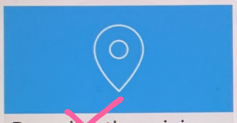
Introduction Linux and Unix

What you will learn

Explain what an operating system (OS) is



Describe the origins of Linux and Unix

List some of the features of Linux and Unix

What is an operating system?

An operating system, or OS, is software that:

- Manages hardware and resources
- Allows interaction with hardware to perform useful tasks

What is Unix?

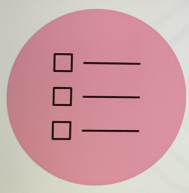
- Unix is a family of operating systems
- Popular Unix-based OSs include:
  - Oracle Solaris (and Open Solaris)
  - FreeBSD
  - HP-UX
  - IBM AIX
  - Apple macOS most popular today

What is Linux?

Linux is one of the OS operator

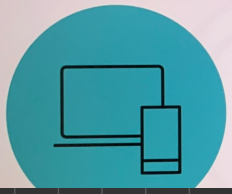
- Family of Unix-like OSs
  - Usually specific distribution
- Originally developed as an effort to create a free, open-source Unix OS

Linux features



- Free and open source
  - ↳ mean anyone can view the source code
- Secure
  - ↳ most secure OS
- Multi-user
  - ↳ support multiple users accessing the system simultaneously
- Multitasking
  - ↳ can perform multiple jobs and application at the same time
- Portability
  - ↳ can run on many different types of devices

Today



- BSD-based macOS runs on millions of devices
- Billions of Linux instances run on servers, serving the modern web
- Modern Linux OSs are gaining popularity for PCs
  - ↳ such as Ubuntu

Linux use cases today

Linux-powered server are clustered together for high-performance computing

Android uses a Linux-based kernel

Supercomputers

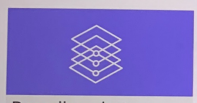
Data centers and cloud services

PCs

most of them installed Linux for Learning Experience or as their daily driver

Linux Distributions

What you will learn



Describe what a Linux distribution is

Differentiate between some common Linux distributions

Identify the use cases of some popular Linux distributions

What is a Linux distribution?

- A specific flavor of Linux OS
- Also referred to as Distro
- Linux kernel is the core component
- There are hundreds of Linux distros

Linux distro differences

- System utilities
- Graphical user interface (GUI)
- Shell commands
- Support types:
  - Community versus enterprise
  - Long-term support (LTS) versus rolling release

Popular Distro:

Linux distros: Debian

- First release in 1993 (0.01), first stable release in 1996 (1.1)

- Stable, reliable, fully open source
- Supports many computer architectures
- Largest community-run distro

Linux distros: Ubuntu

- First release in 2004 (4.10)

- Debian-based
- Developed and managed by Canonical
- Three editions:
  - Ubuntu Desktop
  - Ubuntu Server
  - Ubuntu Core

Linux distros: Red Hat Linux

- Red Hat Linux, like Debian, is a "core" Linux distro.

- Stable, reliable, fully open source
- Managed by Red Hat,
- Red Hat Enterprise Linux (RHEL)

Linux distros: Fedora

- Fedora is known as a stable operating system

- Supports many architectures
- Very reliable and secure
- Actively developed, large community
- Sponsored by Red Hat

Linux distros: SUSE Enterprise

- SUSE Linux Enterprise (SLE) available in two editions:
  - Server (SLES)
  - Desktop (SLED)
- Supports many architectures
- SUSE Package Hub
- Maintained by SUSE

Linux distros: Arch Linux

- Do-it-yourself approach
- Highly configurable
- Requires strong understanding of Linux and system tools
- Leading-edge software

Continue Here

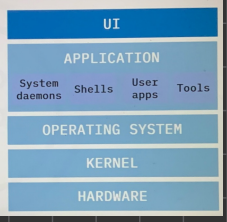
Overview of Linux Architecture

& contains 5 layers

User interface

- Allows users to interact with the machine → by keyboard or mouse
- GUI

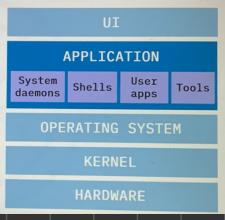
Tasks include:
- Using a Web browser to send an email
- Using a music player to listen to a song



