Session 1 - Introduction to Linux

Installing Linux in VirtualBox

During this session we will start with installing VirtualBox on your computer. VirtualBox is virtualization software which lets you run other operating systems on your computer.

Enable Virtualization

Before we start with the Linux installation we must verify that your laptop's Virtualization Extensions are enabled in the BIOS:

1. Reboot your computer
2. When you see the BIOS splash screen look for the key to press to enter the BIOS Setup menu
    a. It will probably be one of F2, F9, F11, F12
    b. Press the key in order to get into the BIOS
3. Find the menu option which allows you to enable Virtualization
4. Save and exit
    a. Typically F10
5. Boot up your laptop

Download VirtualBox

We will download the VirtualBox software from this website:
https://www.virtualbox.org/wiki/Downloads

1. Download the proper VirtualBox package (Windows or OSX) depending on the type of computer you have
2. Now download the VirtualBox Guest Additions
3. Install the VirtualBox software first (as Administrator)
4. Now install the VirtualBox Guest Additions (as Administrator)

Download and install Ubuntu

Note: you need to have a computer with at least 4GB of RAM to effectively run Ubuntu. Less than this will result in a very slow and sluggish system that will be very difficult to work with.

We will download the Ubuntu installation image from:
https://www.ubuntu.com/download

1. Download the Ubuntu 16.04.3 LTS installation image
2. Start VirtualBox Manager

3. Click the “New” button to create a new virtual environment
    a. Name: Ubuntu
    b. Type: Linux
    c. Version: Ubuntu (64-bit)
    d. Memory size: 2048 MB
    e. Hard disk: Create a virtual hard disk now
    f. Click “Create” button
    g. File location: Ubuntu
    h. File size: 32 GB
    i. Hard disk file type: VDI (VirtualBox Disk Image)
    j. Storage on physical hard disk: Dynamically allocated
    k. Click “Create” button.
4. In the left panel of the VirtualBox Manager software click on the VM and then click the green “Start” button
5. It will prompt you for a start-up disk
    a. Click the small folder with a green arrow icon on the right of the popup box
    b. Navigate to where you saved the Ubuntu installation ISO and select that for your startup disk
    c. Then click the “Start” button. The installation will begin
6. Click the “Install Ubuntu” button
7. Click the “Download updates…” and “Install third-party software…” check boxes then “Continue”
8. Click “Erase disk…” if necessary then “Install Now”
9. Click “Continue” to write the changes to disk
10. Click on the map to set your location to “Denver” then “Continue”
11. Accept the defaults of “English (US)” keyboard then “Continue”
12. Your name: *enter your first name in lower-case*
13. Your computer’s name: *this will be auto-populated*
14. Pick a username: *this will be auto-populated*
15. Choose a password: enter a password you can remember
16. Confirm your password: re-enter the password
17. Click “Continue” and let the installer run
18. Click “Reboot” when the installation is finished
19. Additionally, press “Enter” if necessary

First Login and Updates

1. When your VM (virtual machine) is rebooted log in with the username and password you created during installation
2. Tap your “Windows” key on your keyboard if your computer is running Windows or the “Command” key if your computer is a MAC

    a. If neither of these options work click on the round white and purple button to bring up the search utility
3. Type **term** in the search window then click on the Terminal icon which is displayed
4. In the terminal window type: `sudo apt-get update` followed by your password
    a. This will run a command which updates the list of available software packages
5. Click on “Devices” in the top-most menu
    a. select “Insert Guest Additions CD Image…”
    b. A popup will ask you if you want to run the software on the Guest Additions CD, click “Yes”
    c. Once again, type in your password then click “Authenticate”
    d. The guest additions will be built and installed
    e. Next press “Return” to close the window
6. Back in the terminal window type `sudo poweroff` to shutdown the VM

Installing Guest Additions

1. Click on the VM in the left panel of the VirtualBox Manager
2. Click the “Settings” button
3. A “Settings” window will be displayed
4. Click on “General” in the left panel then click the “Advanced” tab in the right panel and change these settings:
    a. Shared Clipboard: Bidirectional
    b. Drag’n’Drop: Bidirectional
    c. Then click “OK”
5. Click “Start” to startup your VM
6. Log in when your VM is finished booting

Exploring Your Desktop

1. Press the Settings icon in the top far-right corner of the screen
    a. Then select “About This Computer”
    b. Explore the information contained in the new popup window
    c. How much memory does your VM contain?
    d. What kind of Processor does your VM have?
    e. Read the “Legal Notice”. Is there anything to be concerned about?
2. Clicking on the “All Settings” tab gives you many more areas to explore
    a. To return back to the previous screen click on the “Details” icon

Computer Architecture

Your computer is comprised of many components which serve different purposes within the system. We will discuss a few of the more prominent ones.

