<http://blog.sina.com.cn/s/blog_49efd54001019r1j.html>

## <http://hi.baidu.com/1984iris/item/085dbad4f5843dfccb0c3992>

|  |
| --- |
| 想获取一下目标机运行时linux系统的硬件占用情况，写了这几个小程序，以后直接用了。 方法就是读取proc下的文件来获取了。 cpu使用率：    /proc/stat ，内存使用情况：      /proc/meminfo 看程序 ：  typedef struct PACKED         //定义一个cpu occupy的结构体 { char name[20];      //定义一个char类型的数组名name有20个元素 unsigned int user; //定义一个无符号的int类型的user unsigned int nice; //定义一个无符号的int类型的nice unsigned int system;//定义一个无符号的int类型的system unsigned int idle; //定义一个无符号的int类型的idle }CPU\_OCCUPY;  typedef struct PACKED         //定义一个mem occupy的结构体 { char name[20];      //定义一个char类型的数组名name有20个元素 unsigned long total;  char name2[20]; unsigned long free;                        }MEM\_OCCUPY;  get\_memoccupy (MEM\_OCCUPY \*mem) //对无类型get函数含有一个形参结构体类弄的指针O {     FILE \*fd;               int n;                  char buff[256];        MEM\_OCCUPY \*m;     m=mem;                                                                                                                    fd = fopen ("/proc/meminfo", "r");             fgets (buff, sizeof(buff), fd);      fgets (buff, sizeof(buff), fd);      fgets (buff, sizeof(buff), fd);      fgets (buff, sizeof(buff), fd);      sscanf (buff, "%s %u %s", m->name, &m->total, m->name2);           fgets (buff, sizeof(buff), fd); //从fd文件中读取长度为buff的字符串再存到起始地址为buff这个空间里      sscanf (buff, "%s %u", m->name2, &m->free, m->name2);           fclose(fd);     //关闭文件fd }  int cal\_cpuoccupy (CPU\_OCCUPY \*o, CPU\_OCCUPY \*n)  {        unsigned long od, nd;         unsigned long id, sd;     int cpu\_use = 0;             od = (unsigned long) (o->user + o->nice + o->system +o->idle);//第一次(用户+优先级+系统+空闲)的时间再赋给od     nd = (unsigned long) (n->user + n->nice + n->system +n->idle);//第二次(用户+优先级+系统+空闲)的时间再赋给od            id = (unsigned long) (n->user - o->user);    //用户第一次和第二次的时间之差再赋给id     sd = (unsigned long) (n->system - o->system);//系统第一次和第二次的时间之差再赋给sd     if((nd-od) != 0)     cpu\_use = (int)((sd+id)\*10000)/(nd-od); //((用户+系统)乖100)除(第一次和第二次的时间差)再赋给g\_cpu\_used     else cpu\_use = 0;     //printf("cpu: %u/n",cpu\_use);     return cpu\_use; }  get\_cpuoccupy (CPU\_OCCUPY \*cpust) //对无类型get函数含有一个形参结构体类弄的指针O {        FILE \*fd;              int n;                 char buff[256];      CPU\_OCCUPY \*cpu\_occupy;     cpu\_occupy=cpust;                                                                                                                     fd = fopen ("/proc/stat", "r");      fgets (buff, sizeof(buff), fd);          sscanf (buff, "%s %u %u %u %u", cpu\_occupy->name, &cpu\_occupy->user, &cpu\_occupy->nice,&cpu\_occupy->system, &cpu\_occupy->idle);          fclose(fd);      }  int main() {     CPU\_OCCUPY cpu\_stat1;     CPU\_OCCUPY cpu\_stat2;     MEM\_OCCUPY mem\_stat;     int cpu;          //获取内存     get\_memoccupy ((MEM\_OCCUPY \*)&mem\_stat);          //第一次获取cpu使用情况     get\_cpuoccupy((CPU\_OCCUPY \*)&cpu\_stat1);     sleep(10);          //第二次获取cpu使用情况     get\_cpuoccupy((CPU\_OCCUPY \*)&cpu\_stat2);          //计算cpu使用率     cpu = cal\_cpuoccupy ((CPU\_OCCUPY \*)&cpu\_stat1, (CPU\_OCCUPY \*)&cpu\_stat2);          return 0; } |

