实验7 设备管理

1. 实验目的

1、了解Linux下的设备管理情况以及磁盘运行情况。

2、学会编程实现磁盘调度算法。

1. 预备知识
2. **iostat命令**

被用于监视系统输入输出设备和CPU的使用情况。它的特点是汇报磁盘活动统计情况，同时也会汇报出CPU使用情况。

* 语法：

iostat [ 选项 ] [ <时间间隔> [ <次数> ] ]

* 选项：

-c：仅显示CPU使用情况

-d：仅显示设备利用率

-k：显示状态以千字节每秒为单位，而不使用块每秒

-m：显示状态以兆字节每秒为单位

-p：仅显示块设备和所有被使用的其他分区的状态

-t：显示每个报告产生时的时间

-V：显示版号并退出

-x：显示扩展状态

* 参数：

间隔时间：每次报告的间隔时间（秒）

次数：显示报告的次数

* 字段说明：

%user：应用程序使用CPU的时间占比

%nice：拥有高优先级的应用程序使用CPU的时间占比

%system：内核程序使用CPU的时间占比

%iowait：表示等待进行 I/O 所使用 CPU 的时间百分比

%steal ： 显示虚拟机管理器在服务另一个虚拟处理器时虚拟CPU处在非自愿等待下花费时间的百分比

%idle：显示 CPU 的空闲时间

Device：监测设备名称

rrqm/s：每秒需要读取需求的数量

wrqm/s：每秒需要写入需求的数量

r/s ：每秒实际读取需求的数量

w/s：每秒实际写入需求的数量

rkB/s：每秒实际读取的大小，单位为KB

wkB/s：每秒实际写入的大小，单位为KB

avgrq-sz：平均每次设备I/O操作的数据大小

avgqu-sz：平均I/O队列长度

await：平均每次设备I/O操作的等待时间

svctm：平均每次设备I/O操作的服务时间

%util：被I/O操作消耗的CPU百分比

1. **stdlib.h和time.h**

* malloc

函数原型： void \* malloc(unsigned size);

函数功能:：分配size字节的存储区

函数返回:：所分配的内存区地址,如果内存不够,返回0

头文件：stdlib.h

* srand

函数原型：void srand(unsigned seed);

