Southern Connecticut State University

SCSU

CSC424 System Administration

Instructor: Dr. Hao Wu

Week 3 File System

Filesystem

- Two definitions:
  - the entire hierarchy of directories (also referred to as the directory tree) that is used to organize files on a computer system.

  - refers to type of filesystem: how the storage of data is organized on a computer disk or on a partition on a hard disk. Each type of filesystem has its own set of rules for controlling the allocation of disk space to files and for associating data about each file with that file.

Linux Filesystem

- "On a UNIX system, everything is a file; if something is not a file, it is a process."

Linux Filesystem Structure

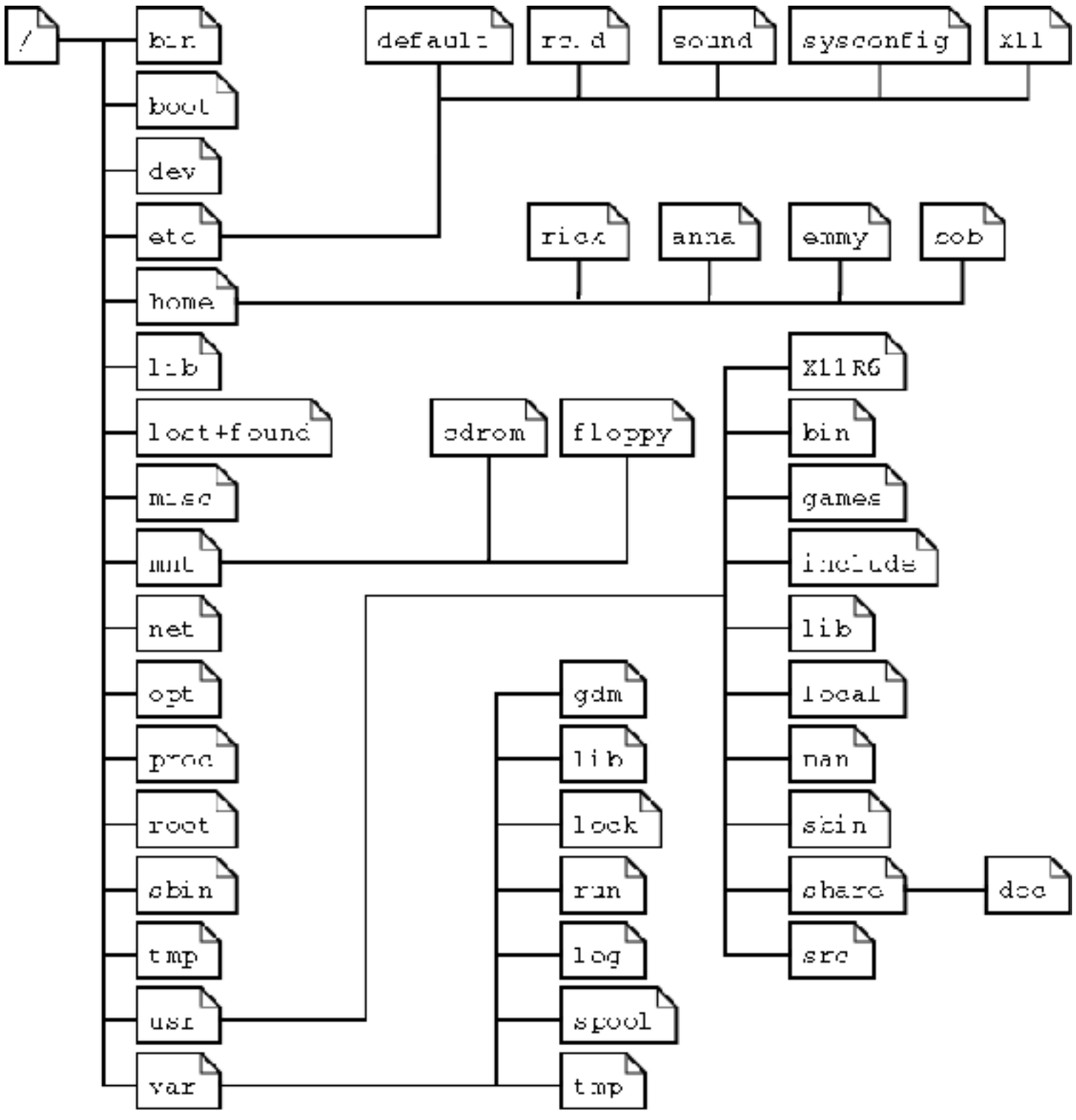
- Each filesystem system contains:
  - ***control block***: holds information about the filesystem
  - ***inodes***: contain information about individual files
  - ***data blocks***: contain the information stored in the individual files

- In Linux, users and kernel see filesystem in different ways
  - User
    - Filesystem appears as a hierarchical arrangement of directories that contain files and other directories.
    - Directories and files are identified by their names.
  - Kernel
    - Filesystem is flat, not have a hierarchical structure
    - no differences between directories and a file
    - identify files by name.

