

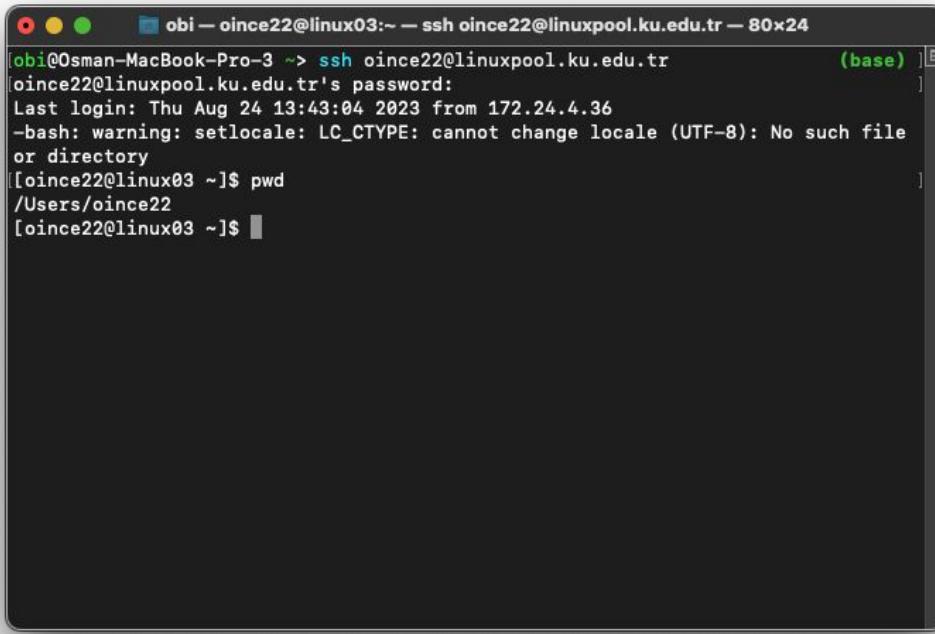
# Intro to Linux Shell

COMP201 - Lab1  
Spring 2024



KOÇ  
UNIVERSITY

# What is shell?



The terminal window shows the following session:

```
obi@Osman-MacBook-Pro-3 ~> ssh oince22@linuxpool.ku.edu.tr - 80x24
oince22@linuxpool.ku.edu.tr's password: (base)
Last login: Thu Aug 24 13:43:04 2023 from 172.24.4.36
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
[oince22@linux03 ~]$ pwd
/Users/oince22
[oince22@linux03 ~]$
```

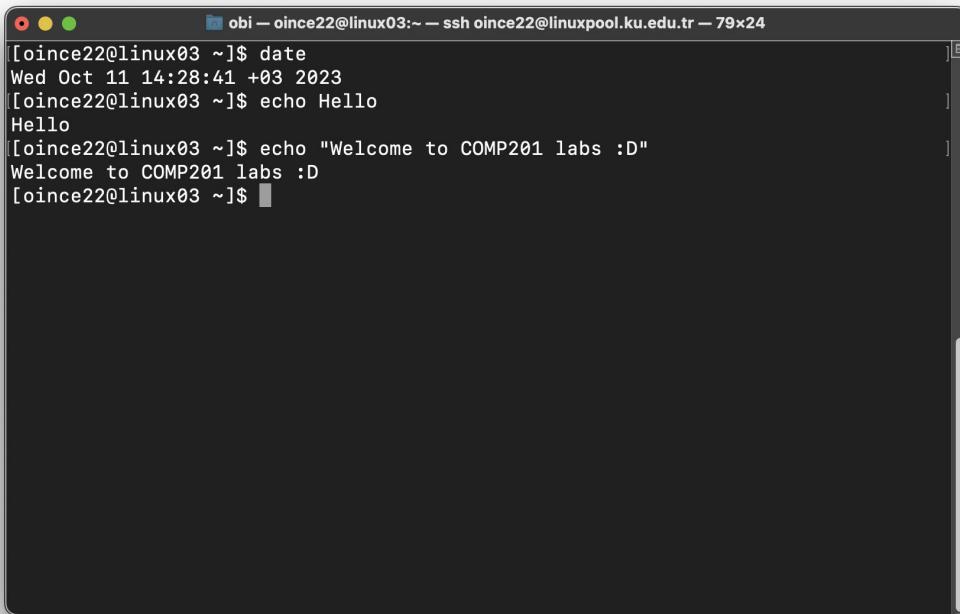
- **Linux shell** is the interface between you and OS that controls hardware.
- The most commonly used shell is called BASH – Bourne Again Shell
  - The default shell in Linuxpool
- `username@hostname:curr_dir$`
  - username: oince22
  - hostname: linux03
  - curr\_dir: /Users/oince22

