一 考前准备

1.1 环境介绍

考试总共有两台台虚拟机，与实验练习环境对应关系如下：

实验练习环境	真实考试环境
server0	serverX

提示：以上的X表示你考试时坐的座位编号，根据实际情况而定。

注意：考试中相关虚拟机IP地址不需要配置，hostname主机名以及root账户登录密码等信息，均已在考题或考试说明文档中说明，考试所有操作均在同一台完成。

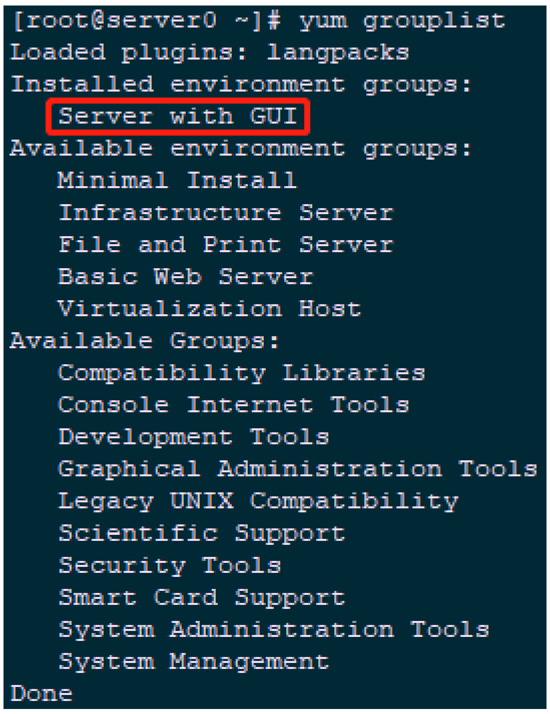
1.2 注意事项

考试考题中涉及的相关执行脚本可能都无执行权限，需要手动添加执行权限，并运行相应脚本测试是否正常运行。

考试中已配置两个yum源，可提前进行检查，两个repo源内容参考如下：

```
[root@server0 yum.repos.d]# cat rh442.repo
[rh442]
gpgcheck = 0
enabled = 1
baseurl = http://content.example.com/rhel7.0/x86_64/rht
name = RH442
[root@server0 yum.repos.d]# cat rhel_dvd.repo
# Created by cloud-init on Wed, 07 Jan 2015 03:21:35 +0000
[rhel_dvd]
gpgcheck = 0
enabled = 1
baseurl = http://content.example.com/rhel7.0/x86_64/dvd
```

Server端已默认安装图形界面，建议提前检查，已安装软件组Installed environment groups中是否存在GUI图形界面。若未安装，可通过以下方式进行安装：**yum -y groupinstall "Server with GUI"。**



```
[root@server0 ~]# yum grouplist
Loaded plugins: langpacks
Installed environment groups:
   Server with GUI
Available environment groups:
   Minimal Install
   Infrastructure Server
   File and Print Server
   Basic Web Server
   Virtualization Host
Available Groups:
   Compatibility Libraries
   Console Internet Tools
   Development Tools
   Graphical Administration Tools
   Legacy UNIX Compatibility
   Scientific Support
   Security Tools
   Smart Card Support
   System Administration Tools
   System Management
Done
```

图形界面已设置为默认开机界面，可通过以下方式查看：**systemctl get-default**，若图形界面未设置为开机模式，可通过**systemctl set-default graphical.target**进行设置。

二 题目解析

2.1 识别硬件特性

The file /root/dmidecode.out contains the output from the dmidecode utility that was run on another system.Using this file identify the size of the L1 and L2 data caches,on the system on which this output was generated.If the output contains information about multiple processors or multiple cores,you should base your answer on only a single processor or core.

Enter the values you find and click on 'Submit Answers' to record your answers.

解析：

- 检查文件类型

```
[root@server0 ~]# file dmidecode.out
```

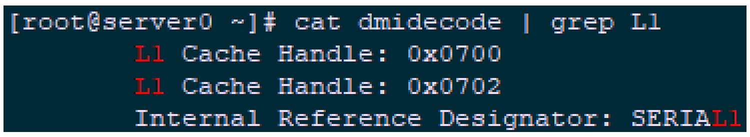
提示：若题意给出的文件类型为ASCII text，则可直接打开，若为data文件类型，则需要使用dmidecode解密。

- 解密加密文档

```
[root@server0 ~]# dmidecode --help      #查看命令相关选择，也可使用man dmidecode
[root@server0 ~]# dmidecode --from-dump dmidecode.out >dmidecode.txt
```

- 查找一级缓存

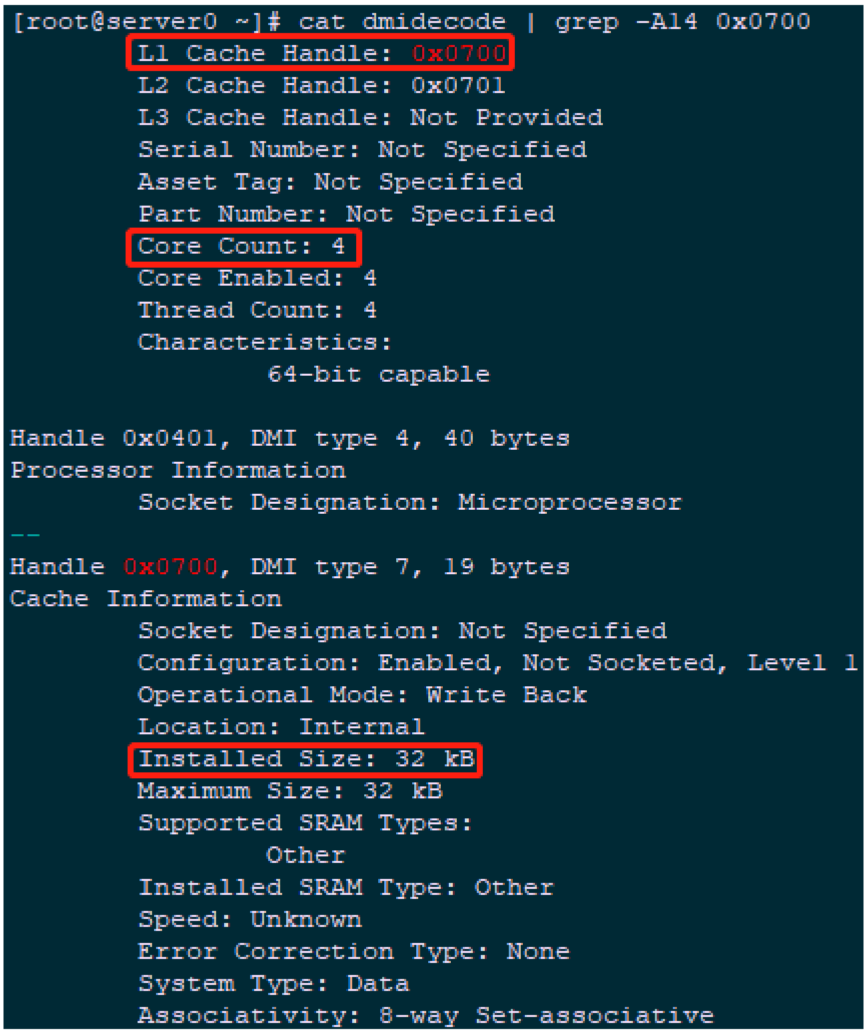
```
[root@server0 ~]# cat dmidecode | grep L1
```



```
[root@server0 ~]# cat dmidecode | grep L1
        L1 Cache Handle: 0x0700
        L1 Cache Handle: 0x0702
        Internal Reference Designator: SERIAL1
```

提示：若存在多个L1，表示存在多个物理CPU槽，任取一个计算即可。

```
[root@server0 ~]# cat dmidecode | grep -A14 0x0700
```



```
[root@server0 ~]# cat dmidecode | grep -A14 0x0700
        L1 Cache Handle: 0x0700
        L2 Cache Handle: 0x0701
        L3 Cache Handle: Not Provided
        Serial Number: Not Specified
        Asset Tag: Not Specified
        Part Number: Not Specified
        Core Count: 4
        Core Enabled: 4
        Thread Count: 4
        Characteristics:
                64-bit capable

Handle 0x0401, DMI type 4, 40 bytes
Processor Information
        Socket Designation: Microprocessor
--
Handle 0x0700, DMI type 7, 19 bytes
Cache Information
        Socket Designation: Not Specified
        Configuration: Enabled, Not Socketed, Level 1
        Operational Mode: Write Back
        Location: Internal
        Installed Size: 32 kB
        Maximum Size: 32 kB
        Supported SRAM Types:
                Other
        Installed SRAM Type: Other
        Speed: Unknown
        Error Correction Type: None
        System Type: Data
        Associativity: 8-way Set-associative
```

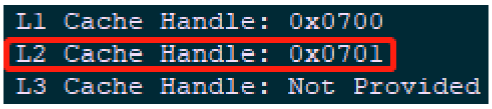
```
[root@server0 ~]# bc                    #进入bc计算器
scale=8                                 #保留8为小数
32/4/2
4
L1 data caches=32/4/2=4
```

提示：若Core Count为1，即如文档所示为单核，则直接查找相应的值，然后除以2，单独计算cata chahes;

若查找后存在多个L1，可能有多颗物理CPU，根据题意回答单个CPU的L1的data cache值即可，L1=Installed Size/n/2。

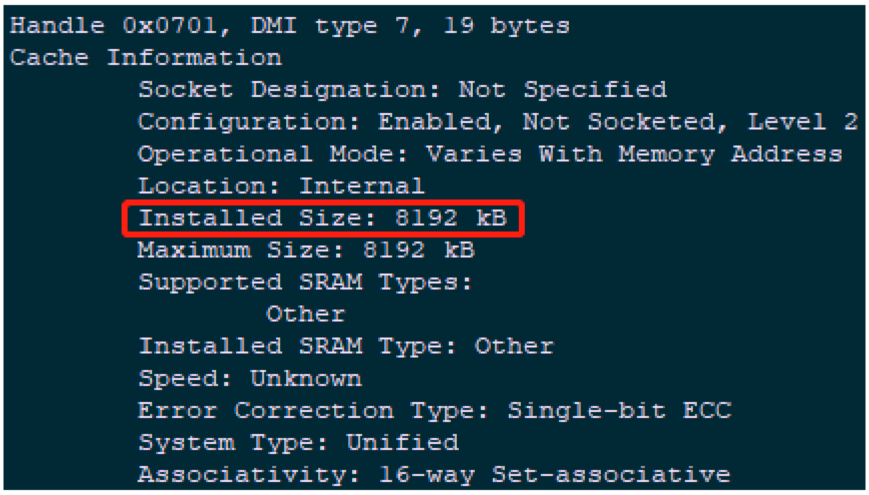
- 查找二级缓存

```
[root@server0 ~]# vi dmidecode          #也可使用vim编辑，进入命令模式
/L2                                     #搜索L2
```



```
L1 Cache Handle: 0x0700
L2 Cache Handle: 0x0701
L3 Cache Handle: Not Provided
```

/0x0701   #定位到0x0701



```
Handle 0x0701, DMI type 7, 19 bytes
Cache Information
        Socket Designation: Not Specified
        Configuration: Enabled, Not Socketed, Level 2
        Operational Mode: Varies With Memory Address
        Location: Internal
        Installed Size: 8192 kB
        Maximum Size: 8192 kB
        Supported SRAM Types:
                Other
        Installed SRAM Type: Other
        Speed: Unknown
        Error Correction Type: Single-bit ECC
        System Type: Unified
        Associativity: 16-way Set-associative
```

L2=8192/4=2048

提示：参考一级缓存查找方法，查找到相应值，L2=Installed Size/n。
若Core Count为n(n>=1)，但不存在三级缓存，则L2为此n核共用；
若Core Count为n(n>=1)，但存在三级缓存，则此n核每个核存在独立的L2；
通过dmidecode查找的L2值都是所有L2总和，对于Core Count为n(n>=1)的情况，都需要除以n。

2.2 分析sar输出

The file sar.data,in the root home directory,contains data recorded on another system for a period of time.Use this file to answer the following questions.Note:the file contains data gathered on a system that used logical volumes.You should ignore any data pertaining to logical volumes and only report values for physical devices.

What was the maximum process count on the system?

Which device had the highest write I/O rate?(device name can be entered in sdX style or devX-XX style)

What was the maximum write I/O rate,in kiB/s,for this device?(rounded to the nearest hundred)

When,in minuters,relative to the beginning of the monitoring period did the system have a burst of activity that resulted in the contents of memorty being written to disk?

Enter the values you find and click on 'Submit Answers' to record your answers.