|  |
| --- |
| 想获取一下目标机运行时linux系统的硬件占用情况，写了这几个小程序，以后直接用了。 方法就是读取proc下的文件来获取了。 cpu使用率：    /proc/stat ，内存使用情况：      /proc/meminfo 看程序 ：  typedef struct PACKED         //定义一个cpu occupy的结构体 { char name[20];      //定义一个char类型的数组名name有20个元素 unsigned int user; //定义一个无符号的int类型的user unsigned int nice; //定义一个无符号的int类型的nice unsigned int system;//定义一个无符号的int类型的system unsigned int idle; //定义一个无符号的int类型的idle }CPU\_OCCUPY;  typedef struct PACKED         //定义一个mem occupy的结构体 { char name[20];      //定义一个char类型的数组名name有20个元素 unsigned long total;  char name2[20]; unsigned long free;                        }MEM\_OCCUPY;  get\_memoccupy (MEM\_OCCUPY \*mem) //对无类型get函数含有一个形参结构体类弄的指针O {     FILE \*fd;               int n;                  char buff[256];        MEM\_OCCUPY \*m;     m=mem;                                                                                                                    fd = fopen ("/proc/meminfo", "r");             fgets (buff, sizeof(buff), fd);      fgets (buff, sizeof(buff), fd);      fgets (buff, sizeof(buff), fd);      fgets (buff, sizeof(buff), fd);      sscanf (buff, "%s %u %s", m->name, &m->total, m->name2);           fgets (buff, sizeof(buff), fd); //从fd文件中读取长度为buff的字符串再存到起始地址为buff这个空间里      sscanf (buff, "%s %u", m->name2, &m->free, m->name2);           fclose(fd);     //关闭文件fd }  int cal\_cpuoccupy (CPU\_OCCUPY \*o, CPU\_OCCUPY \*n)  {        unsigned long od, nd;         unsigned long id, sd;     int cpu\_use = 0;             od = (unsigned long) (o->user + o->nice + o->system +o->idle);//第一次(用户+优先级+系统+空闲)的时间再赋给od     nd = (unsigned long) (n->user + n->nice + n->system +n->idle);//第二次(用户+优先级+系统+空闲)的时间再赋给od            id = (unsigned long) (n->user - o->user);    //用户第一次和第二次的时间之差再赋给id     sd = (unsigned long) (n->system - o->system);//系统第一次和第二次的时间之差再赋给sd     if((nd-od) != 0)     cpu\_use = (int)((sd+id)\*10000)/(nd-od); //((用户+系统)乖100)除(第一次和第二次的时间差)再赋给g\_cpu\_used     else cpu\_use = 0;     //printf("cpu: %u/n",cpu\_use);     return cpu\_use; }  get\_cpuoccupy (CPU\_OCCUPY \*cpust) //对无类型get函数含有一个形参结构体类弄的指针O {        FILE \*fd;              int n;                 char buff[256];      CPU\_OCCUPY \*cpu\_occupy;     cpu\_occupy=cpust;                                                                                                                     fd = fopen ("/proc/stat", "r");      fgets (buff, sizeof(buff), fd);          sscanf (buff, "%s %u %u %u %u", cpu\_occupy->name, &cpu\_occupy->user, &cpu\_occupy->nice,&cpu\_occupy->system, &cpu\_occupy->idle);          fclose(fd);      }  int main() {     CPU\_OCCUPY cpu\_stat1;     CPU\_OCCUPY cpu\_stat2;     MEM\_OCCUPY mem\_stat;     int cpu;          //获取内存     get\_memoccupy ((MEM\_OCCUPY \*)&mem\_stat);          //第一次获取cpu使用情况     get\_cpuoccupy((CPU\_OCCUPY \*)&cpu\_stat1);     sleep(10);          //第二次获取cpu使用情况     get\_cpuoccupy((CPU\_OCCUPY \*)&cpu\_stat2);          //计算cpu使用率     cpu = cal\_cpuoccupy ((CPU\_OCCUPY \*)&cpu\_stat1, (CPU\_OCCUPY \*)&cpu\_stat2);          return 0; } |

我们在搞性能测试的时候，对后台服务器的CPU利用率监控是一个常用的手段。服务器的CPU利用率高，则表明服务器很繁忙。如果前台响应时间越来越大，而后台CPU利用率始终上不去，说明在某个地方有瓶颈了，系统需要调优。这个是即使不懂技术的人都容易理解的事情。

上面理解对吗？我个人觉得不十分准确。这个要看后台你测试的进程是什么类型的。如果是计算密集型的进程，当前端压力越来越大的时候，很容易把CPU 利用率打上去。但是如果是I/O网络密集型的进程，即使客户端的请求越来越多，但是服务器CPU不一定能上去，这个是你要测试的进程的自然属性决定的。比较常见的就是，大文件频繁读写的cpu开销远小于小文件频繁读写的开销。因为在I/O吞吐量一定时，小文件的读写更加频繁，需要更多的cpu来处理I/O 的中断。

