上海工程技术大学

实 验 报 告

实验四　内存管理

1. 实验要求

（1）通过在 Linux 环境下对内存管理的基本操作了解 Linux 如何对内存及虚拟内存进行管理；

（2）掌握可变分区管理内存的方法。

2. 实验内容

1） 验证实验：

① Linux 命令 FREE 显示内存状态情况，观察结果并分析；

② 用 VMSTAT 命令监视虚拟内存使用情况，观察结果并分析；

③ 使用 PS 和 KILL 命令回收内存，观察结果并分析（例如：可以打开两个终端，一个终端先运行无限循环程序，然后再另一个终端 KILL 该进程）；

④ 使用 SYNC 命令将内存缓冲区的数据写入磁盘;

⑤ 观察 ULIMIT 命令结果，写出用法和作用。

2） 编程实验，编写程序实现采用可变分区方法管理内存：

① 设计实现用来记录主存使用情况的数据结构：已分区表和空闲分区表或链表；

② 在设计好的数据结构上设计实现循环首次适应算法；

③ 在设计好的数据结构上设计实现主存回收算法。其中，若回收的分区有上邻空闲分区和（或）下邻空闲分区，要求合并为一个空闲分区登记在空闲分区表的一个表项里。

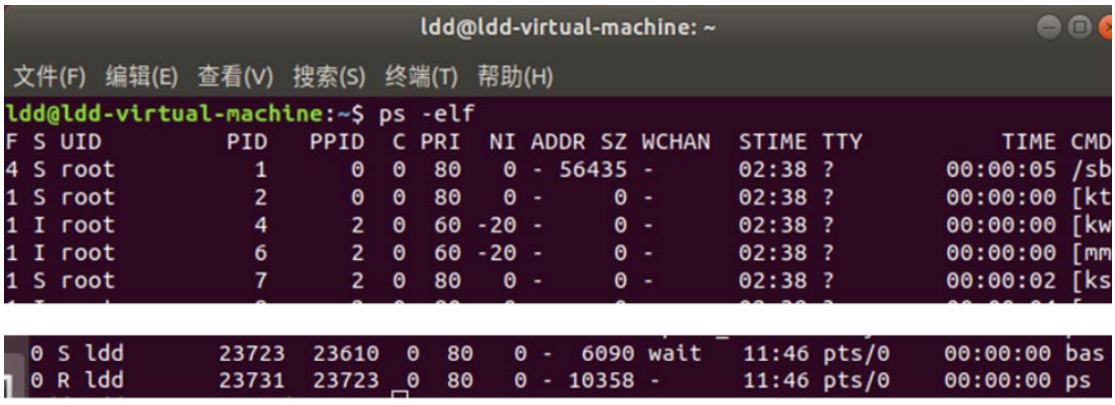
3. 实验结果

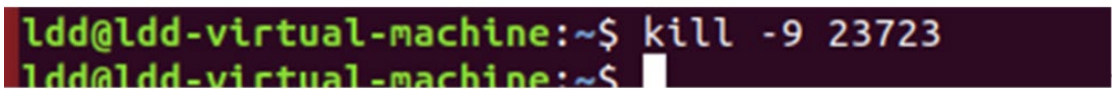
1) 验证实验：

