Access Control Lab

|  |
| --- |
| Copyrights 2016-2017 Frank Xu, Bowie State University.  The lab manual is developed based on the post  <http://www.sis.pitt.edu/lersais/education/labs/access_control.php>  Comments and suggestions can be sent to wxu@bowiestate.edu |

**Objectives**

The objective of the exercises presented here is to familiarize the students with the access control features available in UNIX-based system.

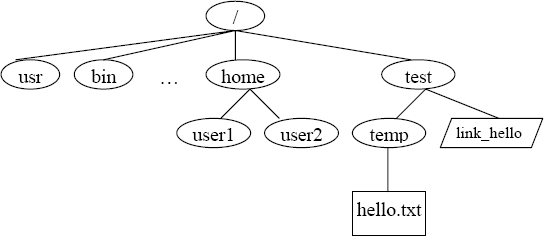
**Lab Environment**

We have created two accounts in the VM. The usernames and passwords are listed in the following:

1. User ID: root, Password: seedubuntu.
   * Note: Ubuntu does not allow root to login directly from the login window. You have to login as a normal user, and then use the command **su** to login to the root account.
2. User ID: seed, Password: dees

**Unix File Hierarchy**

1. The Unix file system is organized as a hierarchy with the root (/) directory at the highest level. Each directory may contain subdirectories and files. Typically, some of the directories that may occur under the root are usr, bin, sbin, home, var, boot, dev, etc. In Figure , user1 and user2 are sub-directories under home. hello.txt is a plain-text file and link\_hello is a linking file that points to hello.txt. In order to access the file /test/temp/hello.txt, the system begins its search from the root(/) folder and then to test and temp folders consecutively and then finally the file hello.txt.

[](http://www.sis.pitt.edu/lersais/education/labs/gfx/access/access2_1.png)

**Ownership and Permissions**

1. Ownership of files in UNIX can be viewed in one of three ways: owner (creator), group or others. Using this simple notion of ownership access to files can be controlled by associating unique user ID (UID) and group ID (GID) with twelve permission bits for each file as shown below.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Permission Bits* | | | | | | | | | | | |
| **Extra** | | | **Owner** | | | **Group** | | | **Others** | | |
| su | sg | t | r | w | x | r | w | x | r | w | x |

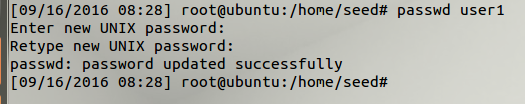
1. Typically these bits are divided into three sets of three bits and three extra bits as shown in table below. r, w and x bits stand for read, write and execute bits for each of the owner, group and others permissions. su, sg and t stand for set\_user\_id, set\_group\_id and sticky bits. These 4 sets of bits are often represented in their octal digits. For example, "100 111 101 101" is represented as "4755." When the su bit is set, whosoever executes the file, the UID of the process will be the owner of the file.

**Unix Lab Procedures**

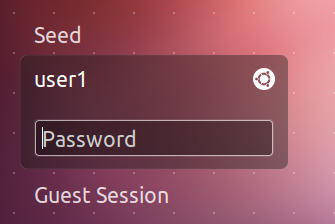
1. Setting up File Structure and User Space. The objective of this exercise is to setup the file hierarchy structure and the users that are required for the exercises in this section. The su command is used to switch users.
   1. Login as root
   2. Use useradd command to create two new users user1 and user2 as follows:
      * useradd user1 –m -g users
      * useradd user2 –m -g users



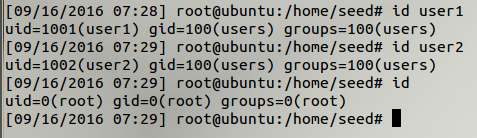
1. Change password (I use 123456).
   * + passwd user1 123456

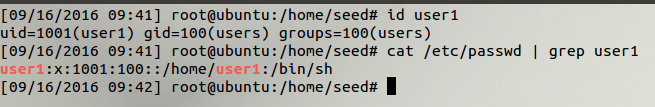


* + - You can login as user1 here

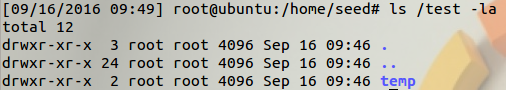


* 1. Check user information with the id command. Note the uid, gid for each output.
     + id user1
     + id user2
     + id





