CST8177 – Linux II

Disks, Filesystems

Todd Kelley
kelleyt@algonquincollege.com

Today's Topics

- sudo and PATH (environment)
- disks
- partitioning
- formatting file systems: mkfs command
- checking file system integrity: fsck command
- /etc/fstab
- mounting file systems: mount command
- unmounting file systems: umount command
- lsof and fuser

Executing a command (review)

- builtin command (part of the shell itself, so there's no notion of "where" the command is)
  - echo "Hello world"
  - exit 2 # inside a script, for example

- by absolute pathname (does not depend on PATH variable):
  - /bin/ls -l
  - /usr/sbin/useradd newuser
  - /usr/bin/sudo -i
  - "$HOME"/bin/myscript.sh # shell expands $HOME so this is really /home/username/bin/myscript.sh

Executing a command (cont'd)

- by relative pathname (does not depend on PATH variable, but DOES depend on your current directory – interactive shells only)
- You MUST NOT do any of these in a shell script
  - ./myscript.sh # script is in current directory
  - ../myprogram # script is in parent directory
  - ../../somedir/anotherscript.sh # two dirs up, then one directory down
  - bin/mycommand # assumes "bin" is a directory in the current directory

Executing a command (cont'd)

- using the PATH environment variable
  - ls -l
  - cp foo ../bar
  - rm ../bar/foo
- none of these commands will run unless they reside in a directory that is listed in the PATH environment variable
- Now that we are using root privileges, we need to be aware that root can have a different PATH than your non-root user

sudo and your environment

- sudo command # just run the command
  - you get 5 min by default to invoke sudo again without password
  - example$ sudo head /etc/shadow
- sudo -s # superuser shell with current env
- sudo -i # simulate root login (root's env)
- sudo -s leaves you in the same directory, and with the same PATH
- to take on root's environment including PATH:
  - sudo -i
  - or
  - sudo -s followed by su -

Disks and disk management

- partitioning
- LVM
- formatting file systems
- mounting file systems
- /etc/fstab

Overview of partitioning (8207 review)

- A partition is a section of disk forming a physical volume that contain a files ystem, or swap space, or be used as a component in LVM or RAID

- The **Master Boot Record** contains the **Disk Partition Table**, which can hold up to four entries due to the way in which the master boot record is structured
  - With certain specialty tools, you can create more than four partitions, but we'll stick to the MSDOS partition table format

- Each Disk Partition Table entry describes a partition by specifying its:
  - first cylinder
  - last cylinder
  - whether it is bootable
  - a partition type identifier.

Partitioning

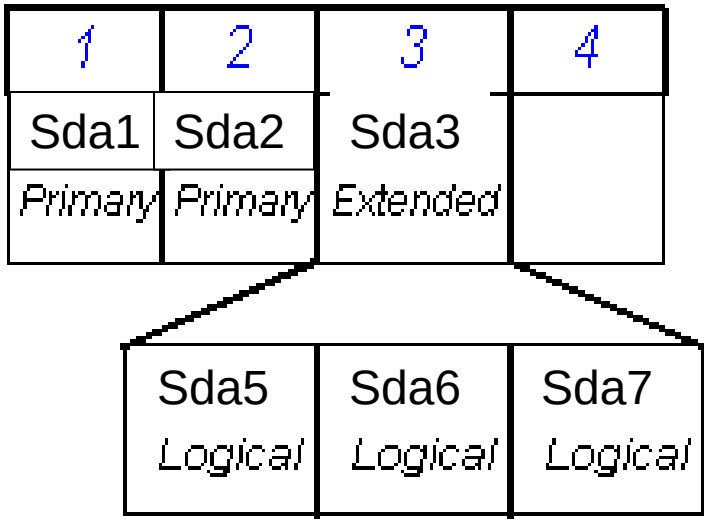
- We deal primarily with the MSDOS Partition Table type
- GPT partition tables getting common: GUID Partition Table
- Globally Unique IDentifier (but back to MSDOS Tables…)

- Up to four Primary Partitions are possible in a single table

- At most one of the four **Primary partitions** can be an **Extended Partition**

- **Logical Partitions** can be created inside an Extended Partition

Identifying Partitions



Naming partitions

- **sd*x*1 – sd*x*4**
  - Primary Partitions recorded in the partition table
- **sd*x*5 – sd*x*63**
  - Logical partitions

Note: You can have up to 4 primary partitions created in your system, while there can be only one extended partition.

Options for Partitioning

- DOS **`fdisk`** program
  - Very limited Linux support
- Linux **`fdisk`** program (we use this)
  - similar to DOS fdisk, but more features available
  - can only be used under Linux/UNIX
- parted can handle more partition table types (e.g. GPT)
- ***Disk Druid*** program
  - Part of the Fedora installation system
  - Cannot be run on its own
- ***gparted*** (*Fedora*, *Ubuntu*)
  - Gnome Partitioning Editor: GUI based partitioning
  - only runs from within Linux/UNIX

Linux `fdisk` command

- **`fdisk [options] device`**
  - command-line partition table manipulator for Linux
  - allows for viewing or modifying existing partition table and/or creating new partition(s) for a specified device
  - can set Partition Type for most of the common files systems in use today
  - fdisk –l /dev/sda