解析:

- 解压文件

```
[root@server0 ~]# file sar.data.bz2
```

```
sar.data.bz2: bzip2 compressed data, block size = 900k
[root@server0 ~]# bunzip2 sar.data.bz2
```

- 检查命令

```
[root@server0 ~]# yum provides sar
[root@server0 ~]# rpm -qa | grep sysstat        #检查系统是否安装sar及iostat命令
```

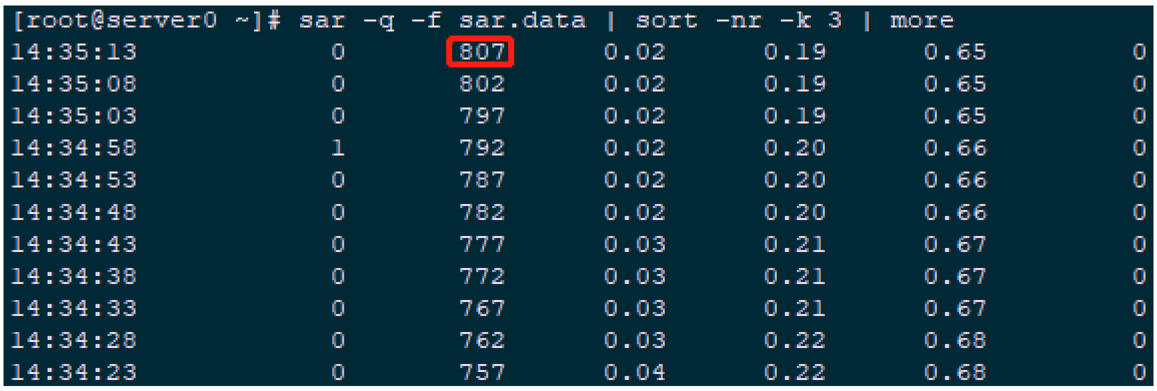
提示：若出现未安装sar和iostat命令的情况，可执行yum -y install sysstat安装相关命令即可。

- 命令参考

```
[root@server0 ~]# sar -h                #通过sar -h可查看相关sar命令选项
sar命令:
用法: sar [ 选项 ] [ <时间间隔> [ <次数> ] ]
选项:
    -d                输出每一块磁盘的使用信息
    -f                从制定的文件读取报告
    -P {cpu|ALL}      报告每个CPU的状态
    -q                输出进程队列长度和平均负载状态统计信息
    -b                显示I/O和传送速率的统计信息，缓冲区使用情况。
    -p                显示友好设备名字，以方便查看，也可以和-d 和-n 参数结合使用，比如 -dp
    或-np
    sort命令:
选项:
    -n, --numeric-sort              #根据字符串数值比较
    -k, --key=位置1[,位置2]          #在位置1 开始一个key，在位置2 终止(默认为行尾)
    -r, --reverse                   #逆序输出排序结果
[root@server0 ~]# alias sar='LANG=C sar'        #建议将此alias添加至bashrc
```

- 统计最大进程数

```
[root@server0 ~]# sar -q -f sar.data | sort -nr -k 3 | more
```

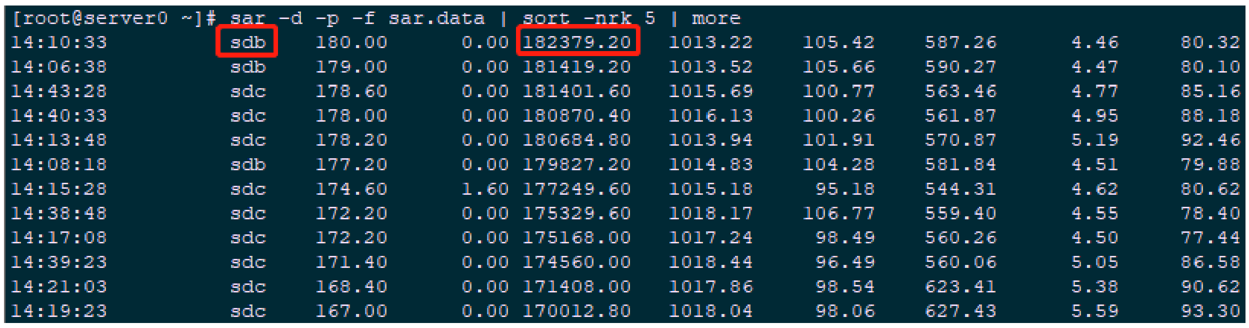


```
[root@server0 ~]# sar -q -f sar.data | sort -nr -k 3 | more
14:35:13           0       807      0.02      0.19      0.65         0
14:35:08           0       802      0.02      0.19      0.65         0
14:35:03           0       797      0.02      0.19      0.65         0
14:34:58           1       792      0.02      0.20      0.66         0
14:34:53           0       787      0.02      0.20      0.66         0
14:34:48           0       782      0.02      0.20      0.66         0
14:34:43           0       777      0.03      0.21      0.67         0
14:34:38           0       772      0.03      0.21      0.67         0
14:34:33           0       767      0.03      0.21      0.67         0
14:34:28           0       762      0.03      0.22      0.68         0
14:34:23           0       757      0.04      0.22      0.68         0
```

最大进程数为**807**

- 统计最大写速率的设备

```
[root@server0 ~]# sar -d -p -f sar.data | sort -nrk 5 | head
```



```
[root@server0 ~]# sar -d -p -f sar.data | sort -nrk 5 | more
14:10:33          sdb    180.00      0.00 182379.20  1013.22    105.42    587.26      4.46     80.32
14:06:38          sdb    179.00      0.00 181419.20  1013.52    105.66    590.27      4.47     80.10
14:43:28          sdc    178.60      0.00 181401.60  1015.69    100.77    563.46      4.77     85.16
14:40:33          sdc    178.00      0.00 180870.40  1016.13    100.26    561.87      4.95     88.18
14:13:48          sdc    178.20      0.00 180684.80  1013.94    101.91    570.87      5.19     92.46
14:08:18          sdb    177.20      0.00 179827.20  1014.83    104.28    581.84      4.51     79.88
14:15:28          sdc    174.60      1.60 177249.60  1015.18     95.18    544.31      4.62     80.62
14:38:48          sdc    172.20      0.00 175329.60  1018.17    106.77    559.40      4.55     78.40
14:17:08          sdc    172.20      0.00 175168.00  1017.24     98.49    560.26      4.50     77.44
14:39:23          sdc    171.40      0.00 174560.00  1018.44     96.49    560.06      5.05     86.58
14:21:03          sdc    168.40      0.00 171408.00  1017.86     98.54    623.41      5.38     90.62
14:19:23          sdc    167.00      0.00 170012.80  1018.04     98.06    627.43      5.59     93.30
```

device即为**sdb**。

提示：考试中可能题目为read I/O或write I/O，同时设备名通常为sdX。

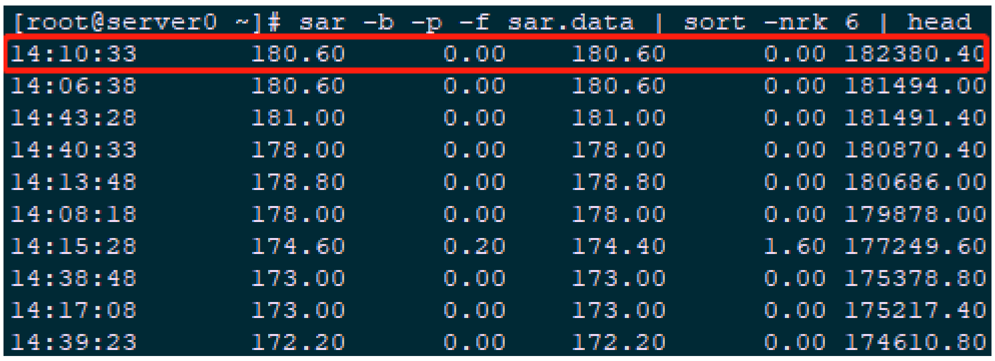
- 统计最大写速率的速率

即为上一个查询后的值换算速率，182379*512/1024=91189，取整即为91190。

提示：sar查询后的结果为扇区，换算成kb方可，考试中选择最接近的值即可。

- 统计磁盘最大并发发生时间

```
[root@server0 ~]# sar -b -p -f sar.data | sort -nrk 6 | head
```



```
[root@server0 ~]# sar -b -p -f sar.data | sort -nrk 6 | head
14:10:33         180.60      0.00    180.60      0.00 182380.40
14:06:38         180.60      0.00    180.60      0.00 181494.00
14:43:28         181.00      0.00    181.00      0.00 181491.40
14:40:33         178.00      0.00    178.00      0.00 180870.40
14:13:48         178.80      0.00    178.80      0.00 180686.00
14:08:18         178.00      0.00    178.00      0.00 179878.00
14:15:28         174.60      0.20    174.40      1.60 177249.60
14:38:48         173.00      0.00    173.00      0.00 175378.80
14:17:08         173.00      0.00    173.00      0.00 175217.40
14:39:23         172.20      0.00    172.20      0.00 174610.80
```

最大并发出现的时间为第10分钟。

2.3 识别系统调用

Identify the most frequently called system call in the command: /bin/dumpkeys

解析：

- 检查命令

```
[root@server0 ~]# yum provides strace
[root@server0 ~]# rpm -qa | grep strace #检查系统是否安装strace命令
```

提示：若出现未安装strace命令的情况，可执行yum -y install strace安装相关命令即可。

- 命令参考

```
[root@server0 ~]# strace -h      #通过strace -h可查看相关strace命令选项
usage: strace [-CdffhiqrtttTvVxxy] [-I n] [-e expr]...
              [-a column] [-o file] [-s strsize] [-P path]...
              -p pid... / [-D] [-E var=val]... [-u username] PROG [ARGS]
   or: strace -c[df] [-I n] [-e expr]... [-O overhead] [-S sortby]
```

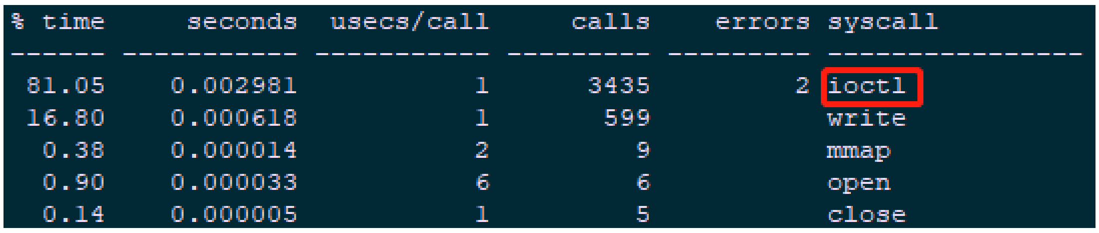
```
              -p pid... / [-D] [-E var=val]... [-u username] PROG [ARGS]
```

选项:

-f	跟踪由fork调用所产生的子进程.
-c	统计每一系统调用的所执行的时间,次数和出错的次数等.
-s strsize	指定输出的字符串的最大长度.默认为32.文件名一直全部输出.
-S sortby	排序syscall计数：时间、调用、名称，默认时间。

- 最多的系统调用

```
[root@server0 ~]# strace -fcS calls /bin/dumpkeys
```



```
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 81.05    0.002981           1      3435         2 ioctl
 16.80    0.000618           1       599           write
  0.38    0.000014           2         9           mmap
  0.90    0.000033           6         6           open
  0.14    0.000005           1         5           close
```

最大的系统调为：**ioctl**。

2.4 配置swap空间

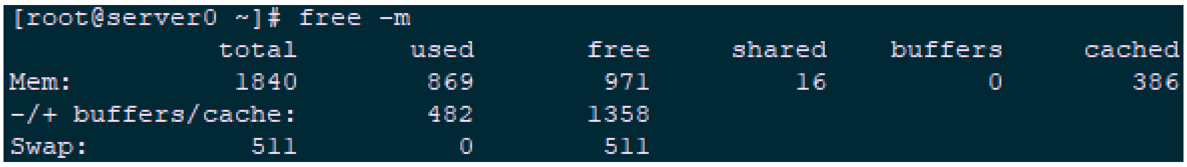
This system should have a total of 2048 MB of swap space.Configure enough swap to make this the case subject to the following requirements:

Do not delete any existing swap volumes;
The additional swap space should be split equally among two new partitions;
The new swap partitions should be mounted at system boot time;
Upon boot up,the kernel should use the new swap partitions before using the existing swap volume.

解析:

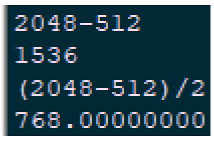
- 查看当前系统使用swap

```
[root@server0 ~]# free -m
```



```
[root@server0 ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          1840        869        971         16          0        386
-/+ buffers/cache:        482       1358
Swap:          511          0        511
```

提示：基于保留系统当前已存在的512M swap。题意要求总共需要配置2048M，且均衡分布在两个分区，因此额外需要增加1536M swap，且每个分区为768M。



```
2048-512
1536
(2048-512)/2
768.00000000
```

