# Linux System Guide for CS20300 Students

Original Version by Taekyung Heo
Updated by Soojin Hwang
Updated by Jinwon Pyo
Updated by Hongbeen Kim

# What is Linux?

- **What is Linux?**
  - Linux is a modern, free operating system.
  - First developed by Linus Torvalds in 1991.
  - **Features**
    - Multi-tasking, multi-user
    - Various distributions (Ubuntu, CentOS, …)
    - Fully customizable
- **Why do we need Linux?**
  - Knowing how to use Linux systems is important for computer engineers.
  - You will do your projects on Linux systems.

# Connection to a Linux Machine

- How can I use a Linux machine?
  - You have to establish a connection to it.
- How can I connect to a machine?
  - Use ssh clients like PuTTY or **VSCode**.

PuTTY

**PuTTY: A Free Telnet/SSH Client**

- Very light-weight ssh client
- Does not need installation
- Not user-friendly interface

VSCode

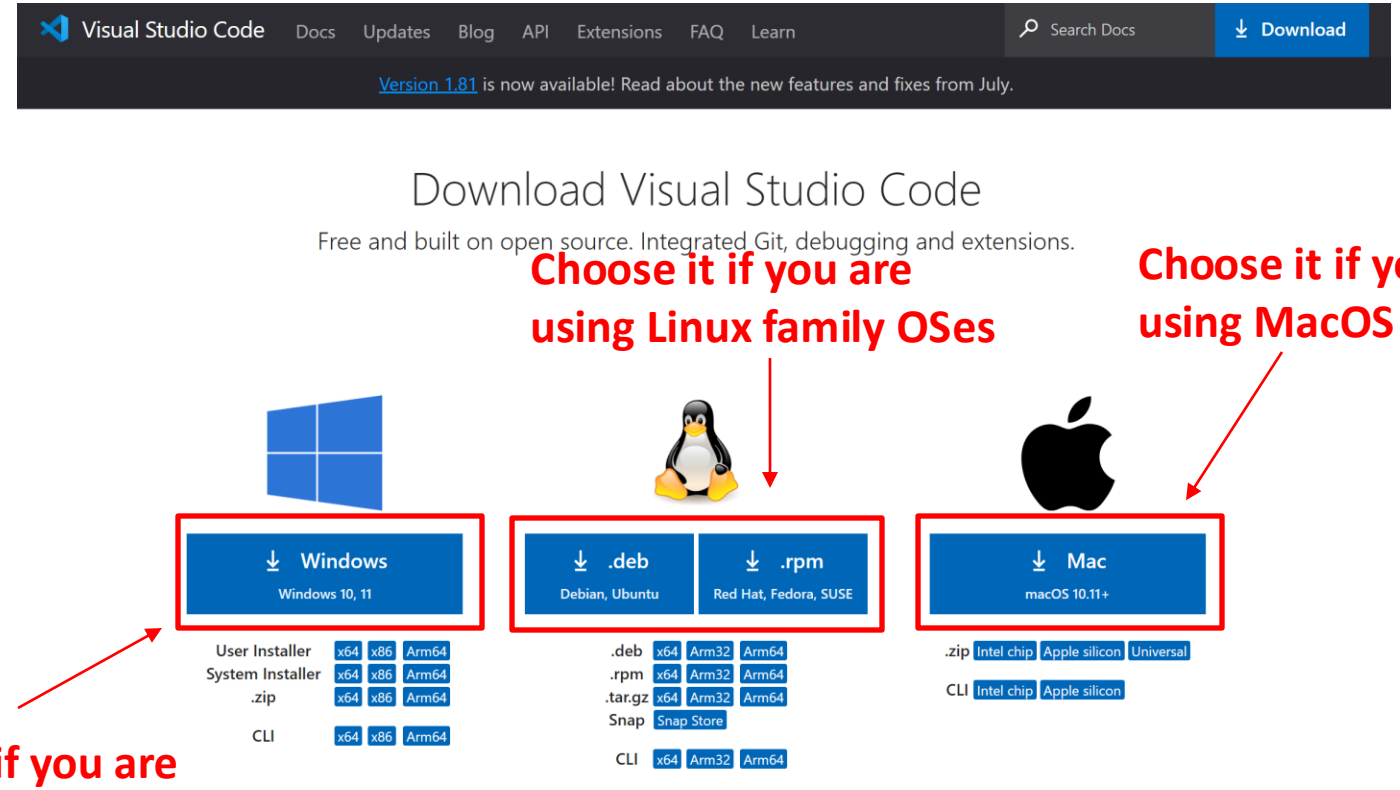
**VSCode (Visual Studio Code)**

- Needs installation & setup
- User-friendly interface

**We will exclusively use VSCode in this lab!**

# Install VSCode

- First You need to install VSCode
  - Link: https://code.visualstudio.com/download
  - Choose the OS version accordingly
  - Install the VSCode (you can just say ‘ok’ or ‘yes’ for installation options during the installation process)