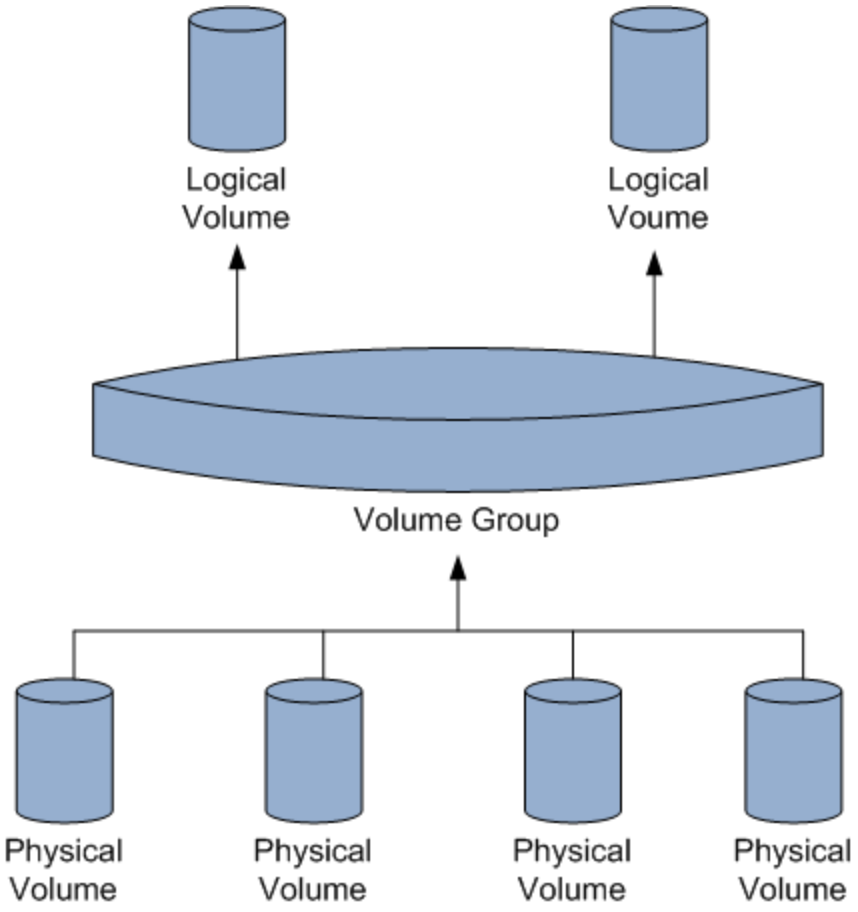
LVM basics

- Logical Volume Manager
- LVM tutorial:
  - http://www.howtoforge.com/linux_lvm
- disk partitions are physical volumes
- one or more physical volumes forms a volume group
- a volume group can be divided into logical volumes
- We create file systems on the logical volumes

Extents

- With LVM, we deal with space in logical and physical volumes in terms of "extents"
- Logical Volumes: LE or Logical Extents
- Physical Volumes: PE or Physical Extents
- Extents are the little pieces of space that can be managed: divided up into volumes, added to volumes

LVM Logical Volume Components



Adding disks and LVM

- Let's explore LVM by adding a disk and putting it under LVM control
- We'll create a file system on that logical volume
- Then we'll add yet another disk and grow that file system so it uses the added space
- physical volume commands /sbin/pv*
- volume group commands /sbin/vg*
- **logical volume commands /sbin/lv***
- **Examples**
  - `lvdisplay  # show logical volumes`
  - `pvdisplay  # show physical volumes`