Linux Filesystem Structure: kernel's view

- inode: an entry in a list of inodes, it contains
  - inode number (a unique identification number)
  - the owner and group associated with the file
  - the file type
  - the file's permission list
  - the file creation, access and modification times
  - the size of the file
  - the disk address

Linux Filesystem Structure: user's view



Linux Filesystem Structure: user's view

- The tree of the file system starts at the trunk or *slash*, indicated by a forward slash (/). This directory, containing all underlying directories and files, is also called the *root directory* or "the root" of the file system.

```
[root@localhost ~]# ls /
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

Linux Filesystem Structure: subdirectories of the root

Pathname	Contents
/bin	Core operating system commands
/boot	Boot loader, kernel, and files needed by the kernel
/compat	On FreeBSD, files and libraries for Linux binary compatibility
/dev	Device entries for disks, printers, pseudo-terminals, etc.
/etc	Critical startup and configuration files
/home	Default home directories for users
/lib	Libraries, shared libraries, and commands used by **/bin** and **/sbin**
/media	Mount points for filesystems on removable media
/mnt	Temporary mount points, mounts for removable media
/opt	Optional software packages (rarely used, for compatibility)
/proc	Information about all running processes
/root	Home directory of the superuser (sometimes just /)
/run	Rendezvous points for running programs (PIDs, sockets, etc.)
/sbin	Core operating system commands[a]
/srv	Files held for distribution through web or other servers
/sys	A plethora of different kernel interfaces (Linux)
/tmp	Temporary files that may disappear between reboots
/usr	Hierarchy of secondary files and commands
/usr/bin	Most commands and executable files
/usr/include	Header files for compiling C programs
/usr/lib	Libraries; also, support files for standard programs
/usr/local	Local software or configuration data; mirrors **/usr**
/usr/sbin	Less essential commands for administration and repair
/usr/share	Items that might be common to multiple systems
/usr/share/man	On-line manual pages
/usr/src	Source code for nonlocal software (not widely used)
/usr/tmp	More temporary space (preserved between reboots)
/var	System-specific data and a few configuration files
/var/adm	Varies: logs, setup records, strange administrative bits
/var/log	System log files
/var/run	Same function as **/run**; now often a symlink
/var/spool	Spooling (that is, storage) directories for printers, mail, etc.
/var/tmp	More temporary space (preserved between reboots)

a. The distinguishing characteristic of **/sbin** was originally that its contents were statically linked and so had fewer dependencies on other parts of the system. These days, all binaries are dynamically linked and there is no real difference between **/bin** and **/sbin**.

Linux Filesystem: File Path

- File Path: is a unique location to a file or a directory in a filesystem. A path to a file is a combination of / and alpha-numeric characters
  - Absolute path/Full path: the location of a file or directory from the root directory (/)
  - Relative path: the present working directory.
  - **`pwd`**: command to print name of current/working directory

```
[root@localhost sysconfig]# pwd
/etc/sysconfig
```

Linux Filesystem: Basic Directory Operation

- **cd** : command used to enter a directory
- Example of usage:
  - **cd** : go to current user's home directory
  - **cd /path** : go to a directory (can be full path/relative path)
  - **cd ~/** : go to current user's home directory, can be followed by subdirectory name
  - **cd ..** : go to parent directory

Linux Filesystem: Basic Directory Operation

- **`mkdir`** : create a directory
- **`rmdir`** : remove a directory (empty)
- Example:

```
[root@localhost ~]# mkdir temp
[root@localhost ~]# ls
anaconda-ks.cfg  dir.txt  err.txt  temp
[root@localhost ~]# rmdir temp
[root@localhost ~]# ls
anaconda-ks.cfg  dir.txt  err.txt
[root@localhost ~]#
```

How to remove non-empty directories?

Linux Filesystem: disks and partition

- **fdisk**: command to manipulate disk partition table
- **df** : command to report file system disk space usage
- **mount** : command to mount a filesystem
  - All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The **mount** command serves to attach the filesystem found on some device to the big file tree.
- **umount** : command will detach it again.

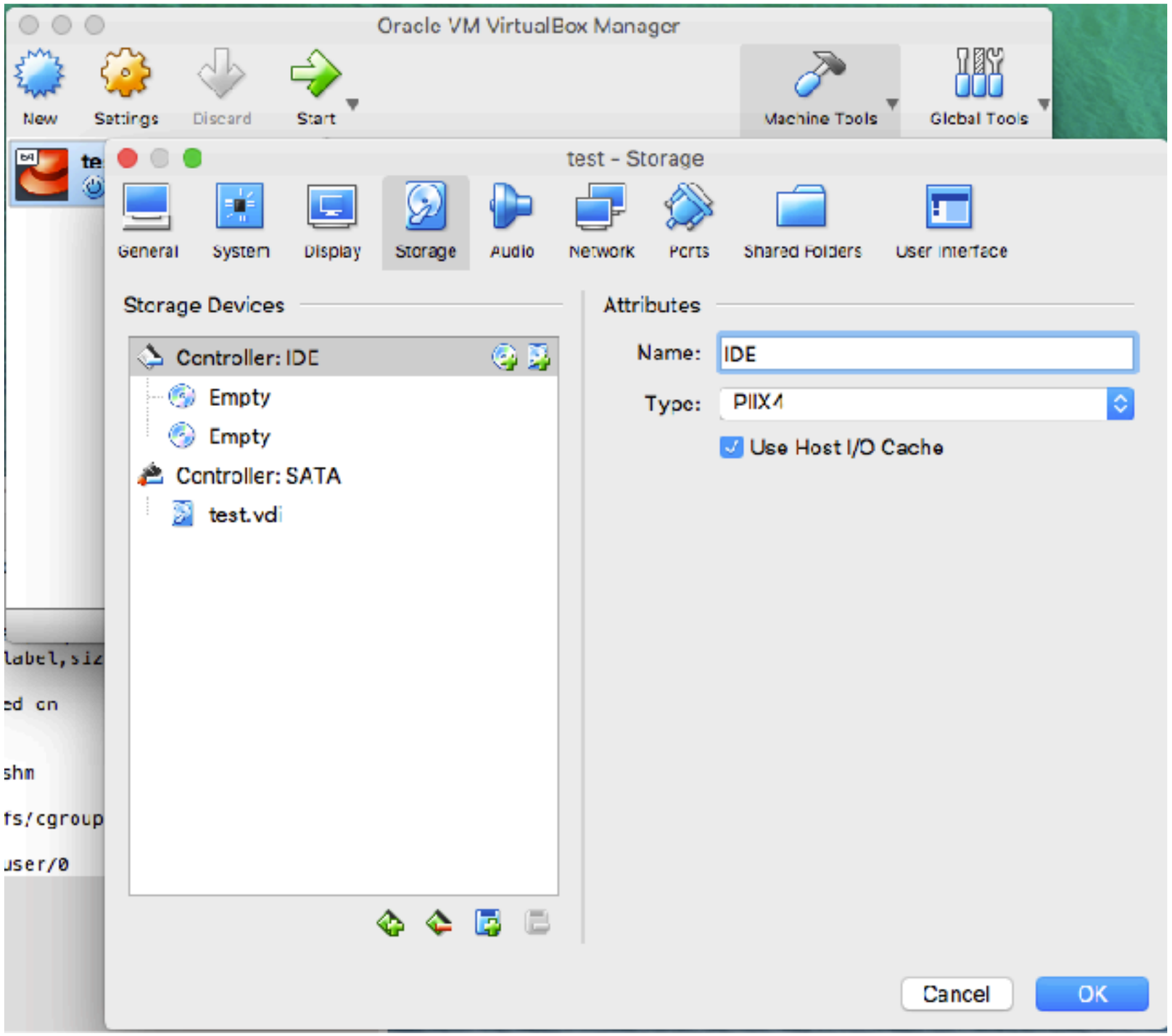
Exercise: Let's add a new hard drive to your system

- Let's check the existing filesystems by execute the following command:
- **df -h**

