Revised: 11/24/2019

Linux Installation and Shell Basics

Michael C. Hackett

Computer Science Department

Community College of Philadelphia

Lecture Topics

- Preparing for Installation
- Installation Media
- Performing the Installation
  - Starting the installation
  - Choosing system options
  - Configuring hard disks
  - Configuring user accounts
- Shells and Terminals
- The Bash Shell
  - Bash Syntax
  - Common Bash Commands
  - Shell Metacharacters
  - Shell Command Help
  - Shell Tips and Tricks
- Shutting Down

Preparing for Installation

- Before installing a distribution, you must ensure the computer meets the distribution's ***minimum system requirements***.
  - Minimum space (memory), speed (processor), and peripheral devices
- Each distribution will specify the minimum requirements.
  - Typically they post it on their website
- For example, Ubuntu's *recommended* minimum system requirements:
  - https://help.ubuntu.com/community/Installation/SystemRequirements

Preparing for Installation

- Many computers have peripheral devices like network interface cards, sound cards, video cards, etc.
- Most modern distributions support nearly all hardware components.
- Most distributions and some hardware manufacturers have a Hardware Compatibility List (HCL) for checking Linux compatibility.

Preparing for Installation

- For example, here is openSUSE's HCL for NVIDIA graphics cards:
  - https://en.opensuse.org/HCL:NVIDIA_video_cards

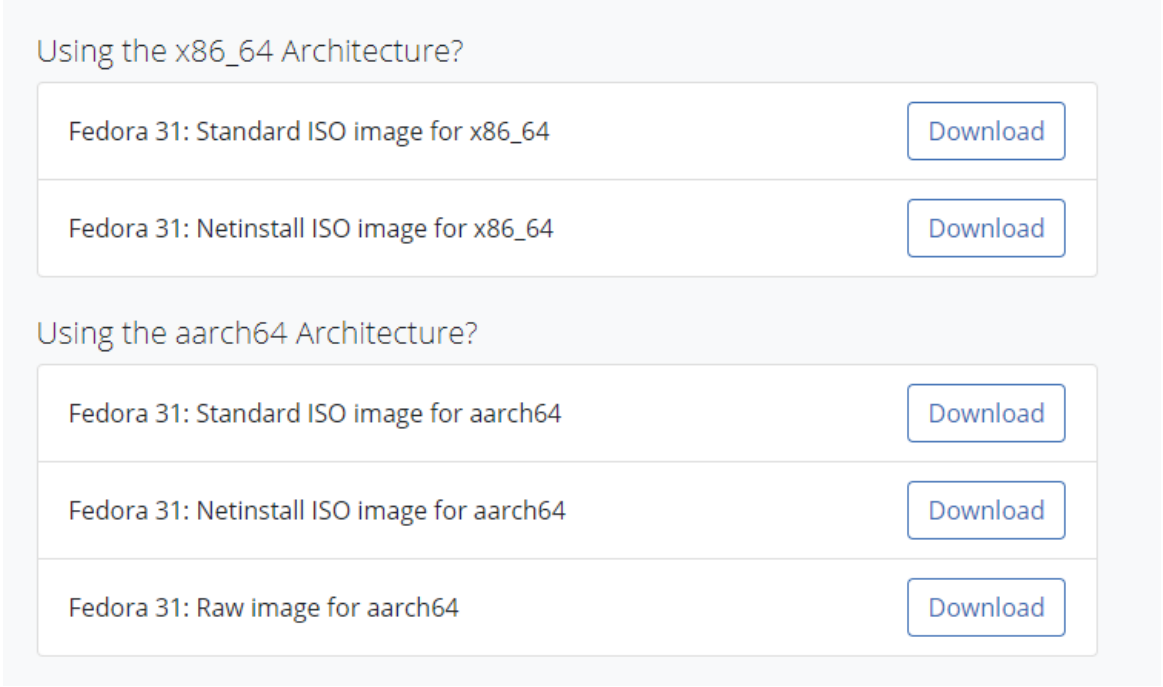
Preparing for Installation

- We will be installing Linux distributions in virtual machines.
  - We'll make sure our virtual machines meet the distribution's recommended requirements.
  - Won't need to worry too much about hardware compatibility.

Installation Media

- Distributions often provide a number of ways to download and install their Linux OS.
- The most common source is on a CD or DVD.
- Distributions provide a ISO image.
  - A disk image that can be burned onto a blank DVD or CD

Installation Media

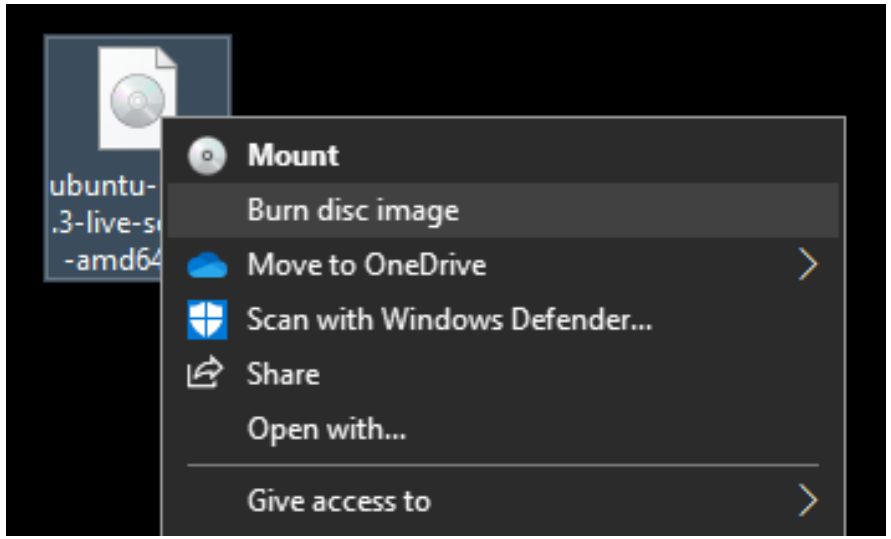


Fedora Installation ISO Download Options

Installation Media

- The ISO file ends with a .iso extension.
- Don’t simply burn the file to a DVD or CD.
  - The file needs to be burned on the DVD or CD as a disk image, not a single file
- We won’t need to burn disk images for this course.
  - We **do** need to know what an ISO file is, though

