实 验 报 告

课程名称：操作系统实验

实 验 四：操作系统资源使用状态监控

班　　级：02 班

学生姓名：白文强

学　　号：20191060064

专　　业：计算机科学与技术

指导教师：杨旭涛

学　　期：2021－2022 学年秋季学期

成　　绩：

云南大学信息学院

一、实验目的

1、掌握操作系统中的文件目录结构，重点掌握 proc 目录下的文件结构；

2、掌握操作系统的内存、CPU、磁盘等状态信息的查询方式，以及状态信息中的字段含义；

3、掌握 vmstat 工具的基本使用方法，能够通过编程实现资源的动态申请及状态观察贺和验证。通过 bonnie++基准测试程序，对操作系统进程测试，并分析结果。

二、知识要点

1、proc 文件结构和内容；

2、系统的主要状态信息查看方式，及其含义；

3、vmstat 系统状态监控程序的使用；

4、bonnie++系统基准测试程序的测试和验证。

三、实验预习（要求做实验前完成）

1、了解 linux 系统中常用命令的使用方法；

2、掌握 proc 文件内容，系统状态的内容；

3、熟悉 vmstat 工具；

4、熟悉 bonnie++基准测试工具。

四、实验内容和试验结果

结合课程所讲授内容以及课件中的试验讲解，完成以下试验。请分别描述程序的流程，附上源代码，并将试验结果截图附后。

1、编写 shell 脚本（命名为 meminfo.sh），定时读取/proc/meminfo 中的 MemTotal、MemFree、Active 等信息。编写一个负载程序，不断向操作系统申请内存，观察负载程序运行时 meminfo 中这三个数据的变化，并使用 excel 等工具绘制折线图。贴出负载程序的源码，并给出运行这个负载程序时的 MemFree 和 Active 的变化曲线。

负载程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
int main()
{
```

```
    char *p;
    while(1){
        p = (char*) malloc(1024*sizeof(char));
        memset(p, 1, 1024);
        sleep(2);
    }
    return 0;
}
```

Shell 脚本：

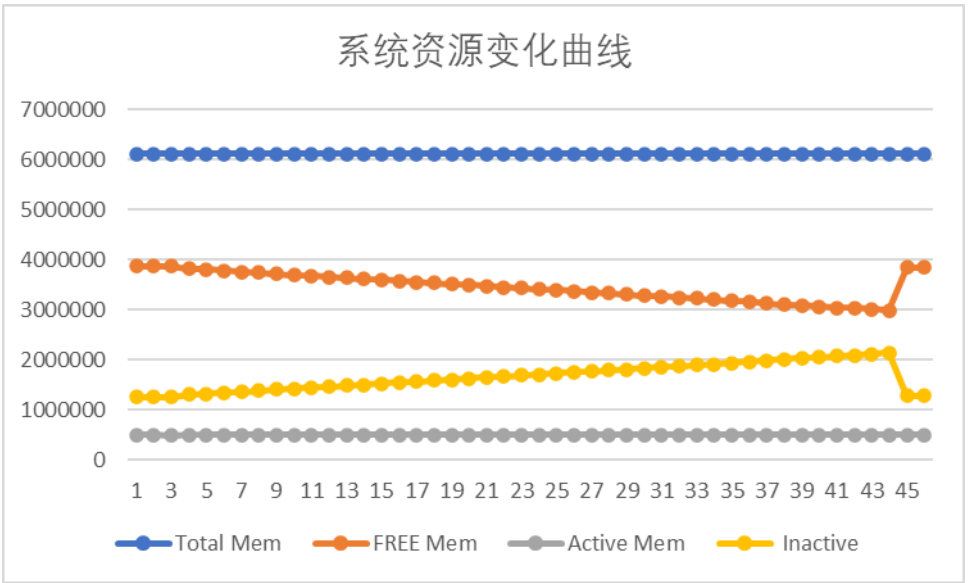
```
memfile="/proc/meminfo"