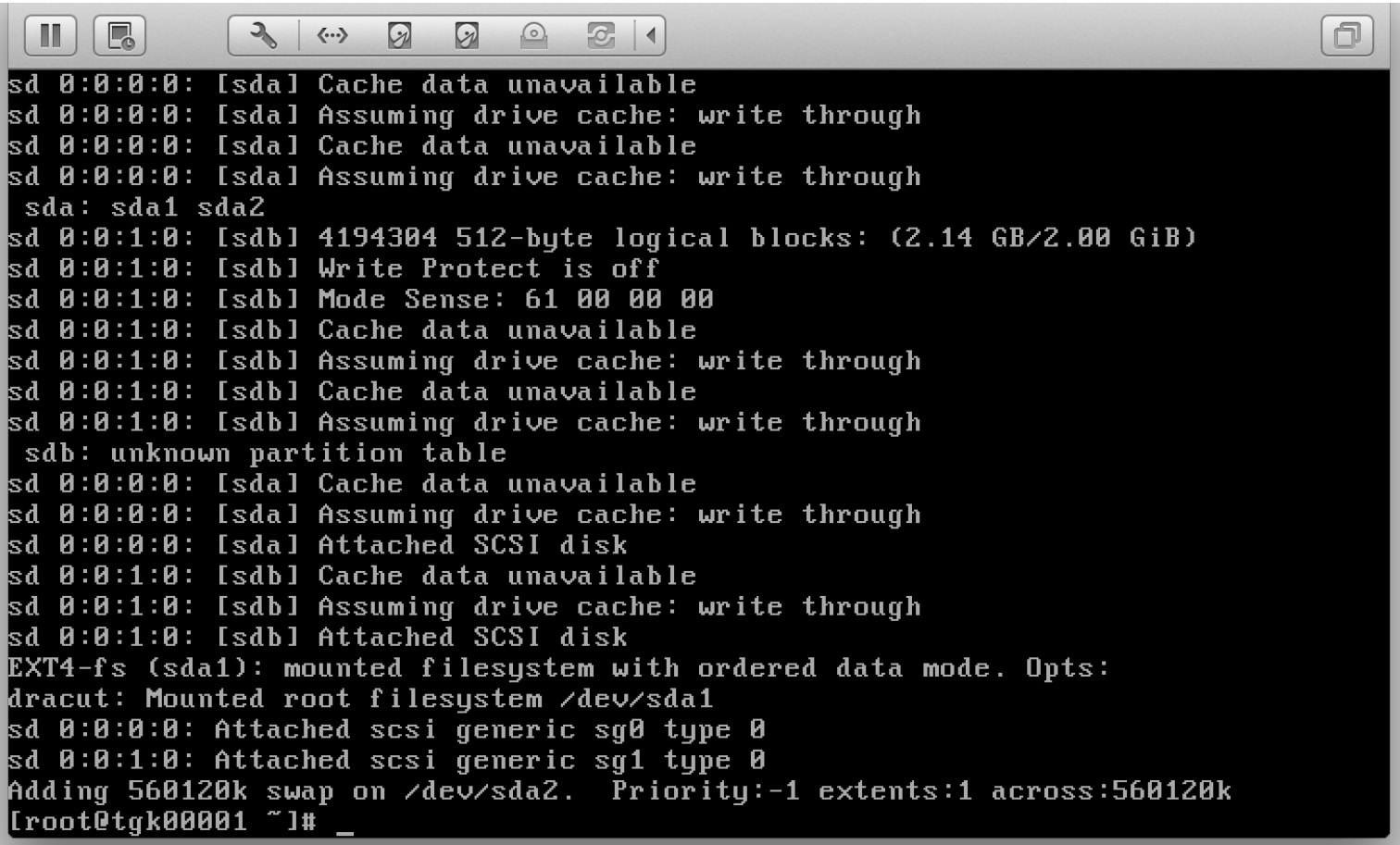
Add a disk

- power down machine (or virtual machine)
- add hard disk
- power up machine
- verify the new disk was detected (following slide)
- if the disk was brand new, it won't be partitioned (our example is this case)
- if the disk is being reused, be sure you can identify its partitions and you do not need the data