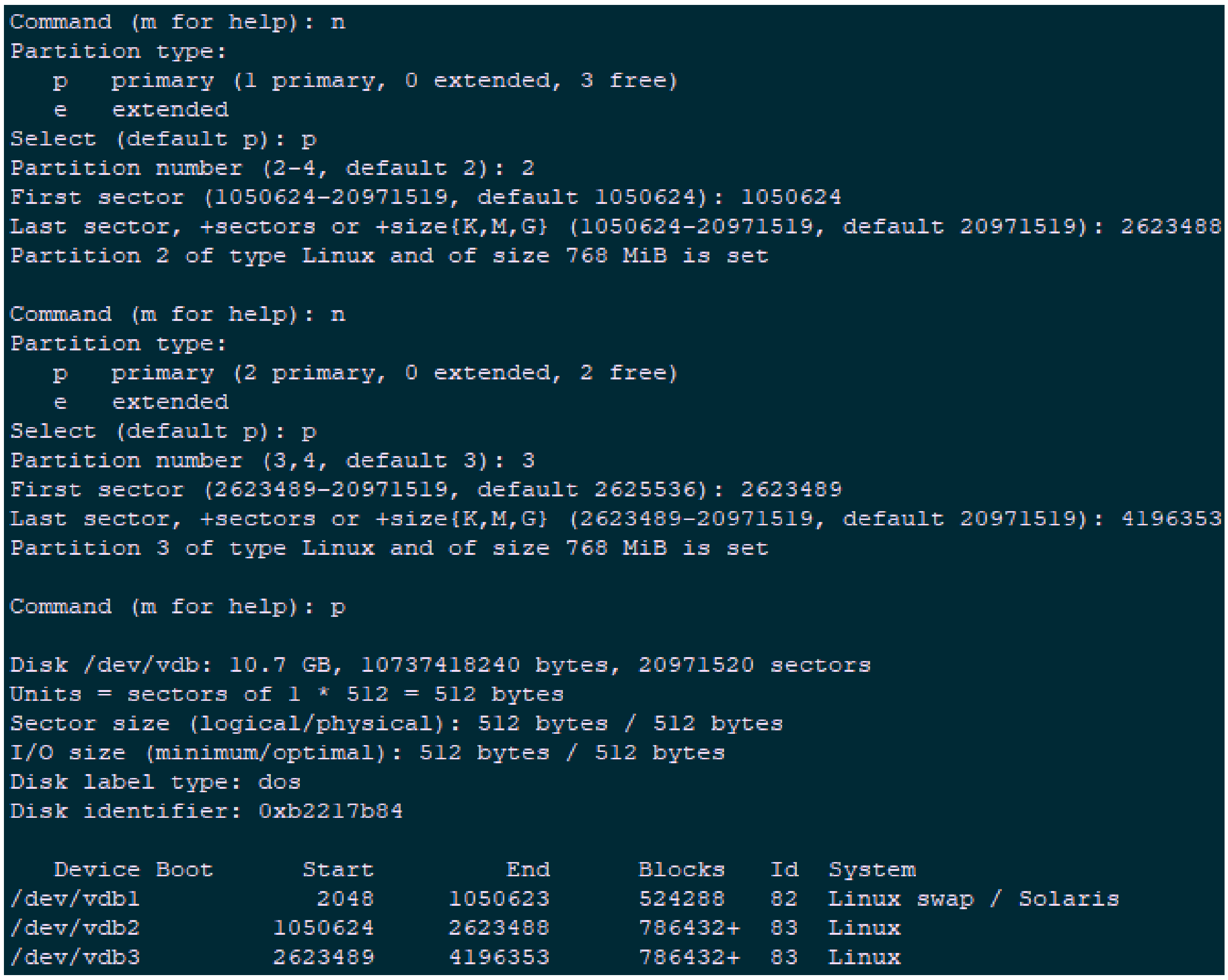
- 计算所需sectors扇区数

(768M*1024*1024)/512=1572864，即总共需要1572864扇区。

如下所示第一个分区开始扇区为1050624，则最后一个扇区为1050624+1572864=2623488。

如下所示第二个分区开始扇区为2623489，则最后一个扇区为2627584+1572864=4196353。

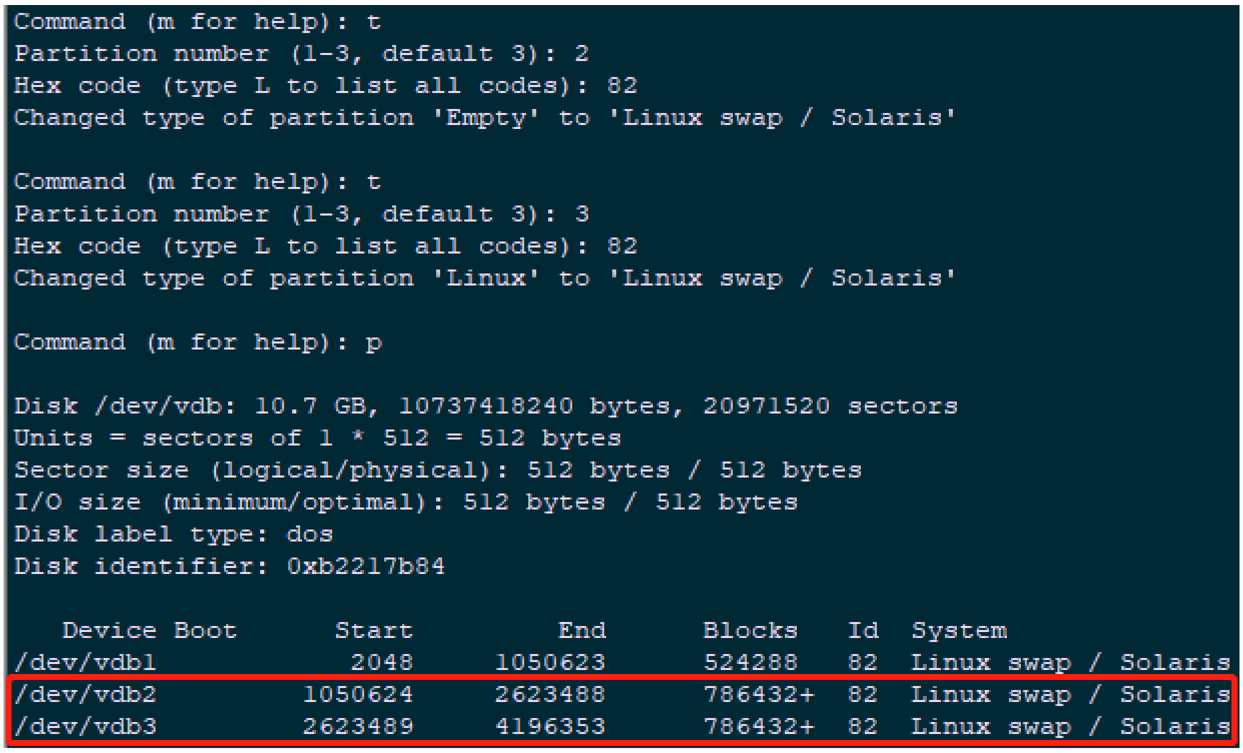
```
[root@server0 ~]# fdisk /dev/vdb                                    #考试可能为其他磁盘
```



```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (2-4, default 2): 2
First sector (1050624-20971519, default 1050624): 1050624
Last sector, +sectors or +size{K,M,G} (1050624-20971519, default 20971519): 2623488
Partition 2 of type Linux and of size 768 MiB is set

Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p): p
Partition number (3,4, default 3): 3
First sector (2623489-20971519, default 2625536): 2623489
Last sector, +sectors or +size{K,M,G} (2623489-20971519, default 20971519): 4196353
Partition 3 of type Linux and of size 768 MiB is set

Command (m for help): p

Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb2217b84

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048     1050623      524288   82  Linux swap / Solaris
/dev/vdb2         1050624     2623488      786432+  83  Linux
/dev/vdb3         2623489     4196353      786432+  83  Linux
```

注意：建议采用扇区数进行计算，并同时观察考试中分区的开始扇区数，考试中允许一定量（如最后显示结果为2047）的偏差。

- 创建swap分区



```
Command (m for help): t
Partition number (1-3, default 3): 2
Hex code (type L to list all codes): 82
Changed type of partition 'Empty' to 'Linux swap / Solaris'

Command (m for help): t
Partition number (1-3, default 3): 3
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'

Command (m for help): p

Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb2217b84

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048     1050623      524288   82  Linux swap / Solaris
/dev/vdb2         1050624     2623488      786432+  82  Linux swap / Solaris
/dev/vdb3         2623489     4196353      786432+  82  Linux swap / Solaris
```

```
[root@server0 ~]# partprobe
[root@server0 ~]# mkswap /dev/vdb2
[root@server0 ~]# mkswap /dev/vdb3
```

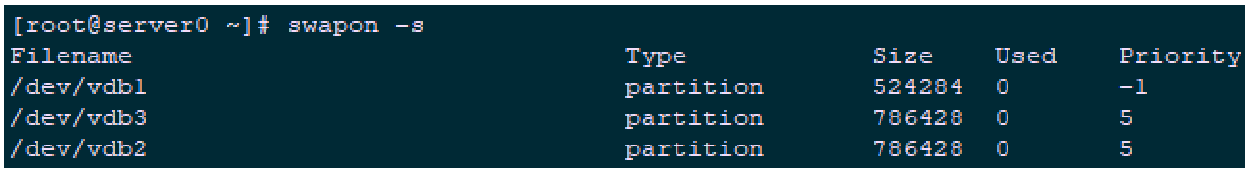
- 设置swap优先级并设置开机自动挂载

```
[root@server0 ~]# swapon -p 5 /dev/vdb2
[root@server0 ~]# swapon -p 5 /dev/vdb3
[root@server0 ~]# vi /etc/fstab
/dev/vdb2       swap    swap    default,pri=5   0 0
/dev/vdb3       swap    swap    default,pri=5   0 0
```

提示：题意要求内核优先使用新增的swap分区，系统默认自带的swap分区优先级为-1，将新增的swap分区设置为任何大于-1的优先级即可。

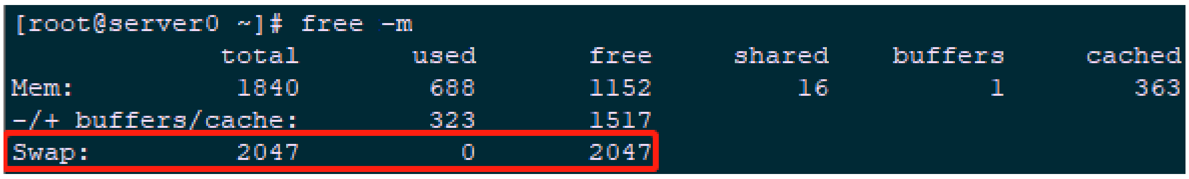
- 确认验证

```
[root@server0 ~]# swapon -s
```



```
[root@server0 ~]# swapon -s
Filename                                Type            Size    Used    Priority
/dev/vdb1                               partition       524284  0       -1
/dev/vdb3                               partition       786428  0       5
/dev/vdb2                               partition       786428  0       5
```

```
[root@server0 ~]# free -m
```



```
[root@server0 ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          1840        688       1152         16          1        363
-/+ buffers/cache:        323       1517
Swap:         2047          0       2047
```

提示：查看相关优先级及分区总大小是否正确，若出现+-1误差为正常误差。

2.5 内存限制

Configure station such that when the application /usr/local/bin/greedy is run,it fail with the message 'unable to allocate memory' but the application /usr/local/bin/checklimit displays the message 'Success:Address space limit is okay'.

解析:

- 命令参考

```
[root@server0 ~]# sysctl -h                    #通过sysctl -h可查看相关sysctl命令选项
Usage:
 sysctl [options] [variable[=value] ...]


Options:
  -a, --all            display all variables
```

选项:

```
-a                              显示所有参数
[root@server0 ~]# sysctl -a | grep memory        #查看当前内存相关参数
vm.overcommit_memory = 0
```

- 设置内存限制

```
[root@server0 ~]# cat /etc/sysctl.conf
[root@server0 ~]# echo "vm.overcommit_memory = 2" >> /etc/sysctl.d/rh422.conf
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh422.conf
```

提示:

vm.overcommit_memory: 内存过量分配。

- **0: 用户申请内存的时候，系统会判断剩余的内存多少，如果不够的话那么就会失败。**
- **1: 用户申请内存的时候，系统不进行任何检查任务内存足够用，直到使用内存超过可用内存。**
- **2: 用户一次申请的内存大小不允许超过可用内存的大小。**

CentOS7系默认设置通过/usr/lib/sysctl.d/00-system.conf来实现，可修改此文件或/etc/sysctl.conf来实现内存限制;
强烈建议不要修改系统自身以上两个文件，在/etc/sysctl.d目录下独立新建子配置文件rh442.conf。

- 运行测试

```
[root@server0 ~]# shutdown -r now              #建议重启
[root@server0 ~]# chmod u+x /usr/local/bin/greedy
[root@server0 ~]# chmod u+x /usr/local/bin/checklimit
[root@server0 ~]# greedy
Error: 1618 unable to allocate memory.
[root@server0 ~]# checklimit
Success: Address space limit is okay
```

提示：若运行greedy出现以下错误提示，则可通过yum provides ld-linux.so.2可得知缺少的so属于glibc，通过yum install -y glibc.i686进行安装。
-bash: /usr/local/bin/greedy: /lib/ld-linux.so.2: bad ELF interpreter: No such file or directory

2.6 配置共享内存

Configure station so that the amount of SYSV shared memory available to all application is at least 1 GiB but no more than 1.5 GiB.This setting should persist across reboots.

解析：

- 查看系统现有内存限制

```
[root@server0 ~]# sysctl -a | grep -i shm
kernel.shm_rmid_forced = 0
kernel.shmall = 268435456                              #全局最大可使用内存page
kernel.shmmax = 4294967295                             #每个共享内存段的最大值
kernel.shmmni = 4096                                   #全局最多可申请共享内存段数
vm.hugetlb_shm_group = 0
```

- 查看页换算单位

```
[root@server0 ~]# getconf -a | grep PAGE
PAGESIZE                                4096           #确认每一个page为多少byte
```

- 将需要限制的内存换算为页

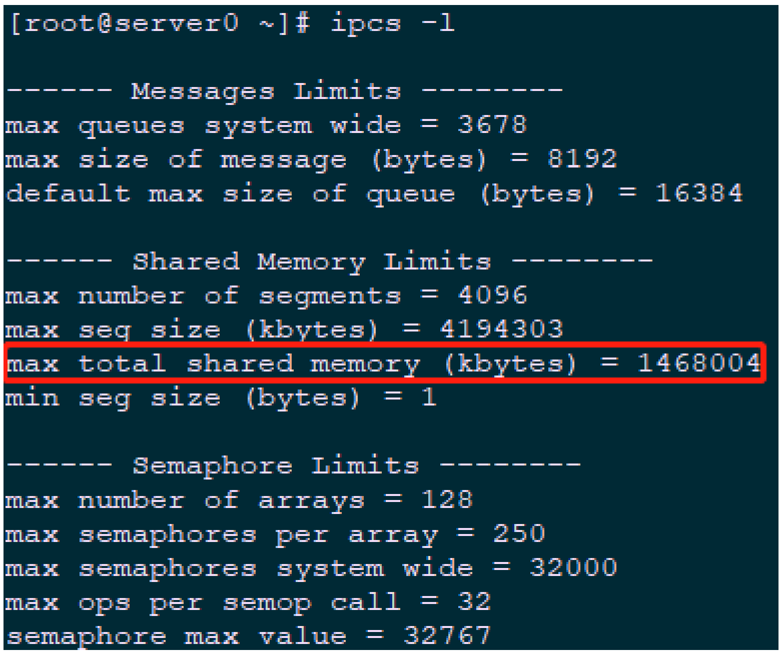
1.4GiB=1.4*1024*1024*1024/4096=367001.6， 取367001。
提示：题意要求内存限制内存使用在1G-1.5G之间，建议取值为1.4G。

- 设置最大共享内存page

```
[root@server0 ~]# echo "kernel.shmall = 367001" >> /etc/sysctl.d/rh442.conf
#追加配置
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh442.conf
```

- 确认验证

```
[root@server0 ~]# ipcs -l
```



```
[root@server0 ~]# ipcs -l

------ Messages Limits --------
max queues system wide = 3678
max size of message (bytes) = 8192
default max size of queue (bytes) = 16384

------ Shared Memory Limits --------
max number of segments = 4096
max seg size (kbytes) = 4194303
max total shared memory (kbytes) = 1468004
min seg size (bytes) = 1

------ Semaphore Limits --------
max number of arrays = 128
max semaphores per array = 250
max semaphores system wide = 32000
max ops per semop call = 32
semaphore max value = 32767
```

提示：max total shared memory为kb值，每一页为4096 byte，可通过bc计算器 1468004*1024/4096转换为页。

2.7 配置进程优先

On station in /usr/local/bin there is an application called realtime.Configure the system so that this application starts up automatically at system boot with a static priority of 27 using round robin priority scheduling.This program should run as a background job.