getmeminfo(){
    a=`more $memfile`
    FREE=`echo "$a"|awk 'NR==2{print $2}'`
    PHYMEM=`echo "$a"|awk 'NR==1{print $2}'`
    ACTIVE=`echo "$a"|awk 'NR==7{print $2}'`
    Inactive=`echo "$a"|awk 'NR==8{print $2}'`
}
while true
do
    getmeminfo
    echo "Total Mem="$PHYMEM",FREE Mem="$FREE",Active Mem="$ACTIVE",
Inactive="$Inactive""
    sleep 2
done
```

FREE Mem 、 Inactive Mem、ActiveMem 的变化

Total Mem	FREE Mem	Active Mem	Inactive
6111920	3863368	491960	1252160
6111920	3862596	491960	1252328
6111920	3862596	491956	1252608
6111920	3818708	491992	1296156
6111920	3798044	491992	1316620
6111920	3777624	492000	1337224
6111920	3756960	492000	1357708
6111920	3736044	492000	1378192
6111920	3715380	492008	1398676
6111920	3694456	492008	1419208
6111920	3674044	492016	1439708
6111920	3653884	492016	1460204
6111920	3633716	492020	1480688
6111920	3613288	492028	1501176
6111920	3592624	492028	1521660
6111920	3571960	492036	1542144
6111920	3551548	492036	1562640

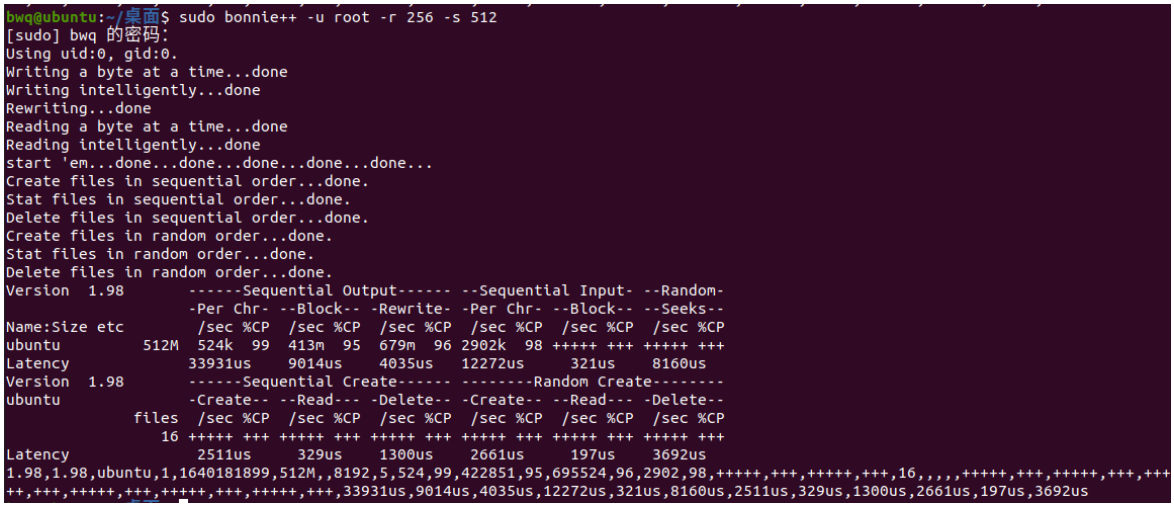
6111920	3531128	492036	1583096
6111920	3510716	492044	1603616
6111920	3489800	492044	1624040
6111920	3468884	492044	1644528
6111920	3448976	492052	1664916
6111920	3428312	492052	1685508
6111920	3407900	492060	1705996
6111920	3387488	492060	1726492
6111920	3366068	492068	1746980
6111920	3345908	492068	1767468
6111920	3325496	492068	1787956
6111920	3304572	492076	1808448
6111920	3283908	492076	1828936
6111920	3263436	492084	1849424
6111920	3242832	492084	1869908
6111920	3222160	492084	1890392
6111920	3201756	492100	1910892
6111920	3181596	492100	1931372
6111920	3160680	492108	1951856
6111920	3125148	492108	1988048
6111920	3098940	492108	2013328
