Computer Science 330
Operating Systems
Siena College
Spring 2012

Lab 8: File Systems

Due: 9:20 AM, Friday, April 20, 2012

In this lab, you will learn about some details of file systems beyond what we have covered in class.

You may work alone or with a partner on this lab.

Create a file in your directory for this lab in which you will answer the questions scattered throughout the lab.

Standard Unix File Protection

Unix is a multi-user operating system, and you must authenticate to indicate "who you are" to a Unix system. A major reason for this is to facilitate file protection. File protection allows the creator of a file to decide who should be able to view a file, modify a file, or execute a file.

We will first consider the "standard" Unix file permission scheme, then look at the more flexible permissions provided with the AFS file system, such as that used by our SoS home directories.

Unix UIDs and GIDs

The files that a user is permitted to access are determined by the user's processes' *user id (UID)* and *group id (GID)*. Each user on a Unix system is assigned a unique UID, which is a number associated with the login name, and can be part of one or more *groups*.

You can find out which UID is associated with your shell using the `id` command.

Question 1: What is the output of `id` on a lab Linux system and on `winterstorm.teresco.org`?

User/Group/Other permissions

All files in a standard Unix filesystem have an *owner*, which should correspond to the UID of a valid user on the system, and a *group*. Each file or directory has a set of permissions that specify what kinds of operations on that file or directory are permitted for the *owner*, i.e., a process with the same UID as the file, for the *group*, i.e., a process whose GID is the same as the file, and for all other processes on the system.

To see a file's permissions, we use `ls -l` to obtain a long-format directory listing. Here are the first few lines printed when I do this on the `/home/cs330/examples` directory on `winterstorm.teresco`

```
-> ls -l
total 82
drwxr-xr-x  2 terescoj  terescoj   512 Jan 19 15:43 addingone
-rw-r--r--  1 terescoj  terescoj   549 Jan 19 15:43 addingone.tar.gz
drwxr-xr-x  2 terescoj  terescoj   512 Feb 24 09:17 everyother
```

```
-rw-r--r--  1 terescoj  terescoj   958 Feb 24 09:17 everyother.tar.gz
drwxr-xr-x  2 terescoj  terescoj   512 Feb 24 09:18 exec
-rw-r--r--  1 terescoj  terescoj  1807 Feb 24 09:18 exec.tar.gz
```

Let's dissect this output. The total line indicates how many kilobytes of disk the files use. Each subsequent line provides information about a file. The first chunk of text specifies file permissions (more on this below). Next is the number of hard links to the file, which we discussed previously. Next are is the UID and GID of the file's owner. The rest is the size of the file in bytes, the date and time that the file was last modified, and the name of the file.

The file permission string starts with a single character indicating a file type, followed by three triples. The first triple specifies permissions for the owner, the next the permissions for the members of the group, and the last the permissions for all other users.

Question 2: What is the UID and GID of the files in your home folder on a lab Linux system and on `winterstorm.teresco.org`?

Each triple indicates whether a category of processes can do each of three operations:

- Read permission - can the file be read?
- Write permission - can the file be modified?
- eXecute permission - can the file be executed?

The file type character is `-` for normal files, and `d` for directories.

Question 3: What other file types have we seen this semester? Hint: there are at least 2 we have seen but there are several others.

Each of the three successive triples specifies the read, write, and execute permissions. The letter is present if the permission is granted, and will be a `-` if not:

- `r` - If present, read permission is given to that category of user
- `w` - If present, write permission is given to that category of user
- `x` - If present, execute permission is given to that category of user

The meaning of protection is interpreted a bit differently for directories:

- `r` - If present, that category of user can list the contents of the directory using `ls`. (They cannot necessarily see the contents of the files in the directory.)
- `w` - If present, that category of user can create or delete files in the directory.
- `x` - If present, that category of user can change to that directory using the `cd` command.

Changing the file protection

Only the owner of a file or the system administrator may change a file's protection. This is done with the `chmod` command. To specify the protection changes, you identify the class(es) of users whose permissions you wish to change:

- `u` - The owner of the file.
- `g` - Members of the group
- `o` - All other users. (Careful! `o` does not mean owner.)
- `a` - All users (owner, group, and others combined)

Next you indicate if you want to add or remove permission:

- `+` - add permission
- `-` - remove permission

Finally, you indicate which type of permission you are adding or removing, using `r`, `w`, and `x`. So, if you want to change your files so that nobody else can read or execute them, you would say:

```
-> chmod go-rx *
```

Question 4: Create a new directory in your home directory on `winterstorm.teresco.org`. What permissions does it have?

Question 5: Change the permissions so only you can read the directory. Specify the `chmod` command you used and show the output of `ls -l` after making this change. Ask a classmate to verify that he cannot `cd` to your directory.

Read the `man` page for `chmod`'s section about specifying an absolute mode. Ignore the part about the setuid bit, the setgid bit, and the sticky bit for now (we'll talk about those in class next week).

Question 6: What absolute mode is used to specify read-write access for the owner, read-only access for group and everyone?

Question 7: What absolute mode is used to specify read-write access for the owner, no access for group and everyone?

Question 8: What absolute mode is used to specify read-only access for the owner, no access for group and everyone?

