ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK2: LESSON 1

UNIX & LINUX FILE MANAGEMENT CONCEPTS
MANAGING DIRECTORIES

LESSON 1 TOPICS

Unix / Linux File Management Concepts

- Purpose of Directories
- Directory Pathnames / Tree Diagrams
- Filename Rules

Managing Directories

- Creating / Viewing Contents of / Manipulating / Removing Directories:
  `mkdir -p`, `rmdir`, `rm -r -i`, `ls -l -d -R` , `tree`, `cp -R`, `mv`
- Demonstration

Homework

- Perform **Tutorial 2: Unix / Linux File Management (Investigation 1)**
  Perform LINUX PRACTICE QUESTIONS (1 – 8)

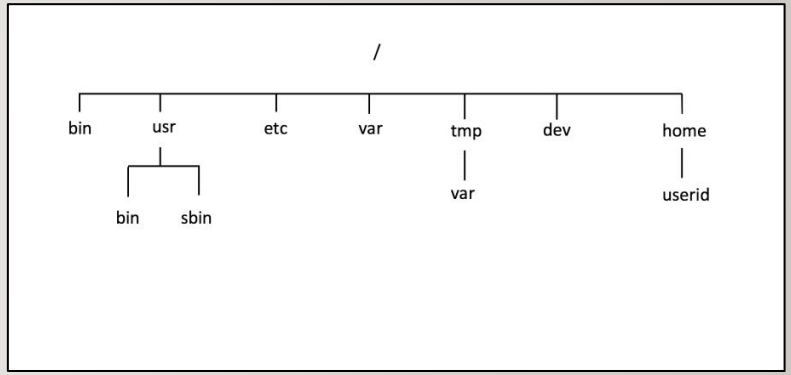
LINUX FILE MANAGEMENT CONCEPTS

Purpose of Unix / Linux Directories

To better **organize** files (eg. text, images, documents, spreadsheets, programs) within your Matrix account, they should be stored in **directories**.

To further organize <u>many</u> files, directories may contain **sub-directories**.

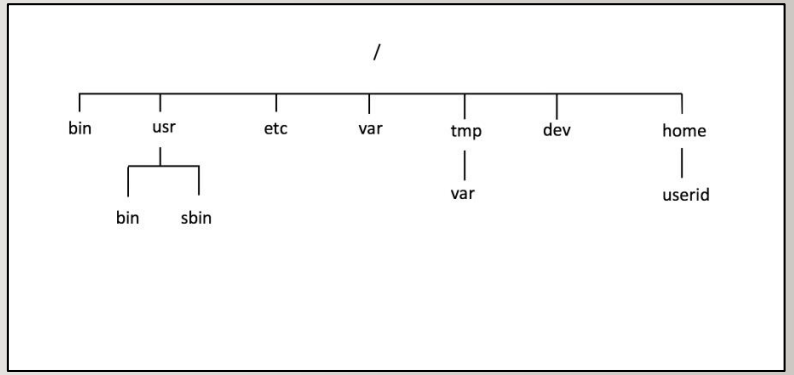
Learning how to issue Linux commands for **navigating** and **manipulating** directory and files within the the Linux filesystem are **essential skills** for Linux users and Linux system administrators (i.e. *sysadmins*).



LINUX FILE MANAGEMENT CONCEPTS

Purpose of Unix / Linux Directories

The Unix/Linux file system is **hierarchical**, like other operating systems such as **Windows**, **Mac OSX**, etc. In Unix / Linux (as opposed to MS Windows), there are no drive letters such as **C:**, or **D:**

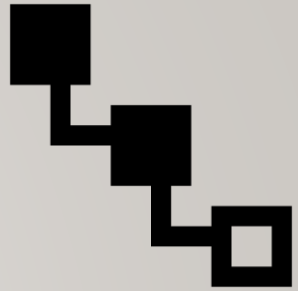


All files and directories appear under a single ancestor directory called the "**root directory**".

