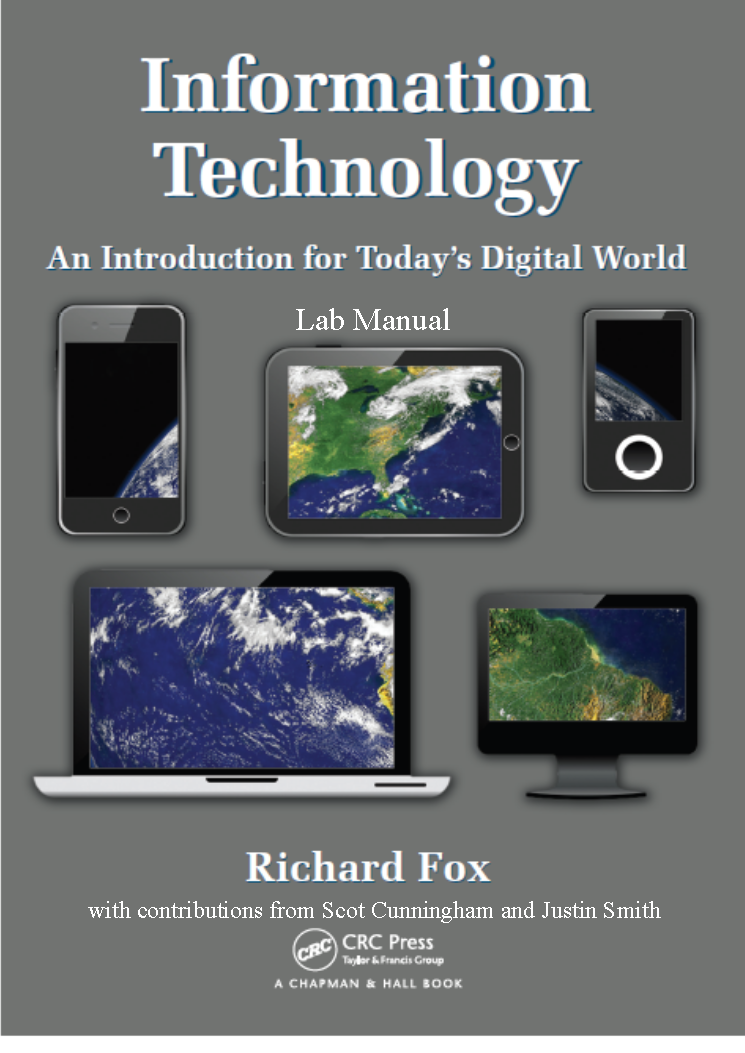
****

About this lab manual

This document is being prepared as an ancillary for the Information Technology: A View of our Digital World textbook. The labs in this document were written for an introductory computer information technology course. The labs primarily cover various aspects of Windows 7 and Red Hat Linux (CentOS). Depending upon your setup, you may or may not be able to use the labs as is and in some cases you might have to perform some substantial revisions. Below are a list of expectations along with steps to accomplish them.

Lab 1: the first lab is a computer assembly lab. You are expected for each group of students to have an empty system unit “shell”, a functioning motherboard, CPU, memory circuit board, a hard disk drive, a power supply unit, standoffs and screws, and appropriate tools. Optional is a CD or DVD player. The hard disk is expected to be of IDE format and the CD/DVD player of SATA format. If you do not have these components or the ability to perform this lab, one option is to demonstrate the assembly of the PC in class and then have the students answer the questions with respect to the component parts and what they saw. As lab 2 involves installing operating systems, installing the Windows operating system has been omitted from this lab but could be included if desired. To accomplish this step, you would need a Windows installation disc and an optical drive (this lab references Windows 10).

Lab 2: this operating system installation lab assumes that you will be using virtual machines. While you could instead have students perform installation on “fresh” machines, using virtual machines allows them to have both a Linux and Windows installation available throughout the semester. The next page provides details on installing virtual machine software and usage. You will need Linux and Windows installation CDs. You can download free copies of Red Hat Linux CentOS from the CentOS website. Instructions on how to do this are provided on the next page. If you do not have extra Windows 7 installation discs available and plan to use already existing machines rather than virtual machines, be warned that some of the steps in these labs assumes that the student is the Administrator and therefore you may have to modify some of these steps. NOTE: you will need approximately 33 GB of hard disk space per student to set up the virtual machines. The recommended size for Windows 10 is 25 GB and the recommended size for Red Hat Linux is 8 GB. If you use “Dynamically allocated”, your VM will probably not take up the maximum amount of space and so will be more efficient.

Labs 3-14: these labs all use either or both of the Windows and Linux VMs. We make no assumptions of software being installed beyond the basic installation that comes with both operating systems. Students will perform lab steps as normal users and as system administrators and so they will need access to the administrator (root) account.

Lab 15: this lab is a discussion that involves the class discussing topics covered throughout the semester but emphasizing material from chapter 16 and in particular, computer ethics.

Each lab involves completing steps and answering questions. Aside from the computer assembly lab, there will be no need to check your student’s work, instead answers to the questions will suffice.

A note to the instructor:

Completion times are estimates based on experience from the course at Northern Kentucky University. Based on your own setup, you may find that times are vastly different. You should work through each lab to determine what material you might retain and what material you might skip.

Lab 2 is the operating system installation. It is assumed that you will use VirtualBox from Oracle and that you will have available installation CDs of both Windows 10 and Red Hat Linux CentOS 6. If not, you will have to modify the lab to state otherwise.

A note to the student:

In order to complete each lab in a timely fashion, you should prepare for the lab in advance by reading the lab and any corresponding material from the textbook.

In labs 1, 2 and 15, perform the steps as specified. Questions appear at the end of the lab. In labs 3-14, questions appear with the steps you are performing.

In labs 2-14, questions appear in *italic font face*. Specific command whether through the GUI or the command line will appear in **bold face font**. In such cases, pressing <enter> after the command is assumed. For instance, when told “type **pwd**”, it means to type “pwd” and press <enter>.

While this lab manual covers Windows 10, the textbook refers to Windows 7 (this lab manual was revised in 2016 to update to Windows 10). Many of the administrator controls in Windows 10 are the same as in Windows 7 so it should not be very challenging moving from 7 to 10.

Table of Contents

[Installing and using virtual machines 4](#_Toc346109411)

[Lab 1: Computer Assembly 6](#_Toc346109412)

[Lab 2: Operating System Installation 13](#_Toc346109413)

[Lab 3: Files 18](#_Toc346109414)

[Lab 4: File Systems 23](#_Toc346109415)

[Lab 5 Managing Users and Groups 27](#_Toc346109416)

[Lab 6 File Permissions 32](#_Toc346109417)

[Lab 7: The Bash Shell 36](#_Toc346109418)

[Lab 8: vi in Linux 42](#_Toc346109419)

[Lab 9: Regular Expressions 46](#_Toc346109420)

[Lab 10: Managing Processes 50](#_Toc346109421)

[Lab 11: Managing Services 56](#_Toc346109422)

[Lab 12: Networking 61](#_Toc346109423)

[Lab 13: Software Management 65](#_Toc346109424)

[Lab 14: Shell Scripting 70](#_Toc346109425)

[Lab 15: Case Study 75](#_Toc346109426)

# Installing and using virtual machines

There are several sources of virtual machine software available. VMWare (www.vmware.com) allows you to install VMs on individual clients or to have a VM server (or more) that can be accessed over the Internet where the VMs are stored on the server and access can be made from any location where Internet access is available. In the former case, you must install VMWare software on each client computer and then install VMs onto each client. With VM server, known as vSphere server, all VMs are held on the server and clients need only install vSphere client, or access the VMs using a web browser (only some web browsers are currently supported). Most but not all VM software will require that you purchase licenses although vSphere client is free. Two free products are VMWare Workstation and VMWare Player, both available for a limited free trial period or for purchase.

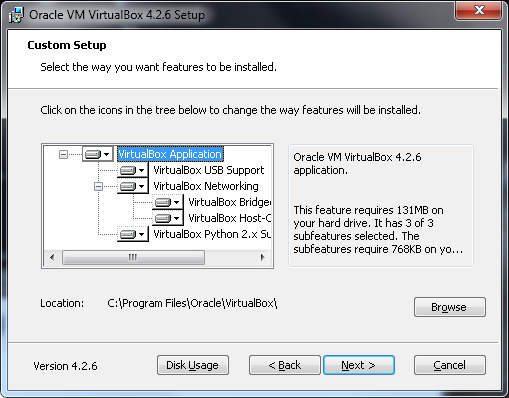
Another option is to create VMs in a corporation’s cloud. Amazon Cloud Servers (AWS) provides access to VMs. AWS usually offers some number of free hours of access for universities although it will probably be too limited for this course.

The best alternate is the open source product Virtual Box, available from Oracle (www.oracle.com). Virtual Box was developed and owned by Sun Microsystems (which has since been purchased by Oracle). You can download free versions of Virtual Box from the Oracle site or directly from www.virtualbox.org, which includes versions for Windows, Mac OS X and Linux.

Here, we briefly provide instructions for downloading and installing VirtualBox in Windows. To install VirtualBox, double click on the installation program and permit the program to run on your computer (if asked). This brings up the setup wizard:



Click next. The next window allows you to perform a custom installation. It will probably look like this:



Here, you can select the application(s) to install and the location of the installation. Unless you wish to change the defaults, just click Next. You will be asked whether to create a shortcut icon on either the desktop or quick launch bar. Select as you desire and click Next. You may receive a network interfaces warning about VirtualBox temporarily resetting your network connection. This is only temporary, so click Yes to proceed with the installation. Then confirm the Installation. The installation may take a few minutes. During the course of installation, you will be told about VirtualBox installing new device drivers. Clicking on the “always trust Oracle” will allow the installation to proceed without any further interruptions. Finally, click on Finish and start VirtualBox. The initial VirtualBox image is a welcome message. You will want to install your virtual machines from this initial windows using the New button. See lab 2 which covers installing both Windows 10 and CentOS 6 in VirtualBox.



# Lab 1: Computer Assembly

Goals: to learn about computer components and assemble them into a working computer.

Introduction: You will construct a PC from component parts. This is a lengthy lab and will take between 2 and 4 hours to complete. It is recommended that students work in groups of 2 (or 3) for assembly but answer questions on their own. This lab requires that you have the following components: System unit, power supply, motherboard, CPU, memory circuit board(s), hard disk drive, optical disc drive (optional), standoffs, screws, thermal heat paste, tools (screwdriver, etc), rails for disk drive(s) (if needed).

Estimated time to completion: 2-4 hours

Before starting, review chapter 2 Computer Hardware and Computer Assembly.

1. Identify the components you will be working with:
   * + - System unit
       - Motherboard
       - CPU
       - Memory circuit board(s)
       - Power supply
       - Hard disk drive
       - Optical disc drive
       - Supplies (standoffs, screws, rails for disk drive(s), tools, thermal heat paste)
2. Unpack the power supply and identify:
   * + - ATX (Main) power connector
       - Auxiliary 4-pin CPU power connector
       - IDE (Legacy) power connectors
       - SATA power connectors
3. Remove the side of the system unit and mount the power supply in the system unit using the bolts provided.
4. Remove the front bezel of the case.

Steps 5-6 are optional depending on whether you are inserting a DVD/CD drive

1. Snap a set of large plastic rails onto the DVD drive (if needed, depending on the type of system unit shell, you may or may not need the rails), with the release tabs towards the front, and insert it into its slot from the front. Don't connect the cables yet.
2. Remove the plastic cover from the front panel bezel corresponding to the location where you will insert the DVD drive. Give the cover to your instructor. Replace the front bezel onto the system unit.
3. Snap a pair of small plastic rails on the SATA hard drive (if needed) and install the hard disk drive in the system unit. Don't connect the cables yet.
4. Put on your grounding wrist strap and connect the grounding end to the metal of the system unit. Now, unpack the motherboard.
   * + - Determine whether the motherboard connector layout matches the connector plate that comes with the case. If not, remove the connector plate and replace with the one supplied by the motherboard manufacturer.
       - Find the standoff holes in the motherboard and match them to a set of standoff holes in the case. Once the correct holes are identified, screw in the standoffs by hand or with a nut driver (do not overly tighten the standoffs).
       - Using the motherboard pin-out diagram, identify:

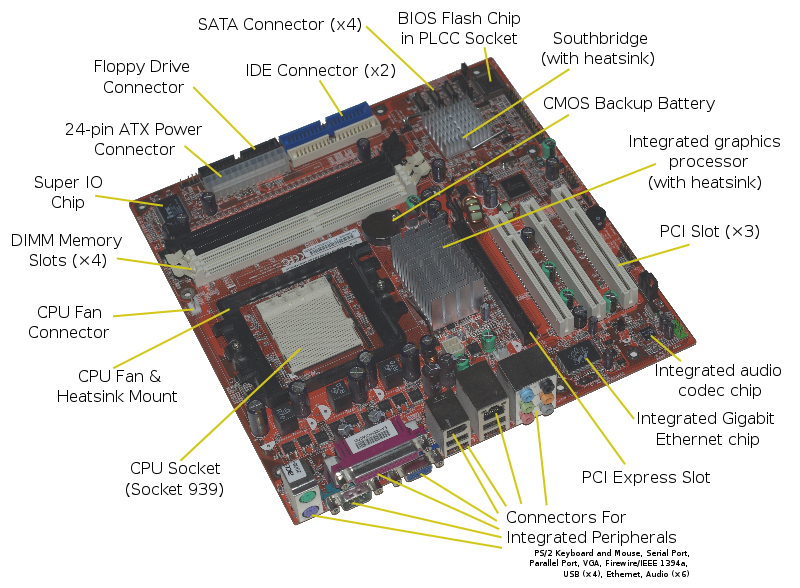
1. The front panel connectors.

2. The SATA, IDE and floppy connectors.

3. The power connector for the CPU fan.

4. The power connector for the case fan.

1. Install the CPU in its socket on the motherboard. Make sure your lab instructor verifies your installation before you lock it down with the locking lever.
2. Before installing the CPU cooling assembly, add a dab of thermal transfer paste to the center of the heat spreader on the CPU.
3. Position the CPU cooling assembly so that the power cable can connect to the power supply on the motherboard. Once you have it in its proper orientation, install the CPU cooling assembly. Lock it down to the motherboard. Connect the cooling assembly’s power cable to the motherboard. The easiest way to lock down the cooling unit is to turn the latches of the cooling unit to the locked position, and push through the motherboard until you hear a click.
4. Install the system memory on the motherboard by opening the levers on each side of the memory slot, sliding the memory board into position until it clicks in place, and closing the levers.
5. Install the motherboard in the system unit, using 6 small bolts.
6. Connect the front panel power switch, reset switch, power LED and HDD activity cables to the appropriate places on the motherboard. Use the motherboard diagram for reference.
7. Find a SATA cable from the motherboard box, and connect it to the hard drive and to the motherboard. Find a SATA power connector from the power supply and plug it in to the socket on the hard drive.
8. Connect the data (IDE) cable for the DVD drive into the motherboard and then connect the power cable into power supply unit.
9. Connect the case fan power cable to an appropriate supply cable from the power supply. Connect the case fan controller lead to the appropriate connector on the motherboard.
10. Make sure the main power switch on the back of the power supply is in the off (O) position. Connect the main ATX power connectors to the motherboard. Also make sure that the CPU auxiliary 4-pin connector is plugged into the motherboard.
11. Recheck all cables, making sure the connectors are seated properly. Pay particular attention to the case fan power connector and the CPU fan power connector. Recheck the RAM sockets, making sure the modules are still locked in place.
12. With the side of the system unit still open, turn on the main power switch in the back, and press the power button in the front of the system unit. See if the computer boots. Also make sure that the CPU fan and the case fan are both operational. If either is not turning, turn off the main power immediately and investigate the problem. If the computer does not boot, turn off the main power and explore all of the cables connected to the motherboard and try again. If the computer continues to not boot, you may have the power supply incorrectly connected, you may have the front panel power switch incorrectly connected, or your CPU may be damaged. If, after several attempts, you cannot get your computer to turn on, ask your instructor for help.
13. Shut down your computer. Place the side back on the system unit. Connect a monitor, keyboard and mouse to the PC and restart it. When the Intel logo screen appears, hit <F2> to enter the BIOS setup program. If you have successfully reached the BIOS setup program, you have completed the PC assembly correctly! Optionally, continue on to step 22 to install an operating system.
14. [optional] Install Windows 10. See lab 2 part 2.

To complete this assignment, answer the following questions. Use the figure below as reference. 

1. Having studied the above figure and worked through the installation lab, now use the motherboard in the figure below to identify each of the following items:
2. The socket to plug in the CPU
3. The sockets to plug in memory chips (DIMM memory)
4. The expansion board sockets
5. Connectors for peripheral devices
6. The power connector
7. The connector for your disk drives



1. The motherboards differ and also differ from the motherboard you will use. One difference is that the above motherboard has a separate integrated graphics processor with heat sink. Instead, you have added a cooling unit to your motherboard. Where does that component go?
2. Why do you need a cooling unit/heat sink? What does a heat sink do?
3. Why are the peripheral connectors located where they are? Why put them all on the same side of the motherboard?
4. The following figure shows the inside of a system unit. The system unit contains one item but otherwise is just a shell. Identify where you will place the motherboard, power supply unit, storage (disk drive) units and the fan. Of these, which item is already in place?



1. Why is it important to place items on the motherboard before you install the motherboard in the system unit?
2. Should the motherboard be installed before or after the drive units? Why?
3. Why do you need to wear the grounding strap? You should wear it when you are working with which components of the computer?
4. If you had any difficulties getting your computer to boot the first time, what problem(s) did you come across and how did you fix it (them)?
5. What is BIOS? What are some of the options available?
6. If you want to replace the computer’s memory, what steps would you take?
7. If you want to add another internal SATA hard disk to your computer, how would you do this?
8. If you want to add an external hard disk to your computer, how would you do this?
9. If you wanted to install Linux instead of Windows 10, how would this lab have differed? You do not have to specify the details of installing Linux, just describe which step(s) would have differed and why.

# Lab 2: Operating System Installation

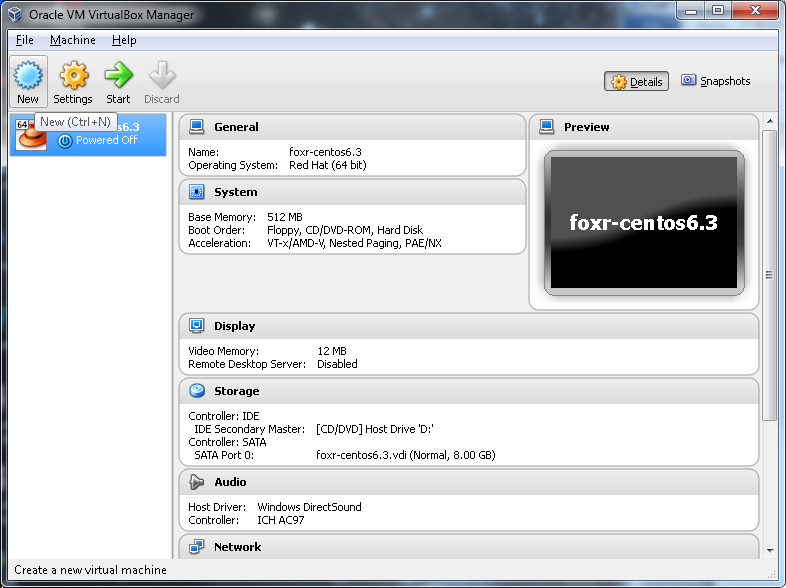
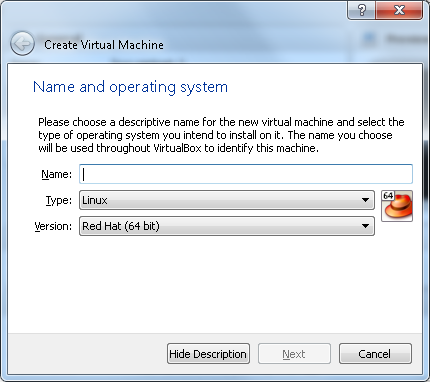
Goals: to learn how to perform operating system installation in both Windows 10 and Red Hat Linux (CentOS).

Introduction: for this lab, you will need to have installed some VM software (VirtualBox is assumed, see pages 4-5). You will need installation discs for both Windows 10 and Red Hat Linux CentOS. If your instructor does not have a CentOS installation disk for you, see the steps in part C of this lab.

Estimated time to completion: 90 minutes to 2 hours

Before starting, review chapter 4 Installing an OS.

Part A: Using VirtualBox

1. Start **VirtualBox**. Have both a Windows 10 and a CentOS installation discs ready. You can complete the installations in either order. The Windows 10 installation is easier.
2. For both virtual machines, follow these steps
   1. In Virtual Box, click **New**.
   2. **Enter** for the name your last name, first initial, a hyphen and either win7 or centos6 as in foxr-win7 or foxr-centos6.
   3. For Microsoft Windows and Windows 10 if you are installing Windows 10, and Linux and Red Hat if you are installing Linux. Click **Next**.
   4. Use the recommended memory size for Windows and 2048 MB for Linux, unless your instructor tells you otherwise. Click **Next**.
   5. Select **Create a virtual hard drive now** and click **Create**.
   6. For the Hard drive file type, use the default (should be VDI, Virtual Box Disk Image). Click **Next.**
   7. Use **Dynamically allocated** storage. Click **Next**.
   8. You will be shown your computer’s available disk space. Make sure it is sufficient for your virtual machine’s hard disk. Click **Create**.
   9. You will see your VM near the upper left hand corner of the Virtual Box window. It is powered off. You will need to power it on to use it. The first time you power it on, you will have to install it. Once you complete the installation steps, your VM will be available any time you need it. Insert your installation CD into your computer’s optical drive. Click **Start**.
   10. Now you will be taken through the installation steps below. Install the appropriate operating system. When done, return to Virtual Box, click new and repeat the above steps for the other operating system.

NOTE: VirtualBox will pop up some information windows. One is to use the Auto capture keyboard option. This allows your mouse and keyboard to control the VM rather than the outer machine. Once you have clicked inside of your VM, any operations are for the virtual machine, not the outer machine. To regain control of the outer machine, press the CTRL button on the right side of the keyboard.