# How to connect?

`ssh USERNAME@linuxpool.ku.edu.tr`

1. Type your password when prompted.
2. If you see a warning about SSH host keys, click or enter “yes.”

# Executing system programs

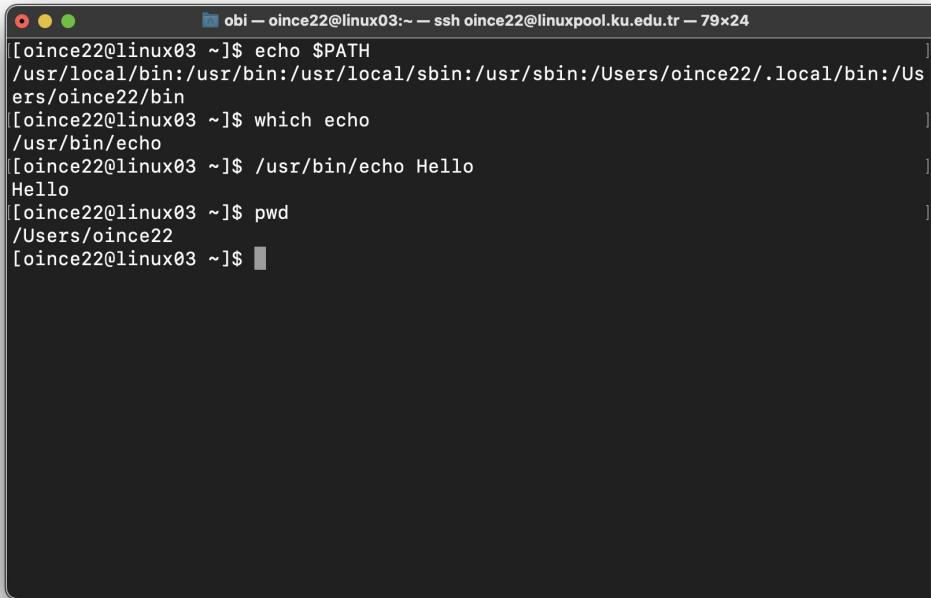


A screenshot of a terminal window titled "obi" showing a Linux command-line session. The session starts with the user running the "date" command, which outputs the current date and time. Then, the user runs the "echo" command with a string of text. The terminal window has a dark background and light-colored text.

```
[oince22@linux03 ~]$ date
Wed Oct 11 14:28:41 +03 2023
[oince22@linux03 ~]$ echo Hello
Hello
[oince22@linux03 ~]$ echo "Welcome to COMP201 labs :D"
Welcome to COMP201 labs :D
[oince22@linux03 ~]$
```

- Execute programs
- **date**
  - This program prints current date and time
- **echo**
  - This program prints the input argument
  - Put quotation marks around the string if the string has more than one word

