CSE 5031 Operating Systems
2020/21 Fall Term

Project: 1 – Part 1
Topics: Deploying & Testing the Virtualization Platform
Date: 05.10.2020 – 16.10.2020

Objectives:
- To introduce OS terminology on network identifiers and Virtualization platform.
- To deploy **Oracle VM VirtualBox** virtualization platform.
- To import a Virtual Machine appliance; and create a **Guest Linux VM.**

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)
- **Oracle VM VirtualBox User Manual** (Help->Contents or https://www.virtualbox.org/manual/UserManual.html)

Section A. Exploring Network Identifiers Associated with a Computer System

A.1 Network Identifiers of a Computer System

Computer systems connected to a network are **identified** with the addresses and labels associated to their network adapters. Each network adapter has:

- ✓ a **physical address**, the **MAC address**, a 48 bits value defining the adapter uniquely within the LAN segment; it is represented as **12 hexadecimal digits** separated by dashes o columns e.g. "**90-2B-34-77-57-D6**";
- ✓ a **CIDR IPv4 address** consisting of a 32 bits long "**IP address**" and its '**subnet mask**" pair; each component can be written in **dotted decimal form** e.g. the IP@ "**10.0.250.12**" and its subnet mask "**255.255.254.0**"; alternatively in UNIX like systems the mask is written as the number of bits set to \`1\` starting from the highest order digit e.g. "**10. 0.250.12 / 23**", where \`**/ 23**\` is the number of bits set to \`**1**\` in **255.255.254.0**" (8+8+7);
- ✓ optionally, a **label** defining system's name e.g. myhost, or a service name e.g. www, ftp, dns that is accessible though this adapter; a label qualified with a domain suffix e.g. \`***iku.edu.tr***\` forms a fully qualified domain name e.g. ***www.iku.edu.tr***. defining this adapter uniquely in the **Internet** domain.

A.2 Network Adapters

Computer systems may be equipped with physical (real) network adapters - **wired** and/or **wireless** - that are configured either by the user or the system. **OS** refers to a configured network adapter as a network "**connection**". Note that configuration changes yields in a new "**connection**" name.

In addition to **physical network adapters** operating systems may create several **virtual network adapters** including:

- ✓ "***Host-only***" Ethernet adapter created by the **ORACLE VM VirtualBox;** connects the **Host** (workstation) and virtual machines to a private **virtual LAN**; referred by VirtualBox as the "***Host-only Network".***
- ✓ **loopback adapter** (**127.0.0.1 /8**), created by the **TCP/IP** protocol stack connects the system to the local **virtual network** (**127.0.0.0 /8**). This legacy facility helped to test network applications on systems that were neither equipped with a network adapter, nor connected to a network.

Network adapters connected to a system are displayed either using the "**ipconfig**" command, through **OS GUIs**.

A.3 Displaying System & Network Adapter Configurations with the "ipconfig" command

i) Start **command line** interface "**cmd.exe**" with **administrative privileges -** in **Win'10** open power users menu by pressing "**Windows+X"**; then click on "**Command Prompt (Admin)**" or "**Power Shell (Admin)**" program.

ii) Enter the **...> ipconfig /all** command to list network configuration of your system and all of its adapters.
   - ✓ The first section displayed labelled "**Windows IP Configurations**" defines:
      - **Computer system name** (if any);

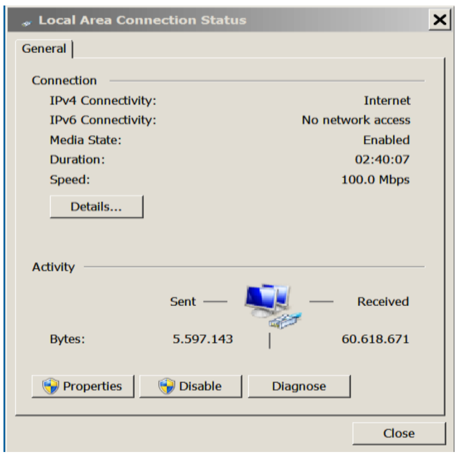
    - **domain name** and **domain suffix** information for the workstation (if any); and
    - "**IP Routing Enabled**" status; the "No" setting indicates that your system **does not** act as a **router** (it does not forward packets received by an adapter through its other adapters).
- ✓ Remaining sections display for <u>each</u> **adapter**:
    - the **MAC** address;
    - the **CIDR IPv4** address (address and its mask);
    - the **IP** address of the **Default Gateway** (if any);
    - the **IP** address the **DNS** and **DHCP** servers (if any).