In the Linux (Unix) OS, the "**root directory**" / is the starting directory, and other "*child directories*", "**grandchild directories**", etc. can be created as required. The hierarchical structure resembles an "*upside-down tree*". There is actually a command called **tree** that displays a "**directory tree diagram**"!

```
tree linux
linux
└── uli101
    ├── notes
    ├── practice
    └── samples
```

LINUX FILE MANAGEMENT CONCEPTS



Directory Pathnames

*A **pathname** is used to specify the **location** of a file within the file system.*

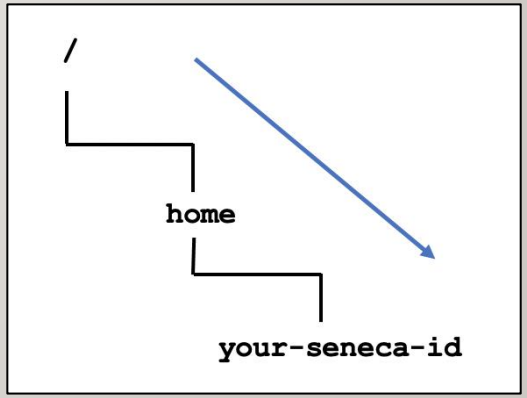
*A pathname **points** to a file system location by **following the directory tree hierarchy** expressed in a string of characters in which path components, separated by a delimiting character, represent each directory.*

*The **delimiting character** is most commonly the slash character ("**/**").*

Reference: https://en.wikipedia.org/wiki/Path_(computing)

Example:

```
/home/your-seneca-id
```

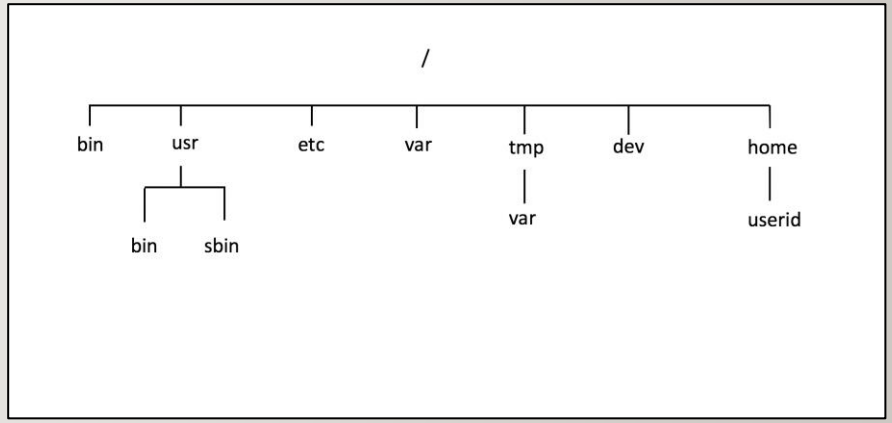


LINUX FILE MANAGEMENT CONCEPTS

Common Unix / Linux Directories

Below are several common Unix / Linux Directories and their purpose:

Directory Pathame	Purpose
/	Root directory (ancestor to all directories)
/home	Used to store users' home directories
/home/username	A **specific** User's Home Directory
/bin , /usr/bin	Common system binaries (commands)
/usr/sbin	Common utilities for system administration
/etc	System administration files (eg. passwd)
/var	Dynamic files (log and mail files)
/tmp , /var/tmp	Temporary files for programs
/dev	Device driver files (terminals, printers, etc.)



LINUX FILE MANAGEMENT CONCEPTS

Directory File Naming Rules

Before learning to **create** directories, it is important to understand what represents an appropriate directory filename. Here are some **rules**:

Unix / Linux File Naming Rules
✓ Unix/Linux characters are **case sensitive** (e.g. always use lowercase letters)
✓ Adopt a **consistent directory naming scheme** (this will help you to better navigate within your directory structure)
✓ Make your directory names **meaningful** (short but descriptive)
✓ **Avoid using spaces** for directory names (consider **periods**, **hyphens**, and **underscores** instead)
✓ **Avoid non-alphanumeric characters**, as they may have a special meaning to the system that will make your work more difficult when changing to directories, etc.

MANAGING DIRECTORIES

Managing Directories

Below are some common Unix / Linux commands to manage Directories:

