Linux 内核实验报告

实验题目：**proc** 文件系统实验

学号：**200900301236**　　（辅修号：　）
日期：**2012.4.11**　　班级：**09** 软 **1**　　姓名：　王添枝
Email：**wangtianzhi@163.com**

实验目的：学习 **Linux** 内核、进程、存储和其他资源的一些重要特征。

硬件环境：微软实验室
软件环境：**gcc4.6.1**
ubuntu11.10
Linux version 3.0.0-12-generic
vi 7.3.154

实验步骤：

1、　设计说明

（1）问题 A：

若命令行的第二个参数是除-b、-c、-d 以外其它的以-开头的参数则执行此过程。

首先需要知道所需的信息在哪个文件的哪个位置才可以准确定位。

机器名字所在文件:/proc/sys/kernel/hostname

cpu 信息所在文件：/proc/cpuinfo

系统版本信息所在文件:/proc/version

在读出的文件里定位到需要的信息(机器名称、CPU 型号、内核版本)，比如 CPU 型号在 cpuinfo 里面是第五项，先空读 4 次，第五次读出的就是 CPU 型号，打印出来（要注意时间格式），其它几项同理。

（2）问题 B：

若命令行的第二个参数是-b 则执行此过程。

系统启动以来经历的时间所在文件：/proc/uptime，其中第一项是启动以来经历的时间，第二项是空闲时间。

（3）问题 C：

若命令行的第二个参数是-c 则执行此过程。

从文件/proc/stat 可以得到 CPU 花费在用户态、系统态和空闲态的时间；

从文件/proc/diskstats 可以得到系统接收到的磁盘请求；

在文件/proc/stat 中定位到内核执行的上下文转换的次数和从系统启动开始创建的进程数，其中上下文转换次数在 stat 中的名称为 ctxt，创建的进程数的名称为 processes.

（4）问题 D：

命令行的第二个参数为 -d；第三个参数为 interval 的值，表示间隔多长时间采集数据；第四个参数为 duration 的值，表示采集负载平衡总共执行的时间。

打开内存信息文件/proc/meminfo；

定位到计算机配置的内存数量；

定位到当前可用的内存数量；

平均负载列表（至上一分钟的平均数）；

每次循环先睡眠一段时间，唤醒后采样当前负载，然后再返回睡眠状态。

2、 调试记录

（1）问题 2

```
stu@stu-Lenovo:~/桌面$ ./exp1_proc -b

******PART B **********

启动以来经历的时间：00:03:10:50
```

（2）问题 3

```
stu@stu-Lenovo:~/桌面$ ./exp1_proc -c

******PART C**********

用户态时间:60122(0.01 秒） 系统态时间:13295(0.01 秒） 空闲态时间:4565750(0.01 秒)

系统接收到的磁盘请求 sda：读 18378 次，写 18221 次，总共 36599 次

上下文转换的次数:9730562

从系统启动开始创建的进程数:3988
```

（3）问题 4

先写一个循环的程序，让 cpu 持续运行，在运行问题 4 才能更好的反映负载均衡问题。

```
stu@stu-Lenovo:~/桌面$ ./exp1_proc -d 3 15
```

******PART D **********

计算机配置的内存大小:4023728

当前可用的内存大小:2000880

一分钟平均负载:

0.080000

0.160000

0.160000

0.220000

0.290000

（4）问题 1
stu@stu-Lenovo:~/桌面$ ./exp1_proc -

******PART A **********

Status report type Short at 12:52:49

Machine hostname: stu-Lenovo

CPU 型号: Intel(R) Core(TM)2 Quad CPU   Q9500 @ 2.83GHz

内核版本： Linux version 3.0.0-12-generic (buildd@crested) (gcc version 4.6.1 (Ub

结论分析与体会：

在实验过程中，我从网上查阅了/proc 文件系统的相关资料，/proc 文件系统提供了一个基于文件的 Linux 内部接口。老师已在实验要求中给出了一部分程序范例，根据这些前期准备进行了实验。实验过程中。实验目的基本完成，但是代码没有进行优化，可能运行性能比较低。通过本次试验，我了解了 Linux 内核、进程以及存储的一些重要特征，学习了一些在/proc 文件系统中文件操作的函数。