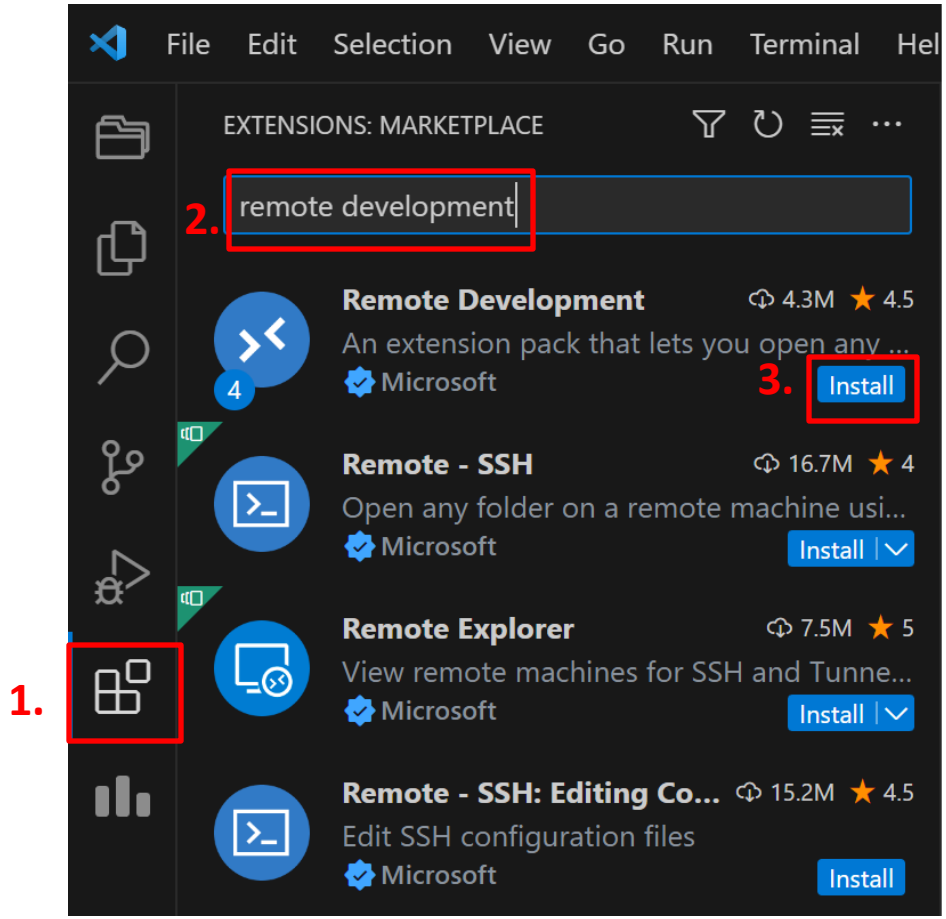
# Connection Example: VSCode

Assuming you have already installed VSCode in your local machine

Let's make a connection to the server machine to which you are designated

First you need to install one VSCode extension

Steps
1. Click the icon
2. Search "remote development"
3. Install it



# Connection Example: VSCode - 2

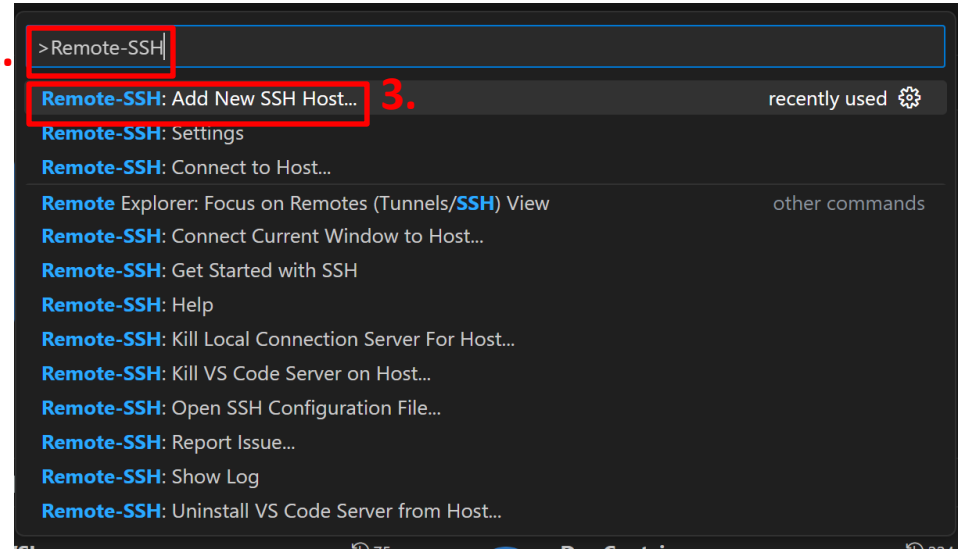
Now we are ready to make a SSH-connection to your designated remote machine

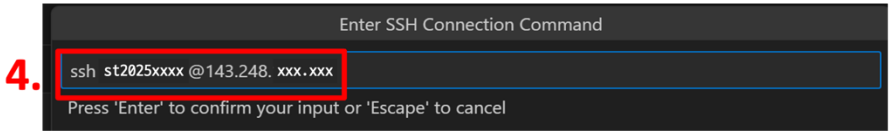
e.g.) Connection information

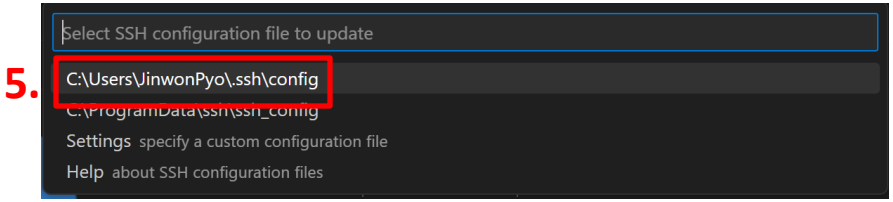
IP address: Your assigned IP address
ID: st[your student id]
Password: Your assigned password

Steps.

1. Hit the “F1” key (dropdown menu will pop)
2. Search “Remote-SSH”
3. Select “Add New SSH Host…”
4. Enter the given information as the following format (**When asked to choose the targeting OS, then choose “Linux”**)
5. Just select the first option



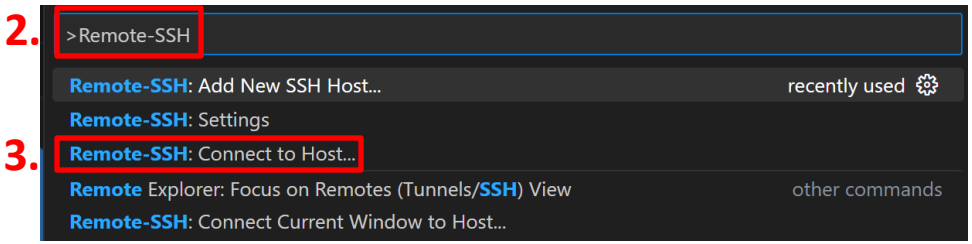



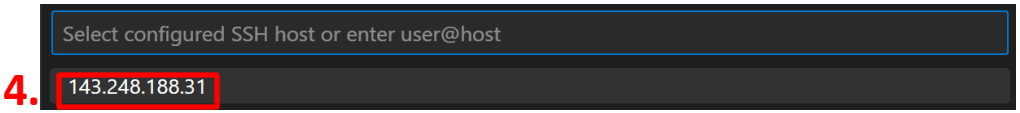



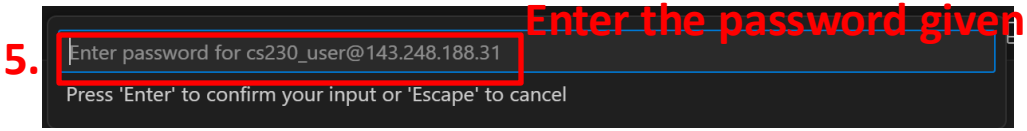
# Connection Example: VSCode - 3

Finally let's make an actual SSH connection to your remote machine

1. Hit the "F1" key (dropdown menu will pop)
2. Search "Remote-SSH"
3. Select "Connect to Host…"
4. Select the IP address you entered during the previous step
5. New VSCode window will pop out and ask you to enter password



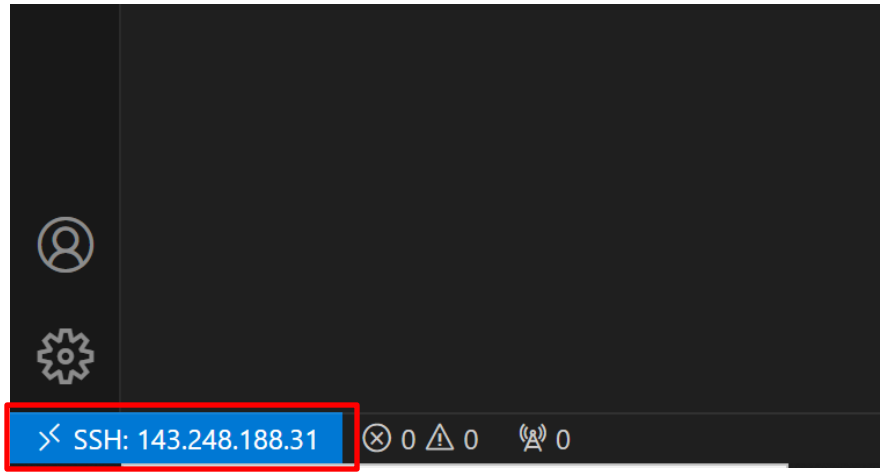




Enter the password given to you

# Connection Example: VSCode - 4

You are now in the remote machine!