```
[root@localhost mnt]# df -h
Filesystem                  Size  Used Avail Use% Mounted on
/dev/mapper/centos-root     6.2G  897M  5.4G  15% /
devtmpfs                    486M     0  486M   0% /dev
tmpfs                       497M     0  497M   0% /dev/shm
tmpfs                       497M  6.6M  490M   2% /run
tmpfs                       497M     0  497M   0% /sys/fs/cgroup
/dev/sda1                  1014M  125M  890M  13% /boot
tmpfs                       100M     0  100M   0% /run/user/0
```

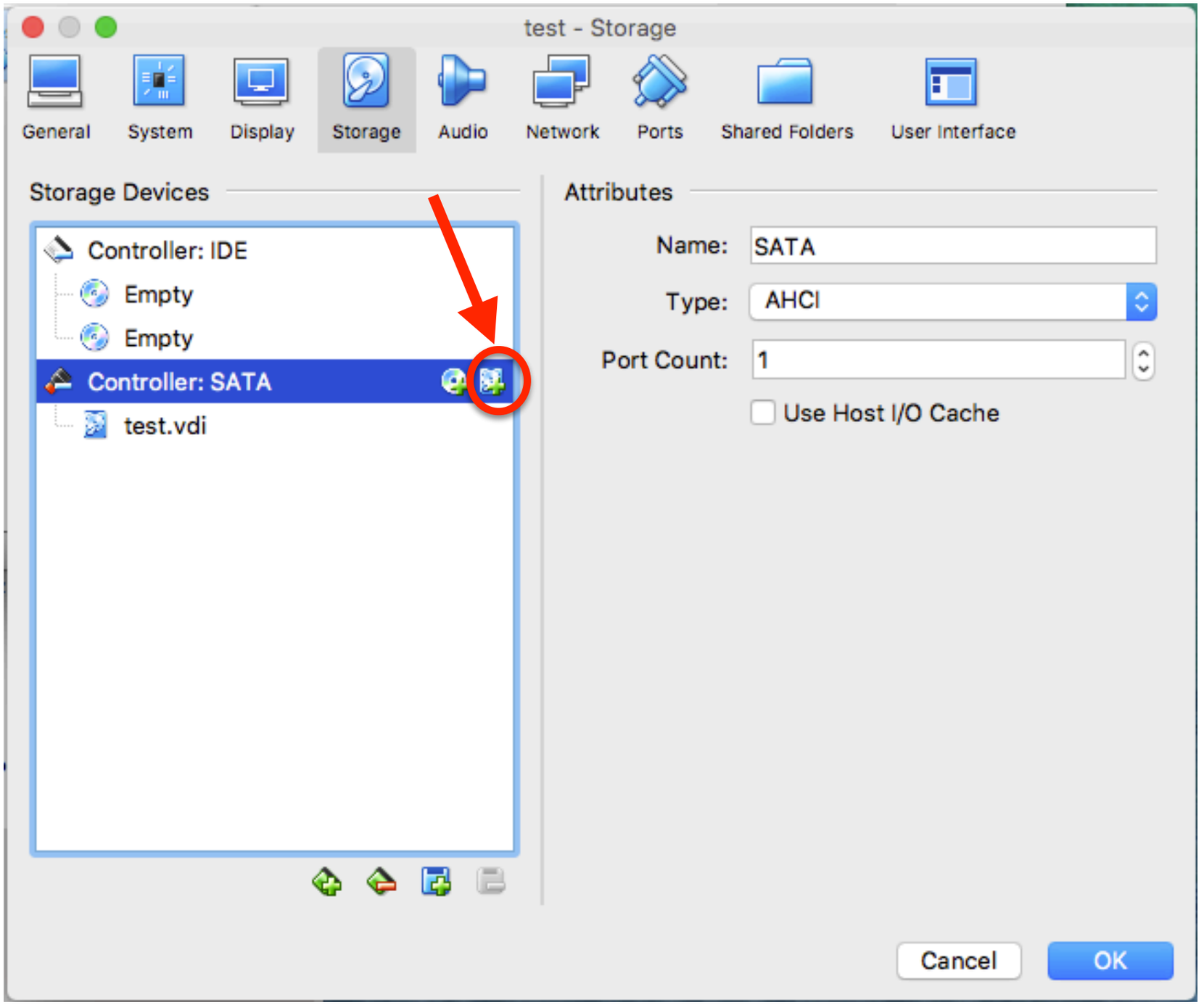
Exercise: Let's add a new hard drive to your system

- Poweroff your VM
- Go to 'Settings' for your VM



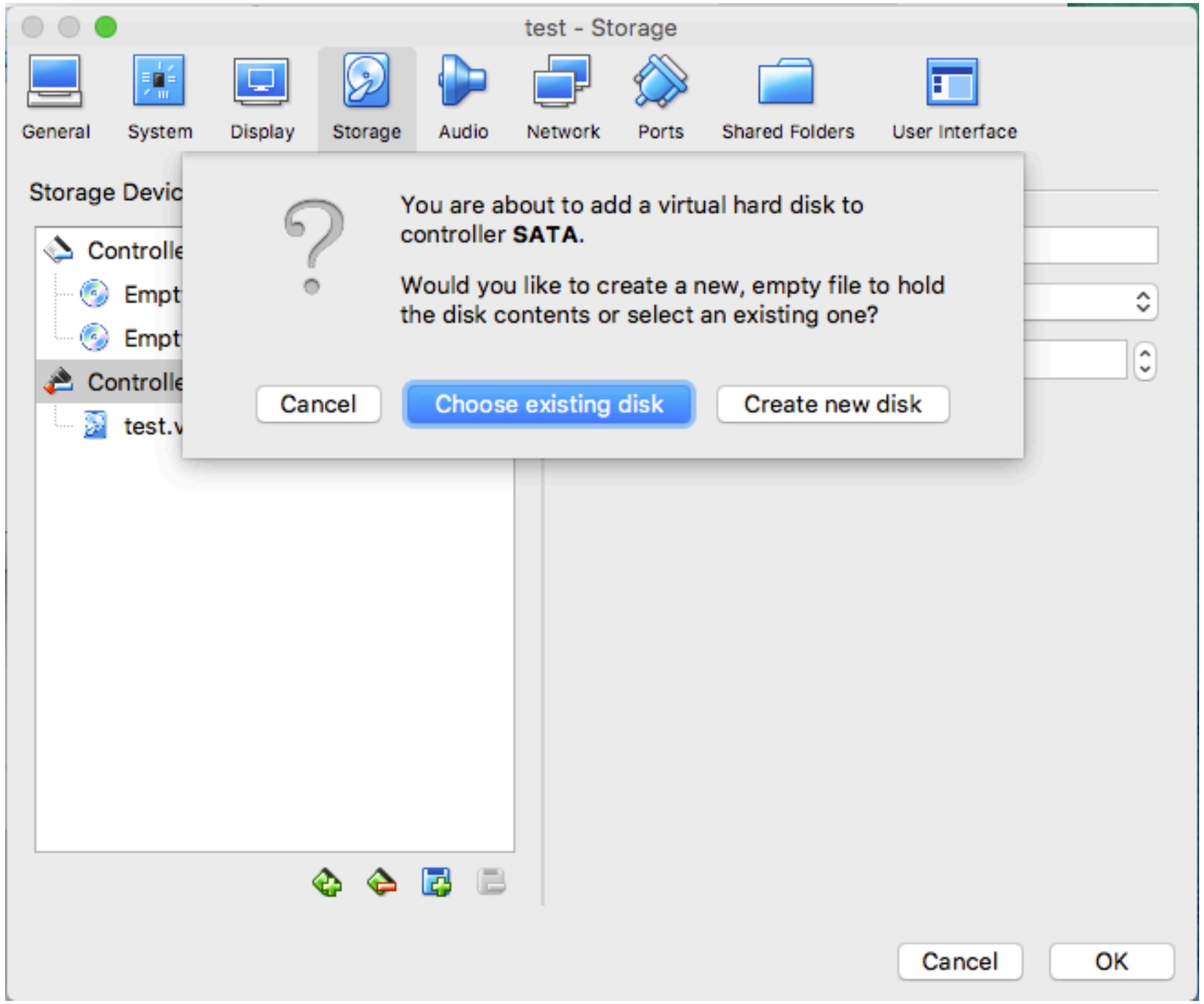
Exercise: Let's add a new hard drive to your system

- Add a new hard drive to your SATA Controller



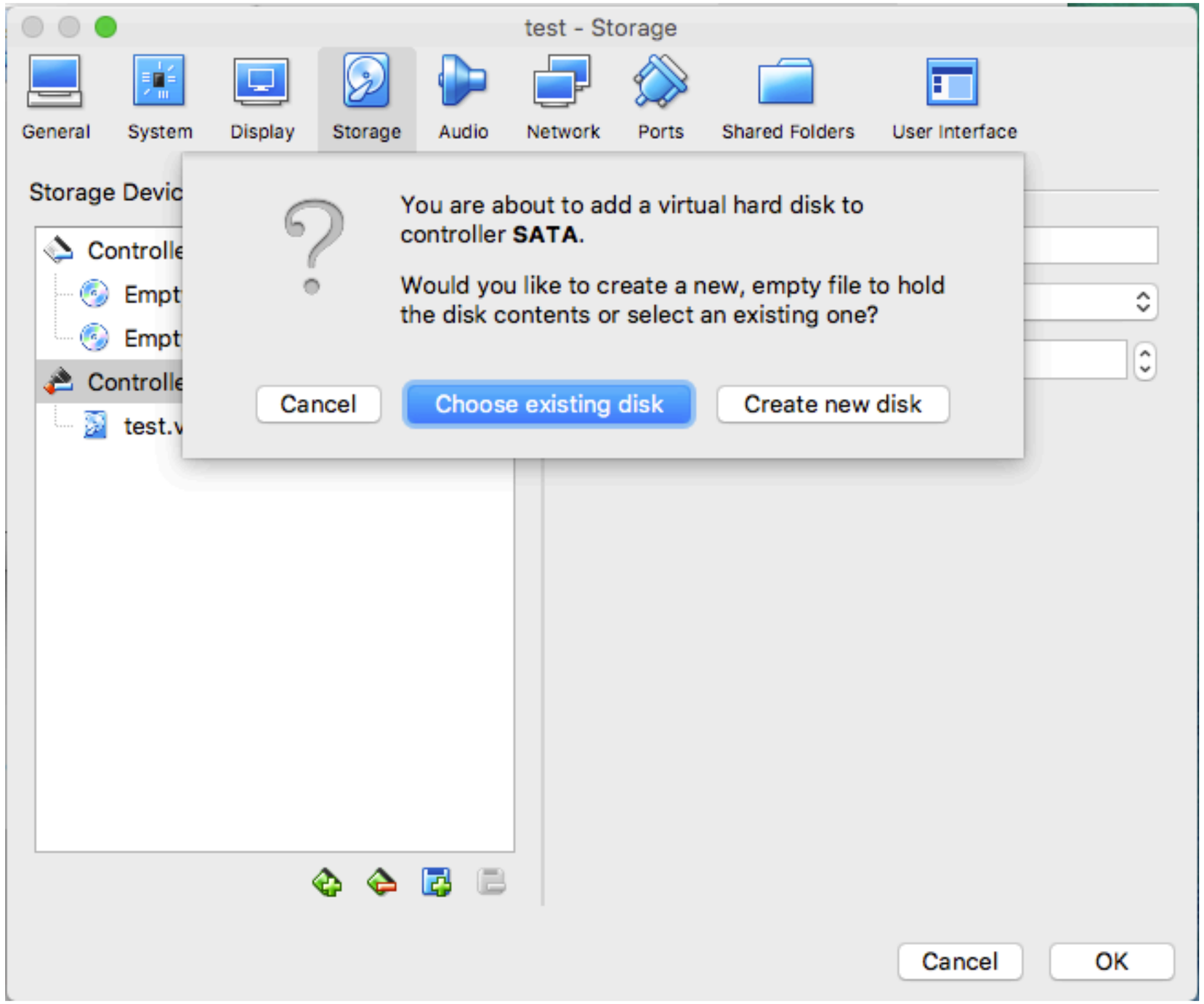
Exercise: Let's add a new hard drive to your system

- Create a new disk



