**Linux 中的计时**

本文描述了Linux系统中一些与计时相关的问题和解决方法。因为在学习和研究的过程中我们经常需要统计程序或程序段的耗时，评估它们的性能好坏。因而这些问题对于我们来说，经常会遇到。掌握多种计时方法对于开发人员或科研工作者来说都是必须掌握的一项技能。本文解决了在Linux计时当中经常会遇到的一些技术问题，相信会对他人的工作有所帮助。

0https://dw1.s81c.com/developerworks/i/v17/dw-cmts-arrow.png [评论](https://www.ibm.com/developerworks/cn/linux/l-time/#icomments)

[车皓阳](https://www.ibm.com/developerworks/cn/linux/l-time/#authorN10017) ([grandiose11@msn.com](mailto:grandiose11@msn.com?subject=Linux%20%E4%B8%AD%E7%9A%84%E8%AE%A1%E6%97%B6))中科院软件所

2003 年 10 月 01 日

* [expand](https://www.ibm.com/developerworks/cn/linux/l-time/#toggle)**内容**



在 IBM Bluemix 云平台上开发并部署您的下一个应用。

[**开始您的试用**](https://developer.ibm.com/sso/bmregistration?lang=zh_CN&ca=dwchina-_-bluemix-_-l-time-_-sidebar)

实际上，可能还会有其它一些方法可以完成本文讨论的任务。我仅讨论了我所使用的方法，这并不意味着除此之外的其它方法就很差劲，只不过对我来说这些方法相对简单有效而已。

**Linux中的时间**

在Linux系统中，时间扮演着一个非常重要的角色，它几乎无处不在。开机时，会显示如下的信息：

Last login: Tue Sep 23 22:12:50 from 192.168.6.100

关机时，我们可以使用shutdown命令指定何时或多长时间后机器将会定时关闭。我们有可能还会设置一台Linux时间服务器与互联网上的一级或二级时间服务器同步，总之，在Linux系统中，我们必须要了解时间。

实际上，linux系统有着自己的计时器时钟。可以实验一下，分别执行date和/sbin/clock（或sbin/hwclock）命令，得到的时间是不同的。

[grandiose@Bygone grandiose]$ date

Sun Sep 28 21:11:02 EDT 2003

[grandiose@Bygone grandiose]$ /sbin/clock

Sun 28 Sep 2003 09:07:07 PM EDT -0.466994 seconds

当你以 root 身份改变了系统时间之后，请记住以 clock -w 来将系统时间写入 CMOS 中。

[**回页首**](https://www.ibm.com/developerworks/cn/linux/l-time/#ibm-pcon)

**使用C语言进行计时**

在用户空间中可以使用C语言函数gettimeofday 得到时间，它的调用格式是：

#include <sys/time.h>

int gettimeofday(struct timeval \*tv, struct timezone \*tz);

int settimeofday(const struct timeval \*tv , const struct timezone \*tz);

结构timeval的定义为：

strut timeval {

long tv\_sec; /\* 秒数 \*/

long tv\_usec; /\* 微秒数 \*/

};

可以看出，使用这种方式计时，精度可达微秒，也就是10-6秒。进行计时的时候，我们需要前后调用两次gettimeofday，然后计算中间的差值：

gettimeofday( &start, NULL );

foo();

gettimeofday( &end, NULL );

timeuse = 1000000 \* ( end.tv\_sec - start.tv\_sec ) + end.tv\_usec - start.tv\_usec;

timeuse /= 1000000;

[**回页首**](https://www.ibm.com/developerworks/cn/linux/l-time/#ibm-pcon)

**Shell计时**

在Linux的Shell下，我们经常也使用Shell内置的time命令和GNU版的time命令来测试程序运行的时间。

内置的time提供的参数选项比较少，而GNU的time则提供了丰富的参数选项，包括指定输出文件等功能。

[grandiose@Bygone grandiose]$ /usr/bin/time --output=foo.txt foo

上句只有时间信息输出到了foo.txt文件中，如果想要包括foo执行的结果，就需要按下句这样使用：

[grandiose@Bygone grandiose]$ /usr/bin/time --output=foo.txt --append foo >foo.txt