Check out the bottom left corner of your VSCode window

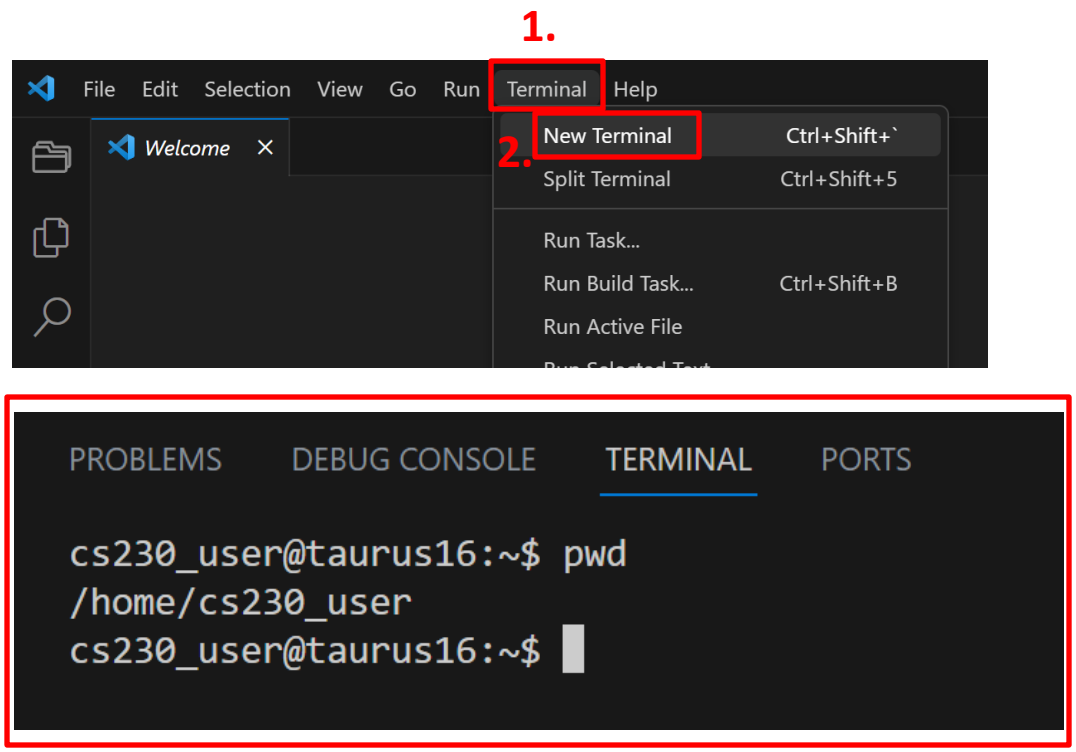


# Connection Example: VSCode - 5

You can also open up the Linux shell (or terminal) in VSCode through which you can interact your remote machine

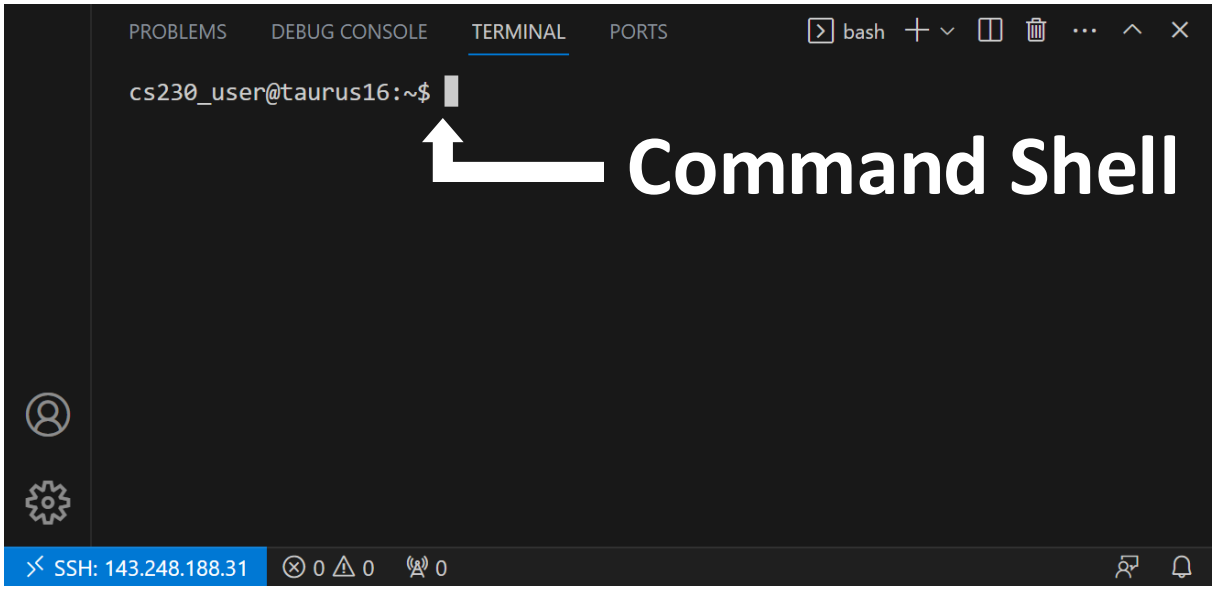
Steps.

1. Find “Terminal”
2. Select “New Terminal”
3. Then you will see the Linux shell (or terminal) pop up in the bottom of VSCode window



# Linux Shell

## Connected!



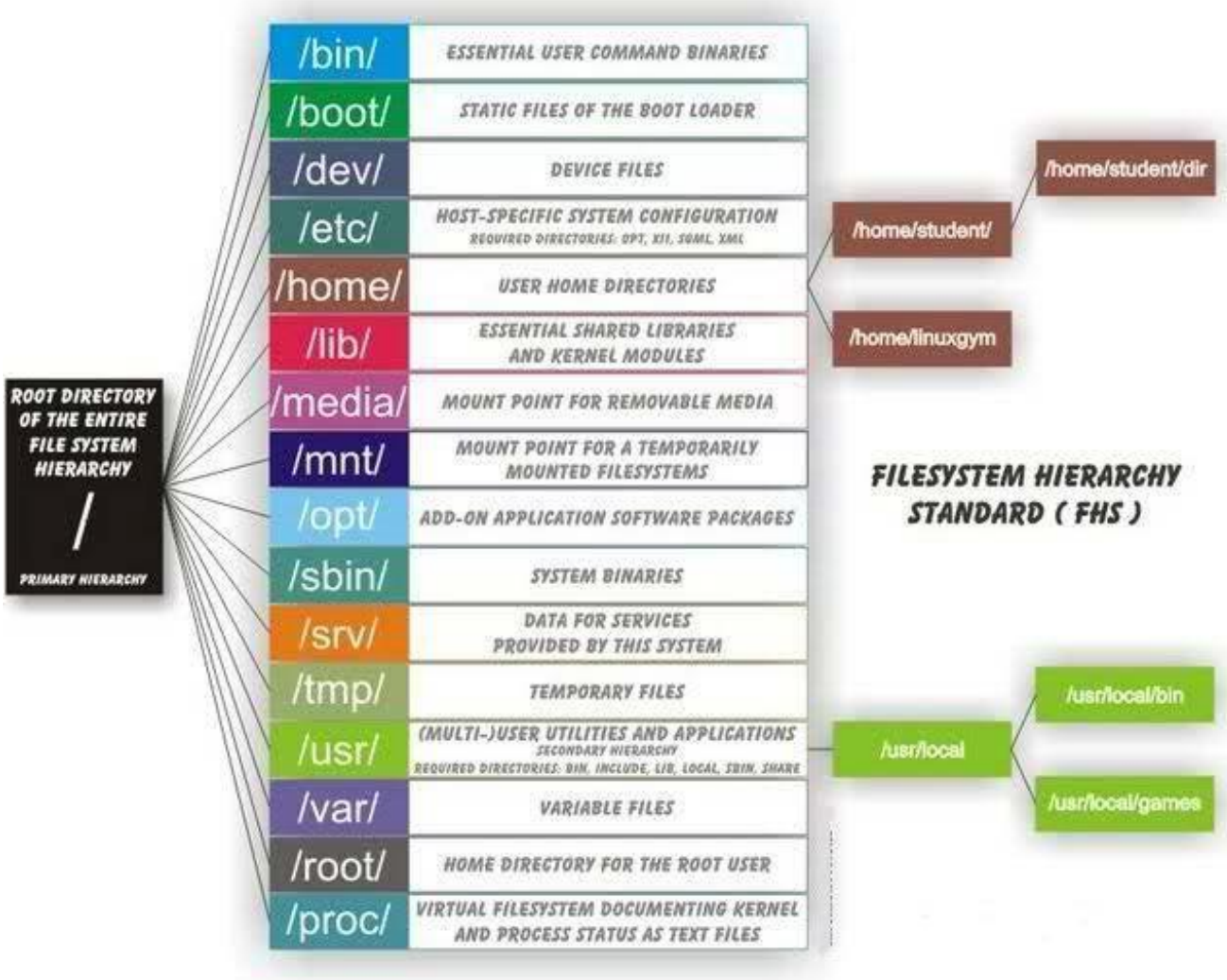

## Shell?

- **a command line interface between a user and Linux machine**

# Linux Directory

- / : root directory
- ~ : user's home directory (usually same as /home/[username])
- . : current(working) directory
- .. : upper(parent) directory