Installation Media



Burning a disk image is a built-in option for ISO files in Windows 10

Installation Media

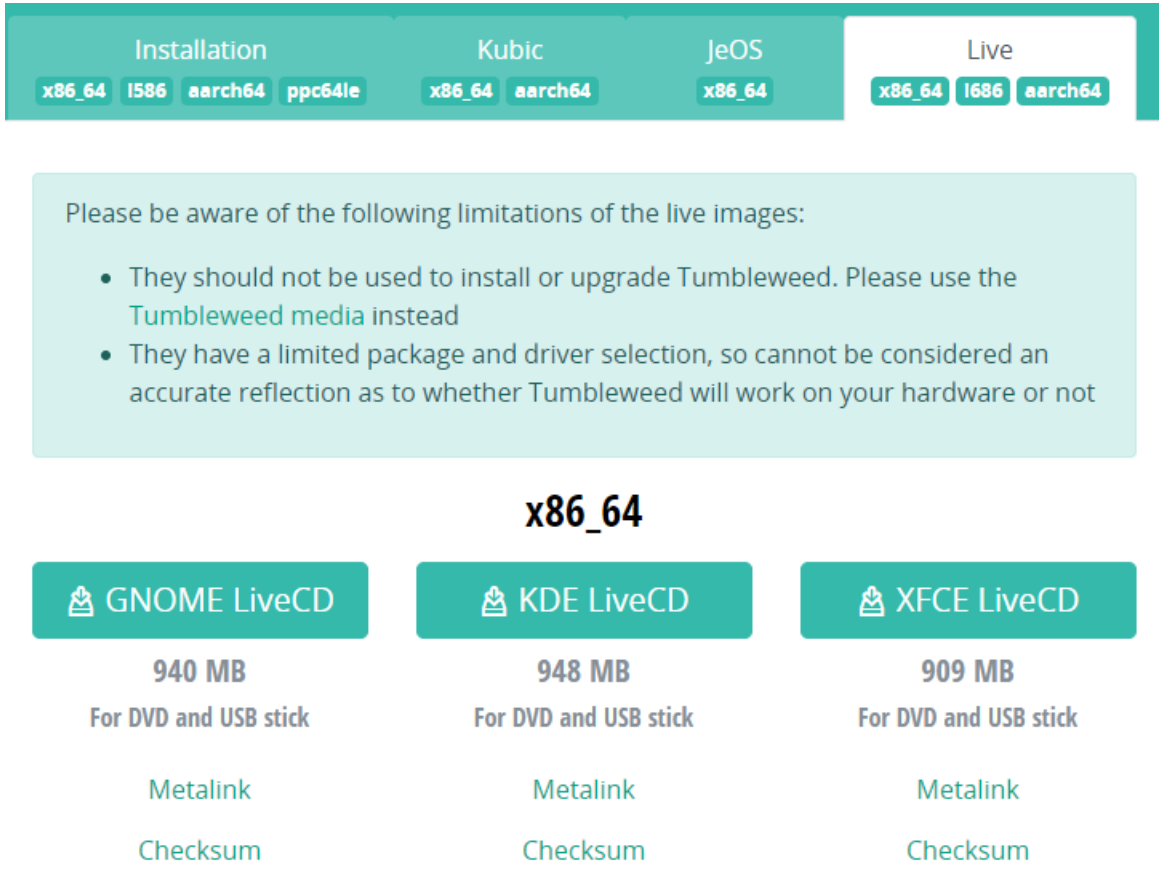
- Distributions may offer ***live media*** images in addition to installation images.

- A ***Live-CD*** (or ***Live-DVD***) is a live media ISO that loads the entire operating system into memory.
  - Burn the Live-DVD ISO to a writable DVD
  - Reboot the computer, booting from the DVD
  - The distribution's Linux OS is loaded

Installation Media

- This allows you to try out the distribution before installing the OS to the computer's hard disk.
- Some Live-CDs/Live-DVDs also offer the option to install the operating system.
- Live media for creating bootable USB flash drives are sometimes offered by a distribution.

Installation Media



openSUSE LiveCD Download Options (CD and USB flash drive)

Performing the Installation

- Regardless of distribution, the installation process typically follows these steps:
  - Start the installation
  - Choose system options
  - Configure the system’s hard disk drive(s)
  - Configure user accounts

Performing the Installation

- Start the Installation
  - When booting to the installation media, you are normally given the option to test the installation media and test the computer's memory
  - Testing the installation media ensures the ISO image was burned correctly to the disk
  - Testing the computer's memory (using a utility like memtest86) ensures the computer's RAM is not defective

Performing the Installation

- Choose system options
  - At this point, you'll select things like localization settings (language, keyboard layout, time zone, etc)
  - The installer may also allow you to select the types of additional/optional software to be installed with the distribution

Performing the Installation

- Configure the system's hard disk drive(s)
  - Each disk drive can be divided into sections called ***partitions.***
    - A disk could contain one partition that takes up the entire drive
    - On Linux systems, it's common for a drive to have multiple partitions
  - Partitions must be formatted to use a ***filesystem***, which is the structure for how data is stored on the disk.

Performing the Installation

- Configure the system's hard disk drive(s)
  - Limits to the number of partitions.
  - 4 primary partitions
  - A primary partition can be an extended partition, which can contain unlimited smaller partitions called logical drives.

