Lab 3 Report

Tejas Nitin Joshi

ECEN 5653 Real Time Digital Media

1.[5 pts]Read A Case for Redundant Arrays of Inexpensive Disks (RAID)**.**
Please summarize the paper's main points (at least 3 or more) in a short paragraph.s

This paper introduces the concept of RAID and explains the different types of RAID configurations/levels. It also discusses advantages over a Single Large Expensive disk(SLED) like reliability, scalability and performance. RAID arrays provide fault tolerance in the form of redundancy . Calculations involving number of reads, wirites, and read-modify-writes determine the life of a disk, which is specially important in hi-speed super computers which operate at high data transaction rates. We must compare the six categories of transactions. Based on these transactions types different kinds of RAID levels are proposed.

RAID 1 (i.e level one) takes on a mirrored approach where there is a backup disk for every datat disk present in the system. Cost for a redundant storage drive is a additional cost for every data disk. The RAID 2 level uses hamming code for Error Checking and Correction, so we need additional disk are requried to store the codes. These disks are called check disks (C) and the number of data disks a denoted by D. In level 2 configuration for every 10 Data disks 4 Check disks are required. Which is a improvement over RAID 1. Applications which require bursty small valume data , usually performs a read-modify-write transacttion causing yhe Group size to increase , which is inappropriate for supercomputers.

RAID 3 uses a single check disk per group , this disk contains parity information from all the other disks. Although this is not commonly used.

RAID 4 is faster in terms of transactions as it splits the data and writes them simeltaneously on multiple data disks and a single drive is maintained for parity.RAID 5 is similar to RAID 4 but it distributes the parity information also makeing it more efficient than RAID 4.

1.**[10 pts]Read** Erasure Code Overview Slides **and** Erasure Code Performance Comparison Paper **and reference** IDF Data Durability Presentation **to understand the added value of erasure encoding compared to traditional RAID levels.**
Please summarize both paper's main points (at least 3 or more) in a short paragraph for each and describe the value of Erasure codes compared to traditional RAID5 and RAID6.

The paper focuses on relaiability in RAID systems contrasting it to the transaction speedup observed in these systems. A remedy to diskfaliures in terms of mathamatical calculations is required for prevventing losses. In a system which contains "k" data disks and "m" parity disks, only a minimum of "m" disk faliures is allowed , so data lost can be reconstructed. A example of such algorithms are the Reed-Solomon code and the Cauchy Reed-Solomon code which use Vandermonde matrices and Genereators for reconstructing data lost during disk failure. A experiment done by the aouthors on twwo machines for decoding and generating parity data showd that that smaller files and less tight loops in coding provided better performance. It was seen RAID 6 provided better prtformance with w =6 on a Macbook, and also RAID 6 performance was better than other levels in terms of speed and data reliability.

Erasure Code Overview slides also emphasizes the advantages and applications of the code in RAID systems. The slides go on to explain the run time complexities

O(mn) = complexity for encoding and decoding

O(m) = updating

n/(n+m)space requirement efficiency

The encoding process involves parity generation for data and decoding involves the reconstructing the data from the parity and the corruptted data.

The Cauchy Reed-Solomon alogoithm take O(wnm) encoding and describes the evenoff method for generation of parity byte. Evenodd encodong reqires O(n^2). In Xcode each coding row is calculated for oppositely sloped diagonals with the same complexity. Other chemes are for encoding and decoding are LDPC, Hover codesm adm Balum Roth algorithm.

2.**[5 pts extra credit]Read** Intelligent RAID 6 Theory - Overview and Implementation**.**
Please summarize the paper's main points (at least 3 or more) in a short paragraph and clearly describe the mathematics behind the 7 erasure scenarios and how data can be recovered on the fly.

The paper explains the concept and implementation of RAID 6 and describes why it is better than RAID 5. The RAID 5 standard can only compensate for single disk faliures and hence would be a lesser choice when reliability is at stake. Failure of more tha one disk would resul in loss of data and functionality. Wheras RAID 6 has counter measures that can reconstruct data even when two disk fail at the same time. In terms of MTDL(Mean Time to Data Loss) the RAID 5 has a poor reliability performace limiting disk life to 1 month, This can be extended to 100 years when using RAID 6. The paper describes how RAID6 uses MDS codes to protect data from Erasure. A read solomon code with n total symbols , k data and 2t parity bits can detect upto 2t erasures and correct upto t erasures. The paper also discusses P+Q RAID6 formed codewords and their parity symbol generations. The paper describes the Falois Field Element Generation method by ising the primitive elements which are roots of a primitive polynomial. The first to elemnts being {0,1} and the succesive progression of elements as {0,1, α} =>{0,1, α, α2 }. The paper describes acceleration by Intel IOP333 which is multifunction device that integrates Intel XScale which is essential part of dual PCI Express to PCI-X bridges. The IOP333 AAU checks for all sero, fills memory blocks m computes XOR values for memoy blocks local memory

core with intelligent peripherals and dual PCI-Express to PCI-X bridges. The IntelIOP AAU

(Application Acceleration Unit) provides low latency, high throughput data transfer between AAU

and IOP333 local memory, checks for all zeros, fills memory blocks, computes XOR and multiplication for enhancing RAID 6 operation , which would otherwise load the CPU unecessarily.

3.[10 pts]Examining PCI devices in Linux (2 part question):
Please use your Fedora Core home lab Linux system. You will find "lspci", "update-pciids", and "setpci" to be helpful Linux commands for this part of the assignment.

1.[5 pts] Use *lspci* command on you home lab machine shell prompt and verify the PCI devices that you see are in fact installed in your system. Please provide a dump of all devices you see and identify their bus, device, and function identifications. Do you see any multi-function devices and if so, please highlight and describe why you believe this device has multiple functions.

Output

```
drix@lynx:~/prog/lab2$ lspci

00:00.0 Host bridge: Intel Corporation 2nd Generation Core Processor Family DRAM Controller (rev 09)

00:01.0 PCI bridge: Intel Corporation Xeon E3-1200/2nd Generation Core Processor Family PCI Express Root Port (rev 09)

00:02.0 VGA compatible controller: Intel Corporation 2nd Generation Core Processor Family Integrated Graphics Controller (rev 09)

00:16.0 Communication controller: Intel Corporation 6 Series/C200 Series Chipset Family MEI Controller #1 (rev 04)

00:1a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #2 (rev 05)

00:1b.0 Audio device: Intel Corporation 6 Series/C200 Series Chipset Family High Definition Audio Controller (rev 05)

00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 1 (rev b5)

00:1c.1 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 2 (rev b5)

00:1c.3 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 4 (rev b5)

00:1c.4 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 5 (rev b5)

00:1c.7 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port 8 (rev b5)

00:1d.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset Family USB Enhanced Host Controller #1 (rev 05)

00:1f.0 ISA bridge: Intel Corporation HM67 Express Chipset Family LPC Controller (rev 05)

00:1f.2 SATA controller: Intel Corporation 6 Series/C200 Series Chipset Family 6 port SATA AHCI Controller (rev 05)
```

00:1f.3 SMBus: Intel Corporation 6 Series/C200 Series Chipset Family SMBus Controller (rev 05)

01:00.0 VGA compatible controller: NVIDIA Corporation GF108 [GeForce GT 540M] (rev ff)

01:00.1 Audio device: NVIDIA Corporation GF108 High Definition Audio Controller (rev ff)

05:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8101E/RTL8102E PCI Express Fast Ethernet controller (rev 05)

09:00.0 Network controller: Atheros Communications Inc. AR9285 Wireless Network Adapter (PCI-Express) (rev 01)

0b:00.0 USB controller: Texas Instruments Device 8241 (rev 02)

The initial portion of every like indcates the bus number the next two digits are for the device number and then the function number.

The computer has

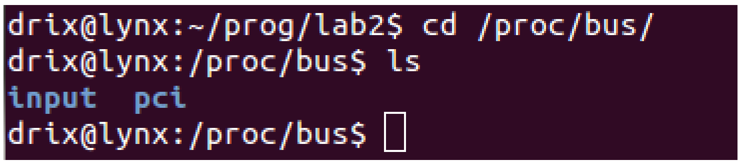
PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family PCI Express Root Port X (rev b5)

which is a multifunction device also the device 1f is also a multifunction device. This device acts as a SATA controller, SMBus Controller and ISA Bridge

A multifunction device can be identified by looking if the device number is constant in the lspci output and the function number remains the same.