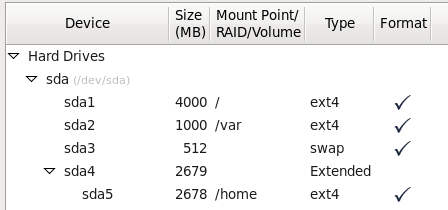
Part B: Windows Installation

1. Once you click on **Start**, you will be presented with the first of several Windows 10 installation dialogs. The first asks for language, time/currency and keyboard/input. Keep all of the default selections and select **Next**.
2. After a few moments, a Windows Setup window appears. Select **Install Now** which should be the only choice.
3. Select **I accept the license terms** and click **Next**. For Type of Installation, select **Custom**. On the Where to install dialog, your only choice should be Disk 0 Unallocated Space. Select **Next** to install Windows 10.
4. From this point, the installer will run for about 10 minutes and will reboot during the installation. During this time you might see a message to hit a key if you want to boot from the CD or DVD. DON'T HIT ANYTHING. If you do, the installer will start all over again.
5. During the installation, you will be asked what settings to use. Select **Use Express Settings**. After a few more minutes, you will be asked how you want to connect. Select **Join a local Active Directory Domain** and **Next**.
6. Use the username and password that your instructor give you for creating an account for your PC. The username will probably be something like your lastnamefirst initial (as in foxr). Add a password hint based on the password given you. Select **Next**.
7. When asked if your device should be discoverable by other devices on the network, select **Yes**.
8. You will now be presented with a Windows desktop. Your windows installation is complete. Notify your instructor that you have completed your Windows installation so that he/she can check your install. At this point, if you are unfamiliar with Windows 10, you might want to explore it.
9. From the Windows menu (the button in the lower left corner), select Power and Shut down. Once your VM shuts down, you can close that window but leave Virtual Box running as you will use it in the next part.

Part C: Linux Installation

1. If you do not have a CentOS installation disc, do the following, otherwise skip to step 2. Go to **wiki.centos.org/Download**. From this page, select the ISO image under CentOS 6.8 (either i386 or x86\_64). From the new page, select a mirror site that is close to you. Do not select a LiveCD or LiveDVD. You may either select a minimal ios or the full binary iso (which is on two DVDs although you should not need the second DVD).
2. Insert the Linux installation CD into your optical drive and start a new VM as per part A step 2 of this lab. Once set up, to install CentOS, click **Start**.
3. After a few moments, you will see the intro CentOS screen followed by a sequence of text. The first step is to perform or skip the media test. You will want to skip it. To gain control of the console with your mouse and keyboard, you need to click somewhere in the console area. Skip the media testing step by pressing **tab** to select **skip** followed by **enter**.
4. From the welcome page, press the **Next** key.
5. Make the following selections: **English** (**Next**), **U.S English** (**Next**), **Basic Storage Devices** (**Next**).
6. You will be given a storage device warning. Make sure “**Apply my choice to all devices**” is selected, click “**Yes, discard any data**”.
7. For hostname, enter your ***username*-centos6** where *username* is your username (such as foxr-centos6). Skip the Configure Network step (for now) and select **Next**. From the next screen, select the **appropriate time zon**e and leave the check box checked for System clock uses UTC (**Next**)
8. Enter the **root password** provided by your instructor, **confirm it** (**Next**). If the password is considered weak, you may use it anyway.
9. For installation type, select “**Use All Space**”, click the checkbox “**Review and modify partitioning layout**” (**Next**). **NOTE**: this assumes that you are installing CentOS in a VM. If you are installing CentOS on a computer that already has an operating system installed, DO NOT SELECT Use All Space. You would instead want to select Use Free Space. See your instructor if this is the case!
10. You will be given a list of hard drives and a LVM (logical volume manager) group. Select **sda** from the list of Hard Drives (not sda1 or sda2) and click **Delete**. From the pop-up window, click **Delete** to confirm the choice.
11. Repeat the following 4 times: Click **Create**. From the pop-up window, select **Standard Partition** and **Create**. Each time after clicking Create an Add Partition pop-up window appears. Specify the following in order:
    1. Mount type: **/**, file system type: **ext4**, Size: **4000**, **OK**
    2. Mount type: **/var**, file system type: **ext4**, Size: **1000**, **OK**
    3. File system type: **swap**, Size: **512**, **OK**
    4. Mount type: **/home**, file system type: **ext4**, under Additional Size Options: **Fill to maximum allowable size, OK**

When done, you should see something like this:



Click **Next**.

1. Select **Format** (**Next**) and from the Format Warnings pop-up window and **Write Changes to Disk**.
2. For boot loader, make sure **Install boot loader** on /dev/sda is selected (**Next**).
3. You are asked about the default installation, select **Desktop**. Leave “**Customize later**” checked (at the bottom) as you will modify the install later in the semester (**Next**).
4. At this point, the Linux installation will start. This takes several minutes (probably 10-15). The disc will be ejected at this point, return it to your instructor.
5. You will be asked to reboot your computer. This is actually the VM that will reboot. Click on **Reboot**.
6. After rebooting, you will be at a Welcome screen. Select **Forward**, agree to the license (**Forward**), **create your own user account**. Use the account name as specified by your instructor. lastnamefirstinitial (as in foxr) is recommended. Enter your full name and **enter the password** provided by your instructor and confirm it. Adjust the **date/clock** if necessary. (**Forward**)
7. You may receive an error about insufficient memory for kdump, click **OK**
8. Click **Finish** (ignore the kdump information).
9. You will be at the CentOS log in window and you will see your account listed. Click on **your username** and log in. You can play around with the Linux GUI. Also, you need to start your network interface. From the **System** menu, select **Preferences** 🡪 **Network Connections**. From the pop-up window, **expand Wired** (if it is not already expanded) and select **your *eth* interface** (probably System eth0) and select **Edit…** From the pop-up window, select **Connect automatically** and make sure **Available to all users** is selected. Click on **Apply…** and from the pop-up window, **enter your root password** (form step 8) and click on **Authenticate**. Close the **Network Connections** window. One addition thing you might need to do is increase the resolution of your display. If you are working in a VM which only provides a limited view into your desktop, you may not be able to see the bottom of some of the pop-up windows that arise in CentOS. Select **System** 🡪 **Preferences** 🡪 **Display**. Under Resolution, increase it to 1024x768 and select **Close**. When done, select **System🡪Shut down** and from the pop-up window, **Shut Down**.

To complete this lab, answer the questions on the next page. You may have to do some research to find answers.

1. Why did you have to specify partition sizes in Linux but not Windows?
2. What happens if you select poorly with respect to your Linux partitions and need to add or resize partitions later?
3. Why do you suppose the Windows install is so much easier (in terms of user effort) than the Linux install?
4. Provide 3 reasons why we are using VMs in this class rather than having you experiment on actual computers.
5. What are 3 reasons why a company may choose to use VMs.
6. In both installations, you had to create an initial account. Why?

# Lab 3: Files

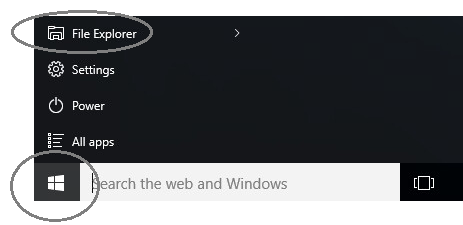
Goals: to learn about files and directories in Windows and Linux; to learn how to navigate around the Windows and Linux file systems; learn to use both GUI-based and text-based controls.

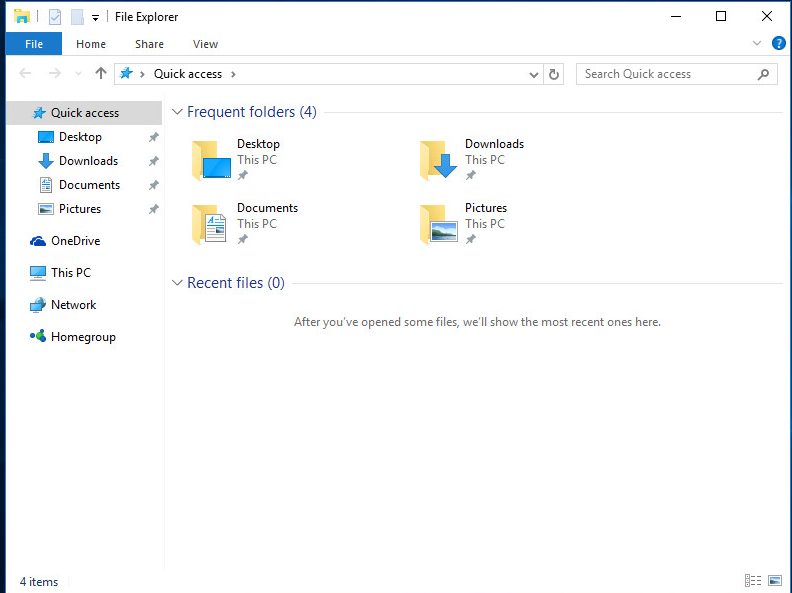
Introduction: in this lab, you will explore directories (folders) and files in both Windows and Linux. You will explore the types of files, how to view files, how to create files, how to move, rename, copy and delete files. You will do this both from the GUI and the command line. If you have experience with files in either or both operating systems, you might find parts of this lab to be very easy. You are expected to complete the lab nonetheless. You will work in both of your VMs, switching between them (Windows in parts 1 & 4 and Linux in parts 2 & 3).

Estimated time to completion: 90 minutes

Before starting, review chapter 5 Files and Directories, and Moving Around the File System.

1. To start with, we will explore some of the system files. In Windows, select the Windows icon (lower right hand corner) and from the list of choices that appear, select File Explorer (this is what used to be called Windows Explorer in older versions of Windows). See the figure below. Beneath it you will see the starting figure for the File Explorer GUI.





File explorer has 6 areas to note. Along the top left are menus where the View menu allows you to control how items are viewed (such as through icons or text). Beneath this is a row that has your navigation bar (controlling where you are and allowing you to move back, up or other) and a search bar. Beneath this on the left are locations you can view including “Quick access” items, the MS OneDrive cloud, the partitions available on your PC, your Network and your Homegroup. On the right are frequently accessed folders and any recently opened files. In this lab manual, we will always use the area on the left to control where we go and view the items of the current directory on the right.

* 1. Move the mouse to the left of the **This PC** icon and you will see a > symbol appear. Click on it. The symbol now points downward and the items available from here are listed. That is, you have expanded This PC into other locations. These locations include your User personal directories (Desktop, Documents, Downloads, etc), and the available partitions which are probably limited to just C: (your local hard disk) and D: (your DVD drive). Click on the > next to **C:** to expand it. Under the **C:** drive, expand **Program Files** and **Program Files (x86)** (note: depending on your installation, you may only have **Program Files**). These two directories store the application programs. The difference between the two directories is that “(x86)” refers to the older software, compiled as “32-bit”. Collapse these two directories (click on the downward arrow and it will go back to a > while collapsing the directories). Now expand the Users directory. *What users have a directory listed?*
  2. Instead of expanding a directory in the left pane, if you click on the directory, you see its contents appear in the right pane. Click on your user account in the left hand pane. *What items appear in the right hand pane?* To enter a directory from the items in the right hand pane, double click on it. Do so for the Documents directory. It should be an empty folder. In the right pane, *right click* and select **New** 🡪 **Text Document**. A new text document appears in the directory with the name highlighted, that is, Windows is waiting for you to either accept the new name or replace it. Type **foo** to change the name to **foo.txt** (the **.txt** extension is probably not displayed). The file is still highlighted, hit enter and the file opens in Notepad. Type in a short paragraph about how you’ve enjoyed this class so far. Save and close the file. *How big is your file? How can you tell?*
  3. Right click on the file and select **Properties**. *How big is the file in size and size on disk?* The discrepancy is that the size is the actual contents while size on disk is the amount of space reserved currently for the file. The reserved space is most likely 1 block’s worth at this point.
  4. In the Properties window, click on **Read-only** and **Ok**. Now, *double click* on the file name again to open it in **Notepad**. Add some text and save it. Save the file. *What happens? Why?* Save the file under a new name, **foo2.txt** (you can omit the .txt) and close the file. Bring up **foo2.txt**’s properties. You will notice under **Opens with:**, it lists Notepad. To change the default, click on the **Change…** button. A pop-up window appears listing all of the options that Windows thinks you will want as well as a link to find an app from the store. The two options listed are **Notepad** and **Wordpad**. You can also click on the **More apps** button to see all programs that Windows finds. We do not want to change the program that opens our .txt files, so instead just click on **OK** to close this window and then **Cancel** from the **Properties** window to close it.

1. We will now repeat what we did here in Linux. Start your CentOS VM and log in. This desktop is different from Windows. Along the top are three menus, Applications (software), Places (similar to what you did with the File Explorer) and System (items for administration), some buttons to run common applications (Firefox web browser, Evolution email/contacts, and the Gedit text editor). Further along the top are controls for volume, network access, the time/date, and a log out control. On the desktop itself are a Computer icon (similar to the My Computer icon in older Windows), a folder of your home directory and a trash can.
   1. Under the **Places** menu, select **Computer** (or *double click* on the **Computer** icon on the desktop) to bring up this Windowing system’s version of the File Explorer. If you select one of the other locations under **Places**, you open the File Explorer in that particular directory such as your user’s home folder. The name of the window that appears tells you your current directory, this one says **Computer**. From the **Edit** menu choose **Preferences**. From the pop-up window, select the **Behavior** and select the checkbox for **Always open in browser window**. **Close** the preferences window. Now, double click on **File System** in the Computer window. This opens a second window which provides the entire mounted File System of your Linux computer. This view looks more like Windows File Explorer with a left hand pane to control where you are and the items in the current location in the right hand pane. Above the left hand pane is a drop down box that says **Places**. Click on it and select **Tree**. Now the items in the left hand pane can be expanded like with Windows File Explorer. Click on the triangle to the right of **File System** in the left hand pane. The pane expands to include all of the top level directories of the file system. Expand the **home** directory. *What two folders are listed?* Click on your **username**. In the right hand pane you will see all of your initial items which are just subdirectories. Double click on the **Documents** directory. As we did in Windows, lets create a file. Right click in the right hand pane and select **Create Document** 🡪 **Empty File**. The new file appears without a name, type in **foo.txt** (in this case, make sure you include the .txt portion of the name). Hit the **enter** key to save the new name. While the file name is still highlighted, hit **enter** again to open the file in Linux’s GUI-based text editor, **gedit**.
   2. Like you did before, type a short something about the class. **Save** the file. Under the **Edit** menu at the top, select **Preferences**. From this pop-up window, you can alter the preferences of a gedit session. Under **View**, notice **Display line numbers**. *Why do you suppose you might want to see line numbers in a text file (hint: think of what you might write in a text file that you would not write in a word processor).* **Close** the preferences window.
   3. Under **Tools**, select **Document Statistics**. *How big is your file in Bytes? How many characters is it? Why are they the same?*
   4. **Close** the statistics window and then **close** **gedit**. Notice in the bottom left corner of the File Browser is a listing of the selected file and its size. *Does the number of bytes from the Document Statistics window match the number of bytes shown in your File Browser?*
   5. Right click on the file name and select **Properties**. You will notice the **Properties** window looks much like that in Windows (similar options) although there is no selection for “read only”. Click on the Permissions tab and you will see drop down windows to change the permission of the file for owner, group and others. Rather than changing the file here, we will examine how to deal with permissions in lab 5. **Close** this window. Right click on the file again and choose **Copy**. Then, in an empty area of the same file manager window, select **Paste**. You will see the new file is called **foo (copy).txt**. Right click on the new file’s name, choose **Rename…**, and rename the file to be **foo2.txt**. Double click on this file to open it, edit it, save it and exit. *How many bytes is the new file?* Close your **File Browser** and the **Computer** window.
2. We will now explore how to do things with the command line in Linux. From the Linux desktop, right click and select **Open Terminal**. A terminal is a window that displays a command prompt, and (of course) allows you to enter commands. Interacting at the command line is often easier than using the GUI. Notice that by starting the terminal window using the Desktop’s right-button menu, we will be placed in our user’s Desktop subdirectory. Had we started a terminal window from the Applications menu, we would instead be placed at the top of our user’s home directory which is also called ~.
   1. To see where you are, use the pwd (“print working directory”) command. Type **pwd**. *What response to do you get?*
   2. Now type **ls**.(This stands for “**l**i**s**t”.) *What do you see?*
   3. Next, type the command: **ls –l** (that’s a lower case “L”). The “-” means “the following is an option”, and the “l” means “list items in the *long* format”. *What information is given that wasn’t given using just “ls”?*
   4. Type **ls –al**. *How does this one differ from the previous?* You can find out more about commands by using the **man** command. This command displays the command’s manual page. Type **man ls**. You can step through the pages of information by using the space bar (go to next screen), enter key (go to next line), up or down arrow (go up or down 1 line) and q to quit. Step through the man page for ls so that you can see what –l and –a do. *What does the “a” stand for?*
   5. Let’s explore a more interesting directory. Type **cd ~**. The cd command changes your directory and ~ takes you to your home directory. Type **ls –l**. Items listed in blue are directories. *How many items listed are directories? How many items listed are not?* Type **ls –al**. How does this listing differ?
   6. You might have noticed when changing directory that your prompt changes. *What does your prompt show?* To change directory, you can type cd .. (go up 1 level), cd /directory or cd directory. Without the /, you can only go into a subdirectory of your current directory. Type **cd /Documents**. *What does your prompt show you now?* Type **cd /etc**. *What does your prompt show now?*
   7. In the /etc directory, type **ls** and you will see a lot of files in three columns (you get multiple columns when there are too many items to list). Type **ls –l** and you will get the long listing in one column. One important file in this directory is **passwd** (the list of all user accounts). To view the contents of a text file, you can use any one of three commands: **cat**, **more**, or **less**. Use all three of these on the **passwd** file (**cat passwd**, **more passwd**, or **less passwd**). You might notice that **more** and **less** cause the listing of the file to stop at the end of each screen’s worth. To move forward, press the **space bar**. To exit **less**, type **q** once you hit the end of the file. The **less** command is better than **more** because you can move backward and forward with **less** by using the arrow keys. Use **more** and **less** again on this file. *What does the* ***more*** *command tell you that the* ***less*** *command does not?*
   8. To copy a file from one location to another, use the command **cp**, as in

**cp *filename*** ***destination***

where ***destination*** will be the directory to move it to. Copy this file to your user account by doing **cp passwd /home/*username*** where ***username*** is your username. Change back to your home directory. You can do this using **cd /home/*username*** or **cd ~**. Try the first approach since you have already used cd ~. *What does your prompt list now?*

* 1. Type **ls** and you should see the **passwd** file along with all the directories that you’ve already seen. We will move this file to a subdirectory. First, create the subdirectory named **temp** by using **mkdir *directoryname***. *What command did you enter?*
  2. Change directory to this new directory. *What command did you enter?*
  3. Do an **ls** and you should see the only thing listed is passwd. To rename or move a file, you can use the same command, **mv**. To move a file, you would enter **mv *filename* *newdirectory*** and to rename a file you would enter **mv *filename newname***. Change the name of the file to **accounts**. *What command did you enter?*
  4. View the contents of the file to make sure it is the same file. *What command did you enter?*
  5. To delete a file, type **rm *filename***, where ***filename*** is the name of the file. Delete this file. *Did you receive a message?* To always be prompted when deleting files, you can use **rm –i *filename***, and to avoid any potential message, you can type **rm –f *filename***. Use man on rm to read about the options. Let’s explore this. Copy the file again but this time we are in a different directory, so we have to specify the full path to the file in our cp statement. Recall the cp command is cp *source destination*. The source is /etc/passwd and the destination is “here”, which we can indicate with the period. Type the proper cp command to copy the passwd file here. Do an ls to make sure it worked. If you did it correctly, *what command did you enter?* Now attempt to delete it using rm –i instead of rm. *What message do you get?* Answer **n** to not delete the file.
  6. View the passwd file using less (or more). *What information do you see about the root account (the first one). Compare this to your own account (which will probably be the last account).*
  7. Move up one level in the file space (cd ..). Let’s try to delete the **temp** directory. Type **rm temp**. *What happened?* To remove a directory, we use rmdir instead of rm. Type **rmdir temp**. *What happened? Why?* We can delete this directory by either first deleting the file from within it, or by using the **rm** command recursively. A recursive delete means that rm will delete everything in the directory first and then delete the directory. If the directory contains subdirectories, it will perform the recursive delete on them first, and so forth for possible subsubdirectories. To do a recursive delete, use the option –r for rm as in **rm –r temp**. Type this in and see what happens. Notice how it deleted temp without asking. rm –r can be risky because you might in fact delete something you don’t want deleted. *How would you use rm –r so that you are prompted before each deletion?*

1. In your Windows VM, from the **Windows** button menu, select **All apps** and scroll down until you find **Windows PowerShell**. Click on it and a sublisting appears. Right click on **Windows PowerShell** and select **Pin to Start**. This item now appears to the right of the listing and below the other tiles. This allows you to bring up the PowerShell more conveniently (and we will use it from time to time in this lab manual). Click **Back** to remove the listing and then click on the Windows PowerShell tile (which will just be called PowerShell from here on). This opens a new window which provides a command prompt that will probably say PS C:\Users\*username*> .
   1. Notice like CentOS, your prompt is telling you where you are but unlike CentOS it is giving you the full path to the current directory. From this prompt, you can type commands. To see the files/subdirectories, type **dir**. This command is equivalent to the Linux **ls** (list) command. You will see a creation date and time, whether the item is a directory (<DIR>), the item’s size if it is a file, and the file size. At the bottom, you will see a summary listing. We want to change directory to the **Documents** directory, where you placed your two previously created text files. You change directory in the same way as Linux, using **cd**. Enter the appropriate command to get to the **Documents** directory. *What command did you enter?*
   2. Notice that unlike Linux, directory and file names do not have to match the proper case (for instance, you could reference **Documents** as **documents** or **DOCUMENTS**). To view a file, use the command **type** followed by the *filename*. Now, view the content of your file **foo.txt**. *What command did you enter?*
   3. To move upward in the directory structure, use **..** as with Linux, so for instance, **cd ..** will take you back up to your user account directory. You can go up or down multiple levels by separating each name with a “directory slash” In Linux, we use / but in Windows we use \. To go up two levels, you would type **cd ..\..** Instead, enter a command to go up 3 levels. *What command did you enter? What directory are you in now (what does your prompt say)?*
   4. Enter **dir**. *What do you see?*
   5. Change directory to **\Windows\System32**. *What command did you enter?* Do a directory listing. *What do you see?* Notice how long the list is; it runs off the screen and keeps going! If you get tired of watching all of the items appear, type c**ontrol+c** to stop the command.
   6. Switch directories to **C:\Windows\Tasks**. *What command did you enter?* Do a directory listing and you should see an empty directory. Or is it? Type **help dir**. This is like the Linux man page although without nearly as much information. You will see that this command (dir) is actually called Get-ChildItem (you can also use gci and ls). Among the options is –Hidden. Enter this command with the hidden option. *What is the file that you find here?*
   7. Change directory back to your user Documents subdirectory (\Users\username\Documents). *What command did you enter?* As with Linux, some of the commands you can issue here are similar:

**move, copy, del, help, mkdir, rmdir**

Copy the file foo2.txt to foo3.txt. *What command did you enter?* To rename a file, you would use move much like you used mv in Linux. Rename foo3.txt to foo4.txt. *What command did you enter?* Now, delete this file (foo4.txt). *What command did you enter?*

* 1. From your **Documents** directory, create the subdirectory **Temp** using **mkdir Temp**. Copy both “foo” files into your **Temp** subdirectory (note by using **foo\*.txt**, you can copy them both with a single command as you did in Linux). *What command did you enter?* Examine your **Temp** directory to make sure the files are there. Now, move up one level (cd ..) and try to delete the temp directory using **rmdir**. *What message did you get?* Respond L so that the deletion is not performed. Type **dir Temp** to ensure that the files are still there.
  2. Do a **help** on **rmdir** and you will see that one of the options is –Recurse, which is similar to Linux’ rm –r. *What are the aliases (other names) for the rmdir command? What is the real name for rmdir?*

At this point, close any open windows in both of your VMs and shut them both down. You can now exit from VirtualBox.