6111920	3079536	492116	2033720
6111920	3064180	492116	2049004
6111920	3038720	492124	2074320
6111920	3029656	492128	2083608
6111920	3008740	492128	2104484
6111920	2976988	492140	2136232
6111920	3837828	492136	1275744
6111920	3837576	492144	1275916



从表和图中可以看出，当启动负载程序后，FREE Mem 大致以每次 20000KB 的速度下降，Inactive Mem 大致以相同的速度上升，说明负载程序申请内存操作成功，从空闲内存中分配了 20MB 给该负载程序，同时由于这部分内存不再被访问，便归入 InactiveMem 中，程序结束后，系统自动一次性释放该程序申请的内存。FREE Mem 和 InactiveMem 再次回到程序运行前的大小。

2、使用 vmstat 工具监控系统中的资源使用状态，运行 bonnie++基准测试程序，观察、记录并分析 bonnie++运行过程中的数据变化。（vmstat 可定时读取相关的 proc 文件，按一定格式给出进程、内存、交换、磁盘 I/O、CPU 利用率等统计信息。）。

运行 bonnie++，同时利用 vmstat 读取系统资源使用状态

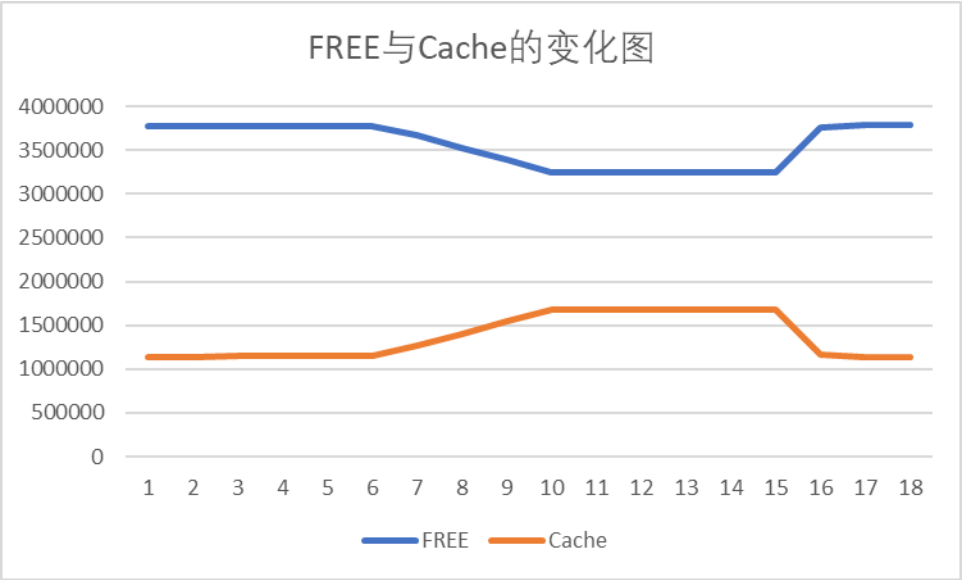


在 vmstat 监控到的数据中，选出变化明显的几列数据，绘制成以时间为横轴的折线图，每个折线图选两处典型的变化说明其意义。

通过观察，FREE（空闲的物理内存）和 Cache（缓存）列变化较明显且有负相关关系：

FREE	Cache
3778868	1142944
3777860	1143432
3777356	1143980
3776852	1144516
3776348	1145028
3775844	1145572
3666372	1266292
3527792	1404152
3390732	1540660
3253168	1677876
3249112	1681528
3249104	1681528

3249104	1681528
3249104	1681528
3249104	1681528
3759944	1164848
3782996	1143540
3782996	1143540



分析：在第 6 秒时，Free 减少、Cache 增加，此时 Bonnie++开始运行，使得空闲内存减少，同时，为了提高访存速度，Cache 相对应增加。在第 10 秒时，空间申请完毕，Free 和 Cache 不再变化。第 15 秒到第 16 秒时，Bonnie++运行完成，删除了生成的临时文件，释放内存空间，同时 Cache 也对应减少。

五、问题讨论

1、proc/meminfo 中的 MemFree 表示什么？观察它的变化，当应用程序不断向操作系统请求内存时，MemFree 会逐渐降低，当其降低到一定程度之后，会发生什么变化？修改 meminfo.sh 脚本，监控/proc/meminfo 中更多的数据，观察 MemFree 增大时，哪些数据减小？此时 MemFree 增加的内存来自哪里？

(1) /proc/meminfo 中的 MemFree 表示：系统中未使用的物理 RAM 量（以 KB 为单位）

(2) 当应用程序不断向操作系统请求内存时，MemFree 会逐渐降低，而 MemActive 会逐渐增大，当 MemFree 降低到一定程度之后，若再申请较大的内存，即应用程序申请的内存比系统的总内存还多时，分配操作便会失败。

(3) 修改的脚本如下：