Applications

Any software that lets you perform a task

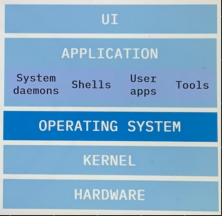
Applications include:
- System tools
- Programming languages
- Shells
- User apps (such as browsers, text editors, games)



Operating system

Controls the jobs and programs vital to health and stability

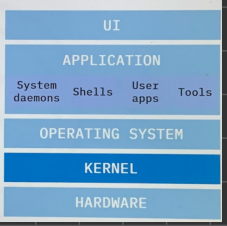
Functions:
- Assigns software to users
- Helps detect errors and prevent failures
- Performs file management tasks



Kernel

- Performs vital operations
- Lowest-level software — lower vital jobs
- Starts on boot
- Bridge between apps and hardware

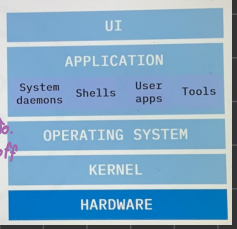
Key jobs:
- Memory management
- Process management
- Device drivers
- Security



Hardware

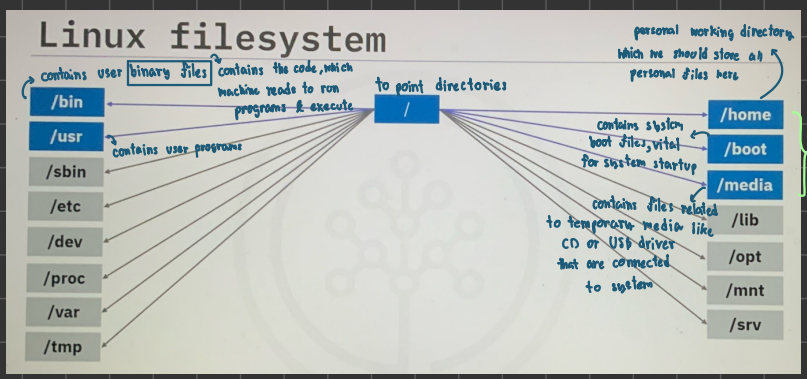
Consists of all physical or electronic devices on your PC

- Includes:
  - CPU : executing
  - RAM : used to hold temporary info
  - Storage : persist when com. power off
  - Screen
  - USB devices



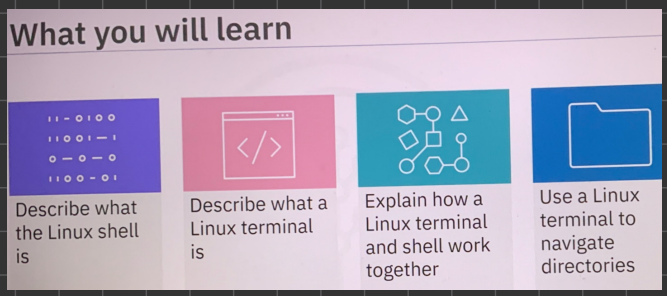
Linux filesystem

is
- Collection of files in your machine — it includes the files needed to run the machine and App.
- Begins at root directory (/)
- Tree-like structure
- Assigns appropriate access rights



Linux Terminal Overview

What you will learn



Overview of the Linux shell

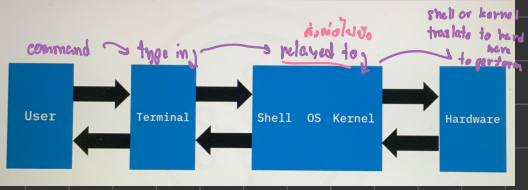
- The **shell** is an OS-level application that interprets commands
- Use commands to:
  - Move and copy files ✓
  - Write to and read from files ✓
  - Extract and filter data ✓
  - Search for data ✓
- Shells:
  - Bash
  - Zsh