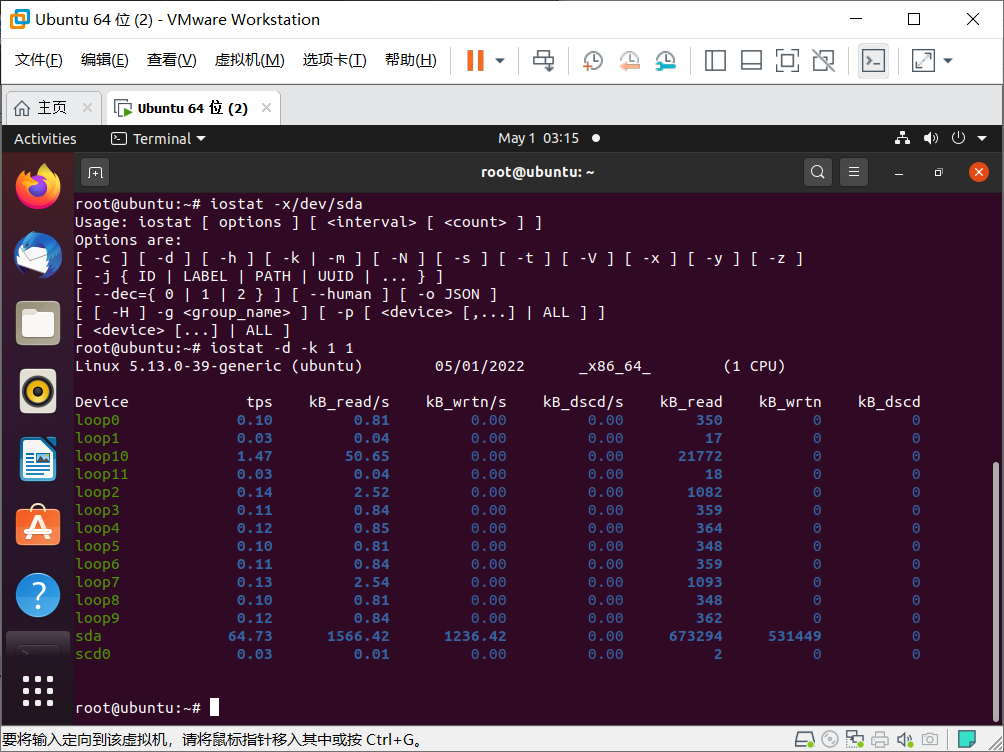
函数功能:：产生随机数

函数用法:：参数seed是整数，通常可以利用time(0)或geypid(0)的返回值作为seed

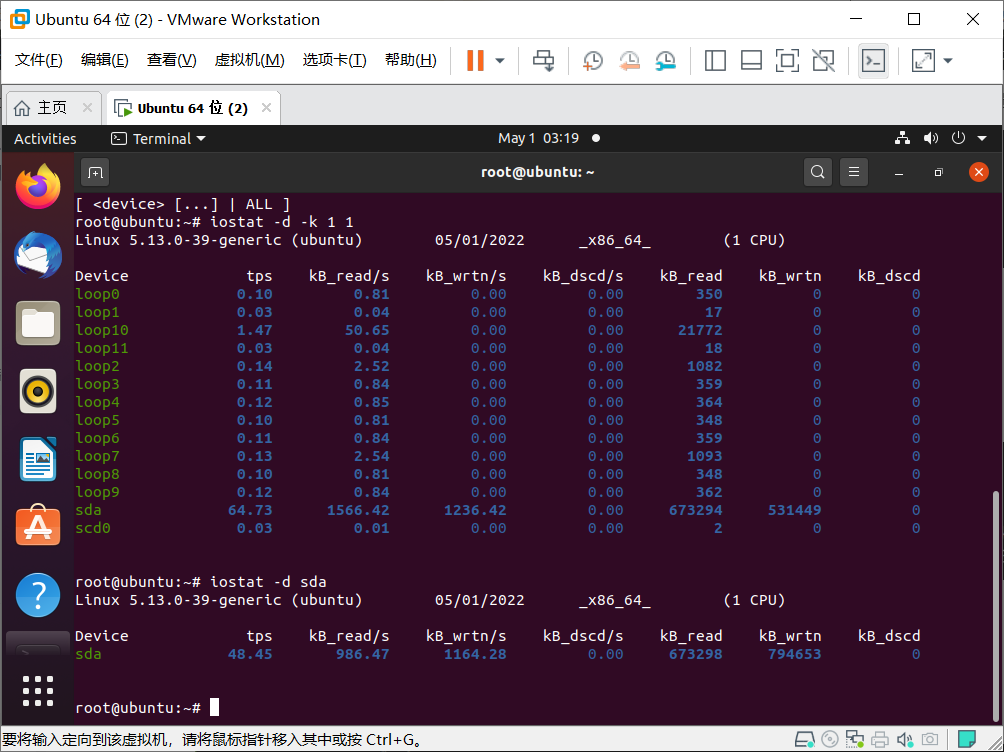
头文件：stdlib.h

1. 实验内容
2. iostat的使用

* iostat -d -k 1 1  
  查看TPS和吞吐量信息



* iostat -d sda 显示指定硬盘信息



1. 查看linux磁盘调度算法的命令

# cat /sys/block/sda/queue/scheduler

1. 磁盘调度最短寻道时间优先算法（SSTF）和循环扫描算法（CSCAN）

#include<stdio.h>

#include <stdlib.h>

#include<time.h>

void SSTF(int a[],int n);

void CSCAN(int a[],int n);

int main()

{

int n;//要访问的磁道个数

int s;//功能号

printf("请输入要访问的磁道个数：");

scanf("%d",&n);

int \*a=(int\*)malloc(n\*sizeof(int));//动态分配n个整型储存单元，首地址赋给a,即数组a的元素个数是程序进行时才确定的。

printf("生成随机磁道号:");

srand((unsigned)time(NULL));//随机种子

for(int i=0;i<n;i++)

{

a[i]=(rand()%100)+1;//产生一个1~100以内的正整数

printf("%d ",a[i]);

}

printf("\n\n");

while(1)