如果想要控制输出时间的格式，可以使用-f开关进行格式化：

[grandiose@Bygone grandiose]$ /usr/bin/time

--output=foo.txt -f "\\t%E real,\\t%U user,\\t%S sys" foo

如果仍需要使用Shell内置的time命令，可以用下面一句将结果输出至文件：

[grandiose@Bygone grandiose]$ (time foo) 2>foo.txt

这样写是因为内置命令time的输出是到标准错误的，文件描述符2表示标准错误stderr。如果还想要包括foo执行的结果，就要这样：

[grandiose@Bygone grandiose]$ (time foo) >foo.txt 2>&1

其中2>&1的含义是2与1 相同，一起送入foo.txt中。

nohup命令可以保证进程在退出系统之后仍能运行，这是它的常规用法。我们也可以这样使用nohup：

[grandiose@Bygone grandiose]$ nohup time foo

结果全部输出至nohup.out，也包括程序运行的时间信息。可以使用下面的语句将时间信息输出至文件foo.txt中。

[grandiose@Bygone grandiose]$ tail -2 nohup.out > foo.txt

为了保证和POSIX一致，输出的时间格式为（nohup.out中的内容除外）：

real 0m0.007s

user 0m0.002s

sys 0m0.004s

我们可以使用linux下面一些过滤命令如awk、sed、grep、tr等过滤出我们想要得到的内容，例如想要得到real段对应的时间：

[grandiose@Bygone grandiose]$ grep real foo.txt | cut -f2，或者

[grandiose@Bygone grandiose]$ sed -n '2p' foo.txt | cut -f2

在Shell下，输出的时间精度为毫秒级，如果需要微秒级的计时，那就应该在程序中进行处理。

[**回页首**](https://www.ibm.com/developerworks/cn/linux/l-time/#ibm-pcon)

**内核空间中的计时**

如果要定制自己的设备驱动程序，可能就会用到内核里的计时功能。Linux内核空间中的计时与用户空间的计时不太相同。在内核空间里，有一个全局变量Jiffies维护着当前的时间。与系统时钟有关的调用有（新的定时服务）：

#include <asm/param.h>

#include <linux/timer.h>

void add\_timer(struct timer\_list \* timer);

int del\_timer(struct timer\_list \* timer);

inline void init\_timer(struct timer\_list \* timer);

结构struct timer\_list的定义为：

struct timer\_list {

struct timer\_list \*next;

struct timer\_list \*prev;

unsigned long expires;

unsigned long data;

void (\*function)(unsigned long d);

};

其中过期时间expires是要执行function的时间。一般在调用add\_timer时jiffies = jiffies + num，表示在num个系统最小时间间隔后执行function。系统最小时间间隔与所用的硬件平台有关， 在核心里定义了常数HZ表示一秒内最小时间间隔的数目，则num\*HZ表示num 秒。系统计时到预定时间就调用function，并把此子程序从定时队列里删除， 因此如果想要每隔一定时间间隔执行一次的话，就必须在function里再一次调用add\_timer。function的参数d即为timer里面的data项。

Jiffies的计时精度是百分之一秒，如果在内核中需要更为精确的计时，就需要用到time\_calls.h中的函数，它们可用于高精度的时间计算。

[**回页首**](https://www.ibm.com/developerworks/cn/linux/l-time/#ibm-pcon)

**补充**

有的时候，我们需要较为精确地得出被测目标的运行时间，这时一般需要多次运行取均值以消除误差。

gettimeofday( &start, NULL );

for ( int i = 0; i< 10; i++ ) foo();

gettimeofday( &end, NULL );

timeuse = 1000000 \* ( end.tv\_sec - start.tv\_sec ) + end.tv\_usec - start.tv\_usec;

timeuse /= 10000000;

上面的统计实际也引入了新的误差，当循环指令的执行时间与foo()相比可忽略的话，这种计时才是可以接受的；否则我们就要除去循环指令的执行时间，才会得到正确的统计计时。

在Linux Shell下，如果统计次数较少，则可以：

for i in 1 2 3 4 5 6 7 8 9 10

do

(time foo) 2>foo.tmp

grep real foo.tmp | cut -f2 >> foo.txt

done

如果计时次数较多，则需要：

i=1

while [ $i -le 100 ]

do

(time foo) 2>foo.tmp

grep real foo.tmp | cut -f2 >> foo.txt

i=`expr $i + 1`

done

写进foo.txt的内容如果手动来计算平均值，会比较费时，我们可以写一段Shell脚本或用C语言来读取文件，计算其均值。

/\*耗时中分部总和\*/

cut -d'm' -f1 foo.txt > foo.tmp

sum=0

while read line

do

sum=$(echo "$sum+$line" | bc -l)

done < foo.tmp

echo $sum

/\*耗时中秒部总和\*/

cut -d'm' -f2 foo.txt | tr -d 's'> foo.tmp

sum=0

while read line

do

sum=$(echo "$sum+$line" | bc -l)

done < foo.tmp

echo $sum

计算出分部与秒部总和之后，然后再手动计算平均值，这样要容易得多。注意，上面没有使用expr进行计算的原因，是因为expr只能支持整型值。在Linux shell下，如果要计算浮点数，就需要使用bc或者是gexpr。

[**回页首**](https://www.ibm.com/developerworks/cn/linux/l-time/#ibm-pcon)

**结束语**

实际上，我们还可以使用诸如Perl、Python等多种语言在Linux系统中进行计时。选择何种工具或语言进行计时，这与被测程序或程序段的类型以及它们的编写语言相关。综合考虑精度、运行时间、运行次数等要求，才能合理可靠地得出程序的运行时间。