A.4 <u>Displaying System & Network Adapter Configurations using Windows Control Panel</u>

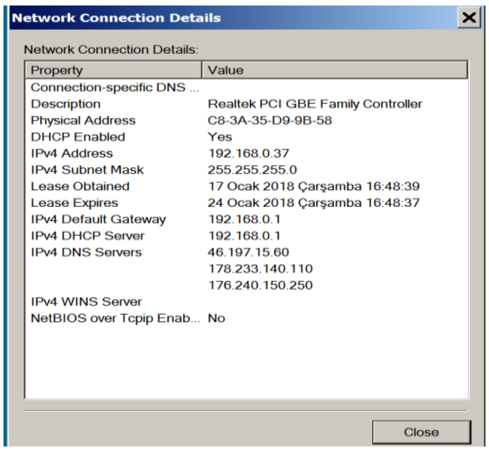
i) <u>Use</u> **Control Panel** to open **"Network and Sharing Center"**, and click on **"Change adapter settings"** tab on the left pane to **display** the "**Network Connections**" window; that displays all the adapters.

ii) <u>Click</u> on the "**Local Area Connection**" icon on the left to open "**Local Area Connection Status**" window (left screen-shut here after); and if the adapter is connected and active sent and received byte count is displayed.

iii) <u>Click</u> on the "**Details**" button to open the "**Network Connection Details**" windows (right screen-shut); compare displayed information against those displayed by the "**ipconfig /all**" command. Do they match?



```
Local Area Connection Status
General
Connection
    IPv4 Connectivity:                  Internet
    IPv6 Connectivity:         No network access
    Media State:                         Enabled
    Duration:                           02:40:07
    Speed:                            100.0 Mbps
    Details...
Activity
                     Sent —        — Received
    Bytes:          5.597.143        60.618.671
Properties   Disable   Diagnose
                                          Close
```



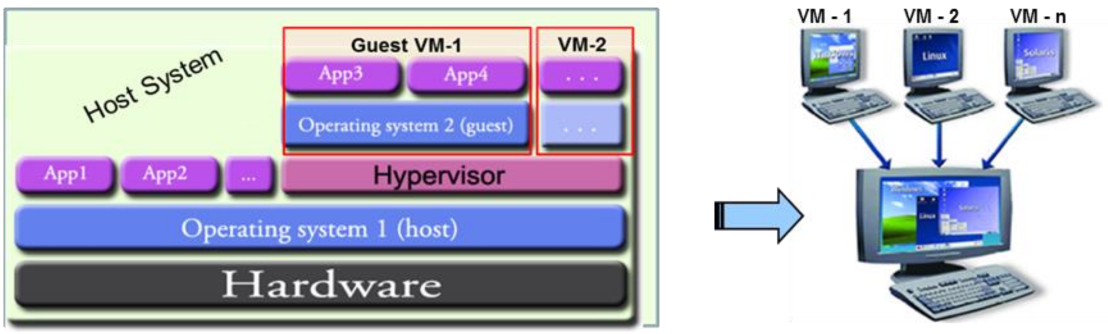
```
Network Connection Details
Network Connection Details:
Property                     Value
Connection-specific DNS ...
Description                  Realtek PCI GBE Family Controller
Physical Address             C8-3A-35-D9-9B-58
DHCP Enabled                 Yes
IPv4 Address                 192.168.0.37
IPv4 Subnet Mask             255.255.255.0
Lease Obtained               17 Ocak 2018 Çarşamba 16:48:39
Lease Expires                24 Ocak 2018 Çarşamba 16:48:37
IPv4 Default Gateway         192.168.0.1
IPv4 DHCP Server             192.168.0.1
IPv4 DNS Servers             46.197.15.60
                             178.233.140.110
                             176.240.150.250
IPv4 WINS Server
NetBIOS over Tcpip Enab...   No
                                               Close
```

Section B. Deploying and Configuring ORACLE VM VirtualBox

B.1 <u>Virtualization Platform Organization and VirtualBox Terminology</u>

The drawing here after depicts on the left a computer system <u>**hosting**</u> a virtualization hypervisor that controls two Guest Virtual Machines configured with their respective OS and applications. The <u>**guest**</u> virtual systems that are generated on this <u>**host**</u> are depicted on the right. Guests may be configured to connect to a **real** or a **virtual** **LAN.**

Your workstation running likely under MS **W'10** or similar <u>**native**</u> **OS** will be configured with **ORACLE VM Virtual Box**.



Virtualization terminology to be used in our projects is defined here after.

- <u>**Host**</u> (hardware). The workstation with the virtualization platform (the hypervisor).
- <u>**Host Operating System**</u>. The **OS** installed on the **Host** (**W'10**).