  - All information about the disk's partitions is stored in the ***Master Boot Record*** (***MBR***)

Performing the Installation

- Configure the system's hard disk drive(s)
  - The MBR is stored in the first sector of all partitions
  - MBR is found on devices smaller than 2 TB
  - A ***GUID Partition Table*** (***GPT***) is used on devices larger than 2 TB
  - MBR and GPT functionally do the same thing
    - **For GPT, no limit to number of primary partitions**

Performing the Installation

- Configure the system's hard disk drive(s)
  - Linux refers to hard drives in the following ways:
  - PATA Hard Disks : hd**xy**, where x is the device letter and y is the partition number
  - SATA/SCSI/SAS Hard Disks : sd**xy**, where x is the device letter and y is the partition number

  - Device letters start with **a**
  - Partition numbers start with **1**

Performing the Installation

- Configure the system's hard disk drive(s)
  - For example, two PATA drives with 3 partitions on each:
    - hda1 (First PATA hard disk, first partition)
    - hda2 (First PATA hard disk, second partition)
    - hda3 (First PATA hard disk, third partition)
    - hdb1 (Second PATA hard disk, first partition)
    - hdb2 (Second PATA hard disk, second partition)
    - hdb3 (Second PATA hard disk, third partition)

Performing the Installation

- Configure the system's hard disk drive(s)
  - For example, three SATA drives with 2 partitions on each:
    - sda1 (First SATA hard disk, first partition)
    - sda2 (First SATA hard disk, second partition)
    - sdb1 (Second SATA hard disk, first partition)
    - sdb2 (Second SATA hard disk, second partition)
    - sdc1 (Third SATA hard disk, first partition)
    - sdc2 (Third SATA hard disk, second partition)

Performing the Installation

- Configure the system's hard disk drive(s)
  - Linux requires a minimum of two partitions
    - At least one to contain the root directory
    - One for virtual (***swap***) memory
  - Swap memory is used as additional memory space if the RAM is ever full.
    - Data is "swapped" from RAM to the hard disk and back again.
  - Swap partition should be the twice the size of the system's physical memory/RAM.

Performing the Installation

- This is the typical partitioning strategy for a Linux workstation:
  - Partition for the root directory (/)
  - Partition for swap memory
- How we will be partitioning our first Linux installation

Performing the Installation

- Typical partitioning strategy for a Linux server:
  - Partition for the root directory (/)
  - Partition for the boot directory (/boot)
  - Partition for the users' home directories (/home)
  - Partition for log files (/var)
  - Partition for swap memory
- Prevents a crash from a user or log files filling up the entire disk.
  - It may fill the entire partition, but not the entire disk

Performing the Installation

- Configure the system's hard disk drive(s)
  - Partitions can be formatted to a variety of filesystems.
  - Most common Linux filesystems:
    - ext2 – Non-journaling filesystem
    - ext3 – Journaling filesystem
    - ext4 – Improved version of ext3
    - VFAT – Non-journaling filesystem (sometimes used in Linux)
    - XFS – High-performance journaling filesystem

Performing the Installation

- Configure the system's hard disk drive(s)
  - A ***journaling filesystem*** uses a journal to keep track of data written to the partition.
  - In a journaling filesystem, before files are moved or copied each step to perform the operation is written to the journal.
    - The filesystem can retrace its steps in the event of a power outage.
  - Faster data transfer and indexing compared to non-journaling filesystems.

Performing the Installation

- Configure the system's hard disk drive(s)
  - It's a lot to remember
  - We'll keep the partitioning scheme simple for your first install
  - We'll revisit filesystems and partitioning in later lectures

Performing the Installation

- Configure user accounts
  - All Linux systems require authenticated access.
  - Two user accounts must be created
    - The administrator account (called root) which has full access rights to the system
    - A regular user account with limited access
  - **The root account should only be used when performing administrative tasks**
  - Most installers will allow you to set the root password and create a regular user account

Shells and Terminals

- When interacting with any operating system, you indirectly interact with the operating system's kernel
  - Be it through a CLI or GUI
  - Either way, there is some channel to pass input to the kernel for processing

- In Linux, the channel is always through the use of a ***terminal***, which allows a user to log in and communicate with the kernel via a user interface.

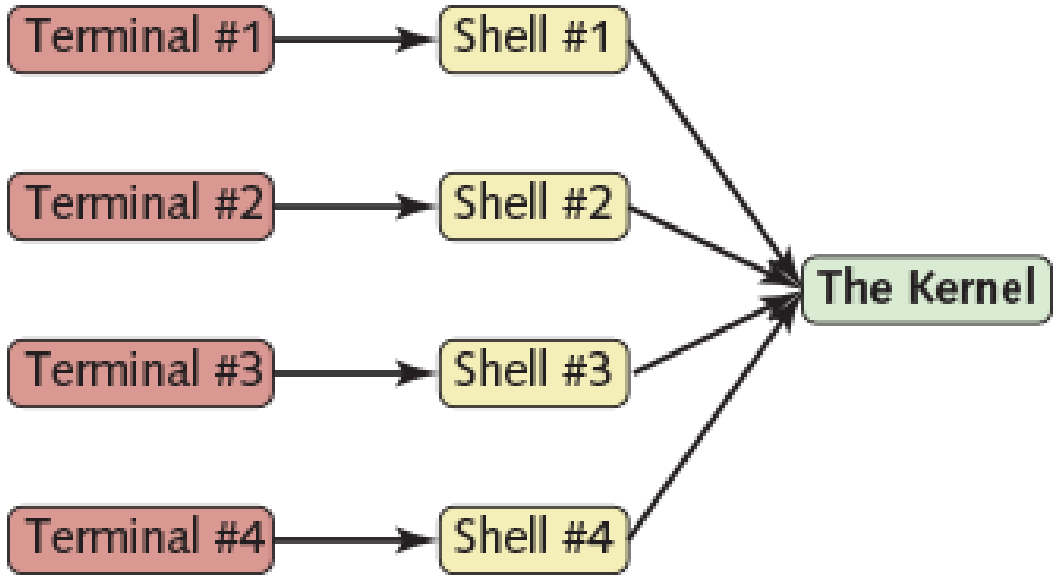
Shells and Terminals

- When you log in to a terminal, the user interface you get called a ***shell*** that accepts your input and passes it to the kernel for processing
  - In its rawest form, a shell uses a CLI
  - A GUI can run in a shell