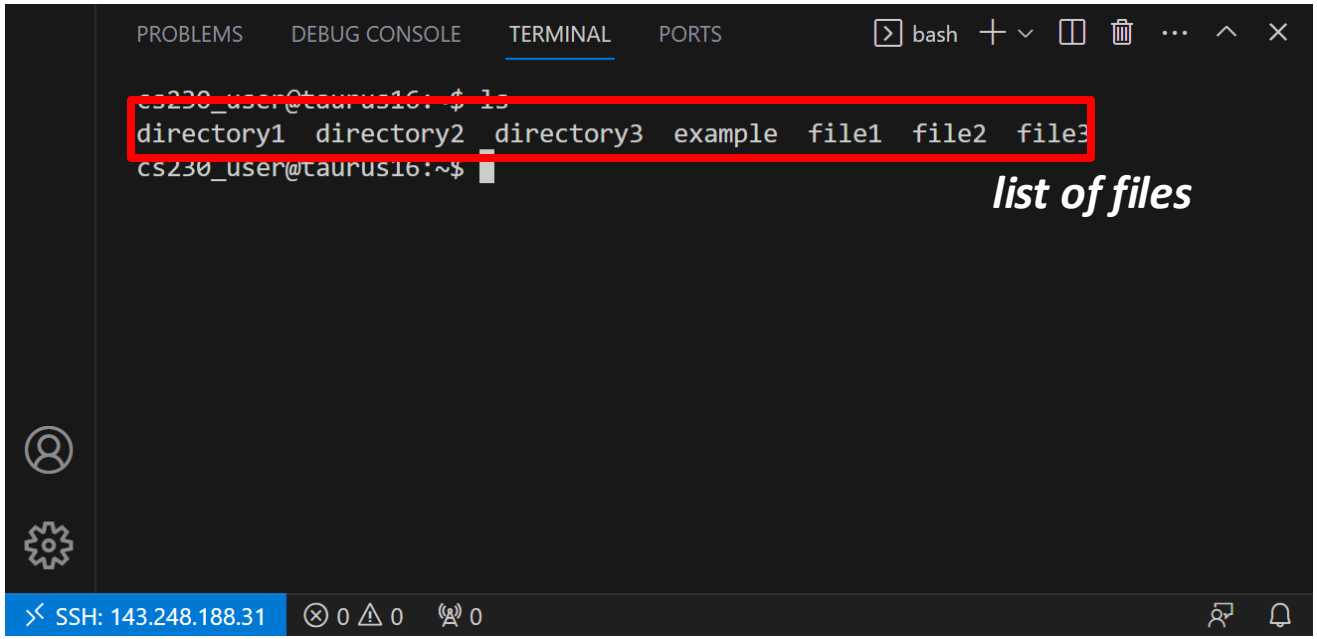
# Linux Directory Structure




http://www.tecmint.com/linux-directory-structure-and-important-files-paths-explained/