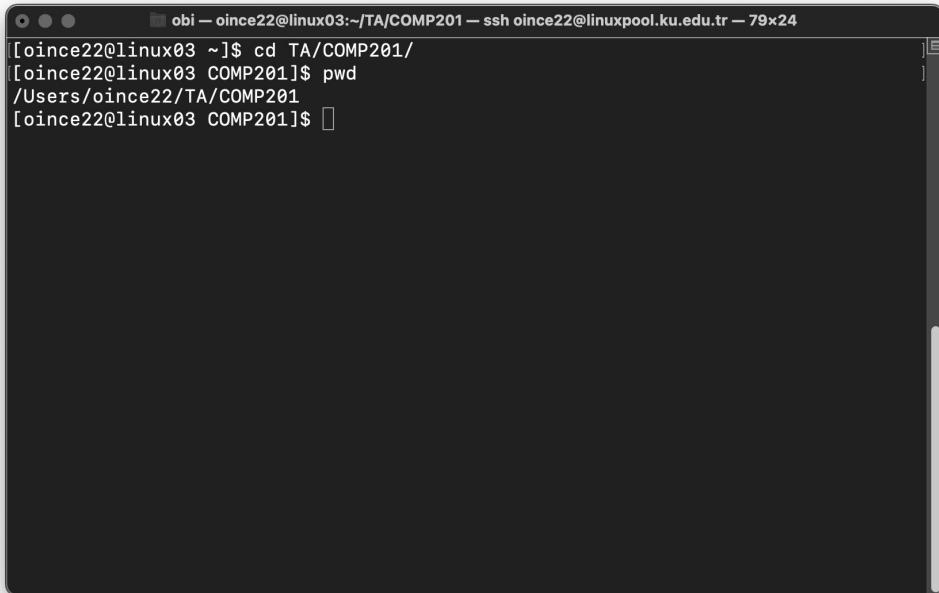
# Path and \$PATH



```
[oince22@linux03 ~]$ echo $PATH  
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/Users/oince22/.local/bin:/Us  
ers/oince22/bin  
[oince22@linux03 ~]$ which echo  
/usr/bin/echo  
[oince22@linux03 ~]$ /usr/bin/echo Hello  
Hello  
[oince22@linux03 ~]$ pwd  
/Users/oince22  
[oince22@linux03 ~]$
```

- \$PATH
  - A variable that contains addresses where system look for programs to execute
- which
  - Prints which file is being executed given an input program name
- pwd
  - This program prints current working directory
  - Stands for “print working directory”

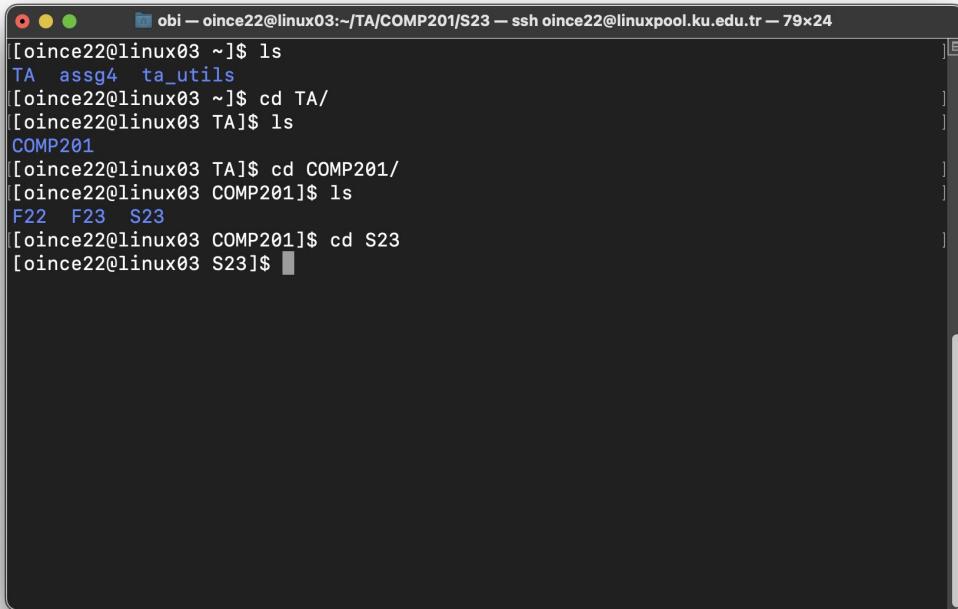
# Path



```
[oince22@linux03 ~]$ cd TA/COMP201/
[oince22@linux03 COMP201]$ pwd
/Users/oince22/TA/COMP201
[oince22@linux03 COMP201]$
```

- **cd**
  - Changes the working directory
  - .. is the parent directory
  - . is the current directory
  - Tilda (~) is the /Users/<username> directory
    - This is true in Linuxpool
    - May be different in another machine
- **Absolute vs relative path**
  - Relative: TA/COMP201 from ~ (home)
  - Absolute: /Users/oince22/TA/COMP201