Overview of a Linux terminal

- The **terminal** is an application you use to interact with the shell
- Enter commands and receive output from them

```
/home/me/ $ python myprogram.py
Hello, World!
```

Communicating with Linux system



Paths in the Linux filesystem

- The **filesystem** is the human-readable directory or file location

```
/home/me/Documents/
```

- **"a/b"** indicates the file or directory named *b* inside the directory named *a*
- Special paths:
  - ~ Home directory
  - / Root directory
  - .. Parent of current directory
  - . Current directory

Changing the current working directory

```
/home $ cd /  ← Go to root Directory
/ $ cd bin  ← to bin directory
/bin $ ./ls  ← Displays in the terminal window the name of all files and directories
[    cat    cp    dash    dd    echo    expr    kill
...
ls   mv     ps    rm      sh    stty    tcsh    unlink
/bin $ cd ~  ← to home directory
/home/me $ ls  ← although it doesn't contain the 'ls' program, we can still run it
Documents  Pictures  Downloads  Movies  Music  Desktop
```

Changing the current working directory

this Ex is root directory

```
/home $ cd ..
/ $ cd /media/my-usb-drive
/media/my-usb-drive$ cd ../../home/me/Documents
/home/me/Documents $ cd ../..
/home/me $ python ./myprogram.py
Hello, World!
```

`/home $ cd ..` → to parent of the existing current working directory

`/ $ cd /media/my-usb-drive` type the path after 'cd' command to navigate to desire path

`/media/my-usb-drive$ cd ../../home/me/Documents` ↳ navigate down to desire directory

`/home/me/Documents $ cd ../..` navigate up to the root

`/home/me $ python ./myprogram.py` means

`Hello, World!`

to navigate up and down the tree → to root direc., so this one is root path for /home/me for Document dir.

Here

to make it brief

Vital Commands to Know for the Shell

- pwd : print path name → showing which directory we are currently working
- ls : list the contents in directory → showing all contents in directory

Tips for Linux Terminal

- press 'Tap': to quickly enter a command to navigate, autocomplete that path (we have to start the path to navigate with some consonants to make Linux knew and autocomplete it)
  ↳ if that navigated path contains more than one directory we have to type further to indicate that path
- press 'Double Taps': to show available options for directory
- press 'Up Arrow': Commands that we have typed in history
  'Down Arrow': Do in opposite direction

LAB 1

1.2. Viewing files and directories within any directory

view contents in that path

If you know the path to a directory, you can view its contents by passing the path name as a *command line argument* to the `ls` command as follows:

`ls [PATH TO DIRECTORY]`

For example:

```
1 ls /
```

will show the contents of 'slash', your Linux system's root directory.

Recall some of the standard subdirectories of 'slash' that you've learned about previously:

to use: ls /bin

Directory	Contains
/bin	System commands, also called *binaries*
/sbin	System administration binaries
/usr	User programs and data
/home	Home directory
/media	Removable media device directories

output

```
theia@theia-tanaphatsa2:/home/project$ ls /
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
```

Exercise 2 - Navigating Directories

`cd`

In this exercise, you will explore directories using the `cd` command.

Recall the symbols used to navigate to special paths:

type after 'cd' command

Symbol	Stands for path to
~	Home directory
/	Root directory
.	Current directory
..	Parent directory

2.5. Changing from your working directory back to your home directory

There are multiple ways to change your current working directory back to your default home directory. One way is to return to your parent directory, `..`, and type the address of your home directory `/home/theia`:

```
1 cd ../home/theia
```

A simpler way to do this would be to use the `~` symbol to quickly and directly navigate to your home directory:

```
1 cd ~
```

Both methods will change your working directory back to your home directory.

2.6. Changing from your working directory to your `project` directory

Directories that are contained within the same parent directory are called **siblings**.

In this environment, we have provided a special empty `project` directory for your work. This `project` directory is located at `/home/project` and is a sibling directory to `/home/theia`.