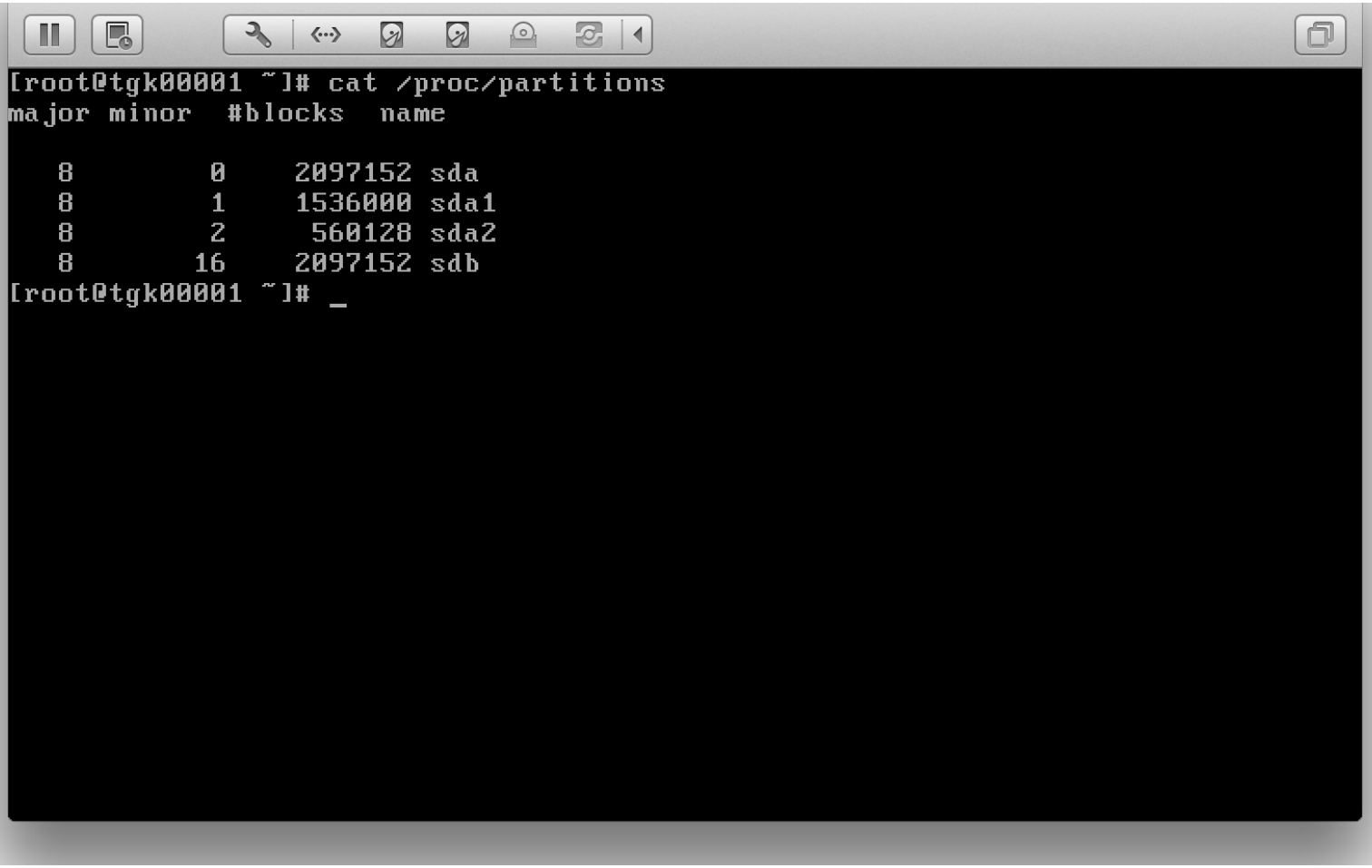
dmesg: kernel ring buffer

- http://teaching.idallen.com/cst8207/14w/notes/580_system_log_files.html
- kernel messages are kept in a ring buffer
- common way to access the boot messages, including device discovery
- dmesg
- example: look for disk discovery:
  - dmesg | grep sd
- (another way): look at disks/partitions that the kernel knows about:
  - cat /proc/partitions

Dmesg (sdb is the new disk)

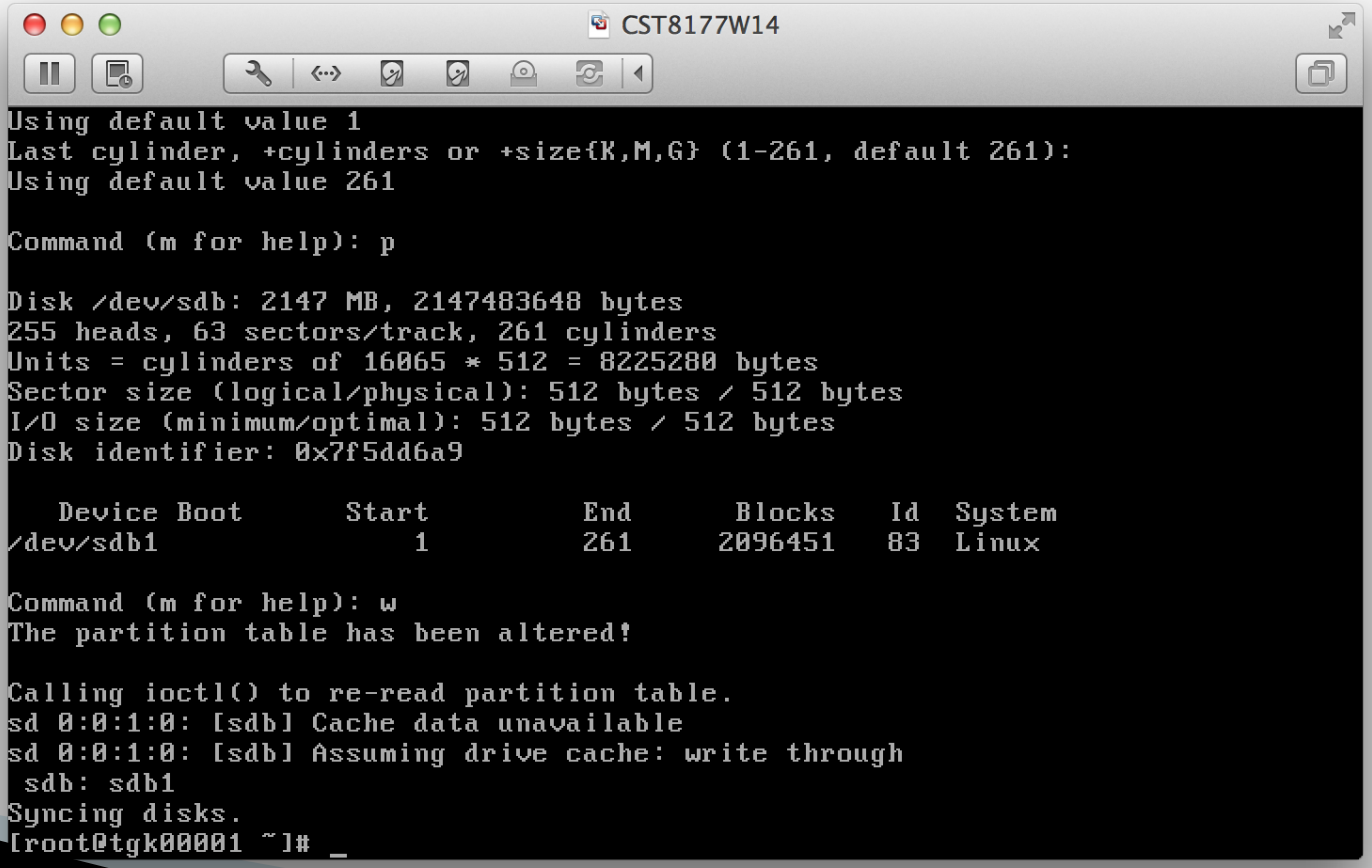


/proc/partitions (sdb is new)



Create partition on new disk

- use fdisk to partition the new disk
- we'll put the whole disk in one partition