Shells and Terminals

- Common Shells:
  - Bourne Shell (sh)
    - Original Unix shell
  - **Bourne-Again Shell (bash)**
    - **Default Linux and OSX shell, replacement for the Bourne Shell**
  - C Shell (csh)
    - Designed for C programming/development environmentson Unix systems
  - Korn Shell (ksh)
    - Uses features of csh with bash syntax, common on Unix systems
  - TC Shell (tcsh)
    - Improved version of csh, common on BSD Unix systems

Shells and Terminals



Terminals, shells, and the kernel

The Bash Shell

- Since it is the default shell on Linux systems, we'll exclusively be using the Bash shell in this course

- All shells have a ***syntax*** (proper structure) for issuing commands
  - Similar to syntax rules in a programming language

Bash Syntax

- Bash syntax has three primary components:
  - The command (program) to execute
  - Options to modify how the command behaves
  - Arguments for the command to act upon

```
command [-options] [arguments]
```

- The square brackets simply denote the options/arguments are optional
  - Some commands can run without needing options or arguments

Common Bash Commands

- We'll begin by looking at the command that lists the contents of a directory.

`ls [-options] [arguments]`

- (Lowercase L and lowercase S)
  - Commands, options, and arguments are **case-sensitive**
  - **`LS`** is not the same at **`ls`**

- Does not require options or arguments

Common Bash Commands

- Issuing the **`ls`** command will list the contents of the current directory.

```
[guest@HOS-VM /etc]$: ls
apparmor.d          grub.d          ld.so.conf.d    passwd          shutdown.allow
bash_completion.d   gshadow         lightdm         passwd-         ssl
bashrc              gshadow-        limits          profile         sudoers
dbus-1              gtk-3.0         localtime       profile.d       sudoers.d
default             hostname        login.access    protocols       sudoers.dist
dhcpcd.conf         hosts           login.defs      pulse           sysconfig
dircolors           init.d          man_db.conf     rc.d            syslog.conf
drirc               inittab         mke2fs.conf     resolv.conf     udev
environment         inittab-orig    modprobe.d      rpc             vimrc
fltk                inputrc         mtab            security        wgetrc
fonts               iproute2        nscd.conf       services        X11
fstab               issue           nsswitch.conf   shadow          xattr.conf
group               ld.so.cache     opt             shadow-         xdg
group-              ld.so.conf      pam.d           shells
[guest@HOS-VM /etc]$:
```

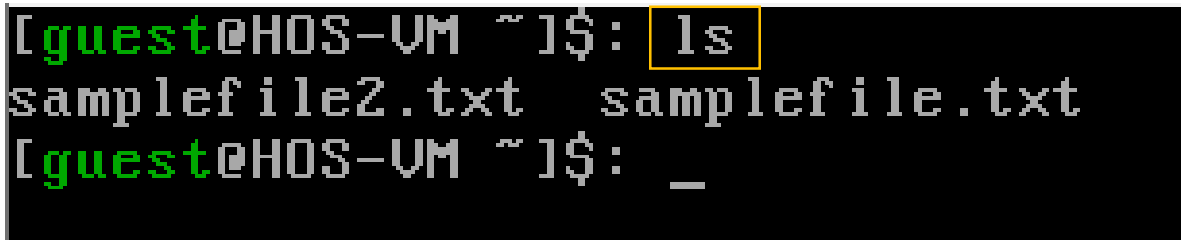
NOTE: Screenshots shown on these slides are from a custom Linux OS, will not match identically to other Linux distributions

Common Bash Commands

- Common **`ls`** options
  - **`-a`** : Lists all (including hidden) files in the listing
  - **`-l`** : Long format listing (includes the file/directory owner, permissions, last access date, etc.)

Common Bash Commands

- In the example below, issuing the **`ls`** command produced two results.



Common Bash Commands

- Here, the **`ls`** command was issued with the **`-a`** option to include hidden files and directories.

```
[guest@HOS-VM ~]$: ls
samplefile2.txt  samplefile.txt
[guest@HOS-VM ~]$: ls -a
.                .bashrc   .fehbg     .local                .Xauthority
..               .config   .fltk      .oracle_jre_usage     .Xdefaults
.bash_history    .dbus     .fluxbox   samplefile2.txt       .xsession-errors
.bash_logout     .dillo    .idlerc    samplefile.txt        .xsession-errors.old
.bash_profile    .dmrc     .links     .viminfo
[guest@HOS-VM ~]$: _
```

Common Bash Commands

- Here, the **`ls`** command was issued with the **`-l`** option to display a long format listing of the directory's contents.

```
[guest@HOS-VM ~]$: ls -l
total 0
-rw-rw-r-- 1 guest guest 0 Nov 24 11:45 samplefile2.txt
-rw-rw-r-- 1 guest guest 0 Nov 24 11:44 samplefile.txt
[guest@HOS-VM ~]$: _
```

