

Course: CIS 68C1-01

Homework: #4

Due: Monday, October 22th, 6pm

Instructions

Complete the problems assigned below and turn in your answers by start of class on the due time above. Please feel free to email your homework or bring it with you to class.

Answer the following questions. This will require you to fully explore and learn the concepts and behavior of the commands presented in the lecture and text. Show the command used to answer and command as well as the output.

1. Use the **find** command to find all the **setuid** files on the Linux system. Hint: look at the find command's man page, especially the **-perm** option.

You could use either of the following:

```
find / -perm -4000 -print
```

or

```
find / -perm +4000 -print
```

The **-4000** version (a dash in front of the permissions) means **all** of the specified permissions must be set in the file, and the **+4000** version means **any** of the specified permissions can be set. Since the problem only asks you to find a single permission bit (4000), both forms end up acting the same way. This would not have been the case if you needed to find, for example, all files that were **either** setuid or setgid. In that case the permissions would be $4000+2000 = 6000$, and you would only want to find files that had **either** bit set. You would use **+6000** as the permissions to find, because **-6000** would cause find to look for files that had **both** the 4000 and the 2000 bit set.

2. Your current working directory is / and you find that you cannot un-mount the /home filesystem. Explain the reason why, and what you would need to do to resolve this.

You cannot un-mount filesystems that are **busy**. This means some process has its current working directory in the filesystem. This could be your shell, or some other process. In the case of /home which would typically contain users home directories, all users logged on would have had their startup shell processes start within the /home filesystem - thus, /home would be busy. The only way to un-mount such a filesystem would be to have all users logout, and perhaps even have to bring the system down to single user mode to kill any processes that were running with there current working directory somewhere under /home.

3. In what directory would you find various system log files?

/var/log

4. In which filesystem does /etc reside and why is it there?

The root filesystem. The /etc directory contains critical system startup and configuration files required to boot.

5. You can create a hard link to a file - can you create one to a directory? Justify why or why not.

No. You are prevented from creating hard links to directories. This would allow for cycles in directory structures. The file tree must not contain cycles. Symbolic links can be created to seemingly create cycles (loops) in the directory structure. There kernel and shells handle this specially, looking for cycles.

6. You have two directories **dir1** and **dir2** under a directory called **testing** in your home directory. You create a symbolic link in **dir2** using the command:

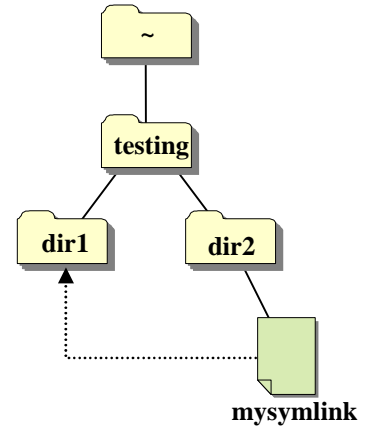
```
ln -s ~/testing/dir1 ~/testing/dir2/mysymlink
```

Then you do the command:

```
rm -rf ~/testing/dir2/mysymlink/..
```

What was removed? Explain your answer.

At first glance, it might seem that the command would recursively remove the directory **dir2**. After all, you generally think of **dot dot** as going back one component in the pathname (i.e. `~/testing/dir2/mysymlink/..` is usually the same as `~/testing/dir2`). However, this only works when the component preceding the **dot dot** is a *directory*, and **mysymlink** is *not* a directory; it is a symlink! When a symlink is encountered in a path, it must be followed *immediately*. Think of it as doing a **cd** into each path component you encounter when reading the path. Trace your steps on the picture to the right. Doing a **cd** into each component of `~/testing/dir2/mysymlink` brings you first to `~`, then to `~/testing`, then `~/testing/dir2`, then when you encounter **mysymlink**, you must immediately follow it to `~/testing/dir1`; then the final `..` brings you up to **dir1**'s parent, which is **testing**. Thus, the command would recursively remove the directory **testing** and everything below it.