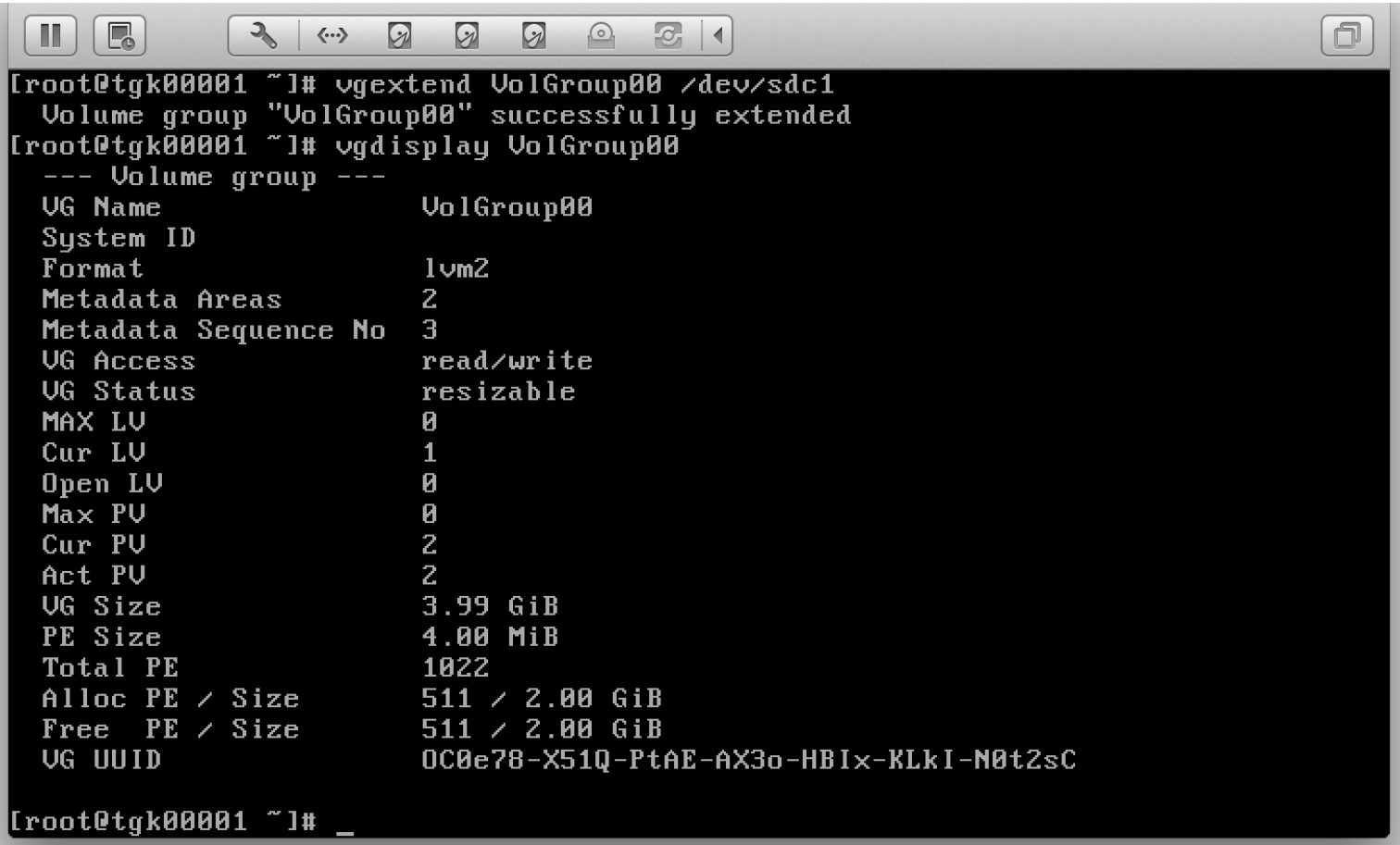


create the PV and VG and LV

- pvcreate /dev/sdb1
  - create the physical volume
- vgcreate VolGroup00 /dev/sdb1
  - add /dev/sdb1 physical volume to a new volume group called VolGroup00
- lvcreate -l 100%FREE -n LogVol00 VolGroup00
  - use 100% of the free space of VolGroup00 to create a new logical volume named LogVol00
  - creates /dev/VolGroup00/LogVol00 on which we can make a filesystem
- mkfs -t ext4 /dev/VolGroup00/LogVol00