You can verify that realtime is running by viewing /var/log/messages.

解析：

- 检查所在目录及相关权限

```
[root@server0 ~]# which realtime
[root@server0 ~]# ll /usr/local/bin/realtime
```

提示：若该应用程序不在题意所示/usr/local/bin目录，可通过mv移动至该目录。

```
[root@server0 ~]# chmod u+x /usr/local/bin/realtime
[root@server0 ~]# realtime                            #测试运行是否正常
```

提示：若运行出现提示缺少glibc相关包，可参考2.5进行安装。

- 修改调度超时时间

```
[root@server0 ~]# sysctl -a | grep -i runtime
kernel.sched_rt_runtime_us = 950000
[root@server0 ~]# echo "kernel.sched_rt_runtime_us = -1" >>
/etc/sysctl.d/rh442.conf
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh442.conf
[root@server0 ~]# sysctl -a | grep -i runtime
kernel.sched_rt_runtime_us = -1
```

提示：修改实时进程调度为-1，即永不超时。

- 静默后台运行

```
[root@server0 ~]# whereis chrt                    #查看chrt命令完整路径
[root@server0 ~]# echo "/usr/bin/chrt -r 27 /usr/local/bin/realtime &" >>
/etc/rc.local
[root@server0 ~]# chmod u+x /etc/rc.local        #授权
[root@server0 ~]# shutdown -r now
```

提示：chrt的"-r"参数：round robin轮巡方式，题意realtime 需要在开机时后台运行，因此必须加上&参数。

- 确认验证

```
[root@server0 ~]# tail /var/log/messages
[root@server0 ~]# ps aux | grep -v grep | grep realtime
root  1209  0.0  0.0  4292  548  ?  S  20:50  0:00  /usr/local/bin/realtime
[root@server0 ~]# ps -o rtprio $(pgrep realtime)                #查看优先级
RTPRIO
    27
```

2.8 区分内存利用率

Determine the amount of physical memory,in pages,used by the realtime application.If necessary,pick the value from the list that rounds up to or down to the closest value.Click on 'Submit Anwsers' to record your answer.

解析：

- 查看进程RSS

```
[root@server0 ~]# ps -o comm,rss,cls,pid,rtprio $(pgrep realtime)
COMMAND            RSS CLS   PID RTPRIO
realtime           548  RR  1209     27
[root@server0 ~]# ps aux | grep -v grep | grep realtime
root       1193  0.0  0.0   4292   548 ?        S    14:58   0:00
/usr/local/bin/realtime
```

提示：rtprio：
RSS：即realtime这个程序在Linux系统中实际分配物理内存的大小；
CLS：即进程优先级的运行模式，确定是否处于RR(round robin轮巡)模式下；
PID：即程序进程号；
RTPRIO：reltime priority，确认是否为调整后的27；
pgrep realtime：查找出realtime的进程ID，同时确认优先级，通过PS查找出的RSS单位为kb。

- 查看页换算单位

```
[root@server0 ~]# getconf -a | grep PAGE
PAGESIZE                              4096 #确认每一个page为多少byte，即1page=4kb
[root@server0 ~]# bc
548/4
137
```

ps命令查看的RSS单位为K，每页单位为4K，即所占内存为548/4，137页。

注意：考试时此题答案为下拉框，若给出的选项中不存在计算出来的值，则选择最接近计算得出的数值，建议选择稍大选项。

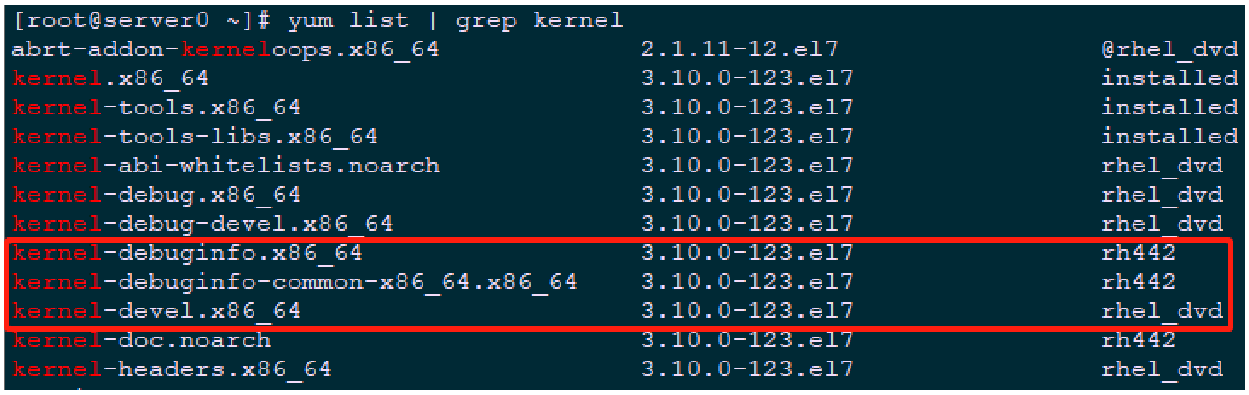
2.9　　配置系统分析支持

On station there is a file in the root home directory called count_jiffies.stp.Configure this system so that the root user can run this script.

解析：

- 查看当前内核版本

```
[root@server0 ~]# uname -r
3.10.0-123.el7.x86_64
[root@server0 ~]# yum list | grep kernel          #查看yum源中的kernel版本
```



```
[root@server0 ~]# yum list | grep kernel
abrt-addon-kerneloops.x86_64                2.1.11-12.el7                @rhel_dvd
kernel.x86_64                               3.10.0-123.el7               installed
kernel-tools.x86_64                         3.10.0-123.el7               installed
kernel-tools-libs.x86_64                    3.10.0-123.el7               installed
kernel-abi-whitelists.noarch                3.10.0-123.el7               rhel_dvd
kernel-debug.x86_64                         3.10.0-123.el7               rhel_dvd
kernel-debug-devel.x86_64                   3.10.0-123.el7               rhel_dvd
kernel-debuginfo.x86_64                     3.10.0-123.el7               rh442
kernel-debuginfo-common-x86_64.x86_64       3.10.0-123.el7               rh442
kernel-devel.x86_64                         3.10.0-123.el7               rhel_dvd
kernel-doc.noarch                           3.10.0-123.el7               rh442
kernel-headers.x86_64                       3.10.0-123.el7               rhel_dvd
```

提示：

若yum源同时存在与当前系统匹配的包，以及高于当前内核版本的所需包，则必须通过# yum -y install kernel-debuginfo-$(uname -r)类似命令安装匹配版本，而不要默认yum;

若yum源只存在高于当前内核版本的所需包，则通过yum -y updata将当前内核升级到最新版，之后正常安装所需包;

系统已安装的内核版本与yum仓库版本一致，则执行下一步操作即可。

- 正常安装所需包

```
[root@server0 ~]# yum -y install kernel-devel kernel-debuginfo kernel-debuginfo-common systemtap                                     #所安装相关包版本必须和系统的内核版本一致
```

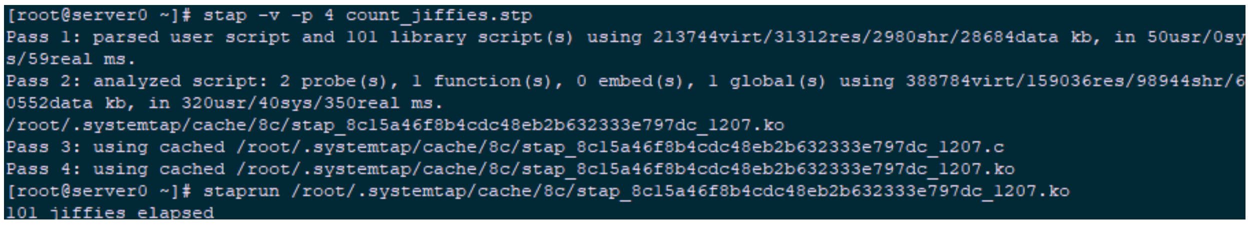
- 验证确认

```
[root@server0 ~]# stap count_jiffies.stp          #运行题意所需stp文件即可
```

```
101 jiffies elapsed
```

- 编入内核模块

```
[root@server0 ~]# stap count_jiffies.stp
[root@server0 ~]# stap -v -p4 count_jiffies.stp
[root@server0 ~]# staprun \
/root/.systemtap/cache/8c/stap_8c15a46f8b4cdc48eb2b632333e797dc_1207.ko
```



```
[root@server0 ~]# stap -v -p 4 count_jiffies.stp
Pass 1: parsed user script and 101 library script(s) using 213744virt/31312res/2980shr/28684data kb, in 50usr/0sy
s/59real ms.
Pass 2: analyzed script: 2 probe(s), 1 function(s), 0 embed(s), 1 global(s) using 388784virt/159036res/98944shr/6
0552data kb, in 320usr/40sys/350real ms.
/root/.systemtap/cache/8c/stap_8c15a46f8b4cdc48eb2b632333e797dc_1207.ko
Pass 3: using cached /root/.systemtap/cache/8c/stap_8c15a46f8b4cdc48eb2b632333e797dc_1207.c
Pass 4: using cached /root/.systemtap/cache/8c/stap_8c15a46f8b4cdc48eb2b632333e797dc_1207.ko
[root@server0 ~]# staprun /root/.systemtap/cache/8c/stap_8c15a46f8b4cdc48eb2b632333e797dc_1207.ko
101 jiffies elapsed
```

提示：题目暗含需要编入内核作为模块开机加载。

2.10 缓存命中率

There are two versions of an application in the home directory for the root account:cache-a and cache-b.Both versions of the application do the same thing but one of the application exhibits improper use of cache memory.

Determine which program has the better cache performance and copy this program into /usr/local/bin.

You may choose which utility,or utilities,to use to make this determination.

解析:

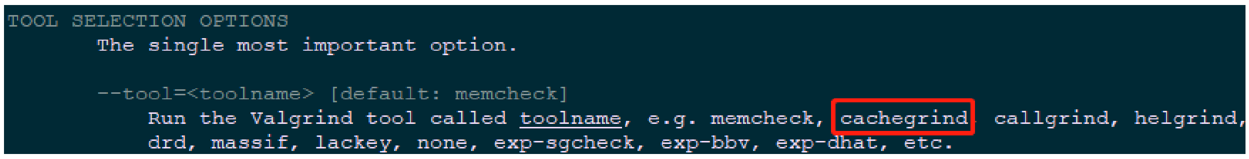
- 检查权限

```
[root@server0 ~]# ll cache-a cache-b
[root@server0 ~]# chmod u+x cache-*
```

提示：若题意所示应用程序无执行权限，可通过chmod u+x cache-a cache-b进行授权。

- 安装所需软件

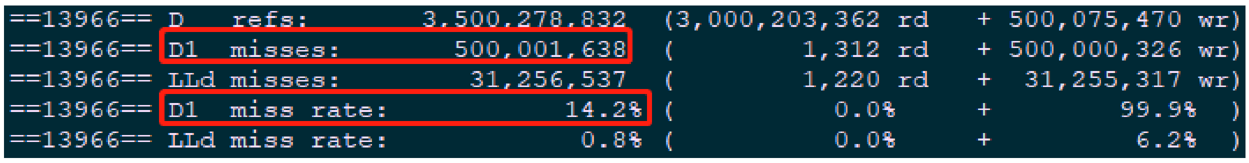
```
[root@server0 ~]# yum -y install valgrind                    #考试中安装x64_64版本
[root@server0 ~]# man valgrind                               #查看valgrind使用方法
```



```
TOOL SELECTION OPTIONS
       The single most important option.

       --tool=<toolname> [default: memcheck]
           Run the Valgrind tool called toolname, e.g. memcheck, cachegrind, callgrind, helgrind,
           drd, massif, lackey, none, exp-sgcheck, exp-bbv, exp-dhat, etc.
```

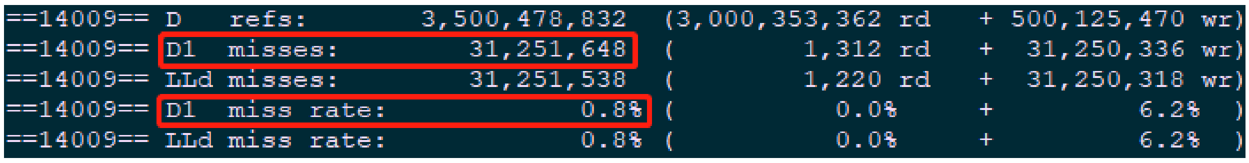
- 检测命中率

```
[root@server0 ~]# valgrind --tool=cachegrind /root/cache-a        #查看D1 cache相关
信息
```



```
==13966== D   refs:         3,500,278,832  (3,000,203,362 rd   + 500,075,470 wr)
==13966== D1  misses:         500,001,638  (        1,312 rd   + 500,000,326 wr)
==13966== LLd misses:          31,256,537  (        1,220 rd   +  31,255,317 wr)
==13966== D1  miss rate:             14.2% (          0.0%     +        99.9%   )
==13966== LLd miss rate:              0.8% (          0.0%     +         6.2%   )
```

```
[root@server0 ~]# valgrind --tool=cachegrind /root/cache-b
```
#查看D1 cache相关信息



```
==14009== D   refs:         3,500,478,832  (3,000,353,362 rd   + 500,125,470 wr)
==14009== D1  misses:          31,251,648  (        1,312 rd   +  31,250,336 wr)
==14009== LLd misses:          31,251,538  (        1,220 rd   +  31,250,318 wr)
==14009== D1  miss rate:              0.8% (          0.0%     +         6.2%   )
==14009== LLd miss rate:              0.8% (          0.0%     +         6.2%   )
```

对比可知cache-b的丢失率远低于cache-a的丢失率，即cache-b命中率更高，因此cache-b程序更优秀。

- 移动所需程序

```
[root@server0 ~]# mv cache-b /usr/local/bin/
```

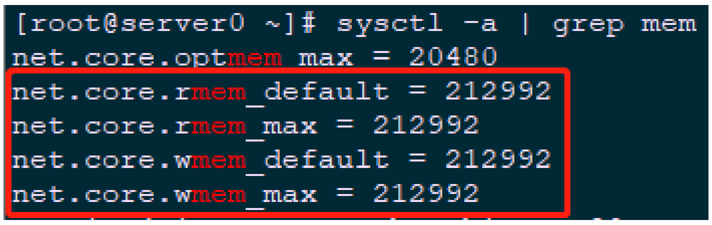
2.11 配置网络缓冲

Configure the network buffers for this system so that each UDP connection,both incoming and outgoing,is guaranteed a minimum of 128 KiB of buffer and a maximum of 192 KiB of buffer space.