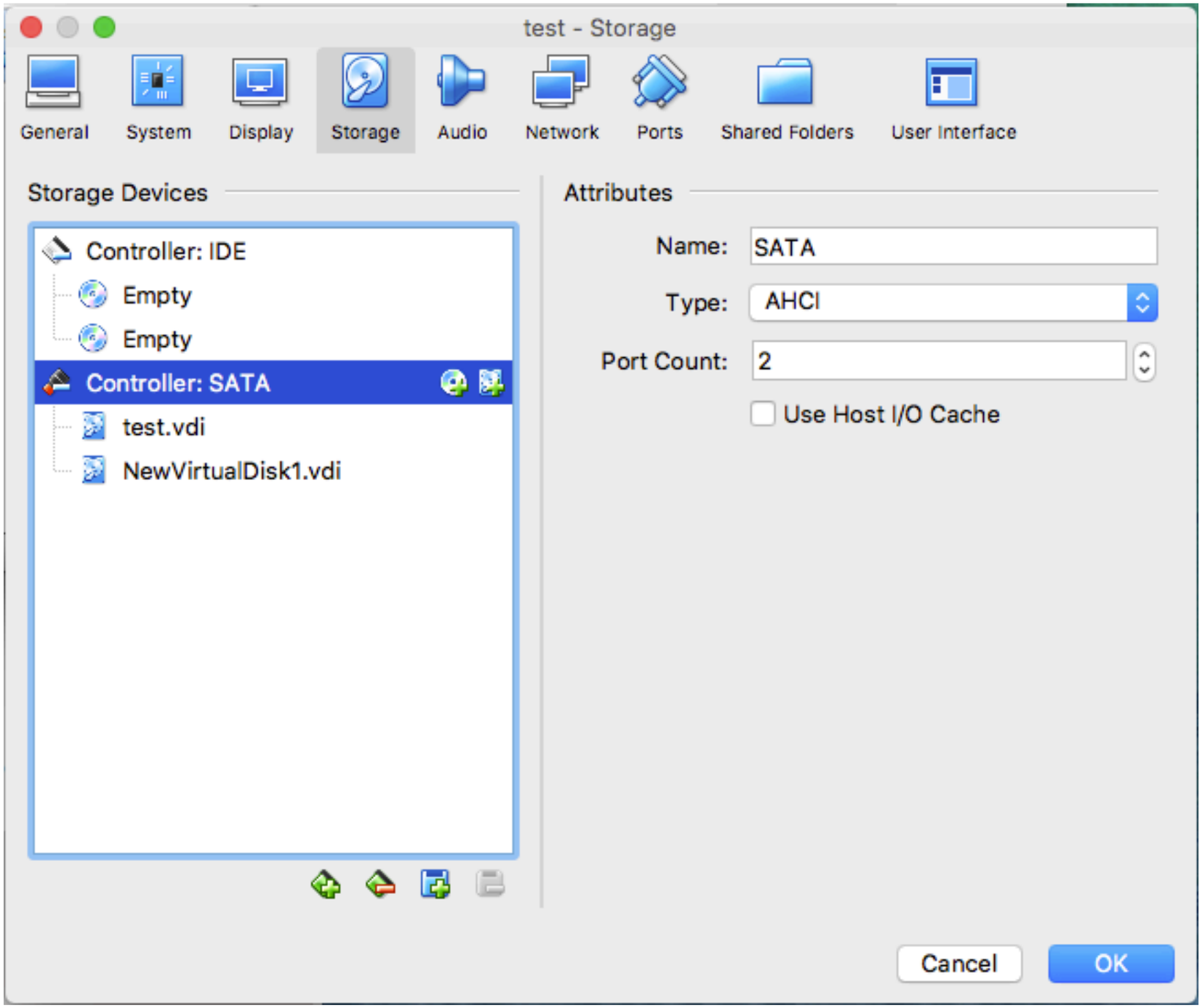
Exercise: Let's add a new hard drive to your system

- Create a new disk
- Follow the instructions to create a new disk with desired size



Exercise: Let's add a new hard drive to your system

- Create a new disk
- Follow the instructions to create a new disk with desired size



Exercise: Let's add a new hard drive to your system

- Poweron your VM
- Check your mounted filesystem with "df"
- Can you find your new HDD?