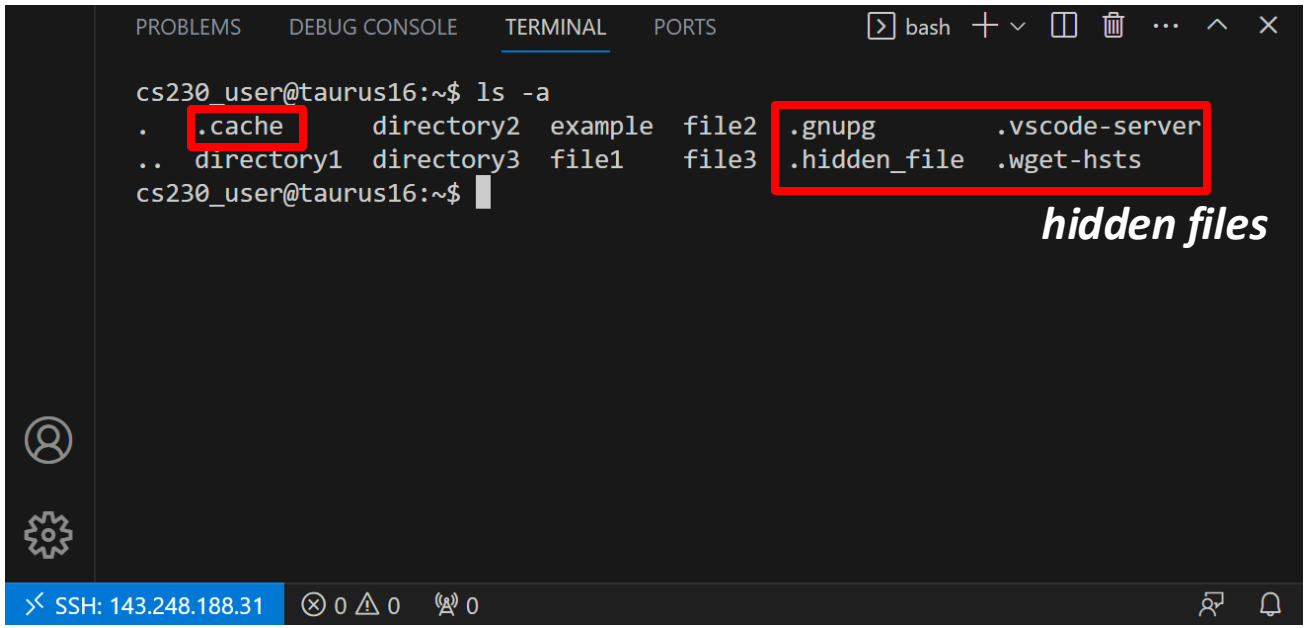
# Linux Commands

- ls [directory] (empty for working directory)
  - list directory contents



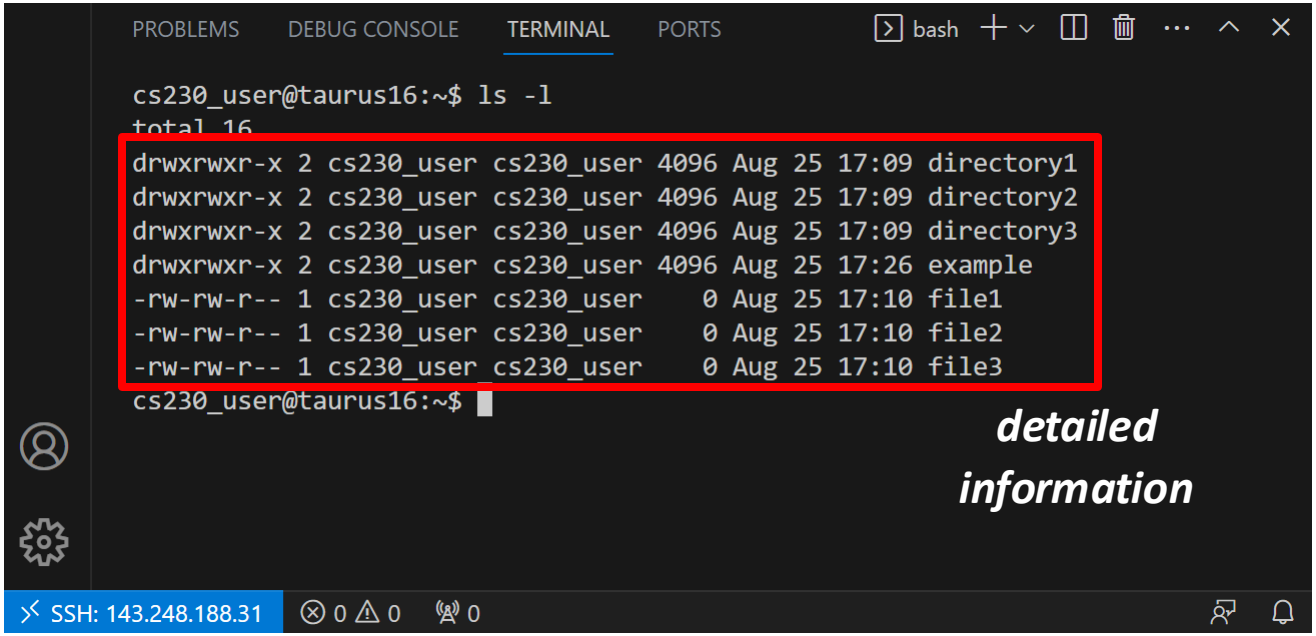
# Linux Commands

- ls [directory] (empty for working directory)
  - -a : Print the list including hidden contents (. name)



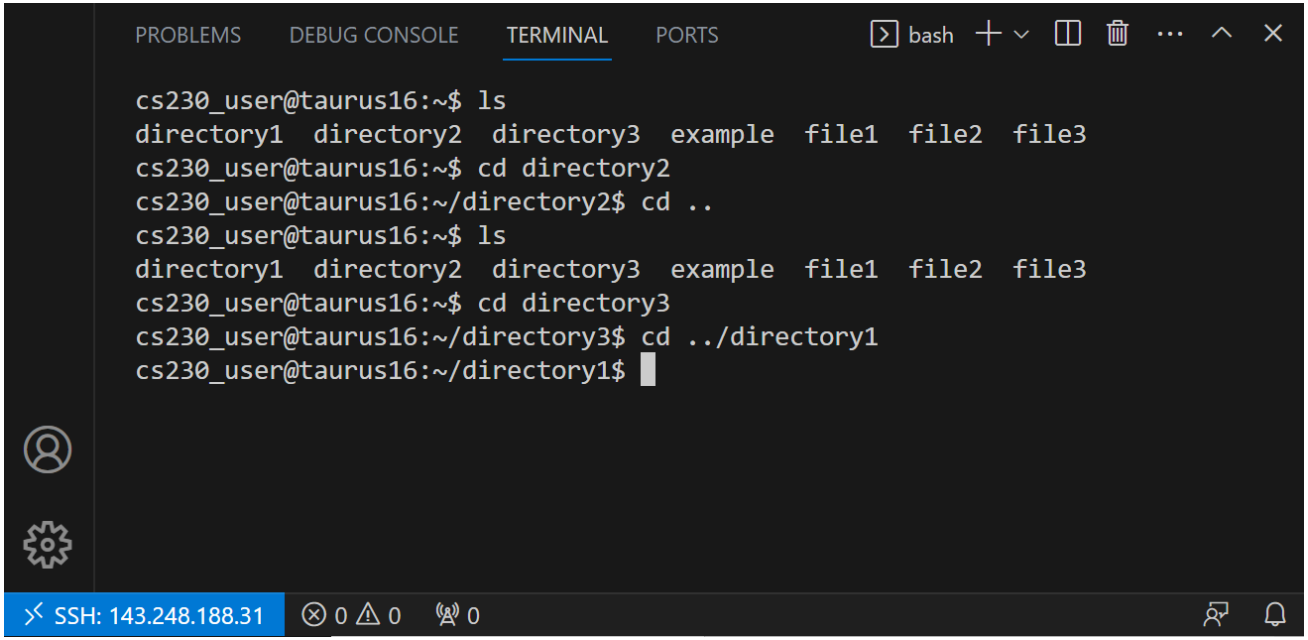
# Linux Commands

- ls [directory] (empty for working directory)
  - -l : Print the list of files with detailed information
    - Permissions, number of links, owner name, owner group, size, last modification time, name