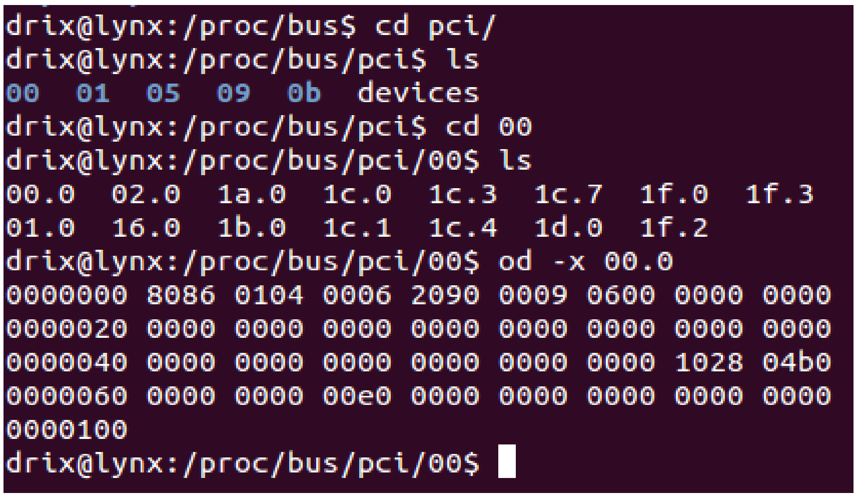
2.**[5 pts] Change directory to the *\/proc/bus* directory on a lab system and describe what you see. Now go down into */proc/bus/pci* and then down into bus "00" entry. Do *od -x 00.0* to dump binary snapshot of bus 0, device 0 configuration registers. Verify the device ID and vendor ID register fields and identify them in the hex dump using web based PCIDatabase.com.**

The /proc/bus directory consists of two folders:



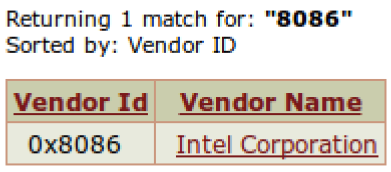
```
drix@lynx:~/prog/lab2$ cd /proc/bus/
drix@lynx:/proc/bus$ ls
input  pci
drix@lynx:/proc/bus$ 
```

Examining /proc/bus/pci



```
drix@lynx:/proc/bus$ cd pci/
drix@lynx:/proc/bus/pci$ ls
00  01  05  09  0b  devices
drix@lynx:/proc/bus/pci$ cd 00
drix@lynx:/proc/bus/pci/00$ ls
00.0  02.0  1a.0  1c.0  1c.3  1c.7  1f.0  1f.3
01.0  16.0  1b.0  1c.1  1c.4  1d.0  1f.2
drix@lynx:/proc/bus/pci/00$ od -x 00.0
0000000 8086 0104 0006 2090 0009 0600 0000 0000
0000020 0000 0000 0000 0000 0000 0000 0000 0000
0000040 0000 0000 0000 0000 0000 0000 1028 04b0
0000060 0000 0000 00e0 0000 0000 0000 0000 0000
0000100
drix@lynx:/proc/bus/pci/00$ 
```

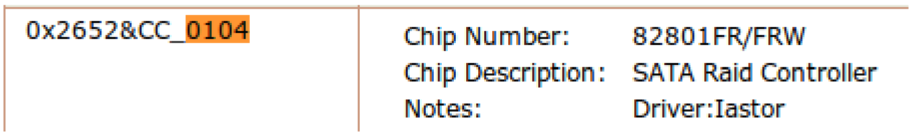
The divice vendor id is 8086 Intel



Returning 1 match for: **"8086"**
Sorted by: Vendor ID

Vendor Id	Vendor Name
0x8086	Intel Corporation

Device ID : 0104

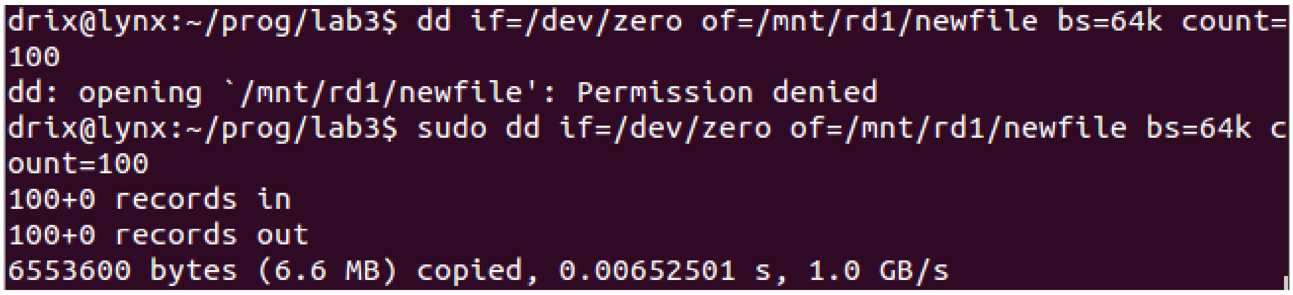


0x2652&CC_0104	Chip Number:	82801FR/FRW
	Chip Description:	SATA Raid Controller
	Notes:	Driver:Iastor

4.**[20 pts]Using Linux RAM Disk and Software RAID Using built-in features of your Fedora Linux install, construct a Solid-State RAM Disk (SSD) and Use software RAID to test. There are 16 modestly sized /dev/ram0 ... /dev/ram15 driver interfaces to built-in Linux RAM disk to use for this exercise. Please see** ECEN 5653 Notes on RAM disk and SW RAID **as well as the** Linux RAM Disk HOW-TO, Linux SW RAID HOW-TO, **MDADM** MDADM Software RAID, **and** Linux LVM HOW-TO **on this topic:**

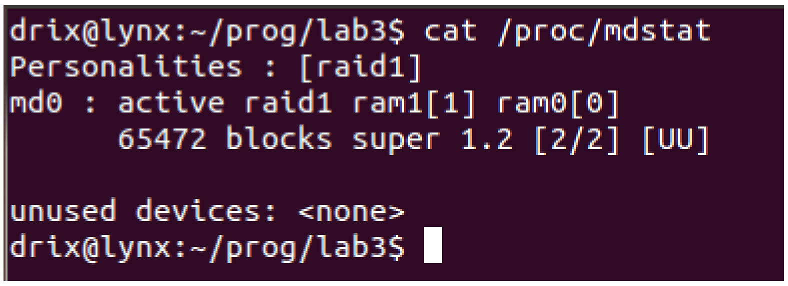
1. **[10 pts] Set up a simple 2 RAM disk RAID-1 (Mirror Pair) configuration using MDADM Software RAID and use "mkfs" to build a file system on it and mount it, and show that you can emulate a disk failure, but still access a test file on the mounted filesystem without error - please include your captured output from your Linux system to prove that your configuration works.**

Output of **dd** command



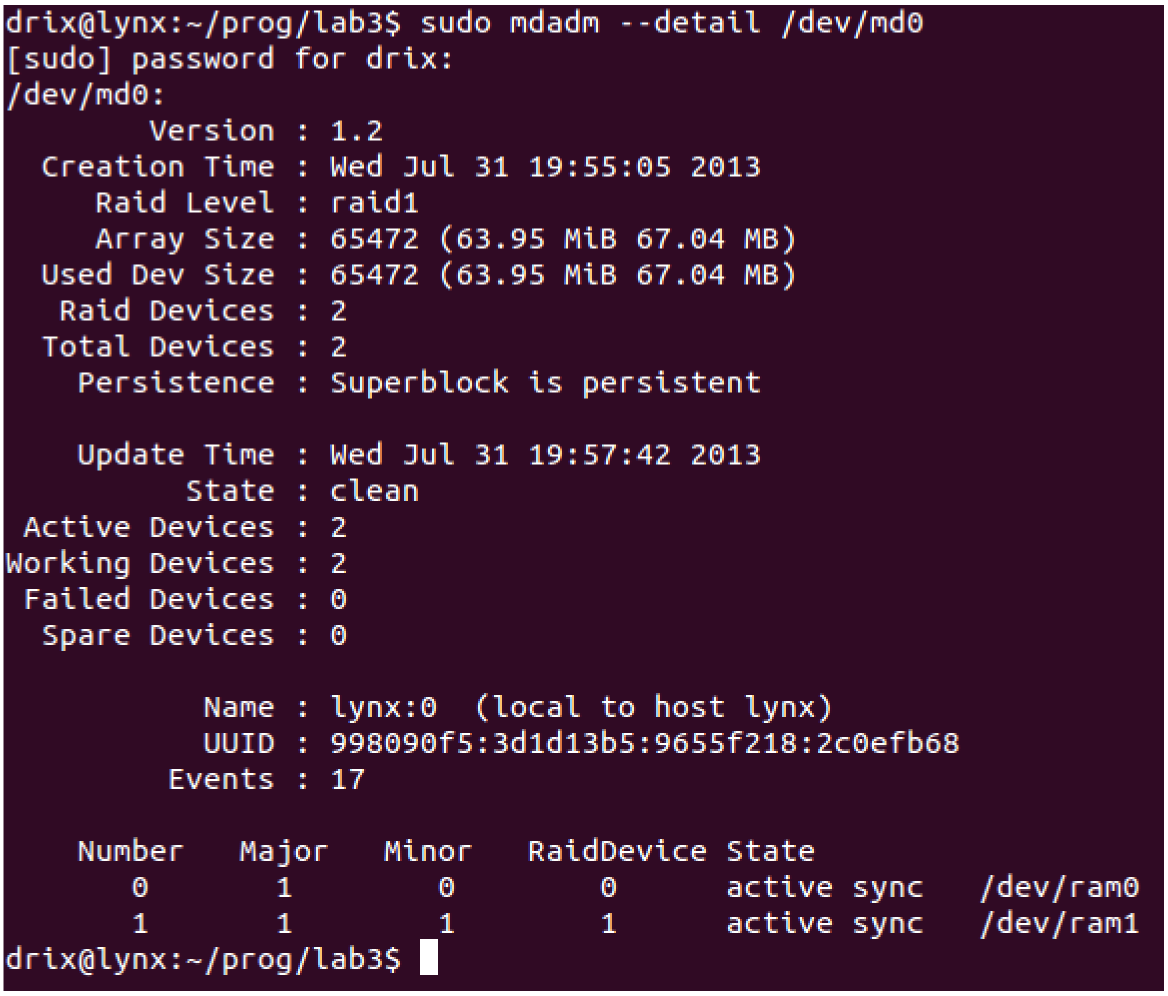
```
drix@lynx:~/prog/lab3$ dd if=/dev/zero of=/mnt/rd1/newfile bs=64k count=
100
dd: opening `/mnt/rd1/newfile': Permission denied
drix@lynx:~/prog/lab3$ sudo dd if=/dev/zero of=/mnt/rd1/newfile bs=64k c
ount=100
100+0 records in
100+0 records out
6553600 bytes (6.6 MB) copied, 0.00652501 s, 1.0 GB/s
```