# Lab 4: File Systems

Goals: to learn about and interact with the Linux and Windows file systems (partitions) and the top level directories.

Introduction: in this lab, you will use both your Windows and CentOS VMs. You will work as administrator/root in order to carry out some file system commands. This lab follows on from material covered in lab 3.

Estimated time to completion: 90 minutes

Before starting, review chapter 5 Linux File System and Windows File System.

1. Exploring the Linux file system
   1. Login as yourself (not root). The df command reports the disk file space usage. Enter **df** and look over the results. *What do the results mean?* (*Hint*: recall the partitioning steps from the Linux installation.)
   2. Partitions are made available by mounting them. You can see what is mounted at *boot time* by looking at the contents of the **/etc/fstab** file, and what is currently mounted by looking at the **/etc/mtab** file. (Recall the use of **less**, **more** and **cat** in the previous lab.) Examine the contents those two files. *Compare the two files by listing which partitions appear in both and which appear in only one.*
   3. The command **mount** will “mount” a partition to the filespace while **umount** will *unmount* a partition. When a partition is unmounted it is not accessible (until it’s mounted again). Enter **umount /var**. *What message did you receive?* Enter **su**, then enter the root password (from lab 2) to become root. Notice that your command prompt has changed to be #. Whenever you are the root user, your command prompt will end with a “#”. As the root user, re-enter **umount /var**. *What happened?*
   4. You cannot **umount** a file system that is currently busy and unfortunately, all of the file systems are busy once the system has started up. We can “unbusy” some of the file systems by switching from the current system mode (known as the runlevel) to the single-user text-based mode (single-user means root only). We do this by entering the command **telinit 1**. As root, this allows you to switch from your current runlevel, which is 5 (GUI, multi-user), to runlevel 1 (text-based, root only). Now you can unmount any of the partitions except for /. Let’s explore this. First, again view the contents of /etc/mtab. You will notice one currently mounted file system is called sys. Type **umount sys**. View /etc/mtab again. *What has changed?* Type **mount sys** to remount it. Again, view /etc/mtab. *Did sys return? Why might you want to unmount a partition? What happens when that partition is unmounted in terms of its availability?* Type **exit** which will return you to run level 5. You will have to log in again (as your normal user).
   5. Open a terminal window again and su to root again. The **du** command (which stands for ***d****isk* ***u****sage*) can be useful to determine the amount of disk usage of any particular directory or file system. Do a man on du. Two useful options are –s, to only summarize the disk usage, and –h to print the usage in a “human readable” form. Change directory to your home directory and type **du –sh**. *What is the result?* Now, obtain the disk usage for /etc. *What command did you enter. How much space is being used by /etc?*Type **exit** to exit out of being logged in as root. You should resume with your normal prompt ($). Repeat the command to obtain the disk usage for etc. *What happened and why?*
   6. Research the Linux top level directories. *In general, what do you find in each of these? /bin, /boot, /dev, /etc, /usr/bin, /usr/sbin, /var?*
2. The **PATH** environment variable.
   1. Remain as your normal user account in this part. The **PATH** variable stores a list of directories that are searched when you issue a command. That is, when you issue a command, say rm, the command is searched for in your current directory and if not found, then each directory in the PATH variable is searched. If not found, you get an error message. Without PATH, you would have to type the full path of the command, /bin/rm, or you would get an error. Type **echo $PATH** to see the contents of the **PATH** environment variable. *What directories are included in your* ***PATH****?*
   2. The **which** command will search all directories in **PATH** for the location of a file. For instance, try **which rm**. Use **which** to find the location of each of these programs: **sed,** **telinit,** **yum**. *Where are each of those programs located (if given)?* Now try **which useradd**. *What result did you get for* ***useradd****?* In fact useradd is in /usr/sbin which is in your PATH but because useradd is not available to ordinary users, you cannot find its location using which.
   3. You can add to your PATH variable if you like. Let’s imagine that you want to also have the directory /usr/games in your PATH to play games stored there (currently that directory is empty). To add a path, use the notation **PATH=$PATH:*newdirectory*** where the new directory is a full path as in /usr/local/bin. *What command would you enter to add this games directory to your PATH?* Instead of entering the command to add games, type **PATH=**  This will clear the contents of the **PATH** variable (reset it to be empty). Now enter the command **mkdir foo**. You should get an error that the command was not found. *Why?* Now try **which mkdir**. *What was the result?* Type **echo $PATH**. *What result did you get?* Type **exit** to leave your current session (which also closes this terminal window) and then open a new terminal window. If you do echo $PATH, you will find your PATH restored. Stay logged in as your normal user, not root.
   4. Another program that helps you locate not just programs but also man pages and source code for programs is called whereis. Type **whereis useradd**. Notice that unlike which, you are given an answer instead of an error. Type **whereis telinit**. *What is the output? How does this differ from what you got in 2b?* Now type **whereis sed**. *Compare the result to whereis telinit (that is, what additional information were you given for sed?)*
3. The best way to find files is with the **find** program which not only searches based on names but on file attributes such as creation date, owner, empty files, permissions and more. We will briefly explore it here. Start by doing a man on find. It is a very lengthy man page, do not try to read it all.
   1. We will start with the basic find command. The structure is

**find *starting-directory search-query***

where the directory indicates the starting point for the search. If you use . then you start searching from the current directory whereas ~ searches from your home directory and / searches the entire Linux file space (note: you should be root to search from / or you will wind up with a lot of permission denied messages). Issue the command: **find ~ -name “\*.txt”**. *What files were listed?* Now try **find /etc –name “\*.conf”**. The output scrolls passed the screen. Redo the command as **find /etc –name “\*.conf” | less** which adds | less meaning “pipe the results to the less command”. A pipe is used to conjoin two instructions. In this case, the result of find is send to less so that the output pauses screen-by-screen. *How many permission denied statements did you receive?* su to root and redo the command. *Did you receive any permission denied statements?*

* 1. Exit as root. To search for modification and access times, use the command options **–mtime** or **-atime** (which stand for “modified” and “access” times) followed by a value n, -n or +n where n is an integer indicating days. The – indicates “less than”, the + means “greater than” and no + or – means “exactly”. For instance, -mtime +2 means “modified at least 2 days ago”. You can also use –o or –a to represent “or” or “and” to combine multiple options as in –mtime +1 –a –mtime -7 to mean “modified at least 1 but no more than 7 days ago” or –mtime -10 –o –atime -10 to mean “modified or accessed less than 10 days ago”. Issue a find command to find all files in your home directory that have been accessed within the last month (use 31 for the number of days). *What command did you enter?* You will be surprised at how many files are listed. Redo the command but pipe it (use |) to wc –l which counts how many lines where returned. *How many files were accessed in this time?*
  2. The **–empty** option will find any file or directory that is empty. Issue a find command using this option. Find all empty files in your home directory. *What command did you enter?* You will see a lot of different files and subdirectories. As you did in part b, add **| wc –l** to obtain just the number of lines. *How many items were found?* Another option is **–perm** ***number*** to find all files whose permissions are set to that number. We study permissions in lab 6 but let’s try a few here. One common permission is 644 (readable and writable by you, readable by everyone else). Search for all files in your home directory with permissions of 644. Again pipe the result to wc –l to obtain the number of files. *How many files were found?* su to root and find all files under /etc whose permissions are 000 (do not use wc –l so that you can see the actual files). These are files that are not accessible to anyone (other than root). *Which files were found?* Exit from root, and close your terminal window. You may now shut down your Linux VM.

1. Exploring and modifying the Windows file system
2. Start your Windows VM and log in. We start by exploring available disk space. Using the Windows start button, open the File Explorer. In the left pane, click on This PC. In the right pane is listed “Devices and drives”. *How much space is available on* ***C:****, (in gigabytes)? How much space is used on* ***C:****, (in gigabytes)?*
3. By default, Windows puts all of your hard disk space in a single partition, the **C:** drive. You can divide your hard disk up into multiple drives if you want. We will do that now. From the Windows start menu, select **Settings**. In the search box in the upper right hand corner, type **Control Panel**. From the list of items shown, click on **Control Panel**. You may now close the **Settings** window. In the **Control Panel**, select **System and Security** and then **Administrative Tools** and then double click on **Computer Management**. A Computer Management window appears. In the pane on the left, if Storage is not expanded, expand it (click on the >) and then double-click on **Disk Management**. The middle pane now shows your available disks where Disk 0 is the C: drive and contains your operating system. You should have plenty of extra space on **Disk 0**. We will use a small portion of this space to create a new partition. Under the unused space portion of Disk 0 (which will probably be listed as something like 31.90 GB NTFS, Healthy (…)), right click in this box and select **Shrink Volume…**. After a few moments, a Shrink pop-up window appears. Under the amount to shrink, enter 2000 (this is 2000 MB or 2GB). This leaves plenty of space leftover in the C: partition. Now, click on the **Shrink** button. *What appears under Disk 0?* Right click in this new box and select **New Simple Volume…** Step through the wizard. When you reach “Assign Drive Letter or Path”, you are allowed to name this new partition. *What choice does it default to?* Change it to **Q**. Click **Next**. The next window provides format partitioning information. Leave the defaults as is. *What is the file system type?* Click **Next**. You will see a summary of what it will do. Click **Finish**. In the upper middle pane, you should now have an extra item listed. *What are the items listed? How do the two statuses for C: and Q: differ?* Right click on your **New Volume** and select **Properties** and make sure the **General** tab is selected. At the top is a textbox that says **New Volume**, replace this with **My Partition** and then click on **Apply**. *How much of the space of this partition is used?* You can close the properties window now. Notice in the Computer Management tool that the new partition is now listed as My partition (Q:). Close the Computer Management tool. You should find a pop-up window waiting asking if you want to format the new partition. Click **Cancel**. Now, close the **Administrative Tools** window and the **Control Panel**.
4. Your File Explorer should still be open from step a, if not, open it. Click on **This PC** and you should see in the right hand pane all of the drives available including your new partition, Q. *How much free and available space is on your* ***C:*** *drive now? How much free and available space is on your new Q****:*** *drive?*
5. Keeping the File Explorer open, we will search for files.
6. Before we start, let’s create a file to find. In File Explorer, go to your home user directory (C:\Users\*username*) and create a new text file called **findme**. If you don’t recall how to create the textfile, see lab 3. Edit this file in Notepad to include your name and today’s date. By default, Windows hides filename extensions. Change this by selecting the **View** menu in File Explorer and then clicking on the **File name extensions** checkbox in the middle of the menu’s ribbon. *What happens to the file name?* Back in File Explorer, click on **This PC** in the left hand pane. In the upper right hand corner of the window you will see “*Search This PC*”. This is a search bar. You can enter text here and Windows will search the given directory (and its subdirectories). Type **findme** and press enter. The search may take a little time. *How many items were found?* You might be surprised that more than one shows up! *Explain as best you can what each of these entries is.*
7. Click the **back arrow** in **File Explorer** until you reach your Users directory (this should take 2 back arrows). **Drag** the icon **findme.txt** into the **Desktop folder** in the right hand pane. You should see the file appear on the desktop now. From the desktop, **drag the icon** into the **Recycle Bin** and then right click on the **Recycle Bin** and select **Empty Recycle Bin** and answer **Yes** to the pop-up window. In File Explorer, again click on **This PC** in the left hand pane and in the search box, search for findme again. *What is found this time? Is the actual file found?* Attempt to open one of the other files (such as the “Recent” one). *What is the result? Summarize what these entries that were found represent.*
8. Again select This PC in the left pane of your File Explorer. In the search box, type \*.txt. The \*, as you probably have figured out in this and the last lab represent a “wildcard” meaning match anything. In this case, you are searching for all files that end in .txt. You will find a great many files listed. If you search through the listing, you should find both foo.txt and foo2.txt listed. If you wanted to locate just these two files, *what would you enter in the search box?* Try it to make sure it works and lists just the two files. You may not close your File Explorer and shut down your Windows VM.

# Lab 5 Managing Users and Groups

Goals: to learn how to create and modify user and group accounts in both Windows and Linux.

Introduction: you will create and modify user accounts and group accounts using both graphical tools and command line tools in Linux and using the GUI in Windows. You will also explore how user and group information is stored in **/etc** files in Linux. You will also see how to set up a user’s personal information in their account. You must complete this assignment before attempting lab 6. You will start in Linux and then move on to Windows later in the lab.

Time to completion: 90 minutes to 2 hours

Before starting, review chapter 6 Setting Up User Accounts and Role of a Group.

Start your Linux VM and log in as yourself, and open a terminal window.

1. From the **Systems** menu, select **Administration > Users and Groups**. You will be asked for the root password. You will notice that there is already one user in place, you. To create a new user, click on the Add User button and then fill in the pop-up window specifying User Name, Full Name, Password, Confirm Password and clicking on OK. You can leave all other information as it defaults. Do the following:
   1. Create new user **Frank Zappa** with username of **zappaf** and password of **gail**. Leave all of the defaults as is. Click on **OK**. You will be warned that the password is too weak but use it anyway. When done, your newly created user will appear in the **User Manager** window below your account. You did not specify a UID (User ID). *What ID was assigned? Why do you suppose zappaf got this ID?*
   2. Select **zappaf’s entry** and select **Properties**. A properties window appears with 4 tabs, User Data, Account Info, Password Info, Groups. You can use this window to change information. We will leave the User Data alone. Click on **Account Info**. Set this account to **expire** in **1 year from today**. *How did you do this?* Select **Password Info**. Click on the **Enable password expiration** checkbox and change **Days before change required** to be **1** and **warning** to be **0**. Click on the Groups tab and scroll down. *What group(s) is zappaf a member of?* Click **OK** to enter the changes and exit the properties window.
   3. In your terminal window, enter **echo ~zappaf**. *What is this value?* Type **ls ~zappaf**. *What was the output? Why?* su to root and try again. *What happens this time?* Now try **ls –a ~zappaf**. *Why did something appear with –a but not without it?* Where did these initial files come from? Type **ls –a /etc/skel**. Compare this listing to the listing from ~zappaf. *What do you notice about the two listings?* This directory is available for the system administrator to establish initial files and directories to populate new user home directories.
   4. Let’s change **zappaf**’s password. As root, type **passwd zappaf**. As root, you are allowed to change anyone’s password. You are asked to enter a new password and then to confirm it. If you were to use passwd to change your own password, you would be required to specify your current password, but this is not the case with root. Make the new password be **dweezil**. *What warning do you get?* Change the password again, this time to frunobulax. *Did you get a warning?* Since this is such a difficult password to remember, change it one more time, back to gail.
   5. Type **su zappaf** which switches you to being user zappaf. Since you did this as root, you do not need to enter zappaf’s password. Type **exit** to switch back to **root,** then **exit** again to switch back to your user account. Now **su zappaf**. You are asked to enter a password, enter **gail**. You are now the **zappaf** user. Type **whoami**. *What was displayed?* Type **exit** to return to your own account.
2. Return to the **User Manager** window and switch to the **Groups** tab. (It’s in the middle on the left next to the **Users** tab.) You will see that there are already two groups, one for each user (your user and **zappaf**). The existing groups are known as “private groups” and only contain a single user. (Whenever you enter **ls –l** to get a directory’s listing, you will you see an **owner** (a user) and a **group** listed for each file.)
   1. Click **Add Group** and enter **infotech** as the group name (note that you can “manually” set the group ID, but you don’t need to do this). *What is the Group ID for this new group? What Group Members are there?* Select the new group and select the **Properties** button. Select the **Group Users** tab, and add your user account and **zappaf** to the group (by checking the appropriate check box). Click **OK**. Notice that the names now appear in the table under **Group Members**.
   2. Create another group called **musicians** and add **zappaf** to it. Click on the **Users** tab, select **zappaf** and select **Properties**. Select the Groups tab and scroll down to see which groups zappaf is now a member of. *How many did you find?* Close the **User Manager** tool (**Close** from the **File** menu or using the **X** in the top-right corner).

-c “comment”

1. We will now see how to manage accounts from the command line. To create a new account, use **useradd**, a program located in **/usr/sbin**. If your PATH does not include /usr/sbin (it should, but it may not), you will have to use the full path name of /usr/sbin/useradd. It might be worth your while to add /usr/sbin to PATH (recall that we did this in the last lab).
   1. **su** to **root.**  Enter **useradd**. *What happens?* (don’t place the response in your answers, just explain what was output). At a minimum, the useradd instruction requires that you specify the new user’s username. When we use it, we will want at a minimum to include a comment using **-c “*comment*”** where the *comment* will be the user’s full name, and **-m**. *What does –m specify?* Enter a useradd command to create a new account for user Ruth Underwood, account name underwoodr. *What command did you enter?* Use the passwd command to assign her an initial password of **xylophone**. *What warning message did you get?*
   2. Re-examine the options for useradd (either type useradd by itself or do a man on useradd). Now, create another new user, Tommy Mars (username of marst) who will automatically be added to the group musicians. *What command did you enter?* Give marst the initial password of xyz12abc. *What warning did you get this time?*
   3. Again, consult how to use useradd. Create user Mike Keneally (keneallym) with a shell of /bin/csh and an initial UID of 1001 rather than the default. *What command did you enter?* Give keneallym an initial password that will not cause a warning. You might have to try several until you get a good one. *What password did you use?*
   4. Create a new user **itperson**, with the comment **The IT Dude**, make this account one that is created *without* a home directory and with the login shell of /sbin/nologin. *What command did you enter?* Give the user an initial password of **citcit**. *What warning did you get?*
2. When you create or modify user accounts, Linux modifies several system-level files. Let’s explore these: /etc/passwd, /etc/group, /etc/shadow.
   1. User account information is placed in /etc/passwd except for their group affiliations, which is in /etc/group and password information which is in /etc/shadow. The passwords used to be stored in an encrypted form in /etc/passwd but since passwd is readable by everyone, it was felt that this was a security problem, so passwords are now in /etc/shadow (still encrypted) and an ‘x’ is placed in the corresponding field in /etc/passwd. Using less (or more), view the contents of /etc/passwd. You might notice that aside from accounts you created, there are numerous accounts for the system itself (root, bin, daemon, adm, etc) and for a number of different application software (mail, gopher, ftp, avahi-autoipd, etc). *How many total accounts are there? How many of these are users you created? What are the range of UID’s for these user accounts? How many of these user accounts have a login shell of /bin/bash and how many have other login shells?* You might notice that all of the accounts you created (except itperson) has a home directory under /home. *Of all the accounts in the passwd file (not just the user accounts)* *which ones have home directories under /bin? Under /sbin? Under /etc?* Most of the rest are stored somewhere under /var.
   2. Now examine the /etc/shadow file. If you look at root, you will see the encrypted password after the account name and a colon. The password is much more lengthy than the actual password because of how encryption works. If you continue to look through the accounts you will see most have no passwords, only those accounts that you created (along with your user account created when you installed the OS). Focus on your user account. After the password are 4 numbers separated by : and then some extra :. If you see :: it means that the value that was supposed to be placed between the two colons was not set. These numbers indicate various password and account settings such as the number of days before an account expires or before a password must be changed. Compare your user account values to that of zappaf. The first number should differ because that number indicates the age of the account and your account was created some weeks ago. The actual number indicates the number of days since the epoch. Research this on the Internet. *What is the epoch?* The third entry in the list of numbers should differ. *What is the value for your account and the value for zappaf? Why are they different?*
   3. Look at /etc/group. Type **tail -8 /etc/group** which will list the last 8 entries. These are all of the groups created automatically when a user account is created (these are known as private groups) or those you created today. *What is the GID for your user account’s private group? For keneallym’s group? Who are members of the musicians group?*

NOTE: You can directly edit the **/etc/passwd** and **/etc/group** files, but it is far safer to use the tools **useradd**, **usermod** or **groupadd**. Unless you have a specific reason for editing one of these files, it’s best NOT TO!

1. We will now use **groupadd** and **usermod**. The **groupadd** instruction (also in **/usr/sbin**) is simpler than **useradd**. One can specify the group ID when creating a group using a command like: **groupadd –g *GID* *groupname***. Typically, there is no need to specify a specific group ID, the default group ID is fine.
   1. Examine the **/etc/group** file to find the last group ID used by the system. Use groupadd to add a group called students with the next available GID. *What command did you enter?* Type **cat /etc/group**. *What was the entry for the group students.*
   2. The **usermod** instruction allows you to modify an already existing user. These parameters are much like **useradd**. Enter **man usermod** to see the option available for this command. (You can use **q** to exit from **man**.) *What is the difference between the –G option and the –g option?* Using usermod, add your user account to the group students. Do the same for zappaf. *What commands did you enter for zappaf? How can you confirm that this worked?*

At this point, you are done with your Linux VM. Close any open windows including terminal windows and start your Windows VM.