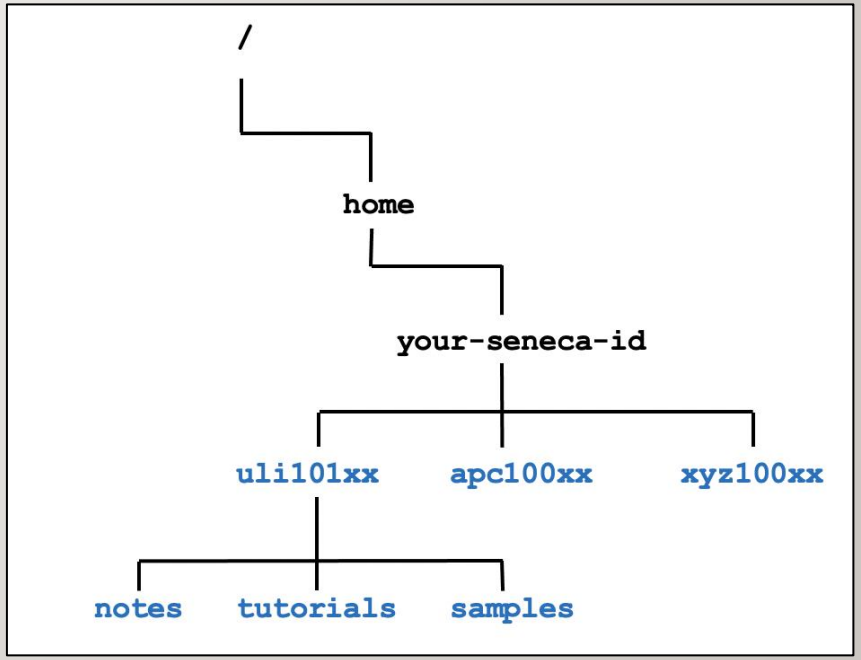
Directory Pathame	Purpose
`mkdir -p`	Creates a directory. The **-p** option creates parent directories then directory pathnames specified.
`rmdir`	Removes empty directories.
`rm -r -i`	Removes files, but when used with **-r** option, will remove non-empty directories and their contents. The **-i** option is used to prompt user to confirm deletion of directory contents
`ls -l -d –R , tree`	List directory contents. Useful to verify if directory was created. The **-d** option lists the directory itself (not contents) The **-R** option displays directories and subdirectory contents. The tree command displays diagram of directory structure.
`cp -R`	Copies directory and its contents (recursive) to a different directory
`mv`	Moves directory and its contents to a different directory

MANAGING DIRECTORIES

Managing Directories Demonstration

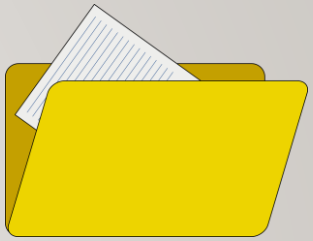
Your instructor will demonstrate how to manage directories by issuing Unix / Linux commands:

- Create directory structure as shown in diagram to the right
- View / Verify created directories
- Copy directories
- Move directories
- Remove empty directories
- Remove non-empty directories



MANAGING DIRECTORIES

Determine Type of File



When issuing the **`ls`** command to view directory contents of viewing a directory, the **`-l`** option can be used to help determine it file type.

```
ls -l
drwxr-xr-x 2 murray.saul users 6 Jan 11 09:42 documents
-rw-r--r-- 1 murray.saul users 0 Jan 11 09:42 file.txt
crw-rw-rw- 1 root root 1, 3 Dec  2 07:25 /dev/null
```

The first character on the **left** of the output indicates the type of file:

`d:` directory file

`-:` regular file

`b` or **`c:`** device file

MANAGING DIRECTORIES

Hidden Files

A file is hidden if its name starts with a period "."
This can hide both regular files and directory files.

Why make files hidden?

- To clean up directories
- To hide backups
- To protect important files from accidental deletion

If you issued the `ls` command without arguments, hidden files do NOT appear.

The `ls` command with the `-a` option will show all files including hidden and non-hidden. Current and Parent directories (. and ..) are displayed

The `ls` command with the `-A` option will show all files including hidden and non-hidden. Current and Parent directories (. and ..) are NOT displayed

HANDS-ON TIME / HOMEWORK

Getting Practice

To get practice to help perform **online assignment #1**, perform the online tutorial **Tutorial2: Unix / Linux File Management** (**ctrl-click** to open link):

- INVESTIGATION 1: MANAGING DIRECTORIES
- LINUX PRACTICE QUESTIONS (Questions 1 – 8)

ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK2: LESSON 2

MANAGING TEXT FILES:
USING TEXT EDITORS TO CREATE & EDIT A TEXT FILE
MANAGING TEXT FILE CONTENT

LESSON 2 TOPICS

Creating Text Files

- Purpose of a Text Editor
- Using the **`nano`** Text Editor / Demonstration
- Using the **`vi`** Text Editor / Demonstration

Managing / Manipulating Text Files

- Linux Commands: **`touch`**, **`cat`**, **`more/less`**, **`cp`**, **`mv`**, **`rm`**, **`diff`**, **`file`**, **`find`**
- Demonstration

Homework

- Perform **Tutorial 2: Unix / Linux File Management (Investigation 2)**
  Perform LINUX PRACTICE QUESTIONS (9 – 16)
- Continue **Assignment #1** (Sections: **1** and **2**)

CREATING TEXT FILES

Text Editors

A **Text Editor** allows users to **create**, **modify** and **save** editing changes of text files.