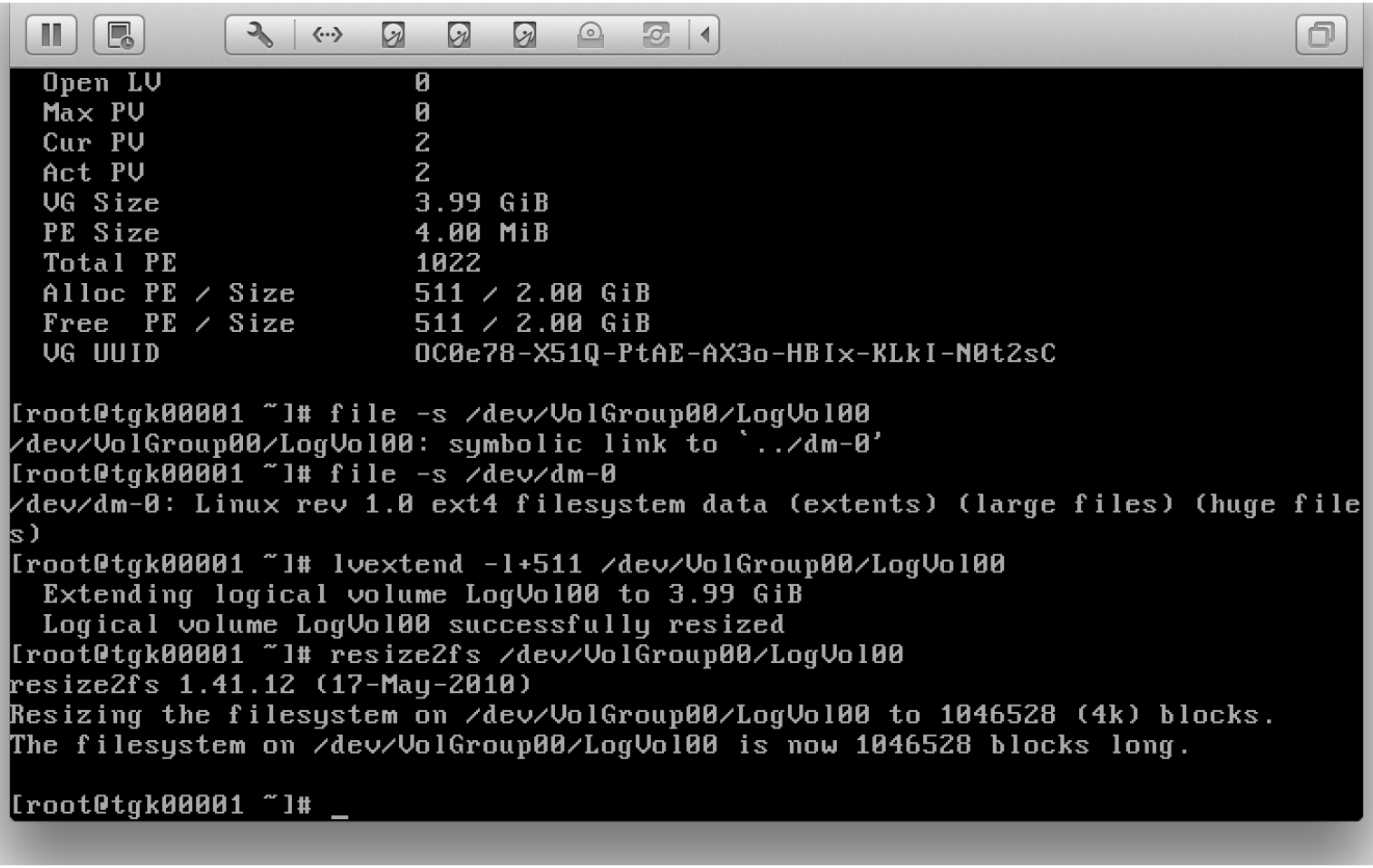
Growing a file system

- add yet another disk (say /dev/sdc)
- partition /dev/sdc to create /dev/sdc1
- Create the new physical volume
  - pvcreate /dev/sdc1
- Add this new physical volume to a volume group (in this case VolGroup00):
  - vgextend VolGroup00 /dev/sdc1
- See how many free extents (Free PE) are available in this volume group (VolGroup00)
  - vgdisplay VolGroup00


[root@tgk00001 ~]# vgextend VolGroup00 /dev/sdc1
Volume group "VolGroup00" successfully extended
[root@tgk00001 ~]# vgdisplay VolGroup00
--- Volume group ---
VG Name VolGroup00
System ID
Format lvm2
Metadata Areas 2
Metadata Sequence No 3
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 1
Open LV 0
Max PV 0
Cur PV 2
Act PV 2
VG Size 3.99 GiB
PE Size 4.00 MiB
Total PE 1022
Alloc PE / Size 511 / 2.00 GiB
Free PE / Size 511 / 2.00 GiB
VG UUID OC0e78-X51Q-PtAE-AX3o-HBIx-KLkI-N0t2sC
[root@tgk00001 ~]# _

Growing a file system (cont'd)

- Suppose the previous "vgdisplay" command showed that VolGroup00 had 511 free extents ("Free PE") and we use them all:
  - lvextend -l+511 /dev/VolGroup00/LogVol00
- Now LogVol00 is bigger, but the filesystem we created before is still the same size.
- Grow the filesystem (ext4) to fill the added space:
  - resize2fs /dev/VolGroup00/LogVol00
  - Now the filesystem is bigger, occupying the new disk space too


Open LV 0
Max PV 0
Cur PV 2
Act PV 2
VG Size 3.99 GiB
PE Size 4.00 MiB
Total PE 1022
Alloc PE / Size 511 / 2.00 GiB
Free PE / Size 511 / 2.00 GiB
VG UUID OC0e78-X51Q-PtAE-AX3o-HBIx-KLkI-N0t2sC
[root@tgk00001 ~]# file -s /dev/VolGroup00/LogVol00
/dev/VolGroup00/LogVol00: symbolic link to `../dm-0'
[root@tgk00001 ~]# file -s /dev/dm-0
/dev/dm-0: Linux rev 1.0 ext4 filesystem data (extents) (large files) (huge file
s)
[root@tgk00001 ~]# lvextend -l+511 /dev/VolGroup00/LogVol00
Extending logical volume LogVol00 to 3.99 GiB
Logical volume LogVol00 successfully resized
[root@tgk00001 ~]# resize2fs /dev/VolGroup00/LogVol00
resize2fs 1.41.12 (17-May-2010)
Resizing the filesystem on /dev/VolGroup00/LogVol00 to 1046528 (4k) blocks.
The filesystem on /dev/VolGroup00/LogVol00 is now 1046528 blocks long.
[root@tgk00001 ~]# _