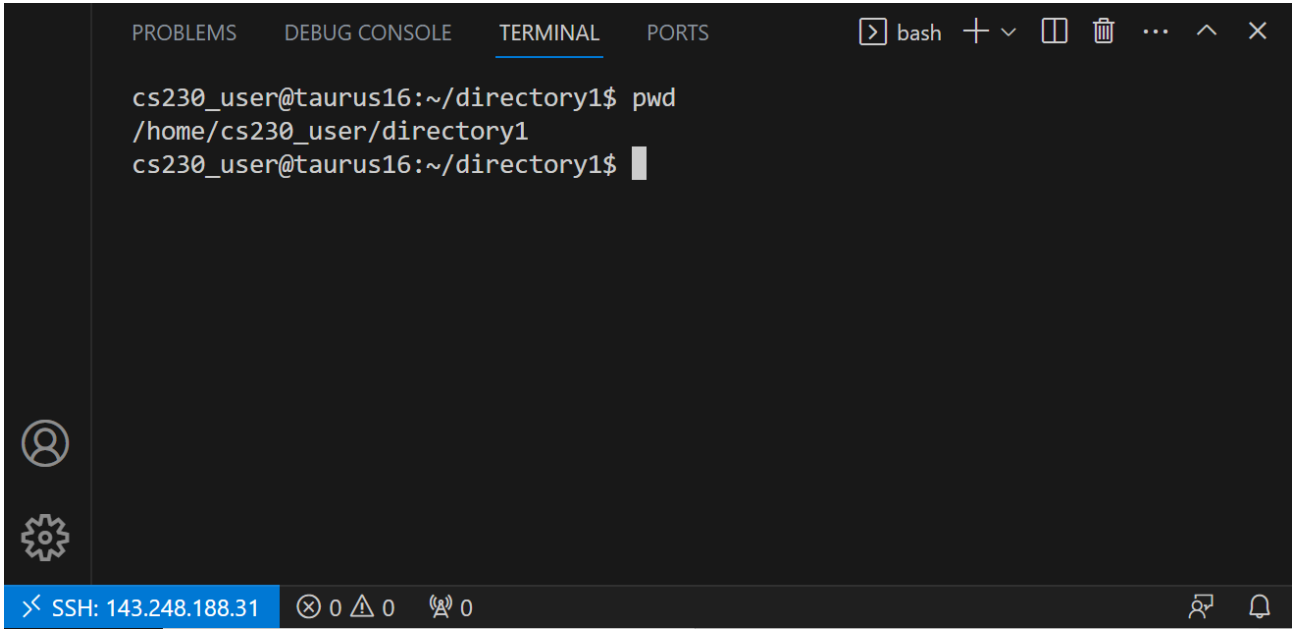
# Linux Commands

- cd [directory]
  - change working directory



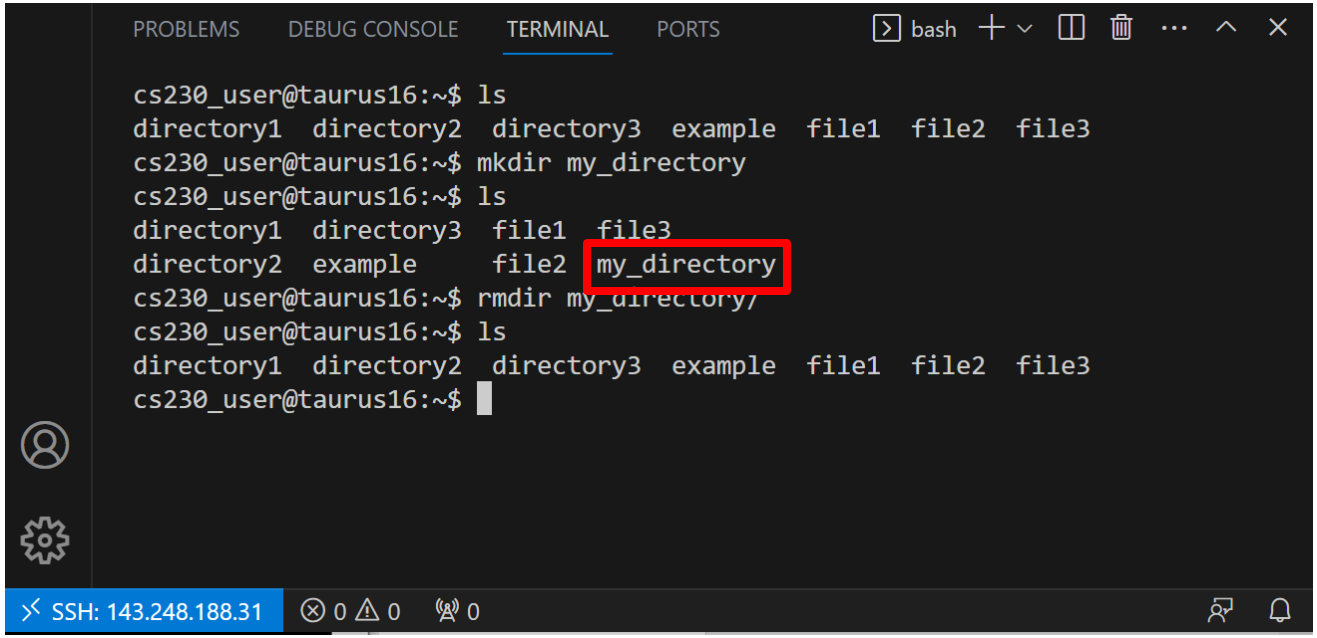
# Linux Commands

- pwd
  - print name(absolute path) of working directory



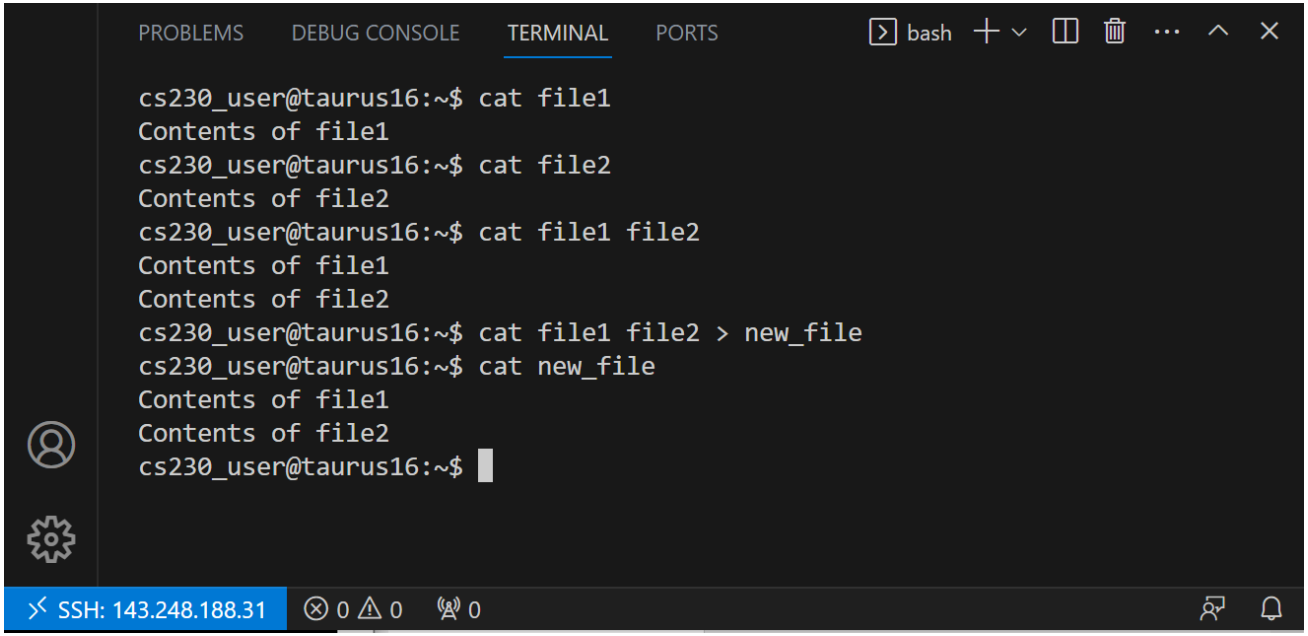
# Linux Commands

- mkdir [name], rmdir [name]
  - mkdir – make directories
  - rmdir – remove empty directories



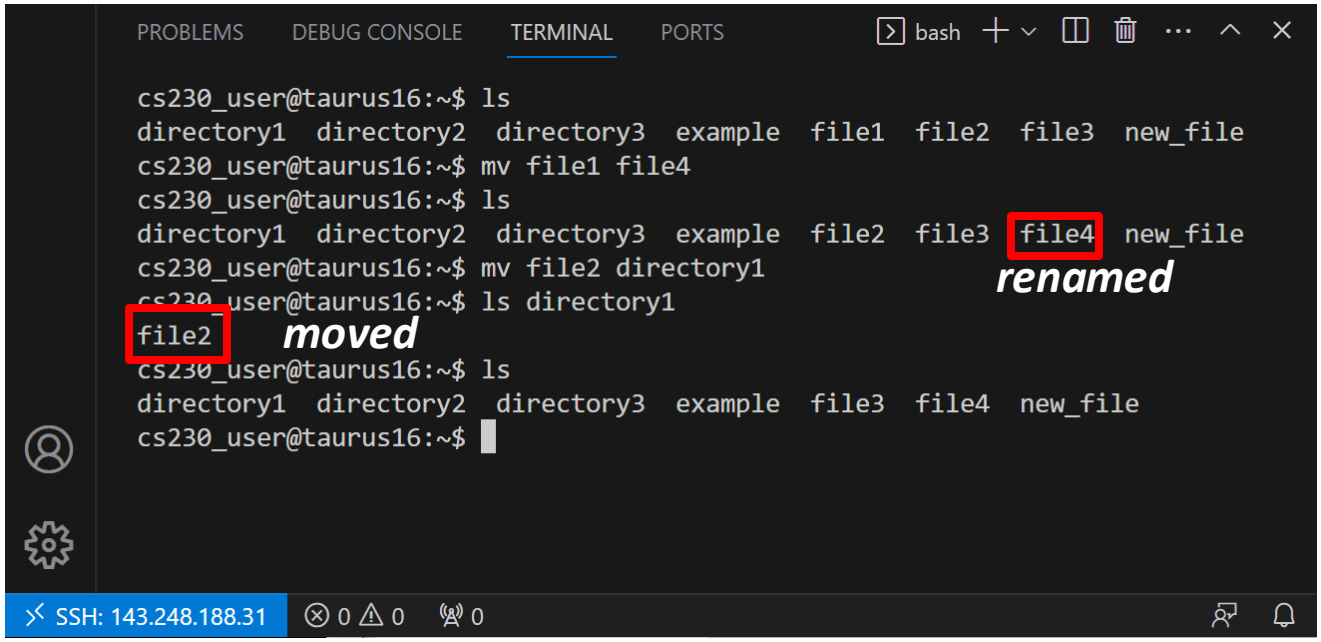
# Linux Commands

- cat
  - concatenate files and print on the standard output
  - > : redirects standard output to a file (overwrite)
  - >> : redirects standard output to a file (append)