解析:

- 查看当前参数

```
[root@server0 ~]# sysctl -a | grep mem
```



```
[root@server0 ~]# sysctl -a | grep mem
net.core.optmem_max = 20480
net.core.rmem_default = 212992
net.core.rmem_max = 212992
net.core.wmem_default = 212992
net.core.wmem_max = 212992
```

- 单位换算

```
[root@server0 ~]# bc
128*1024
131072
192*1024
196608
```

提示：sysctl 相关配置单位为byte，题意需要限制buffer为128kb和192kb之间，需要将单位进行换算。

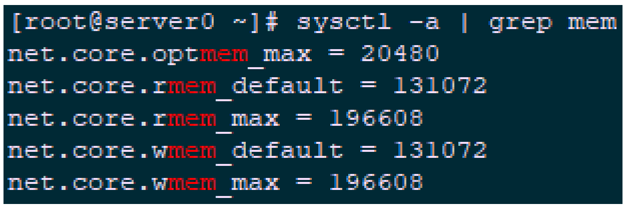
- 配置相关网络buffer

```
[root@server0 ~]# echo "net.core.rmem_default = 131072"
>>/etc/sysctl.d/rh422.conf
```

```
[root@server0 ~]# echo "net.core.wmem_default = 131072"
>>/etc/sysctl.d/rh422.conf
[root@server0 ~]# echo "net.core.rmem_max = 196608" >>/etc/sysctl.d/rh422.conf
[root@server0 ~]# echo "net.core.wmem_max = 196608" >>/etc/sysctl.d/rh422.conf
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh422.conf
```

- 确认验证

```
[root@server0 ~]# sysctl -a | grep mem
```



```
[root@server0 ~]# sysctl -a | grep mem
net.core.optmem_max = 20480
net.core.rmem_default = 131072
net.core.rmem_max = 196608
net.core.wmem_default = 131072
net.core.wmem_max = 196608
```

2.12 配置sar

Configure the sar data collection scripts on station to run at 3 minute intervals.

解析:

- 修改crontab调度

```
[root@server0 ~]# vi /etc/cron.d/sysstat
# Run system activity accounting tool every 10 minutes
*/3 * * * * root /usr/lib64/sa/sa1 1 1
[root@server0 ~]# systemctl restart sysstat
[root@server0 ~]# systemctl restart crond              #建议重启一次
[root@server0 ~]# systemctl enable crond
[root@server0 ~]# systemctl enable sysstat             #建议设置为开机启动
```

- 确认验证

```
[root@server0 ~]# systemctl status crond
[root@server0 ~]# systemctl status sysstat
```

2.13 为延迟的网络配置缓存

Most of the network connections between this system and other systems will be via a low earth orbit satellite link.The latency of this connection is approximately 500 milliseconds and the bandwidth of the link is 1.5 Mib/s(mebibits/second).Tune the system so that for all TCP connections:

The minimum amount of memory available for buffering per connection is large enough for this latency and bandwidth;
The default amount of memory available for buffering per connection is equal to the minimum amount of memory for buffering;

The maximum amount of memory available for buffering per connection is equal to 1.5 times the minimum amount of memory available for buffering.

解析：

- 查看当前参数

```
[root@server0 ~]# sysctl -a | grep mem
```

- 计算BDP值（带宽与时延积）

```
500ms=0.5s                              #毫秒换算成秒
1.5Mib/s÷8=0.1875MiB/s                  #比特换算成MByte
0.1875MiB/s*1024=192KiB/s               #MByte换算为KByte
192KiB/s*1024*0.5=98304Byte/s
```

即BDP为98304Byte/s。

```
[root@server0 ~]# bc
scale=8
1.5*0.5/8*1024*1024
98304.00000000
```

提示：注意单位为Mib/s，需要换算为B/s。

- 配置相关网络buffer—题型A

```
[root@server1 ~]# vi /etc/sysctl.d/rh422.conf
net.ipv4.tcp_mem = 41868 58255 83736         #单位为page，通常默认即可
net.core.wmem_max = 147456                   #单位为Byte，下同
net.core.rmem_max = 147456                   #根据题意最大memory为最小值1.5倍
net.ipv4.tcp_rmem=98304 98304 147456
net.ipv4.tcp_wmem=98304 98304 147456         #最小值、默认值、最大值
```

注意：此方式适合考试中存在两台虚拟机，与2.11题分别在两台虚拟机进行配置的情况；

net.ipv4.tcp_mem此项为tcp总阈值，单位为页，如上所示换算为K单位后的值为41868*4 58255*4 83736*4，若此总阈值大于之后每一项，则不需要调节；

若net.ipv4.tcp_mem此项tcp总阈值小于以下需要调整的项，则需要调整为24 24 36，即题意中需要调节的TCP参数值98304 98304 147456换算为页之后的值。

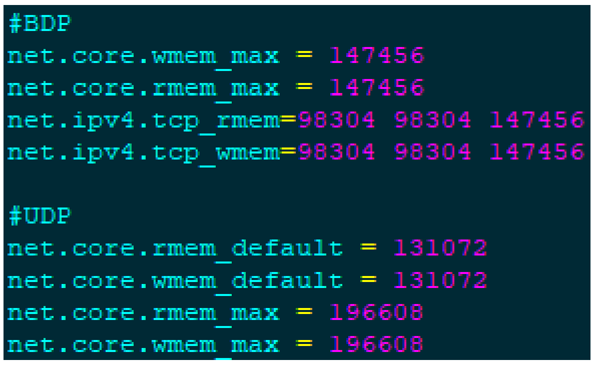
```
[root@server1 ~]# sysctl -p /etc/sysctl.d/rh422.conf
```

- 配置相关网络buffer—题型B

```
[root@server0 ~]# vi /etc/sysctl.d/rh422.conf
#BDP                                         #第十三题BDP所做调优置前
net.core.wmem_max = 147456
net.core.rmem_max = 147456
net.ipv4.tcp_rmem=98304 98304 147456
net.ipv4.tcp_wmem=98304 98304 147456
```

```
#UDP
net.core.rmem_default = 131072
net.core.wmem_default = 131072
net.core.rmem_max = 196608
net.core.wmem_max = 196608
```

#第十一题UDP所做调优置后



```
#BDP
net.core.wmem_max = 147456
net.core.rmem_max = 147456
net.ipv4.tcp_rmem=98304 98304 147456
net.ipv4.tcp_wmem=98304 98304 147456

#UDP
net.core.rmem_default = 131072
net.core.wmem_default = 131072
net.core.rmem_max = 196608
net.core.wmem_max = 196608
```

注意：此方式适合考试中只有一台虚拟机，与2.11题在同一台进行配置的情况；

net.ipv4.tcp_mem此项为tcp总阀值，单位为页，如上所示换算为K单位后的值为41868*4 58255*4 83736*4，若此总阀值大于之后每一项，则不需要调节；

若net.ipv4.tcp_mem此项tcp总阀值小于以下需要调整的项，则需要调整为24 24 36，即题意中需要调节的TCP参数值98304 98304 147456换算为页之后的值。

```
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh422.conf
```

- 确认验证

```
[root@serverX ~]# sysctl -a | grep mem
```

2.14 配置应用程序限制

There are two files in the /usr/local/bin directory:memapp1 and memapp2.The user memhog should be able to run memapp1 but should be prevented from running memapp2 or any other application which exhibits the same characteristic as memapp2.

解析:

- 查看页换算单位

```
[root@server0 ~]# getconf -a | grep PAGE
PAGESIZE                            4096
```

#确认每一个page为多少byte，即1page=4kb

- 检查用户及脚本权限

```
[root@server0 ~]# id memhog
```

提示：若用户或执行权限不存在，可通过以下方式创建用户和授权：

```
[root@server0 ~]# useradd memhog
[root@server0 ~]# echo redhat | passwd --stdin memhog
[root@server0 ~]# chmod uo+x /usr/local/bin/memapp*
```

- 分别检查两个程序实际运行所需内存

```
[root@server0 ~]# ./memapp1
……
Grabbing 4096 pages of memory
……
[root@server0 ~]# ./memapp2
……
Grabbing 8192 pages of memory
……
```

根据题意可知memapp1需要4096页，即4096*4=16384KB内存，memapp2需要8192页，即8192*4=32768KB内存。

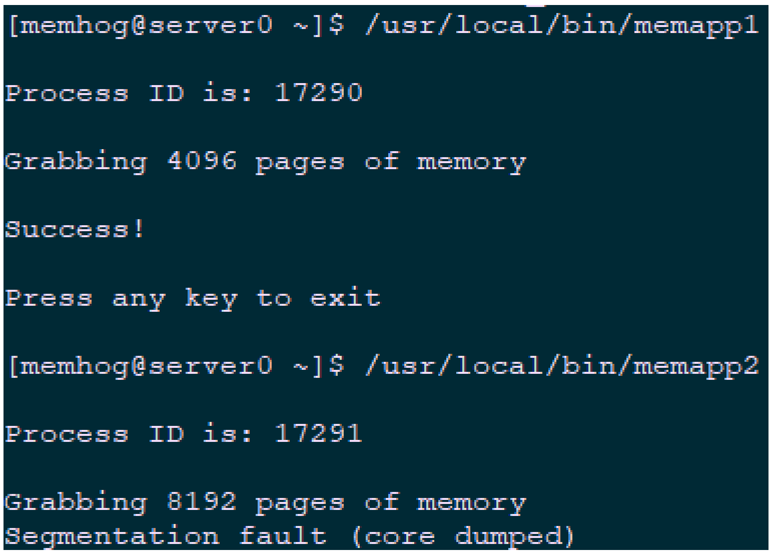
- 限制程序运行

```
[root@server0 ~]# vi /etc/security/limits.d/rh442-memhog.conf
memhog          hard            as              30000
```

提示：
根据题意通过对memhog能运行程序的最大内存限制，从而实现能运行memapp1但不能运行memapp2程序。
CentOS7系统主limits.conf配置文件为/etc/security/limits.conf，也可修改此文件来实现用户内存限制；
强烈建议不要修改系统以上自带配置文件，在/etc/security/limits.d目录下独立新建子配置文件EX-rh442-memhog.conf。

- 测试验证

```
[root@server0 ~]# su - memhog
[memhog@server0 ~]$ memapp1
[memhog@server0 ~]$ memapp2
```



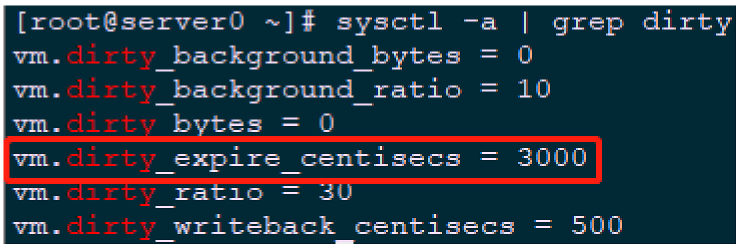
```
[memhog@server0 ~]$ /usr/local/bin/memapp1
Process ID is: 17290
Grabbing 4096 pages of memory
Success!
Press any key to exit
[memhog@server0 ~]$ /usr/local/bin/memapp2
Process ID is: 17291
Grabbing 8192 pages of memory
Segmentation fault (core dumped)
```

2.15　　配置脏页刷盘时间

Configure the system so that modified data can remain in memory for up to 45 second before considered for flushing to disk.
解析：

- 查看当前参数

```
[root@server0 ~]# sysctl -a | grep dirty
```



```
[root@server0 ~]# sysctl -a | grep dirty
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 3000
vm.dirty_ratio = 30
vm.dirty_writeback_centisecs = 500
```

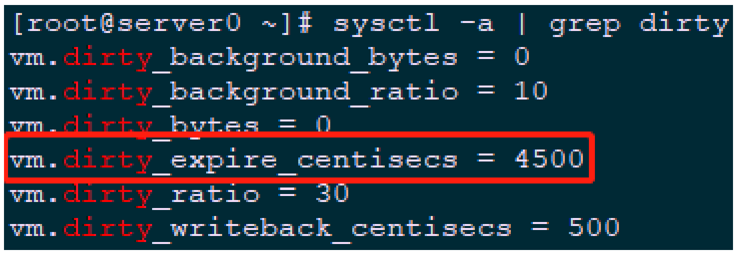
提示：系统默认3000单位为1/100秒，30秒，需要将题意中的时间45s换算为4500。

- 配置刷盘时间

```
[root@server0 ~]# echo "vm.dirty_expire_centisecs = 4500"
>>/etc/sysctl.d/rh422.conf
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh422.conf
```

- 确认验证

```
[root@server0 ~]# sysctl -a | grep dirty
```



```
[root@server0 ~]# sysctl -a | grep dirty
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 4500
vm.dirty_ratio = 30
vm.dirty_writeback_centisecs = 500
```

2.16-A版　　　　分析应用程序的性能

There is a tarball in your root home directory,called application.tgz that contains a client-server application.The file contains two versions of a server application,server-v1 and server-v2 along with a client application.Bother the client and server are designed to run on the same system.

The client application accept input from the keyboard and sends it to the server.Typing a ctrl-D will terminate the client.

Both versions of the server application do the same thing:they accept input from the client and copy it to the file /tmp/application.out.
Once it is put into production,the server application will be running for long periods without a restart and ,during the course of an average day,will be

serving out hundreds of connection requests.You must analyze each server application and identity the best version of the server application to use.

Once you have decided which version of the server application to use,copy your choices of the server application into /usr/local/bin.