Change your current working directory to its sibling directory, your `project` directory, by entering the following command:

```
1 cd ../project
```

Creating and Editing Text file

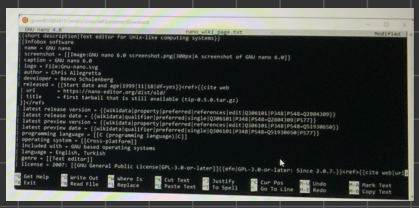
What you will learn

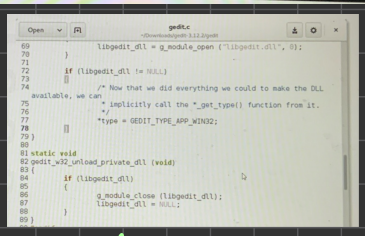
- List popular text editors for Linux
- Describe a popular GUI-based text editor
- Use command-line editors to work with a file

Popular text editors

- Command-line text editors:
  - GNU nano: smart and friendly
  - vi: traditional command-line, for Unix
  - vim: powerful mode based
- GUI-based text editors:
  - gedit default editor for GNOME
- Command-line or GUI:
  - emacs oldest free

popular GUI-based editor





↓ popular, modern text editor

↓ most installed on Linux distros

↓ simplicity and ease to use

gedit

Features of gedit

A general-purpose editor, easy to use with a clean, simple GUI:

- Integrated file browser
- Undo and redo
- Search and replace
- Extensibility using plugin

GNU nano

Features of GNU nano

A command-line text editor provides:

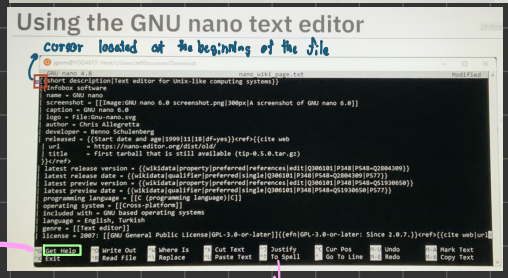
- Undo and redo
- Search and replace
- Syntax highlighting
- Indenting groups of lines
- Line numbers
- Line-by-line scrolling
- Multiple buffers

Using the GNU nano text editor

- To open a text file in GNU nano from the command prompt, type:

```
nano <filename>
```

Using the GNU nano text editor



to use press "control" button, then type the letter for the desire command

Ex "Get Help" control + "G"

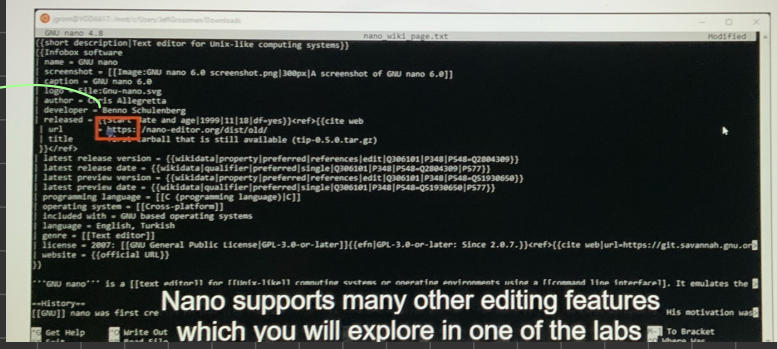
Using nano to search for a string

```
Search [1999]:
^G Get Help          M-C Case Sens        M-B Backwards
^C Cancel            M-R Regexp           ^R Replace
```

searching for "https"

```
Search [1999]: https
^G Get Help          M-C Case Sens        M-B Backwards
^C Cancel            M-R Regexp           ^R Replace
```

Using nano to search for a string



cursor moved to this line

Nano supports many other editing features which you will explore in one of the labs

VIM

File editing with vim

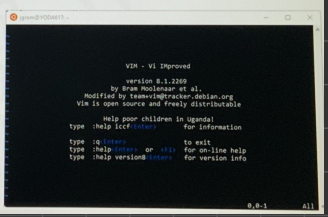
Vim is a traditional and very powerful command-line editor