- **Hypervisor**. The virtualization platform: **ORACLE VM VirtualBox.**
- **Guest/Virtual Machine**. Virtual computer system created & controlled by the hypervisor (VirtualBox).
- **Guest Operating System**. The **OS** installed on a **Guest** e.g. **W'7**, **Linux** etc.
- **Host/Guest System.** A **Host/Guest** configured with an **OS.**

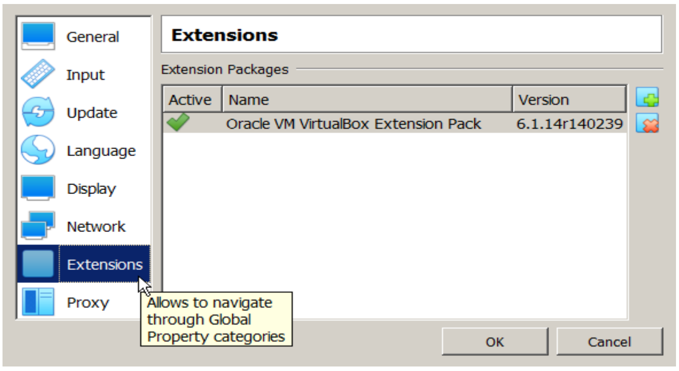
B.2 Installing VirtualBox

To run **VM**s with **64-bit** OSs **ORACLE VM Virtual Box** requires the support of **hardware virtualization extensions** such as Intel VT-x or AMD-V. Check your **BIOS/UEFI** settings of the Host system and enable them if they are not.

The latest version of **ORACLE VM Virtual Box** is **6.1.14**. Download and install it in a dedicated folder such as "**C:\VMs**". Do not forget to download and install the "**Guest Additions**" package associated with this version. Following the installation, confirm that the proper "**Guest Additions**" package is installed by:

✓ starting **Virtual Box** and using its Management screen; and opening the "Preferences" window (File->Preferences) shown on the right;

✓ selecting the "Extensions" entry to display the "**Guest Additions**" package version.



Active	Name	Version
✓	Oracle VM VirtualBox Extension Pack	6.1.14r140239

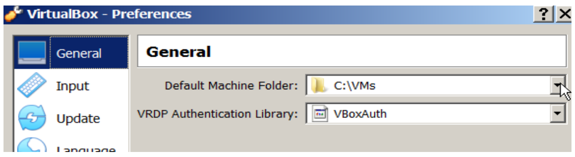
In case the box is empty or its version is not **6.1.14** download "**Guest Additions**" package and run it.

Failing to perform this operation successfully will prevent you from installing the **VirtualBox Guest Additions** package customizing the imported VM!

Define the directory where imported VMs will be installed.

✓ Open **Virtual Box** management screen.

✓ Select the "General" entry to define "**Default Machine Folder**" as shown in the screen shut on the right.



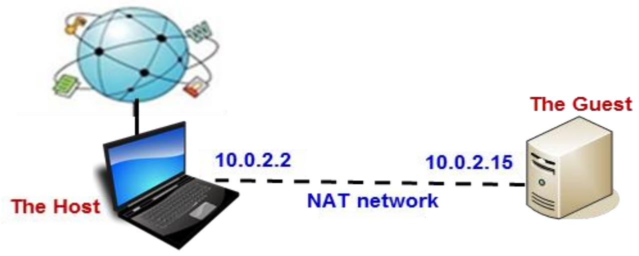
Default Machine Folder: C:\VMs
VRDP Authentication Library: VBoxAuth

✓**Virtual Box** installation will create on the host (your workstation) a **virtual network adapter**, the **Host only adapter**, and a **virtual bridge** "**VirtualBox NDIS6**" appliance that will enable the sharing of the real adapter by the **Guests**.

B.3 VirtualBox Alternatives for Connecting VMs to a Network

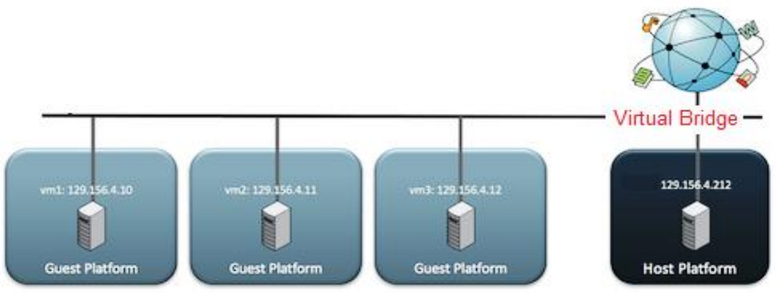
VirtualBox offers several alternatives to connect **VM**s to a **real** or **virtual** networks (refer to VirtualBox User Guide *Chapter 6. Virtual Networking*). The following sections summarize 2 of them that will be used in your projects.