在Linux/Unix下，CPU利用率分为**用户态**，**系统态**和**空闲态**，分别表示CPU处于用户态执行的时间，系统内核执行的时间，和空闲系统进程执行的时间。平时所说的CPU利用率是指：**CPU执行非系统空闲进程的时间 / CPU总的执行时间**。

在Linux的内核中，有一个全局变量：Jiffies。 Jiffies代表时间。它的单位随硬件平台的不同而不同。系统里定义了一个常数HZ，代表每秒种最小时间间隔的数目。这样jiffies的单位就是 1/HZ。Intel平台jiffies的单位是1/100秒，这就是系统所能分辨的最小时间间隔了。每个CPU时间片，Jiffies都要加1。 CPU的利用率就是用执行用户态+系统态的Jiffies除以总的Jifffies来表示。

在Linux系统中，可以用/proc/stat文件来计算cpu的利用率(详细的解释可参考：http: //[www.linuxhowtos.org/System/procstat.htm)。这个文件包含了所有CPU活动的信息，该文件中的所有值都是从系统启动开始累计到当前时刻。](http://www.linuxhowtos.org/System/procstat.htm)%E3%80%82%E8%BF%99%E4%B8%AA%E6%96%87%E4%BB%B6%E5%8C%85%E5%90%AB%E4%BA%86%E6%89%80%E6%9C%89CPU%E6%B4%BB%E5%8A%A8%E7%9A%84%E4%BF%A1%E6%81%AF%EF%BC%8C%E8%AF%A5%E6%96%87%E4%BB%B6%E4%B8%AD%E7%9A%84%E6%89%80%E6%9C%89%E5%80%BC%E9%83%BD%E6%98%AF%E4%BB%8E%E7%B3%BB%E7%BB%9F%E5%90%AF%E5%8A%A8%E5%BC%80%E5%A7%8B%E7%B4%AF%E8%AE%A1%E5%88%B0%E5%BD%93%E5%89%8D%E6%97%B6%E5%88%BB%E3%80%82)

如：

1. [sailorhzr@builder ~]$ cat /proc/stat
2. cpu 432661 13295 86656 422145968 171474 233 5346
3. cpu0 123075 2462 23494 105543694 16586 0 4615
4. cpu1 111917 4124 23858 105503820 69697 123 371
5. cpu2 103164 3554 21530 105521167 64032 106 334
6. cpu3 94504 3153 17772 105577285 21158 4 24
7. intr 1065711094 1057275779 92 0 6 6 0 4 0 3527 0 0 0 70 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7376958 0 0 0 0 0 0 0 1054602 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8. ctxt 19067887
9. btime 1139187531
10. processes 270014
11. procs\_running 1
12. procs\_blocked 0

**输出解释**

CPU 以及CPU0、CPU1、CPU2、CPU3每行的每个参数意思（以第一行为例）为：

|  |  |
| --- | --- |
| 参数 | 解释 |
| user (432661)   nice (13295)   system (86656)  idle (422145968)   iowait (171474)   irq (233)  softirq (5346) | 从系统启动开始累计到当前时刻，用户态的CPU时间（单位：jiffies） ，不包含 nice值为负进程。1jiffies=0.01秒 从系统启动开始累计到当前时刻，nice值为负的进程所占用的CPU时间（单位：jiffies） 从系统启动开始累计到当前时刻，核心时间（单位：jiffies） 从系统启动开始累计到当前时刻，除硬盘IO等待时间以外其它等待时间（单位：jiffies） 从系统启动开始累计到当前时刻，硬盘IO等待时间（单位：jiffies） ， 从系统启动开始累计到当前时刻，硬中断时间（单位：jiffies） 从系统启动开始累计到当前时刻，软中断时间（单位：jiffies） |

**CPU时间=user+system+nice+idle+iowait+irq+softirq**

“intr”这行给出中断的信息，第一个为自系统启动以来，发生的所有的中断的次数；然后每个数对应一个特定的中断自系统启动以来所发生的次数。

“ctxt”给出了自系统启动以来CPU发生的上下文交换的次数。

“btime”给出了从系统启动到现在为止的时间，单位为秒。

“processes (total\_forks) 自系统启动以来所创建的任务的个数目。

“procs\_running”：当前运行队列的任务的数目。

“procs\_blocked”：当前被阻塞的任务的数目。

那么CPU利用率可以使用以下两个方法。先取两个采样点，然后计算其差值：

1. cpu usage=(idle2-idle1)/(cpu2-cpu1)\*100
2. cpu usage=[(user\_2 +sys\_2+nice\_2) - (user\_1 + sys\_1+nice\_1)]/(total\_2 - total\_1)\*100

