对状态队列的扫描检查来管理和调度任务[2]。

由于本 IPC 测控系统需要周期性地对若干量进行重复检测，所以针对不同被检测量，需要设定不同的定时和周期阈值，以便唤醒相应的任务，使之进入就绪状态。利用 Linux 的间隔定时器产生固定的以滴答(tick)为单位的系统时钟。设定的滴答为 10 ms，于是所有的周期任务的启动时刻都以这个“10 ms”为基准。对于运行周期为 1 s 的任务，该任务的周期阈值就是 100。每一个滴答，该值减 1，减到 0 时即表明该任务的运行周期到，应当被唤醒而进入就绪状态。系统初始化及任务调度框图如图 1 所示。

信号处理函数(handler)的框图如图 2 所示。

下面的程序段说明了主程序中对于信号处理函数的调用：

```
#include <stdio.h>
#include <signal.h>
#include <sys/time.h>
…
void handler ( int sig ){
…
```

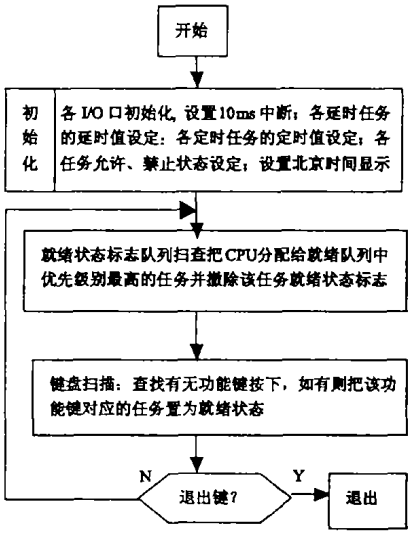

图 1　初始化及任务调度框图

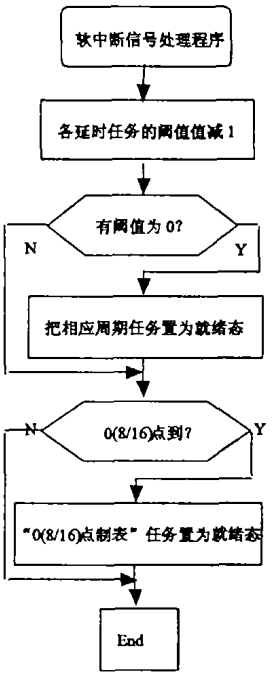

图 2　信号处理函数框图

```
}
…
main ( ){
…
  if ( signal (SIGALRM,handler ) = =SIG_ERR)
    {
        printf ("Unable to create handler for SIGALRM
\n");
        exit (0);
    } /*安装闹钟信号 SIGALRM*/
  tv.it_interval.tv_sec =0;
  /*设置定时器的计时间隔为 10 000 微秒，即 10 毫秒*/
  tv.it_interval.tv_usec = 10 000;
```

```
    tv. it_value. tv_sec = 0;
    tv. it_value. tv_usec = 10 000;  /* 设定定时器的当前值 */

    /* 调用 setitimer 设置定时器,这个函数三个参数的含义分别是设置 ITIMER_REAL 类型的定时器,要设置的值存放在变量 tv 中,该定时器设置前的值在设置后保存的地址,这个参数为 NULL,就放弃保存设置前的值 */
    setitimer (ITIMER_REAL, &tv, NULL);
...
}
```

3 结束语

通过上面的介绍可知, Linux 的闹钟机制将进程定时器和软中断信号结合起来应用是方便且简单的,而且功能很强大。这种多任务调度的实现方法适用于任务不太多,并且没有复杂的内存分配和资源管理的测控系统。缺点是受限于 Linux 的时钟精度为 10 ms,所以在标准 Linux 上实现的这套系统所能达到的最高调度精度为 10 ms。但对于那些实时性要求不高的工业测控系统来说已经能满足需要了。在对该系统进行了长时间的运行测试后,发现系统完全实现了预定功能,并且稳定性良好。

参考文献:

[1] 李善平,陈文智. 边干边学 - Linux 内核指导[M]. 杭州:浙江大学出版社,2002.

[2] 陈曾汉,刘明白,赵志强,等. 工业 PC 及测控系统[M]. 北京:机械工业出版社,2004.

[3] STEVENS W R. UNIX 环境高级编程[M]. 尤晋元译. 北京:机械工业出版社,2000.

[4] 怀石工作室. Linux 上的 c 编程[M]. 北京:中国电力出版社,2000.

[5] NUTT G. 潘登,冯锐,陆丽娜,等. Linux 操作系统内核实习[M]. 北京:机械工业出版社,2002.

[6] The GNU C Library[EB/OL]. http://www. gnu. org/software/libc/manual/,2001.

Realization of Multi-task Scheduling in Linux Based IPC Measurement and Control System

LI Zhi-wen, CHEN Zeng-han

(State Key Laboratory for High Voltage and New Technology of Electrical Engineering, Chongqing University, Chongqing 400030, China)

Abstract: Industrial Personal Computer(IPC) system generally works in long duration, and so a stable software is essential for an IPC system. Linux has a good deal advantages, such as, stability in use, open source code and free of charge, therefore an IPC measurement and control system has developed based on Linux. Multi-task scheduling is the critical part in system software. A simple method to realize multi-task scheduling in IPC measurement and control system based on Linux alarm clock mechanism is presented. Linux alarm clock mechanism is implemented by the cooperation of software timers and system call that is used to install and handle alarm clock. First, Linux process timers, soft interrupt signals, relevant system call are introduced, as well as applications of Linux alarm clock mechanism in multi-task scheduling. And then an example of the application in the IPC measurement and control system is given. The system has passed longtime run test, accomplished expected functions and shown well stability, so that practicability of the scheduling method has been authenticated.

Key words: IPC measurement and control system; Linux; multi-task scheduling; alarm clock mechanism; process timers; signal

(编辑 陈移峰)