解析:

- 解压题意所示压缩包

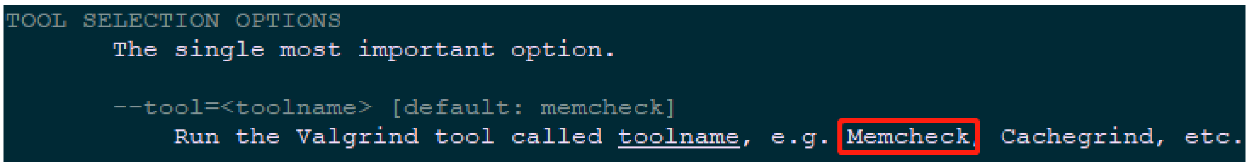
```
[root@server0 ~]# tar -zxvf application.tgz
```

- 相关命令

```
[root@server0 ~]# rpm -qa | grep valgrind
```

提示：使用2.10已安装valgrind软件，若未安装可通过以下命令安装。

```
[root@server0 ~]# yum -y install valgrind            #考试中安装x64_64版本
[root@server0 ~]# man valgrind                        #查看valgrind使用方法
```



```
TOOL SELECTION OPTIONS
       The single most important option.

       --tool=<toolname> [default: memcheck]
           Run the Valgrind tool called toolname, e.g. Memcheck, Cachegrind, etc.
```

- 依次运行题意所示程序

```
[root@server0 ~]# valgrind --tool=memcheck ./server-v1
[root@server0 ~]# valgrind --tool=memcheck ./server-v2
[root@server0 ~]# ./client
```

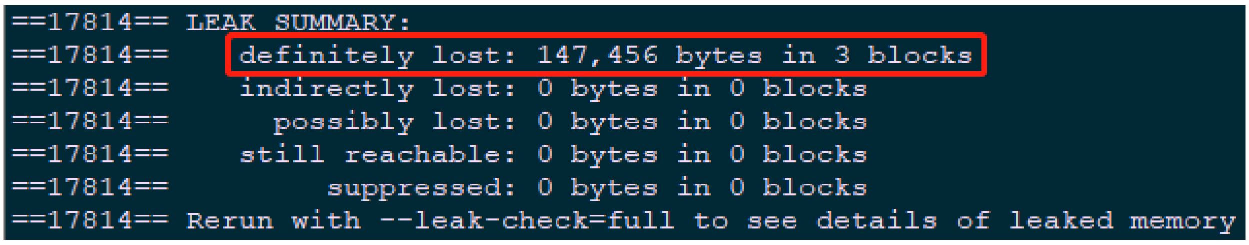
提示：必须打开多个窗口，同时运行以上程序，然后根据题意从client端随机输入相应字符。

- 测试程序并观察

```
[root@server0 ~]# ./client
fjdsalfjdsjfdsjfs;jfdsjfsafj                    #clent端随机输入字符，然后ctrl+d退出
[root@server0 ~]# valgrind --tool=memcheck ./server-v1          #client退出后ctrl+c退出
[root@server0 ~]# valgrind --tool=memcheck ./server-v2          #client退出后ctrl+c退出
```

- 比较优秀程序

```
[root@server0 ~]# valgrind --tool=memcheck ./server-v2
```



```
==17814== LEAK SUMMARY:
==17814==    definitely lost: 147,456 bytes in 3 blocks
==17814==    indirectly lost: 0 bytes in 0 blocks
==17814==      possibly lost: 0 bytes in 0 blocks
==17814==    still reachable: 0 bytes in 0 blocks
==17814==         suppressed: 0 bytes in 0 blocks
==17814== Rerun with --leak-check=full to see details of leaked memory
```

提示：通过LEAK SUMMARY可知，server-v2出现了内存泄露，则优秀的程序为server-v1。

- 复制脚本

```
[root@server0 ~]# cp server-v1 /usr/local/bin/
```

2.16-B版　　　　分析应用程序的性能

There is a tarball in your root home directory,called application.tgz that contains a client-server application.Bother the client and server are designed to run on the same system.

The client application is designed as a test utility for the server.The client accept input from the keyboard and sends it to the server.Typing a ctrl-D will terminate the client.

The server accept input form the client and logs it to the file /tmp/application.out.

Once it is put into production,the server application will be expected to run for long periods without a restart and ,during the course of an average day,will be serving out hundreds of connection requests.The expected amount of data to handled by the server will average around 50 KiB per second.The server will be deployed on a system which has sufficient disk space to handle this data rate and log file will be rotated daily to ensure that disk utilization does not become a concern.

Analyze the server application and identity,if any,problematic behavior that it exhibits,Record yuor answer below.

解析：

B版本类似A版本，仅从操作（cp操作）改变为下拉选择。

第一个下拉框选择：memory，第二个下拉框选择：leak。即内存泄漏。

提示：若提醒为B版，则解压后只有一个server，只需要判断此server运行后的状态——即内存溢出。

2.17　　加载模块

This system will be receiving a hardware upgrade in the form of a SCSI controller and a SCSI tape backup unit.Make sure that the SCSI tape controller module st.ko loads with a default buffer size of 24 KiB.You should verify that that when the module is loaded the kernel report the correct buffer size in the kernel ring buffer.

解析：

- 检查命令

```
[root@server0 ~]# which modprobe
```

```
/usr/sbin/modprobe
```

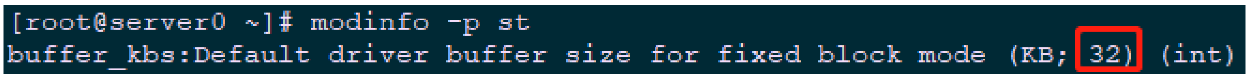
提示：modprobe可能在/usr/sbin/和/sbin/modprobe下都存在，任一路径下命令都可以。

- 创建st加载模块

```
[root@server0 ~]# vi /etc/sysconfig/modules/st.modules          #建议根据题意命名
/sbin/modprobe st
[root@server0 ~]# chmod u+x /etc/sysconfig/modules/st.modules   #授权
```

- 修改默认缓存大小

```
[root@server0 ~]# modinfo -p st                           #查看初始缓存大小
```



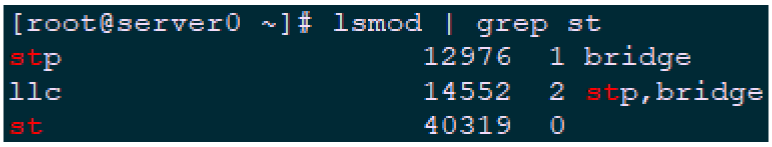
```
[root@server0 ~]# modinfo -p st
buffer_kbs:Default driver buffer size for fixed block mode (KB; 32) (int)
```

```
[root@server0 ~]# vi /etc/modprobe.d/st.conf              #建议根据题意命名
options st buffer_kbs=24
```

提示：根据查询出的参数buffer_kbs，在/etc/modprobe.d目录下创建*.conf结尾的配置文件，并写入题意所需值。

- 确认验证

```
[root@server0 ~]# shutdown -r now                         #重启当前系统
[root@server0 ~]# lsmod | grep st
```



```
[root@server0 ~]# lsmod | grep st
stp                    12976  1 bridge
llc                    14552  2 stp,bridge
st                     40319  0
```

```
[root@server0 ~]# cat /sys/bus/scsi/drivers/st/fixed_buffer_size
24576
```

提示：此参数值单位为byte，换算成KiB，24576/1024=24KiB。

2.18 I/O调优

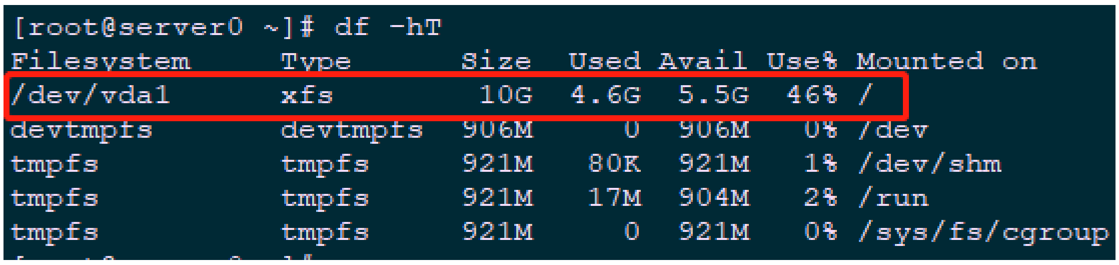
You want the majority of the disk I/O that is performed by the primary application running on your virtual system to have to wait no longer than a guaranteed amount of times before the I/O request is serviced.

Create an automatic tunning profile named exam that sets the default I/O scheduler for the drive that contains the root filesystem for your virtual system to the appropriate I/O scheduler.For the purpose of the exam you may consider the drive to be an actual physical device.This profile should be enabled and the default profile when the system boots.

解析：

- 确认需要I/O调度的磁盘

```
[root@server0 ~]# df -hT
```



```
[root@server0 ~]# df -hT
Filesystem        Type      Size  Used Avail Use% Mounted on
/dev/vda1         xfs        10G  4.6G  5.5G  46% /
devtmpfs          devtmpfs  906M     0  906M   0% /dev
tmpfs             tmpfs     921M   80K  921M   1% /dev/shm
tmpfs             tmpfs     921M   17M  904M   2% /run
tmpfs             tmpfs     921M     0  921M   0% /sys/fs/cgroup
```

提示：根据题意暗含需要调度的磁盘为当前系统/目录所在分区。

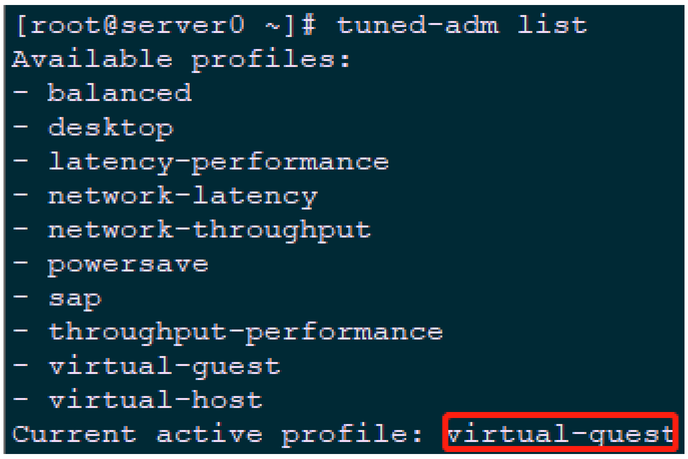
- 安装所需软件并启动

```
[root@server0 ~]# yum -y install tuned
[root@server0 ~]# systemctl start tuned
[root@server0 ~]# systemctl enable tuned
[root@server0 ~]# systemctl status tuned
```

- 查看已存在算法

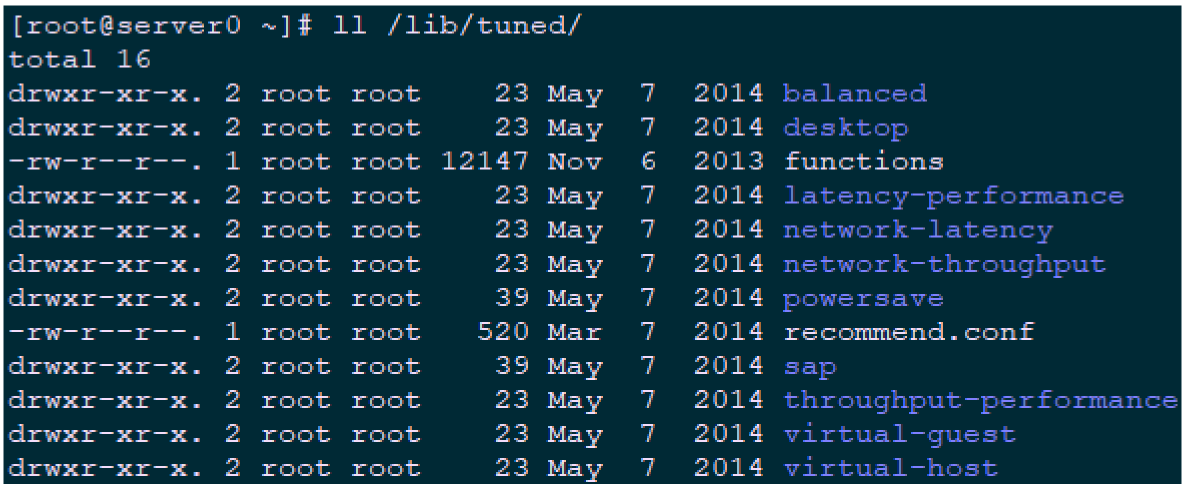
[root@server0 ~]# tuned-adm -h #查看命令帮助

[root@server0 ~]# tuned-adm list #查看已存在算法及当前采用算法



```
[root@server0 ~]# tuned-adm list
Available profiles:
- balanced
- desktop
- latency-performance
- network-latency
- network-throughput
- powersave
- sap
- throughput-performance
- virtual-guest
- virtual-host
Current active profile: virtual-guest
```

提示：所有系统自带的调度方案目录为：/lib/tuned/，其具体调度算法具体配置文件为其目录下的tuned.conf文件，如下所示：



```
[root@server0 ~]# ll /lib/tuned/
total 16
drwxr-xr-x. 2 root root    23 May  7  2014 balanced
drwxr-xr-x. 2 root root    23 May  7  2014 desktop
-rw-r--r--. 1 root root 12147 Nov  6  2013 functions
drwxr-xr-x. 2 root root    23 May  7  2014 latency-performance
drwxr-xr-x. 2 root root    23 May  7  2014 network-latency
drwxr-xr-x. 2 root root    23 May  7  2014 network-throughput
drwxr-xr-x. 2 root root    39 May  7  2014 powersave
-rw-r--r--. 1 root root   520 Mar  7  2014 recommend.conf
drwxr-xr-x. 2 root root    39 May  7  2014 sap
drwxr-xr-x. 2 root root    23 May  7  2014 throughput-performance
drwxr-xr-x. 2 root root    23 May  7  2014 virtual-guest
drwxr-xr-x. 2 root root    23 May  7  2014 virtual-host
```