1. Log into Windows using your user account. Your account was set up initially as an **Administrator** account. To create users in Windows, we can do this from the GUI. From the **Windows** start button, select **Settings**.
   1. From the **Settings** window, click on **Accounts**. *What does it say about your account in terms of the roll (listed underneath your account name)?* Click on the button **Family & other users** in the left pane. You will see on the right a selection to add someone else to this computer. Instead of using this approach to creating accounts, we will use the computer Manage tool. **Close** this window.
   2. Start the **File Explorer**, right click on **This PC** and select **Manage**. From the Computer Management tool, expand **Local Users and Groups** and click on **Users**. In the right pane you will see the already existing using accounts. *What are the accounts on this computer?*
   3. In the left hand pane, right-click on **Users** and select **New User…**. For the **User name**, enter **zappaf**. For the **Full name** enter **Frank Zappa**. You can leave description blank. For a password, use **gail**. Leave the **User must change password at next login** box checked. Click **Create** and then **Close**. The new user, **zappaf**, should now be listed under the list of users. Create two other new users, underwoodr and marst, as you did in steps 3a and 3b using the same passwords as in part 3. *What users are now listed and which ones have full names?*
   4. Files will be created for this new user. Return to **File Explorer** and expand C: and Users. *Do you see a directory for* ***zappaf****? Why or why not* (you might need to come back to this question after you complete a few more steps)?
2. Set up a group for users.
   1. Return to the Computer Management window from step 6. Click on **Groups**. *How many groups do you see listed? What is the description for the Users group? What is the description for the Administrators group?* Right click on the **Users** group and select **Properties**. *What users are listed as members (list all items)?*
   2. Right-click on the **Remote Desktop Users** group and select **Add to Group…** That will pop up a **Select Users** window. Click on **Add…** and type in the user name **zappaf** and then click **Check Names**. The name will change to something longer but include zappaf at the end. Click on **OK**. You should now see **zappaf** added to the remote desktop users list in the **Remote Desktop Users Properties** window. Click on **OK**. You can add users to any of the listed groups in the same manner. From the **Windows** start button, at the top of the window of choices that appear, select your **user account** and from the list of choices, select the user **Frank Zappa**. Login as zappaf (password of **gail**). Notice you are asked to change your password. Do so changing it to **dweezil** and then click on **sign in**. It will take a moment for Windows to set up the new user’s desktop. Once logged in, immediately sign out (from the **Windows** start button, select **Frank Zappa** near the top and **Sign Out**). You will be at a list of users. Select your own account and enter your password. File Explorer should still be running but if not, restart it. Find zappaf’s home directory under C:\Users. Recall in part a it did not appear. As your own account, you don’t have access to zappaf’s home directory but as administrator you can look at it, so you are asked whether to continue or not. Select **Continue**. Look at zappaf’s items and compare them to the items in your own User directory. *What differences do you find*?
   3. To finish off, we will create a new group and populate it. In the **Computer Manager**, right click on **Groups** in the left pane and select **New Group…** A pop-up window appears. Enter the group name **Students**. Under the **Members** window, click **Add…** type **zappaf** in the **Select Users** pop-up window and **Enter** and then repeat for **underwoodr**. When both names are listed, select **Create** and then **Close**. You will see a new group, Students, listed at the bottom of the middle pane with new description. Right click on Students and select Properties. **Enter** a description. Click on **OK**. Imagine that later you want to add other users to this account. *Explain how you would do this.* You may now close the Manager GUI.

You may shut down your Windows VM now.

# Lab 6 File Permissions

Goals: to learn about file/directory permissions, how to view them and change them in both Linux and Windows.

Introduction: in the last lab, we created user and group accounts. We will now see how users can establish permissions so that other users can have different forms of access. In Windows, we use the GUI to control file permissions. In Linux, we will use the command line and the commands chmod, chgrp and chown. We will start with Windows.

Time to completion: 90 minutes

Before starting, review chapter 6 Permissions.

1. Start your Windows VM and log in as yourself. In Windows, permissions are specified on each file and directory. You can view and alter the permissions through the file/directory’s properties window. For any file/directory, you can add specific users and specific groups and alter their properties. We will explore this in this part of the lab.
   1. Open up **File Explorer**. Go to your **Users home directory**. **Create** 3 new text files, calling them a1.txt, a2.txt and a3.txt (refer back to lab 3 if you do not recall how to do this). **Open** each file one at a time and in the file, place some simple text like “this is file a1.txt”. **Save** and **close** each file. **Right click** on the **a1.txt** icon and select **Properties**. From the **Properties** window, select the **Security** tab. You will see several groups/user names (your own account, Administrators and SYSTEM, possibly others). Step through them all. *Which ones have full control and full permissions (other than special) on this file? Do any of them not have permissions?* Select the **Edit…** button. From the new window, click on **Add...** and type **zappaf** in the box and click on **OK** to return back to the Properties window. You will see that user **Frank Zappa** has been added to the group/user names. Select **Frank Zappa** and you can modify his permissions for this file. *What permissions does he currently have?* Click the **Deny** checkbox under **Full Control**. *What happens to his permissions?* Click on **OK**. Click **Yes** to the warning popup window. When you return to a1.txt’s properties window, see what each user’s permissions are now for this file. **Close** this property window. Repeat this operation for a2.txt except only deny zappaf Write permission to the file. Leave a3.txt alone.
   2. In the File Explorer’s left pane, right click on your **user account** and select **Properties**. Select the **Security** tab. We will give all Users access to this directory. Similar to what you just did, from the **Security** tab select **Edit…** and from the Permissions window, select **Add…** and then from the Select Users or Groups window, enter **Users** and click on **Check Names**. It should complete the name for you. Click **OK** to close this window and leave the default permissions as they are now set for Users (Allow Read & execute, List folder contents, Read) and click **OK** to close this window and **OK** to close the Properties window. Let’s see if this worked. From the Windows start button, select your user name and then Frank Zappa from the list of users to switch to his account (remember that you changed his password to dweezil). Open **File Explorer** and go to **your personal user home directory** (not zappaf’s). You might recall in the last lab that when you tried to access zappaf’s directory you were initially denied and that it took you serving as administrator to gain access. *Why did that not happen to zappaf trying to access your home directory?* Try to **open** each of a1.txt, a2.txt and a3.txt and in each, **add or alter** the text and save the file. *What happens in each case (a1, a2, a3)?* If you cannot open a file, report this. If you can open and change the file but not save it, you will be asked to save it elsewhere, try to save it under a new name but in the same directory and report what happens. *When you attempted to save this directory, you were unable to. Why? If we changed the permissions of this directory to allow people to write to it, would that have resolved the issue? Explain as best you can.* As zappaf, **open a1.txt’s permissions** and look at the **security** tab. Do the same for a2.txt. *How do they differ?* Sign out as zappaf and sign in again as yourself.
   3. Using the File Explorer, change a2.txt’s permissions by adding the **Students** group to have rights to modify, read & execute, read and write to this file. The steps will be very similar to what you did in step 1b. Now sign in as underwoodr (recall her password is xylophone). Being that this is the first time she is logging in, she will be forced to change her password. Change it to whatever you like. Once logged in, bring up **File Explorer** and select Computer 🡪 Local Disk 🡪 Users 🡪 Your User Directory (not underwoodr’s). Double click on **a2.txt**. Add some text and **save** the file. *How does this differ from what happened in step 2 when zappaf tried to modify this file? Summarize what you have learned about the difference between the owner’s permissions, members of the groups permissions and other members permissions to files.*
   4. As underwoodr, in the File Explorer, right click on **This PC** and select **Manage** to bring up the Computer Manager. *What happens? Why did this happen (hint: consider who you are logged in as).*  Select **No** to the User Account Control window. At this point, you can sign out of both user accounts and shut down your VM.
2. Start your Linux VM. Log in as yourself and open a terminal window. In Linux, permissions are specified on each file and directory broken into three categories: the owner (u for user), the group owner (g) and the rest of the world (o for other). For each of these, there are three types of permissions: r (read), w (write) and x (execute). For files, these are self-explanatory. For directories, read means you can ls the contents, write means you can save files to that directory and execute means that you can cd into the directory. To view a file’s permissions, use the ls –l command. A file/directory’s permissions might look like this –rwxrw-r-- where the first character (- in this case) is the type of entity (- for file, d for directory, l for symbolic link, there are several others). In the above example, it is a file with rwx privileges for the owner of the file (u), rw for members of the file’s group, and r for the rest of the world. To change permissions, we use the chmod command. Read about this in chapter 6 and also view chmod’s man page. Change directories to **/etc**.
   1. Type **ls –l \*.conf** **| less** to view the permissions of files in this directory. Nearly all of these files have permissions of rw-r--r--. *Explain what this permission allows for user, group and other. Which of these conf files in this directory provide no read access for the world (o)?*
   2. Perform an **ls –l** on each of the following items. *For each, what is its 3-digit permission?* For instance, a permission of –rw-r--r-- would be 644. Also for each, specify whether it is a directory or file. abrt, audit, auto.net, securetty, shadow, sudoers.
   3. Type **ls –l | less**. *Which of the files in this directory have different owner from group owner?* Type **ls –l | less**. *Which of the files and directories in this directory can only be accessed by root?*
   4. Change to your home directory. Type **ls –al | less**. The long listing, aside from providing permissions and ownership, also lists the number of hard links pointing at the item. This is the number after the permissions. *How many hard links are there for your file .bashrc? For the directory Desktop? For the directory .mozilla?* Now look at the permissions (if necessary, redo the ls –al command). *How many of your files have access set to just read/write for you and nothing for group or world? How many have permissions of 644?* Recall that permissions that start with a “d” are directories. *How do the directory permissions differ from the files? That is, what access type is commonly provided for a directory but not a file? Why?*
   5. In your home directory, type **touch f1 f2 f3 f4 f5 f6** to create six empty files. Enter the command **ls –l***.* *What initial permissions do the files have, expressed as a three-digit number?* Type **chmod 000 f1 f2** and then **ls –l**. *What permissions do these two files (f1, f2) have now?*
   6. There are there ways to use chmod, the 3-digit approach from step 2e, using ugo+/- as in u+x,g-w, and ugo= as in u=rwx,g=rw,o=. As we continue, we will explore each of these starting with +/-. The way this command works is to add or subtract permissions from what the item current has. Current f3 has permissions rw-rw-r-- or 664. Alter f3 to allow members of the group to write to it and remove read access from the world using **ugo+/-** method. *What command did you enter?* The ugo= approach allows you to change all permissions to a new set of values where you would specify u=…,g=…,o=… with no spaces where … are any combination of r, w, x or nothing as in u=rwx,g=rw,o= Alter the permissions for f4 so that only the owner has access and the owner has read, write and execute access, using the **ugo=** method. *What command did you enter?* Alter the permissions for f5 so that everyone has read and write access and the owner has execute access, using the **3-digit** method (such as chmod 000, to compute the 3 values, use 4 for read, 2 for write, 1 for execute, view the textbook for help). *What command did you enter?* Change the permissions for f1 so that the owner has read/write access and group has read access, using the **3-digit** method. *What command did you enter?*
   7. Enter the command **chmod u+x,g-x f1 f2 f3 f4 f5**. *Summarize what this instruction does.* Another option for chmod is to use a= or a+/- to mean “all users”. Change f6’s permissions so that all users can read it only using **a=**. *What command did you enter?* Type **ls –l f\***. *What are the permissions of all 6 files now?*
   8. Type **ls –al**. Along with the files and subdirectories, two of the other items listed are denoted as . and .. These are the current directory (your home directory) and the parent directory (/home) respectively. *What are their permissions?* Change your home directory’s permissions so that members of the group have rwx and others have r-x privileges using the **3-digit** approach. *What command did you enter?*
3. Su to zappaf (his password is gail). You should still be in the same directory, but make sure by typing **cd ~*username*** where *username* is your username (not zappaf).
4. Type **cat f1**. *What happens?* Type **cat f3**. *What happens?* Type **cat f5**. *What happens?*
5. Type **ls –al .cache.** *What happens?* Type **ls –al .dbus**. *What happens?* **Exit** from zappaf (type **exit**) back to your user account.
6. *Summarize what you have learned here regarding permissions and being able to access files and directories of other users.*
7. In your home directory, type **echo “echo Hello World > output” > script1** (type this exactly as shown). This creates the file script1 whose contents are echo Hello World > output. This is a script file that, when run, outputs “Hello World” to the file output.
8. To run a script, type **./*scriptname***. So in this case, you would type **./script1**. Run the script. *What happens?* Why? *What are the permissions for the script1 file?* We need to change the script to be executable by us and the world. Change the script’s permissions to be –rwxr--r-x using the **3-digit** approach. *What command did you enter?* Run the script now. **Did it work?** Type **ls –l** and see if the file output is there. *What is output’s permissions?*
9. su to zappaf. As zappaf, run the script. *Did it work?* Zappaf has permission to run the script *but what doesn’t zappaf have permission to do?* (hint: think about what the running script does in terms of your own personal home directory). Exit from zappaf.
10. As yourself, type **echo “echo Hello World” > script2**. This creates a second script which is nearly the same as the first except that the output goes to the terminal window, not a text file. Change this script’s permissions appropriately. *What command did you enter?* Su to zappaf and run script2. *Did it work?*
11. The chown and chgrp commands allow you to change the file’s owner and group owner. We will explore chgrp. Recall that we set up a group called students that included both your account and zappaf. Note that you must be root to use either of these instructions.
12. su to root. Enter the command **chgrp students f1 f3**. Exit from root. Type **ls –l**. You should now see that f1 and f3 have a group of students. su to zappaf and type **cat f1**, **cat f3**, **cat f4**. *Which of these does zappaf have read access to?*
13. Now type **echo hello > f1** and **echo hello > f2**. *Which of these does zappaf have write access to?*
14. *Summarize what you have learned about owners, groups and permissions.* You may log off of your Linux VM.

# Lab 7: The Bash Shell

Goals: to learn more about command line entry in Linux; to learn many of the shortcuts available in support of command line entry in Linux.

Introduction: the shell offers the user an interpreted environment which includes the ability to define your own constructs (aliases, variables). In Bash, you have a number of powerful tools available to make interaction easier. In this lab, we explore many of the Bash features.

Time to completion: 2 hours

Before starting, review chapter 9 Bash Shell Editing Features, Exploring the Bash Interpreter and Personalizing Your Bash Shell.

Start your Linux VM and open a terminal window. You will not work with your Windows VM in this lab.

1. Let’s start by looking at the history list and recalling instructions.
   1. Type separately the commands **whoami**, **ls –al**, **ls**, **pwd**, **who,** **ps** and **ls /etc**. Each of these is a simple program that directs its output to the current window. *In the output of* ***ps****, what commands to you see under the “CMD” column?*
   2. Type **history**. This will display your history list. Review chapter 9 regarding how to recall commands from the history list. NOTE: since you have worked in Linux before, your history list may not be starting from today’s work. Type **!!**. *What happens?* From your history list, find the number for the who command from step 1 and type **!*n*** where *n* is that number (this may or may not be the first instruction). *Did it work?*
   3. Assume you want to re-execute the instruction whoami. You can do this by searching the history list for the number, by typing **!*str*** where *str* is a string that captures the instruction, or by pressing control+p repeatedly until you reach the instruction. We will try the !str approach. Type **!w**. *What instruction is recalled? Why?* Try **!whoa**. *Did it work? Why were the extra characters necessary?*
   4. Let’s assume we want to redo the ls of our directory. Type **history** again and you will notice ls appears 3 times (recently), ls –al, ls, ls /etc. *How can we recall the ls that does not list /etc?* Try it until you get it right. Hint: don’t forget that you can move forward and backward through the history list using control+p and control+n.
2. Next, we will explore environment variables. These are variables established by the operating system when you start a new Bash session. These variables are all fully capitalized. To see the value of a particular variable, type **echo $*VARNAME*** as in **echo $USERNAME** or **echo $SHELL**. You can also see all defined environment variables by typing **env**.
   1. *What are the values stored in USERNAME, PWD, MAIL, HOME and DISPLAY?*
   2. Type **cd /etc**. *What is the value of PWD now? What is the value of OLDPWD?* Type **cd $OLDPWD**. This should return you to your previous directory. Type **pwd** to make sure. Notice that the pwd command and the variable PWD are the same while OLDPWD retains the name of the most recent directory visited before the current one.
   3. We already explored the PATH variable in lab 4. Recall that it stores all of the directories that will be tested when you issue a command in case the command is not present in the current directory. To add to your PATH variable use the notation PATH=$PATH:newdirectory as in PATH=$PATH:/usr/sbin. Examine your PATH variable. Provide a command to add the following directories: /usr/games and /opt (this directory is often used for 3rd party application software). Try to add these directories with a single command (you do so by separating each directory with a colon).
   4. *What is the value stored in the PS1 variable?* This defines your prompt. Refer back to the textbook and *explain what each of the parts defines* (e.g., \u is the user’s username).
3. You can define your own variables as well. We use an assignment statement for this. The general form is ***VAR*=*VALUE*** as in **AGE=21** or **NAME=Frank**. If the item on the right hand side of the equal sign has spaces in it, enclose the items in “” as in **NAME=“Frank Zappa”**. Never put blank spaces on either side of the equal sign like **NAME = Frank** as this will cause an error.
   1. Define the variables FIRST and LAST to store your first and last names respectively. Type echo $FIRST $LAST to make sure it worked. *What assignment statements did you enter to define FIRST and LAST?* Now type **FULLNAME=“$FIRST $LAST”**. *Explain why we had to use the “” and $ in this statement.*
   2. Define the variable AGE to be your current age. Type **echo $AGE**. Did it work? Now type **AGE=$AGE+1**. This should set your AGE to be 1 greater, for instance if you are 20, it would change it to 21. Type **echo $AGE**. *What is the output? Why didn’t this work? Since it didn’t add 1 to AGE, what did it do?* The proper way to perform an arithmetic operation is put the arithmetic inside $((…)) notation. So instead, you would do AGE=$((AGE+1)). Set AGE back to your initial age and then type this new assignment statement. Make sure it works!
   3. The echo statements and assignment statement can handle literal values (e.g., A=1, echo Hello) and values in variables (A=$B, echo $A). You can also insert Linux commands so that the commands’ outputs are either echoed or stored in a variable. We do this by placing the Linux command in either ` ` marks or $( ) marks. Type **date**. We want to output the date using this instruction but do so with a message like “Today is…”. Type **echo Today is date**. *What do you see?* Type **echo Today is `date`**. *What does that output?* Now try **date=`date`** followed by **echo Today is $date**. Now type **my\_disk\_usage=`du –sh ~`**. *What did this instruction do?* Use echo to output the value stored in my\_disk\_storage. *What value is stored in the variable?* Now type **$my\_disk\_usage**. *What is the output?*
   4. Now type **echo The listing of your directory is `ls`**. *How does the listing appear differently from a normal ls instruction?* Type **mylist=`ls`** and then **echo $mylist** (you should see the same output). *What can you conclude about how the output of the ls command when it is in an echo statement versus issuing the command from the command line?*
4. By default, Linux instructions receive input from a disk file and send output to the terminal window. Redirection allows you to alter the behavior so that input can come from keyboard and output can either go to a disk file or another command. We use the redirection operators to handle redirection. These operators are <, <<, >, >>, | (review chapter 9 to see what each does). We will use the cat program to experiment with some of these. Make sure you are in your home directory (~) before starting these steps.
   1. Type **cat /etc/passwd**. You should see its contents appear in your terminal window. Now try **cat /etc/passwd /etc/group**. This will display both files, one after the other, to the terminal window. By adding a pipe to either more or less, we can force the instruction to pause as in **cat /etc/passwd /etc/group | less**. Try this. Now type **cat /etc/passwd /etc/group > myfile.txt**. *What appears in your terminal window?* Type **ls –l**. The new file myfile.txt should appear. *How big is this file? What does > do?* Type **cat /etc/fstab >> myfile.txt**. Now examine the contents of myfile.txt. *What changed?* Type **cat /etc/fstab > myfile.txt** and look at its contents again. *What can you conclude between > and >>?*
   2. Since most programs expect input to come from a file, the redirection < is not very useful as it says “obtain input from the following file”. We will use it to send input files to scripts that we write but otherwise, < has little usage. The << redirection says “obtain input from keyboard instead of a disk file. Type **wc myfile.txt**. The wc program performs a “word count”, outputting the number of characters, words and lines found in the given file. *What values did you get?* Now type **wc << quit**. This form of redirection forces wc to obtain its input from whatever you type in the keyboard, ending when you type in the word **quit**. You will see the prompt has changed to be >. **Type something and press <enter>**. You will receive another > but no output. **Continue to type and press <enter> and when you are done, type quit** on a line by itself. *What was the output?*
   3. Let’s combine redirection as follows. Type **cat << quit > file1.txt**. Now type in a few lines of text, ending with the word quit on a separate line. Repeat this using file2.txt instead of file1.txt. The result should be two files of text of whatever you typed. Now type **cat file1.txt file2.txt > file3.txt** and finally **cat file3.txt | sort**. The output of the last command will be the lines of file3.txt (which consists of the lines of the first two files) sorted alphabetically. Now try this one: **cat << quit | sort >> file4.txt**. *Explain what each form of redirection, <<, |, >>, does in this statement. How would the command have differed if you had used > instead of >>?*
5. Here, we will briefly look at command line editing. Review the forms of command line editing as shown in chapter 9. For instance, control+a to move to the beginning of a line, escape+f to move forward to the end of the current word.
   1. You should have 4 files in this directory named file1.txt, file2.txt, file3.txt and file4.txt. Imagine that you wanted to rename them all to be of the form File1.dat, etc. Type the command **mv file1.txt File1.dat**. Now rather than repeat this statement 3 times with slight variations for the other 3 files, let’s try command line editing. Type **control+p** to recall this instruction. Type **control+a**. *What does this do?* Now type **escape+f** twice. *Where does your cursor now lie?* Press **backspace** and then type **2**. This should replace 1 with 2. Now type **control+e**. *Where is your cursor?* Type **control+b** 5 times. Your cursor should be over the 1 in File1.dat. Type **control+d** and then **2**. Your line should now read **mv file2.txt File2.dat**. Press **enter** if correct. Notice that you did not have to move to the end of the line to press enter. Do not modify the names of file3.txt or file4.txt.
   2. Type the command **ls –l /etc**. You realize that you really meant to do an ls –al /etc. Rather than typing the command anew, you want to use command line editing. *Specify what keystrokes you would use to recall the instruction and then modify it.* You can test this out while you answer the question to make sure you have it done correctly.
6. Another useful feature of the Bash shell is tab completion. This will allow you to specify a partial directory or file name and let Bash complete it for you when you press the tab key. Let’s see how it works.
   1. From an empty command line, type **ls /etc/sysco<tab>**. *What happens?* Press the **enter** key. Now type **ls /etc/sys<tab>**. *What happens?* Bash cannot complete a directory/filename if the string you have entered is not unique and there are several items in /etc that start with sys. Type **control+a** and **control+k** to delete what is on this line. Now type **ls /etc/sys<tab><tab>**. *What happens?* Again, type **control+a** and **control+k** to clear the prompt. NOTE: if you type control+k, it “kills” whatever is on the command line from the point of your cursor forward. If you type control+y, it “yanks” back the last thing you killed. So if you accidentally type control+k, you can undo it immediately by typing control+y.
   2. Type **ls /etc/<tab><tab>**. This time, you get a different response. Answering y will give you the full listing of everything that matches (which is all of the files) and answering n and you go back to your prompt. Type **n** and then press the **enter** key.
   3. Type cd /etc for these next two steps. Type **cat p<tab><tab>**. *What happens?* Now type **cat pas<tab>**. *What happens?* Hit the **enter** key. Recall from earlier in this lab that you added your home directory to your PATH variable and that you have files file3.txt and file4.txt. Will tab completion complete file names based on your PATH variable? Let’s find out. Type **cat file<tab>**. If there were multiple files that start with “file”, Bash would beep at you alerting you to do file<tab><tab> But that does not happen here. *What happens?* This filename completed is a file in /etc, so tab completion ignores your PATH variable and only operates on the current working directory or the directory as specified in the command as in cat /etc/file<tab>.
7. Next, we look at aliases. These are definitions that allow you to specify commands more easily. Type **cd** ~ to go to your home directory.
   1. Type **alias lsa=‘ls –al’**. Now type **lsa**. *What happens?* You have defined an alias to simplify a task. Now try **alias ..=‘cd ..’**. Type **..** and then **pwd**. *Where are you?* Notice that an alias does not need to start with a letter but there are some restrictions in naming our aliases. We use aliases for a number of reasons: to shorten long commands, to shorten commands that are hard to remember (for instance, you might use alias empty=‘find ~ -empty –exec rm {} \;’ which uses the find command to locate and delete any empty files), and to handle typos, for instance by defining alias mr=rm if you happen to type mr in place of rm a lot).
   2. To see all of the aliases, type **alias**. *How many aliases are defined?* Why are there some definitions that you did not enter yourself? We will explore that later.To remove an alias, type **unalias *name***. Type **unalias ..** Now try **..** *What happens?*
   3. Type **bash**. This will start a new bash shell. Type **lsa**. *What happens?* Why? Because the alias you had typed is only available in that particular session. Again, later, we will explore where aliases can be defined. Type **exit** to return to your previous bash session. Type **alias** to make sure your alias of lsa is still defined. By the way, the new bash session also did not have any of your previously defined variables including the modified PATH variable from step 2c.
8. Wild cards allow you to substitute a list of files in Linux instructions by specifying a single string of characters. For instance, \* means “match any file” and \*.txt means “match any file ending in .txt”. The Bash interpreter performs an action called globbing or filename expansion before executing the instruction. In globbing, Bash replaces \*.txt with all items in the current directory that match. If there are four files a1.txt, a2.txt, file1.txt and foo.txt in the current directory, the instruction ls \*.txt becomes ls a1.txt a2.txt file1.txt foo.txt. The wildcard characters are:

\* - match anything

? – match any one character

[chara-charz] – match a character in the range from chara to charz

There are other wildcards available, but we will stick with these. Let’s give this a try.