* 1. Create a directory structure
     + mkdir /test
     + mkdir /test/temp



* 1. Switch user roles as user1 and then back to root using the su command
     + whoami
     + su user1
     + su OR su root
  2. Create a new file as root user and change group ownership as well as user ownership of the file.
     + touch /home/user2/HelloWorld
     + ls -l /home/user2/HelloWorld (observe owner and group)



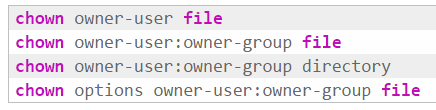
* + - chgrp users /home/user2/HelloWorld



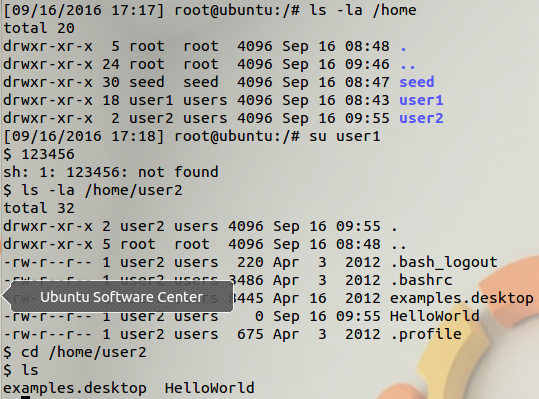
* + - chown user2:users /home/users/HelloWorld
    - ls -l /home/user2/HelloWorld (observe owner and group)



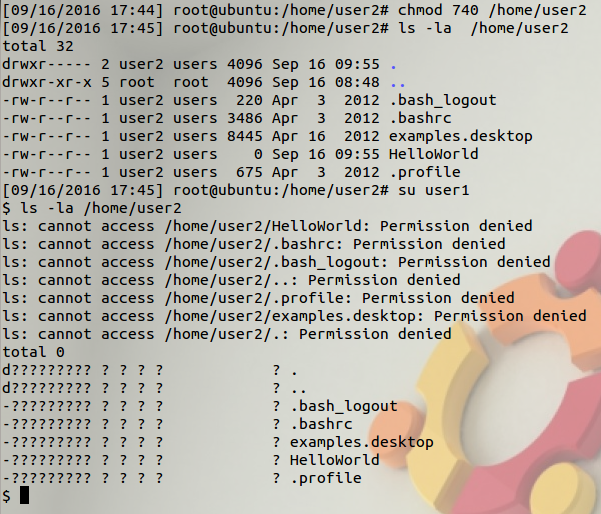
* + - Notes:



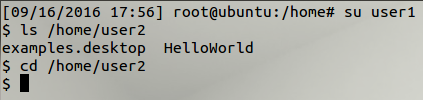
1. Questions.
   1. Explain what chgrp and chown do?
   2. What do -g and -m options mean?
2. Differences in File and Folder Permissions. The objective of the following exercises would be to see the differences in file and folder permissions. The chmod command will be used to change file and directory permission to demonstrate the slight differences in permissions for files and directories.
   1. Observe the result of ls and cd commands
      * cd /
      * ls -l
      * ls -la /home
      * Switch to user1 using su user1
      * ls -la /home/user2 (Can you list directory? Why?)
      * cd /home/user2 (Can you change directory? Why?)



* 1. Change directory permissions of user2 directory and try again as user1.
     + su root
     + chmod 740 /home/user2
     + Repeat steps written in RED in the previous example (Can you list or change directory of user2 if you are the user 1? Why)



* + - su root
    - chmod 750 /home/user2
    - Repeat steps RED in the previous example (Can you list or change directory? Why?)



* + - touch /home/user2/hello12.txt(Can you create new file? Why?)