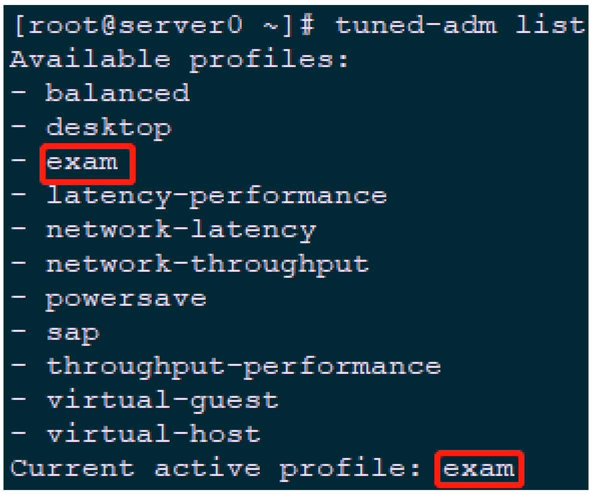
- 创建自定义调度配置文件并导入

```
[root@server0 ~]# mkdir /etc/tuned/exam                    #创建题意要求的exam
[root@server0 ~]# vi /etc/tuned/exam/tuned.conf
[disk]
elevator=deadline
[root@server0 ~]# tuned-adm profile exam                   #手动切换配置文件
```

提示：也可手动修改/etc/tuned/active_profile配置文件，指定调度方案为exam。

- 确认验证

```
[root@server0 ~]# tuned-adm list                           #确认验证
```



```
[root@server0 ~]# tuned-adm list
Available profiles:
- balanced
- desktop
- exam
- latency-performance
- network-latency
- network-throughput
- powersave
- sap
- throughput-performance
- virtual-guest
- virtual-host
Current active profile: exam
```

```
[root@server0 ~]# cat /sys/block/vda/queue/scheduler       #查看调度算法
noop [deadline] cfq
```

提示：根据题意要求，选择采用deadline算法，具体deadline细节见扩展。

扩展：

deadline算法对request进行了优先权控制调度，主要表现在如下几个方面：

1）读写请求分离，读请求具有高优先调度权，除非写请求即将被饿死的时候，才会去调度处理写请求。这种处理可以保证读请求的延迟时间最小化。

2）对请求的顺序批量处理。对那些地址临近的顺序化请求，deadline给予了高优先级处理权。例如一个写请求得到调度后，其临近的request会在紧接着的调度过程中被处理掉。这种顺序批量处理的方法可以最大程度的减少磁盘抖动。

3）保证每个请求的延迟时间。每个请求都赋予了一个最大延迟时间，如果达到延迟时间的上限，那么这个请求就会被提前处理掉，此时，会破坏磁盘访问的顺序化特征，回影响性能，但是，保证了每个请求的最大延迟时间。

2.19 配置cgroup

configure a cgroup slice named power on your system according to the following requirements:

The slice should be limited to 1024 relative CPU shares and 1024M amount of memory;

Make sure that the Httpd service runs under the power slice automatically at boot time;

Do not use the yes Bool flag on the slice unit。

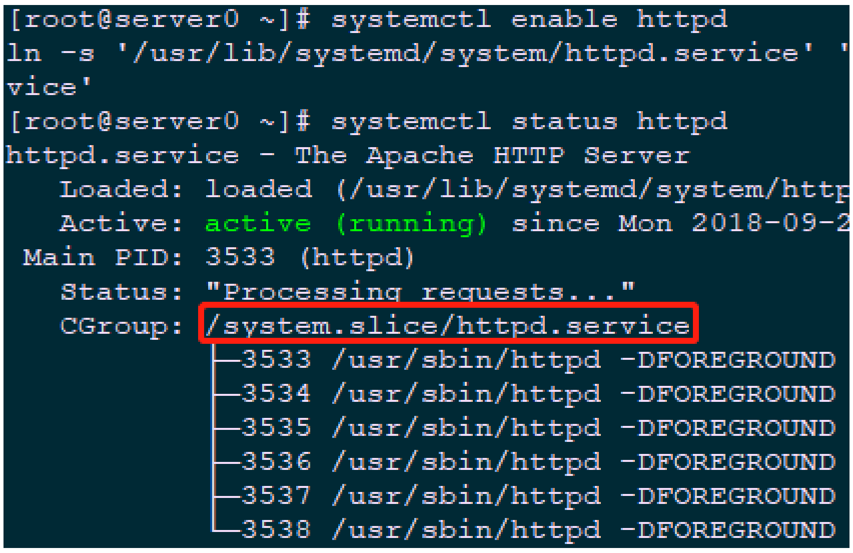
解析:

- 安装所需软件并启动

```
[root@server0 ~]# yum -y install httpd
[root@server0 ~]# systemctl start httpd
[root@server0 ~]# systemctl enable httpd
[root@server0 ~]# systemctl status httpd
```

- 查看当前slice

```
[root@server0 ~]# systemctl status httpd
```



```
[root@server0 ~]# systemctl enable httpd
ln -s '/usr/lib/systemd/system/httpd.service' '
vice'
[root@server0 ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/http
   Active: active (running) since Mon 2018-09-2
 Main PID: 3533 (httpd)
   Status: "Processing requests..."
   CGroup: /system.slice/httpd.service
           ├─3533 /usr/sbin/httpd -DFOREGROUND
           ├─3534 /usr/sbin/httpd -DFOREGROUND
           ├─3535 /usr/sbin/httpd -DFOREGROUND
           ├─3536 /usr/sbin/httpd -DFOREGROUND
           ├─3537 /usr/sbin/httpd -DFOREGROUND
           └─3538 /usr/sbin/httpd -DFOREGROUND
```

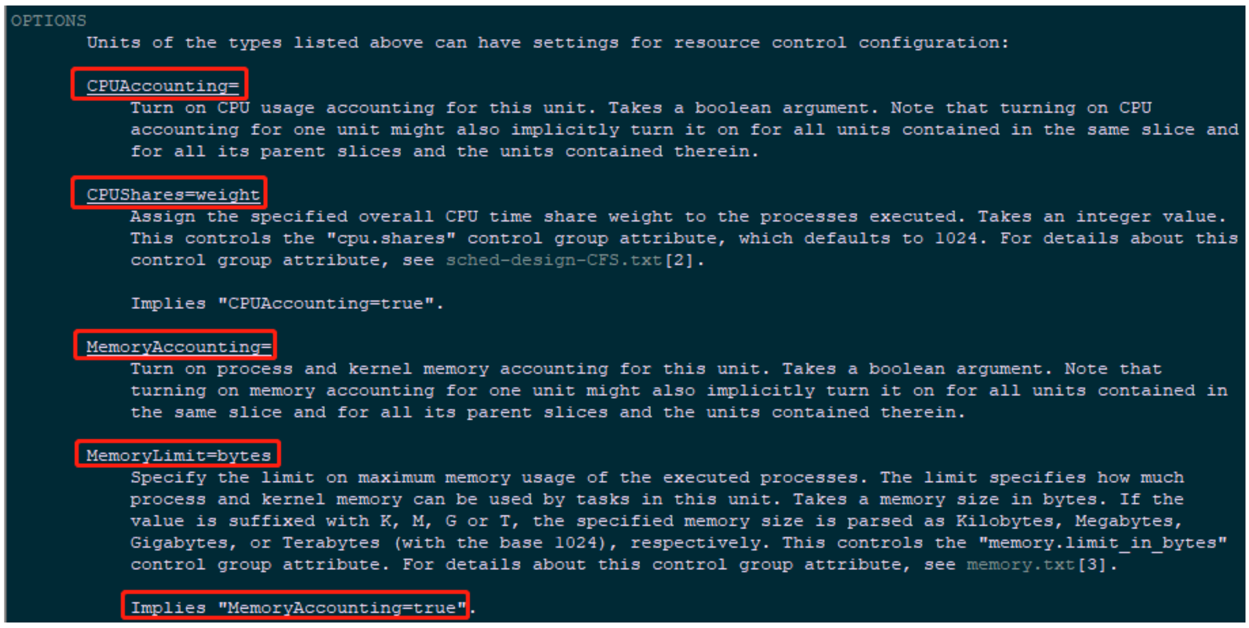
提示：如上所示，可知当前httpd启动采用slice为system.slice。

- 查找所需参数、

```
[root@server0 ~]# mandb                         #必须刷新man数据库
[root@server0 ~]# man -k systemd.re
systemd.resource-control (5) - Resource control unit settings
```

提示：有时候刷新man之后依旧找不到关于资源控制的条目，可推出重新登录即可。

```
[root@server0 ~]# man systemd.resource-control
```


```
OPTIONS
       Units of the types listed above can have settings for resource control configuration:

       CPUAccounting=
           Turn on CPU usage accounting for this unit. Takes a boolean argument. Note that turning on CPU
           accounting for one unit might also implicitly turn it on for all units contained in the same slice and
           for all its parent slices and the units contained therein.

       CPUShares=weight
           Assign the specified overall CPU time share weight to the processes executed. Takes an integer value.
           This controls the "cpu.shares" control group attribute, which defaults to 1024. For details about this
           control group attribute, see sched-design-CFS.txt[2].

           Implies "CPUAccounting=true".

       MemoryAccounting=
           Turn on process and kernel memory accounting for this unit. Takes a boolean argument. Note that
           turning on memory accounting for one unit might also implicitly turn it on for all units contained in
           the same slice and for all its parent slices and the units contained therein.

       MemoryLimit=bytes
           Specify the limit on maximum memory usage of the executed processes. The limit specifies how much
           process and kernel memory can be used by tasks in this unit. Takes a memory size in bytes. If the
           value is suffixed with K, M, G or T, the specified memory size is parsed as Kilobytes, Megabytes,
           Gigabytes, or Terabytes (with the base 1024), respectively. This controls the "memory.limit_in_bytes"
           control group attribute. For details about this control group attribute, see memory.txt[3].

           Implies "MemoryAccounting=true".
```

提示：根据题意，针对资源控制只需要前四个参数即可。

- 创建slice

```
[root@server0 ~]# vi /etc/systemd/system/power.slice
[Unit]
Description=http Slice


[Slice]
CPUAccounting=true
CPUShares=1024
MemoryAccounting=true
MemoryLimit=1024M
```

提示：根据题意对相关资源进行限制，若题意说明不能使用true的布尔值，可采用yes配置，参考如下：

CPUAccounting=true，MemoryAccounting=true

MemoryAccounting=true

也在直接在系统当前slice所在路径创建新slice——

[root@server0 ~]# find / -type f -name system.slice   #查找系统自带slice路径

/usr/lib/systemd/system/system.slice

[root@server0 ~]# vi /usr/lib/systemd/system/power.slice   #创建如上所示内容的slice

- 应用策略

```
[root@server0 ~]# mkdir /etc/systemd/system/httpd.service.d
[root@server0 ~]# vi /etc/systemd/system/httpd.service.d/rh442-http.conf
[Service]
Slice=power.slice
```

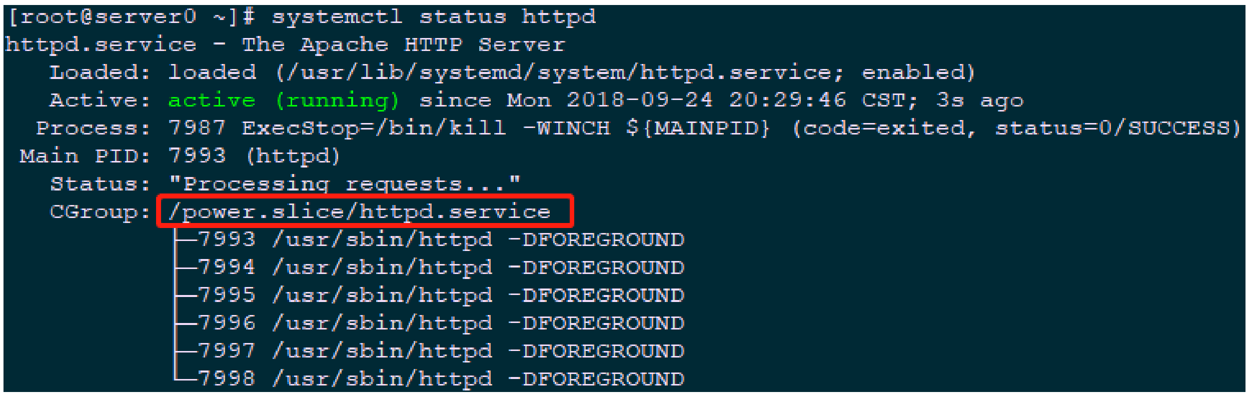
提示：必须创建名为httpd.service.d的目录，conf文件名可自定义；

也可参考如下在当前系统安装的httpd已存在的httpd.service中[Service]模块进行关联新slice参数：

```
[root@server0 ~]# find / -type f -name httpd.service     #查看当前系统httpd.service文件
/usr/lib/systemd/system/httpd.service
[root@server0 ~]# vi /usr/lib/systemd/system/httpd.service
……
[Service]
……
Slice=power.slice
……
```

- 确认验证

```
[root@server0 ~]# systemctl daemon-reload
[root@server0 ~]# systemctl restart httpd
[root@server0 ~]# systemctl status httpd
```



```
[root@server0 ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
   Active: active (running) since Mon 2018-09-24 20:29:46 CST; 3s ago
  Process: 7987 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/SUCCESS)
 Main PID: 7993 (httpd)
   Status: "Processing requests..."
   CGroup: /power.slice/httpd.service
           ├─7993 /usr/sbin/httpd -DFOREGROUND
           ├─7994 /usr/sbin/httpd -DFOREGROUND
           ├─7995 /usr/sbin/httpd -DFOREGROUND
           ├─7996 /usr/sbin/httpd -DFOREGROUND
           ├─7997 /usr/sbin/httpd -DFOREGROUND
           └─7998 /usr/sbin/httpd -DFOREGROUND
```

2.20 分析pre-recorded

The file /root/.pcp/pmlogger/20141231.06.00.01.folio contains approximately one hour of pre-recorded system activity from a production server.At around 40 minutes and 58 seconds in the recording,there is a sudden burst of I/O activity. Analyze the collected data and answer the following questions:

Which device has the highest write IO rate??