i) **NAT (Network Address Translation)** is the simplest way of accessing an external network from a VM. It does not require any configuration on the host network and guest system.
A virtual machine with NAT enabled acts much like a real computer that connects to the **Internet** through a **router**. The router, in this case, is the **VirtualBox** networking engine, which maps traffic from and to the virtual machine transparently. In **VirtualBox** this router is placed between each virtual machine and the host. This separation maximizes security since by default VMs cannot talk to each other. Disadvantage of **NAT** mode is that, much like a private network behind a router, the **VM is invisible** and unreachable from the.



✓**Note**: Even though the NAT engine separates the VM from the host, the VM has access to the host's loopback interface and the network services running on it. The host's loopback interface is accessible as IP address **10.0.2.2**.

ii) **Bridged networking** connects a **Guest adapter** to the **physical LAN** to which the **Host** is connected. The **Guest** exchanges data packets directly with the external network, bypassing **Host OS** network stack, using a **virtual bridge** "**VirtualBox NDIS6**" driver (check its presence via **adapter properties** menu).



Section C. Importing and Testing a Guest System

C.1 Creating Guest Systems

There are 3 alternatives to create new **Guest Systems** (Hardware + OS) and add them to the virtualization platform.

i) **Create** a **Virtual Machine** (define its hardware configuration), then install its **Operating System** following the same procedures used in the installation of real systems.

ii) **Import** a **Guest System** (Hardware + OS) that has been created then exported. Import entity is referred to in the VirtualBox terminology as an "**appliance**" and stored using the "**ova**" file format file for our projects.

iii) **Clone** an existing **Guest System.**

In this project you will use the '**import'** method to add **VMs** to your **VirtualBox** testbed.

C.2 Characteristics of the first Guest Appliance

The **Guest** to import is the **CentOS 7 VM appliance "C7 PrjRef.ova",** an open source community distribution of **RedHat Enterprise Linux** **(RHEL) 7**.

The **VM** has been configured as a small server with:

- ✓ **1** CPU and**1 GB** memory
- ✓ **20 GB** hard disc
- ✓ **1** Network card attached to NAT network
- ✓ Pointing device: USB tablet

The OS **CentOS 7** has been installed as a server with the **Gnome GUI**; configured with:

- ✓ System administration tools.
- ✓ Development tool (gcc compiler and libraries).
- ✓ Firewalls connected to **trusted** networks.
- ✓ Security (selinux) set to **permissive** mode (issues just warnings).

C.3 Importing the Guest

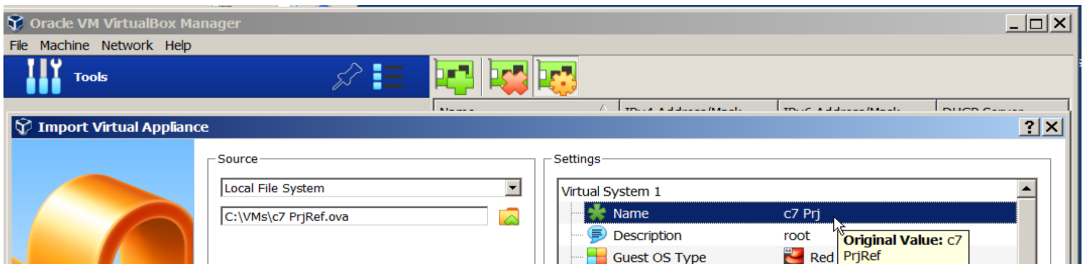
Apply the following steps to import your Guest.

i) Download the **Guest** appliance **"C7 PrjRef.ova", in** the work folder you have created (**C:\VMs**), from University **ftp server,** using the procedure defined in the "Download VMs" document that is stored in the course CATS site under **Resources/How to?** folder.
✓**Note** that, most of the **download errors** are related to password mistakes, and slow Internet downlinks.

ii) Start **VirtualBox** and open "**Import Virtual Appliance**" menu using "File->Import Appliance" path.

iii) Select **"C7 PrjRef.ova"** and name it **"C7 Prj"** as shown in the screen shut here after.

iv) Click the "**Import**" button.



Source: Local File System — C:\VMs\c7 PrjRef.ova

Settings: Virtual System 1
- Name: c7 Prj
- Description: root
- Guest OS Type: Red

Original Value: c7 PrjRef

v) At the completion of the import process, the project Guest is displayed on the left pane of the **VirtualBox Manager**; select it (do not run) and review the settings posted on the right pane.