7. Give the command that you would place, and the name of the file you would place it in, if you wanted to have newly created files and directories for each user created such that **others** would not have write or execute/examine access.

Place the command **umask 003** in the global startup files such as `/etc/profile` and `/etc/csh.cshrc`. This would set a default **umask** that removed **execute** and **write** access for **others**. Certainly, each user could override this by placing a **umask** command in their own startup files.

8. How would you know if a file was a link to another file or just a copy?

The only way to know would be to look at the inode number. You can use **ls -li** to see inode numbers for files and directories. If the inode numbers are the same, they are not copies.

9. If you change the value of the UID for a user in `/etc/passwd`, what additional steps would you need to take? Give all commands required to perform such actions.

You would want to find all files owned by the user, and change their owner IDs to the new UID. Otherwise, the user will not be able to access their files. You can use the **find** command to find all files owned by a particular UID. Once the files were found, you would need to do a **chown** command to change those files to the new owner. There are a couple of ways to do this:

```
find / -uid oldUID -exec chown newUID {} \; -print
```

or

```
chown newUID `find / -uid oldUID -print`
```

10. If user **emmapeel** (UID=9999, GID=2468) created a new file in the directory **guesthome** below, what would be the user ID and group ID of the new file? The UID for user **guest** is 6767, and the GID for group **project1** is 1213.

```
drwxrwsrwx  2 guest project1      512 Oct 15 23:39 guesthome/
```

Because the **setgid** bit is set on the directory **guesthome**, all files created in that directory will take on the group ID of the directory itself (and not the creator's GID). The new files owner ID is not affected. The owner ID for the file would be 9999 and the group ID would be 1213.

11. You see a message on your UNIX system console that says something like "Out of inodes: file system = /home". What does this mean and what was the cause? How can you do to fix this problem?

This means the inode table has become full for that filesystem; no additional files can be created until files are deleted. The inode table is **fixed** in size and **cannot grow**. Thus, each filesystem has a fixed maximum number of files that it can hold. The only way for a filesystem to hold more files is to backup the files and recreate the filesystem with **mkfs**, this time allocating a larger inode table. Fortunately, this problem rarely occurs with today's larger disk.

12. What is the purpose of alternate superblocks in the ext2 implementation of the Berkeley Fast File System?

Alternate superblocks are used as backup superblocks, for **fsck**, in the event that the portion of the disk that contains the primary superblock becomes corrupted. Because the superblock is a master index into all the other tables that manage files and directories, its loss is catastrophic. Backup superblocks give an extra measure of safety. You can tell **fsck** to use an alternate superblock when checking and correcting disk problems.

13. What are the major and minor numbers for the console device in Linux?

The device is **/dev/console**, and its major and minor number are **5** and **1**, respectively.

14. Describe as specifically as you can what the command below would do:

```
mkfs -t iso9660 /dev/hdc
```

The command creates a new filesystem of type **iso9660** on the device **/dev/hdc**. Your device is the **secondary-master** drive on the IDE controller. From the device name alone it is indeterminable what type of device it is - it could be a CD-ROM or it could be a hard disk.

15. Explain the steps required to cause the filesystem on device **/dev/sd01** to be automatically mounted read-only at boot time onto the directory **/mnt/disk1**. Give the names of, and the lines added to, any files modified.

Add the line below to **/etc/fstab**:

```
/dev/sd01 /mnt/disk1 ext2 ro 0,0
```

16. You run a filesystem check, and the checker utility finds that the filesystem is corrupted; specifically, it finds several files that do not have a parent directory. What happens to these files?

The **fsck** command will place the files in the filesystems **lost+found** directory, at the top of that filesystem.

17. If you have six SCSI disks on Linux system, what would the full path of the device name that references the third partition of the fifth drive?

/dev/sde3