```
[root@localhost ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/centos-root   6.2G  897M  5.4G  15% /
devtmpfs                  486M     0  486M   0% /dev
tmpfs                     497M     0  497M   0% /dev/shm
tmpfs                     497M  6.6M  490M   2% /run
tmpfs                     497M     0  497M   0% /sys/fs/cgroup
/dev/sda1                1014M  125M  890M  13% /boot
tmpfs                     100M     0  100M   0% /run/user/0
```

Exercise: Let's add a new hard drive to your system

- Use '`fdisk -l`' to find your new HDD

```
[root@localhost ~]# fdisk -l

Disk /dev/sda: 8589 MB, 8589934592 bytes, 16777216 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: 0x0000837e

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    16777215     7339008   8e  Linux LVM

Disk /dev/sdb: 8589 MB, 8589934592 bytes, 16777216 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 /dev/mapper/centos-root: 6652 MB, 6652166144 bytes, 12992512 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 /dev/mapper/centos-swap: 859 MB, 859832320 bytes, 1679360 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
```

Exercise: Let's add a new hard drive to your system

- Use '`fdisk /dev/sdb`' to partition your new HDD

```
[root@localhost ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0xd3813d0d.

Command (m for help):
```

Exercise: Let's add a new hard drive to your system

- Enter m for help with different options

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   g   create a new empty GPT partition table
   G   create an IRIX (SGI) partition table
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Exercise: Let's add a new hard drive to your system

- Enter **n** to create a new partition
- Use default for partition type, partition number and first sector
- Set your size of the partition:
  - in this case, we want to partition a 8G HDD into two partitions, each has 4G space
  - type '+4G'

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
Using default response p
Partition number (1-4, default 1):
First sector (2048-16777215, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-16777215, default 16777215): +4G
Partition 1 of type Linux and of size 4 GiB is set
```

Exercise: Let's add a new hard drive to your system

- Enter 'w' to write the partition table to disk

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Exercise: Let's add a new hard drive to your system

- Use 'fdisk -l' to check your partition:

```
[root@localhost ~]# fdisk -l

Disk /dev/sda: 8589 MB, 8589934592 bytes, 16777216 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: 0x0000837e

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    16777215     7339008   8e  Linux LVM

Disk /dev/sdb: 8589 MB, 8589934592 bytes, 16777216 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: 0x74d100b5

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     8390655     4194304   83  Linux
/dev/sdb2         8390656    16777215     4193280   83  Linux
```

Exercise: Let's add a new hard drive to your system

- before you can use any partitions, you have to create a filesystem (type of filesystem) in your partitions
- use '`df -T`' to check the existing filesystem type

```
[root@localhost ~]# df -hT
Filesystem                Type      Size  Used Avail Use% Mounted on
/dev/mapper/centos-root xfs       6.2G  897M  5.4G  15% /
devtmpfs                  devtmpfs  486M     0  486M   0% /dev
tmpfs                     tmpfs     497M     0  497M   0% /dev/shm
tmpfs                     tmpfs     497M  6.6M  490M   2% /run
tmpfs                     tmpfs     497M     0  497M   0% /sys/fs/cgroup
/dev/sda1                 xfs      1014M  125M  890M  13% /boot
tmpfs                     tmpfs     100M     0  100M   0% /run/user/0
[root@localhost ~]#
```

Exercise: Let's add a new hard drive to your system

- use 'mkfs.xfs' to format both of your new partitions to xfs filesystem

```
[root@localhost ~]# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=262144 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=1048576, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

Exercise: Let's add a new hard drive to your system

- use 'mkfs.xfs' to format both of your new partitions to xfs filesystem

```
[root@localhost ~]# mkfs.xfs /dev/sdb2
meta-data=/dev/sdb2              isize=512    agcount=4, agsize=262080 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=1048320, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