{

printf("1、最短寻道时间算法（SSTF）\n");

printf("2、循环扫描算法（CSCAN）\n");

printf("0、退出\n");

printf("请选择功能号:");

scanf("%d",&s);

if(s<0||s>6)

{

printf("输入有误!\n");

}

else switch(s)

{

case 1:SSTF(a, n);break;

case 2:CSCAN(a,n);break;

case 0: exit(0);

default: break;

}}

return 0;

}

//最短寻道时间算法（SSTF）

void SSTF(int a[],int n)

{

int temp;

int now,l,r;

int i,j,sum=0,k=1;

float aver=0;

//将磁道号按递增排序

for(i=0;i<n;i++)

for(j=i+1;j<n;j++)

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

printf("按递增顺序排好的磁道：\n");

for( i=0;i<n;i++)

{

printf("%d ",a[i]);//输出排好的磁道顺序

}

printf("\n请输入当前的磁道号：");

scanf("%d",&now);//确定当前磁头所在位置

printf("磁盘调度顺序为：\n");

if(a[n-1]<=now)//当前磁头位置大于最外围欲访问磁道

{

for(i=n-1;i>=0;i--)

printf("%d ",a[i]);

sum=now-a[0];

}

else if(a[0]>=now)//当前磁头位置小于最里欲访问磁道

{

for(i=0;i<n;i++)

printf("%d ",a[i]);

sum=a[n-1]-now;

}

else

{

while(a[k]<now)//确定当前磁道在已排的序列中的位置

{

k++;

}

l=k-1;//在磁头位置的前一个欲访问磁道

r=k;//磁头欲访问磁道

while((l>=0)&&(r<n))

{

if((now-a[l])<=(a[r]-now))//选择离磁头近的磁道

{

printf("%d ",a[l]);

sum+=now-a[l];

now=a[l];

l=l-1;

}

else

{

printf("%d ",a[r]);

sum+=a[r]-now;

now=a[r];

r=r+1;

}

}

if(l=-1)//磁头位置里侧的磁道已访问完

{

for(j=r;j<n;j++)//访问磁头位置外侧的磁道

{

printf("%d ",a[j]);

}

sum+=a[n-1]-a[0];

}

if(r==n)//磁头位置外侧的磁道已访问完

{

for(j=k-1;j>-1;j--) //访问磁头位置里侧的磁道

{

printf("%d",a[j]);

}

sum+=a[n-1]-a[0];

}

}

aver=((float) sum)/n;//求平均寻道次数

printf("\n移动的总道数:%d\n平均寻道长度:%0.2f\n",sum,aver);

}

//循环扫描算法（CSCAN）

void CSCAN(int a[],int n)

{

int temp;

int now,l,r;

int i,j,sum=0,k=1;

float aver=0;

for(i=0;i<n;i++)//对访问磁道按由小到大顺序排列输出

for(j=i+1;j<n;j++)

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

printf("按递增顺序排好的磁道：\n");

for( i=0;i<n;i++)

{

printf("%d ",a[i]);

}

printf("\n请输入当前的磁道号：");

scanf("%d",&now);//确定当前磁道号

if(a[n-1]<=now)//磁头位置大于最外围欲访问磁道

{

printf("磁盘调度顺序为：");

for(i=n-1;i>=0;i--)

printf("%d ",a[i]);

sum=now-2\*a[0]+a[n-1];

}

else if(a[0]>=now)//磁头位置小于最里欲访问磁道

{

printf("磁盘调度顺序为：");

for(i=0;i<n;i++)

printf("%d ",a[i]);

sum=a[n-1]-now;

}

else //磁头位置在最里侧磁道与最外侧磁道之间

{

int d;

while(a[k]<now)

{

k++;

}

l=k-1;//在磁头位置的前一个欲访问磁道

r=k; //磁头欲访问磁道

printf("请输入当前磁头移动的方向 (0 表示向内 ，1表示向外) : ");

scanf("%d",&d); //确定磁头访问的方向

printf("磁盘调度顺序为：");

if(d==0||d==1)

{

if(d==1) //磁头向外侧访问

{

for(j=r;j<n;j++)//先访问外侧磁道再转向最里欲访问磁道

{

printf("%d ",a[j]);