Software RAID test:



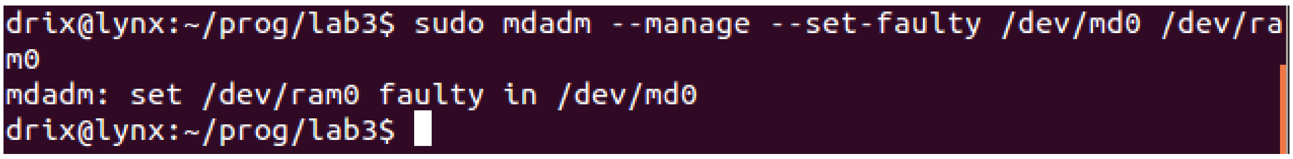
```
drix@lynx:~/prog/lab3$ cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 ram1[1] ram0[0]
      65472 blocks super 1.2 [2/2] [UU]

unused devices: <none>
drix@lynx:~/prog/lab3$ 
```

mdadm –detail command



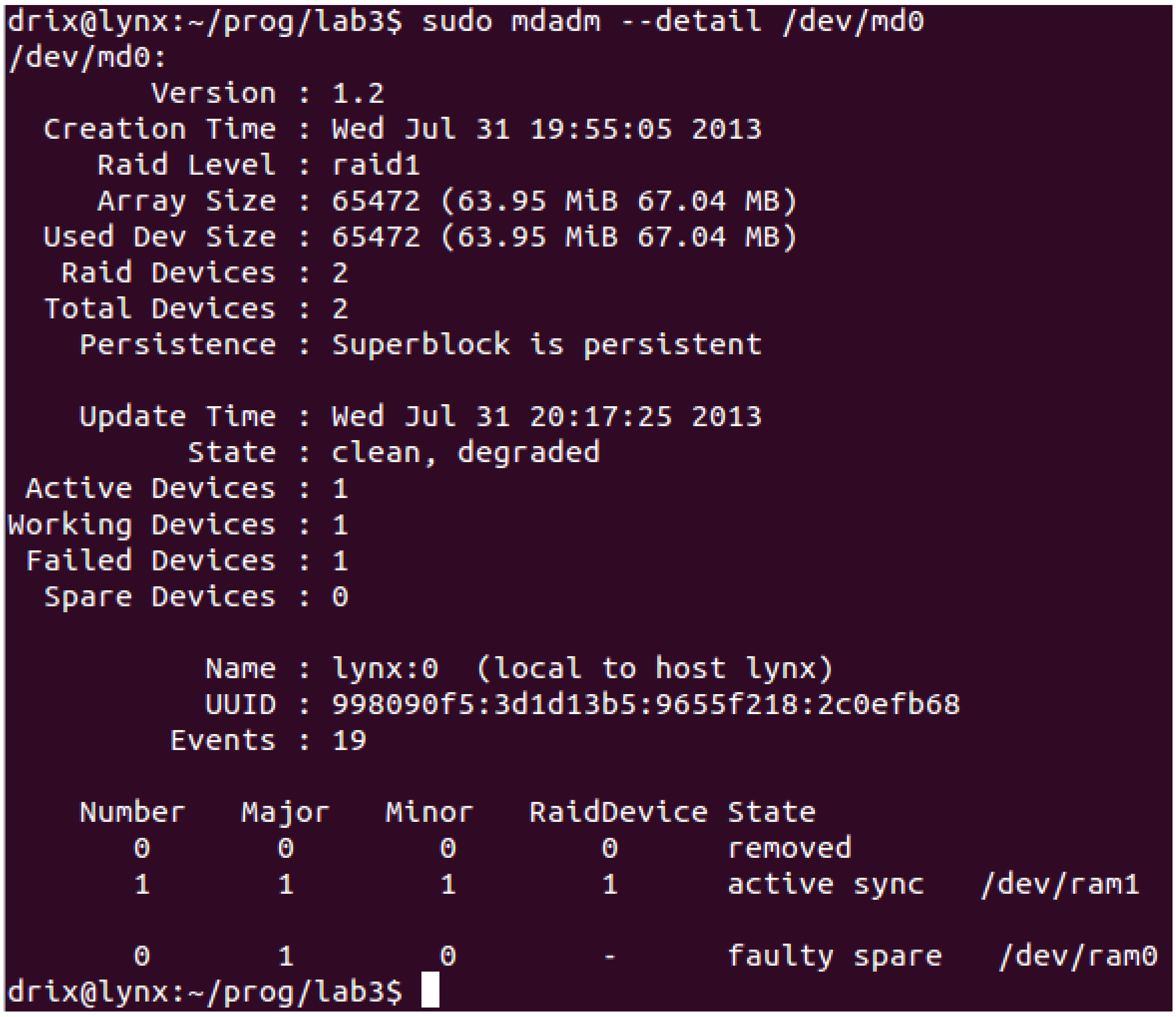
```
drix@lynx:~/prog/lab3$ sudo mdadm --detail /dev/md0
[sudo] password for drix:
/dev/md0:
        Version : 1.2
  Creation Time : Wed Jul 31 19:55:05 2013
     Raid Level : raid1
     Array Size : 65472 (63.95 MiB 67.04 MB)
  Used Dev Size : 65472 (63.95 MiB 67.04 MB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Wed Jul 31 19:57:42 2013
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : lynx:0  (local to host lynx)
           UUID : 998090f5:3d1d13b5:9655f218:2c0efb68
         Events : 17

    Number   Major   Minor   RaidDevice State
       0       1        0        0      active sync   /dev/ram0
       1       1        1        1      active sync   /dev/ram1
drix@lynx:~/prog/lab3$
```

Setting ram0 as faulty:



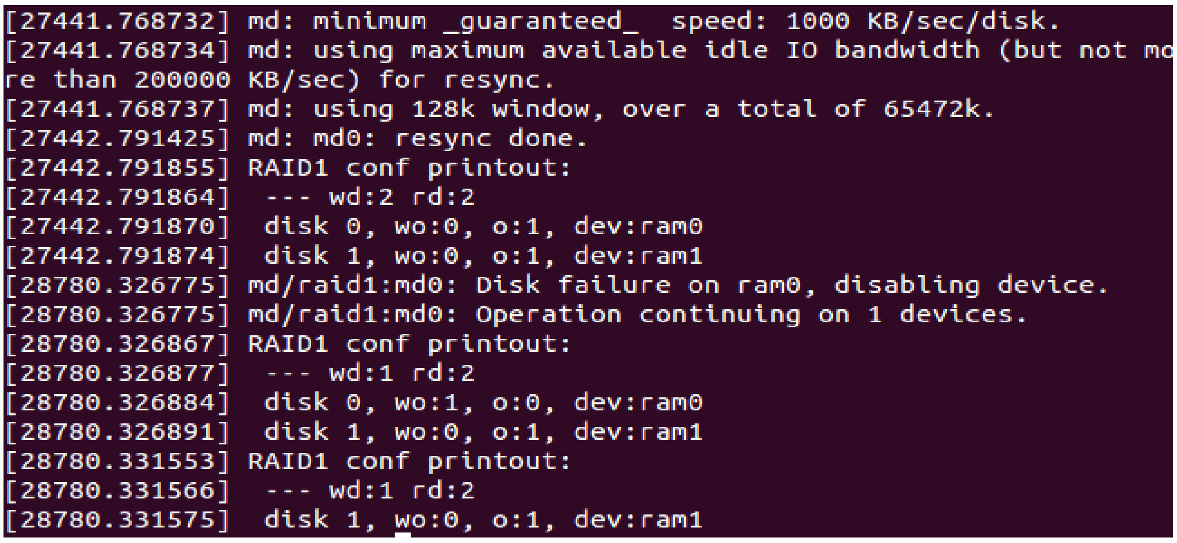
```
drix@lynx:~/prog/lab3$ sudo mdadm --manage --set-faulty /dev/md0 /dev/ra
m0
mdadm: set /dev/ram0 faulty in /dev/md0
drix@lynx:~/prog/lab3$
```

Cheking the status of md0 –detail



```
drix@lynx:~/prog/lab3$ sudo mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Wed Jul 31 19:55:05 2013
     Raid Level : raid1
     Array Size : 65472 (63.95 MiB 67.04 MB)
  Used Dev Size : 65472 (63.95 MiB 67.04 MB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Wed Jul 31 20:17:25 2013
          State : clean, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 1
  Spare Devices : 0

           Name : lynx:0  (local to host lynx)
           UUID : 998090f5:3d1d13b5:9655f218:2c0efb68
         Events : 19

    Number   Major   Minor   RaidDevice State
       0       0        0        0      removed
       1       1        1        1      active sync   /dev/ram1

       0       1        0        -      faulty spare   /dev/ram0
drix@lynx:~/prog/lab3$
```

dmesg causes resync of RAID so i get output:

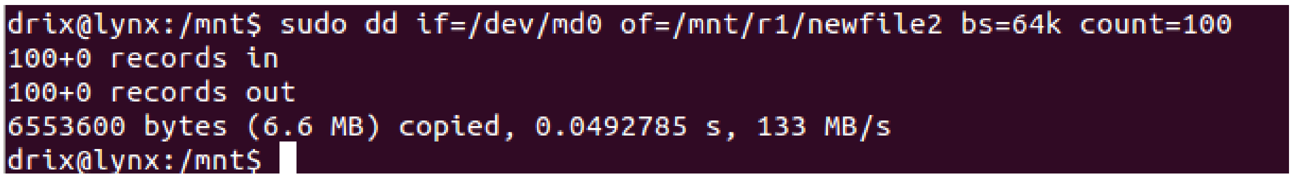


```
[27441.768732] md: minimum _guaranteed_  speed: 1000 KB/sec/disk.
[27441.768734] md: using maximum available idle IO bandwidth (but not mo
re than 200000 KB/sec) for resync.
[27441.768737] md: using 128k window, over a total of 65472k.
[27442.791425] md: md0: resync done.
[27442.791855] RAID1 conf printout:
[27442.791864]  --- wd:2 rd:2
[27442.791870]  disk 0, wo:0, o:1, dev:ram0
[27442.791874]  disk 1, wo:0, o:1, dev:ram1
[28780.326775] md/raid1:md0: Disk failure on ram0, disabling device.
[28780.326775] md/raid1:md0: Operation continuing on 1 devices.
[28780.326867] RAID1 conf printout:
[28780.326877]  --- wd:1 rd:2
[28780.326884]  disk 0, wo:1, o:0, dev:ram0
[28780.326891]  disk 1, wo:0, o:1, dev:ram1
[28780.331553] RAID1 conf printout:
[28780.331566]  --- wd:1 rd:2
[28780.331575]  disk 1, wo:0, o:1, dev:ram1
```

wo = working disk

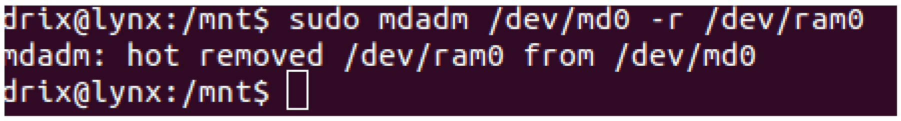
o = online disk

write test:



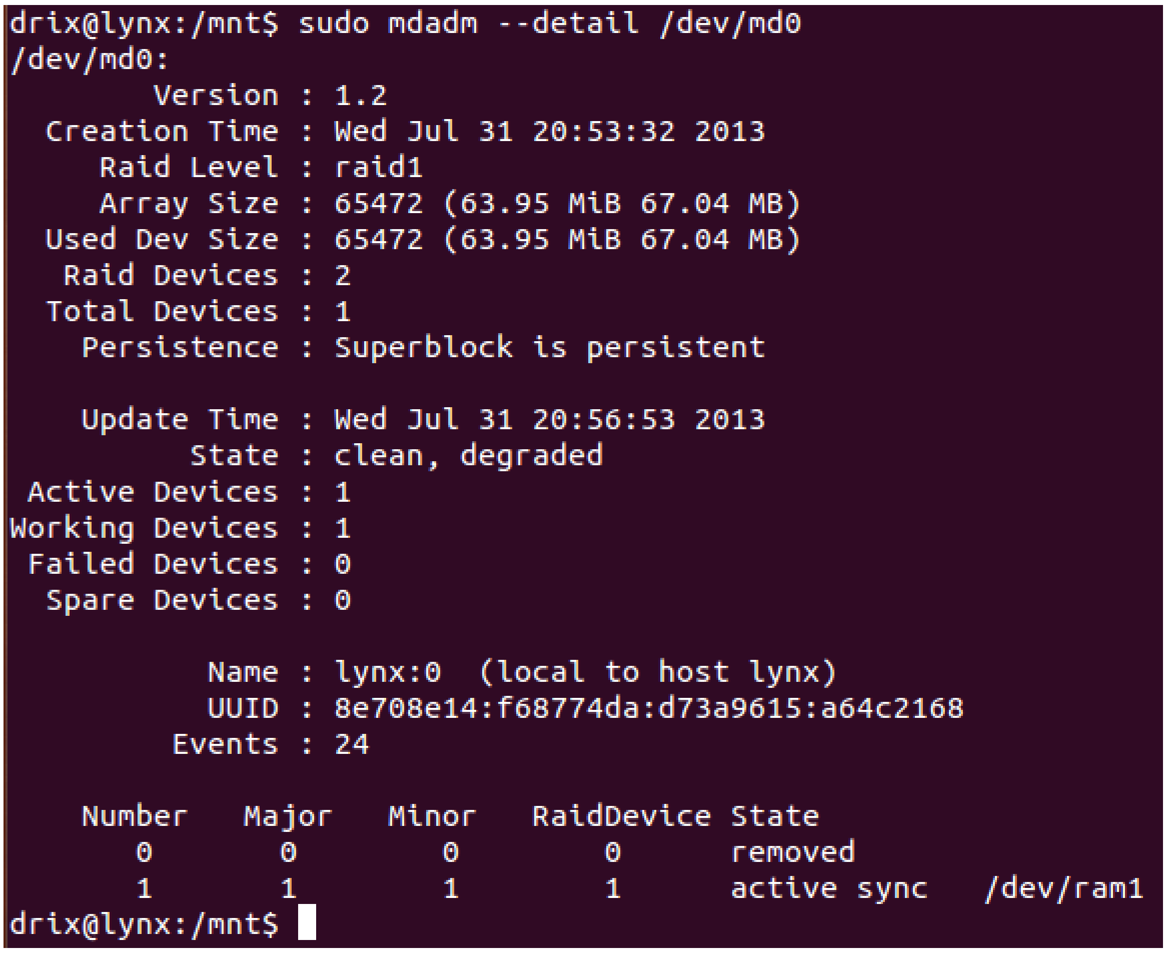
```
drix@lynx:/mnt$ sudo dd if=/dev/md0 of=/mnt/r1/newfile2 bs=64k count=100
100+0 records in
100+0 records out
6553600 bytes (6.6 MB) copied, 0.0492785 s, 133 MB/s
drix@lynx:/mnt$
```

hot removing md0:



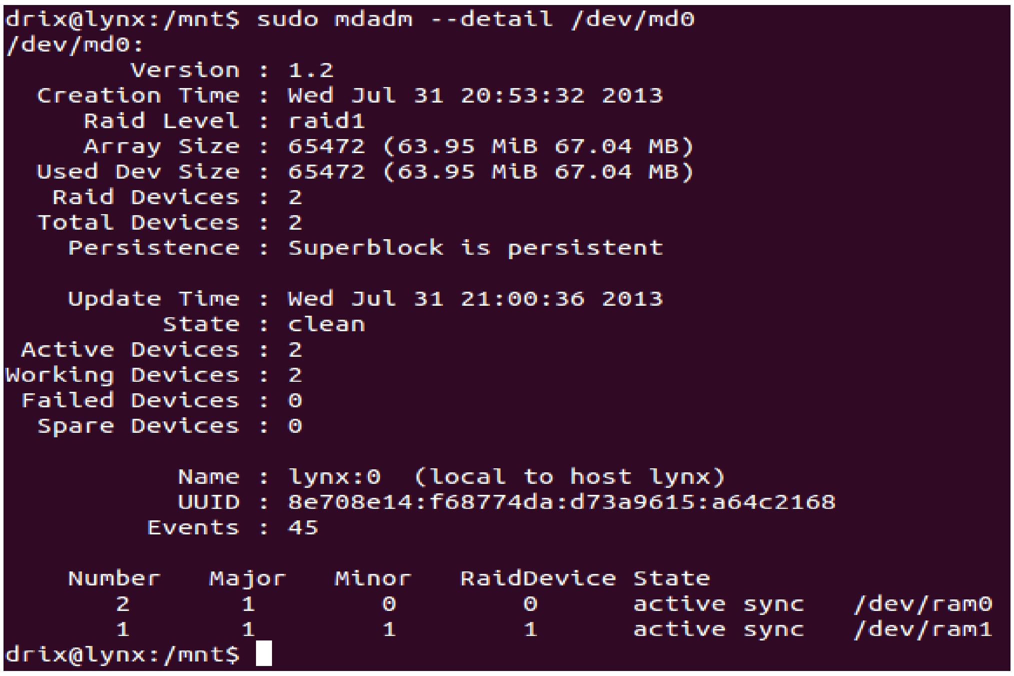
```
drix@lynx:/mnt$ sudo mdadm /dev/md0 -r /dev/ram0
mdadm: hot removed /dev/ram0 from /dev/md0
drix@lynx:/mnt$
```

detail reflecting the same:



```
drix@lynx:/mnt$ sudo mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Wed Jul 31 20:53:32 2013
     Raid Level : raid1
     Array Size : 65472 (63.95 MiB 67.04 MB)
  Used Dev Size : 65472 (63.95 MiB 67.04 MB)
   Raid Devices : 2
  Total Devices : 1
    Persistence : Superblock is persistent

    Update Time : Wed Jul 31 20:56:53 2013
          State : clean, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 0
  Spare Devices : 0

           Name : lynx:0  (local to host lynx)
           UUID : 8e708e14:f68774da:d73a9615:a64c2168
         Events : 24

    Number   Major   Minor   RaidDevice State
       0       0        0        0      removed
       1       1        1        1      active sync   /dev/ram1
drix@lynx:/mnt$
```

Adding back ram0:

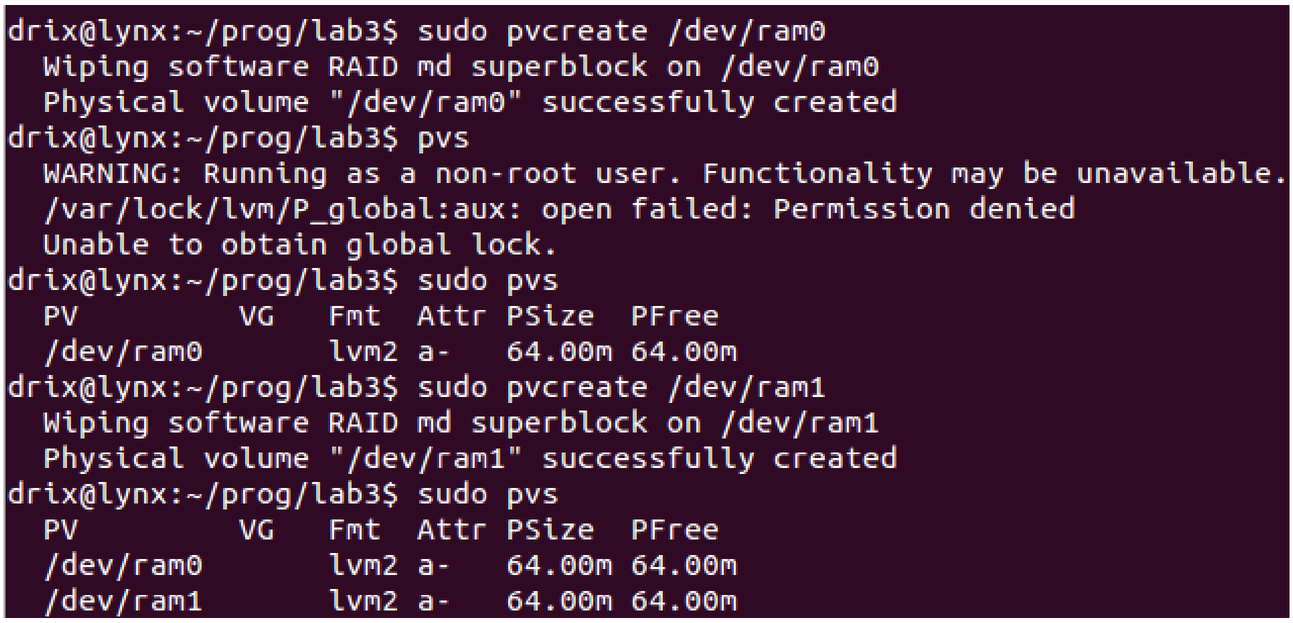


```
drix@lynx:/mnt$ sudo mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Wed Jul 31 20:53:32 2013
     Raid Level : raid1
     Array Size : 65472 (63.95 MiB 67.04 MB)
  Used Dev Size : 65472 (63.95 MiB 67.04 MB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Wed Jul 31 21:00:36 2013
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : lynx:0  (local to host lynx)
           UUID : 8e708e14:f68774da:d73a9615:a64c2168
         Events : 45

    Number   Major   Minor   RaidDevice State
       2       1        0        0      active sync   /dev/ram0
       1       1        1        1      active sync   /dev/ram1
drix@lynx:/mnt$
```

2.[10 pts] Set up a simple 8 RAM disk RAID-0 (Striped Concatenated Volume) configuraiton using LVM PV, VG, and LV create commands, use "mkfs" to build a file system on it, and then test by proving that you can copy files to this Solid-State RAID-0 volume - please include your captured output from your Linux system.

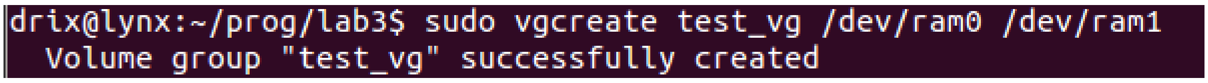
For this example i started with the hint file example :

Creating the two RAM disk from the example :



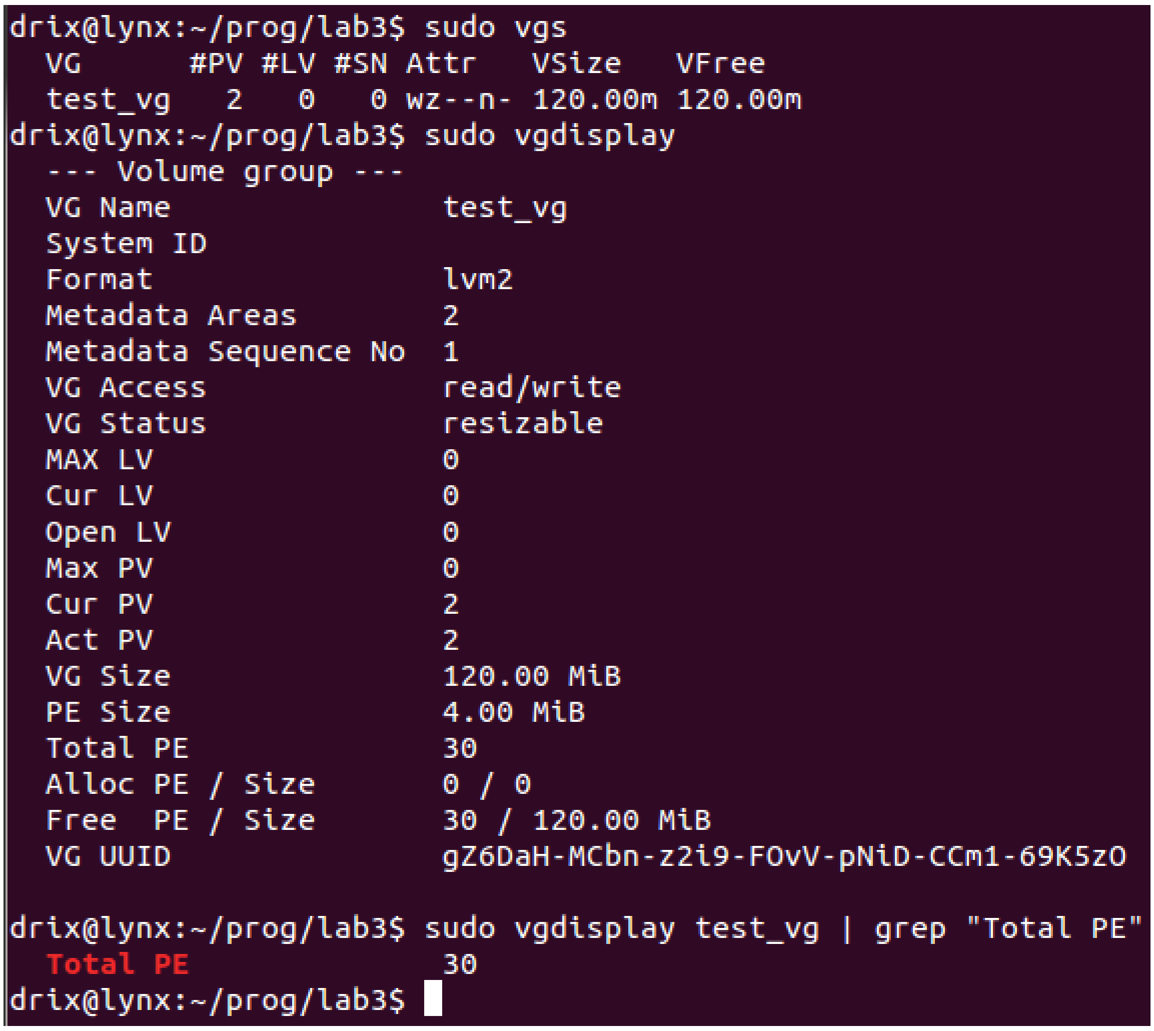
```
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram0
  Wiping software RAID md superblock on /dev/ram0
  Physical volume "/dev/ram0" successfully created
drix@lynx:~/prog/lab3$ pvs
  WARNING: Running as a non-root user. Functionality may be unavailable.
  /var/lock/lvm/P_global:aux: open failed: Permission denied
  Unable to obtain global lock.
drix@lynx:~/prog/lab3$ sudo pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/ram0       lvm2 a-   64.00m 64.00m
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram1
  Wiping software RAID md superblock on /dev/ram1
  Physical volume "/dev/ram1" successfully created
drix@lynx:~/prog/lab3$ sudo pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/ram0       lvm2 a-   64.00m 64.00m
  /dev/ram1       lvm2 a-   64.00m 64.00m
```

Adding the two to a virttual group:



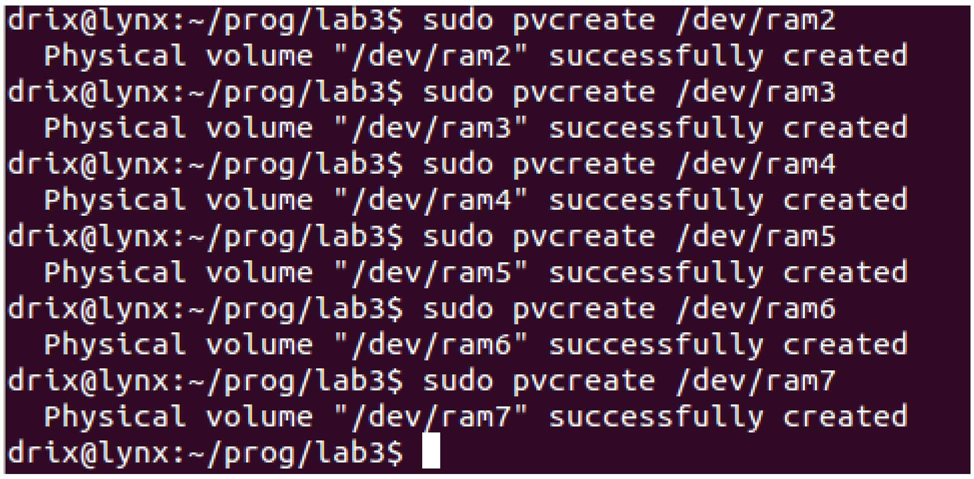
```
drix@lynx:~/prog/lab3$ sudo vgcreate test_vg /dev/ram0 /dev/ram1
  Volume group "test_vg" successfully created
```