Although **programming students** can use **graphical IDE's** to code and compile programs, students can **create source code** using a text editor and **compile their source code** in their Matrix account to generate **executable programs**.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <arpa/inet.h>

void serveur1(portServ ports)
{
    int sockServ1, sockServ2, sockClient;
    struct sockaddr_in monAddr, addrClient, addrServ2;
    socklen_t lenAddrClient;

    if ((sockServ1 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }
    if ((sockServ2 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }

    bzero(&monAddr, sizeof(monAddr));
    monAddr.sin_family = AF_INET;
    monAddr.sin_port = htons(ports.port1);
    monAddr.sin_addr.s_addr = INADDR_ANY;
    bzero(&addrServ2, sizeof(addrServ2));
```

CREATING TEXT FILES

Text Editors

Networking and Tech Support students use a text editor to **edit configuration files**.

Throughout their program, students will become familiar with the process of **installing**, **configuring**, and **running** network services on their Linux servers.

Text editors are an important tools to help setup but also "**tweak**" or make **periodic changes in networking services configuration**.

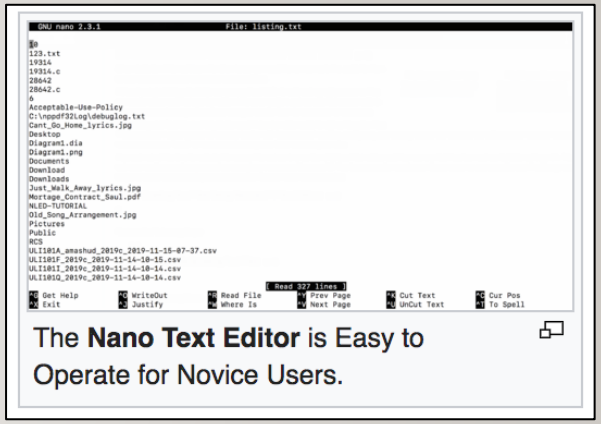
```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
PS1="\e[0;36m[\u @ \h \W] \e[m "
PS2="/Finish command/ "
export LC_ALL=C
export LC_COLLATE=C
who
echo
mesg n
PATH=$PATH:~/scripts

