CSE 5031 Operating Systems
2020/21 Fall Term

Project: 2 – Part 1
Topic: Linux User Management and File Attributes
Date: 23.10 - 02.11.2020

Objectives:

- Linux user and group management; identifying file attributes
- Using **GNU C Library API** for user and group databases access

References:

- **Red Hat Enterprise Linux-7 System Administrators Guide** (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/index.html)
- **The GNU C Library Reference Manual** (http://www.gnu.org/software/libc/manual/pdf/libc.pdf)

Section A. Linux Users and Groups

A.1 Users and Groups

An **OS** operates under the control of its **users** registered and identified by an **account** defining their personalities and operational realm. **Accounts** identify two category of users:

- **real users** who log in the system and perform open ended actions (root, admin, user1, xyz…), or
- **OS agents** that perform predefined tasks to run **system services** (mail, ftp, abrt, halt, etc.).

Users are associated at least to one **Group,** gathering users with a common purpose e.g. a project, a department, a function. When a **user account** is created **Linux** also creates a private **Group** with the same name and defines this **user** to be the **Group-owner**. Administrators can add new members to **Groups** or remove them.

OS maps **User** and **Group** names to **unique identification numbers,** the **user ID** (**UID**) and the **group ID** (**GID**), for conciseness and uniformity. Refer to **Linux 7 System Administration Guide section 4.1** for further details and learn about **Reserved User and Group IDs** on **Linux**.

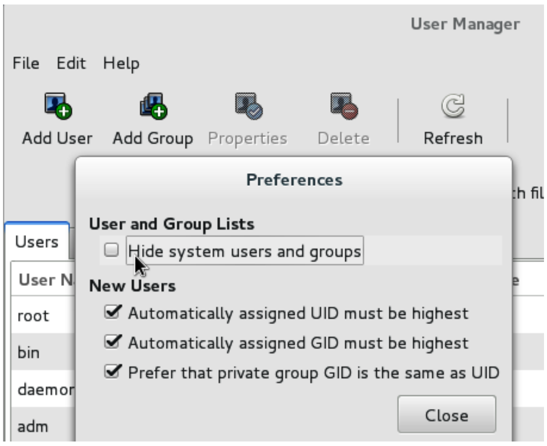
A.2 Creating a User Account

A user account may be created using either the **CLI** commands (**Linux 7 System Administration Guide section 4.3.1),** or the GUI application (**section 4.3.2)**.

i) Logon as the "**root**" user. Note that the logging screen does not display the root account to limit its misuse; select the "**not listed?**" option to logon.

ii) Start the user management GUI through the "**Applications** → **Sundry** → **Users and Groups**" path.

iii) Unhide system users & groups through "**Edit** → **Preferences**" path (left screen shut here after).



Preferences

User and Group Lists

☐ Hide system users and groups

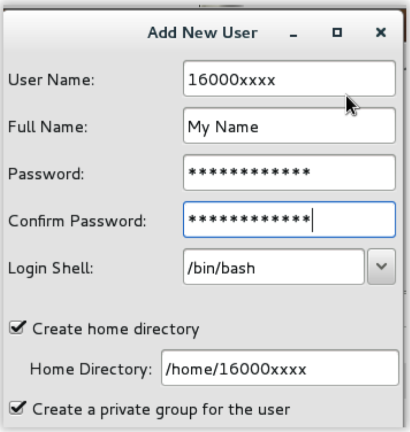
New Users

☑ Automatically assigned UID must be highest

☑ Automatically assigned GID must be highest

☑ Prefer that private group GID is the same as UID

Close



Add New User

User Name: 16000xxxx

Full Name: My Name

Password: ************

Confirm Password: ************

Login Shell: /bin/bash

☑ Create home directory

Home Directory: /home/16000xxxx

☑ Create a private group for the user

iv) Select “**Add User**” option to open “**Add New User**” menu (right screen shut above).

v) Enter your **“student-id”** as the **User Name** (account).

vi) Define “Full Name” and “Password” (choose a short password i.e. “cse”).

vii) Accept default parameters, then press “OK”.

viii) Select the **Groups** tab; identify your **group name** and its id.
Explain why **UID** and **GID** of **sysadmin** and **student-id** are respectively **1000** and **1001**?