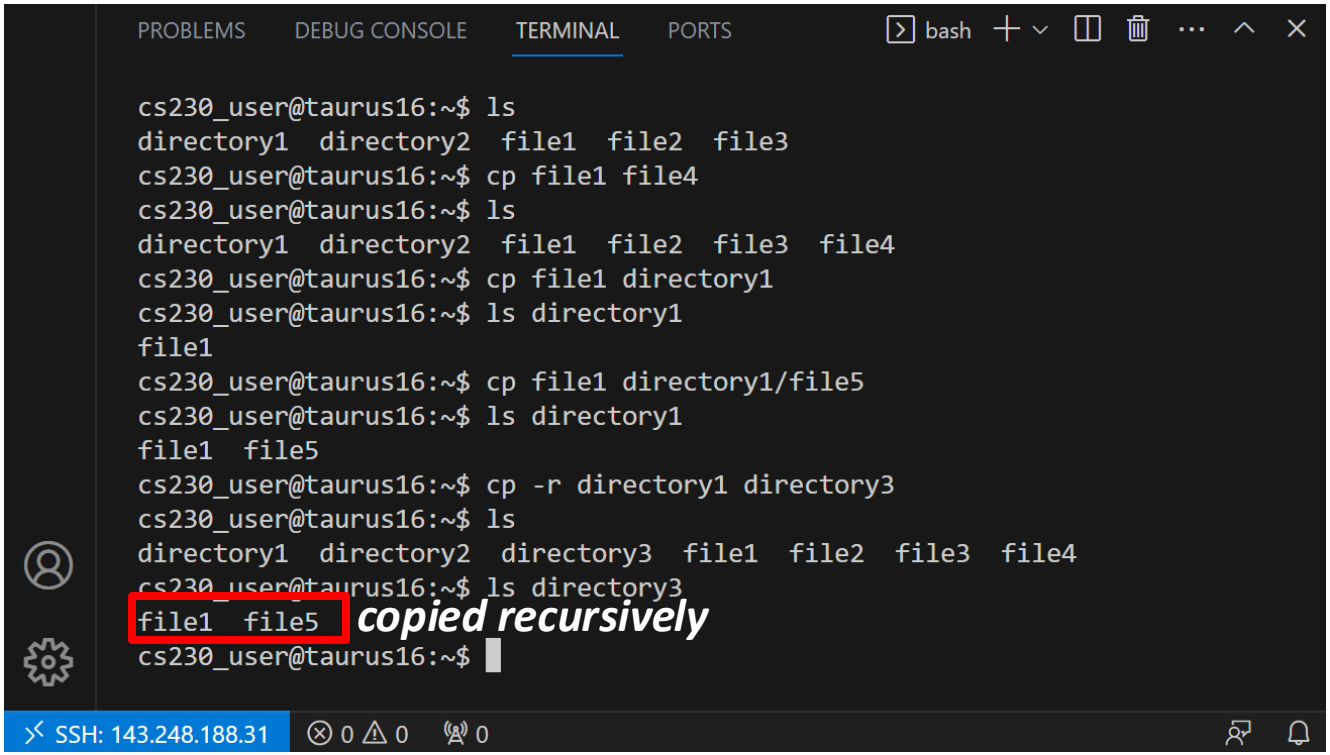
# Linux Commands

- mv [source] [destination]
  - Move or rename files or directories



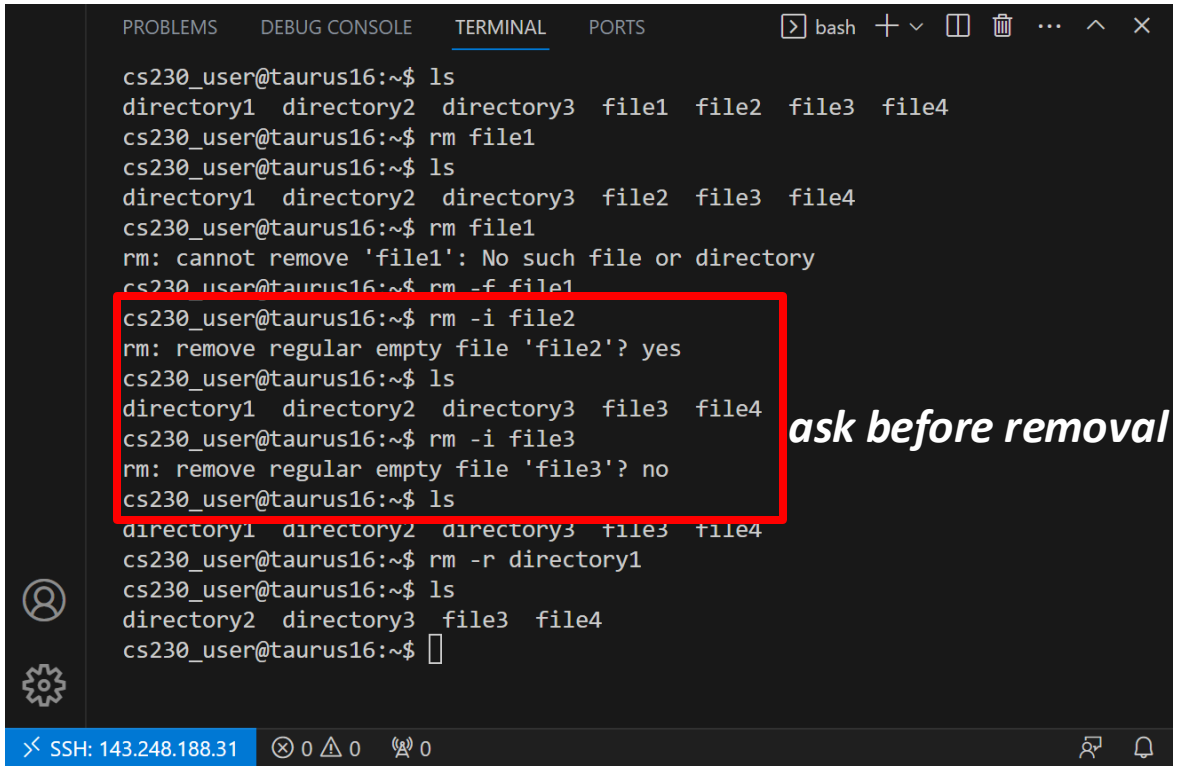
# Linux Commands

- cp [source] [destination]
  - Copy one or more files to specified location
  - -r : copy directories recursively