✓ Display "**Description**" tab of the "**General**" section for the passwords of the **root** and **sysadmin** users.

✓ Check the "**Network**" section and make sure that:
→ Network Adapter 1 is **Enabled** and attached to a "**NAT**" type;
→ **Network cable** is connected (expand Advanced options).

C.4 Running the Guest

Apply the following steps to run your Guest.

i) Start the Guest from the menu bar using '**Start'** arrow, or the "**Machine->Start**" path.

ii) Double click on the Guest VM window; and login as '**sysadmin**'.
Guest OS (CentOS7) **GUI** is displayed within the **VirtualBox Manager** window; the mouse and the keyboard are trapped by the **Guest OS**.

iii) Sharing (capturing and releasing) the **keyboard** and the **mouse** between the Host and the Guest.

The **native Guest OS** (CentOS7) is not aware that it is running on a **virtual machine** and is sharing these resources with a **Host**. The **Host** and the **Guest** share the keyboard and the mouse as follows.

- **Host OS** (W' 10) tracks the mouse pointer and when it detects that a mouse click over a window that belongs to a **Guest**; it releases the control of these resources, notifies the hypervisor (**VirtualBox**) which in its turn passes the control of the keyboard and the mouse to the **Guest OS**. The host has learned to distinguish between its own windows and those of the guest when the hypervisor (**VirtualBox**) is installed.

- When the control of the **keyboard** and the **mouse** is delegated to the **Guest OS**, it proceeds with its native course of action **unaware** that it is sharing these **resources** with **another OS.** Therefore, when the mouse reaches the boundaries of the **Guest OS** screen -*which is a host OS window*- it does not release the pointing device if the Guest **VM** has been configured with the "**PS/2 mouse" option**! If configured with the "**USB Tablet**" option the mouse pointer across windows but the **keyboard remains locked in**!

- The **Host OS** (W'10) gets the control of the **keyboard** and the **mouse** back, when the user presses the "**host key**" (by default the '**right control'** key). This escape mechanism is created when **VirtualBox** is installed; it warns the hypervisor and transfers the control of the keyboard and the mouse to the **Host OS**.

  Note that following this operation, the mouse pointer is still displayed at the **Guest OS** window, and the **Host OS** displays its own; you will see two mouse pointers at once!

C.5 Customizing the Guest

To implement seamless resource sharing between the **Host** and the **Guest** we need to customize the **Guest OS** and instruct it on the sharing process. Perform the following to install **VirtualBox Guest Additions**:

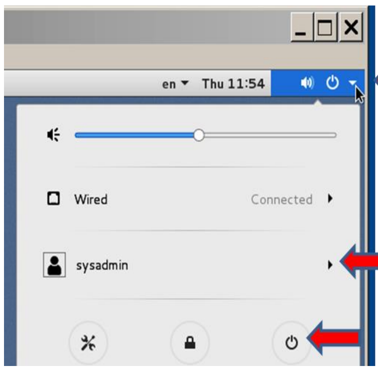
i) Login to the Guest as **sysadmin**.

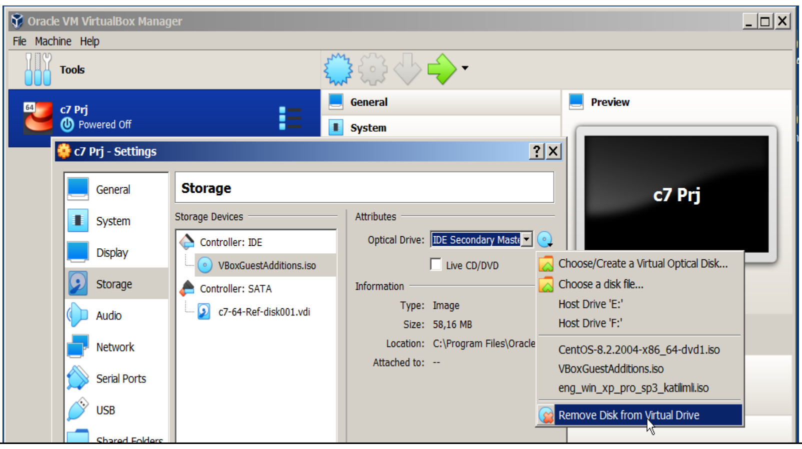
ii) Select from **VirtualBox Manager** Menu bar "**Devices->Insert Guest Additions Cd Image**" path.

iii) **Run** the application and enter password; then finish the installation by entering '**return'.**

iv) Shutdown the **VM** by clicking on the 'circle' at the bottom (left screenshot here after).

v) Remove the '**Guest Additions' CD** by selecting from the **VirtualBox Manager** menu bar "**Devices->Optical Devices-> Remove disk..** " path (right screen shot here after); then restart it.