Cheking with vgdisplay and getting the Total PE



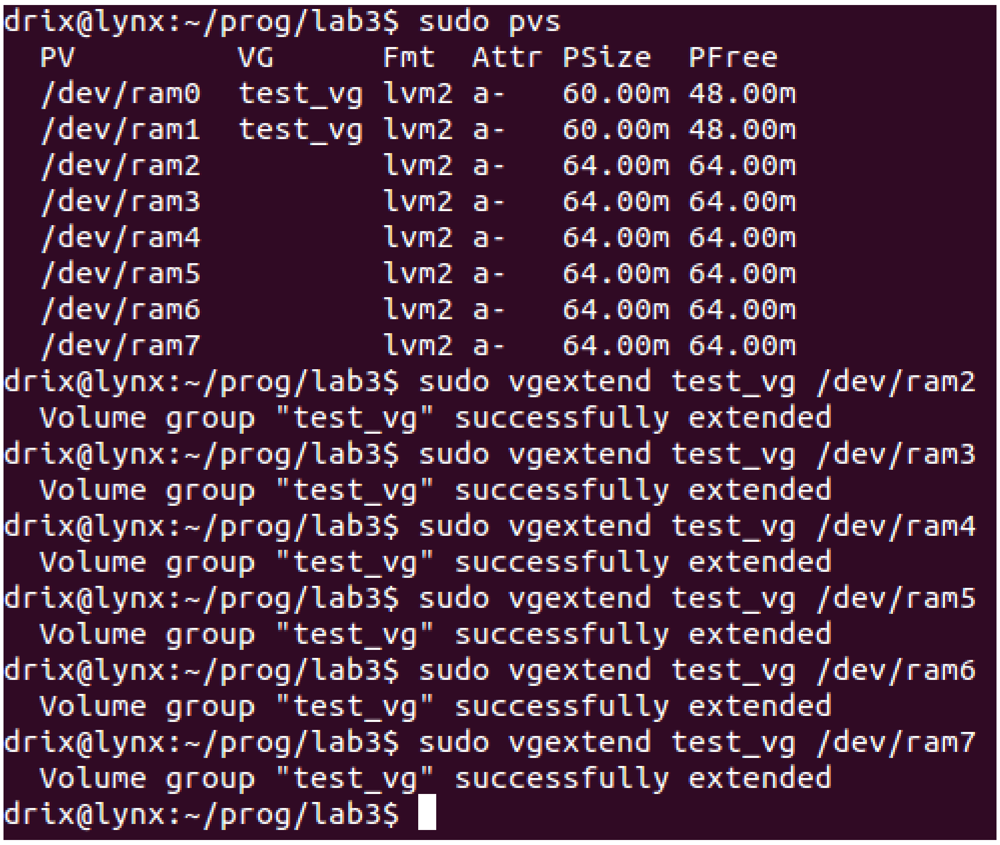
```
drix@lynx:~/prog/lab3$ sudo vgs
  VG      #PV #LV #SN Attr   VSize   VFree
  test_vg   2   0   0 wz--n- 120.00m 120.00m
drix@lynx:~/prog/lab3$ sudo vgdisplay
  --- Volume group ---
  VG Name               test_vg
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               120.00 MiB
  PE Size               4.00 MiB
  Total PE              30
  Alloc PE / Size       0 / 0
  Free  PE / Size       30 / 120.00 MiB
  VG UUID               gZ6DaH-MCbn-z2i9-FOvV-pNiD-CCm1-69K5zO

drix@lynx:~/prog/lab3$ sudo vgdisplay test_vg | grep "Total PE"
  Total PE              30
drix@lynx:~/prog/lab3$ 
```

creating the remaining partitons: ram 2 to ram 7



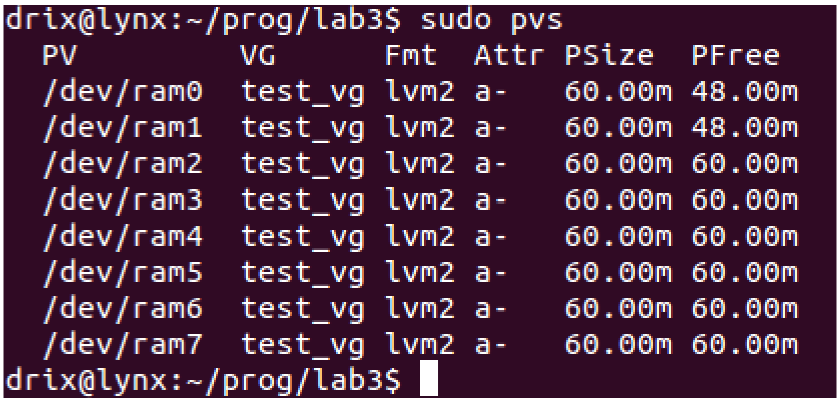
```
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram2
  Physical volume "/dev/ram2" successfully created
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram3
  Physical volume "/dev/ram3" successfully created
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram4
  Physical volume "/dev/ram4" successfully created
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram5
  Physical volume "/dev/ram5" successfully created
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram6
  Physical volume "/dev/ram6" successfully created
drix@lynx:~/prog/lab3$ sudo pvcreate /dev/ram7
  Physical volume "/dev/ram7" successfully created
drix@lynx:~/prog/lab3$ 
```

pvs output ater creation and adding them to the volume group vg_test



```
drix@lynx:~/prog/lab3$ sudo pvs
  PV         VG      Fmt  Attr PSize  PFree
  /dev/ram0  test_vg lvm2 a-   60.00m 48.00m
  /dev/ram1  test_vg lvm2 a-   60.00m 48.00m
  /dev/ram2          lvm2 a-   64.00m 64.00m
  /dev/ram3          lvm2 a-   64.00m 64.00m
  /dev/ram4          lvm2 a-   64.00m 64.00m
  /dev/ram5          lvm2 a-   64.00m 64.00m
  /dev/ram6          lvm2 a-   64.00m 64.00m
  /dev/ram7          lvm2 a-   64.00m 64.00m
drix@lynx:~/prog/lab3$ sudo vgextend test_vg /dev/ram2
  Volume group "test_vg" successfully extended
drix@lynx:~/prog/lab3$ sudo vgextend test_vg /dev/ram3
  Volume group "test_vg" successfully extended
drix@lynx:~/prog/lab3$ sudo vgextend test_vg /dev/ram4
  Volume group "test_vg" successfully extended
drix@lynx:~/prog/lab3$ sudo vgextend test_vg /dev/ram5
  Volume group "test_vg" successfully extended
drix@lynx:~/prog/lab3$ sudo vgextend test_vg /dev/ram6
  Volume group "test_vg" successfully extended
drix@lynx:~/prog/lab3$ sudo vgextend test_vg /dev/ram7
  Volume group "test_vg" successfully extended
drix@lynx:~/prog/lab3$ 
```

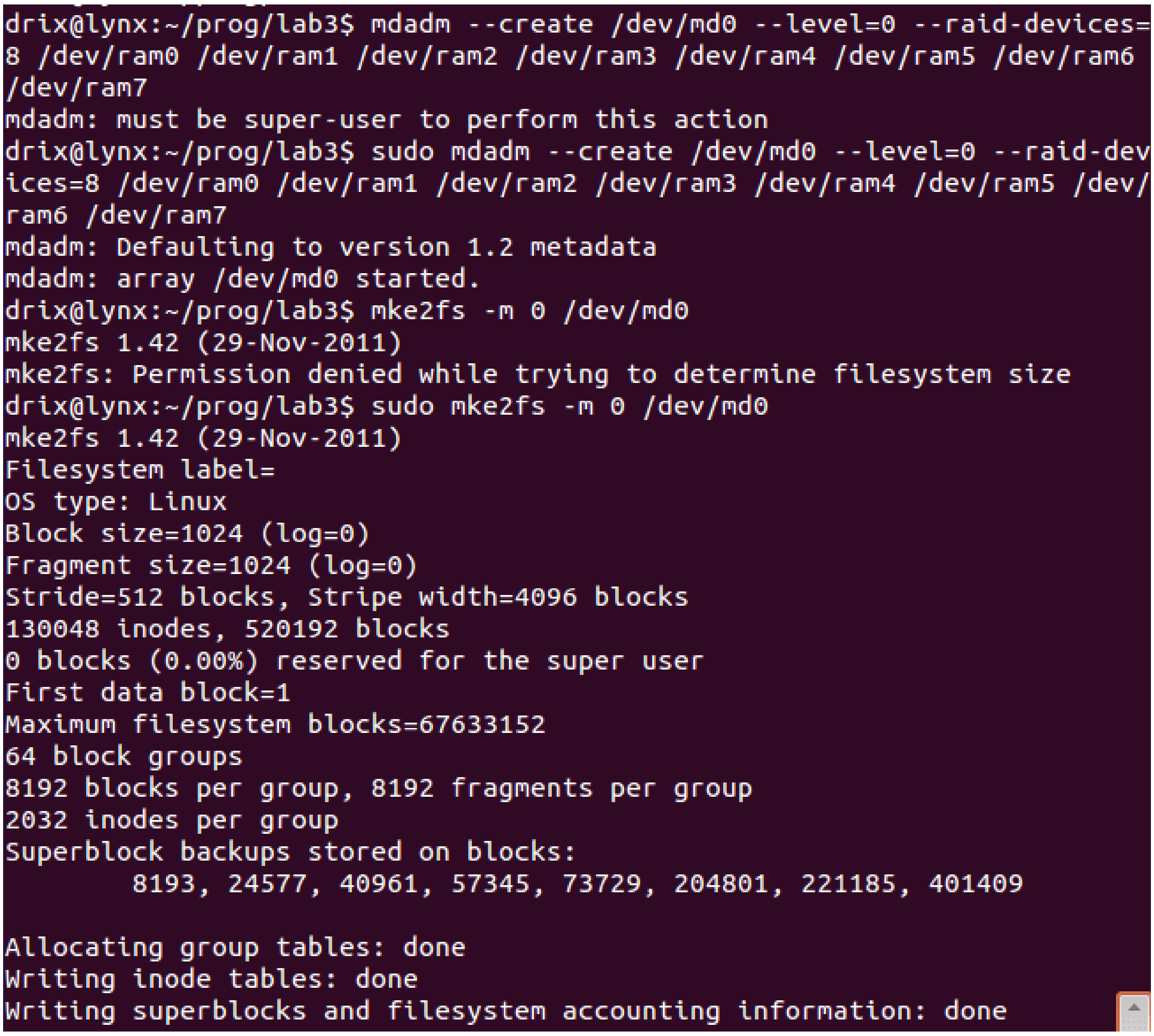
cheking if rams's got added to the test_vg volume group



```
drix@lynx:~/prog/lab3$ sudo pvs
  PV         VG      Fmt  Attr PSize  PFree
  /dev/ram0  test_vg lvm2 a-   60.00m 48.00m
  /dev/ram1  test_vg lvm2 a-   60.00m 48.00m
  /dev/ram2  test_vg lvm2 a-   60.00m 60.00m
  /dev/ram3  test_vg lvm2 a-   60.00m 60.00m
  /dev/ram4  test_vg lvm2 a-   60.00m 60.00m
  /dev/ram5  test_vg lvm2 a-   60.00m 60.00m
  /dev/ram6  test_vg lvm2 a-   60.00m 60.00m
  /dev/ram7  test_vg lvm2 a-   60.00m 60.00m
drix@lynx:~/prog/lab3$ 
```