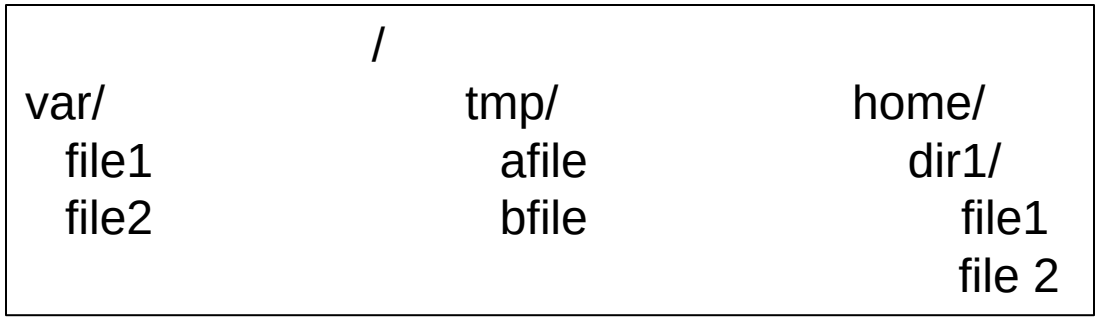
File systems (8207 review)

- http://teaching.idallen.com/cst8207/14w/notes/720_partitions_and_file_systems.html

Linux/Unix mounting

- no drive letters!

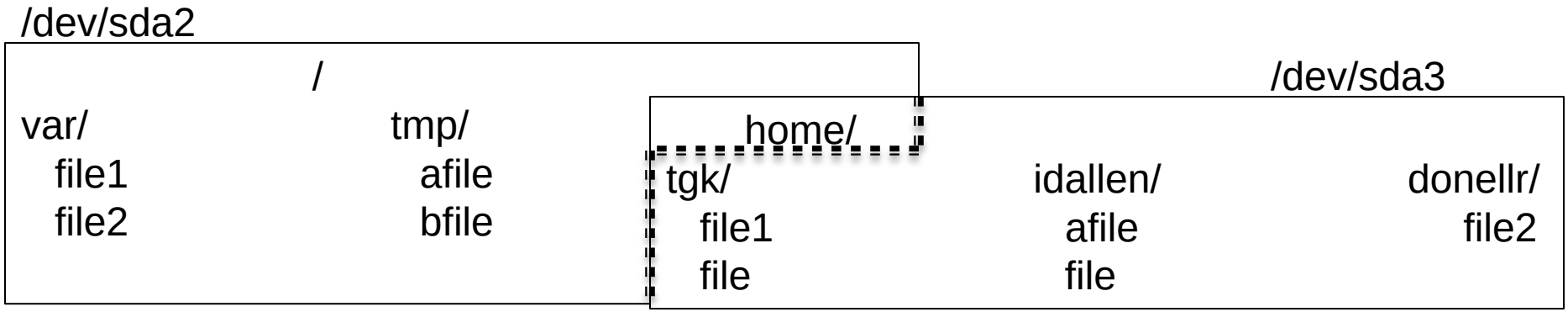
/dev/sda2



/dev/sda3

Linux/Unix mounting

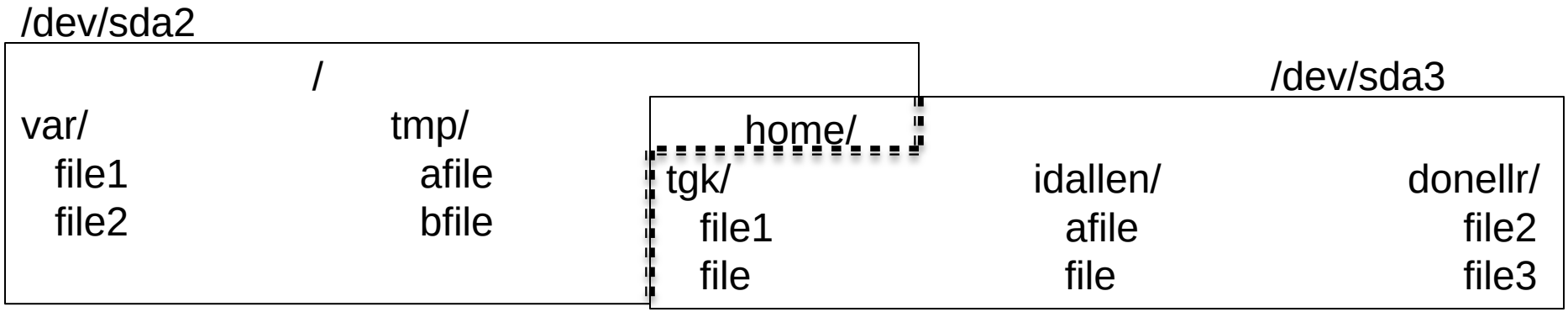
- mount /dev/sda3 /home



- the /home directory name still on /dev/sda2
- the contents of /home are on /dev/sda3
- the previous contents of /home are hidden