* + - su root
    - chmod 770 /home/user2
    - su user1
    - touch /home/user2/hello12.txt (Can you create new file?)
    - ls -l /home/user2

1. Question. What are the directory permissions for user1, user2 and test directories?
2. Alternative Syntax for chmod Command. You are expected to learn both the ways to use chmod. The access permissions for the file hello.txt is to set the su bit only, allow all access permissions to owner, read and execute rights to the group and only read rights to others. In other works the 12 bit permission required on the file hello.txt is as follows: "100 111 101 100." This can be achieved in several ways using chmod command:
   1. chmod 4754 hello.txt
   2. chmod u+srwx g+rx o+r hello.txt
   3. chmod u=srwx, g=rx, o=r hello.txt
3. (optional) New Text Files and Linking Files. Unix supports two kinds of link files--a hard link and a symbolic link. A hard link is a file with the actual address space of some ordinary file's data blocks. A symbolic link is just a reference to another file. It contains the pathname to some other file.
   1. In the /test/temp/ directory, as root user, create a new text file ("hello") and fill it with some text using touch, pico, vi etc.
   2. Create a link link\_hello in the test folder pointing to hello.txt in the temp folder (refer to file structure in introduction)
      * cd /
      * ln -s /test/temp/hello /test/link\_hello
      * Is there any difference in file permissions of link\_hello and hello?
      * cat /test/link\_hello What is the output?
4. (optional) Default file permissions and Group Access Control. Whenever a new file is created using C program, default permissions can be assigned to it. UNIX system allows the user to filter out unwanted permissions by default. This default setting can be set by the user using the umask command. It is a system call that is also recognized by the shell. The command takes the permissions set during file creation and performs a bitwise AND to the bitwise negation of mask value. Some common umask values are 077 (only user has permissions), 022 (only owner can write), 002 (only owner and group members can write), etc.
   1. In a terminal window, make sure you are a root user. If not the root user, then switch back to root user (use your password to switch).
   2. Use umask command to check the current mask permission and assign a new mask.
      * umask
      * What is the current mask? How is it interpreted? (try umask -S or the man pages)
      * cd /test
      * touch testmask1
      * ls al
      * What are the permissions of the file testmask1
      * umask 0077
      * touch testmask2
      * Now what are the permissions of the file testmask2
   3. What is the effect of setting mask value to 0000?
5. (optional) Setuid Bit, Setgid Bit and Sticky Bit. As explained in the background above, the highest three bits of the permission bits represent the setuid bit, setgid bit and the sticky bit. If the setuid bit is set then the uid will always be set to the owner of the file during execution. If the setuid bit is not set then the uid will be the user who executes the process. Similarly, if the setgid bit is set then the gid will be set to the group that owns the file during execution. If the setgid bit is not set then the gid will be the group that executes the process. The sticky bit is set to keep processes in the main memory. In the following exercise, the objective is to demonstrate how processes are affected when the setuid bit is set. The exercise must be begun with root privileges.
   1. which touch
   2. ls -l /bin/touch
   3. chmod 4755 /bin/touch
   4. ls -l /bin/touch
   5. ls -l /home/user2
   6. chmod 700 /home/user2/HelloWorld
   7. ls -l /home/user2 (observe timestamp and permissions)
   8. su user1
   9. touch /home/user2/HelloWorld
   10. ls -l /home/user2 (observe timestamp)
   11. su root
   12. chmod 0755 /bin/touch
   13. su user1
   14. touch /home/user2/HelloWorld
6. (optional) Question. Why is permission denied?
7. Restore the System. After the series of exercises, it is most essential that the system is restored to its normal state so that other students may undertake the exercises again. Below are the series of commands that are expected to restore the system to its original form.
   1. su root
   2. umask 0022
   3. chmod 0755 /bin/touch
   4. userdel user1
   5. userdel user2
   6. rm -rf /home/user1
   7. rm -rf /home/user2
   8. rm -rf /test
   9. rm -rf /home/test/

Reference

<http://www.sis.pitt.edu/lersais/education/labs/access_control.php>

<http://www.tutorialspoint.com/unix_commands/adduser.htm>

**Extra Tutorial: useradd**

# **NAME**

useradd - create a new user or update default new user information

# **SYNOPSIS**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| **useradd** [*options*] *LOGIN*  **useradd** -D  **useradd** -D [*options*] | |

# **DESCRIPTION**

When invoked without the **-D** option, the **useradd** command creates a new user account using the values specified on the command line and the default values from the system. Depending on command line options, the useradd command will update system files and may also create the new user’s home directory and copy initial files. The version provided with Red Hat Linux will create a group for each user added to the system by default.