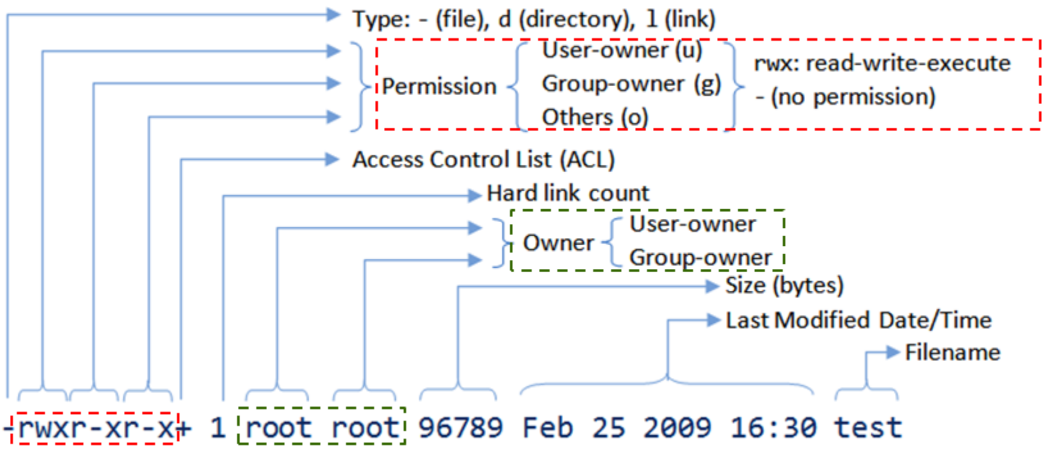
A.3 Access Rigths of the Files and Directories

User and **Group** identifiers define together access rights to **UNIX/Linux** system entities (files, directories, devices, processes, services etc.).

For instance, a user who creates a file/directory becomes the **owner** and the **group owner** of it; and owns special privileges. Following drawing depicts the attributes of the “**test**” file listed with the “ls -al test” command. Note that:

✓ the **root** is the file **owner** and the **group-owner**; and

✓ different **read, write, execute** rights are defined for the **owner**; the **group owners**, and the **rest of the users** in the system.



```
-rwxr-xr-x+ 1 root root 96789 Feb 25 2009 16:30 test
```

Given the **access rights** defined for of the “**test**” file:

✓ the **user root** may list the file (“**r**” bit set); update or delete the file (“**w**” bit set); run it (“**x**” bit set);

✓ **a user** in the **root group** (is there one?) may list the file (“**r**” bit set); run it (“**x**” bit set);

✓ **the rest** may list the file (“**r**” bit set); run it (“**x**” bit set).

Yet, to perform these operations the user should have the **access right** “**x**” of the directory containing the “**test**” file set for his category!

✓ **Note that**, the semantics of **directory** access rights are different than those of the files.

✓ “**x**” is the access control bit; if unset for the user he cannot perform a “**cd**” to this directory, or do an operation controlled by the other 2 attribute bits “**w**” and “**x**”;

✓ “**r**” read bit controls if directory entries can be listed by this user.

✓ “**w**” write bit controls the creation and deletion rights of directory entries.

Test the access rights by performing the following operation and explain your findings.

i) Logon as **student-id**; open a terminal window.

ii) List the home directory of **sysadmin** using “**ls –al /home/sysadmin**”
explain the reason of the “permission denial” error message.

iii) Display the “/etc/shadow” file using “**cat /etc/shadow**”
explain the reason of the “permission denial”.

A.4 Gaining Administrative Privileges

Access to secured **OS** files (i.e. /etc/shadow) and using **administrative commands** (i.e. mounting a volume, system shut down) require the privileges of the super user, the "**root**" account.

Working as the "**root**" or adding a user in the "**root**" **group** is **hazardous.** Any mistake may compromise the consistency of **OS** files and those of other users. Yet, there are instances when an ordinary user may require higher privilege levels, e.g. to mount a file system.

Ordinary users may gain "**root**" privileges temporarily by being member of a special group, the "**wheel**", which grants them the privilege to use the **substitute user** "**su**" or the "**sudo**" command. They may then perform tasks which would normally be available only to the root user. Refer to **System Administrators Guide** section **6** for details.

i) Adding "**student-id**" to the `**wheel**` group (section 6.1).