umask 077
```

CREATING TEXT FILES

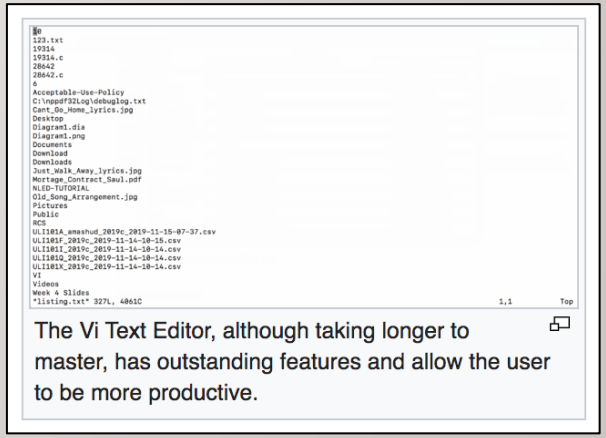
Text Editors

Regardless of the IT stream that they are in, it is useful for students to **expose themselves to different text editors and then use one that they feel most comfortable working with**.

The two most readily-available command line text editors in Linux are **`nano`** and **`vi`**.

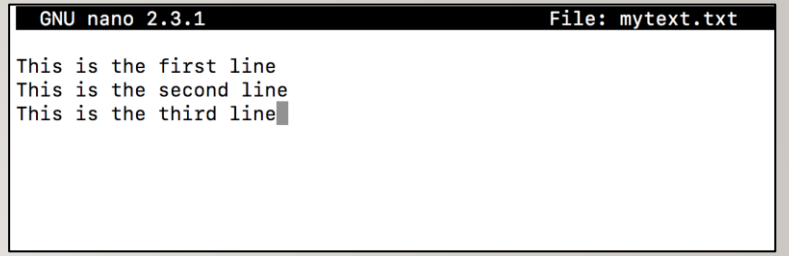


The **Nano Text Editor** is Easy to Operate for Novice Users.



The Vi Text Editor, although taking longer to master, has outstanding features and allow the user to be more productive.

CREATING TEXT FILES



Nano Text Editor

The **nano** text editor is considered to be an easiy-to-use text editor. When using the nano text editor, you are placed in **INPUT** mode, to enter text immediately.

Nano editing **commands** typically consist of the **^** symbol which represents the **<ctrl>** key followed by a character

NOTE: There is no **undo** command in Nano!

The table on the right list a few Nano commands and their purpose. Refer to **week 2 notes** for a **nano reference sheet.**

NOTE: In the Nano reference sheet, the letter **M** represents the **<esc>** key

Key Combination	Purpose
<ctrl><space> , **<esc><space>**	Move forward / backward one word
<ctrl>a , **<ctrl>e**	Move to beginning / end of line
<ctrl>k	Cut line
<esc>6	Copy Line
<ctrl>u	Paste Cut / Copied Text
<ctrl>g	Display help screen
<ctrl>x	Save and exit editing session

MANAGING DIRECTORIES

Instructor Demonstration

Your instructor will demonstrate how to create and edit a text file using the nano text editor.

CREATING TEXT FILES

`vi` Text Editor

The **vi** (**vim**) text editor (although taking longer to learn) has outstanding features to increase coding productivity..

The major different between nano and vi is that **vi starts in COMMAND LINE mode**. You need to issue letter commands to perform text editing or press colon ":" to enter last line mode to issue more complex commands.

To make it easier to learn how to use this text editor, an **online tutorial** was created (two decades ago) to provide you "hands-on" experience in command editing techniques.

To run this tutorial, issue the following command in Matrix:
/home/murray.saul/vi-tutorial

You can refer to your **week 2 notes** for a **vi command reference sheet**.

Key Combination	Purpose
`i`	Enter **INSERT** mode
`<esc>`	Return to COMMAND mode
`B , W`	Move forward / backward one word
`0 , $`	Move to beginning / end of line
`dd`	Cut line
`yy`	Copy Line
`p , P`	Paste below / above line
`:help`	Display help screen
`:x`	Save and exit editing session

MANAGING DIRECTORIES

Instructor Demonstration

Your instructor will demonstrate how to create and edit a text file using the **vi** text editor.

MANAGING TEXT FILES

Purpose

It is **essential** for students in this course not only to create text files but also to learn how to **manage** text files.

Students need to learn how to **create** empty files, **copy** files for backup purposes, **move** or **rename** incorrectly spelled filenames, **edit** files as well as **view** text file contents without the danger of editing or corrupting those files.

Students also need to learn how to **remove** files, check for **differences** between a couple of files as well as **obtain information** regarding the status of a file and information regarding the file's content.

MANAGING TEXT FILES

Text File Management Commands

Here are common text file management commands:

Linux Command	Purpose
`touch`	Create empty file(s) / Updates Existing File's Date/Time Stamp
`cat`	Display text file's contents without editing (small files)
`more , less`	Display / Navigate within large text files without editing
`head , tail`	View lines at top/bottom of file
`grep`	Display lines in file that match a pattern
`cp`	Copy text file(s)
`mv`	Move / Rename text files
`rm`	Remove text file(s)
`diff`	Displays differences between 2 files

MANAGING TEXT FILES

Text File Management Commands

Here are some **additional** text file management commands:

Linux Command	Purpose
`sort`	Display contents of file in sorted order
`uniq`	Display identical adjacent lines only once
`file`	Gives info about the contents of the file (e.g. file with no extension)

`find`	To find files matching specified characteristics:
	
	`find . -name "file*"`
	lists pathname of any filenames beginning with "file", from the current directory and any subdirectories
	`find . -size +50k`
	lists pathname of any files larger than 50 kb, from the current directory and any subdirectories
	`find . -mmin -5`
	lists files modified less than 5 minutes ago

MANAGING DIRECTORIES

Managing Manipulating Text Files

Your instructor will demonstrate how to **manage** / **manipulate** text files

- Create empty files
- View small and large text files
- Sort files
- Display matched pattern file content
- Remove duplicate lines
- Compare files for differences
- Obtain file information / List file pathnames

HANDS-ON TIME / HOMEWORK

Getting Practice

To get practice to help perform assignment #1, perform the online tutorial
Tutorial2: Unix / Linux File Management (**ctrl-click** to open link):

- INVESTIGATION 2: MANAGING TEXT FILES
- LINUX PRACTICE QUESTIONS (Questions 9 – 16)

Perform **section 2: Basic Unix Commands** (parts **4**, **5** and **6**) of your **online assignment #1**.

NOTE: You should have completed the first two sections of assignment #1. By the end of **next** week, you should have all the skills to complete the remainder of **your assignment #1**!