# **OPTIONS**

The options which apply to the **useradd** command are:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| **-c**, **--comment** *COMMENT* | |
|  | Any text string. It is generally a short description of the login, and is currently used as the field for the user’s full name. |
| **-b**, **--base-dir** *BASE\_DIR* | |
|  | The default base directory for the system if **-d** dir is not specified.*BASE\_DIR* is concatenated with the account name to define the home directory. If the **-m** option is not used, *BASE\_DIR* must exist. |
| **-d**, **--home** *HOME\_DIR* | |
|  | The new user will be created using *HOME\_DIR* as the value for the user’s login directory. The default is to append the *LOGIN* name to*BASE\_DIR* and use that as the login directory name. The directory*HOME\_DIR* does not have to exist but will not be created if it is missing. |
| **-e**, **--expiredate** *EXPIRE\_DATE* | |
|  | The date on which the user account will be disabled. The date is specified in the format *YYYY-MM-DD*. |
| **-f**, **--inactive** *INACTIVE* | |
|  | The number of days after a password expires until the account is permanently disabled. A value of 0 disables the account as soon as the password has expired, and a value of -1 disables the feature. The default value is -1. |
| **-g**, **--gid** *GROUP* | |
|  | The group name or number of the user’s initial login group. The group name must exist. A group number must refer to an already existing group. */etc/default/useradd*. |
| **-G**, **--groups** *GROUP1*[*,GROUP2,...*[*,GROUPN*]]] | |
|  | A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the **-g** option. The default is for the user to belong only to the initial group. |
| **-h**, **--help** | |
|  | Display help message and exit. |
| **-M** | The user’s home directory will not be created, even if the system wide settings from */etc/login.defs* is to create home dirs. |
| **-m**, **--create-home** | |
|  | The user’s home directory will be created if it does not exist. The files contained in *SKEL\_DIR* will be copied to the home directory if the **-k**option is used, otherwise the files contained in */etc/skel* will be used instead. Any directories contained in *SKEL\_DIR* or */etc/skel* will be created in the user’s home directory as well. The **-k** option is only valid in conjunction with the **-m** option. The default is to not create the directory and to not copy any files. |
| **-l** | Do not add the user to the last login log file. This is an option added by Red Hat. |
| **-n** | A group having the same name as the user being added to the system will be created by default. This option will turn off this Red Hat Linux specific behavior. When this option is used, users by default will be placed in whatever group is specified in */etc/default/useradd*. If no default group is defined, group 1 will be used. |
| **-K**, **--key** *KEY*=*VALUE* | |
|  | Overrides /etc/login.defs defaults (UID\_MIN, UID\_MAX, UMASK, PASS\_MAX\_DAYS and others).  Example: **-K***PASS\_MAX\_DAYS*=*-1* can be used when creating system account to turn off password ageing, even though system account has no password at all. Multiple **-K** options can be specified, e.g.: **-K***UID\_MIN*=*100* **-K***UID\_MAX*=*499*  Note: **-K***UID\_MIN*=*10*,*UID\_MAX*=*499* doesn’t work yet. |
| **-o**, **--non-unique** | |
|  | Allow the creation of a user account with a duplicate (non-unique) UID. |
| **-p**, **--password** *PASSWORD* | |
|  | The encrypted password, as returned by **crypt**(3). The default is to disable the account. |
| **-r** | This flag is used to create a system account. That is, a user with a UID lower than the value of UID\_MIN defined in */etc/login.defs* and whose password does not expire. Note that **useradd** will not create a home directory for such an user, regardless of the default setting in*/etc/login.defs*. You have to specify **-m** option if you want a home directory for a system account to be created. This is an option added by Red Hat |
| **-s**, **--shell** *SHELL* | |
|  | The name of the user’s login shell. The default is to leave this field blank, which causes the system to select the default login shell. |
| **-u**, **--uid** *UID* | |
|  | The numerical value of the user’s ID. This value must be unique, unless the **-o** option is used. The value must be non-negative. The default is to use the smallest ID value greater than 999 and greater than every other user. Values between 0 and 999 are typically reserved for system accounts. |
| **-Z**, **--selinux-user** *SEUSER* | |
|  | The SELinux user for the user’s login. The default is to leave this field blank, which causes the system to select the default SELinux user. |