Exercise: Let's add a new hard drive to your system

- Now we need to create a mount point to mount the new hard drive so that we can access the new partitions through file system
- Let's create two new directories in root directory

```
[root@localhost ~]# mkdir /newPartition1 /newPartition2
[root@localhost ~]# ls /
bin   dev  home  lib64  mnt           newPartition2  proc  run   srv  tmp  var
boot  etc  lib   media  newPartition1  opt            root  sbin  sys  usr
```

Exercise: Let's add a new hard drive to your system

- Let's mount partition /dev/sdb1 on /newPartition1 and mount partition /dev/sdb2 on /newPartition2:
- `mount /dev/sdb1 /newPartition1`
- `mount /dev/sdb2 /newPartition2`
- use 'df' to check the disk usage

```
[root@localhost ~]# mount /dev/sdb1 /newPartition1
[root@localhost ~]# mount /dev/sdb2 /newPartition2
[root@localhost ~]# df -lh
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/centos-root   6.2G  897M  5.4G  15% /
devtmpfs                  486M     0  486M   0% /dev
tmpfs                     497M     0  497M   0% /dev/shm
tmpfs                     497M  6.6M  490M   2% /run
tmpfs                     497M     0  497M   0% /sys/fs/cgroup
/dev/sda1                1014M  125M  890M  13% /boot
tmpfs                     100M     0  100M   0% /run/user/0
/dev/sdb1                 4.0G   33M  4.0G   1% /newPartition1
/dev/sdb2                 4.0G   33M  4.0G   1% /newPartition2
```

File Types

- Regular files
- Directories
- Character device files
- Local domain sockets
- Named pipes (FIFOs)
- Symbolic links

Check File Information

- `file` command is used to determine the file types
- `ls` command is used to list directory contents
  - commonly used options:
    - `-a: list all files including entries starting with .`
    - `-d: list directories`
    - `-l: list detailed information`
    - `-h: list files with human readable format`
    - `-R: Recursively list Sub-Directories`
    - `-r: Reverse output order`
    - `-S: Sort files by file size`
    - `-i: print the inode number`

Understand detailed file list information

- Use **`ls -al`** to list all files in your directory

```
total 36
dr-xr-x---.  2 root root  165 Jan 24 01:28 .
dr-xr-xr-x. 17 root root  224 Jan 23 23:24 ..
-rw-------.  1 root root 1235 Jan 23 23:25 anaconda-ks.cfg
-rw-------.  1 root root  270 Jan 24 01:38 .bash_history
-rw-r--r--.  1 root root   18 Dec 28  2013 .bash_logout
-rw-r--r--.  1 root root  176 Dec 28  2013 .bash_profile
-rw-r--r--.  1 root root  176 Dec 28  2013 .bashrc
-rw-r--r--.  1 root root  100 Dec 28  2013 .cshrc
-rw-r--r--.  1 root root   48 Jan 24 01:25 dir.txt
-rw-r--r--.  1 root root   47 Jan 24 01:29 err.txt
-rw-r--r--.  1 root root  129 Dec 28  2013 .tcshrc
```

Understand detailed file list information

- Use **`ls -al`** to list all files in your directory
- Meaning of each field:
  - File permissions
  - Number of links
  - Owner name
  - Owner group
  - File size
  - Time of last modification
  - File/Directory name

Understand detailed file list information

- Use `ls -al` to list all files in your directory
- Meaning of each field:
  - File permissions: 10 characters
    - First character is file type

File type	Symbol	Created by	Removed by
Regular file	-	editors,cp,etc.	rm
Directory	d	mkdir	rmdir, rm -r
Character device file	c	mknod	rm
Block device file	b	mknod	rm
Local domain socket	s	socket (system call)	rm
Named pipe	p	mknod	rm
Symbolic link	l	ln -s	rm

Understand detailed file list information

- Use `ls -al` to list all files in your directory
- Meaning of each field:
  - File permissions: 10 characters
    - First character is file type
    - Three sets of characters, three times, indicating permissions for owner, group and other:
  - r : readable
  - w: writable
  - x: executable

Regular files

- Most files are just regular files
- Consist of a series of bytes
- Filesystems impose no structure on their contents
- Text files, data files, excitable programs, shared libraries and etc.
- Both sequential and random access are allowed

Create a File

- File creation: there are many ways to create a file
- We can create a file by redirecting output to a file (>, >>)
- `echo 'xxx' > file`
- We can use 'touch' command
- `touch filename`
- We can use editors (vim) to create file

File Name

- General rules for file naming:
  - All file names are case sensitive.
  - You can use upper and lowercase letters, numbers, “.” (dot), and “_” (underscore) symbols.
  - You can use other special characters such as blank space, but they are hard to use and it is better to avoid them.
  - Most modern Linux and UNIX limit filename to 255 characters (255 bytes). However, some older version of UNIX system limits filenames to 14 characters only.
  - A filename must be unique inside its directory.
  - No file extension in Linux
  - files start with '.' are hidden files

File Name

- Avoid using the following characters from appearing in file names
  - /
  - >
  - <
  - |
  - :
  - &

File Operation

- Move or rename a file command: **mv**
  - Move a file to another directory:
    - `mv file_name target_directory`
  - Rename a file: move file in the same directory
    - `mv file new_filename`
  - Move a directory to another directory:
    - `mv dir new_dir`
  - Rename a directory:
    - `mv dir new_dir_name`
  - Move multiple files at a time:
  - `mv file1 file2 file3 new_dir`

File Operation

- Copy a file/directory:
  - Copy a file to another directory:
    - `cp file_name target_directory`
  - Copy a directory to another directory (recursively copy all sub dirs):
    - `mv -R dir new_dir`
- Remove a file:
    - `rm file name`
  - Option: -f (force to remove without confirmation)
  - remove all files in a directory:
    - `rm *`
  - remove a directory:
  - `rm -fr dir_name`

View a File

- **`cat`**: Concatenate files and print on the standard output
- Commonly used options:
  - -n : number all output lines
  - -v : show nonprinting
- cat is not good for long files

- For crt viewing:
  - **`more`** : a filter for paging through text one screenful at a time.
  - **`less`** : provides **more** emulation plus extensive enhancements.