Question 9: What absolute mode is used to specify all access for the owner, read-execute access for group and everyone?

Question 10: Change the permissions back to allow everyone to be able to change into the directory you made for these questions. Now create two files in the directory. What permissions do they have?

Question 11: Change the permissions of one of the files so only you have permission to read the file. Specify the `chmod` command you used and show the output of `ls -l` after making this change. Ask a classmate to verify that he can still see the contents of the file whose permissions you did not change but cannot see the contents of the one you did change.

Question 12: Change the permissions of the directory so that no one has write premission. What happens when you try to create a file in the directory?

Question 13: Change the permissions of the directory so that no one has read permission but you have execute permission. What happens when you try to `cd` to the directory? What happens when you try to `ls` its contents? What happens when you try to view the contents of a file in the directory?

Question 14: Change the permissions of the directory so it has read and write access for you, but not execute access. What happens when you try to `cd` to the directory? What happens when you try to `ls` its contents. What happens when you try to view the contents of a file in the directory?

User file-creation masks

Read the `man` page for `bash` and search for the paragraph about the `umask` builtin command. Cryptic, isn't it? Let's figure out what it's all about.

Question 15: What output does the builtin command "`umask`" produce for you on `winterstorm.teresco.c`

Question 16: Create a file on `winterstorm.teresco.org`. What are the file's permissions?

Question 17: Now change your user file-creation mask using the command "`umask 0`" and create another file. What are the file's permissions? Create a directory. What are the directory's permissions?

Question 18: Now change your user file-creation mask using the command "`umask 022`" and create another file. What are the file's permissions? Create a directory. What are the directory's permissions?

Question 19: Now change your user file-creation mask using the command "`umask 077`" and create another file. What are the file's permissions? Create a directory. What are the directory's permissions?

Question 20: Now change your user file-creation mask using the command "`umask 777`" and create another file. What are the file's permissions? Create a directory. What are the directory's permissions?

Question 21: Explain briefly how the `umask` value affects the default permissions for files and directories you create.

AFS File Protection

The *Andrew File System*, now known just as *AFS*, is a globally-distributed file system. Originally from CMU, later supported by a company called Transarc, which has since become part of IBM. IBM has released AFS as an open source product.

The SoS Linux systems use an AFS partition to manage home directory space (which as we know also serves as the "Z drive" in Windows). So we have the opportunity to experiment with file permissions in AFS, which are much more flexible than the standard Unix file permissions.

There are many existing documents describing AFS permissions. Start by reading this one and feel free to search for other information. Then answer these questions.

Question 22: What are the AFS permissions for your home directory on the SoS Linux system? Explain briefly what this means.

Question 23: Create a new directory. Verify that you can add a file to this directory. Remove "insert" permissions from the directory. What command did you use? Verify that this worked by showing the permissions for the directory (and paste in your output). Now try to add another file. What happens?

Question 24: Return to your home directory and remove all permissions from the directory you created for the previous step. What happens when you try to `cd` to this directory? What happens when you try to `ls` the contents of this directory?

Question 25: For the next part of the lab, you will create a directory to which you will allow access to some or all your classmates. Create a new directory and set it so that it has read and list access for any user on the system. What command did you use? Ask a classmate to verify that he can now `cd` to your directory.

Question 26: Create a file in this directory and give it world read permission using a standard Unix `chmod` command. Can a classmate view this file? (Ask one to try it to make sure you're right.)

Question 27: Now use the standard Unix `chmod` command to give a file in your directory read-write permissions for you, no access for group or others. Can a classmate view this file? (Ask one to try it to make sure you're right.)

So we have seen how to give everyone various types of access or just yourself various types of access, but what about groups? In the standard Unix permissions system, each file can have a single GID attached, and we can set r,w,x permissions for other members of that group. Unfortunately, this is quite limited in many circumstances. The most important limitation is that you as a regular user of the system do not have the ability to create a new group or modify its membership. This is something that can be done only by a system administrator.

AFS allows regular users to manage their own groups and grant various types of access to directories to members of those groups. Read the section "How to create and manage AFS groups" on this page.

Question 28: Create a group for yourself and a classmate. Show the commands you use to accomplish this and show the output of an appropriate "`pts membership`" command.

Question 29: Create a directory in your home directory and grant access to this directory to the group you just created. Show the "`fs la`" output in this directory. Place a file in the directory and verify that your classmate who is a member of your group can access it. Then ask a second classmate who is not a member of the group to try to access the file and verify that he cannot access it.

Question 30: Many of our courses make use of AFS file permissions to create “dropbox” directories where students can submit files for grading. These are the “hw” folders you may have seen and used, especially for lower-numbered courses. Change to one of these directories and list the access rights. Explain how these rights allow you, as the student, to work in that directory, and your instructor has the rights to read the files for grading.

The Latest in Filesystems

One of the big developments in filesystems over the last several years is the development of ZFS by Sun Microsystems, now Oracle, and the subsequent release of the ZFS project as open-source software.

Read this presentation about ZFS.

There is nothing to submit for this part of the lab.

Submission and Evaluation

This lab will be graded out of 30 points (1 point per question).

By 9:20 AM, Friday, April 20, 2012, submit your answers to the lab questions by email to *jteresco@siena.edu*.