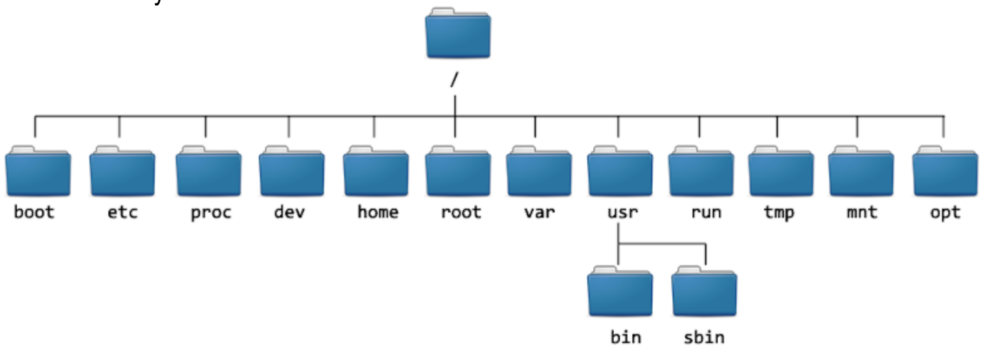




Section D. Linux File System and Directories

D.1 Linux File System Organization

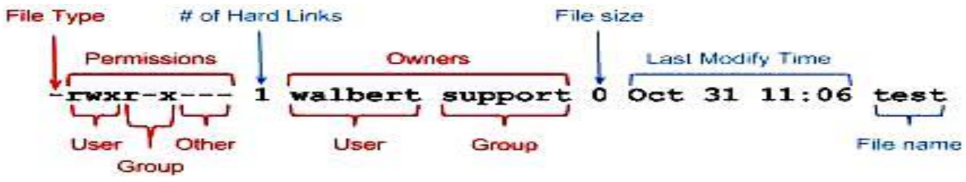
CentOS 7 operating system, as any **UNIX** descendant, stores the OS user files in a hierarchical directory structure organized as a tree that starts at "/". **File System Hierarchy Standards** documented by ***The Linux Foundation LSD Workgroup*** (https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf) consists of a set of requirements and guidelines for file and directory placement under UNIX-like operating systems. *FSH* guidelines are intended to support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems.



When you have defined an account e.g. "**sysadmin**", a home directory **with the same** name is created under "**/ home**".
✓As you logon using the " **sysadmin**" account, the **OS** will set the **current directory** the your home directory, and initialize your **environment variables** such as "**USER**", "**HOME**" etc. with corresponding values.

D.2 User Home Directory and Environment

- ✓ Logon with the "**sysadmin**" account; and open a terminal window using the path **Applications -> Favorites -> Terminal** (note that a graphical short cut is also available at the left boundary of the GUI).
- ✓ Display the current directory associated with the terminal using the "**pwd**" (print working directory) command.
- ✓ Display the current directory contents using the "**ll**" (long list) command. Listed entries should consists of the file name (last item) and the file attribute fields as shown here after.



✓ Note that, your **home directory** contains several files and directories that are defined at **OS** generation time by the choice of the command shell (**bash**) and the **OS GUI** (**GNOME**). The contents and the use of most of them are beyond the scope of this course and fall in the area of interest of System Administration.

- ✓ Find the location of the shell program that processes your commands using the "**which bash**" command.
- ✓ Display selected environment variables with "**echo $HOME and $USER**"; or get a complete list with "**env**".

D.3 Creating a Subdirectory and Specifying Paths in Linux FSH

You are strongly advised to create a separate subdirectory to implement each of your projects to avoid mishaps.

- ✓ Open a terminal window; and enter the command "**mkdir prj1**".
- ✓ Set your current working directory to **prj1** using the command "**cd prj1**".
- ✓ Create the empty file named "**test.c**" under Prj1 using the command "**touch test.c**".
- ✓ Display the current directory (home directory) using the "**ls –al**". The output should contain the **touch.c** file and two more definitions:
    - single dot "**.**"stands for the FSH path for the current directory root (**/home/sysadmin/prj1**); and
    - double dots "**..**" stands for the FSH path of the current directory's parent directory **/home/sysadmin**.

  For instance you may specify the path of the **test.c** file:
    - as an **absolute path** starting from the root "**/home/sysadmin/prj1/test.c**", or
    - as a relative path to the current working directory "**./test.c**" ; in this case the dot "**.**" is used as the substitute of the path string "**/home/sysadmin /prj1**"
- ✓ Use the command "**ll ..**" to display the directory "**/home/sysadmin**".
  (Note that double dots are mainly used in scripts which are programs formed by a set of commands.