① Linux 命令 FREE 显示内存状态情况，观察结果并分析；

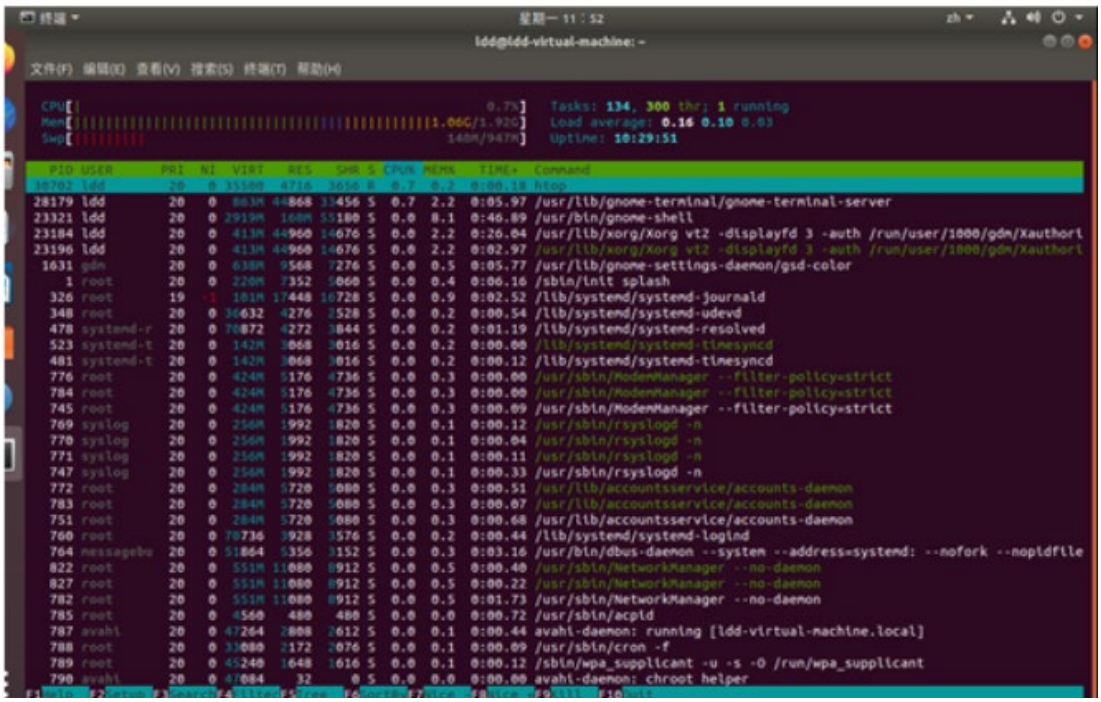
② 用 VMSTAT 命令监视虚拟内存使用情况，观察结果并分析；

③ 使用 PS 和 KILL 命令回收内存，观察结果并分析（例如：可以打开两个终端，一个终端先运行无限循环程序，然后再另一个终端 KILL 该进程）；

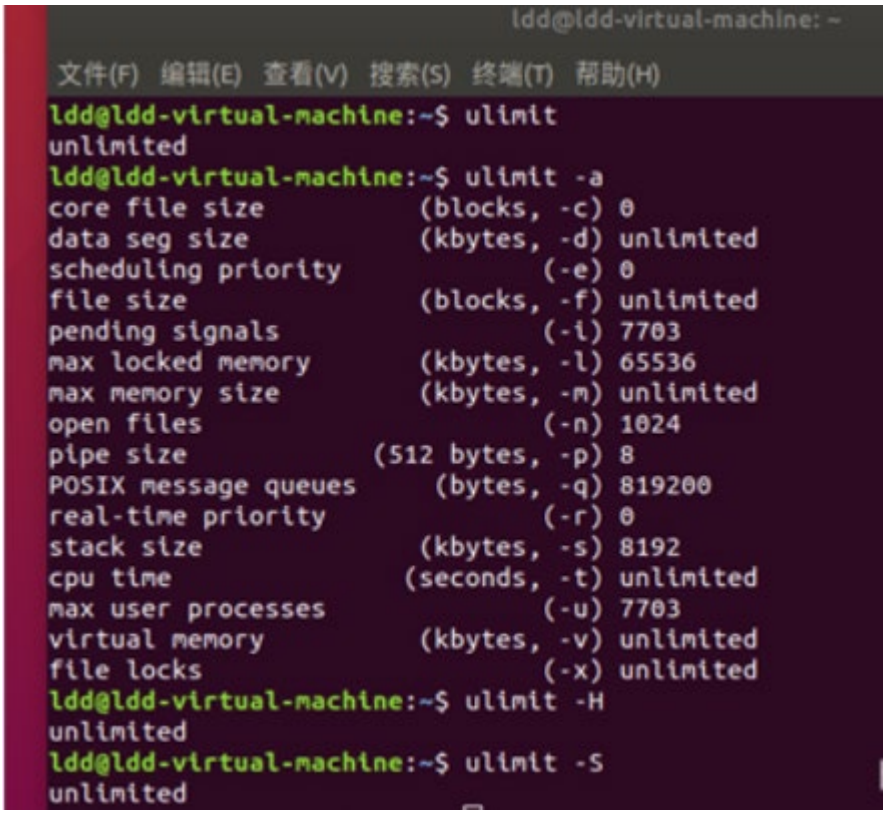


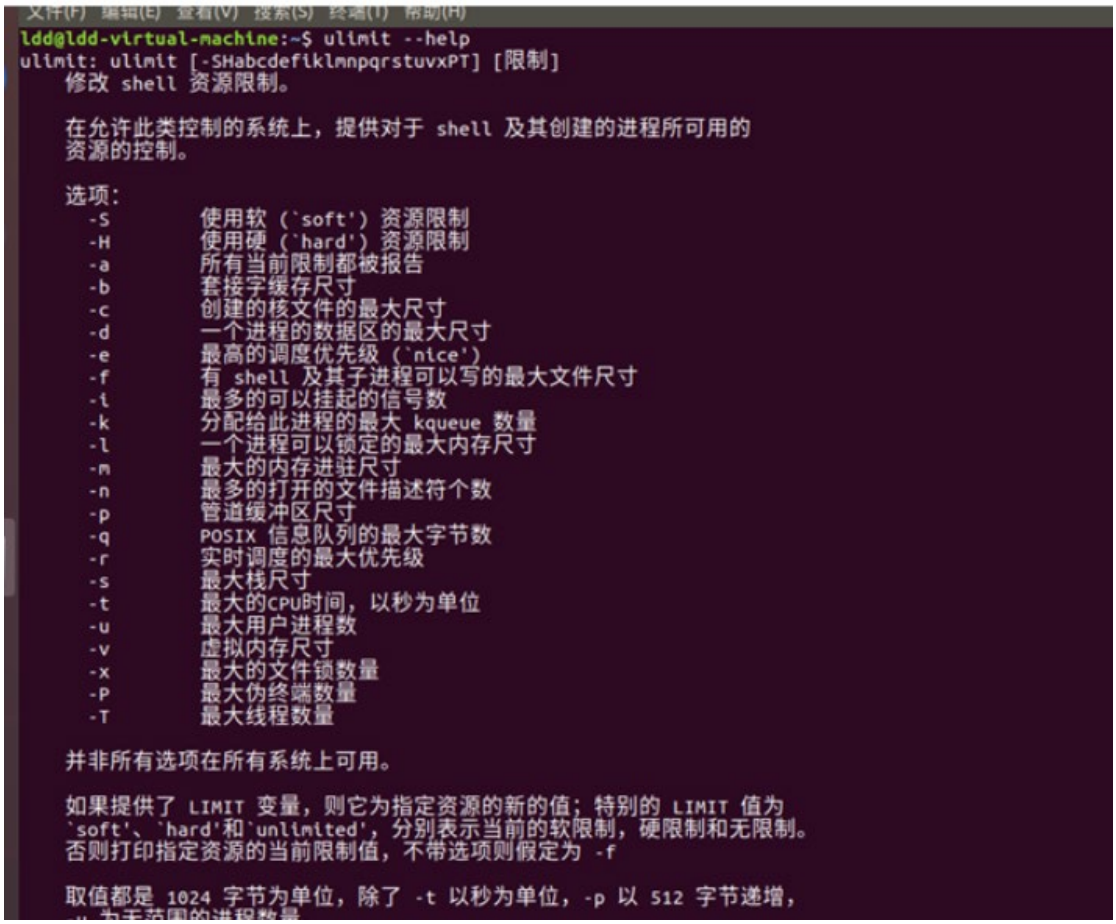


④ 使用 SYNC 命令将内存缓冲区的数据写入磁盘;