Common Bash Commands

- We can use both **`-l`** and **`-a`** options to display a long format listing of the directory's contents, including hidden files and directories.

```
[guest@HOS-VM ~]$: ls -l -a
total 104
drwxr-xr-x 11 guest guest  4096 Nov 24 11:45 .
drwxr-xr-x  3 root  root   4096 Jan  6  2019 ..
-rw-------  1 guest guest    57 Apr 14  2019 .bash_history
-rw-r--r--  1 guest root     87 Jan  6  2019 .bash_logout
-rw-r--r--  1 guest root    472 Jan  6  2019 .bash_profile
-rw-r--r--  1 guest root    453 Jan  6  2019 .bashrc
drwx------  5 guest guest  4096 Jan  6  2019 .config
drwx------  3 guest guest  4096 Jan  6  2019 .dbus
drwx------  2 guest guest  4096 Jan  6  2019 .dillo
-rw-r--r--  1 guest guest    26 Jan  6  2019 .dmrc
-rwxr-xr--  1 guest root     49 Apr 14  2019 .fehbg
drwx------  3 guest guest  4096 Jan  6  2019 .fltk
```

Common Bash Commands

- We can combine the **`-l`** and **`-a`** options into one **`-la`** option

```
[guest@HOS-VM ~]$: ls -la
total 104
drwxr-xr-x 11 guest guest  4096 Nov 24 11:45 .
drwxr-xr-x  3 root  root   4096 Jan  6  2019 ..
-rw-------  1 guest guest    57 Apr 14  2019 .bash_history
-rw-r--r--  1 guest root     87 Jan  6  2019 .bash_logout
-rw-r--r--  1 guest root    472 Jan  6  2019 .bash_profile
-rw-r--r--  1 guest root    453 Jan  6  2019 .bashrc
drwx------  5 guest guest  4096 Jan  6  2019 .config
drwx------  3 guest guest  4096 Jan  6  2019 .dbus
drwx------  2 guest guest  4096 Jan  6  2019 .dillo
-rw-r--r--  1 guest guest    26 Jan  6  2019 .dmrc
-rwxr-xr--  1 guest root     49 Apr 14  2019 .fehbg
```

- Issuing **`ls -al`** would have the same effect

Common Bash Commands

- An argument can be provided to the **`ls`** command to specify the directory to display the contents of

- Without providing an argument, the **`ls`** command displays the current directory.
  - As seen in the previous examples

Common Bash Commands

- Displaying the contents of the /var/log directory:

```
[guest@HOS-VM ~]$: ls /var/log
auth.log   daemon.log   lastlog    sys.log    wtmp
boot.log   faillog      lightdm    tallylog   Xorg.0.log
btmp       kern.log     mail.log   user.log   Xorg.0.log.old
```

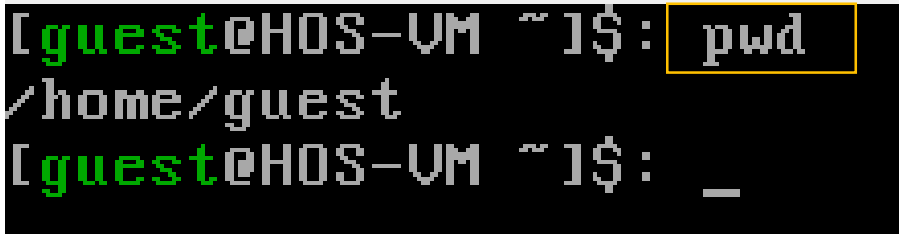
Common Bash Commands

- Displaying a long format listing of the contents of the /var/log directory:

```
[guest@HOS-VM ~]$: ls -l /var/log
total 3676
-rw-r--r-- 1 root    root      47020 Nov 24 11:38 auth.log
-rw-r--r-- 1 root    root      42934 Nov 24 11:37 boot.log
-rw------- 1 root    root       2688 Nov 24 11:38 btmp
-rw-r--r-- 1 root    root      44696 Nov 24 11:37 daemon.log
-rw-r--r-- 1 root    root      24024 Nov 24 11:39 faillog
-rw-r--r-- 1 root    root    1503853 Nov 24 11:37 kern.log
-rw-rw-r-- 1 root    utmp     292292 Nov 24 11:39 lastlog
drwxrwx--- 2 lightdm lightdm    4096 Nov 24 11:37 lightdm
-rw-r--r-- 1 root    root          0 Nov 23  2018 mail.log
-rw-r--r-- 1 root    root    1554361 Nov 24 11:37 sys.log
-rw------- 1 root    root      64064 Jan  6  2019 tallylog
-rw-r--r-- 1 root    root       2472 Apr 14  2019 user.log
-rw-r--r-- 1 root    root     454272 Nov 24 11:39 wtmp
-rw-r--r-- 1 root    root      20097 Nov 24 11:37 Xorg.0.log
-rw-r--r-- 1 root    root      20929 Apr 14  2019 Xorg.0.log.old
```

Common Bash Commands

- The **pwd** command displays the current working directory.
  - (Where we are in the file system)
  - No options or arguments needed



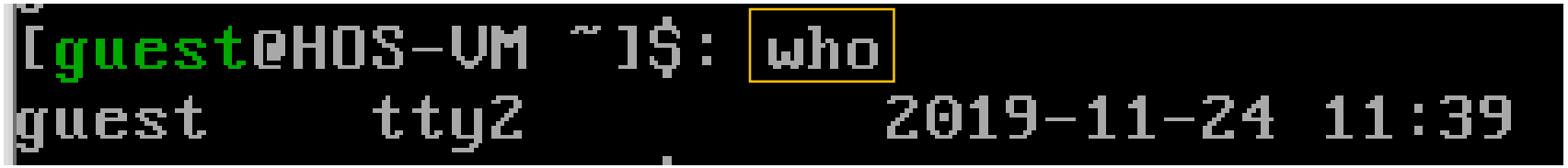
Common Bash Commands

- The **`whoami`** command displays your login name.
  - No options or arguments needed

```
[guest@HOS-VM ~]$: whoami
guest
```

Common Bash Commands

- The **who** command displays currently logged in users.
  - No options or arguments needed