File Operation

- diff : file comparison (line by line)
  - commonly used options:
    - -q : report only when files differ
    - -s : report only when files are the same
    - -c : output NUM (default 3) lines of copied context
- wc : word count of a file (newline, word, and byte)
  - commonly used options:
    - -c : print the byte counts
    - -m : print the character counts
    - -l : print the newline counts
    - -w : print the word counts

Directories

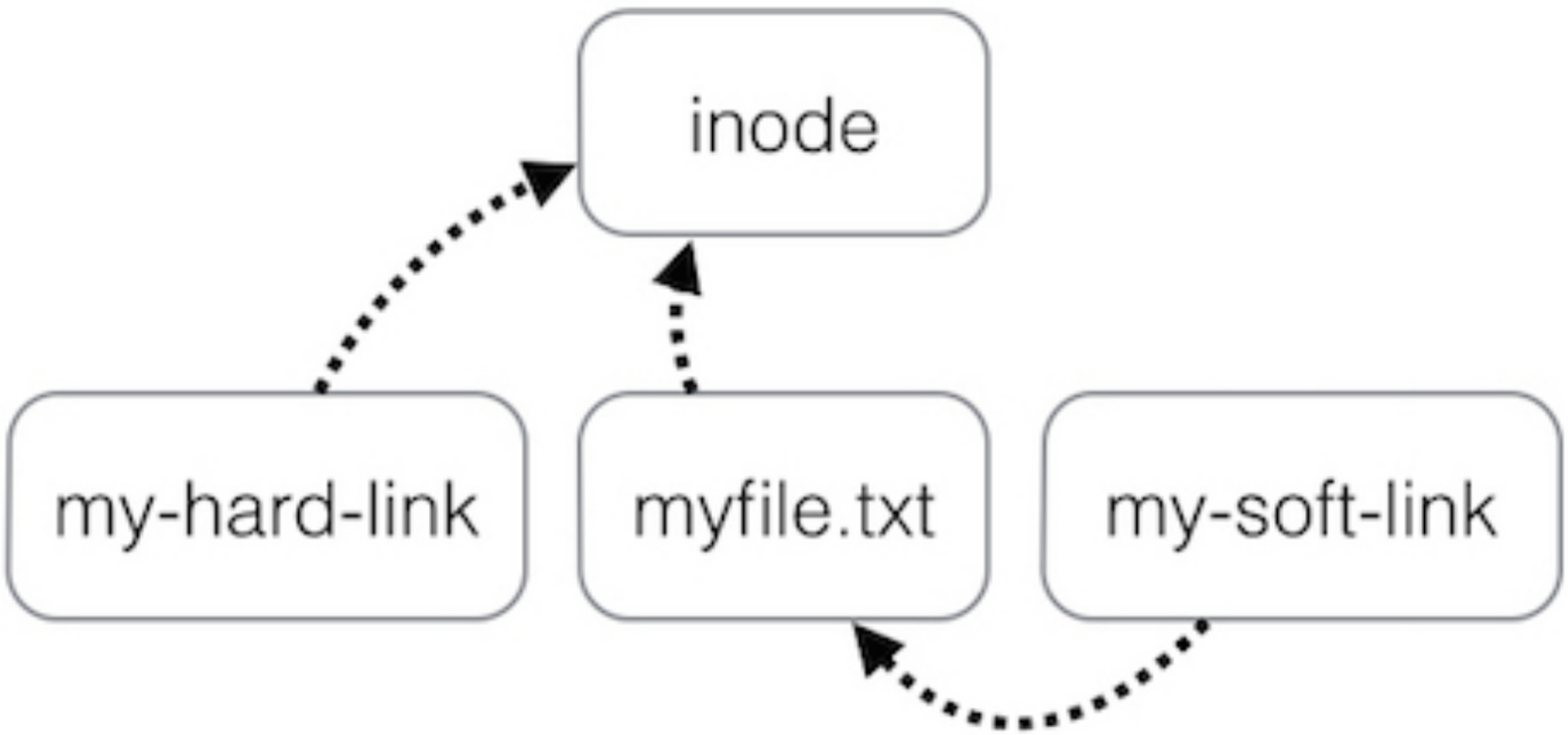
- A directory contains named references to other files.
- "." : refers to directory itself
- ".." : refers to its parent directory

Hard links

- A file name is stored within its parent directory, not with the file itself
- More than one directory can refer to a file at one time, and the references can have different names
- More than one entry in a single directory can refer to a file at one time, and the references can have different names
- These additional references (Hard links) are synonymous with the original file
- System maintains a count of the number of links
- Cannot be created for directories
- Cannot cross filesystem boundaries or span across partitions

Symbolic Links (soft links)

- A symbolic link (soft link) points to a file by name



Create hard and symbolic links

- Create a hard link:
- `ln oldfile newfile`

- Create a symbolic link:
- `ln -s oldfile newfile`