⑤ 观察 ULIMIT 命令结果，写出用法和作用。



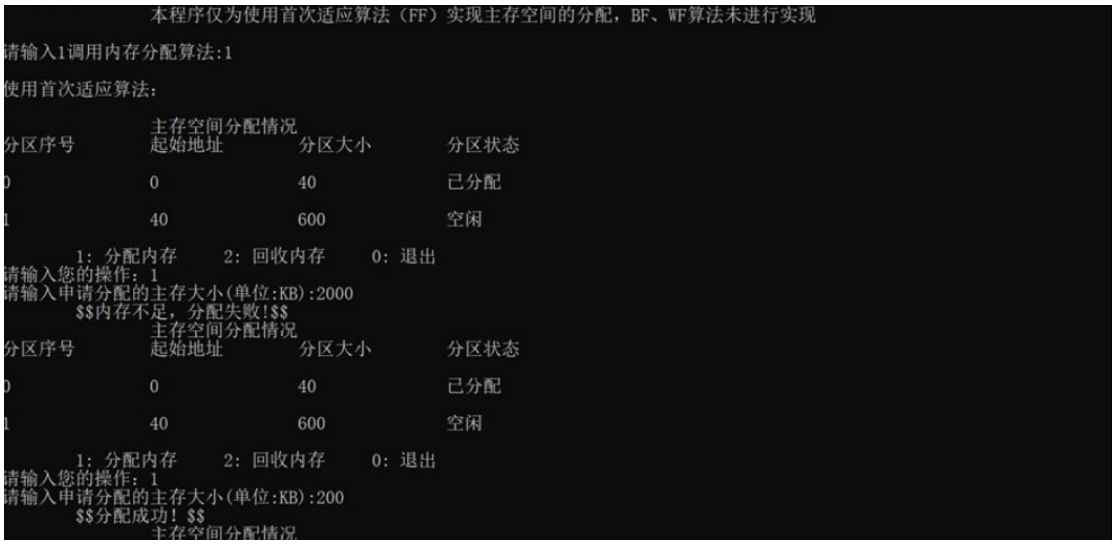


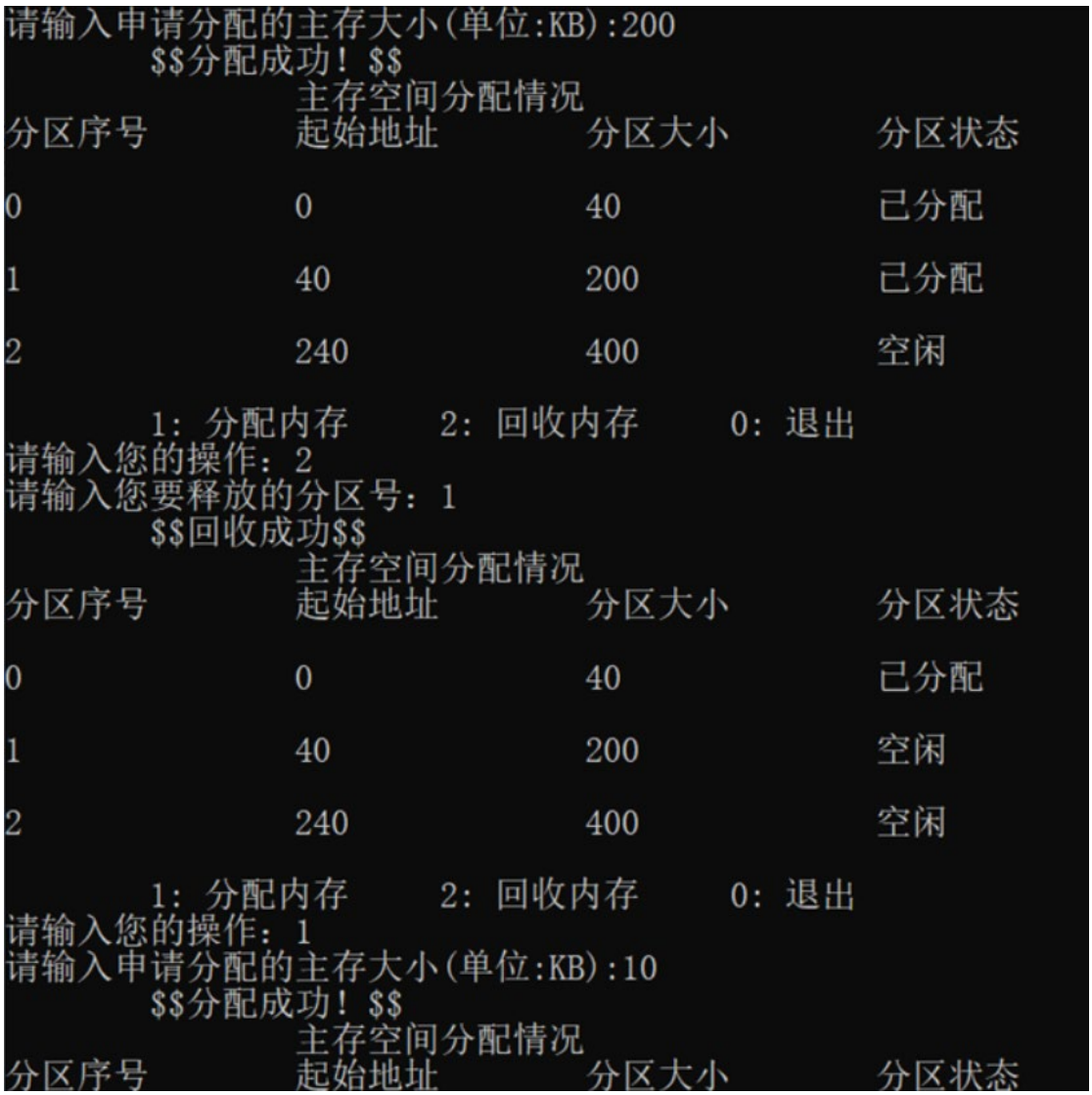
编程实验核心代码：

```
void show()
{
    int flag = 0;//用来记录分区序号
    Node* p = first;
    p->data.num = 0;
    p->data.address = 0;
    p->data.length = 40;
    p->data.state = 1;
    sort();
    printf("\n\t\t 主存空间分配情况\n");
    printf("分区序号\t 起始地址\t 分区大小\t 分区状态\n\n");
    while (p)
    {
        printf("%d\t\t%d\t\t%d", p->data.num, p->data.address, p->data.length);
        if (p->data.state == 0)
            printf("\t\t 空闲\n\n");
        else
            printf("\t\t 已分配\n\n");
        p = p->next;
    }
```

```
}
Status First_fit(int request) {//为申请作业开辟新空间且初始化
    Node* p = first->next;
    LinkList temp = (LinkList)malloc(sizeof(Node));
    temp->data.length = request;
    temp->data.state = 1;
    p->data.num = 1;
    while (p)
    {
        if ((p->data.state == 0) && (p->data.length == request))
        {//有大小恰好合适的空闲块
            p->data.state = 1;
            return OK;
            break;
        }
        else if ((p->data.state == 0) && (p->data.length > request))
        {//有空闲块能满足需求且有剩余
            temp->prior = p->prior;
            temp->next = p;
            temp->data.address = p->data.address;
            temp->data.num = p->data.num;
            p->prior->next = temp;
            p->prior = temp;
            p->data.address = temp->data.address + temp->data.length;
            p->data.length -= request;
            p->data.num += 1;
            return OK;
            break;
        }
        p = p->next;
    }
    return ERROR;
```