}

for(j=0;j<r;j++)

{

printf("%d ",a[j]);

}

sum=2\*a[n-1]-now-2\*a[0]+a[l];

}

if(d==0) //先访问内侧磁道再转向最外欲访问磁道

{

for(j=r-1;j>=0;j--)

{

printf("%d ",a[j]);

}

for(j=n-1;j>=r;j--)

{

printf("%d ",a[j]);

}

sum=2\*a[n-1]-2\*a[0]+now-a[r];

}

}

else

printf("请输入0或1!");

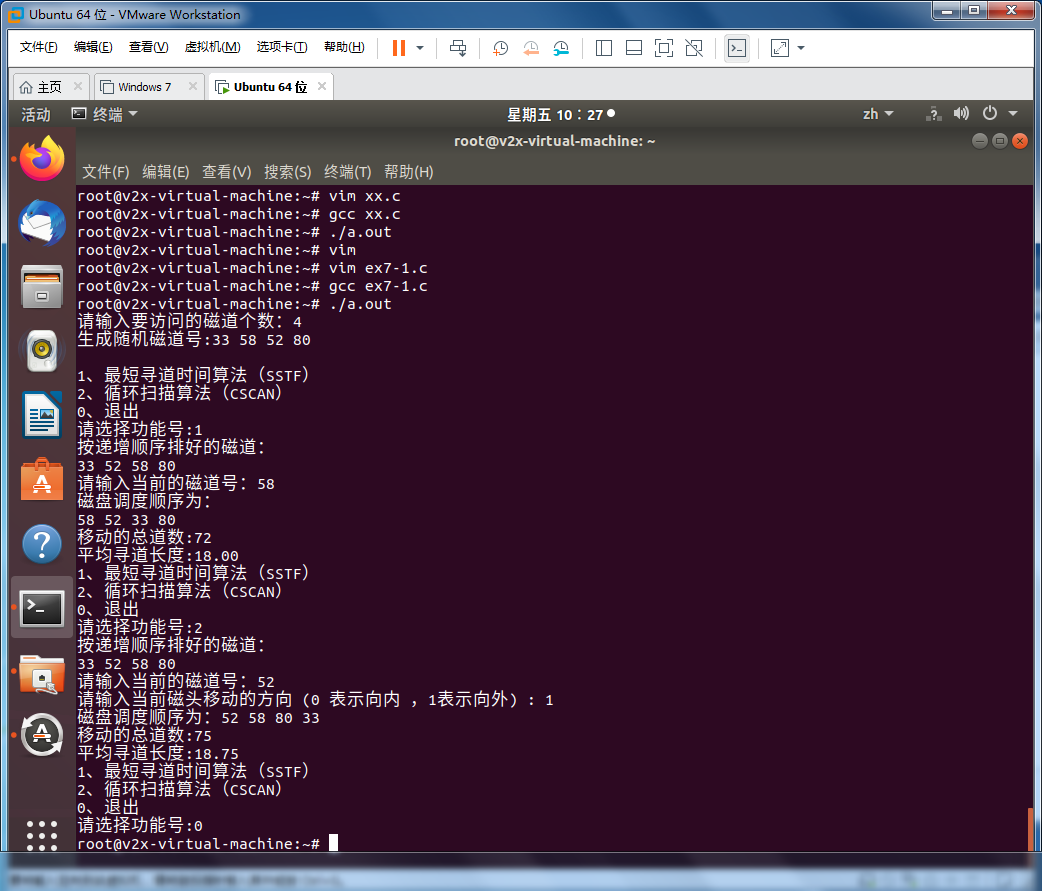
}

aver=((float) sum)/n;//求平均寻道次数

printf("\n移动的总道数:%d\n平均寻道长度:%0.2f\n",sum,aver);

}

运行结果示例：



1. 实验题目
2. 利用iostat命令完成以下功能，然后查看结果并截图。

* 报告每秒向终端读取和写入的字符数。
* 查看设备使用率（%util）、响应时间（await）

1. 编写先来先服务算法（FCFS）和扫描算法（SCAN）

五、实验报告

将调试运行成功的程序和写好的实验报告一起压缩打包，以实验X-学号-姓名.rar这样的形式命名，并上传提交。