Section E. The C Program – Operating System Interface

E.1 C Program Arguments and Environment Variables

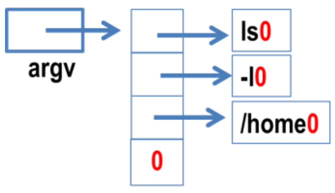
A command shell e.g. **bash** reads the command line, such as "**ls -l /home**"; locates and loads the executable file ("**ls**"); then calls the function **main** defined in the "**ls**" program, and passes the arguments, **"-l"** and "**/home**" to the **main** function using the data structures and formats defined here after.

In **ISO C** the **main** function can be defined **with no arguments**, in which case you have access to the arguments, or you may define **main** with **two arguments**, using the syntax:

```
int main ( int argc,  char **argv )
```

where:

- argc provides the count of the tokens in the command line; argc=3 for the above command (command + 2 arguments)
- argv is a **pointer**, to an array of **character pointers** terminated by a **NULL** pointer, as shown on the right; each array entry is the address of one token string, starting from the executable "**ls0**"



When a program is executed, it also receives information about the context in which the **process** running it operates via **Environment Variables** (e.g. user name and its home directory, the path to executables etc.), In **UNIX/Linux** the environment variables can be accessed by defining the **main** function with a **3d argument**, using the syntax:

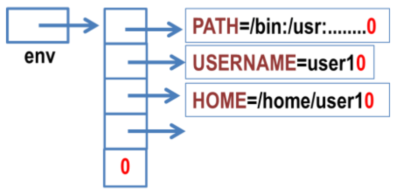
```
int main ( int argc,  char **argv,  char **env )
```

where:

- argc and argv have the above definitions;
- env is a **pointer**, to an array of **character pointers** terminated by a **NULL** pointer as shown on the right; and each array entry is the address of string defining an environment variable using the format:

  Environment Variable Name=the value string

  e.g. **USER=**user10



E.2 Accessing C Program Arguments and Environment Variables

C is a purposely limited programming language defined to implement **UNIX**. It **does not have** I/O statements, nor define **composite data types** such as "string", complex numbers.
These functionalities are provided with the help of platform independent **Standard C** libraries e.g. **stdlib.h**, **stdio.h** etc.
C programs may also use **platform dependent** system libraries (e.g. **unistd.h**) to access to **OS** services and for higher performance or extended functionality.

✓ The "**glibc**" **GNU C library** documented in **The GNU C Library Reference Manual**" supports all these libraries.

The following **C** program displays the values of the **Environment Variables** given as program arguments:

" ./myx USER=" or "./myx HOME= PATH=" or "./myx USER= HOME= PATH=" etc.

Analyze the program carefully, using the corresponding data structures defined above.
✓Try for instance how "**char ****" definition relates with them? Also note that environment variable arguments an appended with an equal sign "**=**" e.g. "USER**=**" rather than "USER" to shorten their search code in the program.

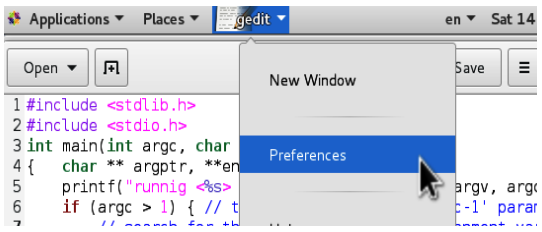
```
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char **argv, char **env)
{
    char ** argptr, **envptr;
    printf("running <%s> with %d parameter\n ", *argv, argc-1);
    if (argc > 1) { // the command line has 'argc-1' parameters
            for (argptr = argv+1; *argptr != NULL; argptr++) {
                    printf("searching %s in the Environment Vector\n", *argptr);
                    for (envptr = env; *envptr != NULL; envptr++){
                            // compare'parameter=' with Environment Vector Entries
                            if (memcmp(*argptr,*envptr, strlen(*argptr))== 0){
                                    printf("%s\n ", *envptr);
                                    break;
                            }
                    }
            }
    }
    return 0;
}
```