To start vim, type:

```
vim
```

To specify a file to edit, type:

```
vim <filename>
```



Insert mode

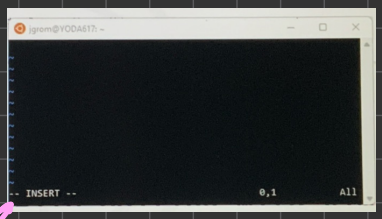
File editing with vim

Two basic modes:

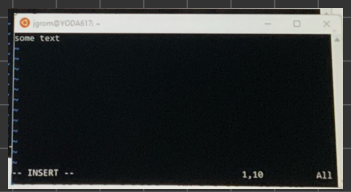
- **Insert** and **Command**

Insert mode:

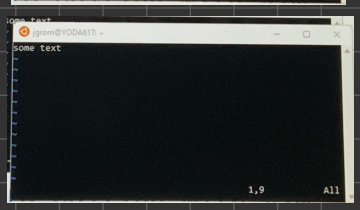
- Press *i* to enter Insert mode



- Type some text



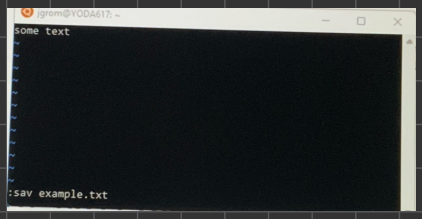
- Press Esc to exit Insert mode and switch to Command mode
- The text is written to the buffer at the cursor location



Command mode

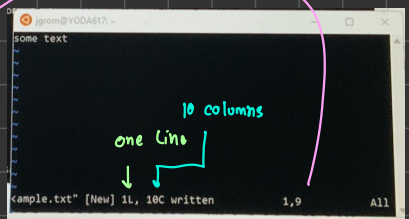
File editing with vim

- **Command mode**:
  - Enter `:sav example.txt` to create a file and write the buffer to the file



- Enter `:w` to write the buffer to the file without exiting
- Enter `:q` to quit vim session
- Enter `:q!` to quit without saving

when success execute



there are many command in vim

Lab : Upgrading and installing packages for Nano and Vim

Sudo (strand for super-user do): • to access a system administration utility
↳ in Linux, it's required password to run sudo

apt (activate the powerful command): • best practise to first 'update' package list index (sudo apt update)

• use 'sudo' to query, than 'apt', then 'upgrade' for nano
↳ also use 'apt' to install Vim

start create .txt file →

2.2 Creating and editing a text file with nano

To create a new file, enter

```
nano hello_world.txt
```

in the terminal. This will simultaneously create a new file called `hello_world.txt` and enable you to begin editing it using the nano text editor.

> *Double-check that your new file was created by opening another terminal window and running the `ls` command on `/home/project`. You should see `hello_world.txt` listed.*

In your nano terminal, whatever you type will be added to your *text buffer*, where text is stored until you save it. Type the following text in your nano terminal:

```
Hello world!
```

This will create the text `Hello world!` in your text buffer.

To create another line of text, press `Enter`. In your new line, type

```
ond line of my first-ever text file created with nano.
```

to create a second line of text in your text buffer.

Now:

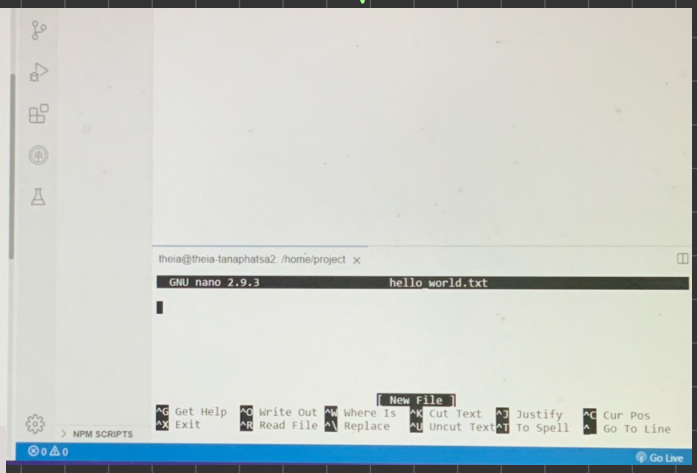
1. Press `CTRL` + `X` to exit nano.
2. You will be prompted as follows:

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES)
Y Yes
N No           ^C Cancel
```

Press `Y` to save your new lines of text to your file.

3. Press `Enter` to confirm the file name.

At this point, nano should have exited and returned you to the command prompt.



Call file ↓

2.3 Verifying your new text file

By entering a familiar command, such as

```
cat hello_world.txt
```

you should be able to inspect and verify that your new file contains the two lines you wrote to it with nano. Cool!

showed the line that I have written

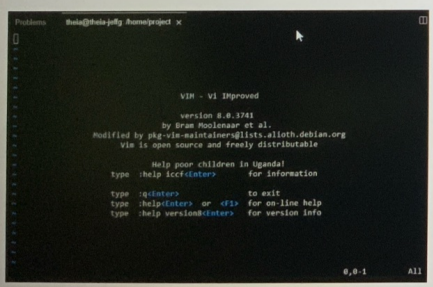
```
theia@theia-tanaphatsa2:/home/project$ nano hello_world.txt
theia@theia-tanaphatsa2:/home/project$ cat hello_world.txt
Hello world!
This is the second line of my first-ever text file created with nano.
theia@theia-tanaphatsa2:/home/project$
```

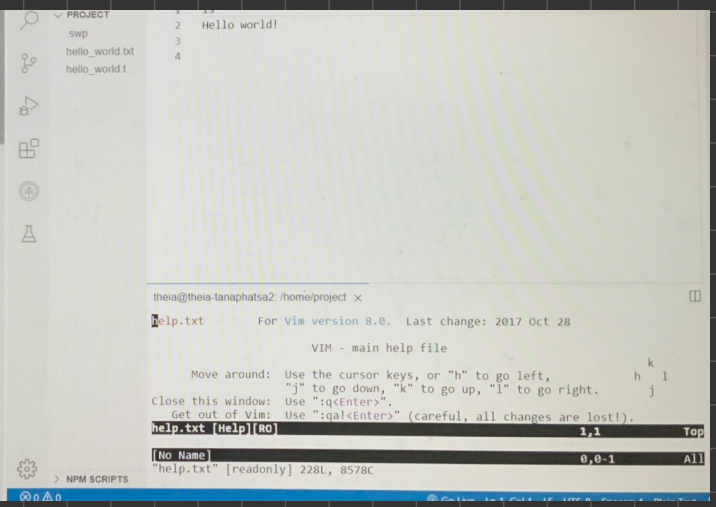
3.1 Quick intro to Vim

Recall that Vim has two basic modes: `Insert` mode, where you enter text, and `Command` mode, where you do everything else. You can start Vim simply by entering

```
vim
```

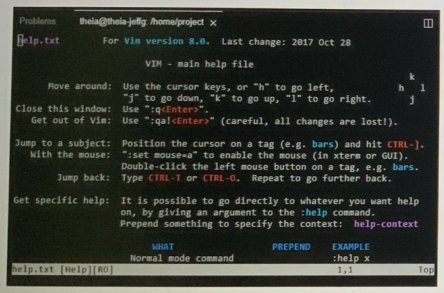
at the command prompt, which displays something like the following in your terminal window:





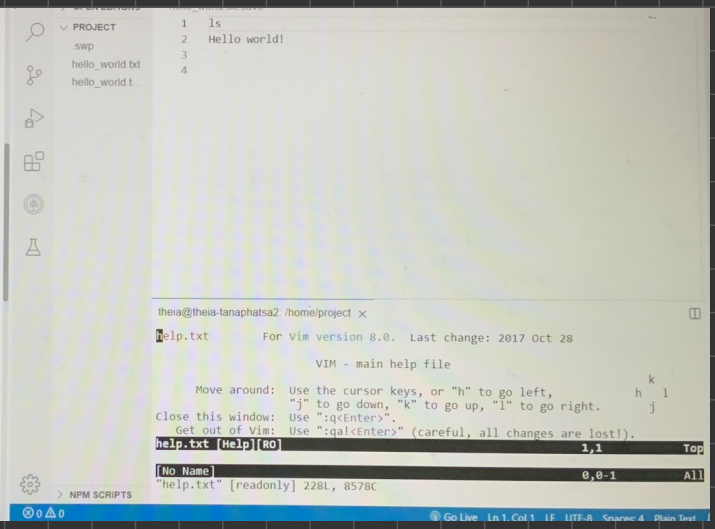
Notice that you can get help on Vim by entering `:help`, and you can quit Vim by entering `:q`.

Go ahead and enter `:help`. This brings up an informative help file you can scroll through.



When you are done reading, simply enter `:q` to quit Vim and return to the command prompt.

Vim is very powerful and takes some time to learn. We're just covering the very basics here and leaving it to you to explore further. Check out the official Vim site at https://www.vim.org.



3.2 Creating and editing a text file with Vim

Begin by navigating back to your `/home/project` directory if you aren't already there.

Type the command,