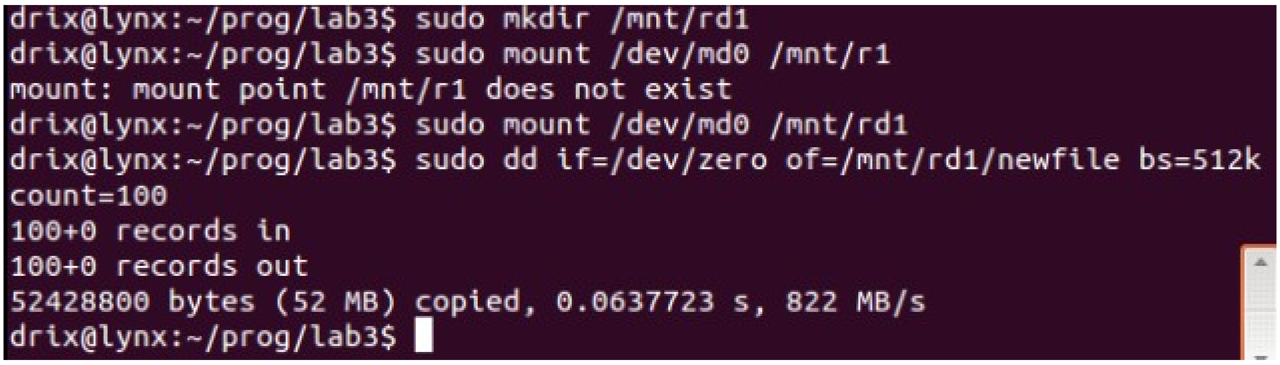
creating raid level 0 with mdadm command and to create a /dev/md0

then creating the filesystem with “mkfs”



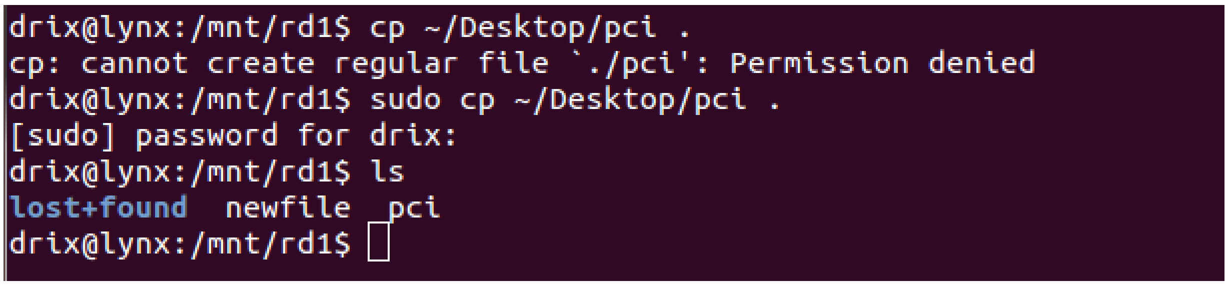
```
drix@lynx:~/prog/lab3$ mdadm --create /dev/md0 --level=0 --raid-devices=
8 /dev/ram0 /dev/ram1 /dev/ram2 /dev/ram3 /dev/ram4 /dev/ram5 /dev/ram6
/dev/ram7
mdadm: must be super-user to perform this action
drix@lynx:~/prog/lab3$ sudo mdadm --create /dev/md0 --level=0 --raid-dev
ices=8 /dev/ram0 /dev/ram1 /dev/ram2 /dev/ram3 /dev/ram4 /dev/ram5 /dev/
ram6 /dev/ram7
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
drix@lynx:~/prog/lab3$ mke2fs -m 0 /dev/md0
mke2fs 1.42 (29-Nov-2011)
mke2fs: Permission denied while trying to determine filesystem size
drix@lynx:~/prog/lab3$ sudo mke2fs -m 0 /dev/md0
mke2fs 1.42 (29-Nov-2011)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=512 blocks, Stripe width=4096 blocks
130048 inodes, 520192 blocks
0 blocks (0.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
64 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

mounting the file system and then peforming a write test



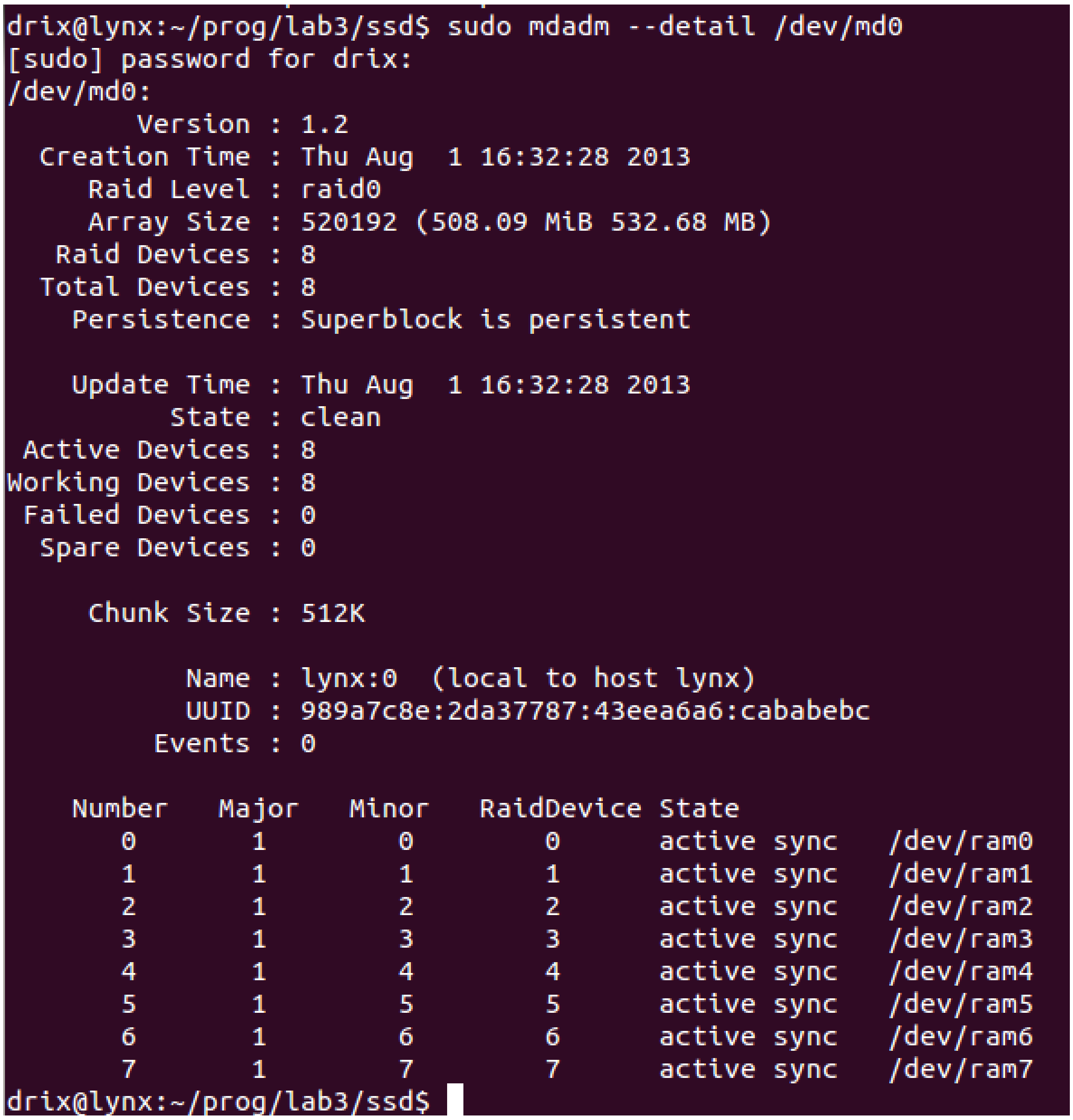
```
drix@lynx:~/prog/lab3$ sudo mkdir /mnt/rd1
drix@lynx:~/prog/lab3$ sudo mount /dev/md0 /mnt/r1
mount: mount point /mnt/r1 does not exist
drix@lynx:~/prog/lab3$ sudo mount /dev/md0 /mnt/rd1
drix@lynx:~/prog/lab3$ sudo dd if=/dev/zero of=/mnt/rd1/newfile bs=512k
count=100
100+0 records in
100+0 records out
52428800 bytes (52 MB) copied, 0.0637723 s, 822 MB/s
drix@lynx:~/prog/lab3$
```

copying a file into the newly created drive:



```
drix@lynx:/mnt/rd1$ cp ~/Desktop/pci .
cp: cannot create regular file `./pci': Permission denied
drix@lynx:/mnt/rd1$ sudo cp ~/Desktop/pci .
[sudo] password for drix:
drix@lynx:/mnt/rd1$ ls
lost+found  newfile  pci
drix@lynx:/mnt/rd1$
```

mdadm detailed outpu



```
drix@lynx:~/prog/lab3/ssd$ sudo mdadm --detail /dev/md0
[sudo] password for drix:
/dev/md0:
        Version : 1.2
  Creation Time : Thu Aug  1 16:32:28 2013
     Raid Level : raid0
     Array Size : 520192 (508.09 MiB 532.68 MB)
   Raid Devices : 8
  Total Devices : 8
    Persistence : Superblock is persistent

    Update Time : Thu Aug  1 16:32:28 2013
          State : clean
 Active Devices : 8
Working Devices : 8
 Failed Devices : 0
  Spare Devices : 0

     Chunk Size : 512K

           Name : lynx:0  (local to host lynx)
           UUID : 989a7c8e:2da37787:43eea6a6:cababebc
         Events : 0

    Number   Major   Minor   RaidDevice State
       0       1        0        0      active sync   /dev/ram0
       1       1        1        1      active sync   /dev/ram1
       2       1        2        2      active sync   /dev/ram2
       3       1        3        3      active sync   /dev/ram3
       4       1        4        4      active sync   /dev/ram4
       5       1        5        5      active sync   /dev/ram5
       6       1        6        6      active sync   /dev/ram6
       7       1        7        7      active sync   /dev/ram7
drix@lynx:~/prog/lab3/ssd$
```

5.**[25 pts]Basic Linux Driver (2 part question) using** SSD example driver**:**
Please make sure your Linux test setup has full Linux source for your distribution and/or that you download full Linux source and build and run this kernel for driver exercises. This can be done with the following commands on Fedora Core:

1.yum install kernel
2.yum install kernel-devel
3.yum install kernel-headers

For Ubuntu, please see the following helpful link - Kernel Compile

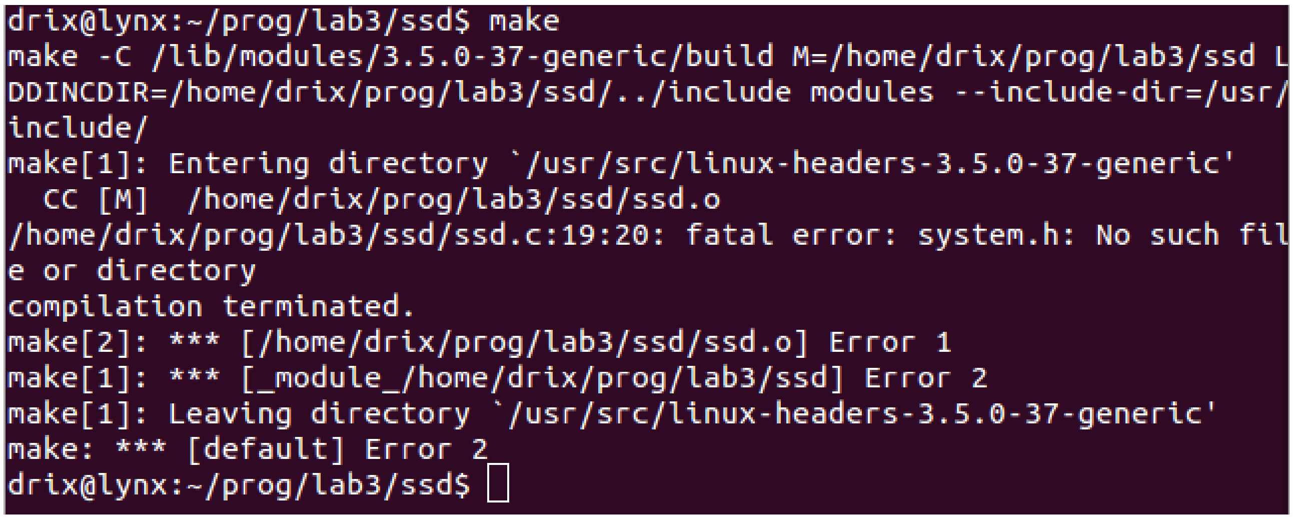
4.[10 pts] Download the SSD example driver, build it, install it with *./ssd_load* into your kernel and dump kernel printk messages using *dmesg* to verify that the driver has loaded properly. Review the source and describe how SSD sets up a RAM disk, how many disks you can create, and describe in your own words how well this emulates a real disk device.

5.[15 pts] Write some simple test application code to set up and use the SSD. Write your test code so that it proves SSD works using read, write, lseek, and ioctl. Show that you can read all F's after SSD is loaded, that you can write a 512 byte pattern to each of the seven SSDs, read it back, and then erase and read again using the ioctl. Verify that the dmesg trace matches what you expect from your sequence of calls into the entry points.

Unable to compile the program here. Ran into teouble with header files <asm/system.h>

I tried specifying the -I (include path argument) and point it to /usr/include/

in both the makefile and alsi tried it as a argument to make command but kept getting error.



```
drix@lynx:~/prog/lab3/ssd$ make
make -C /lib/modules/3.5.0-37-generic/build M=/home/drix/prog/lab3/ssd L
DDINCDIR=/home/drix/prog/lab3/ssd/../include modules --include-dir=/usr/
include/
make[1]: Entering directory `/usr/src/linux-headers-3.5.0-37-generic'
  CC [M]  /home/drix/prog/lab3/ssd/ssd.o
/home/drix/prog/lab3/ssd/ssd.c:19:20: fatal error: system.h: No such fil
e or directory
compilation terminated.
make[2]: *** [/home/drix/prog/lab3/ssd/ssd.o] Error 1
make[1]: *** [_module_/home/drix/prog/lab3/ssd] Error 2
make[1]: Leaving directory `/usr/src/linux-headers-3.5.0-37-generic'
make: *** [default] Error 2
drix@lynx:~/prog/lab3/ssd$ 
```

i shall keep trying to fix this .

6.**[30 pts]Extension of SSD RAM Disk Driver for RAID (3 part question) using** example-raid **code as a starting point and reference code along with the** SSD **character driver: This portion of the lab tests your basic understanding of Linux devices and RAID mappings onto them. You'll want to set up 7 1024 sector sized RAM disks for**

7. **this exercise using the SSD example, which will use a total of 7x512K=3.5MB of memory. Because the SSD character driver is not a block device, you'll need to modify the driver extensively to make sure that it reads, modifies, and writes only 512 byte LBAs in the kernel space using RAID-5 protocol despite user space test code access at a byte level. Make a new device entry point for SSD so that all SSD minor devices used in the RAID set now appear as on /dev/ssdr rather than the seven independent devices /dev/ssd0 ... /dev/ssd6 provided in the example, but allow the RAM disks to be acccessed either way.**

1. **[10 pts] Build a simple RAID-5 /dev/ssdr mapping for sectors which computes XOR parity for 4 sectors and stores parity striped in with the data blocks on 5 RAM disk drives so that sectors A,B,C,D,E,F,G,H are striped over the 5 drives as A,B,C,D,P(ABCD),E,F,G,P(EFGH). Provide your RAID-5 mapping code and compute how many RAID-5 sector size (512 byte) XOR encoder operations you can do per second writing to the SSD RAM disk. How many XOR check and recovery operations per second on reads? (Hint you should use the TSC or gettimeofday timing methods used in Lab 1). Note that you should do all reads and writes through your new abstracted /dev/ssdr RAID-5 device interface.**

2. **[10 pts] Build a RAID-6 mapping for sectors which computes P,Q parity and check codes for 5 sectors and stores parity striped in with the data blocks on 7 RAM disk drives as documented by James Plank, using the**Jerasure open source code **to implement your Reed-Solomon encoding and decoding. Provide your RAID-6 mapping code and compute how many RAID-6 sector size (512 byte) write encode operations you can do per second writing to the SSD RAM disk. Do not implement the recovery code, just the encode for writes. (Hint you should use the TSC or gettimeofday timing methods used in Lab 1). Again, you should do all reads and writes through your abstracted /dev/ssdr RAID-6 device interface.**

3. **[10 pts] Build an example which proves that you can corrupt data on one of your RAM disks and recover the correct data (use the erase ioctl). Likewise, show that you can fully fail one of the 5 disks and still recover all data. Prove that it does not matter which RAM disk you fail and that you can read your data with any one of the RAM disks not responding. You should add an ioctl which specifies which SSD to fail and while an SSD is in a failed state, it should not respond to read or write requests, but rather should return an error. Show that when you access the data through your /dev/ssdr RAID-5 device interface it always returns the right data despite erase corruption or failure of any ONE drive in the RAID set.**