# Listing files and directories



A screenshot of a terminal window titled "obi" showing a Linux command-line session. The session starts with the user navigating through their home directory (~) and listing files ("ls"). The user then changes directory to "TA/" and lists its contents, which include "assg4" and "ta\_utils". Next, the user changes to "COMP201/" and lists its contents, which include "F22", "F23", and "S23". Finally, the user changes to "S23" and lists its contents. The terminal window has a dark background and light-colored text.

```
[oince22@linux03 ~]$ ls
TA assg4 ta_utils
[oince22@linux03 ~]$ cd TA/
[oince22@linux03 TA]$ ls
COMP201
[oince22@linux03 TA]$ cd COMP201/
[oince22@linux03 COMP201]$ ls
F22 F23 S23
[oince22@linux03 COMP201]$ cd S23
[oince22@linux03 S23]$
```

- **ls**
  - Prints files and directories under current working directory

# Flags with Commands in Linux

- Many Linux commands have **flags** that can be used to modify their behavior.
- **Flags** are usually preceded by **one** or **two** dashes, followed by a letter or a word.
- **Flags** can be used to:
  - Control the output of a command
  - Specify a file or directory to work with
  - Modify the command's behavior in other ways

# Flags with Commands in Linux

- Let's look at an example: **ls** command.
- By default, it lists contents of the current folder.
- But we can use **flags** to modify its behavior.
- For example,
  - l** flag to list the contents of the directory line-by-line, long-format including additional info about file permissions, owner, and size.
  - a** flag to display all files, including hidden files (usually not displayed by default).
- To use both flags together, type **ls -la**
  - Combine as many as you want!

```
[oince22@linux03 S23]$ ls
Assignment_2  Lab2  Lab3  Lab6
[oince22@linux03 S23]$ ls -l
total 28
drwxr-xr-x  4 oince22 domainusers  4096 Oct 10 13:55 Assignment_2
drwxr-xr-x  7 oince22 domainusers  4096 Mar 16 2023 Lab2
drwxr-xr-x  9 oince22 domainusers  4096 Oct 10 13:52 Lab3
drwxr-xr-x  5 oince22 domainusers 16384 May 22 16:12 Lab6
[oince22@linux03 S23]$ ls -a
.  ..  .hidden_lab  Assignment_2  Lab2  Lab3  Lab6
[oince22@linux03 S23]$ ls -al
total 40
drwxr-xr-x  7 oince22 domainusers  4096 Oct 11 15:24 .
drwxr-xr-x  5 oince22 domainusers  4096 Oct 11 14:40 ..
drwxr-xr-x  2 oince22 domainusers  4096 Oct 11 15:24 .hidden_lab
drwxr-xr-x  4 oince22 domainusers  4096 Oct 10 13:55 Assignment_2
drwxr-xr-x  7 oince22 domainusers  4096 Mar 16 2023 Lab2
drwxr-xr-x  9 oince22 domainusers  4096 Oct 10 13:52 Lab3
drwxr-xr-x  5 oince22 domainusers 16384 May 22 16:12 Lab6
[oince22@linux03 S23]$
```

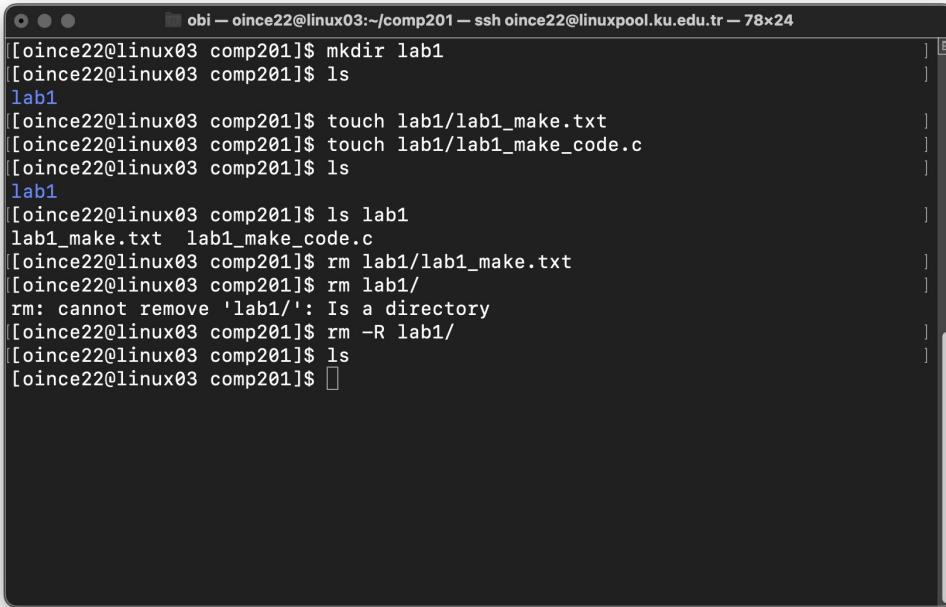
To learn more about the flags available for a command, type `man command`  
To learn details about the `ls` command and its flags → `man ls`