CPU

What is a CPU? It is the central processing unit (or processor) of a computer. The CPU carries out the instructions of a program by performing control, input/output, arithmetic, and logical operations. It is the brains of your computer.

A CPU implements an instruction set for carrying out computations. An instruction set is the set of operations the CPU can perform. Some examples of instructions are

- Add
- Subtract
- Multiply
- Divide
- Compare
- Jump
- Increment
- Decrement

Some CPU's like Intel and AMD have compatible implementations of instruction sets

- The x86 instruction set was created by Intel based on the 8086 CPU and implemented by other CPU manufacturers like AMD, Cyrix, and VIA
- The x86_64 instruction set, which is a 64-bit extension of the x86 instruction set, was created by AMD and implemented by other companies like Intel and VIA

Components of a CPU

- Arithmetic Logic Unit (ALU) which performs the arithmetic and logical operations
- Control Unit (CU) which retrieves instructions from memory and decodes and executes them. The CU calls on the ALU when necessary
- Registers are small memory locations in the CPU which can hold data. The registers come in increments of 8 bits. 8 bits is the same as 1 byte. For example the registers in a 32-bit system are 4 bytes in length and the registers in a 64-bit system are 8 bytes in length.

What is the difference between 16-bit, 32-bit, and a 64-bit CPU's?
In the 1960's through 1970's 16 bit processors were being developed for use with the earliest computers. In June 1978 Intel release a 16-bit CPU called the 8086 for the fledgling home PC market. The term 16-bit/32-bit/64-bit CPU refers to:

- The ***primary meaning*** refers to the size of the CPU registers. A register on a 16-bit CPU can store 16 bits of data; a register on a 32-bit system can store 32 bits of data; a register on a 64-bit system can store 64 bits of data
    - A bit is the smallest unit of data in a computer. It can hold only 2 values: 0 or 1
- The size of addressable memory
- The range of values which can be stored in 16 bits (binary) is `00000000 00000000` to `11111111 11111111` which equals 0 through 65535.
- The range of values which can be stored in 32 bits (binary) is `00000000 00000000 00000000 00000000` to `11111111 11111111 11111111 11111111` which equals 0 through 4294967295.
- The range of values for a 64-bit address is similar to the two examples above: 64 zeros through 64 ones. The range of values for a 64-bit binary number 0 through 18446744073709551615 (16 exabytes)

The effects of having a 64-bit CPU vs. a 32-bit CPU is that the computer can access **MUCH MORE** memory (RAM) and the data travels down the busses quicker because the data path is wider. 64-bit CPU's are the norm for all computers, cell phones, and tablets.

In the PC market Intel x86_64 processors have the most market share. Windows computers, MACs, and Linux all run on Intel x86_64. The x86_64 designator specifies the CPU instruction set architecture.

Run the following commands to find your processor type:

1. On Windows, in a cmd window type `set` and press enter. Look for the `PROCESSOR_IDENTIFIER` and `NUMBER_OF_PROCESSORS` fields
2. On Mac, in a terminal window type `% sysctl -n machdep.cpu.brand_string` and press enter
3. On Linux, in a terminal window type `cat /proc/cpuinfo` and press enter

Memory

Computer memory called RAM (Random Access Memory) is a form of data storage which your computer uses while it is powered on. RAM is volatile memory which means it loses the values stored in it when the power is removed from the system. The amount of memory accessible by programs running on your computer is dependent on the instruction set. If the computer is a 32-bit computer it has a 32-bit instruction set and the amount of memory available to programs is $2^{32} - 1$ = 4294967295 (4 GB of memory). If a computer is a 64-bit computer it actually only allows $2^{48} - 1$ = 281474976710655 (256 TB of memory) in the virtual address space.