```
memfile="/proc/meminfo"
```

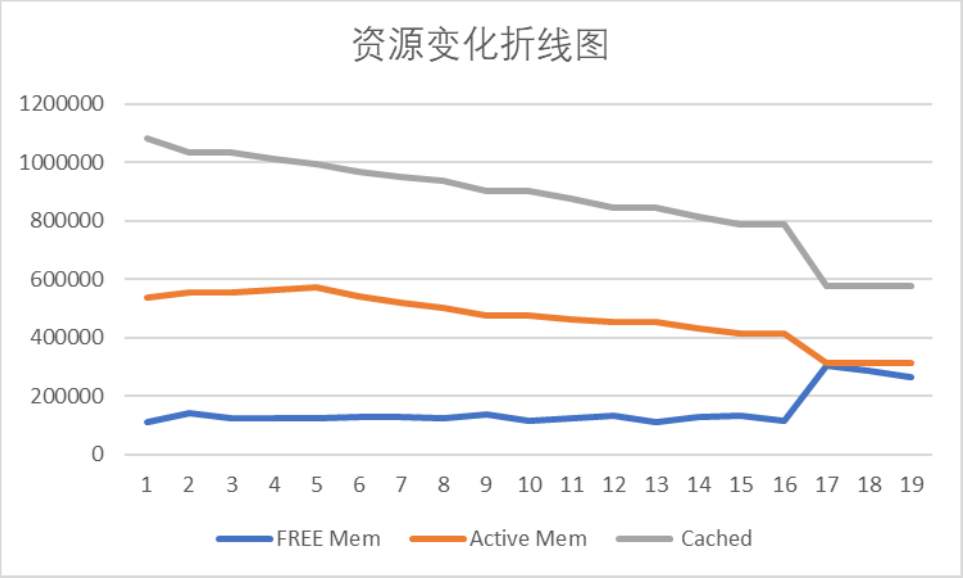
```
    getmeminfo(){
        a=`more $memfile`
        FREE=`echo "$a"|awk 'NR==2{print $2}'`
        PHYMEM=`echo "$a"|awk 'NR==1{print $2}'`
        ACTIVE=`echo "$a"|awk 'NR==7{print $2}'`
        Cached=`echo "$a"|awk 'NR==5{print $2}'`
    }
    while true
    do
        getmeminfo
        echo "Total Mem="$PHYMEM",FREE Mem="$FREE",Active
Mem="$ACTIVE", Cached="$Cached""
        sleep 2
    done
```

当负载程序运行一段时间后，FREE Mem 已经剩余比较低，此时 FREE Mem 开始波动增加。

FREE Mem	Active Mem	Cached
111844	536068	1082312
142252	555184	1032276
121840	555184	1032276
123356	564184	1010348
121556	570840	992156
128688	539504	967676
126600	519008	951128
122472	502240	935472
135368	473404	903212
114452	473840	903320
121136	461308	876648
131812	454612	845968
111668	454664	845968
125696	432708	813636
133512	413144	786060
113352	413160	786060
304228	312952	576304
284928	313108	576304
264516	313108	576304

绘出 FREE Mem、Active Mem、Cached 三种系统资源的变化图，可以看出：

系统从 Active Mem 和 Cached 中分配资源供给 FREE Mem，然后 FREE Mem 再次被负载程序申请，导致 FREE Mem 出现波动的情况。



2、说明程序中如何管理空闲页框。当 bonnie++运行时，是否观察到内存交换？如果观察到内存交换，请截图，并说明交换发生在什么时候？（注意观察 swpd、si、so、free、cache、buff 等指标的变化。）

在 bonnie++运行时，在观察 vmstat 监测数值的变化情况后，没有观察到内存交换，因为 swpd(交换)为 0，也就是使用的虚拟内存大小为 0。

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b 交换 空闲 缓冲 缓存   si   so    bi    bo   in   cs us sy id wa st
 3  0      0 3778868  52000 1142944    0    0  2785   767  467  981 11 17 71  1  0
 1  0      0 3777860  52000 1143432    0    0     0     0  928 1611  7 51 42  0  0
 1  0      0 3777356  52000 1143980    0    0     0     0  381  312  4 49 48  0  0