实验结果：


本程序仅为使用首次适应算法（FF）实现主存空间的分配，BF、WF算法未进行实现
请输入1调用内存分配算法:1
使用首次适应算法:
主存空间分配情况
分区序号 起始地址 分区大小 分区状态
0 0 40 已分配
1 40 600 空闲
1：分配内存 2：回收内存 0：退出
请输入您的操作：1
请输入申请分配的主存大小(单位:KB):2000
$$内存不足，分配失败!$$
主存空间分配情况
分区序号 起始地址 分区大小 分区状态
0 0 40 已分配
1 40 600 空闲
1：分配内存 2：回收内存 0：退出
请输入您的操作：1
请输入申请分配的主存大小(单位:KB):200
$$分配成功！$$
主存空间分配情况


请输入申请分配的主存大小(单位:KB):200
$$分配成功！$$
主存空间分配情况
分区序号 起始地址 分区大小 分区状态
0 0 40 已分配
1 40 200 已分配
2 240 400 空闲
1：分配内存 2：回收内存 0：退出
请输入您的操作：2
请输入您要释放的分区号：1
$$回收成功$$
主存空间分配情况
分区序号 起始地址 分区大小 分区状态
0 0 40 已分配
1 40 200 空闲
2 240 400 空闲
1：分配内存 2：回收内存 0：退出
请输入您的操作：1
请输入申请分配的主存大小(单位:KB):10
$$分配成功！$$
主存空间分配情况
分区序号 起始地址 分区大小 分区状态

4. 实验小结