# Listing files and directories

```
● ● ● obi - oince22@linux03:~/TA/COMP201/S23/Lab2/archive/lab2-material/lab2-examples — ssh oince22@linuxpo...
[oince22@linux03 lab2-examples]$ ls
bits.c btest.c decl.c fshow.c tests.c
[oince22@linux03 lab2-examples]$ ls -lS
total 36
-rw-r--r-- 1 oince22 domainusers 15752 Mar 16 2023 btest.c
-rw-r--r-- 1 oince22 domainusers 7565 Mar 16 2023 bits.c
-rw-r--r-- 1 oince22 domainusers 3009 Mar 16 2023 fshow.c
-rw-r--r-- 1 oince22 domainusers 2795 Mar 16 2023 tests.c
-rw-r--r-- 1 oince22 domainusers 2662 Mar 16 2023 decl.c
[oince22@linux03 lab2-examples]$ ls -lSr
total 36K
-rw-r--r-- 1 oince22 domainusers 2.6K Mar 16 2023 decl.c
-rw-r--r-- 1 oince22 domainusers 2.8K Mar 16 2023 tests.c
-rw-r--r-- 1 oince22 domainusers 3.0K Mar 16 2023 fshow.c
-rw-r--r-- 1 oince22 domainusers 7.4K Mar 16 2023 bits.c
-rw-r--r-- 1 oince22 domainusers 16K Mar 16 2023 btest.c
[oince22@linux03 lab2-examples]$
```

- You can use **-S** flag to display files sorted by their sizes, and **-r** option for reverse sorting.
- You can use **-h** flag to display file sizes in a human-readable format.

# Making/Removing folders and files



The screenshot shows a terminal window titled 'obi' with the command line 'oince22@linux03 ~/comp201 - ssh oince22@linuxpool.ku.edu.tr - 78x24'. The terminal history is as follows:

```
[oince22@linux03 comp201]$ mkdir lab1
[oince22@linux03 comp201]$ ls
lab1
[oince22@linux03 comp201]$ touch lab1/lab1_make.txt
[oince22@linux03 comp201]$ touch lab1/lab1_make_code.c
[oince22@linux03 comp201]$ ls
lab1
[oince22@linux03 comp201]$ ls lab1
lab1_make.txt  lab1_make_code.c
[oince22@linux03 comp201]$ rm lab1/lab1_make.txt
[oince22@linux03 comp201]$ rm lab1/
rm: cannot remove 'lab1/': Is a directory
[oince22@linux03 comp201]$ rm -R lab1/
[oince22@linux03 comp201]$ ls
[oince22@linux03 comp201]$
```

- **mkdir <folder\_name>**
  - Makes a new directory in the given working directory with the given "folder\_name".
- **touch**
  - Creates a file with desired extension and name
- **rm**
  - Removes a file or folder.
  - For removing folders you need to use -R option

# Chmod

- Chmod (short for "change mode") is a command in Linux that allows users to change the read, write, and execute permissions of files and directories.
- The syntax for chmod is as follows:
  - `chmod [options] MODE FILENAME`
- The mode is a combination of the letters "**r**" (read), "**w**" (write), and "**x**" (execute).
- Permissions can be granted to three different user groups:
  - The file owner
  - The group owner
  - All users