* 1. Type **cd /etc**. Type **ls –d c\***. The –d tells ls to display directory names but not their contents. *How many items are listed with c\*? How many are directories?* Change directory to /bin. Type **ls c[hp]\***. *What items are displayed? What does the notation c[hp]\* mean?* Now try **ls m\*n\***. *Which files are displayed? What does the notation m\*n\* mean?* Change directory to /usr/bin. Type **ls at\***, **ls at?** and **ls at??**. *What results did you get for the three commands? How does the ? differ from the \* in these ls statements?*
  2. You want to match any file of whose name is file#.extension where the ‘f’ in file can be lower or upper case, where # is any character such as 0 or a, and the extension can be anything like .txt or .dat or even .DAT. *Write an ls command do to this.*
  3. Remaining in /usr/bin, write an ls command to list all file names that have the number 2 in them, piping your result to wc –l. *How many files were found? What was the instruction that you entered?* Now repeat this but list all files that have a number in their name anywhere. *How many files were found? What instruction did you enter?*

We will look at regular expressions in a couple of labs and they are similar to these wildcards but have slightly different meanings.

1. To wrap up this lab, we look at some initialization files. Whenever you log in, your shell executes the script /etc/profile. Whenever you start a new Bash shell (including when you type bash inside your terminal window), Bash executes the scripts /etc/bashrc and your own .bash\_profile and .bashrc. Let’s examine the contents of the files in your directory.
   1. Make sure you are in your home directory. Type **cat .bash\_profile** (note: use tab completion once you type the \_!) You will see an if statement that basically says “if .bashrc exists, run it now” followed by a statement that adds $HOME/bin to your PATH variable. *What is $HOME/bin?* Use echo to find out the value of $HOME. Look at your PATH variable and you should see this directory as one of the last entries. Finally, the script exports PATH to make the variable available outside of the script. This is how a variable, defined in a script, can become an environment variable. Now type **cat .bashrc**. This script tests to see if /etc/bashrc exists and if so, it is executed. Otherwise, this script is empty. Type the following: **echo “alias lsa=‘ls –al’” >> .bashrc**. Now if you examine your bashrc file, you will find this alias added to it! Type **bash** to start a new bash shell and type **lsa**. *Did it work?* Type **exit** to return to your previous shell session.
   2. The /etc files for profile and bashrc are for the system administrator to set up global definitions – that is, variables and aliases that the system administrator wants every user to have. The .bashrc and .bash\_profile files are for the user to add their own definitions. We would prefer that all users use rm –i instead of rm. If we want to force this on users, we might define alias rm=‘rm –i’. *Should this definition go in one of the /etc files or in one of the user’s own files? Why?*

You may shut down your Linux VM.

# Lab 8: vi in Linux

Goals: to learn how to use the vi text editor.

Introduction: vi (vim) is a very useful text editing tool for Linux users. You might find it useful when creating simple text files, writing shell scripts, defining data files, or as a system administrator, modifying system text files. Although gedit and emacs are usually available in Linux and might be deemed as easier, you might find that you are logged in from a text-only interface in Linux so that you could not use gedit, and you might find emacs unavailable. Thus, learning vi is essential for any system administrator.

Time to completion: 60-90 minutes

Before starting, review chapter 9 Text Editors.

Start your Linux VM and open a terminal window. You will not work with your Windows VM in this lab.

1. From your home directory, type **vi first.txt**. Your vi editor takes control of your terminal window and fills it entirely. Your cursor will be at the top of the editor on a blank line followed by lines that consist of the tilde (~) character. These indicate no lines (not blank, just non-existent). At the bottom left you will see the file name and [New File] to indicate that it is a new file. In the lower right, you will find cursor location details. You are currently in “command mode” which means that keystrokes are interpreted as commands rather than text to be entered.



* 1. To start editing the file, you have to switch to *insert mode*. There are multiple ways to enter insert mode. The simplest, is to type **i** (lower case). Do so. You will see the filename disappear from the bottom of the file to be replaced by -- INSERT --. To leave *insert mode* and return to *command mode*, type **<esc>**. In insert mode, type your ***name*<enter>**, ***your course number/section*** (e.g., CIT 130.001)**<enter>**, ***the date*<enter><enter>**. Your file might now look like this:

Richard Fox

CIT 130.001

July 9, 2016

Press **<esc>**. *What do you see along the bottom of your terminal window now in both the left side and right side?* Save the file by typing **:w** (w for write). *Now what do you see along the bottom left?*

* 1. Cursor movement (other than the arrow keys which may or may not work) can only be performed in command mode. To move the cursor to the beginning of a line, type ***#*G** where *#* is the line number. **G** by itself takes you to the end of the file. Type **1G**. *Where is your cursor?*  Type **G**. *Where is your cursor?*
  2. Other cursor movements are: j (down), k (up), h (left), l (right), w (forward 1 word), b (back 1 word), H (top of current screen), L (bottom of current screen), 0 (first character on the line), $ (last character on the line). *From where you are, how would you reach the end of the date line? From where you are, how would you reach the period (or space) between your course number and section?*
  3. To reach the insert mode, we can also use O to insert a new line above the current line or o to insert a new line below the current line. Move your cursor the line with your course section and type **o**. This should insert a blank line between your course section and the date and place you into insert mode. Type your instructor’s name in this line and press <esc> to exit insert mode. You want to also add the lab number, immediately before this current line with your instructor’s name. *How do you insert above?* Do so and enter Lab 8 (or whichever lab this is if this is not your 8th lab). Exit insert mode again. Save the file (:w).
  4. We want to add your email address immediately below your name. *From where you are, how will you do this? Explain the keystrokes precisely.* Go ahead and do this and then exit insert mode.
  5. Let’s assume you got the date wrong by one day. Move your cursor to the last digit of the date. If there is only one digit, just put the cursor there. You want to replace the date’s digit with the next digit (or, if the date ends with a 9, the previous one). So for instance, July 9, 2016 will become July 8, 2016 or if you have July 15, 2016 it will become July 16, 2016. Once you have the cursor positioned, rather than typing i, we will use r. This enters replace mode for 1 character only. Type **r** and then type the **replacement digit**. Notice that since you are replacing 1 character, you do not have to use <esc> to leave replacement mode.
  6. Type **G**. *What did this do?* Type **o**. *What did this do?* Type a few sentences about whether you have enjoyed exploring Linux yet or not and how you feel about Linux compared to Windows. Make sure you have added at least 3-5 lines to the file but no more than 5. DO not press the enter key while typing your sentences. Words will wrap around when you reach the right hand margin. When done, type **<esc>**. Move your cursor to the beginning of the last sentence that you added. Type **R**. *What changes in your vi editor window?* Now begin typing a new sentence. *What happens?* Make sure the new sentence is as long or longer than the previous last sentence. When done, press **<esc>** to leave this mode.
  7. Move the cursor to the 5th line of the file. *How did you do this?* Type **dd**. This deletes the entire line, moving it into a “cut and paste” buffer. Move the cursor above your instructor’s name. Type **P**. This pastes whatever was cut above the current line. This should have moved the date before the instructor. Let’s assume that you don’t like this. To undo the last step, type u. To continue to undo, type more u’s. So type **u u** and it should undo the paste and the cut.
  8. You can issue a command repeatedly by typing ***#command*** where *#* is a number and *command* is the keystroke(s). The previous 2 undos could have been done as **2u**. Type **3j**. *What happens?* Type **4w**, *what happens?*
  9. Return to the first line in the file. *How did you do this?* Type **i** to enter insert mode. Before your first name, add a proper title (e.g., Mr., Miss, Mrs., Dr.) and a space before your name. For instance, I might change mine to

Mr. Richard Fox

Type **<esc>**. Let’s imagine that you don’t like what you did. You could undo this of course, but let’s look at how to delete this. Move to the start of the line by typing **0**. Now type **dw**. This deletes the current word from the cursor position to the word or the next punctuation mark. If it left the period and a space, we can delete these two characters by using the deletion key, x. To delete two characters, you can type **xx** or **2x**. Delete any remaining characters until you reach the beginning of your name. Don’t forget if you make a mistake, use u until you have undone the mistake and then try again.

* 1. We saw that to save the file we use :w. Let’s use “save as” by changing the name. To do this, from command mode, use **:w *name***where *name* is the new filename. Save this using the name second.txt. *How did you do this? What is at the bottom left of your editor now?*
  2. Move the cursor to the beginning of your test that you added step 9. Type **dd** followed by **10y**. *What did this do?* Notice that the entire several sentences were cut as one because you did not press the enter key when typing them in so the entire set of sentences is treated as one line.
  3. The reason we did step 1m in part was to add enough text so that the text in the file runs passed the bottom of the window. *What does the bottom right of your editor say now? What does this mean?*
  4. Type 2u to undo both the paste and the copy. Rather than cut and paste, you can copy and paste. Copy uses yy (instead of dd for cut), followed by p or P for paste. Copy the paragraph and paste it twice. How did you do this?
  5. Move the cursor to the beginning of the line with your instructor’s name. Move the cursor to be on the first letter of your instructor’s last name with one keystroke. What keystroke did this? We will replace your instructor’s last name with all capital letters. The keystroke cw will place you in delete the current word and place you INSERT mode. Type **cw**, your **instructor’s last name** in all capital letters and **<esc>**.
  6. To end this portion of the lab, we will exit this editor session. Type **:q** to quit. *What happens?* We must either save the file before exiting or tell vi that we don’t want to save the file and exit anyway. To save and exit you can type **:w** and then **:q**, or **:wq** combined. Instead, let’s exit without saving. Type **:q!**.

1. We will start a new file. You will have to refer back to the earlier steps (or the textbook) for cursor movement and editing commands as they will not be specified here. Type vi. Since we did not name this file, we will have to name it later. Notice that the opening screen looks different, we have the vi “splash” screen. Type **i** to enter insert mode. Type the following:

Illinois

Indiana

Ohio

Michigan

Wisconsin

Minnesota

Pennsylvania

Iowa

Nebraska

* 1. Exit insert mode. Using cut (dd) and paste (p or P), rearrange these lines so that the states are in sorted order. *Explain how you did this.*
  2. Insert North Carolina in its proper location alphabetized. Copy North Carolina and paste it below that line. *How did you do this?* Edit the new line and replace North with South. *How did you do that?*
  3. With South Carolina out of place, we need to sort it. We don’t however have to resort to the steps you did in part 20. Instead, vi has its own sorting command. We can call upon a vi command by typing **:*command***. The command we want is sort, so type **:sort**. *What happened?* Add New Mexico anywhere. Copy and paste it. Edit the second version by changing Mexico to Jersey. You should be able to do this using either R (replace) or cw. When done, resort the file.
  4. Now we want to save the file. Type **:w**. *What happened? Why?* Remember how to save the file with a filename. Do so saving the file as states.txt. You can exit this file.

1. Start a new file called schedule.txt. Use vi to input and edit your current class schedule. If you are not taking at least four classes, make a few up to give you at least four classes. For each line, list the course number/section, days, start time, end time, credit hours, and class location. Separate each of these with a tab. For instance:

CIT 130.003 MWF 10:00 am – 10:50 am 3

Save your file and exit. After examining your file, you decide you want to make some changes. Go back and edit the file so that the courses are listed in order of days first and then times. For instance, list all MWF (or MW) classes first, then TR. And within each set of days, order by time. Save and exit the file. Return to the file and add a header row of:

CLASS DAYS TIME HOURS

Make sure the headers line up correctly with the rows underneath. Save your file and exit. *As part of your assignment, hand in the 3 files you created (second.txt, states.txt, schedule.txt) with your answers.* Log off of your VM.

# Lab 9: Regular Expressions

Goals: to understand what regular expressions are and how to use them to solve problems; to learn how to use the egrep program.

Introduction: regular expressions are a powerful tool that most Linux users and system administrators will use often. Here, we examine what regular expressions are and put them to use using the Linux grep tool. Grep is a program that will search one or more files for any lines that match a given regular expressions. It responds with the lines of the file(s) that matched. You can alter grep’s behavior to output line numbers along with the lines, or just output the number of matches. Grep itself uses only the normal regular expression set of metacharacters. We will use egrep throughout this lab which uses the extended set of regular expressions. The early portions of this lab step you through many examples without having any questions for you to answer. You will answer the questions in parts 3 and 4 only.

Time to completion: 2 hours

Before starting, review chapter 10 Metacharacters and The GREP Program.

Start your Linux VM and open a terminal window. You will not work with your Windows VM in this lab.

1. cd to the /etc directory and su to root. We will start with some easy regular expressions.
   1. Find any file that contains at least one digit in the text: **egrep ‘[0-9]’ \***
   2. Find any file that contains at least one upper case letter in the text: **egrep ‘[A-Z]’ \*** Let’s change this to output just the file names by adding the option **–l**. Try it.
   3. Find any file that contains a sequence of at least 12 upper case letters: **egrep ‘[A-Z]{12,}’ \*** Notice to find “any”, we only need [A-Z], but to find “at least” we use {x,} where x is the minimum number we are looking for.
   4. Find any file that contains the notation VAR=VALUE where VAR is a variable name and VALUE is a numeric value assuming all variable names will be composed of upper case letters: **egrep ‘[A-Z]+=[0-9]+’ \***
   5. Find any IP addresses (note: chapter 10 has a lengthy discussion for how to develop a regular expression for IP addresses, we will use a simpler one here): **egrep ‘[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}’ \***. Can you figure out this regular expression? Give it a try.

NOTE: some students will think an IP address octet (0-255) should be recognized using the notation [0-255]. Again, see the textbook for a discussion of why that is incorrect. Try it out by using **egrep ‘[0-255]\.[0-255]\.[0-255]\.[0-255]’ \*** and compare your results to 1e above.

* 1. Many of the files in this directory are scripts. Scripts use # to denote a comment. Most files that have comments also include blank comment lines, that is, lines that contain nothing but # (or in some cases, ## or ###). Let’s look for all of the files in /etc that have a blank comment line. **egrep ‘^#$’ \***. What does this look for? Lines that start with # and then end with nothing else in that line. Rather than listing all of them, we only want to see the file names, so use **egrep –l ‘^#$’ \***. One such file is fstab. If you look at it, you will find 3 lines that are just a #.
  2. Another thing to look for are blank lines. These can be found using ‘^$’. How many blank lines exist in the files in this directory? If we just do **egrep ‘^$’ \***, we won’t see anything but file names followed by blank lines! Try it. Instead, let’s pipe the result to wc –l. Type **egrep ‘^$’ \* | wc –l**. I found 246 blank lines, how about you?
  3. One of the files in /etc is bashrc. This script is executed whenever someone starts a new Bash shell. There are a few things we might want to search for in this file. How many assignment statements are there? This will look like VAR= where the VAR is upper case letters, so we can use the following regular expression: [A-Z]+=. How many if statements are there? An if statement always starts with if [. Since [ is a metacharacter, we have to use **if \[** as our regular expression. How many functions are there? A function is defined using notation name ( ) {. We can’t directly use ( ) or {, so we again have to use \ before them. Here are three egrep statements to try:

**egrep ‘[A-Z]+=’ bashrc** I found 7

**egrep ‘if \[’ bashrc**  I found 9 (although 1 was inside a comment)

**egrep ‘\(\) \{’ bashrc**  I found one (pathmunge)

* 1. Let’s use some regular expressions to explore the passwd and group files. First, how many user accounts have a home directory in /var? **egrep ‘/var’ passwd**. Notice that we use /var instead of var because it’s possible that someone’s user name might contain var as in vargase. Do any of the users use csh instead of bash? **egrep ‘/bin/csh’ passwd**. Alternatively, which user accounts do not use bash? **egrep –v ‘/bin/bash’ passwd**. The –v option “inverts the match” by returning all lines in the file that did not match. How many groups have a GID less than 100? **egrep ‘:[0-9]{1,2}:’ group**. If you look at the group file you will find that each GID is surrounded by : symbols. Notice that our regular expression was not just [0-9]{1,2} because this would match a GID like 123. If you don’t remember why, review the chapter. How about groups that have more than one user? Every row of the group file looks something like this:

groupname:x:GID:users

where users are separated by commas. The only commas we will see in this file then occur if a group has multiple users. So we solve this by using **egrep ‘,’ group**. How about which groups have no users assigned? In this case, after the GID: there is nothing. So we can just search with **egrep ‘:$’ group** to indicate that the : should end the line.

1. cd to your home directory. We will use two of the files that you created in the vi lab, the states.txt and schedule.txt files.
   1. How many states are two words? Search for a blank space. **egrep ‘ ’ states.txt**. If you happen to have included a blank space on a line, say at the end of a line, this would catch such an entry. So more properly, we want to make sure that the blank does not end the line. We could enhance our expression using **egrep ‘ [[:alpha:]]+’ states.txt**. This says “find any line with a blank followed by one or more letters”.
   2. How many states have two (or more) consecutive vowels? **egrep ‘[aeiou]{2}’ states.txt**. What about Iowa? It isn’t included because we didn’t specify upper case letters. We can fix this using either egrep **‘[AEIOUaeiou]{2}’ states.txt** or **egrep –i ‘[aeiou]{2}’ states.txt**.
   3. How many states end with the letter ‘a’? **egrep ‘a$’ states.txt**. How many states start with the letter I? **egrep ‘^I’ states.txt**.
   4. Now, focusing on the schedule, which courses meet on MW but not MWF? We can’t just use **egrep ‘MW’ schedule.txt** because a line that includes MWF will match the MW part. We have to find MW but not followed by an F. The easiest way to do this is through **egrep ‘MW[^F]’ schedule.txt**. This says “find any line that has a MW not immediately followed by F”. If you used lower case letters, you might use mw[^f] or use –i.
   5. What classes start on the hour (:00)? We can’t just search for :00 because we don’t want to match classes that end on the hour, just start. The time looks like this: hour:minute a/pm – hour:minute a/pm. So we want to match :00 that appears before a/pm -. The ending time does not precede a hyphen. This gives us the regular expression **‘:00 [ap]m –’**. If we used AM or PM, we could either use [AP]M or use the –i option. Our statement is **egrep –i ‘:00 [ap]m –’ schedule.txt**
   6. Finally, let’s find all classes that are either CIT, CSC or INF classes (you can substitute your own designators if your classes use different designators, at our university, we use CIT for computer information technology, CSC for computer science and INF for informatics). **egrep ‘(CIT|CSC|INF)’ schedule.txt**.
2. From here on out, you are on your own. First, you will search the Linux dictionary file (/usr/share/dict/words) for words that fit a given pattern. *Your answers for part 3 will be the egrep instructions you come up with*. Type **cd /usr/share/dict**. Issue your egrep commands on the file **words**. If you are asked for “the number of”, use the option –c. If you want line numbers, use the option –n. If you want to ignore case, use the option –i. And if you want to “invert the match”, use the option –v.
   1. *List all words that have the letter* ***a*** *followed immediately by the letter* ***z****.*
   2. *List all words that have the letter* ***a*** *followed sometime later by the letter* ***z*** *(there must be at least one letter in between).*
   3. *List all words that start with the letter* ***a*** *and end with the letter* ***z****.*
   4. *List all five letter words that start with the letter* ***a*** *and end with the letter* ***z****.*
   5. *List all words that start with two capital letters followed immediately by at least one lower case letter.*
   6. *List all words with at least 3 consecutive vowels (upper or lower case).*
   7. *List all words with two consecutive* ***a****’s (upper or lower case)*
   8. *List all words with two consecutive* ***a****’s or two consecutive* ***i****’s or two consecutive* ***u****’s (lower case only).*
   9. *List all words that contain a* ***q*** *where the* ***q*** *is* not *immediately followed by a* ***u****. For instance,* ***queen*** *should* not *be in your list but* ***Iraqi*** *should be.*
   10. *List all entries that contain at least one non-letter.*
   11. *List all entries that comprise only digits.*
   12. *List all entries that contain punctuation marks other than hyphens, colons, periods and commas. HINT: grep on punctuation marks and then pipe this to grep that eliminates those four characters.*
3. For the remainder of the lab, you will use **ls –l** and pipe the results to **egrep**. This will allow you to search not on filenames which you could do using normal Bash wildcards, but on various properties of the files such as permissions or owner. The format will look like this: **ls –l | egrep ‘*regex’*** where ***regex*** is a regular expression. Change directories back to **/etc**.
   1. Let’s imagine that we want to find all files created in the month of January. Enter **ls –l** and you will see that the 6th column contains the month of creation (abbreviated). Enter **ls –l | egrep ‘Jan’** and you will see such a list. Write an instruction that will list the files that were created in **2011**. *What instruction did you come up with?*
   2. Come up with the proper instruction to find all items that are symbolic links. HINT: such files have permissions that start with the letter **l** (that’s a lower case **L**). *What instruction did you come up with?*
   3. List all items that are not *normal* files (i.e., where permissions do *not* start with a hyphen). *What instruction did you come up with?*
   4. List all files whose size is greater than 9999 (that is, the size is at least a 5-digit number). *What instruction did you come up with?*
   5. List all files whose names have three consecutive **r**’s, **s**’s or **t**’s (in any order or combination, e.g., rss, tts, str). *What instruction did you come up with?*
   6. List all files whose names contain two **a**’s separated by either 1 or 2 letters (e.g., **java-6-sun** or **aliases**). *What instruction did you come up with?*
   7. List all files whose permissions are 644. You will have to translate 644 into the proper characters to match against the permissions of files. *What instruction did you come up with?*
   8. List all files which are not accessible by “others” (non-owner, non-group). Any such file will end with permissions ---. *What instruction did you come up with?*
   9. List all files which do not have root as both the owner and group. HINT: the 3rd and 4th items in the ls –l are the owner and group and most of the items in /etc have root root as these two entries. We want to find all entries that do not have root root. *What instruction did you come up with?*
   10. List all files which have root as either owner or group but not both. HINT: pipe ls –l to egrep and pipe that result to another egrep. This is challenging!