Common Bash Commands

- The **w** command displays currently logged in users and their current tasks.
  - No options or arguments needed

```
[guest@HOS-VM ~]$: w
 12:20:55 up 43 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
guest    tty2      11:39    5.00s  0.29s  0.00s w
```

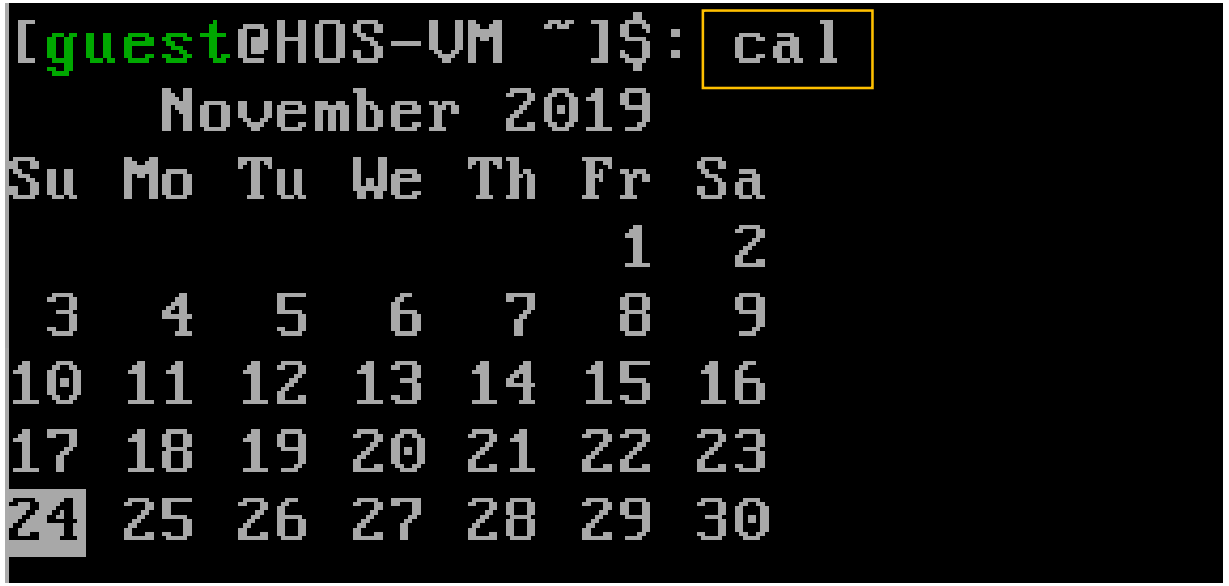
Common Bash Commands

- The **`date`** command displays the current date and time.
  - No options or arguments needed

```
[guest@HOS-VM ~]$: date
Sun Nov 24 12:25:43 EST 2019
```

Common Bash Commands

- The **`cal`** command displays a calendar for the current month.
  - No options or arguments needed



Common Bash Commands

- The **`history`** command displays the user's bash history (Previously entered commands)
  - No options or arguments needed

Common Bash Commands

- Some other common commands (that require no options or arguments)
  - **`clear`** – Clears the terminal screen
  - **`reset`** – Resets your terminal to use default terminal settings
  - **`exit`** – Logs out of the terminal
  - **`logout`** – Same as exit

Common Bash Commands

- The **`su`** command ("substitute user") is used to switch user credentials
  - Syntax: **`su -`** `username`
  - Replace `username` with the actual login account name

- Will prompt for the account password

Common Bash Commands

- A common use is to switch to the root account to perform administrative tasks
  - Issue the command **`su - root`** and enter the root password.
  - Issuing **`su -`** is equivalent to **`su - root`**

```
[guest@HOS-VM ~]$: su - root
Password:
[root@HOS-VM ~]#: _
```

Common Bash Commands

- When logged in as root, you can switch to any user without needing to enter a password.

```
[root@HOS-VM ~]#: su - guest
[guest@HOS-VM ~]$:
```

Common Bash Commands

- To switch back, enter the **`exit`** or **`logout`** command.
  - Both will do the same thing.

```
[root@HOS-VM ~]#: su - guest
[guest@HOS-VM ~]$: exit
logout
[root@HOS-VM ~]#:
```

```
[root@HOS-VM ~]#: logout
[guest@HOS-VM ~]$: _
```

Shell Metacharacters

- Shell ***metacharacters*** are keyboard characters that have special meaning to the shell

- We'll only see a few today, but we will see the others in later lectures

Shell Metacharacters

Metacharacter(s)	Description
$	Shell variable
~	Special home directory variable
#	Shell script comment
&	Background command execution
;	Command termination
< << > >>	Input/Output redirection
|	Command piping
* ? [ ]	Shell wildcards
' " \	Metacharacter quotes
`	Command substitution
( ) { }	Command grouping

Shell Metacharacters

- You should avoid using metacharacters unless you need to make use of their functionality

- To demonstrate some metacharacters, we'll use the **`echo`** command, which simply prints text to the shell
  - Syntax: **`echo [arguments]`**

Shell Metacharacters

- The command **`echo Hello World`** will print the text `Hello World`

```
[guest@HOS-VM ~]$: echo Hello World
Hello World
```

Shell Metacharacters

- Printing the value of the $PATH variable
  - The $PATH variable is the list of directories where the shell looks for executable programs
  - The $PATH variable's list of directories may vary from user to user.

```
[guest@HOS-VM ~]$: echo $PATH
/usr/local/bin:/bin:/usr/bin
[guest@HOS-VM ~]$: echo The user path is $PATH
The user path is /usr/local/bin:/bin:/usr/bin
```

Shell Metacharacters

- To prevent characters from being interpreted as metacharacters, surround the arguments in single quotes

```
[guest@HOS-VM ~]$: echo 'The user path is $PATH'
The user path is $PATH
```