Disk Drive

A disk drive is a storage mechanism where data is recorded on various types of rotating disks. There are optical disks (CDs and DVDs), disk drives (hard drives), and floppy drives (but they

have been obsolete for decades now). Disk drives are non-volatile so they retain the data stored on them after power is removed from the system.

Thumb Drive

Also known as a USB flash drive is another data storage device which uses non-volatile flash memory. Thumb drives are often used to transport files between systems because of their ease of use and compatibility with almost all computers.

Keyboard and Mouse

These are the typical and most often used input devices for a computer. Laptops have keyboards with a mouse device integrated into them.

Operating Systems

Definition of an Operating System

An operating system is the base software which manages the hardware components and software resources of a computer system. All programs which users interact with require an operating system to run upon.

History of UNIX

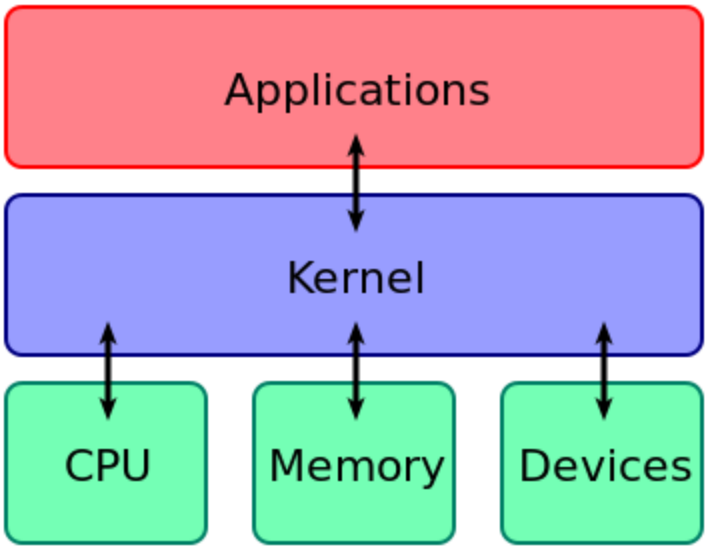
- UNIX was developed in the 1970's by AT&T Labs and the source code was leased out to companies and universities.
- Solaris is a UNIX implementation developed by Sun Microsystems (now Oracle Corp.) based on the UNIX specification
- HP/UX is a UNIX implementation developed by Hewlett-Packard Corporation based on the UNIX specification
- FreeBSD (the core of Apple's' OSX operating system) was developed by the University of California at Berkeley back in the 1970's based upon the Unix source code leased from AT&T labs. FreeBSD does not contain any AT&T source code, the code from AT&T was rewritten and made open-source and thus FreeBSD is a free distribution like Linux
- Minix was a small UNIX-like operating system written by a Computer Science professor in the Netherlands named Andrew Tanenbaum. A book about the AT&T UNIX source code (version 6) had been written by John Lions (University of New South Wales) which university professors around the world had been using to teach their students about UNIX. When AT&T found out that their UNIX source code was being used to teach university students they changed the license agreement in version 7 to prohibit the use of the source code for teaching purposes. Because of this restriction Tanenbaum wrote a completely new operating system called Minix based on his knowledge of the AT&T source code for use in his teaching and for his students to learn from. Minix is still around and is free and open source.

- Linux was written by Linus Torvalds (a college student at the University of Helsinki). Linus gained inspiration for the Linux project by studying the source code of Minix. Linus wrote the Linux kernel from scratch. Linux does not use or borrow any of the AT&T source code.
- https://en.wikipedia.org/wiki/Unix#/media/File:Unix_history-simple.svg

Other Operating Systems

- Android is the most used operating system on mobile devices in the world. It is based on the Linux kernel. The Android operating system is open source and has been released by Google under an open source license.
- iOS is the an operating system developed by Apple for use on their devices. iOS is the second most popular operating system for mobile devices.
- Windows is a closed source (proprietary) operating system from the Microsoft Corporation. There have been many versions of Windows over the years. Windows 7, 8.1, and 10 are all very prominent today.

The Linux Kernel



https://en.wikipedia.org/wiki/Kernel_(operating_system)#/media/File:Kernel_Layout.svg

A kernel is the core program of an operating system which manages the hardware of the computer and provides interfaces for applications running on the system. The kernel is the first thing loaded on system startup and takes over the remaining tasks while the system is booting up. All operating systems have a kernel; operating systems like Windows, OSX, and Linux have differing types of kernels.