Linux/Unix mounting

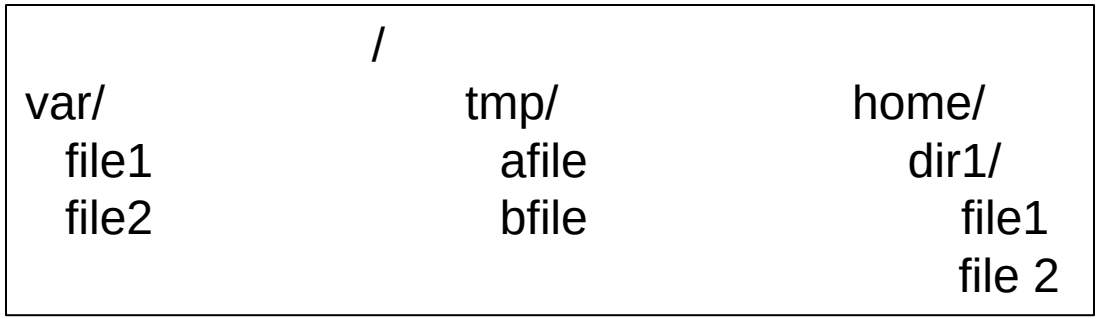
- touch /home/donellr/file3



Linux/Unix mounting

- umount /dev/sda3

/dev/sda2



/dev/sda3

/etc/fstab

- `fsck`, `mount`, and `umount` use this file
- man 5 fstab
- note that records for swap space appear in `/etc/fstab`, although swap space is not a filesystem (files are not stored in swap space)
- first field: device name
- second field: mount point
- third field: type
- fourth field: mount options
- fifth field: backup related (dump program)
- sixth field: file system check order

/etc/fstab commands

- `mount -a`
  - issued as part of the boot process
  - all file systems listed in `/etc/fstab` will be mounted accordingly (except those with "`noauto`" option or "`ignore`" file system type)
- `mount <mount point>`
  - mount will consult `/etc/fstab` to find the device and options for that mount point, and mount it
- `mount <device>`
  - mount will consult `/etc/fstab` to find the mount point and options for that mount point, and mount it

/etc/fstab: device name

- device name, the first field, names the block special device (e.g. /dev/sda1) on which the file system resides
- the first field can also be expressed in terms of LABEL or UUID (e.g. LABEL=root) (e.g. see the /etc/fstab on our CentOS 6.5 machines)
  - `blkid` command prints the UUIDs of the system's block devices
  - `e2label` command prints/sets file system labels

/etc/fstab: mount point

- The mount point is the directory on which the file system should be mounted
- swap is not a file system but is still controlled by /etc/fstab, so the mount point is `none`

/etc/fstab: file system type

- `ext4` is the file system type we use often
- `/proc/filesystems` contains the list of file systems supported by the currently running kernel
- `swap` for swap space
- `ignore` for an unused filesystem
- `none` for bind mounts

/etc/fstab: mount options

- fourth field in `/etc/fstab`
- expressed as a comma-separated list
- different file systems support different options (see `man 8 mount`)
- `defaults`: a set of default options
- example options common to all file system types:
  - noauto: do not mount when "mount -a" called
  - user: allow a user to mount
  - owner: allow device owner to mount

/etc/fstab: options (cont'd)

- mount options
  - on CentOS 6.5, "defaults" means
    - rw: read and write
    - dev: interpret device nodes
    - suid: setuid and setgid bits take effect
    - exec: permit execution of binaries
    - auto: mount automatically due to "mount -a"
    - nouser: regular users cannot mount
    - async: file I/O done asynchronously
    - relatime: update access times a certain way
- other options:
  - these are for quota utilities to see rather than mount
    - usrquota
    - grpquota

/etc/fstab: dump

- the dump program uses this field to determine which file systems should be backed up by the dump command
- the dump program can back up an entire file system to tape, for example
- dump supports incremental backups
- when restoring, it can provide an index of what's in the file system, do partial restores, etc
- we don't use dump in this course

/etc/fstab: fsck order

- the sixth and last field is used to determine the order in which file system checks are done at boot
- root file system: 1
- other file systems: 2
- no fsck: 0

/etc/mtab and /proc/mounts

- `/etc/mtab` is used by `mount` and `umount` to keep track of what is currently mounted
- `mount` command (no args) prints this file
- `/proc/mounts` is the kernel's list of what's mounted, and might be more up-to-date than `/etc/mtab`

Adding a disk

- # migrating the /usr directory to be a separate partition on new disk
- shut down machine
- connect new disk to machine
- power on machine
- partition new disk (fdisk command)
- make filesystem in new partition (mkfs command)
- single user mode (shutdown command)
- ensure target directory is backed up
- move the target directory out of way (/usr to /usr1) (mv command)
- create the mount point (to replace dir we just moved, same name)
- mount new filesystem (mount command)
- /usr1/bin/rsync –aHv /usr1/. /usr (notice where rsync is!)
- add a record for the new filesystem /etc/fstab
- exit, to return to runlevel 3
- remove /usr1 (content should be backed up)

device busy

- when trying to unmount a filesystem, you might get an error:

```
umount: /dirname: device is busy
```

- probably some process is using the filesystem (it's busy -- make sure you're not in that directory!)
- `lsof /mountpoint` # list open files in the filesystem mounted on /mountpoint

```
lsof +D /directory
```

this will show you what processes are using the directory or (+D) any directory under it

lsof and fuser

- Note the difference between a mountpoint and a directory
  - mountpoint: both of these commands will apply to the entire filesystem mounted there
  - directory: both of these commands will apply to just that directory, not recursively every subdirectory underneath it

- summary of lsof:
  - http://www.thegeekstuff.com/2012/08/lsof-command-examples/

- fuser: similar in purpose to lsof
- examples:
  - `fuser /mountpoint` # all processes using the filesystem mounted at /mountpoint
  - `fuser /home/dir` # all processes using the directory dir
- summary of fuser:
  - http://www.thegeekstuff.com/2012/02/linux-fuser-command/