You may shut down your Linux VM now.

# Lab 10: Managing Processes

Goals: to learn the tools for process monitoring in both Windows and Linux; to learn to control process movement between foreground and background; to adjust process priority; to schedule Linux processes using at; to learn how to terminate (kill) processes.

Introduction: as a user, you must understand how to interact with your running processes. Tools are available to monitor CPU usage, process status, and to change process priority. As a system administrator, it is even more critical that you know how to examine the status of the system through the various tools available. As a system administrator, you will also have to schedule tasks to run based on either time or event. In this lab, we look at the tools available in both Windows 10 and Linux.

Time to completion: 2 hours

Before starting, review chapter 11 Starting a Process, Process Execution, Process Status, Process Scheduling, and Terminating Processes.

You will use your Windows VM in the first half of this lab and your Linux VM in the second half.

1. We’ll start by using Windows power shell.
   1. From the **Windows** start menu, select the **Windows PowerShell** tile (if you did not pin it from lab 3 then find it in the listing of **All apps**). Type **tasklist**. This displays the list of running processes on your system. Each process is listed by name, PID (process ID), session name, session number, and memory usage. *What process is using the most amount of memory?* *What process is using the least?* Now type **tasklist /v**. This option for tasklist includes the owner of each process. *How many of the running processes are owned by you?* Another interesting option is /svc. Try it. It lists for each process, the services that the process uses. Leave the PowerShell window open.
   2. Start **Task Manager** (right click somewhere in your **taskbar** at the bottom of the screen and select **Start Task Manager**, you can also start the task manager by typing **control+shift+esc**). You will probably be in the Applications “tab” which will display only your running applications (Task Manager and Windows PowerShell). At the bottom of the window is a down arrow and More details. **Click** on the **arrow**. Task Manager changes now to list Apps, Background Processes and Windows Processes (for these, scroll down) as well as giving you a menu and tabs. You should be at the **Processes** tab now. You will see similar information displayed as to what you saw from tasklist in step 1a. *What does the Task Manager show you that the tasklist command did not?* Click on CPU at the top of that column and it will re-sort the items by CPU usage. *What process is using the most amount of CPU time?* Now sort the list on Disk usage. *What process(es) is(are) accessing the disk at more than 0 MB/s?* Click on the Performance tab and you will see a summary of the four items (CPU, memory, Disk, network) usage summed by all running processes, but changing over time. *Of the four, which is(are) not changing?* In the lower right of this tab you can see the number of running processes and threads. *How many are there of each?* Leave the Task Manager running but switch back to the Processes tab.
2. Let’s see what happens when we start a new process. In your PowerShell window type **calc**. This runs the calculator app, which will appear.
   1. In your command prompt window, type **tasklist**. You should find calc.exe to be the second to bottom process. *What PID has it been given? How much memory is it using?* Look at your Task Manager window and you will see Calculator.exe has been added as an App but this view does not show us its PID. Switch to the Details tab. *Do you see the same PID? Do you see the same memory usage?* Switch to the **Applications** tab. *Is Calculator listed there?* Type in the PowerShell window taskkill /f /pid *PID* where PID is the *PID* for calculator.exe as you noted both via tasklist and the Task Manager. The /f option forces the program to terminate. Run tasklist again and look in the Task Manager window. *Does calculator.exe appear in the list?*
   2. Return to the Task Manager and select **Run new task** from the **File** menu. From the pop-up window, type **calc**. This does the same thing as typing **calc** from your PowerShell window. In Task Manager, select the **Processes** tab again and find Calculator, **right click** and view the options: Switch to allows you to bring this process to the foreground, End task, Create dump file, Properties and some other items. Select on **Switch to**. *What happens?* There are multiple ways to “kill” a process. We’ve already seen taskkill and now from the Task Manager, we can right click on the process name and select End Task. However in Windows, it is a frequent occurrence that a process will stop responding in which case End Task may not do anything. Go to the **Details** tab and right click on **Calculator.exe**. Among the options here that you didn’t see from right clicking on Calculator in the Applications tab are End Process and End Process Tree. These should kill a process even when you cannot kill it from the Applications tab. End Process Tree causes any processes launched by this process to also exit. This is usually the safest approach as you do not want child processes to remain running. Try **End Process Tree**. A pop-up window asks if you are sure, click **End process tree**. *What happens?*
   3. You may close your PowerShell window but leave Task Manager running. Again from select **Run new task** from the **File** menu and from the pop-up window, type **notepad** and click on **OK**. This will start a Notepad session for you. Return to the **Details** tab, right click on **notepad.exe** and select **Set Priority**, changing it to **Low**, and from the pop-up window, click on **Change priority**. Type something in Notepad. *Do you notice a change in its performance?* Probably not but it may now be dragging slightly. For more computationally intensive processes, lowering the priority should become noticeable. Reset the priority this time to be **Realtime**. In doing this, you might find other processes lag. Right click on notepad.exe and selection **Change Affinity…** A pop-up window appears. You can just click on **Cancel**. *But from what appears in the pop-up window, what do you surmise affinity is?* Right click on **notepad.exe** and select **Properties**. From the pop-up window, look at the **Security** tab. *For the various users and groups (aside from TrustedInstaller), what access rights do they have?* Now look at the **Details** tab. *What version are you running?* You may close the Properties window and then Notepad (do not save the file).
   4. Click on the **Performance** tab in the Task Manager. *What is the amount of current CPU usage and memory usage? What % usage do you see for disk (this may change, so record the largest value you see while watching).*
   5. From the Performance tab, click on **Open Resource Monitor** which is at the bottom of the Task Manager window near the left. A new window appears. This provides even more information than the Performance tab. Click on **Overview** if it not already selected. You can select any process(es) you want in the upper pane. Select **System**. If you expand any of the other areas, Disk, Network, Memory, you can see System’s utilization. Click on the **Memory** tab. *How many hard faults per second have occurred? How much of your memory is Committed, Working, Shareable and Private?* Near the bottom of the window you will see a summary of Physical Memory statistics. *How much memory is reserved for hardware? How much is in use?* Click on the **Disk** tab. *Approximately how much is being written per second (in Bytes)?* You may close this window.
   6. *Summarize why you might use the Task Manager and the Resource Monitor.*

You may now shutdown your Windows VM. If not already started, start your Linux VM.

1. Open a terminal window. First, we will look at moving processes from foreground to background and back. Most of the processes that we have run execute so quickly enough that we do not need to specify foreground or background.
   1. Type **man vi**. This brings up a man page. Type **control+z**. This stops the process and returns you to your prompt. Type **jobs**. This lists all of the tasks running in this window that either are running in the background or have been stopped. Type **vi**. This launches vi which fills the entire terminal window and keeps you from having a command line prompt. Again, type **control+z**. Type **jobs**. You should see both man vi and vim listed. The numbers [1] and [2] indicate the order that the jobs were started. The + and – indicate the most recently stopped job (+) and the second most recent job stopped (-). To resume a job, type fg to resume the + job and fg # where # is the number to resume in any other job. Type **fg 1** to resume man. *Did it work?* Again, type **control+z**. Type **jobs**. *Did the + and – switch? Why?* Type **man ls**. Type **control+z**. Type **jobs**. *What is listed now?*
   2. Type **gedit**. This starts the gedit GUI editor. Notice in your terminal window that there is no prompt available. This is because gedit, even though open in a separate window, is in the foreground of your terminal window. In order to have a prompt, you must stop gedit either by exiting out of it or stopping it with control+z. **Click** in the **terminal window** and type **control+z**. **Click** in the **gedit window** and you will find that it doesn’t respond correct. **Click** in your **terminal window** again and type **jobs**. Typing fg will resume gedit but will lose you your prompt. Instead, we will move gedit to the background. Type **bg**. *What was returned in your terminal window? Do you have a prompt in your terminal window? Can you access the gedit window?* **Exit** out of **gedit**. You can launch any process directly into the background by adding & after the command. Type **gedit &**. Type **jobs**. What *is the status for gedit unlike the other stopped jobs?* From the command line, type **fg**. *Which job resumes?* Notice that the last active job was gedit but it is not the one that resumed. Type **q** to exit the man page. Type **fg** and **:q** to exit vi and then type **fg** and **q** to exit man. Also **close gedit**. Type **jobs**, you should have nothing listed.
2. We will examine the two common command line process management programs: ps and top.
   1. Type **ps**. Type **ps a**. Type **ps x**. The first of these provides very little information. What do the options a and x show you (if you can’t determine this by looking at the output, read the man page). Type **ps fx**. Scroll up and down to see the full list. This version combines x with f, which connects related processes using “Ascii art” to denote parent/child relationships. Find the entry in this listing for ps fx. *What are its parent and grantparent processes?*
   2. Type **ps aux**. The u means “user-oriented format” which displays even more information. *In looking at the output, what does TTY mean and what does an entry of ? in TTY mean?* What do the states S, Ss, Ssl and S+ mean? You can restrict ps to only provide processes owned by a particular user by adding –U username as in –U zappaf in the command (alternatively, you could also pipe the results of ps to grep). Type **ps –U *username*** where *username* is your username. *What is the range of PIDs for your processes (the smallest and largest PIDs)?*
   3. Type **ps aux | less**. You will find all of the initial processes to be owned by root. PIDs are assigned in numeric order from the beginning of system initialization. So this should make sense because users are not able to start using the system until after the system has been initialized. Therefore, all the early processes are started by root. *What is the first process?* Scroll through the processes. *What is the PID for the process auditd? What about crond?*
   4. Type **top**. Unlike ps which provides the process status at the time you enter the command, top is interactive and remains running. It updates itself roughly every 3 seconds. A summary of your system is provided at the top of the display. *How long has your system been “up”? How many users are using the system? What does load average say? How many processes (tasks) are running, sleeping, stopped and zombie? What processes are using the most CPU time?*
   5. While top is running, type **h**. This brings up top’s help screen. You can see that you can alter what is being displayed or make other changes. One command is d to change the update time. Another is u so that you can specify a user. Hit **any key to return to top**. Type **d** and from the prompt that appears, type **.75**. This modifies the update time from 3 seconds to .75 seconds. *Do you notice a difference?* Type **u** and when prompted, enter **your user name**. *What change happens this time?*
   6. *Which of the two programs top or ps would you prefer to use to identify how your system is running?* Type **q** to exit top. Leave your terminal window open.
3. We’ll now begin using the GUI tools.
   1. From the desktop, click **Applications->System Tools->System Monitor**. This is a tool that is similar to Windows Task Manager. Click on the **Processes** tab. From here, you can see the running processes much like in top or ps. *What is the status of most of the processes listed?* If you select any process in the list and right click, you get similar options as to windows: stop process, kill or end process, change priority, open file, etc. Select **any process** and click on **Change Priority…** Here, the priority is called niceness. The nicer the process, the more it is willing to give up its CPU time to other processes, thus a nicer process has a lower priority. Alter the priority to have a lower niceness (**move the slider to the left**). You will see the nice value become negative. Once you have lowered the niceness (raised the priority), click on **Change Priority**. *What happens?* A normal user is not allowed to raise priority. Click on Cancel. Try again but **raise the niceness** (lower the priority). *What happens when you click on Change Priority this time?* If you look in the process listing, you should see a difference Nice value. Select **another process** and click on **Open Files**. This displays the files (and file-like objects) that this process currently has open. *How many objects are open? How many are files, how many are pipes and how many are local sockets?* If you selected a process which has nothing open, select a different one instead. Close the Open Files window.
   2. Return to the System Monitor and examine the **Resource** and **File Systems** tabs. The Resource tab is similar the Performance Tab in Windows Task Manager although it looks less impressive. The File Systems tab shows you information similar to what you can find using the df command. *Which Devices/Directories are listed in the File Systems tab?* These are your partitions (aside from the swap space). *Approximately how much have each of these partitions used?*
   3. Select **Edit🡪Preferences**. The pop-up window has 3 tabs, Processes, Resources and File Systems. You can change the information that appears in each of these tabs in the Resource Monitor. Click on the **Processes** tab and click on the checkboxes for **User** and **Started**. Click on **Close**. In your System Monitor, go back to the **Processes** tab and you will find these columns added. *What is the earliest date (or time if started today) of a process started by you? What is the earliest date/time of any process started?* You may close your System Monitor.
4. We wrap up this lab by looking at scheduling and killing processes in Linux. You will use your terminal window for these steps. su to root.
   1. Type **man top** and then **control+z** out of it. Type **ps** and make a note of the PID for the man top instruction. Type **kill -9 *PID*** where *PID* is the PID of the man command. This kills the man command. Type **jobs** to make sure the process is no longer there. *What response did you get from jobs?* The -9 is a signal that kill sends to the operating system indicating how to kill the process. -9 is the most powerful and should be used for processes that have stopped responded or that you are done with which are not using open files. NOTE: you can kill processes through the System Monitor program as well, just like with Windows Task Manager.
   2. To schedule a command, you have three choices of programs, at, batch and crontab. We will only examine at. As root, type **at now + 1 minute**. Because you did not specify a file, you are dropped into an at> prompt. From here you can enter the task(s) that you want carried out in one minute. Type **du –sh /home/*username* >> /home/*username*/diskusage** (where *username* is your username). After pressing <enter>, you will be at the at> prompt again. Type **control+d**. *What does at respond with?* In 1 minute, this task will run. You can test to see if the task is scheduled by typing **atq**. Do so. *What response do you get?* Check in 1 minute to make sure that it worked (you should have a new file in your directory).
   3. Using **vi**, create a file called **myatfile** that contains the following two lines:

**echo `date` >> /home/*username*/diskusage**

**du –sh /home/*username* >> /home/*username*/diskusage**

From the command line, type **at –f myatfile *TIME*** where *TIME* is 1 or 2 minutes from the current time, for instance 09:35am, 11:50am, 03:40pm. Alternatively, you can use military time and omit the am or pm as in 09:35 or 15:40 (3:40 pm). Notice this time you are not dropped into that at> prompt because you specified the file. Type **atq**. *What response did you get?* Type **at –f myatfile teatime tomorrow**. Type **atq**. *What time and date is this task scheduled for?* To delete a scheduled job, type atrm # where # is the job number as specified when you do an atq. Delete the job for tomorrow. *What command did you enter?* Use atq to make sure that there are no more scheduled jobs.

You may log out of your Linux VM.

# Lab 11: Managing Services

Goals: to learn about services in both Windows and Linux; to learn how to control, start and stop services; to learn how to configure a couple of Linux services.

Introduction: services are operating system programs that run in the background, waiting for a request that the service can then handle. Services range from dealing with network communication to file system handling to logging messages to dealing with I/O devices. There are dozens of services in both Windows and Linux. As a system administrator, you will have to understand these services so that you can start and stop them as needed or identify that the system is not functioning correctly because a service is not responding. You can alter a service’s behavior by changing its configuration. This is more common in Linux so we will examine this in Linux but not Windows.

Time to completion: 2 hours

Before starting, review chapter 11 Services, Configuring Services and Establishing Services at Boot Time.

You will use your Windows VM in the first half of this lab and your Linux VM in the second half.

1. Bring up the Computer Management program: from the **Windows** menu, select File Explorer, right click on This PC and select **Manage**). Next, expand **Services and Applications** and click on **Services**. This will open the available Windows services in the middle pane. Also make sure that the **Extended** tab is selected at the bottom of the middle pane. If your Computer Management window is not large enough, you will see a scroll bar at the bottom of the list of services, keep this in mind as you might need to scroll to the right to see more detail about a particular service.
   1. You will see for each service, its name, description, status, startup type and log on as value. Review chapter 11 section 6 if you are unsure what startup type and log on as means. Click on any service and you will see its description appear to the left of the list. Scroll down until you find **DHCP Client**. Click on it. You can see in the left that you have two options, Stop or Restart (this assumes the service is running). Read the description. *What type of service is this, logging, network, security, file system, peripheral device or other?*
   2. In examining the services, you will see many are started but some are not. Select one. *What option(s) is(are) available (above the description)?*
   3. *In examining the list of services, are any disabled? If so, which ones? Are any services started automatically yet are not currently running? If so, which ones?*
   4. Let’s examine a service up close. Right click on **DNS Client** and select **Properties**. There are four tabs. Under **General**, you should see the name and description, the location of the service’s executable, the Startup type and its current status. DNS Client should be set to Automatic. *What are the other possible choices? What is the difference between the first two choices?* Click on the **Log** **On** tab. Some services do not require being logged in. This one does. *Which account is used by this service to log on?* Notice that a password has already been entered. Do not change this! Click on the **Recovery** tab. This tab specifies how the service should be restarted if the service happens to fail. *For this service, what will happen the first two times it fails? What about the third time?* The final tab, **Dependencies**, lists any services that this service depends on and any services that depend on this service. That is, for this service to run, these other services must be running already. *What services does DNS Client depend on?* What happens if you stop a service? Click on the **Stop** link in the list of options in the upper left hand corner of the middle pane. *What is the status of this service now?* What impact will stopping this service have? *Does it prevent you from accessing the Internet?* Try some URL in your web browser and you should find that you can access the Internet without a problem. But stopping the service makes your computer less efficient because your computer will no longer cache IP addresses that this service caches. **Start** the service. Typically there is little or no need to stop a service in Windows unless you modify the service’s configuration file. We will examine how to do that for Linux in step 3 below, but not in Windows. Close the DNS Client Properties window.
   5. Search through the services. Find one that does not depend on any other service (that is, the top Dependencies box is empty). *What did you find?* Find one that takes no action on a failure. *What did you find?* You can close the Computer Management window now.
   6. Start **Task Manager**. Click on the **Services** tab. You will see that the same list of services from the Computer Management listing is shown here. Here, you also see for those services that are running its PID and its Group. Click on the **PID** column header which will sort all of the entries by PID. You will see for all Stopped services no PID. *What is the smallest PID? What service has this PID? What is the largest PID? What service has this PID?* You might notice that some services share the same PID. Right click on any service and you will see that you can start a stopped service or stop a running service. Find any running service and right click on **that service** and select **Go to Process**. *What happens?* Return to the **Services** tab. Click on the **Services…** button and a pop-up window will appear that contains the same information as the center pane in the Computer Manager. Close the **Services** window, the **Computer Management** window, and the **Task Manager** window and then you can log off of your Windows VM.
2. Start your Linux VM. Open a terminal window and su to root.
   1. From the **System menu** select **Administration 🡪 Services**. The Service Configuration window appears. This window is poorly named because it lets you start and stop services but not configure them! We will do that later. You will find services listed as either running (the electrical plug symbol is plugged in) or stopped (the plug is not plugged in) and enabled or disabled. Find a service that is enabled but stopped. *What service did you find?* Find a service that is disabled but running. *What service did you find?* Running versus stopped is the same as in Windows. Enabled versus disabled determines whether the service is automatically started at system initialization time or not (we will review this in more detail in 2c and 2d). Much like with Windows, you can easily start and stop services through the GUI. Click on a service and then you can click on Stop or Restart (if it is running) or Start (if it is stopped). Select the **crond** service and **stop** it. *What happens?* **Log in as root** to complete the operation. **Start** the **crond** service. While crond is still selected, click on **Customize**. The Customize Runlevels window appears for crond. *For what levels is it enabled?* **Close this window** and **close the configuration window**.
   2. From the command line, type **/sbin/service crond status**. *What is output?* You can control any service from the command line by issuing the command **/sbin/service *servicename* *command*** where *servicename* is the service’s name like crond and *command* is one of start, stop, restart or status. Change directory to **/etc/init.d**. Do an **ls**. *What do you see here?* These are all scripts used to control the services. You can also control the service right from here. **Type ./crond status**. *Did you get the same response as with the /sbin/service crond status instruction?* Type **less crond**. Here, you see the contents of this script. The first screen is mostly or all comments. Starting with the second screen you see the actual script code. Near the bottom of the second screen you should see a function called start. When you execute ./crond start, the operations listed here execute. *What are the other functions defined in this script?*
   3. Change directory to **/etc** and type **less inittab**. This script has one responsibility, to establish the run level. This will probably be at or near the end of the script using notation like this: id:#:initdefault: where # is the runlevel. *What runlevel is specified?* This is your computer’s default runlevel. To change the runlevel each time you boot your computer, you would modify this file. Before this instruction you will see in comments what each runlevel does. *Which runlevel would you use to boot to single user mode? (single user mode means root) What runlevel would you use to boot to the exact same setup that you see now but with text instead of GUI?*
   4. The init process (you might recall it from the last lab) calls numerous initialization scripts including inittab. Among those executed are /etc/init.d/rc.sysinit, /etc/init.d/rc and /etc/init.d/rc.local. Let’s look at the rc script. **cd** back to **/etc/init.d**. Type **less rc**. In scrolling down through the code you should see

**for i in /etc/rc$runlevel.d/K\* ; do**

…

**$i stop**

…

**for i in /etc/rc$runlevel.d/S\* ; do**

…

**$ start**

Let’s examine these two for loops in more detail. $runlevel is a variable that was defined to contain the number of your runlevel (e.g., 5). Thus, the for loop iterates over /etc/rc5.d/K\* and /etc/rc5.d/S\*. Let’s look at the /etc/rc5.d directory. Type **ls –l /etc/rc.5**. You will see that this is actually a symbolic link to rc.d/rc5.d. So type **ls –l /etc/rc.d/rc5.d**. *What kind of thing makes up all of the contents of this directory? Where are they all pointing?* You might recall from step 2b that /etc/init.d stores all of the scripts for controlling the services. So what are we seeing here? An entry like K80kdump is used in the first for loop since it starts with a K, so kdump it is stopped. A service like S12rsyslog is used in the second for loop since it starts with an S, so rsyslog is started. The numbers are used just to dictate the order the services are stopped or started. *In your own words then, summarize what is found in each of the /etc/rc.d/rc#.d directories (where # is a number from 0 to 6), what the runlevel controls, and what the rc script does.* The last script executed during system initialization is rc.local. This is a script that is currently nearly empty and provided so that the system administrator can add his or her own tasks that should execute during system initialization. These might include, for instance, mounting a remote file system (e.g., a networked file server), run an antiviral program, run some statistical program to examine log files, check the file system for bad permissions, etc.