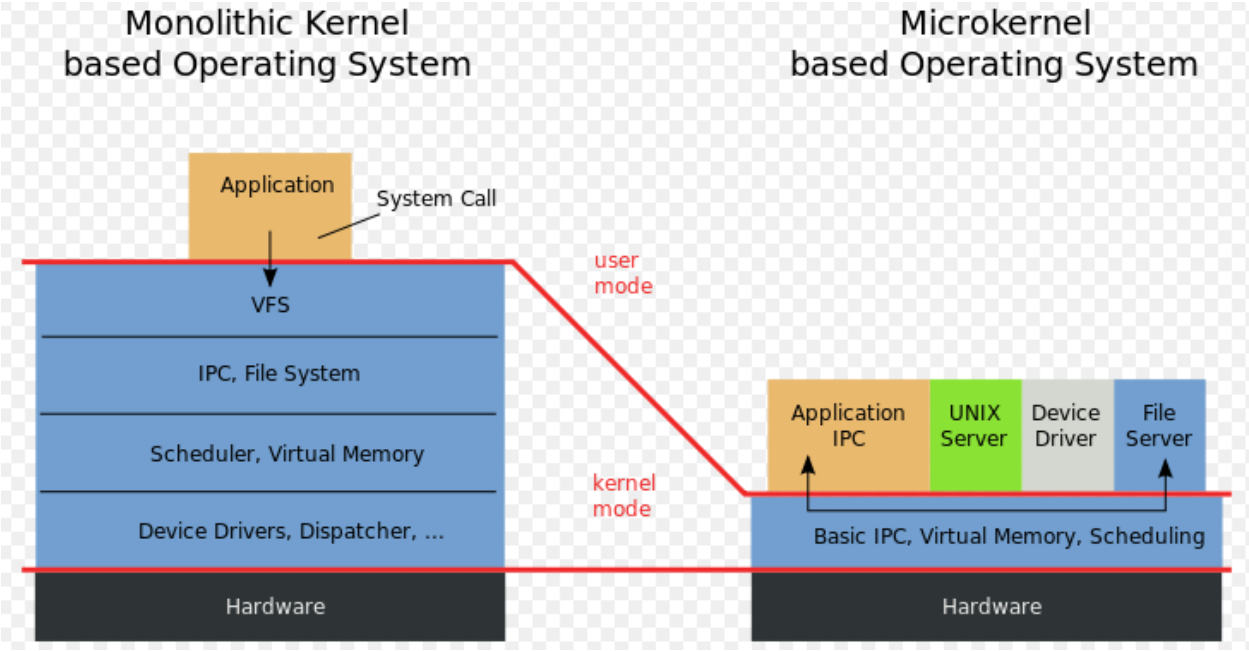
Computers are made to process data and preserving data integrity is the kernel's highest priority. Having a computer system that does not protect data integrity is a useless system. What good is using a system when you are not guaranteed that the data in the system will be protected?
Because of the importance of this task, the kernel manages access to the data on a computer; regular users are not allowed unmanaged access to computer data. The kernel runs in a protected area of memory called *kernel space*. This area of memory is protected from *user space* programs. Remember, the kernel is used to manage system resources like CPU, memory, disk drives, but its number one priority is preserving data integrity. Have you ever seen a "Stop Error" screen (aka BSOD - blue screen of death) on Windows or a kernel panic on Linux?

> A kernel panic also known as computer death or PC death, is a safety measure taken by an operating system's kernel upon detecting an internal fatal error in which it either is unable to safely recover from or cannot have the system continue to run without having a much higher risk of major data loss.
>
> https://en.wikipedia.org/wiki/Kernel_panic

The Linux kernel was developed by Linus Torvalds in 1991. As the kernel quickly matured (with the help from other developers) many pieces of software were integrated into Linux from the GNU project. The GNU project is a Unix-like operating system started by Richard Stallman (from MIT) in 1983. The GNU project had many operating system utilities like shells, core utilities, compilers, libraries, etc) already developed but the GNU kernel was not complete. The completed utilities of GNU were combined with the Linux kernel to form a completely free and open source operating system called GNU/Linux.
Linus took advantage of tools which were available to him while developing the Linux kernel. For example, Linus used the Minix operating system for developing the Linux kernel. He took the source code for the GNU gcc compiler and compiled it on Minix using the Minix cc C compiler. Then he used the gcc compiler to develop the Linux kernel.