E.3 Compiling and Linking a C program with the glibc library

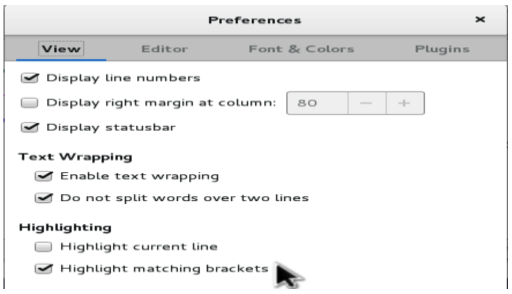
- ✓ Open a terminal window using the path "Applications->Favorites->Terminal".
- ✓ Set your current working directory to **prj1**.
- ✓ Start to edit the "**test.c**" file by lunching the text editor "**gedit**" as a separate process using the command:

  gedit test.c &

  The ampersand "**&**" character located at the end of the command line instructs the **OS** to run "**gedit**" as a separate process and to use a new window; you will use the first window to enter further commands.
- ✓ Open the project (pdf) document in a browser; copy and paste the **C** program listed above to "**test.c**".
- ✓ Open **gedit** preferences from the menu bar (left screen shut); set the options: "**Display line numbers**" and "**Highlight matching brackets**" as shown on the right.



```
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char
{   char ** argptr, **en
    printf("runnig <%s>                  argv, argc
    if (argc > 1) { // t                 c-1' parar
```



Preferences

View | Editor | Font & Colors | Plugins

- ☑ Display line numbers
- ☐ Display right margin at column: 80
- ☑ Display statusbar

Text Wrapping
- ☑ Enable text wrapping
- ☑ Do not split words over two lines

Highlighting
- ☐ Highlight current line
- ☑ Highlight matching brackets

- ✓ Save the "**test.c**" file.

- ✓ To compile and link "**test.c**", and generate the executable object program named "**myx**" (option -o) enter:
  gcc test.c -o myx
- ✓ Display **prj1** using **ls -al** and identify that the executable "**myx**" is created with proper **execution rights**.

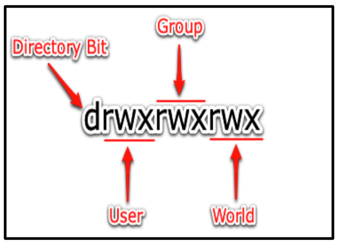
E.4 Running an Executable Object Program

UNIX/Linux and **Windows** differ at two instances: (a) they identify executable files using **different attributes**; (b) they locate them by applying **different search policies**.

✓**Windows identifies** executables by their suffix (exe, com etc.), **Linux** uses the "**File Permissions**" bits that define if a file content can be executed by its owner, by users that belong to a group, or by anyone.

"**myx**" permission bits show that it is an executable ("**x**" bit set) by its owner, or by users that belong to a group, or by anyone.

```
-rwxrwxr-x 1 Std-Id Std-Id 8568 Sep  8 15:48 myx
-rw-rw-r-- 1 Std-Id Std-Id  207 Sep  8 15:38 test.c
```



✓ Note that, to execute a program, the user should not only have the file`s "**x**" bit set, but also should have the **access rights** to the directory that contains the executable file – directory`s "**x**" bit set for the user/group it belongs to-.

✓ Unlike **Windows** OSs, **Linux** does not search the executable file **in the current working directory first**!! Instead, it tries to locate the file, by searching its name in a series of directory paths defined in the environment variable "**PATH**" (directory paths are aggregated by semi columns "**;**" display them using **echo $PATH).**

✓ Note that the file name is searched in sequence in the directories defined in "**PATH**". The first matching file name is run. In case several directories contain the same file, the first one in the "**PATH**" sequence is executed.

You can specify the path to run "**myx**" in three ways:

i) by defining its full path starting from the root of the **FHS** as **"/home/sysadmin/prj1/myx**"; or
ii) by appending the path of the directory **"/home/ sysadmin /prj1/**" to the "**PATH**" environment variable; then just entering "**myx**"; or
iii) if the current working directory is **"/home/ sysadmin /prj1/",** by just entering its relative path **"./myx**" (refer to the procedure defined in Section II.2).

✓ Run "**myx**" with different parameters e.g. " **./myx HOME=** " and "**./myx USER= HOME= PATH=" etc.**

Section F. Project Report

F.1 C Program Development

Extend the **C** program provided in **Section E.2** using the "**glibc**" functions defined in **GNU C Library Reference Manual**.

- + Replace the Environment Variable search sequence in the "**env**" data structures, by a call to "**getenv**" function (section **25.4**), and test it with 1, 2,3 environment variables.
- + Display using "**uname"** function (section **31.2)** :the name of the **operating system** and its **release; and**the description of the hardware (machine).
- + Add at the beginning of your **C** program a comment line consisting of **your name** and **student-id.**

F.2 Project Report Submission

If the **code** you developed in **Section F.1** is partially or completely **operational** and if it is compiling without errors store it in the "**Prj1-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 lab. session group you are registered in.

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.