* 1. As a system administrator, you can easily change the services that start or stop or the order that they start or stop for each runlevel by modifying the names of the entries found in the specific rc.d directory. For instance, if in /etc/rc.d/rc5.d you change S90crond to be S75crond, you will have changed the order so that crond starts before postfix. *What would you have changed if you alter K88sssd to be S88sssd?* You can also modify which services start or stop automatically by runlevel using the Service Configuration tool and the Customize button. Another option though is the command chkconfig. Type chkconfig. You will see for every service and runlevel whether that service should be started or stopped. Aside from runlevels 0 (halt) and 6 (reboot), what services are automatically started for the other levels (1-5)? How many services are always off (off for all 7 runlevels)? You can alter the runlevels that any service will be started on by using **chkconfig --level *levels* *servicename* on** where levels are the runlevels that you want to have this service started for and servicename is the name of the service. The levels are specified all together with no spaces or commas, for instance, chkconfig --level 35 crond on. You can use off in its place to have the service stopped for those runlevels.

1. We will wrap up this lab by examining how to configure two Linux services. Change directory to /etc where you will find many of the Linux service configuration files. We will examine a few and then modify two.
   1. One service, ntpd, controls the network time protocol (NTP). Its configuration file is /etc/ntp.conf. Type **less ntp.conf**. You will see many lines that start with #, these are *comments*. Other lines contain *directives*. These are the statements that are enacted when you start the ntp service. These will include the location of files (e.g., driftfile), values for variables such as server, and other statements such as restrictions for the service. Examine the **yp.conf** file which is the configuration file for the ypbind service. *What directives are defined in this file?*
   2. One service, rsyslog (what used to be called syslogd), specifies for running operating system applications where messages should be logged. Type **vi rsyslog.conf**. This is the rsyslogd configuration file and it specifies for different types of operating system software and different message priorities, where to log those messages. Press the page down key to reach the second page of this file. Near the bottom of the second screen you will find the rules for rsyslog. The first, kern.\*, is commented out. This rule, if uncommented, would send all kernel messages to /dev/console which is the main console window. The next rule is uncommented. It sends all info messages to /var/log/messages. It also sends some mail, authpriv and cron messages there. The next rule deals with authpriv.\*. *What category of system software does this represent? Where are these messages sent?* Let’s add a rule. Beneath authpriv, type **o** to enter insert mode. Type the rule

**daemon.\* /var/log/daemons**

Hit **<esc>** and then save the file and exit (**:wq**). Let’s see if this file has been created. Type **less /var/log/daemons**. *What happens?* Why? With any change made to a service’s configuration file, we must restart that service before that change takes effect. Type **/sbin/service rsyslog restart**. *What messages did you receive?*  Type the **less** command again. The file should now exist but is probably empty. We need a service to do something that will generate a message. We will use at. Type **at now**. This drops you into the at prompt. We will do something that will cause an error. At the at> prompt, type **atd <enter> control+d**. From at, we are trying to launch the atd service which is already running! Type **cat /var/log/daemons**. *What do you see listed? Where did this message originate from (what service)?* You could similarly have other types of software generate different types of messages. The rule we added, daemon.\*, could potentially generate far too many messages. Re-executive **vi rsyslog.conf** and **comment** **out** the line you added. To comment it out, insert # at the beginning of that line. **Restart** the **rsyslog** service. *How did you do that?*

Type **cd sysconfig**. One file here is called iptables. This is the Linux firewall. **Load** it into **vi**. This file contains rules for your firewall. Each of these rules is a –A rule (add). This means that each rule is attempted, in order, until we reach an action (-j). Nearly all of these rules have an action of ACCEPT meaning that the message is accepted. Only if we reach the last two rules will the message be rejected. Let’s see what happens if we make some change(s) to this file. Move down to the first REJECT line. Type **dd** (cut) and then move up to the first –A rule and type **P** (paste above). **Save** and **exit** the file and **restart** the iptables service. *How did you do this?* **Start your web browser** and type in **any URL**. *What happens?* Why? Because any incoming message is now rejected no matter what! Shrink your browser and **load iptables** back into **vi**. **Move** that REJECT line back to where it was (dd, move to the last –A INPUT and type p). Now enter the following rule immediately before the first –A INPUT rule:

**-A INPUT –s 173.194/16 –j REJECT**

This rule says to reject any message that comes from the IP address starting with 173.194 (the /16 means “the first 16 bits of the IP address” and the –s means “source IP address). The IP address 173.194 are used by all of Google’s servers. **Save** and **exit** the file and **restart** iptables. Now in your browser, enter any URL other than for Google*. Did it work?* Now try **www.google.com**. *What happens? Explain how you could use this approach to ensure that no user would be able to access Facebook from this computer.* **Return** to editing **iptables**, **delete** the rule you added, **save** the file and **exit**. **Restart** iptables.

* 1. Another service is called kdump. This service provides a core dump when the kernel crashes. The kdump.conf file (located under /etc) contains directives to alter kdump’s behavior. *Briefly explain how you would modify kdump so that the core dump is produced under /var/kdump instead of /var/crash.*

You may log off of your Linux VM now.

# Lab 12: Networking

Goals: to learn about network services, commands, files and programs in Linux and Windows; to learn concepts related to TCP/IP.

Introduction: in order to communicate over the network, your computer must utilize a number of different tools. These include the physical hardware (network card or wireless modem), network devices (e.g., routers), network software and network services. Supporting those services will be a number of files that store various network information. We will explore networking in Linux in some detail and then see how some of these details are implemented in Windows.

Time to completion: 90 minutes - 2 hours

Before starting, review chapter 12 Network Protocols and Network Software.

1. Start your Linux VM. Open a terminal window. su to root in the terminal window.
2. Type **/sbin/ifconfig**. You will see two different sets of information provided in both windows, the eth# information (the # will be a number, probably 0, we will assume this is eth0) and the lo information. The eth0 entry (Ethernet 0) is the machine’s external connection to the world which includes the IPv4 address (inet addr) and an IPv6 address (inet6 addr) if you have one? *If so, what is it?* The lo entry is known as the “loopback” device. Software often uses this to communicate to other software or parts of the computer without actually going out onto the network. *What is your machine’s lo IP address and what is your machine’s eth0 IP address?* A MAC address is an address assigned to your network card and used by external switches. You can find this address listed under HWaddr for eth0. *What MAC address does your computer have?*
3. In order for your computer to communicate to other computers over the network, you need to specify IP addresses. It is more convenient to use IP aliases though (e.g., www.google.com instead of 173.193.77.99). To perform this address translation, your computer consults a DNS server. The address(es) of your local DNS server(s) is stored in /etc/resolv.conf, listed as nameservers. Type **less /etc/resolv.conf**. *What are the IP addresses listed here?*
4. One tool to map aliases to addresses is called nslookup. It works by specifying the destination computer’s IP alias. You can request this query of your own DNS server or of others. Type **nslookup www.nku.edu**. Aside from this machine’s IP address, you may also see a canonical name. This is the machine’s true name. *What IP address and true name did you get?* Notice at the top of the output you will see Server and Address which are the DNS server’s IP address that you used. Now use nslookup on www.google.com. You will receive multiple names and addresses. *Why do you suppose this was the case?*
5. A useful tool is called ping. It sends out packets to a destination computer and displays information about the responses. This allows a system administrator to query a remote computer to see if it is available. Obtain your IPv4 address from #1 above and type **ping *address*** where *address* is your IP address. After a few responses occur, type **control+c** to exit out of ping. *How many packets were transmitted and received? Were any lost?* (hopefully not!) *What was the minimum and average time per packet transmission/reception?* Now repeat this with 127.0.0.1. You might recall from #1 above that this is the address of your loopback device. The minimum and average times should be a little less because this did not actually involve the network. Now type **ping www.google.com**. Again, stop after a few packets. *Compare the minimum and average times to your own IP address.*
6. Another tool is called traceroute, which provides the IP addresses (or aliases) of devices that were used in the pathway between your computer and the destination. This allows you to see how a message is steered over the Internet. Type **traceroute www.google.com**. Note that it could take a little (several seconds) while to complete. Entries listed as \* \* \* indicate deadend pathways. *How many different devices were listed?* The first one should be your computer’s gateway and the last one should be the destination computer. Now try **traceroute *address*** where *address* is your IP address from #1 above. *How many hops were listed?*
7. Another command available in Linux is called route (not to be confused with traceroute). The route command provides you your router table information. Type **route**. The first number should be your network’s network address. The same line will also contain a genmask (netmask). Obtain your IP address from #1 and AND it with your Genmask. The result should be your network address. You will have to convert these addresses from decimal to binary.

*Write down your IP address in binary: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Write down your genmaks in binary: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*AND these values: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*This is your network address in binary*

*Convert this address to decimal: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

The last entry in your routing table should be your router. Its address will be listed under Gateway. NOTE: both ifconfig and route have been replaced in Linux with ip. As ip is a far more complicated program, we will ignore it. You might wish to explore it on your own.

1. In order to obtain access to the network, your network service must be running. It runs by default in any of the network-based runlevels (3 and 5). Let’s see what the consequence is of shutting down this service. Type **/sbin/service network stop**. Type **/sbin/ifconfig**. *Do you have an inet address for eth0? Do you have an entry for lo?* Shutting down the network causes your computer to “give up” your IP address. Neither eth0 nor lo is accessible. Try to ping both your IP address (refer back to your answer to #1) and 127.0.0.1. Type **route**. *What information is provided?* Type **/sbin/service network start**. Type **/sbin/ifconfig** and **route** again. NOTE: depending on your computer’s setup, restarting the network service may have changed your IP address!
2. Type **cd /etc/sysconfig/network-scripts**. You should have a file called ifcfg-eth0 and another called ifcfg-lo. These files configure your network access. Look at ifcfg-eth0. *What is the value for BOOTPROTO?* It should be DHCP. *What is DHCP?* (you might need to research this) The other option is static. *What is a static IP address? Are there advantages to having a static IP address over a dynamic one?* Compare this file to ifcfg-lo. You might notice that the IP address is “hardcoded” into this file. *Why are we able to do that with lo but not eth0?* Notice in this directory are additional script files. Of note are ifdown-eth and ifup-eth which allow you to bring down or up just eth0. Also, in /etc/sysconfig are files for your firewall: iptables, ip6tables, iptables-config, ip6tables-config. We briefly examined iptables in lab 11.
3. You might recall from chapter 12 that level 2 of the OSI model deals with Ethernet (MAC) addresses while higher layers deal with IP addresses. ARP is the Address Resolution Protocol which is used to map from the higher level IP addresses to the lower level addresses. This is done through an ARP table. You can query the ARP table with the arp command. Type **/sbin/arp –a**. You will find a hexadecimal address. This value is inserted when you first access the low levels of the network. Shut down your network service and redo the arp command. *What is the response this time?* Restart your network service. Repeat the arp command and you will find the address to reappear. The Ethernet address is cached in your ARP table for only a few minutes at a time. It is replenished whenever you do another network command. To view this, type **/sbin/arp –d *IPaddr*** where *IPaddr* is the IP address that appeared in the /sbin/arp –a command initially. The –d option flushes the ARP address. Type **/sbin/arp –a** and you will see no address listed. Now type **ping 127.0.0.1**. After a couple of packets are returned, **control+c** out of it. Repeat the arp command. You will find the value restored. You may shut down your Linux VM.
4. You should now start your Windows VM.
5. We will explore ping in Windows. Open up a **PowerShell window**. Type **ipconfig**. This program is similar to the Linux ifconfig program. It responds with a description of your network connection including your Ethernet connection to the network, your IPv4 address, possibly an IPv6 address, your subnet mask and default gateway. *What is your IPv4 address? What is your subnet mask? What is your default gateway?* Compare these three values to what you found from Linux in part 1. *What is the value of your connection-specific DNS suffix?* This is your network’s alias so that, when you attach your computer’s name at the front, you have your full IP alias.
6. Type **ping 127.0.0.1**. **Control+c** out of this after a few packets are returned. Now type **ping *ipaddress*** where *ipaddress* is the address of your machine as you discovered in 2a. Again, **control+c** out of this after a few packets. As you can see, you can ping yourself. Now **try ping www.google.com**. **Control+c** out of this after a few packets. *Compare the time it took to reach yourself and google to what you saw in 1.d. Is there a significant difference in time?* There is no equivalent to traceroute in Windows.
7. Type **nslookup**. Usually nslookup expects an IP alias as an argument (such as nslookup www.google.com) but we are using it simply to obtain the IP address of your DNS server. The command will place you into the nslookup prompt (>); **control+c** out of it. *What IP address and default server name did you receive?* **Close** your **command prompt window**.
8. At the bottom of your desktop is a search box. Type **Control Panel** and then select Control Panel from list of items that appear (it should be the only one). Note that in Windows 10, you typically change settings through the **Settings** tool, but the Control Panel gives you more options. From the Control Panel, select **Network and Internet** and then from the new listing, **Network and Sharing Center** and from there, on the lower right, select **Windows Firewall**. Select **Advanced Settings**. A firewall window appears. In the upper left hand pane, expand **Monitoring** and then select **Firewall**. In the middle pane you will find the Firewall rules that your firewall is using. Allow actions are those that define the messages permitted through your firewall and Block actions define those that are blocked. *How many block rules are there?* Inbound are rules that handle incoming messages while Outbound are rules that handle outgoing messages. *Do you have any outgoing messages defined?* You might notice above Monitoring in the left hand pane are Inbound Rules and Outbound Rules. Click on **Outbound Rules** and you will see that in fact you do have rules for outgoing messages but none of them are listed under Monitoring. In the right hand pane, click on **New Rule…** Click on **Port** and then **Next**. You will see that you can define a rule to apply either to UDP or TCP packets of a given port. Select **TCP** and enter port **80**, click on **Next**. You can allow the connection, allow secure connections, or block the connection. Click on **Block the Connection** and **Next**. Have the rule apply to all locations (**domain, public, private**), click on **Next**. Finally, you are asked to name the rule. Click on **Cancel** as we do not want to apply this rule. But this is how you would modify your firewall to add rules. *Why might you want to open port 80 for TCP packets to come into your computer? Describe how you add a rule that blocks all TCP packets from any port that are coming in from a public network location?*
9. Return to the control panel. Click on the top left link, **Allow an app or feature through Windows Firewall**. With this tool, you can more easily define firewall rules. Click on **Allow another app...** A pop-up window appears with a box for all Apps. There may be apps listed or the box may be empty. You can locate other programs by clicking on Browse… Once you find a program, selecting it item adds it to the list of programs allowed passed the firewall. You can also control whether the software’s messages are permitted in a public or private network. Click **Cancel** to close this window and then the **back arrow** in the control panel to return to the previous window. Click on **Turn Windows Firewall on or off**. From here, you can enable or disable your firewall on public and private networks. You will be warned if you attempt to turn it off and you should especially never have it off when using a public network! Click on the **back arrow** again.Click on **Troubleshoot my network**. If you have difficulty reaching the network, you might consult this window to test the Network Adapter and your Internet Connections. There are no questions to answer in this step. You may **close** the **Control Panel** and log off of your Windows VM.

# Lab 13: Software Management

Goals: to learn how to install new software in Windows using an installation program and in Linux using several different methods; to learn how to install open source software.

Introduction: Windows and Linux have simplified the process of software maintenance through the use of installation “wizards” and package manager programs. We briefly look at installation through these approaches. In Linux, we look at both rpm and yum. We also look at the tools in Windows for controlling software maintenance. Finally, in Linux, we will download a piece of open source software and install it from source code.

Time to completion: 1 hour

Before starting, review chapter 13 Software Management.

1. Start your Windows VM.
   1. Type **Control Panel** in the search box and select **Control Panel**. Click on **Programs**. From here, you can control the current programs (upgrade, uninstall, change default programs, add desktop gadgets, view updates). Click on **View installed updates**. Updates are handled automatically by Windows 10. Scroll down in the list of programs until you find Microsoft Windows (your operating system). *What is the date of the most recent upgrade?* **Right click** on the first entry. Here, you can uninstall the installed update if desired (this rolls back your operating system to how it was before the update). Usually updates are provided to fix known bugs and security holes. There is little reason to uninstall an update although occasionally an update may interfere with other installed programs (particularly non-Microsoft programs). Click on the **Back arrow** in the Control Panel browser.
   2. Under the title Default Progams, click on **Change default settings for media or devices** underneathDefault Programs. For each type of media file (e.g., Audio CD, DVD movie, Video files), you can select the software to run it. If the checkbox **Choose what to do with each type of media** is not selected, select it. Click on the **drop down box for Video files**. *What are your choices?* Do not select anything. *For all of the different types listed, how many have a selection other than “Choose a default”?*  *What does that mean to “Ask me every time”?* Click the **Back arrow**. Leave the Control Panel open for later in the lab.
   3. Here, we will install some software (Gimp, which is an open source image editor program). From the Windows start button, select All Apps and scroll down until you find Microsoft Edge. Note that you might also have Internet Explorer (listed under Windows Accessories), Google Chrome or Mozilla Firefox available. You may select whichever web browser you wish. **Start your web browser** and enter the URL **www.gimp.org/downloads**. The web site should pick your OS as Microsoft Windows, if not select Microsoft Windows. Select **Download GIMP .xxx** **directly** (where .xxx is the most recent version). If asked to run a security scan, select **Run**. You will be asked if you want to run GIMP setup, click **Yes**. Running the setup program will bring up a pop-up window asking you what language to use, English is the default. Once you **select the language**, click on **Ok**. You will be taken to the Gimp setup window. Usually, an installation program takes you through several windows including a selection of regular or custom installation. Here, we are asked whether to perform a custom installation from the first window. **Click on Customize**. You are presented the software license agreement. Clicking on Next is an agreement to follow the license. Briefly review this license, it is used in just about all open source software. *What organization is responsible for the GNU GPL?* Scroll down to Terms and conditions. *Which version of the GNUs GPL does this license refer to?* Under Basic Permissions, read the second paragraph. *What are you permitted to do with this software?* Ok, now back to installation. Click on **Next**. You are now asked where to locate the program files. **Leave the default** and click on **Next**. What are you supposed to do at this step of the installation? Here, you can select components. *What are the choices for types of installation?* Leave the defaults as is and click on **Next**. Here, you can select which types of files Gimp will open automatically. Click on **Next**. Now you are asked whether to create a desktop icon and a quick launch icon. Select **desktop icon** and **Next**. Finally, you can review your installation settings. Click on **Install**. This takes a couple of minutes while the files are unpacked and placed into the directory specified earlier. Click on **Finish**. You will find a Gimp shortcut icon now on your desktop. **Double click** **this shortcut icon**. It takes a couple of minutes for Gimp to start. You will be presented with the Toolbox and a blank palette. We won’t use Gimp here. From **File** select **Quit**. Leave your web browser running.
   4. Now return to the Control Panel. Click on **Uninstall a program**. You will see a list of all installed software. **Right click** on the **the entry for GIMP**. *What choices are listed?* Other choices might be change either repair or remove the software, and repair to reinstall portions of the software that have been damaged or deleted. Select **uninstall** for GIMP. Answer **yes** to the pop-up window. Once deleted, a pop-up window appears, click **OK**. If you look at the list of programs now in the control panel window, you will see that Gimp is no longer there. You will also see that the desktop icon is gone. You may **close** the **Control Panel**.
   5. If you do not have Mozilla Firefox running in your operating system, enter the URL **www.mozilla.com** in your browser and select the **Download** button from the webpage. When asked if you want to run the setup for Mozilla, answer **Yes**. In the Set Up Wizard, you are asked if you want to import from other browsers, you may select **Don’t import anything** as you will probably not have anything to import. Select **Next** followed by **Finish**. In Firefox, by default there are no menu items listed. Move up to the top of the browser where the web page tabs are shown, right click and select **Menu Bar**. From **Tools**, select **Options**. An Options tab appears. Select **Advanced** on the left hand side and then the **Update** tab. Some software will automatically update for you such as the Windows OS. Other software requires that you update it yourself. Mozilla falls in between where it will update itself but you can state how you want it updated. In this page, you are able to change from automatic installation to check for updates where you can control which updates are installed or you can select that it should never check for updates. *Why might you change this from automatic?* Click on **Show Update History**. *When was the last update made?* **Click** on **Cancel** to remove the Options window. Different software will have options like this to control updates. Close all open windows (your web browser(s), Control Panel). You may log off of your Windows VM.
2. Start your Linux VM. Open a terminal window and su to root. There are several ways to install software in Linux. The simplest approach is with the program yum. Yum itself uses a program called rpm, the Red Hat Package Manager. We will start with it and then move on to yum. You can also install software using the installation CD or by downloading installation software from the Internet (like you did in step 1c). Finally, you can install open source software right from source code if you have access to the GNUs C compiler, gcc. We will look at all of these approaches.
   1. Start your Linux **web browser** and enter into the location box the URL **centos.karan.org/el4/extras/stable/i386/RPMS**. From the list running along the left hand column, select **Applications.TextBase**. From the list that appears in the middle, select the most recent version of **banner** (probably banner-1.3.1-2). This takes you to a page about the banner open source project. Select the link under **Download**. **Save** the file (do not open it in the Package Manager). This will save a file with a .rpm extension. Locate the file (probably your user’s Downloads directory such as ~foxr/Downloads). cd to that directory. Type **ls** to see if the file is there. If not, you either did not download it successfully or you are in the wrong directory. Repeat this step until you find the file. In order to install software using rpm, we use the command **rpm -i *filename*** where –i means “install” and the *filename* will be the full name of the .rpm file. In our case, the filename is probably banner-1.3.1-2.el4.kb.i386.rpm. Recall that tab completion can help you rather than typing the full name! Issue the rpm command. *What happens?* You will probably receive both a warning and an error. The warning can be ignored as it arises because the RPM file doesn’t have a key assuring that the software is valid. The error is more important. It should report that there are Failed dependencies. This means that files that need to be present to complete the installation are missing from your Linux system. How can we install this software? There is a website called **rpmfind.net/linux/rpm2html** which can locate RPM files for you that fulfill missing dependencies. In your browser, go there and enter **libc.so.6** in the search box. *What happened?* Let’s try something else. Type **GLIBC** (if you notice, the errors in RPM told us that libc.so.6 is part of GLIBC). *What happens when you search for GLIBC?* Now you have to select the proper GLIBC package, download that RPM file and install it. Unfortunately, not all of the dependent files (libraries) are present here so we won’t be able to install banner this way. We will return to this after the next step. You may close your web browser.
3. Next up, we briefly explore yum. yum is a program that uses rpm but yum automatically downloads RPM files and installs them for us, discovering dependencies and if possible, downloading the proper RPM files that resolve those dependencies. We will start with some simple yum commands. Type **yum list**. This lists all of the packages found at the repository that yum searches. There are a lot of them so they appear too rapidly in your terminal window to read. Don’t worry about the details. Let’s examine just a few up close. We can use wildcards to only produce a specific list. Type **yum list gcc\***. gcc is the GNU’s C compiler which we will need in step 2f. Here, we see several different gcc-related packages. In fact, the one we want to install is the first list, gcc.x86\_64. To install from yum, the command is **yum install *packagename*** where *packagename* is just the name of the software, gcc in our case. Type **yum –y install gcc**. The –y option tells yum to proceed with the installation rather than pausing to ask you for permission. In the printout, you will see a Transaction Summary. *How many total packages are being installed and how many are being Upgraded?* You will notice that these packages are all RPM files. In order to install using rpm, you would have had to download and install each of these in order. *What is the total download size? Did this software have a key?* To confirm that gcc has been installed, type **which gcc**. *Where is it stored?* You can also consult gcc’s man page which is now available.
4. Now we return to banner. We need to install a set of libraries to resolve the dependencies. We will use yum to install the set which is known as libc.so.6. Type **yum –y install libc.so.6**. This will take a couple of minutes. Now redo the rpm command from step 2a to try to install banner. You will receive the warning of no key but you will not receive any errors. Type **which** **banner**. *Where is banner installed?* To run banner, type **banner *string*** as in **banner hello**. Run banner with whatever string you wish. *What happens?* Using rpm can be easy if the rpm package’s dependencies are already installed. Tracking down the dependent files though can be a challenge and in some cases there might be dozens of dependent files needed. You can upgrade software by using the –U option, or delete software using –e.
5. At the top central part of your GUI, you may see this symbol: This symbol indicates that updates are available for Linux. If the symbol is present, **click** on it. The Software Update window opens up with a list of all of the software that current have updates available. *How many updates are there? What is the total size of the updates?* You can select any package or packages to update. If you were to install of the updates, it could take minutes or even hours. Instead, **Quit** out of this. As a system administrator, you would want to perform updates weekly if not more often.
6. From the **System** menu, select **Administration 🡪 Add/Remove Software**. In the left hand pane you will see the types of packages available. Expand **Applications**. **Click** on **Emacs**. In the right hand pane you will see a list of packages for the Emacs editor and supporting packages. Select **GNU Emacs text editor**. In the lower portion of the right pane is information about Emacs. *What is its download size? What is its Source?* Click **Apply**. *What happens?* **Install** any additional packages as needed. You will need to enter the root password when asked to perform the installation. You will see “Download packages”. *When complete, you will receive a “Run new application?” window.* Select **Close** and then **close** the **Add/Remove Software window**. Type **which emacs**. *Where is it stored?* Type **emacs &** to launch it. *What does the & do?* You will see that emacs runs in a graphical editor. You can use either the GUI features or control emacs using keystrokes. Review chapter 9 section 4 for details. **Exit** out of emacs.
7. We will download a different form of package to install by hand. This type of package has the actual source code so that we will have to compile it ourselves. We discuss why we might want to install from source code in class. To obtain the software package, we will use the wget program. From the command line type **wget www.adel.nursat.kz/apg/download/apg-2.2.3.tar.gz.** APG is the Linux Automated Password Generator program, open source. Type **ls** to make sure the download took place. This file is called apg-2.2.3.tar.gz. This file is both tarred and zipped so we have to use the tar program to unzip and unbundle it. Type **tar –xzf apg-2.2.3.tar.gz**. Remember that tab completion can help you type this. When you untar this, it creates a directory whose name is apg-2.2.3. cd to that directory. Type **ls**. *What do you see here?* View the README and INSTALL files in that order. *What type of information do you find in each file?* The directory has a few subdirectories, one of which is called doc. View doc’s contents and you will see a man subdirectory which contains the man page for apg. Installation is performed by calling upon the Makefile script, which we do by issuing the make command. Type **make**. After this completes, type **make install**. Both instructions cause a bunch of text to scroll by, ignore it. Type **which apg**. *Where is it stored?* Type **apg**. *What is the output?* Remember, apg generates random passwords. Type **apg –n 1**. *What happens this time?* Type **apg –m 15**. *What happens this time?* If you cannot answer these, read apg’s man page.
8. Now let’s examine the apg Makefile itself. Type **less Makefile**. The Makefile starts off by defining a number of variables. The CC variable defines the compiler instruction. *Which compiler is used? What compilation flags are used? What is the installation prefix?* This is used to determine where the program will ultimately be placed upon installation. *For which versions of Unix and Linux will apg operate?* About halfway through the file, you will see standalone:, all:, etc. These define what make should do when called with one of these terms such as make install or make all. *How many different commands can we use for make?* When done, type **cd ..** and then type **rm –rf apg\***. You may now shut down your Linux VM.

