<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adduser</td>
<td>Add a user to the system</td>
</tr>
<tr>
<td>addgroup</td>
<td>Add a group to the system</td>
</tr>
<tr>
<td>alias</td>
<td>Create an alias</td>
</tr>
<tr>
<td>apropos</td>
<td>Search Help manual pages (man -k)</td>
</tr>
<tr>
<td>apt-get</td>
<td>Search for and install software packages</td>
</tr>
<tr>
<td>aptitude</td>
<td>Search for and install software packages</td>
</tr>
<tr>
<td>aspell</td>
<td>Spell Checker</td>
</tr>
<tr>
<td>awk</td>
<td>Find and Replace text, database sort/val</td>
</tr>
<tr>
<td>basename</td>
<td>Strip directory and suffix from filename</td>
</tr>
<tr>
<td>bash</td>
<td>GNU Bourne-Again SHell</td>
</tr>
<tr>
<td>bc</td>
<td>Arbitrary precision calculator language</td>
</tr>
<tr>
<td>bg</td>
<td>Send to background</td>
</tr>
<tr>
<td>break</td>
<td>Exit from a loop</td>
</tr>
<tr>
<td>builtin</td>
<td>Run a shell builtin</td>
</tr>
<tr>
<td>gzip</td>
<td>Compress or decompress named file(s)</td>
</tr>
</tbody>
</table>

**Linux Commands**

This material is based on work supported by the National Science Foundation under Grant No. 0802551

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Linux can be used with a graphical user interface (GUI) that uses menu, buttons, icons, and labels to represent tasks, actions, or choices. Selecting a GUI element performs the selected action. The GUI is a very powerful tool that makes computing easier—especially for new users. However, a GUI interface requires many steps and mouse clicks to perform tasks and not all GUI interfaces are intuitive.

A faster method for interacting with a Linux system is the use of a command line interface in which a user instructs the computer to perform a variety of tasks by simply typing commands on a command line or in a terminal window.

In this lesson, you will be using the command line interface to execute a series of commands. You will explore useful Linux commands, scripts, and programming concepts that every budding administrator must know. Learning Linux commands is important because many complex task require the command line, or are easier to perform in that environment. Linux systems require command line use for some tasks, and all good Linux administrators must be comfortable using it.
Student Expectations

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives.

Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.
Objective

Given a Linux command line interface, a student will be able to execute the shell command and use the appropriate syntax to perform basic command line shell functions as per industry standards.
During this lesson, you will explore:

- Programming terms
- Bash shell
- Command line tasks
- Syntax & Characters
- Text Editors
Choosing a Text Editor

Text editors are simple computer programs that allow you to create or edit text files. Text editors are often used for programming purposes or to write and edit computer code. Text editors differ from word processors in several ways, but one prominent difference is that text editors process only the actual characters in a file while word processors add special formatting and other hidden code.

To edit scripts and computer code, you need a text editor. Linux offers more than 15 different editors, so you are free to use the one that offers the most value to you. Some of the more prominent editors include:

- VI
- VIM
- EMACS
- GEDIT
- CREAM
- JEDIT
- NANO

- SCITE
- LEAFPAD
- GEANY
- BLUEFISH
- XEMACS
- KWRITE
- SCRIBES
- LYX

Select PLAY below to view videos on text editors.

View Video VideoLesson2TextEditors(C2L2S14V1).mp4

Text editor overview

View Video VideoLesson2TextEditors(C2L2S14V2).mp4

Description of text editors
To set the default editor on your Ubuntu system, use the `update-alternatives` command. Open a terminal window and type the following command:

```
sudo update-alternatives --config editor
```

Enter your `sudo` password to continue . . .

The resulting screen will provide you with a list of choices for your editor. You may keep the default editor—indicated by an asterisk (*) or you may select your preferred editor from the list of options that generally include the following:

<table>
<thead>
<tr>
<th>Selection</th>
<th>Path</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/bin/nano</td>
<td>40</td>
<td>Auto</td>
</tr>
<tr>
<td>1</td>
<td>/bin/ed</td>
<td>-100</td>
<td>Manual</td>
</tr>
<tr>
<td>2</td>
<td>/bin/nano</td>
<td>40</td>
<td>Manual</td>
</tr>
<tr>
<td>3</td>
<td>/usr/bin/vim.tiny</td>
<td>10</td>
<td>Manual</td>
</tr>
</tbody>
</table>

Note:
You can test your editor choice by typing `crontab -e` at the command line.
Recommended Editors

Of the many available text editors, four are recommended:

**Vi** is a display oriented, interactive text editor which allows a user to create, modify, and store files on the computer via a terminal.