Monolithic and Micro Kernels

The Linux kernel is *monolithic* meaning that the entire kernel (kernel, resources, and device drivers) runs in the privileged memory space and the kernel runs in one process address space. OSX and Windows NT use a *microkernel*. A microkernel is a kernel which offers only a few services and it runs in privileged memory space. The other services and device drivers run as services in unprivileged/user memory space. This makes the microkernel (theoretically) more secure than monolithic kernels since a buggy device driver cannot crash the entire kernel. Microkernels are generally a few percent slower than monolithic kernels.



https://en.wikipedia.org/wiki/Microkernel#/media/File:OS-structure.svg

Processes and Process ID's

In Linux a process is a running instance of a computer program. Running processes are owned by the user who launches the process and the processes are assigned a process identifier (PID). The ps command lists currently running processes. In a terminal window run

```
ps -ef | less
```

This command lists the programs currently running on your Linux system. The first column shows the owner of the process (user ID - UID), the second column shows the PID, the third column shows the Parent PID (PPID), and the last column shows the name of the running process.

At boot-up the Linux kernel (PID 0) starts up a process called init (PID 1) and a thread daemon (kthreadd PID 2). kthreadd is used to start up certain kernel threads running kernel code which are managed as processes. init is responsible for launching critical system services. Run the following commands to see the process trees of kthreadd and init.

```
pstree 1 | less
pstree 2 | less
```

New processes are created when a parent process "forks" a new process. There are two important commands used in forking (duplicating) a process; they are fork and exec. For example if you are in a terminal window and you type `echo $$` and press enter it will show you the current PID of the terminal window you are running in. Now type `xclock` and press enter; you will see a new application which displays a clock.

In a separate terminal window type

```
pstree -a PID from the echo $$ you just ran
```

This will show you that `bash` is the parent of `xclock`. When you run the `xclock` command the `bash` process calls `fork()` which creates a new process which is an exact duplicate of the current `bash` process. Next, the `exec(xclock)` command is called; this command reads the `xclock` program from disk and replaces the contents of the new `bash` process with `xclock` program and begins running the new `xclock` process.
This process is important to remember! To start a new process Linuxs fork()'s then exec()'s.

Daemons

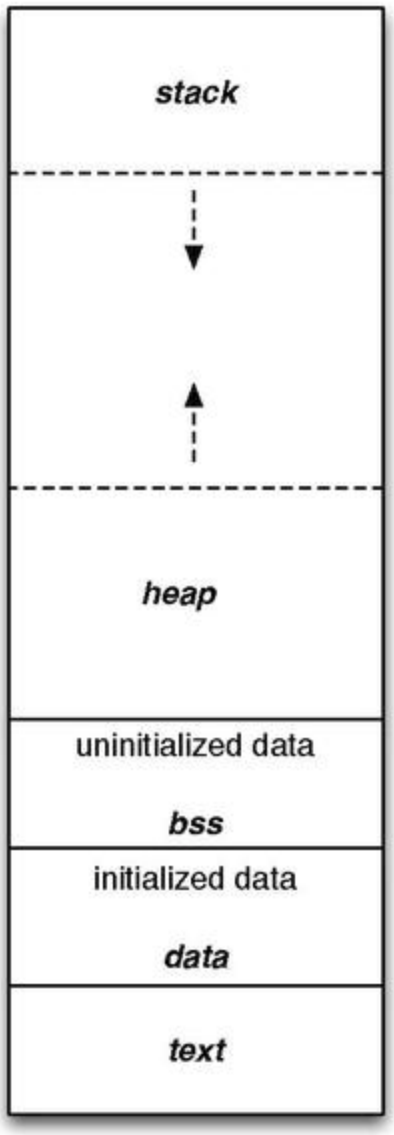
A daemon is a long-running process which executes in the background and is not interactively controlled by a user. Daemon process names often end with a 'd' to separate them from other processes. Daemons are usually started at bootup and they remain running until the system is shut down. A few examples of daemons are rsyslogd (which handles system log messages), sshd (which handles incoming ssh login requests), and httpd (which handles incoming requests for Web pages).