```
vim hello_world_2.txt
```

to create a new file called `hello_world_2.txt` and edit it using Vim.

Once your Vim session has started, go ahead and press `i` to enter `Insert` mode. This is the mode where you can enter and delete text in the text buffer.

Go ahead and type some text in the buffer, for example:

```
Hello World!
```

Just like in nano, press `Enter` to start a new line, and then type

```
This is the second line.
```

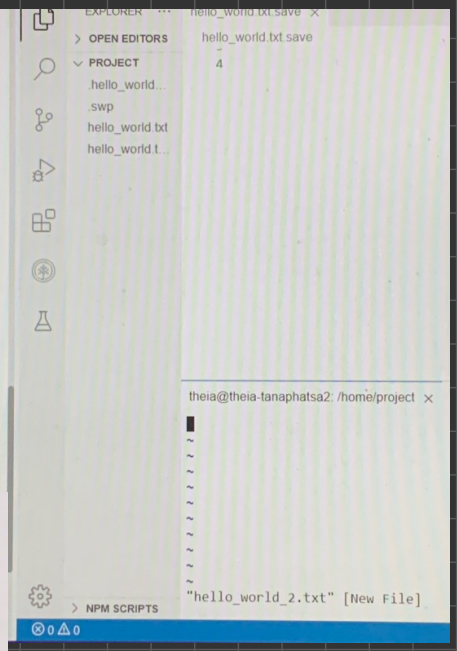
to create a second line of text in the buffer.

When you're done typing text in the buffer, press the Escape key, `Esc`, to exit the `Insert` mode. This brings you to `Command` mode.

> *In Vim, it's easy to accidentally end up in a mode you didn't intend to be in. No worries - you can use the Esc key to return to Command mode.*

Now that you are back in `Command` mode, you can save your work by entering the command `:w`. This writes the contents of the text buffer to your text file.

Finally, you can exit your Vim session by entering `:q`.



2. Using Vim, create a file called `done.txt` that prints `"I am done with the lab!"` when you execute the file with Bash.

In this exercise you will use Vim to create a file that contains the `echo` command, which you may not have seen yet. You will also *run* the file using Bash, a shell scripting language that you will learn more about later in this course. Basically, the file you create will contain a basic command that Bash can interpret and do something with.

▸ Click here for Hint

▸ Click here for Solution 1

▾ Click here for Solution 2

> *Press `i` to enter `Insert` mode*
>
> *Add the following content to the file:*

```
echo "I am done with the lab!"
```

> *Then save your work and exit by:*
>
> 1. Using `Esc` to exit out of `Insert` mode and return to `Command` mode
> 2. Typing `:wq` and presssing `Enter`

▾ Click here for Solution 3

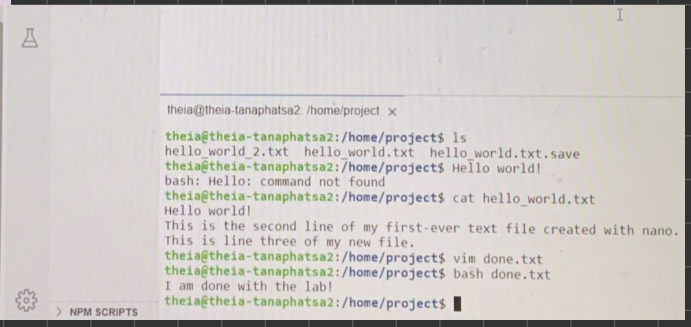
> *Once you've saved and exited and are back at the command prompt, you can run the file with Bash by entering:*