**Vim** is a highly configurable text editor built to enable efficient text editing. It is an improved version of the Vi editor distributed with most UNIX systems. Vim is short for 'Vi Improved.'

**GNU Emacs** is an extensible, customizable text editor—and more. It has several features including content-sensitive editing modes and complete built-in documentation.

**Cream** is a set of scripts and add-on tools that can be added to Vim. Cream does not change the visual look of Vim, but Cream expands the functionality of Vim and makes it easier to use.

---

**Required Reading: Vi**
- Calling Vi
- Vi and Vim Editors
- Evil text editor?
- The Vi Editor
- Using Vi

**Required Reading: Vim**
- Vim home
- Vim Installation
- Vim color editor

**Required Reading: Emacs**
- Emacs home
- Emacs popularity
- Emacs guide
- Emacs commands
- More Emacs commands

**Required Reading: Cream**
- Cream for Vim
- Adding Cream to Vim
- Cream for Vim
- Cream
After selecting your editor, it is time to explore commands.

Select **PLAY** below to view two videos on command line tasks:

- **Useful commands**
  - View Video
    - VideoLesson2UsefulCommands(C2L2S17V1).mp4

- **Network commands**
  - View Video
    - VideoLesson2NetworkCommands(C2L2S17V2).mp4
Once you access a terminal window in Linux, there are hundreds of commands at your disposal. The list below introduces a few of these commands:

- **ls** - list contents of a directory
- **sudo** - "super-user do" (grants administrator rights)
- **cd** - change directory
- **aptitude** - APT package management system (update, install, remove, search)
- **clear** - clear screen
- **chmod** - change file access permissions
- **chown** - change file and group ownership
- **cp** - copy
- **mv** - move
- **rm** - remove
- **cat** - concatenate files (dump to screen)
- **nano** - basic text editor
- **vi** - advanced text editor
- **fdisk** - partition table manipulator
- **df** - disk free (remaining / used disk space)
- **users** - users currently logged in
- **useradd** - add a user
- **usermod** - modify existing user
- **uname** - show system data (try `uname -a`)
- **mount** - mount a file system, cd or removable drive
- **umount** - un-mount a file system, cd or removable drive
- **top** - show current running processes
- **touch** - create new, empty, file
- **reboot** - reboot your system
- **shutdown** - shutdown your system
- **passwd** - change user password
- **ping** - ping a network device or location (ping google.com)
- **more** - show output one screen at a time
- **exit** - logout of the terminal
- **eject** - eject a cdrom or removable device

**Additional Links**
- [Commands in PDF format]
The `grep` command searches the given file for lines containing a match to the given strings or words. By default, `grep` prints the matching lines.

Use `grep` to search for lines of text that match one or many regular expressions, and outputs only the matching lines.

A regular grep search would have this format:

```
% grep "\<[A-Za-z].*" file
```

### Grep Regular Expression

<table>
<thead>
<tr>
<th></th>
<th>Denotes the beginning of a line</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Denotes the end of a line</td>
</tr>
<tr>
<td>.</td>
<td>Matches any one characters</td>
</tr>
<tr>
<td>*</td>
<td>Matches 0 or more of the previous characters</td>
</tr>
<tr>
<td>.*</td>
<td>Matches any number or type of characters</td>
</tr>
<tr>
<td>[]</td>
<td>Matches on character for the one listed in the the Square brackets</td>
</tr>
<tr>
<td>[^ ]</td>
<td>Does not match any characters listed</td>
</tr>
<tr>
<td>&lt;, &gt;</td>
<td>Denotes the beginning and end (respectively) of a word</td>
</tr>
</tbody>
</table>
BASH Shell is a software tool that interprets commands for the GNU operating system. It can run on most Unix-based system and is the default shell for the Linux kernel and Mac OS X. Bash shell allows a computer user to interface with the Linux kernel and other areas of the operating system.