# Lab 14: Shell Scripting

Goals: to gain experience writing basic shell scripts in DOS and Linux; to experiment with passing parameters to scripts.

Introduction: scripting is an integral part of any system administrator’s job. In this lab, we look at very basic scripting commands in DOS and then move on to slightly more complicated scripts in Linux. In each part, you will first be presented with the scripting instructions to experiment with and then you will use this to write your own scripts from scratch. NOTE: Windows 10 has moved on to PowerShell which is a much more powerful and useful language; however since the textbook covers DOS, we will only cover DOS in this lab.

Time to completion: 2 hours

Before starting, review chapter 14 Scripting Languages.

1. Start your Windows VM. Open a **PowerShell window**. Also **open Notepad**. Change directory in PowerShell to your User’s home directory under C:\Users\Username. Type **cmd**. This command opens inside of PowerShell a DOS shell so that we can type in DOS commands. Note that if you close your PowerShell window, upon reopening it you will have to type **cmd** again.
   1. Type **echo Hello World!**. The echo statement outputs whatever you specify after echo. Type **set first=*Firstname***. Type **set last=*Lastname***. (where *Firstname* and *Lastname* are your first name and last name). Type **echo Hello %first% %last%, how are you?** *What output did you receive?* Type **date /T**. Type **time /T**. Type **hostname**. These are all DOS commands that operate from the command line but can also be placed into a script. In Notepad, enter the following:

**echo off**

**echo Date and Time:**

**date /T**

**time /T**

**hostname**

**set NAME=Richard Fox**

**echo Hello %NAME%, how are you?**

**Save** the file under your **User directory** as **first.bat**. To run this program, in the command prompt window, type **first.bat**. The bat extension stands for “batch” file, which is what all DOS scripts used to be called. Today, we more commonly see DOS scripts stored as .cmd (command) files or .btm files, but we will stick with .bat.

* 1. In order to get input from the user, you would use set /p VAR=prompting message, for instance, **set /p FIRST=Enter Your First Name** . Notice the blank spaces after “Name”, we add these to make sure that the prompt will appear followed by some blanks before the user enters their name. Before the set command in your script, add two instructions to obtain FIRST and LAST from the user. **Replace** the **set NAME** command in first.bat with **set NAME=%FIRST% %LAST%**. *Why do you suppose we put a blank space between %FIRST% and %LAST%?* **Save** the file and **rerun** your script and **input your name** when asked. *Did it work? What was the output of running first.bat?*
  2. **Open** Notepad to start a new file. The first line should be **echo off** (this should always be the first line in any DOS script to prevent every instruction from being echoed to the screen). For the second and third instruction type

**set FIRST=%1**

**set SECOND=%2**

%1 and %2 represent the first and second parameters passed to the script. You pass parameters from the command line when you call the script, for instance by typing second.bat string1 string2. The next instruction should repeat the set NAME instruction that you used as your replacement from part 1. Finally, **copy** the **echo statement** from first.bat as the last instruction in your new file. **Save** this file as **second.bat**. Run it from the DOS command line prompt as **second.bat *first* *last*** where *first* and *last* are your first and last names. *What was the output?* We will come back to this script in step 1f.

* 1. All variables in DOS store strings by default. To alter their behavior, we have to specify that the value should be treated numerically instead. Start a **new** Notepad file with the following. Remember to add blank spaces at the end of the prompting messages (after the word “number” in both the second and third instructions).

**echo off**

**set /p first=Enter a number**

**set /p second=Enter a second number**

**set sum=%first%+%second%**

**echo The sum is %sum%**

**Save** this file as **third.bat**. **Run** it entering 5 and 7 for the two numbers. *What is output? Why?* Add **/a** to the third set statement **between** the words **set** and **sum**. **Save** your file and **rerun** the script. *Did it work? What does /a refer to?*

* 1. An if statement tests a condition and if the condition is true, it performs the action provided, otherwise it does nothing. An if-else provides an alternate action so that if the condition is true, one action is performed and if false, a second action is performed. **Return** to the file **third.bat**. Add the following instruction after the last echo statement.

**if %first% gtr %second% (echo %first% is greater than %second%) else (echo %second% is greater than or equal to %first%)**

The if instruction must be placed on one line (do not press the enter key while typing it). **Save** the file and **rerun** the third.bat script several times, using different combinations of the two numbers to test all of the cases (for instance, 5 and 7, 7 and 5, 5 and 5, 5 and 0, 0 and 10, 0 and 0). *Did the script run as you expect?* Research the DOS scripting language (see the textbook). Modify the if statement so that it tests to see if the first is greater than or equal to the second outputs in the echo statement correctly for this change. *Insert your updated script as the answer to the last part of this step.*

* 1. **Return** to the script **second.bat**. Aside from comparing numbers as shown in part 4, you can compare strings by using the notation if %var% EQU %var% or %var% EQU value. **Modify** the **echo** statement in second.bat to be an if-else statement so that it compares the value stored in **%NAME%** to “*your full name*” (place your first and last name, with a space between them, in quote marks), and if there is a match (they are equal), output the greeting message as you have it, otherwise output the message “I do not know you, goodbye”. **Save** the file as **fourth.bat** and **run** it adding your first and last name as parameters as in **fourth.bat Richard Fox**. Did it work correctly?You will receive an error because %NAME% has two words in it and EQU is expecting to compare only a single item. To get this to work, use **“%NAME%”**. **Save** and **rerun** the script using your name as parameters. Did it work this time? If so, test it again using someone else’s name. Once your script is working correctly, *place your script in your answer file as the answer to this question.*

1. Start your Linux VM and open two terminal windows. Change directory to your user directory in both of your terminal windows. In one window, use vi to write your scripts and in the other window you will type Linux commands. Every Bash script will need to start with the following line:

**#!/bin/bash**

Additionally, after writing a new script and saving it, you must change its permissions to 745 before trying to execute it. To run a script named script1.sh, use the notation **./script1.sh**. Make sure you are in the same directory as your script! If you have parameters to pass to the script, the notation would be **./script1.sh param1 param2 param3** etc.

* 1. For your first script, beneath the **#!/bin/bash** line, add the following three lines:

**echo Hello $USER**

**echo You are logged into $HOSTNAME**

**echo And currently at $PWD**

**Save** this script **as fifth.sh**, **change** its **permissions** and **run** it. This script demonstrates the use of environment variables in a script. You can also define your own variables. **Alter** the script so that you have **FIRST=*firstname***, **LAST=*lastname***, **FULLNAME=“$FIRST $LAST”**, on three separate lines (where *firstname* and *lastname* are your first and last names). Now **change** the first **echo** statement to be **echo Hello $FULLNAME**. **Save** and **run** the script. *Did it work?* **Replace $FULLNAME** with **FULLNAME**, **save** the file and **run** it. *How did the output differ?* **Add** the instruction **echo Today is `date`** at the end of the file, **save** your script and **run** it again. *What do the ` ` do? What happens if you use echo date without the quote (tick) marks?*

* 1. We can obtain data from the user in two ways, through read statements and via parameters. These are both similar to what we did with DOS. Copy **fifth.sh** to **sixth.sh** using the appropriate Linux cp command, and **edit** **sixth.sh** in **vi**. **Change** the first **two** assignment statements to be **FIRST=$1** and **LAST=$2** respectively. $1 and $2 represent the first two parameters passed to the script. **Save** the script, **change** the permissions of sixth.sh to 745 and run it as **./sixth.sh**. *What is the output?* Now run it as **./sixth.sh *first last*** where *first* and *last* are your first and last names. **Rerun** it with only **one** parameter. *What happens?* Try it with 3 parameters. *What happens?* Now **edit** this file and **remove** the first **two** assignment statements (FIRST=$1, LAST=$2), replacing them with the following:

**echo –n “Enter your first name ”**

**read FIRST**

**echo –n “Enter your last name ”**

**read LAST**

**Save** the file and **run** it (without parameters). When prompted by the program, **input your first** and **last names**. *Did it work?* **Remove one** of the **–n**’s, **save** the file and **rerun** the script*. What does the –n do in echo?*

* 1. **Create** a new script called **seventh.sh**. Use the following code.

**#!/bin/bash**

**for I in 1 2 3 4 5; do echo $I; done**

The for loop iterates over the list so that the variable I takes on each value of 1, 2, 3, 4 and 5. The for loop body then outputs each number, one at a time. **Save** the file, **change** the file’s permissions and **run** the script. *What was the output?* Now **modify** the **for loop’s body** (echo $I) replacing it with **sum=$((sum+I))** and **add echo $sum** after the for loop (after the **done** statement). **Save** the file and **run** it. *What happens?* *What does the instruction sum=$((sum+I)) do? What would happen if you placed echo $sum before the word done instead of after?*

* 1. **Modify** seventh.sh by **replacing** the **list** of values 1 2 3 4 5 with **$@**. The $@ notation means “the list of parameters passed to this script”. In this case, the script expects to receive some parameters. For each parameter, it is added to sum. **Save** the file and **run** it as **./seventh.sh 50 13 18 22 61 48**. This will sum up the values. *What sum did you get?* **Rerun** the script with no parameters (that is, rerun it as **./seventh.sh**). *What is the output?*
  2. Bash has if statements like DOS, but the Bash if statement is more useful. One thing we might use the if statement for is to test the number of parameters that the user supplied when invoking the script. In the case of seventh.sh, we expect *at least* 1 parameter. You can obtain the number of parameters with the notation $#. You can test to see if the user supplied parameters using an if statement with the notation: **if [ $# -gt 0 ]; then** … **fi** The word fi indicates that this ends the if statement.Prior to the for loop in **seventh.sh**, **add** **the above if statement** where the instruction in the then portion is **an echo Warning, no parameters supplied!**. Don’t forget to add the fi statement. **Save** the script and **run** **it** with **several** parameters and then with **0** parameters. *Does it work?* The if-then-else statement, like with DOS, allows you to have two actions, one if the condition is true and one otherwise. We want to perform the sum and the output if the condition to our if statement is false. That is, if the condition is true, output the warning message otherwise do the for loop and the last echo statement (to output sum). So, **inside the if statement**, before the fi, **move** the entire for loop and then the echo $sum statement. **Save** your script and **run** it with **no** parameters and rerun it with **several** parameters. *Once you have it working correctly, insert the script into your answer to this step.*
  3. **Write** a script **eighth.sh** which expects to receive two numbers. If the script receives two parameters (test this using $# in an if statement using the comparison –eq), compare the two parameters to see which is greater and output the larger of the two (hint: use logic similar to the DOS script from step 1e). Note that if they are the same value, it doesn’t matter which one you print out. This script has two if statements, to the structure of this script will look something like this:

**if [ *compare* $# *to 2* ]; then**

**if [ *compare* $1 and $2 ]; then**

***Output***

**else**

***Output***

**fi**

**else *Output a warning***

**fi**

**Save** the script, **change** its permissions and **run** the script. The script expects parameters, so you might call it using notation like **./eighth.sh 15 20**. Run it with several variations (2 parameters where the first is larger, where the second is larger, where they are equal, where there is 0 or 1 parameter, where there is more than 2 parameters). *Once it is running, insert the script as the answer to this part.*

* 1. **Write** a script called **ninth.sh** containing the following:

**#!/bin/bash**

**for file in \*; do**

**if [ -r $file ]; then echo $file is readable; fi**

**done**

**Save** this script, **change** its permissions and **run** it. The condition **[ -r $file ]** tests each item $file (which itself is each file found in the current directory) to see if it is a readable file. The notation for file in \*; means to iterate over \*, which Bash expands to be all items in the current directory. You should be able to make sense of what the script does. Now **alter** this script so that rather than iterating over **\***, it iterates over the parameters passed to it. Recall how to obtain the list of parameters from step 2d. When you run the script now, you must supply it with a list of parameters which in this case should be file and directory names. The files and directories do not have to be in the current directory as long as you supply the path to each. **Save** and **run** the script. For instance, you might try

**./ninth.sh /etc/passwd /etc/shadow /etc/group**

**/sbin/ifconfig /usr/sbin/useradd**

Once ninth.sh is running, modify it as follows. We can test multiple conditions by using [[ condition1 && condition2 ]] or by using [[ condition1 || condition2 ]]. The && means “and” (both conditions must be true to evaluate to true) and the || means “or” (the condition is true if either conditions are true. We want to count all of the files given as parameters that are either readable or executable. We saw that readable is –r $file, executable is –x $file. Rather than outputting the file name as we did above, we want to add 1 to a counter variable. This is accomplished using **COUNT=$((COUNT+1))**. This instruction would be placed in the then clause instead of the echo. Finally, after the for loop ends (after the done), output the value in COUNT. **Revise** ninth.sh for these changes and **save it**. Now **run** it using the following parameters: **/etc/passwd, /etc/shadow, seventh.sh, /sbin/ifconfig, /usr/sbin/useradd**. *Once it is running correctly, insert the script as the answer to this part.* Once you have all of the scripts placed in your answers, you may log off of your Linux VM.

# Lab 15: Case Study

Goals: to learn about and discuss ethical issues related to the IT field.

Introduction: in this lab, we examine several ethical dilemmas. We provide several work-related (or school-related) scenarios and you are asked to report on how you would handle them. Research each of these in advance by using the information on the page 79 of the lab manual. The result of this lab consists of both in-class discussion and a written paper to submit to your instructor.

Time to completion: 3 hours

Before starting, review chapter 16 IT ETHICS and the material on page 79.

This lab involves three things, participating in class discussion, doing some research, and writing an essay. Your grade will be based on your class participation and your submitted essay. Below are several scenarios. In lab, you will listen to the instructor present each scenario and then you will discuss the options available as an IT individual. Your discussion should not just be based on your own opinion but should incorporate ethics, famous situations that you may know about that are related and/or situations that you have experienced yourself.

After lab, you are to select one of these scenarios, research it further and submit a 500 word write-up of the situation, the possible solutions, the legal, social and ethical ramifications and your own suggestion. The instructor may provide supplemental resources for you to utilize.

1. As the system administrator for your small company, your boss has asked you to start monitoring your employee’s Internet behavior during work hours. Specifically, he wants you to search through your proxy server logs to see if employees are visiting sites that are not related to the company. The company does not have any stated policy regarding the use of work computers, nor has the boss ever expressed a prior concern that employees should not use computers for non-professional use. What should you do?
2. Your large company has a written policy against personnel using work email for personal or private use. Your boss has asked you to read through employee emails to see if any are violating the policy. What should you do?
3. While examining your Linux file server for excessive disk usage, you have found that an employee has filled his directory with what appears to be illegally downloaded music. There is no stated policy that permits you to explore employee directories, but at the same time, your job is to ensure that the information technology infrastructure is running efficiently and you have found that your file server has very little free space available. What should you do?
4. As a web developer, you have decided to use a commercial product to help produce a company’s web site. You have obtained the commercial product via a free 30-day trial. Do you 1. need to inform the company of the product you have chosen and 2. obtain further licenses permitting the web site to go on line? Did you make an ethical mistake in choosing this product?
5. As a student in IT, you have completed the much-hated CIT 470 term project of Dr. Walden. A friend of yours is struggling to complete the lab and asks for your help. You discover several problems with their installation. What should you do? To what extent should you help them? Not at all? Give them a few hints? Show them specifically what is wrong? Share with them your own solution?
6. Your best friend was the system administrator for petcareofcincinnati.com but was fired recently. He told you that prior to being laid off, he planted a back door (a secret account) into their computers and plans retribution on this company by breaking into their computer system and wreaking havoc. What should you do?

The following topics should be mentioned in your essay. Your essay should make it clear that you have a basic understanding of the legal/ethical issues involved in the case you have selected.

**Aiding and Abetting** - whoever commits an offense against the United States *or* aids, abets, counsels, commands, induces or procures its commission, is punishable as a principal. The word “abet” essentially means to encourage. Can neglecting to notify authorities of a crime that will (or might) be committed make you an accomplice?

**Reasonable expectation of privacy** -Under current law, to establish a reasonable expectation of privacy a person must establish two things: that the individual had a subjective expectation of privacy; and that that subjective expectation of privacy is one that society is prepared to recognize as reasonable. What impact has technology had on the expectation of privacy?

**Computer Fraud and Abuse Act** - a law passed by the United States Congress in 1986, intended to reduce cracking of computer systems and to address federal computer-related offenses; sections 4, 5 and 6 of the law are particularly relevant. This law has been used in several high-profile cases (see Wikipedia). Do the results for any of these cases affect your scenario?

**NKU’s Code of Student Rights and Responsibilities** - a description of a student’s rights and responsibilities. Were you aware that there was such a document? Are you surprised by anything mentioned in it?

What follows are resources that might help you write your essay.

* [**Wikipedia**](file:///\\fileserv3.hh.nku.edu\Departments$\Computer%20Science\CIT%20130\fall%202012%20001-004\labs\en.wikipedia.org) – A good point of entry for most any subject. (Yes, I know anyone can edit it.) Be sure to check the cited sources before accepting any claim therein.
  + <http://en.wikipedia.org/wiki/Legal_aspects_of_computing>
  + <http://en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act>
  + <http://en.wikipedia.org/wiki/Aiding_and_abetting>
  + <http://en.wikipedia.org/wiki/Expectation_of_privacy>
* [**itlaw.wikia.com**](file:///\\fileserv3.hh.nku.edu\Departments$\Computer%20Science\CIT%20130\fall%202012%20001-004\labs\itlaw.wikia.com) – A free content encyclopedia with a focus on IT-related law. (It can also be edited by anyone, but might be more reliable in some cases than Wikipedia.)
  + <http://itlaw.wikia.com/wiki/Reasonable_expectation_of_privacy>
  + <http://itlaw.wikia.com/wiki/Privacy>
  + <http://itlaw.wikia.com/wiki/Information_technology_law>
* [**privacyrights.org**](https://www.privacyrights.org/) – It has a special section for work related privacy as well as general online privacy.
  + <https://www.privacyrights.org/fs/fs7-work.htm>
  + <https://www.privacyrights.org/fs/fs18-cyb.htm>
* [**NKU Code of Student Rights and Responsibilities**](http://deanofstudents.nku.edu/codes_and_policies/codeofstudent_rights/index.php)– This document is available online. You should be familiar with it since it affects all of your NKU-related activities while a student.
* [ACM Code of Ethics and Professional Conduct](http://www.acm.org/about/code-of-ethics) – The ACM (Association of Computing Machinery) is an association for students, professionals and academics in fields related to computer science and information technology. Their *code of ethics* provides an ethical guideline for the proper use of technology.