# Linux Commands

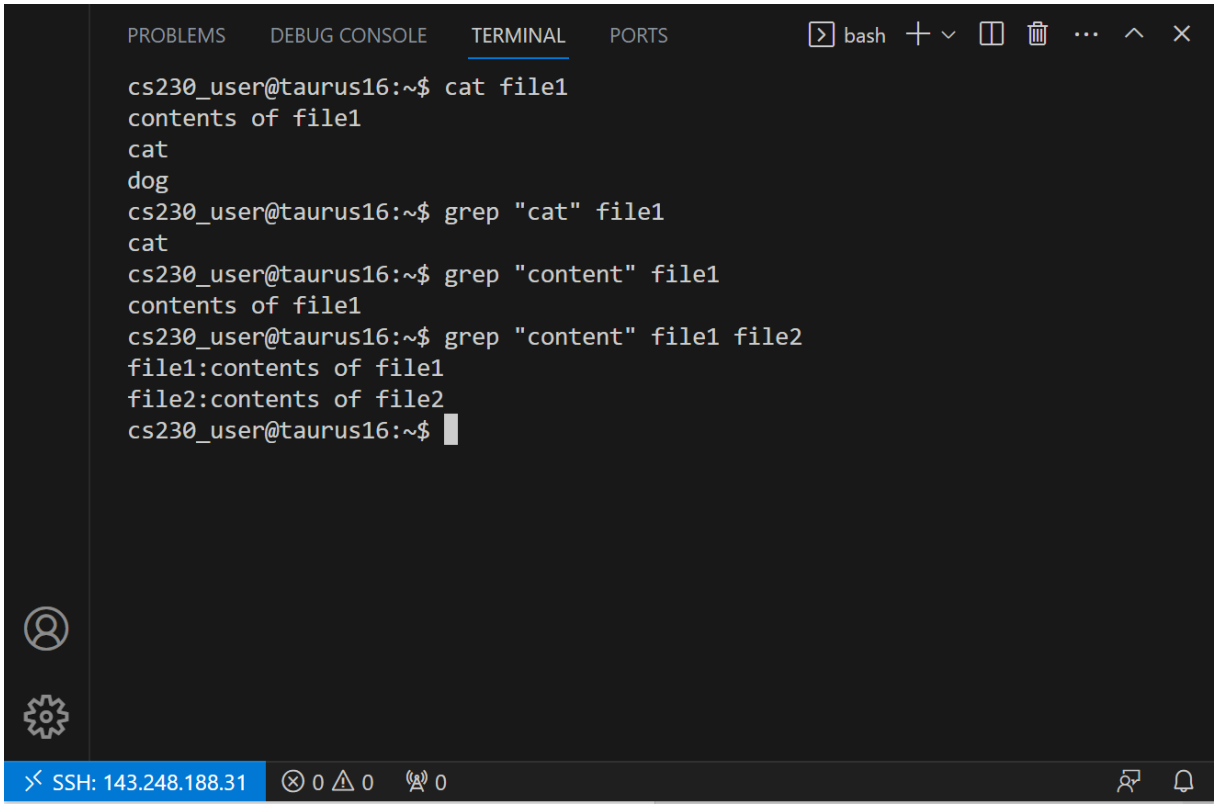
- rm [file/directory]
  - remove files or directories
  - -f : ignore nonexistent files, never ask
  - -i : ask whether really remove file or not
  - -r : Remove directories and their contents recursively




*ask before removal*

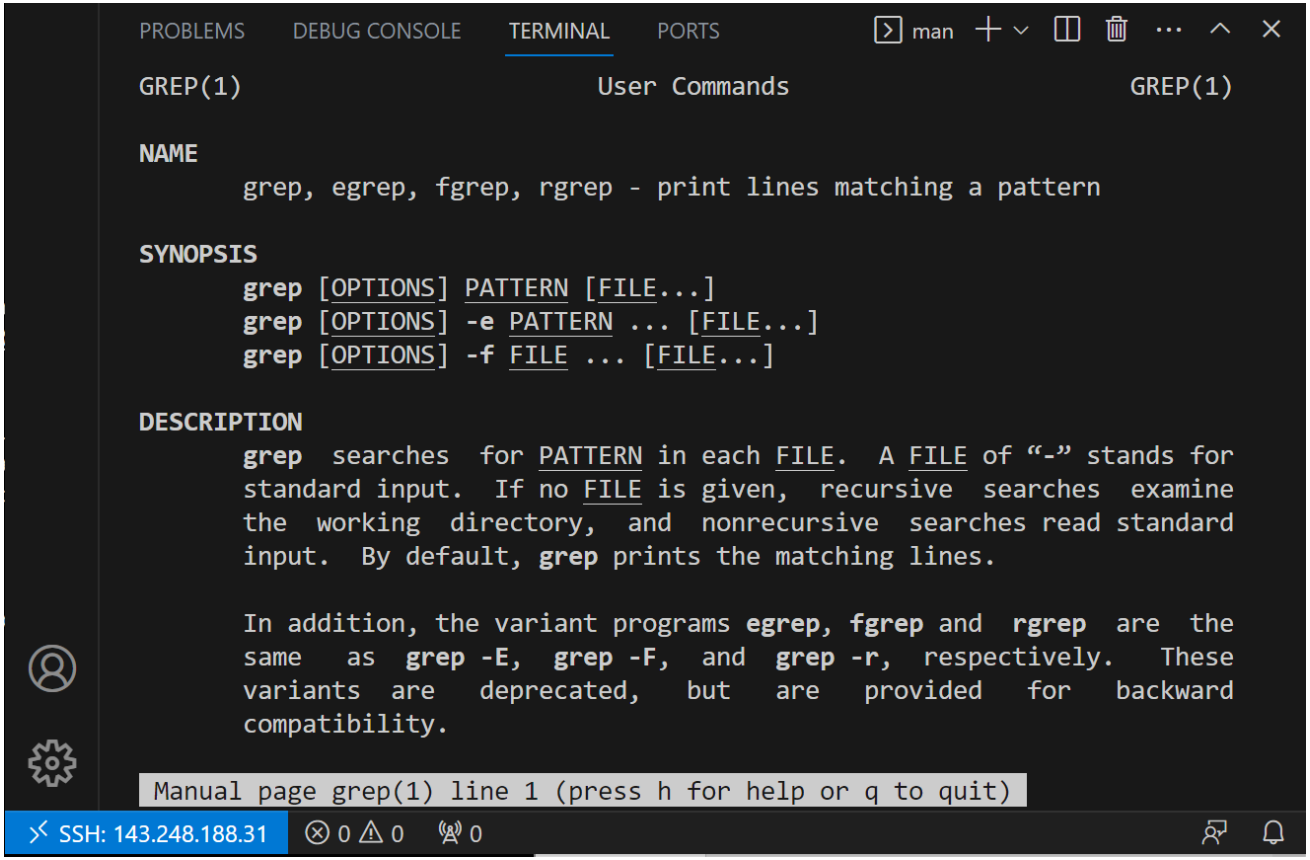
# Linux Commands

- grep [pattern] [files]
  - print lines matching a pattern



# Linux Commands

- man [command name]
  - an interface to the on-line reference manuals



# Linux Commands

- Running a command in background :
  $ [command] &

- Running many commands using a single line :
  $ [command1] ; [command2] ; [command3]

- Use the output of [command1] as an input of [command2] :
  $ [command1] | [command2]

- Save the printed output of command in a file :
  $ [command] > [file to save]

# gcc

- gcc
  - gcc hello.c -E -o hello.i ⟵ preprocessed file
  - gcc hello.c -S -o hello.s ⟵ assembly file obj
  - gcc hello.c -c -o hello.o ⟵ ect file

# make

- make
  - Make is a utility that automatically builds executable programs and libraries from source code by reading files called Makefiles (Wikipedia)

- Structure of Makefiles

```
target ... : prerequisites ...
        recipe
...
...
```

http://www.tecmint.com/a-brief-introduction-to-makefiles-in-open-source-software-development-with-gnu-make/

# make

- A sample Makefile

```
$cat Makefile
final: main.o end.o inter.o start.o
        gcc -o final main.o end.o inter.o start.o
main.o: main.c global.h
        gcc -c main.c
end.o: end.c local.h global.h
        gcc c -c end.c
inter.o: inter.c global.h
        gcc c -c inter.c  s
tart.o: start.c global.h
        gcc -c start.c
clean:
        rm -f main.o end.o inter.o start.o
```

http://www.tecmint.com/a-brief-introduction-to-makefiles-in-open-source-software-development-with-gnu-make/

# make

- A sample Makefile

```
$make
$ls | grep final
final
```

http://www.tecmint.com/a-brief-introduction-to-makefiles-in-open-source-software-development-with-gnu-make/

# Self-study material

- Linux : https://www.youtube.com/watch?v=CpTfQ-q6MPU