- ✓ Logon as "root".
- ✓ Use either "**Users and Groups**" GUI or the "**usermod –a G wheel studentid**" command to define "**studentId**" as a trusted user.
- ✓ Logout.

✓ **Note that** the **root** user is part of the **wheel** group by default.

ii) Gaining administrative privileges with the substitute user "**su account**" command.

A trusted user is allowed to logon in any account with "**su**". Its use without a specific account logs the user as the "**root**" user and grants him absolute administrative access to the system until he enters the "**exit**" command or **logs out**.

- ✓ Logon as "**student-id**".
- ✓ Enter "**su**" command; and observe the changes of the command prompt before and after the "**su**" command; try to explain what "**su**" command/program may have done to cause this result!
- ✓ List the **"/etc/shadow**" file.
- ✓ …execute other privileged operations of your choice …..
- ✓ enter "**exit**" command.

iii) Executing a single command with "**root**" privileges\, , the "**sudo**" command (section **6.2).**

Given the **hazardous** situation created by substituting an ordinary user as the **root** and allowing him to work with **root** privileges till he **exits**; **trusted users** are allowed to execute a single command with **root** privileges by preceding it with **sudo** command.

- ✓ List the **"/etc/shadow**" file using the "**sudo**" command prefix.

Another advantage of using the **sudo** prefix is that system administrators can define different user groups and grant them different access rights based on their needs (section **6.2).**

Section B. Sharing Folders between Host & Guest Systems

Users of a **Guest OS** can access files that are stored on the **Host** using **the pseudo-network** file system '**vboxsf**' installed with the **Guest Additions**. The '**pseudo-network**' prefix implies that '**vboxsf**' file system operates as if the **Host** and **Guest** systems are connected via a common network.

On the **Windows Host**, **file sharing** is implemented the same way you would use network shares; and the **Linux Guest** uses the **network file system** (**nfs**) to access **Host** folders. Note that access to **Host** system files does not require the presence of a physical network connecting the **Host** and the **Guests**; virtualization platform implements the network emulation (Oracle VM VirtualBox User Manual Section 4.3).

In this project, you will share **Host**'s folder of your choice e.g. "**CSE5031**" and create a permanent connection, one that survives across system boots.

✓ Note that, the **Guest OS Linux** shared folder is mounted on the **"sf_CSE5031"** directory located in the **FSH** under "**/media".** User accounts will require further configuration to access this share folder.

B.1 Configuring the Guest System for Accessing Host Folders

Perform the following procedures to configure the **Guest System** to mount shared Host folder on its file system.

i) Start **VirtualBox** and select the **Guest** (do not run it!).

ii) Open "**Shared Folders**" menu using "**Settings->Shared Folders**" from **VirtualBox Manager** menu bar.

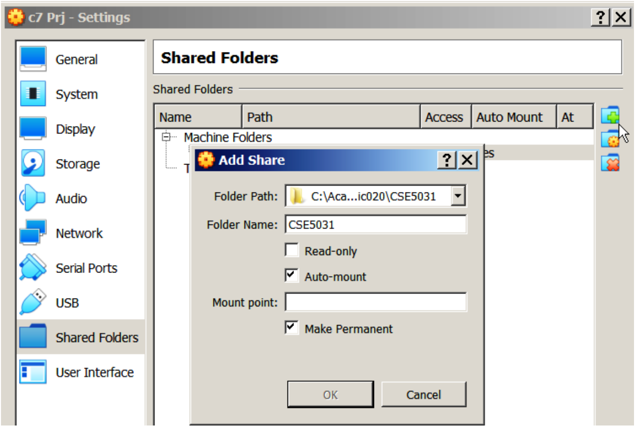
iii) Select "**Machine Folders**" entry on the right pane in the "**Folders List**", and click on the "**add folder**" **icon** at the upper left corner (left screen shut here after).

iv) From the "**Add Share**" menu:

- ✓ select the **Folder Path** "**C:\....\CSE5031**"; **Folder Name** will be automatically set as "**CSE5031**";
- ✓ select "**Auto-mount**" and "**Make Permanent**" options; enter "**OK**".

v) Start the **Guest**, and login as "**student-id**".

vi) Host folder's icon labeled as **sf_CSE5031** is displayed on the desktop (right screen shut here after). **Guest OS Linux** has mounted shared Host folder under its File System Hierarchy using this name.