 3  0      0 3776852  52000 1144516    0    0     0     0  476  926  9 51 41  0  0
 2  0      0 3776348  52000 1145028    0    0     0     0  502  905  6 51 43  0  0
 5  0      0 3775844  52008 1145572    0    0     0    32  611  826  3 51 47  0  0
 1  0      0 3666372  52016 1266292    0    0     0  3108  844 1576  9 55 35  0  0
 2  0      0 3527792  52016 1404152    0    0     0     0  397  267  0 51 50  0  0
 1  0      0 3390732  52016 1540660    0    0     0     0  412  292  0 51 49  0  0
 1  0      0 3253168  52088 1677876    0    0    64 55360  636  404  0 54 46  0  0
 2  0      0 3249112  52132 1681528    0    0     0 839804 1292  650  5 70 24  1  0
 1  0      0 3249104  52140 1681528    0    0     0 155684  979 1164 10 53 37  0  0
 0  0      0 3249104  52148 1681528    0    0     0    32  451  707  5 27 68  0  0
 0  0      0 3249104  52148 1681528    0    0     0     0   84  154  0  0 100  0  0
 0  0      0 3249104  52148 1681528    0    0     0     0   67  153  0  0 100  0  0
 1  0      0 3759944  57288 1164848    0    0   888  6520 1509 2565 16 27 57  1  0
 0  0      0 3782996  57652 1143540    0    0     0 37344 1341 1990 12 42 46  0  0
 0  0      0 3782996  57652 1143540    0    0     0     0  170  342  1  1 99  0  0
```

3、观察 CPU 利用率的变化情况（包括用户态（us）和系统态（sy）），在 bonnie++运行的过程中，当文件系统进行什么操作时，CPU 利用率最高？

在 bonnie++运行过程中，通过观察 vmstat 监测数值的变化情况，通过观察，得到如下的部分数据：

bi	bo	us	sy
2785	767	11	17
0	0	7	51
0	0	4	49
0	0	9	51
0	0	6	51

0	32	3	51
0	3108	9	55
0	0	0	51
0	0	0	51
64	55360	0	54
0	839804	5	70
0	155684	10	53
0	32	5	27
0	0	0	0
0	0	0	0
888	6520	16	27
0	37344	12	42
0	0	1	1

从表中可以看到，在 us 和 sy 较高的时刻，bo 值较高，此时系统正在进行较频繁的写操作。