CST8177 - Assignment 2
Administering the system

Submission requirements

This project requires two submissions. If any one of the submission standards is not followed, deductions will result. Plagiarism will not be tolerated. Any assignment found to be plagiarized from another student, whether from a previous semester or in this semester, will result in immediate enforcement of the College plagiarism policy.

Part 1 – emailed gzip-tar file

Send a gzipped tar file submission to your lab teacher's email. The file must be named in the form **A2_8177_<your_name>.tgz** or it will not be considered for marking.

Example: **A2_8177_Laurel.tgz** or for a team of 2 (maximum) **A2_8177_Laurel_Hardy.tgz**

The submission consists of one document, named **A2_output**, containing the following files or command outputs from your Linux system in the order given below after completion of your assignment. Clearly label part using the corresponding file name or command line:

1. Identify the task with the label **Task #1**
    - Content of **/etc/crontab**
    - Content of the file created by the **cron** job
    - Edited contents of **/etc/rsyslog.conf**, **/var/log/rootadmin**, and **/root/cronlog** (only provide relevant information).
2. Identify the task with the label **Task #2** , a repeat of Assignment 1 with ACLs, but provide only the information below.
    - Listing of ACLs of all directories and files. Create and fill out a table that contains the following two columns:
        - Directory name / File name: directories as listed previously in Assignment 1 plus at least one file in each directory
        - ACL entries: all the entries in an ACL pertaining to each item.
3. Identify the task with the label **Task #3**
    - Content of **/boot/grub/grub.conf**

Part 2 – paper document

Place the paper-based submission in your lab teacher's physical dropbox The submission must be stapled and may NOT be submitted in an envelope or folder. It is the paper equivalent of the gzipped tar file with an added cover page and table of contents.

The submission consists of one document, named **A2_output**, containing the following files or command outputs from your Linux system in the order given below after completion of your assignment. Clearly label part using the corresponding file name or command line and attach a cover page listing course information, lab information, and student information as for Assignment 1, including a brief table of contents.

1. Identify the task with the label **Task #1**
    - Same as the Task 1 output above
2. Identify the task with the label **Task #2**
    - Same as the Task 2 output above
3. Identify the task with the label **Task #3**
    - Same as the Task 3 output above

Assignment specifications

Task #1: Scheduling a system task

In this task you have to schedule a task, described below, and log the archiving action. The archiving and logging have to be accomplished using a single **cron** job entry. To set up the **cron** job use command substitution, redirection, and a single command line to accomplish this. For testing, temporarily add an additional entry that executes the same job at a shorter time interval (such as every 10 minutes). Be sure to remove it after testing is complete.

Tip: Additional useful commands are: **logger**, **date**, and **man** (!).

Set up the following **cron** job:

- Create a file in **root**'s home directory that lists all files that have the **setuid** bit set. Only capture good output, eliminating all error messages. Create this file on a monthly basis at the beginning of the month. Set it up as a **cron** job in the system **crontab** file.
- Give the file a descriptive file name and include appropriate date information (month and/or week and/or day) as part of the filename. Every time the file is created log the action to a separate log file named **/var/log/rootadmin**. Use the facility **user** for logging. The format of the log entry has to follow the format used in log files (date/time, host, pid) and the message must be descriptive.
- The scheduler logs via **rsyslog** using the **cron** facility. Extract - using **grep** - the log entries of the scheduler that pertain only to the job you have set up for **root** and save it to the file named /**root/cronlog**. Note that you don't have to include this as part of the **cron** job.

Task #2: Working with the file system

The managers in Assignment #1 are receptive to the idea of introducing ACLs to allow for sharing files between the different groups. In this task you will solve the problem presented in Assignment #1 using ACLs.

The change in the setup must be transparent to the users. In other words, when users log on they must not notice a difference or realize that the location of their common project directory may have changed (except for omitted “filter” directories).

In addition to the brief ACL overview below, be sure to check the man pages and the text.

Task #3: Working with the boot loader

Add an additional boot image entry to your boot loader: the new entry boots into single user mode; protect the boot entry with an encrypted password (use **sesame** as your password).