c7 Prj - Settings

Shared Folders

Add Share

Folder Path: C:\Aca...ic020\CSE5031

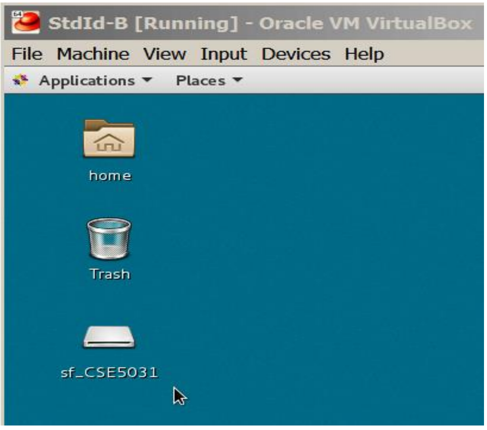
Folder Name: CSE5031

☐ Read-only

☑ Auto-mount

Mount point:

☑ Make Permanent



StdId-B [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications
Places
home
Trash
sf_CSE5031

B.2 Configuring Users of the Guest OS to Access the Shared Folder

Logon as **studentId**; click on the **Linux** desktop over shared folder icon "**sf_CSE5031**"; the **error message** "*you do not have the permissions necessary to view the contents of* ***sf_CSE5031*** will be displayed.

The **error** stems from the fact that **Linux mounted** the **Host folder** (make it accessible) in the Linux file system hierarchy, as the "**/media/sf_CSE5031**" directory and the user does not have necessary access rights to operate on this directory (refer to **section A.3)**.

i) Enter "**ls –al /media/**" command and examine the **attributes** of "**sf_CSE5031**" directory;

- ✓ check the access rights for the **owner** and the **group**; they should deny "**rwx**" rights for **others**;
- ✓ confirm that you are neither the owner of **sf_CSE5031"** directory; nor part of its owner-group.
- ✓ The solution is to add the account **studentId** in the "**vboxsf"**' group using:
  - ○ either "**Users and Groups**" GUI,
  - ○ or the "**usermod –a G vboxsf studentid**" command.
- ✓ Logout then login to let the **group** settings take effect; and click on the folder icon "**sf_CSE5031**" to display **Host OS** folder**.**
- ✓ You can use the file browser GUI on the Guest to open Host`s shared folder and drag and drop the file of your choosing.

✓ **Note that**

- ○ text files that are created on **Windows** with the **end-of-line** conventions (**cr+lf**) are processed by **Linux** text editors without problem;
- ○ whereas, text files under **Linux** use dıfferent **end-of-line** conventions (**nl**) and require reformatting on the Windows. Use for instance. **Notepad++** Edit->EOL conversion service to format text files.

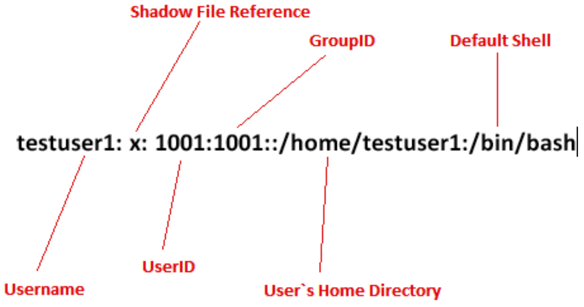
Section C. Accessing User and Group Databases using the GNU C Library API

C.1 User Accounts Repository "/etc/passwd"

User accounts repository is the **"/etc/passwd"** file; whose name is reminiscent from pioneering **UNIX** versions that stored account passwords in this file as well. Later on, as user population grew, security became an important issue and passwords were **encrypted** and stored in the **"/etc/shadow"** fıle.

The repository**"/etc/passwd"**:

- ✓ is a sequential **ASCII** file;
- ✓ contains **one** varying length record per account. terminated by **"\n"** (the new line character);
- ✓ records consist of **7** varying length fields, that are separated by a colon "**:**" character; essential account data stored in them is depicted over the drawing on the right.



C.2 Groups Repository "/etc/group"

Records of existing **Groups** along with the list of their members are stored in the **"/etc/group"** file.