程序完整源代码：

```
/
*************************************************************************
**

    main.c  -  description

    begin                :4月/10日/2012年

    copyright            : (C) 2012 by wangtianzhi

    Function             : 观察linux内核行为

*************************************************************************
******/

#include <stdio.h>

#include <sys/time.h>

#define LB_SIZE 80

enum TYPE{STANDARD,SHORT,LONG};

FILE *thisProcFile;

  //Proc打开文件指针

struct timeval now;

  //系统时间日期

enum TYPE reportType;

 //观察报告类型

char repTypeName[16];

char *lineBuf;    //proc文件读出行缓冲

int interval;     //系统负荷监测时间间隔
```

```
int duration;    //系统负荷监测时段

int iteration;

char c1,c2;     //字符处理单元

void sampleLoadAvg(){ //观察系统负荷

     float m;

     iteration=0;

      printf("一分钟平均负载:\n");

     while(iteration<duration){

         thisProcFile=fopen("/proc/loadavg","r");

         sleep(interval);

         fgets(lineBuf,LB_SIZE+1,thisProcFile);

         rewind(thisProcFile);//将文件内部的位置指针重新指向一个流（数据流/文件）的开头

         sscanf(lineBuf,"%f",&m);

         printf("%f\n",m);

         fclose(thisProcFile);

         iteration+=interval;

     }

}


void sampleTime(){//观察系统启动时间

      int uptime,idletime;

      int  day,hour,minute,second;
```

```
int  i,j;
char temp[80];
i=j=0;
//打开计时文件
thisProcFile=fopen("/proc/uptime","r");
if(thisProcFile==NULL){
    printf("/proc/uptime file open failure");
    exit(1);
}

fscanf(thisProcFile,"%d%d",&uptime,&idletime);
//转换成日时钟秒
day=hour=minute=second=0;
day=uptime/86400;
i=uptime%86400;
hour=i/3600;
i=i%3600;
minute=i/60;
second=i%60;
//打印处理好的信息内容
  printf(" 启 动 以 来 经 历 的 时 间 ： %02d:%02d:%02d:%02d\n",day,hour,minute,second);
```

```
    fclose(thisProcFile);
}

int main(int argc,char *argv[])
{
      lineBuf = (char *)malloc(LB_SIZE+1);
     // printf("%d",sizeof(struct time_t));
     // now = malloc(80);
      reportType = STANDARD;
      strcpy(repTypeName,"Standard");

      if(argc >1)
         sscanf(argv[1],"%c%c",&c1,&c2);//取命令行选择符
      if(c1!='-'){     //提示本程序命令参数的用法
         exit(1);
      }
      if(c2 == 'b'){//观察部分B
          printf("******PART B **********\n");
          sampleTime();
      }
      else if(c2=='c'){//观察部分C
          printf("******PART C**********\n");
```

```
 //打开系统状态信息文件

thisProcFile = fopen("/proc/stat","r");

 //读出文件全部的内容

char c[512],d[30];

int usrtime,m,systime,idletime;

                                        fscanf(thisProcFile,"%s%d%d%d
%d",c,&usrtime,&m,&systime,&idletime);

printf("用户态时间:%d(0.01秒) 系统态时间:%d(0.01秒) 空闲态
时间:%d(0.01秒)\n",usrtime,systime,idletime);

 fclose(thisProcFile);



//打开磁盘状态信息文件

thisProcFile = fopen("/proc/diskstats","r");

int r,w;

while(fgets(c,512,thisProcFile)!=NULL)

{

        sscanf(c,"%d    %d    %s    %d    %d    %d    %d
%d",&m,&m,d,&r,&m,&m,&m,&w);

    if(d[0]=='s'){

        printf("系统接收到的磁盘请求 %s：读%d次，写%d次，总
共%d次\n",d,r,w,r+w);

        break;

    }

}
```

```
    fclose(thisProcFile);
    //打开文件，读取上下文切换和进程信息
    thisProcFile = fopen("/proc/stat","r");
    while(fgets(c,512,thisProcFile)!=NULL){
        sscanf(c,"%s %d",&d,&m);
        if(strcmp(d,"ctxt")==0)
            printf("上下文转换的次数:%d\n",m);
        if(strcmp(d,"processes")==0)
            printf("从系统启动开始创建的进程数:%d\n",m);
    }
    fclose(thisProcFile);
}
else if(c2 == 'd'){ //观察部分 D
     printf("******PART D **********\n");
     if(argc<4){
         printf("usage:observer [-b] [-c][-d int dur]\n");
         exit(1);
     }
    interval = atoi(argv[2]);//将 string 转换成 int
    duration = atoi(argv[3]);
     reportType = LONG;
     strcpy(repTypeName,"Long");
```

```
    int i;
    char d[30];
    //打开内存信息文件
    thisProcFile = fopen("/proc/meminfo","r");
    while(fgets(lineBuf,LB_SIZE+1,thisProcFile)!=NULL){
        sscanf(lineBuf,"%s %d",&d,&i);
        if(strcmp(d,"MemTotal:")==0)
            printf("计算机配置的内存大小:%d\n",i);
        if(strcmp(d,"MemFree:")==0)
            printf("当前可用的内存大小:%d\n",i);
    }
    fclose(thisProcFile);
    //查看负载均衡
    sampleLoadAvg();
}
else{//观察部分 A
     int stoday;
printf("******PART A **********\n");
     reportType = SHORT;
     strcpy(repTypeName,"Short");
     //取出并显示系统当前时间
    gettimeofday(&now, NULL );
```

```
            //  now=   time(NULL);

               // printf();

        stoday = now.tv_sec%(3600*24);

            printf("Status report type %s at %02d:%02d:
%02d\n",repTypeName,stoday/3600,(stoday%3600)/60,

            (stoday%3600)%60);

            //读出并显示机器名

           thisProcFile = fopen("/proc/sys/kernel/hostname","r");

           if(thisProcFile==NULL){

              printf("Source file can't open");

              exit(1);

           }

           fgets(lineBuf,LB_SIZE+1,thisProcFile);

           printf( "Machine hostname: %s",lineBuf);

            fclose(thisProcFile);

            //读出并显示全部CPU信息

           thisProcFile=fopen("/proc/cpuinfo","r");//打开cpuinfo文件

           if(thisProcFile==NULL){

              printf("Source file can't open");

               exit(1);

           }


           //获取需要显示的信息
```

```
        int i;
        for(i=0;i<5;i++)
            fgets(lineBuf,LB_SIZE+1,thisProcFile);
        printf("CPU 型号: %s",lineBuf+13);
        fclose(thisProcFile);

         //读出并显示系统版本信息
        thisProcFile=fopen("/proc/version","r");//打开 version 文件
        if(thisProcFile==NULL){
            printf("Source file can't open'");
            exit(0);
        }
        fgets(lineBuf,LB_SIZE+1,thisProcFile);
        printf("内核版本：%s",lineBuf);

         fclose(thisProcFile);

      }
    return 0;
}
```

参考材料

linux 内核设计与实现