Virtual Memory

When processes are started in Linux they have a process address space allocated to them but they do not directly access physical memory. Linux is a virtual memory operating system meaning that memory presented to system processes have a virtual address which is mapped to a physical memory address. These mappings between virtual address and physical address are maintained in kernel memory mapping tables.

Theoretically a process on a 64-bit OS has a virtual memory address range from 0 to $2^{64} - 1$.

Two raised to the 64th power is an incredibly huge number, approximately 18 Quintillion or 16 Exabytes. There are no computers with 16 Exabytes of RAM. Processes on 64-bit operating systems use a notation which theoretically allows that much memory but it is never actually realized. Currently Intel and AMD 64-bit processors only use the lower 48 bits of the 64-bit address space for process memory. This is still a huge number, 256 Terabytes for process address space. Since current computers do not have physical memory this large virtual memory mapping is used. As a process runs, any memory which is needed is dynamically allocated and the virtual-to-physical mapping is done during runtime. Each process has its own virtual address space and thus they are protected from the actions of other processes. The kernel maintains the integrity of each process's memory by allocating a dedicated address space to each process and processes are only allowed access to their own virtual memory.



The **stack** is the memory portion of the program which contains automatically created variables and variables which belong to a function invocation. The stack is typically allocated at the end of the memory address space and as things are added to the stack it grows downward. The stack is a last-in-first-out construct
The **heap** area is placed below the stack with a large unallocated area between them. The heap contains memory allocated with functions like malloc, calloc, and realloc. Memory is released from the heap by using the free() function.
The **bss** (block started by symbol) area contains global and statically allocated variables which are uninitialized (set to zero by default)
The **data** area contains global or static variables which have a predefined value or can be modified. These would be variable defined outside the scope of functions or declared inside a function with the static keyword.
The **text** area contains the executable instructions of the program. This area is usually fixed size and read-only.

https://en.wikipedia.org/wiki/File:Program_memory_layout.pdf

Filesystems

The term filesystem means two things in the context of Linux. The first usage describes the methods and data structures that an operating system uses to keep track of files on a disk or a disk partition; that is, the way the files are organized on the disk.
The second usage describes an instance of the filesystem (first meaning) on a disk or disk partition. One might say they have two filesystems on their disk drive (if it is divided into two partitions and each partition contains a filesystem).
Most applications on Linux computers interact with the filesystem; they do not interact directly with the raw sectors of a disk drive. One exception would be the Linux program `mkfs` (make filesystem) which creates the filesystem on the raw disk device.
`mkfs` initializes the raw disk and creates the bookkeeping data structures on the disk which are necessary for the filesystem to work. The bookkeeping data structures written to a disk include:

- Superblock - contains information about the filesystem as a whole, like the size
- Inode - contains all the information about a file except its name. The inode contains the number of the data blocks used by the file.
- Data block - the actual file data is stored in data blocks
- Directory block - filenames are stored in a directory along with the number of the inode corresponding to the file
- Indirection block - pointed to by an inode when the inode does not have enough space to hold all the numbers of the data blocks comprising a file

Linux systems do not have the concept of drives (c:\ or d:\) like Windows. Instead it has a single hierarchical directory structure designated by a forward slash `/` and known as the root partition. Everything in the Linux filesystem starts from this root partition and expands into subdirectories. A filesystem on a disk or disk partition is not accessible until it is mounted by the kernel. The kernel first mounts the root partition at bootup and other partitions are mounted on mount points contained on the root filesystem. When a Linux system shuts down gracefully (by using init 0, poweroff, shutdown) it writes any in-memory cached filesystem data to the physical disk before it unmounts the partition. One of the implications of this method is that you should always gracefully shutdown your Linux system. You risk corrupting your filesystem if you just power off your system.

http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/foreward.html

Session 1 Homework

Use Google to look up the Usage Share of Operating Systems on the internet. Find the percentage breakdown of operating systems for:

- Desktop and laptop computers
- Mobile devices
- Public servers on the internet

Bring your research results to the next class period and we will share and compare the answers we got to see if we can come up with a consensus for these figures.

References:

https://en.wikipedia.org
http://www.webopedia.com/quick_ref/computer-architecture-study-guide.html
http://tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf
http://www.tldp.org/LDP/abs/abs-guide.pdf