```
bash done.txt
```

This command invokes the Bash shell to interpret the text contained in `done.txt` as a command. In particular, Bash runs the `echo` command along with the quoted text as the input to `echo`, while `echo` simply prints the input text to the terminal window.

When finished, you should see the following text echoed to your terminal:

```
I am done with the lab!
```



Installing Software and Updates

What you will learn

- Describe packages and package managers
- Differentiate between packages for deb- and RPM-based distros
- Use a package manager to install updates
- Use a package manager to install software

Packages and package managers

- Packages:
  - Archive files
  - For installing new software or updating existing software
- Package managers:
  - Manage the download and installation of packages
  - Available for different Linux distros
  - Can be GUI-based or command-line tools

Deb and RPM packages

→ are used by Linux managers

- Packages for Linux OS
- Distinct file types for different Linux OSs
- .deb files:
  - For Debian-based distributions such as Debian, Ubuntu, and Mint
  - deb stands for Debian
- .rpm files:
  - For Red Hat-based distributions such as CentOS/RHEL, Fedora, and openSUSE
  - RPM stands for Red Hat Package Manager

Deb and RPM packages

- deb and RPM formats are equivalent
- If a package is only available in one format, you can use alien to convert it:
  - RPM to deb:

```
alien <package-name>.rpm
```

  - deb to RPM:

```
alien -r <package-name>.deb
```

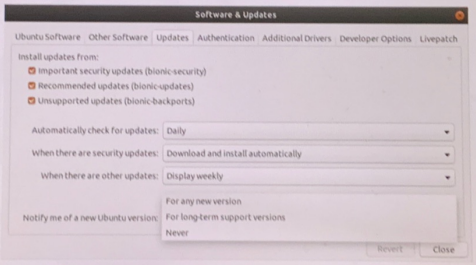
Package managers

- Benefits:
  - Automatically resolve dependencies
  - Notify you when updates are available
  - GUI-based package managers can automatically check for updates
  - Automatic or manual installation
- Linux distro package managers include PackageKit and Update Manager

Updating deb-based Linux

GUI tool: **Update Manager**

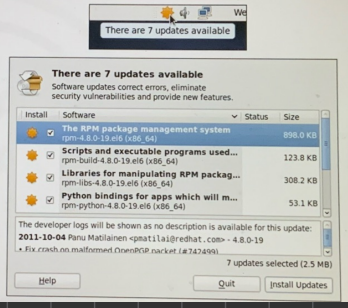
- Automatically checks for updates at configurable intervals
- Supports manual checking for updates



Updating RPM-based Linux

GUI tool: **PackageKit**

- Notifies you when updates are available
- Lists available software updates
- Installs updates in the background while you work



Updating RPM-based Linux

Command line tool: **yum**

```
$ sudo yum update
[sudo] password for user:
Fedora Modular 35 - x86_64 - Updates            3.1 MB/s | 2.8 MB     00:00
Last metadata expiration check: 0:00:01 ago on Fri 25 Mar 2022 09:08:11 PM EDT.
Dependencies resolved.
================================================================================
Package                    Arch   Version                Repo      Size
================================================================================
Upgrading:
…
kernel                     x86_64 5.16.16-200.fc35       updates   85 k
…
python3                    x86_64 3.10.3-1.fc35          updates   26 k
Transaction Summary
================================================================================
Install   30 Packages
Upgrade  752 Packages
Total download size: 1.2 G
Is this ok [y/N]:
```

Installing new software

Installing a deb package with apt:

```
sudo apt install <package-name>
```

Installing an RPM package with yum:

```
sudo yum install <package-name>
```