以下用分别用bash和perl做的一个cpu利用率的计算：

本人注:以下代码则采用公式为:

1. total\_0USER[0]+NICE[0]+SYSTEM[0]+IDLE[0]+IOWAIT[0]+IRQ[0]+SOFTIRQ[0]
2. total\_1=USER[1]+NICE[1]+SYSTEM[1]+IDLE[1]+IOWAIT[1]+IRQ[1]+SOFTIRQ[1]
3. cpu usage=(IDLE[0]-IDLE[1]) / (total\_0-total\_1) \* 100

**###bash 代码**

1. CODE:#!/bin/sh
3. ##echo user nice system idle iowait irq softirq
4. CPULOG\_1=$(cat /proc/stat | grep 'cpu ' | awk '{print $2" "$3" "$4" "$5" "$6" "$7" "$8}')
5. SYS\_IDLE\_1=$(echo $CPULOG\_1 | awk '{print $4}')
6. Total\_1=$(echo $CPULOG\_1 | awk '{print $1+$2+$3+$4+$5+$6+$7}')
8. sleep 5
10. CPULOG\_2=$(cat /proc/stat | grep 'cpu ' | awk '{print $2" "$3" "$4" "$5" "$6" "$7" "$8}')
11. SYS\_IDLE\_2=$(echo $CPULOG\_2 | awk '{print $4}')
12. Total\_2=$(echo $CPULOG\_2 | awk '{print $1+$2+$3+$4+$5+$6+$7}')
14. SYS\_IDLE=`expr $SYS\_IDLE\_2 - $SYS\_IDLE\_1`
16. Total=`expr $Total\_2 - $Total\_1`
17. SYS\_USAGE=`expr $SYS\_IDLE/$Total\*100 |bc -l`
19. SYS\_Rate=`expr 100-$SYS\_USAGE |bc -l`
21. Disp\_SYS\_Rate=`expr "scale=3; $SYS\_Rate/1" |bc`
22. echo $Disp\_SYS\_Rate%

**###perl 代码**

1. #!/usr/bin/perl
3. use warnings;
5. $SLEEPTIME=5;
7. if (-e "/tmp/stat") {
8. unlink "/tmp/stat";
9. }
10. open (JIFF\_TMP, ">>/tmp/stat") || die "Can't open /proc/stat file!/n";
11. open (JIFF, "/proc/stat") || die "Can't open /proc/stat file!/n";
12. @jiff\_0=;
13. print JIFF\_TMP $jiff\_0[0] ;
14. close (JIFF);
16. sleep $SLEEPTIME;
18. open (JIFF, "/proc/stat") || die "Can't open /proc/stat file!/n";
19. @jiff\_1=;
20. print JIFF\_TMP $jiff\_1[0];
21. close (JIFF);
22. close (JIFF\_TMP);
24. @USER=`awk '{print /$2}' "/tmp/stat"`;
25. @NICE=`awk '{print /$3}' "/tmp/stat"`;
26. @SYSTEM=`awk '{print /$4}' "/tmp/stat"`;
27. @IDLE=`awk '{print /$5}' "/tmp/stat"`;
28. @IOWAIT=`awk '{print /$6}' "/tmp/stat"`;
29. @IRQ=`awk '{print /$7}' "/tmp/stat"`;
30. @SOFTIRQ=`awk '{print /$8}' "/tmp/stat"`;
32. $JIFF\_0=$USER[0]+$NICE[0]+$SYSTEM[0]+$IDLE[0]+$IOWAIT[0]+$IRQ[0]+$SOFTIRQ[0];
33. $JIFF\_1=$USER[1]+$NICE[1]+$SYSTEM[1]+$IDLE[1]+$IOWAIT[1]+$IRQ[1]+$SOFTIRQ[1];
35. $SYS\_IDLE=($IDLE[0]-$IDLE[1]) / ($JIFF\_0-$JIFF\_1) \* 100;
36. $SYS\_USAGE=100 - $SYS\_IDLE;
38. printf ("The CPU usage is %1.2f%%/n",$SYS\_USAGE);