What is the highest write IO rate?(MB, Select the closest value)

What is the highest write IO rate for another device?(MB, Select the closest value)

解析：

- 安装所需软件并启动

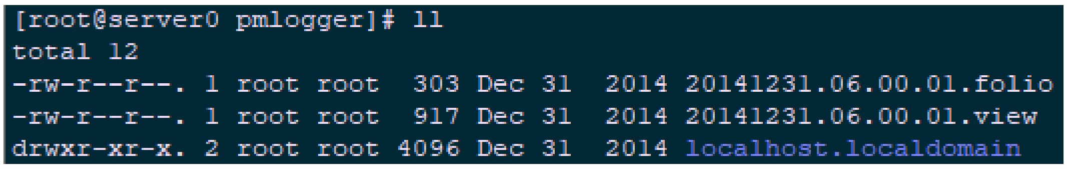
```
[root@server0 ~]# yum -y install pcp pcp-gui
```

```
[root@server0 ~]# systemctl start pcp
[root@server0 ~]# chkconfig pcp on
```

- 查看文件类型及监控参数

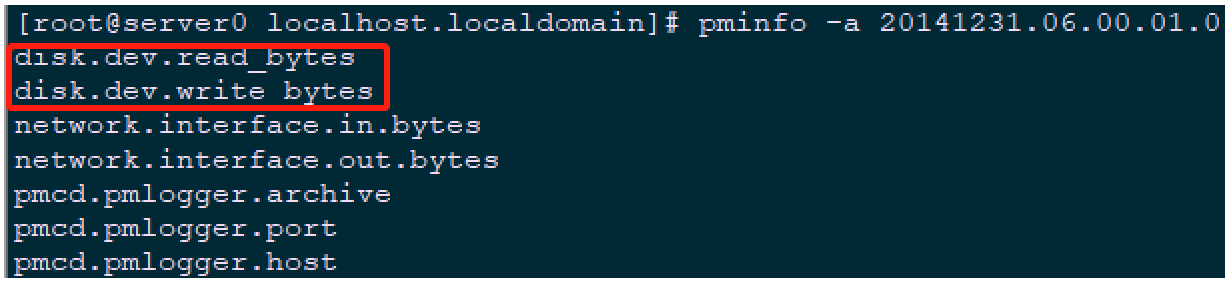
```
[root@server0 ~]# cd /root/.pcp/pmlogger/
[root@server0 pmlogger]# ll
```

提示：若考试环境通常给出的为cpio文件，可通过cpio -idmv <20141231.06.00.01.cpio方式解压，若为gz可通过tar -zxvf解压。



```
[root@server0 pmlogger]# ll
total 12
-rw-r--r--. 1 root root  303 Dec 31  2014 20141231.06.00.01.folio
-rw-r--r--. 1 root root  917 Dec 31  2014 20141231.06.00.01.view
drwxr-xr-x. 2 root root 4096 Dec 31  2014 localhost.localdomain
```

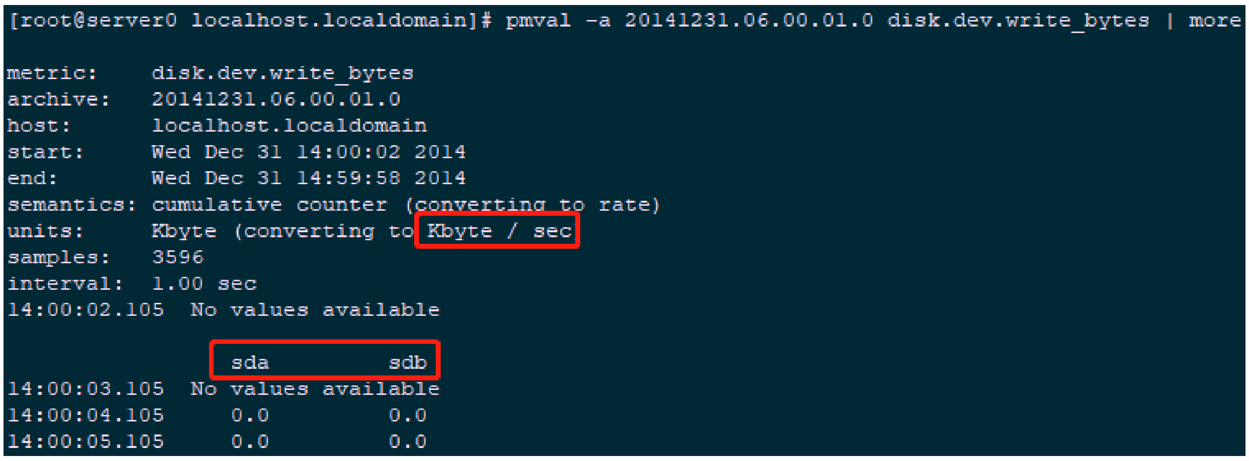
```
[root@server0 ~]# cd /root/.pcp/pmlogger/localhost.localdomain
[root@server0 localhost.localdomain]# pminfo -a 20141231.06.00.01.0
```
#确认所需读写指标



```
[root@server0 localhost.localdomain]# pminfo -a 20141231.06.00.01.0
disk.dev.read_bytes
disk.dev.write_bytes
network.interface.in.bytes
network.interface.out.bytes
pmcd.pmlogger.archive
pmcd.pmlogger.port
pmcd.pmlogger.host
```

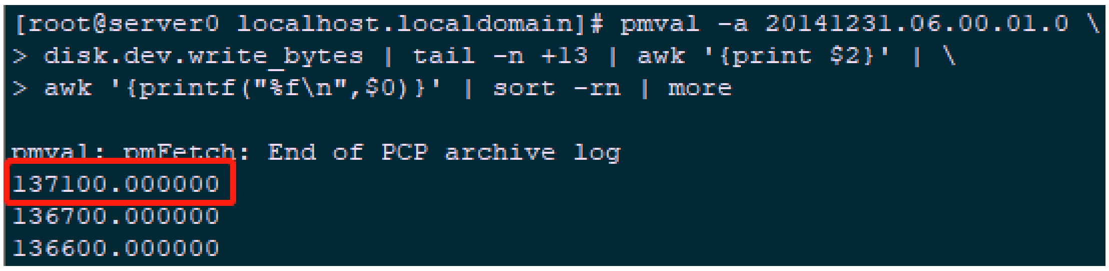
- 查看写入速率最高的设备

```
[root@server0 localhost.localdomain]# pmval -a 20141231.06.00.01.0
disk.dev.write_bytes | more
```
#查看相关设备所在列及单位



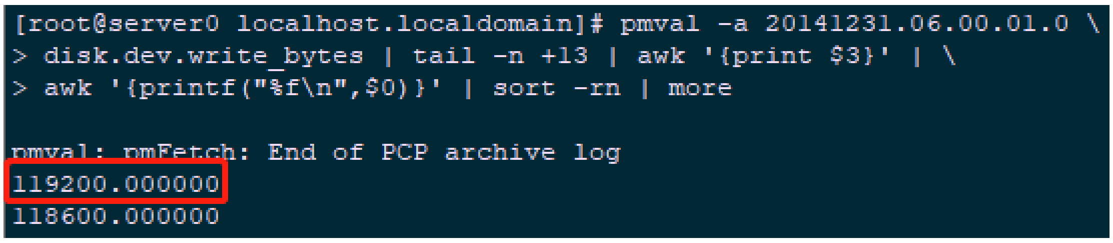
```
[root@server0 localhost.localdomain]# pmval -a 20141231.06.00.01.0 disk.dev.write_bytes | more

metric:    disk.dev.write_bytes
archive:   20141231.06.00.01.0
host:      localhost.localdomain
start:     Wed Dec 31 14:00:02 2014
end:       Wed Dec 31 14:59:58 2014
semantics: cumulative counter (converting to rate)
units:     Kbyte (converting to Kbyte / sec
samples:   3596
interval:  1.00 sec
14:00:02.105  No values available

                 sda          sdb
14:00:03.105  No values available
14:00:04.105     0.0          0.0
14:00:05.105     0.0          0.0
```

```
[root@server0 localhost.localdomain]# pmval -a 20141231.06.00.01.0 \
disk.dev.write_bytes | tail -n +13 | awk '{print $2}' | \
awk '{printf("%f\n",$0)}' | sort -rn | more
```
#查看sda设备最大写速率



```
[root@server0 localhost.localdomain]# pmval -a 20141231.06.00.01.0 \
> disk.dev.write_bytes | tail -n +13 | awk '{print $2}' | \
> awk '{printf("%f\n",$0)}' | sort -rn | more

pmval: pmFetch: End of PCP archive log
137100.000000
136700.000000
136600.000000
```

```
[root@server0 localhost.localdomain]# pmval -a 20141231.06.00.01.0 \
disk.dev.write_bytes | tail -n +13 | awk '{print $3}' | \
awk '{printf("%f\n",$0)}' | sort -rn | more
```
#查看sdb设备最大写速率



```
[root@server0 localhost.localdomain]# pmval -a 20141231.06.00.01.0 \
> disk.dev.write_bytes | tail -n +13 | awk '{print $3}' | \
> awk '{printf("%f\n",$0)}' | sort -rn | more

pmval: pmFetch: End of PCP archive log
119200.000000
118600.000000
```

提示：由上可知写IO最高的设备为sda。

- 计算最高设备（sda）的写速率

```
[root@server0 ~]# bc
scale=8
137100/1024
133.88671875
```

提示：由上可知写IO最高的设备sda的写入速率可取值134（或选项中最接近计算后的值）。

- 计算最高设备（sdb）的写速率

```
[root@server0 ~]# bc
scale=8
119200/1024
116.40625000
```

提示：由上可知写IO最高的设备sda的写入速率可取值117（或选项中最接近计算后的值）。

2.21 配置大页

There are two versions of a program in /root that both allocate a 64 MiB segment of shared memory.Configure your system so that it can run one or the other of these program s according th the following requirements:

The amount of memory consumed by the page tables for the process running the program is minimized;

Assuming this system is primarily running online this program,TLB flushes are minimize when the program is running;

The configuration changes you make persist across reboots.

The program hugepage.shm uses SYSV shared memory to allocate the 64 MiB segment of memory. The program hugepage.fs uses a pseudofilesystem to handle the memory allocation.

You only need to configure your system to support one or the other program. You do not need to configure your system to support both. You should copy only the program you decide to use into the /usr/local/bin directory.

If you choose to support the hugepage.fs program, the pseudofilesystem should be mounted to the directory /bigpages. This filesystem should be automatically mounted after a system reboot. When running the hugepage.fs program, you will be prompted for the name of a file under the mountpoint for your pseudofilesystem: you should use the file /bigpages/memory.

Both programs will pause after allocating the shared memory and prompt you to continue when you are ready. This is so that you may examine the system while the program is running to confirm that memory is being properly allocated.

解析:

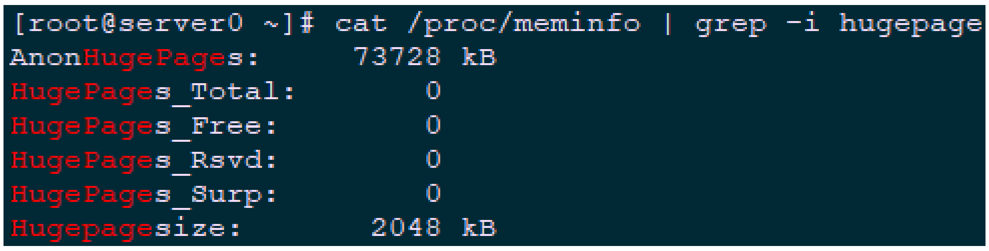
- 检查权限

```
[root@server0 ~]# ll /root/hugepage.shm
[root@server0 ~]# ll /root/hugepage.fs
```

提示：若题意所示应用程序无执行权限，可通过chmod u+x /root/hugepage.*进行授权。

- 查看当前大页大小

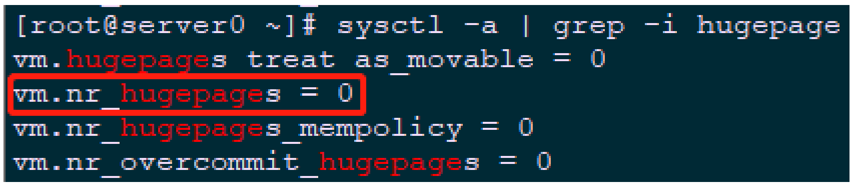
```
[root@server0 ~]# cat /proc/meminfo | grep -i hugepage
```



```
[root@server0 ~]# cat /proc/meminfo | grep -i hugepage
AnonHugePages:      73728 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
```

提示：查看结果可知，当前一个大页大小为2M。

```
[root@server0 ~]# sysctl -a | grep -i hugepage
```



```
[root@server0 ~]# sysctl -a | grep -i hugepage
vm.hugepages_treat_as_movable = 0
vm.nr_hugepages = 0
vm.nr_hugepages_mempolicy = 0
vm.nr_overcommit_hugepages = 0
```

- 修改大页大小

```
[root@server0 ~]# sysctl -a | grep -i hugepage
[root@server0 ~]# echo "vm.nr_hugepages = 32" >>/etc/sysctl.d/rh422.con
```

```
[root@server0 ~]# sysctl -p /etc/sysctl.d/rh422.con
```

提示：每个大页大小为2M，根据题意所需配置的大页大小为64M，因此此处配置为32。

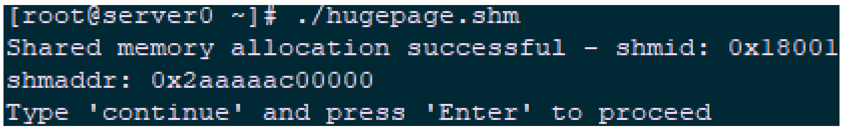
- 复制脚本

```
[root@server0 ~]# cp /root/hugepage.shm /usr/local/bin/
```

提示：根据题意选择共享内存方式，即hugepage.shm脚本即可。

- 确认验证

```
[root@server0 ~]# hugepage.shm                        #运行测试
```



```
[root@server0 ~]# ./hugepage.shm
Shared memory allocation successful - shmid: 0x18001
shmaddr: 0x2aaaaac00000
Type 'continue' and press 'Enter' to proceed
```