- Using double quotes will interpret metacharacters

```
[guest@HOS-VM ~]$: echo "The user path is $PATH"
The user path is /usr/local/bin:/bin:/usr/bin
```

Shell Metacharacters

- When using double quotes, metacharacters can be ignored with a backslash

```
[guest@HOS-VM ~]$: echo "The user path is \$PATH"
The user path is $PATH
```

Shell Metacharacters

```
[guest@HOS-VM ~]$: echo $PATH is $PATH
/usr/local/bin:/bin:/usr/bin is /usr/local/bin:/bin:/usr/bin
[guest@HOS-VM ~]$: echo '$PATH is $PATH'
$PATH is $PATH
[guest@HOS-VM ~]$: echo "$PATH is $PATH"
/usr/local/bin:/bin:/usr/bin is /usr/local/bin:/bin:/usr/bin
[guest@HOS-VM ~]$: echo "\$PATH is $PATH"
$PATH is /usr/local/bin:/bin:/usr/bin
```

Shell Metacharacters

- Back quotes (also called backticks) are metacharacters for command substitution

```
[guest@HOS-VM ~]$: echo Today is date
Today is date
[guest@HOS-VM ~]$: echo Today is `date`
Today is Sun Nov 24 15:28:59 EST 2019
[guest@HOS-VM ~]$: echo Today is `date +%A`
Today is Sunday
```

  - We haven’t seen options for formatting the date command’s output like the last command in the above example
  - This is only to show the use of backticks to substitute the output of the date command as part of the arguments to the echo command

Getting Command Help

- Linux has a lot of commands to remember, and it’s difficult to memorize all of the different options each command has.

- A few ways to get help within the shell.

Getting Command Help

- Most commands and programs come with manuals, or ***man pages***.

- Man pages are documents that explain the usage of a command or program.

Getting Command Help

- The syntax for displaying the man page for a command is

```
man [options] [argument]
```

- For displaying the man page for the ls command:

```
man ls
```

Getting Command Help

man ls

Getting Command Help

- Typical sections of a man page
  - NAME – Name of the command or program
  - SYNOPSIS – Basic syntax of the command
  - DESCRIPTION – List of all the options and what they do
  - AUTHOR – Person(s) who wrote the command/program
  - REPORTING BUGS – How to report issues with the command/program
  - COPYRIGHT – License information
  - SEE ALSO – Additional resources

Getting Command Help

- Navigating man pages:
  - **Home** : Move to beginning of man page
  - **End** : Move to end of man page
  - **Page Up** : Scroll up
  - **Page Down** : Scroll down
  - **/** : Begins a search
  - **n** : Move to next occurrence of search term
  - **p** : Move to previous occurrence of search term
  - **q** : Exit

Getting Command Help

- Frequently used man options:
  - **`-a`** : All entries matching the query
  - **`-D`** : Debugging information
  - **`-f`** : Short description of the command
  - **`-h`** : Help options for the man command
  - **`-k`** : Lists all man pages containing a keyword
  - **`-K`** : Searches for a string on all man pages
  - **`-t`** : Formats the man page to enable printing

Getting Command Help

- You'll notice at the top the ls command is listed as LS(1)

- The 1 denotes the section number of the man pages.

- The next slide shows the category/section numbers of man pages

Getting Command Help

- Man page sections types:

Manual page section	Description
1	Commands that any user can execute
2	Linux system calls
3	Library routines
4	Special device files
5	File formats
6	Games
7	Miscellaneous
8	Commands that only the root user can execute
9	Linux kernel routines
n	New commands not categorized yet

Getting Command Help

- To search the man pages, use the -k option

```
man -k usb
```

- This command will list all commands that have the word usb in their names or description.
  - From there, you can use the man command on the command you are interested in.