**View Video**
VideoLesson2Bash1(C2L2S20V1).mp4

**View Video**
VideoLesson2Bash2(C2L2S20V2).mp4

**Required Reading**
- Bash shell programming
- Bash command list
- Advanced Bash
- Bash Unix command
Script
Normally a script is a text file that is a listing of single file commands. A script can be one line or thousands of lines of commands.

Variable
A variable is a predefined structure that represents a number, phrase, term or other relevant data.

Expression
In programming, expression refers to a line of source code that returns a value when executed. An expression can comprise any combination of variables, values, operators, and functions.

Process
A process is a program that is being executed and can be identified by its unique PID (process identification number). In other words, it is a program in execution!

- Linux process
- Controlling system processes
- Show running processes
- Learning Linux process

Handbook on Shell Scripting
Linux Shell Scripting
Teach Yourself Scripting
Linux shell for beginners
Shell Pipe
The term pipe in shell programming refers to the process of directing the output of one script or program to the input of another script or program.

Alias
Simply stated, an alias is just another name for something. For example, a young dog could have an alias of puppy.

Responsibilities of the Shell
- Reading inputs and parsing the command line
- Evaluating special characters
- Setting up pipes and background processing
- Handling various signals
- Setting programs up to be executed

Required Reading: Shell Pipe
- Pipes & Redirectors
- Intro to Bash Programming
- Bash Tutorial
- Named Pipes
- Bash Shell

Required Reading: Alias
- Linux alias
- Using aliases

Required Reading: Shell
- Shell Script
- Shell scripting tutorial
Shell scripting uses special characters that have specific meanings:

<table>
<thead>
<tr>
<th>CHARACTERS</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>\” ‘ ”</td>
<td>(backslash, double quotes, single quote) are used for quoting</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Greater than and less than signs are used for input/output direction</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>;</td>
<td>The semicolon separates multiple commands written on a single line</td>
</tr>
<tr>
<td>[Space] and [Tab]</td>
<td>A single space and a tabbed space are use to separate the command words</td>
</tr>
<tr>
<td>[Newline]</td>
<td>A new line completes a command or set of commands</td>
</tr>
<tr>
<td>( )</td>
<td>Parenthesis enclose command(s) to be launched in a separate shell (subshell).</td>
</tr>
<tr>
<td>{ }</td>
<td>Curly brackets enclose a group of commands to be launched by the current shell. E.g. { dir }. Be sure to include the space before and after the command within the brackets.</td>
</tr>
<tr>
<td>&amp;</td>
<td>Ampersand causes the preceding command to execute in the background (i.e., asynchronously, as its own separate process) so that the next command does not wait for its completion</td>
</tr>
</tbody>
</table>
Shell scripting uses special characters that have specific meanings:

<table>
<thead>
<tr>
<th>CHARACTERS</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>When a filename is expected, the asterisk matches any filename except those starting with a period (or any part of a filename, except the initial period)</td>
</tr>
<tr>
<td>?</td>
<td>When a filename is expected, the question sign matches any single character</td>
</tr>
<tr>
<td>[ ]</td>
<td>When a filename is expected, the square brackets match any single character enclosed inside the pair of [ ]</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Ampersands serve as AND characters connecting two commands. For example, <code>command1 &amp;&amp; command2</code> will execute <code>command2</code> only if <code>command1</code> exits with the exit status 0 (no error). Additionally, <code>cat file1 &amp;&amp; cat file2</code> will display <code>file2</code> only if displaying <code>file1</code> succeeded.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>The equal sign assigns a value to a variable. Critical: Use no spaces before or after the “=” when assigning a value!</td>
</tr>
</tbody>
</table>
Review the following links for special characters used in shell scripting:

- [Special characters](#)
- [Escape characters](#)
- [Advanced guide](#)
- [Metacharacters](#)
- [Linux cookbook](#)
A Linux Administrator can create powerful scripts to accomplish almost any administrative task. The administrator that is “fluent” in using shell scripting will have the competitive advantage over those who do not!

Shell scripting is the use of shell commands stored in a plain text file that can be executed. Shell scripting is a very powerful tool that may be used to automate many complex computing tasks.

To write commands, programmers may use text editors, which are powerful programs that are compatible with plain text characters and will not add invisible coding or formatting to text. Linux users may choose from a wide variety of text editors including Vi, Vim, and Emacs.

Shell scripting assigns special meaning to certain characters including the asterisk (*), question sign (?), backslash (\) and ampersand (&).