- ✓ Use "**man group**" command to display the structure of this file.
- ✓ List **"/etc/group"** file and identify the fields and members of the "**wheel**" group.
- ✓ Use the "**id**" command to display the **UID** and **GID** of the current user.
- ✓ Use the "**groups**" command to list the **groups** of the current user belongs to

User and **Group** databases as well as any other **OS** files can be read and written by a program using input/output primitives. Yet, this approach requires detailed knowledge of the file's organization and may not be portable across **OS** variants. As such, it is advisable to use library functions to access them.

GNU C Library offers several functions that can be readily used to access and update **User** and **Group** databases. Refer to sections **30.13** and **30.14** of **The GNU C Library Reference Manual** to develop the programs that are specified in this section.

✓ Note that, **section 30.15** presents a **C** program example that illustrates the use of data structures and functions you may need to use.

C.3 Accessing Accounts Repository "/etc/passwd"

User database, kept in the **OS** file **"/etc/passwd"**, can be managed with the functions that are declared in "**pwd.h**" header file (section **30.13**).

Write a **C** program that:

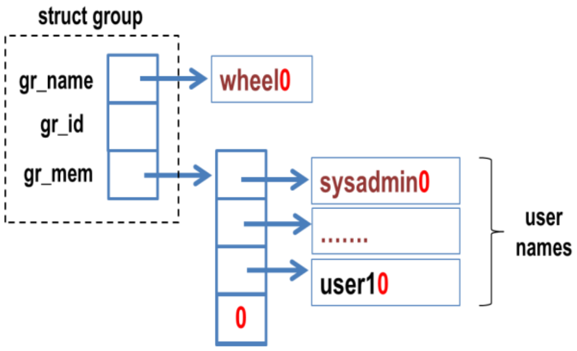
- ✓ reads or gets an account name as parameter;
- ✓ searches the **User database** for the account and if found displays on the standard output:
  - **user name**,
  - **UID**, **GID**
  - **home directory**.

Note that, your program should only define a pointer to the data structure "**passwd**", the memory for the structure will be allocated by the "**getpwent**" function.

C.4 Accessing Groups Repository "/etc/group"

Database of registered **Groups** kept in the **OS** file **"/etc/group"**, can be managed using the functions declared in the "**grp.h**" header file. Refer to **The GNU C Library Reference Manual** section **30.14**) for detailed explanation.

Note that, functions that handle **OS Groups** store the information of the file**"/etc/group"** records in the structure "**group**", described here after.



Third field "**gr_mem**" of the structure "**group**" defines a **pointer** to an **array of pointers** to user names strings; which is terminated by a **NULL** pointer.

*-mind that a similar data structure has been used to access the **Environment Variables** from a C program-.*

Note that **a private group** i.e. **student-id** is created with an **empty** member names list, and may not include thegroup owner (it is assumed to be part of). However, the list is updated as soon as other users are added as a member; and group owner is then added.

Write a **C** program that:

- ✓ reads or gets an account name as parameter;
- ✓ displays on the standard output: the name and **GID** of the groups to which this user is a member.

Hint. Use the "**getgrent**" function to scan all the entries of the Group database. This function works in a similar way to the "**getpwent**" function you have used in **C.3**.

Section D. Project Report

Perform the following operations as the user "**student-id**" to prepare your project report:

1. Run the **user database access** program developed in **C.3** and store its output in the file labeled as "**udb.txt**".
2. Run the **group database access** program you have developed in **C.4**, for the accounts "**admin**", "**student-id**"; store the output in the file labeled as "**gdb.txt**".
3. Name the **user database access** program as "**udb.c**" and the **group database access** program as "**gdb.c**", but submit them only if they produce correct results.
4. If the **programs** you have developed in **C.3** and **C.4** are compiling without error and operate as specified herein store the files:
   - **udb.c**" and **udb.txt**"
   - **gdb.c**" and **gdb.txt**"

   in the "**Prj2-Part1**" folder, located at the course web site under the tab **CSE5031 - OS Section -X/Assignment**; where "**X**" stands for (1,2,3,4) your laboratory session group.

Warning

You are encouraged to discuss the implementation procedures and general concepts behind the projects with your fellow students. However, **plagiarism is strictly forbidden**! Submitted report should be the result of **your personal work**!

Be advised that you are **accountable** of your submission not only for this project, but also for the mid-term, and final examinations. Your project grade may be reevaluated retrospectively, had you fail to answer correctly the same or a similar examination questions that you have solved with success in your submissions.