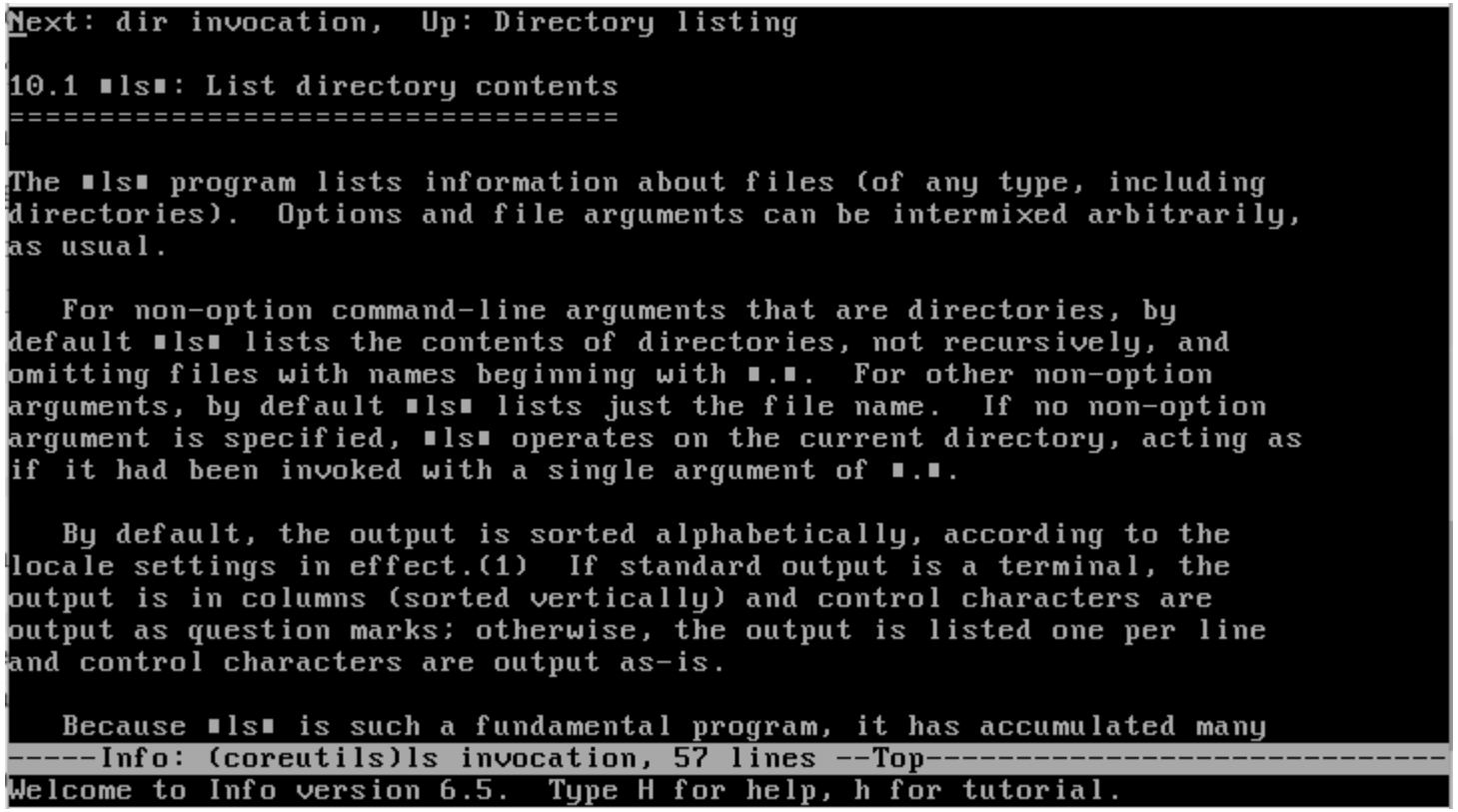
Getting Command Help

- Info pages were intended to replace man pages.

- The **`info`** command gives an easy to read description of a command.
  - Syntax: **`info [options] [argument]`**

Getting Command Help

`info ls`



Getting Command Help

- Use the up and down arrow keys (or page up and page down keys) to navigate through the info page.

- Press q to exit.

Getting Command Help

- Some commands may not have man pages or info pages

- In this case, use the **`help`** command
  - Syntax: **`help [options] [argument]`**
  - Alternatively: **`command --help`** or **`command -h`**

- You can also use the **`whatis`** command for brief descriptions of a command.
  - Example: **`whatis echo`**

Getting Command Help

help echo

```
[guest@HOS-VM ~]$: help echo
echo: echo [-neE] [arg ...]
    Write arguments to the standard output.

    Display the ARGs, separated by a single space character and followed by a
    newline, on the standard output.

    Options:
      -n        do not append a newline
      -e        enable interpretation of the following backslash escapes
      -E        explicitly suppress interpretation of backslash escapes

    `echo' interprets the following backslash-escaped characters:
      \a        alert (bell)
      \b        backspace
      \c        suppress further output
      \e        escape character
      \E        escape character
      \f        form feed
      \n        new line
```

Shell Tricks and Tips

- To scroll up and down in the terminal, use the following key combinations:
  - **Shift+Page Up** to scroll up
  - **Shift+Page Down** to scroll down

- Use the **Up** and **Down** arrow keys in the shell to cycle through your bash history.
  - Quick way to reuse a previously entered command

Shell Tricks and Tips

- Press the **Ctrl+C** key combination to cancel a running command.

- Use the **Tab** key to autocomplete command and file/directory names.
  - Entering **ech** and pressing Tab *should* autocomplete to **echo**
    - If there is more than one possibility, nothing will autocomplete
    - Press Tab again to get all possibilities

Shell Tricks and Tips

- For example, entering **ch** and pressing Tab will produce no autocompletion.
  - **ch** is not specific enough; there are numerous possibilities
  - Pressing Tab again will list the possibilities

```
[guest@HOS-VM ~]$: ch
chacl     chattr    checkmk   chfn      chmem     chown     chsh
chage     chcon     chem      chgrp     chmod     chrt      chvt
[guest@HOS-VM ~]$: ch_
```

Shutting Down

- Forgetting to shut the system down properly may result in damaged user and system files.

- Must allow the operating system to shut itself down.

- The **`shutdown`** command is used to power off, halt, or reboot the system
  - Syntax: **`shutdown [options] [arguments]`**

Shutting Down

- Common shutdown options:
  - **-P** : Power off (Shutdown and power off the computer)
  - **-H** : Halt (Shutdown but do not power off the computer)
  - **-r** : Reboot
  - **-c** : Cancels a scheduled shut down

- To schedule a shutdown, include a **+N** argument after **-P** , **-H**, or **-r**
  - Replace the N with a number
  - (Number of minutes until shutdown)
  - Use **now** for immediate shutdown/reboot

Shutting Down

- Shuts down immediately:

  `shutdown -P now`

- Reboots immediately:

  `shutdown -r now`

- Halts immediately:

  `shutdown -H now`

Shutting Down

- Shuts down immediately (alternative):
  `poweroff`
- Reboots immediately (alternative):
  `reboot`
- Halts immediately (alternative):
  `halt`

Shutting Down

- Shuts down in 10 minutes:

  `shutdown -P +10`

- Reboots in 10 minutes:

  `shutdown -r +10`

- Halts in 10 minutes:

  `shutdown -H +10`

Shutting Down

- You can warn users of an impending shutdown/reboot using the warn all (**`wall`**) command.

- This will send a message to all users on the system.

```
wall System shutting down in 10 minutes. Save your work and log out.
```

Shutting Down

- Alternatively, you can include a message with the shutdown command
  - The message comes after the time

shutdown –r +10 Save your work an log out.

```
[root@HOS-VM ~]#: shutdown -r +10 Save your work and log out.

Broadcast message from root@HOS-VM (tty2) (Sun Nov 24 16:19:35 2019):

Save your work and log out.
The system is going DOWN for reboot in 10 minutes!
```