Overview of ACL

A traditional permission block consists of three types of users: user, group and other. For the discussion about ACL we will refer to these types of users as three classes of users: the user class, the group class, the other class. The user class is the owner; the group class includes all members of the owning group; and the other class is everybody else. We have a total of three permission blocks to work with: user, group, and other.

ACL also works with the same three user classes: user, group and other. However, the group class can consist of multiple entries, such as the owning group, additional groups, called named groups, and additional users, called named users.

Note: A basic ACL has three ACL entries. ACLs with more than the three entries are called extended ACLs. Extended ACLs also contain a mask entry and may contain any number of named user and named group entries.

ACL stands for access control list. In a file system that supports ACLs the permissions of each file object have an ACL representation. An ACL consists of a set of entries that define the access privileges of users. Each of the three classes of users (user, group and other) is represented by an ACL entry. Permissions for additional users or groups occupy additional ACL entries.

The group class can consist of multiple entries. If it does, an additional entry is added to the ACL called the mask entry. The mask entry will take on the highest access of all entries of the group class.

Example: If the owning group is set to **x4x** (read-only)and a named group is set to **x6x** (read-write), the mask is automatically set to **x6x**. The effective permission of the owning group is still **x4x**, and the effective permission of the named group is still **x6x**. When the permission block of the mask is set, then the permissions of the mask acts as a filter.

Example: Based on the example above, assume that you change the mask from **x6x** to **x4x**. In that case The effective permission of the owning group is still **x4x**, but the effective permission of the named group is **x4x**, and not **x6x**.

The two commands used to work with ACLs are **getfacl** and **setfacl**.

Working with basic ACL

Preparation

- ACLs are now the default for ext2, ext3, and ext4 filesystems. To verify that you have ACL set, issue a **setfacl** and confirm it with a **getfacl**. **Only** if either command fails, or the results are not as expected, then you will need to:
    - Create a partition and format it with ext3 or ext4. Note: If you don't have free space on your virtual hard drive or you have LVM installed, add a new virtual hard drive via VMware.
    - Mount the partition as **/home/project**. When mounting the partition, enable ACL support. Set up the partition so it is mounted at startup or set up the partition so it can be easily mounted on demand.
- As root, to test if ACL is on:

```
touch xxx
setfacl -m u:root:6 xxx
getfacl -c xxx
```

  The output should resemble this for the user

```
user::rw-
user:root:rw-
```

Working with extended ACL

- Add read-only access for a named group on the file using **setfacl**:

  setfacl -m g:1000:r-- file

  Note: Throughout the exercise we use the GID 1000; no such group need exist.
- Display the file attributes using **ls**. You will notice two changes: A "**+**" has been added to the permission block to indicate that extended ACL has been used.
- The group permission has changed. The group permission with extended ACL displays the mask entry, NOT the entry of the owning group.
- Display the ACL using **getfacl**. Notice that a mask entry has been added.
- Change the permission for the named group:

  setfacl -m g:1000:rwx file.
- Display the file attributes using **ls**. Notice that the group permission has changed to reflect the most permissive access of all entries in the group class.
- Display the ACL using **getfacl**.
- Change the mask entry using either

  chmod: chmod g-wx file.

  setfacl: setfacl -m m:-wx file
- Display the file attributes using **ls**. Notice that the group permission has changed.
- Display the ACL using **getfacl**. Notice that only the mask entry has been changed, not the owning group entry.

  Explanation: The **chmod** command changes the permissions of the group class. Without extended ACL, the group class is represented by the owning group; with extended ACL, the group class is represented by the mask. Therefore, the permission of the mask has changed.

  Notice that comments have been added to identify the effective permissions of all entries that are influenced by the mask setting. The mask permissions set the upper limit for all group class entries.

  The mask is applied by the bit-wise AND operation, where the result in binary is 1 only if both bits are 1. Otherwise, the result is 0.

mask	**010**	**-w-**
permission	**110**	**rw-**
result	**010**	**-w-**
- Remove an ACL entry:

  setfacl -x g:1000 file
- Remove ACL from the file (removes **mask ---**):

  setfacl -b file

Conclusion

If an entry in the group class is changed to a higher access value, the mask is adjusted to that value. The mask is the upper limit for all objects in the group class. When the mask is set specifically, the effective permissions of group objects is bound by the mask.