# File Permission in Linux

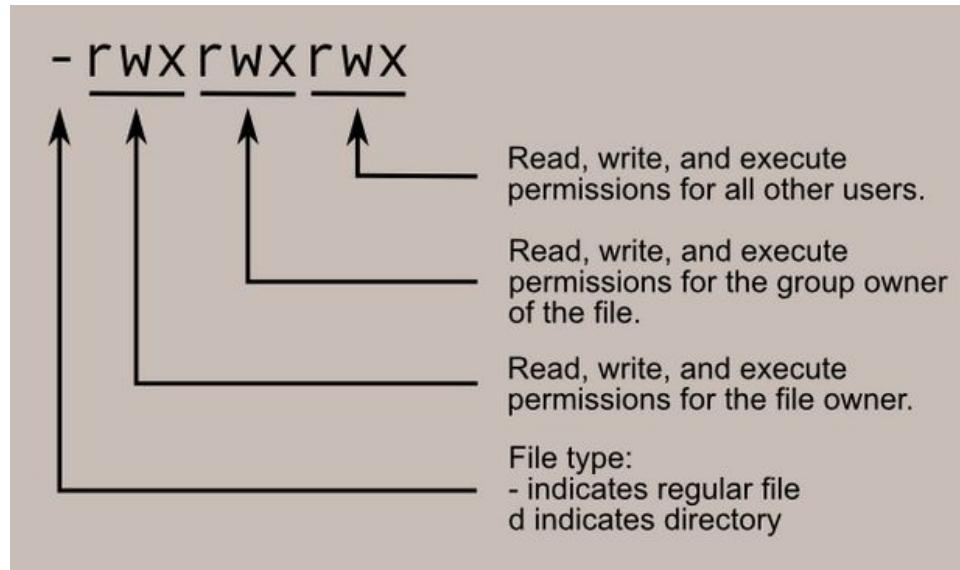


Image source: [http://linuxcommand.org/lc3\\_lts0090.php](http://linuxcommand.org/lc3_lts0090.php)

# File Permission in Linux

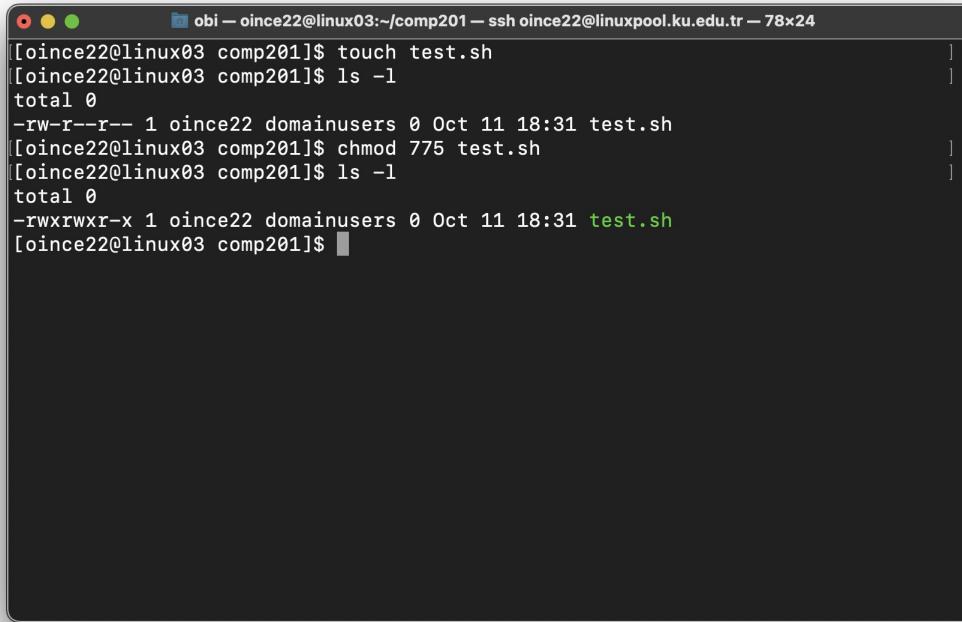
```
rwx rwx rwx = 111 111 111  
rw- rw- rw- = 110 110 110  
rwx --- --- = 111 000 000
```

and so on...

```
rwx = 111 in binary = 7  
rw- = 110 in binary = 6  
r-x = 101 in binary = 5  
r-- = 100 in binary = 4
```

Image source: [http://linuxcommand.org/lc3\\_lts0090.php](http://linuxcommand.org/lc3_lts0090.php)

# File Permission in Linux

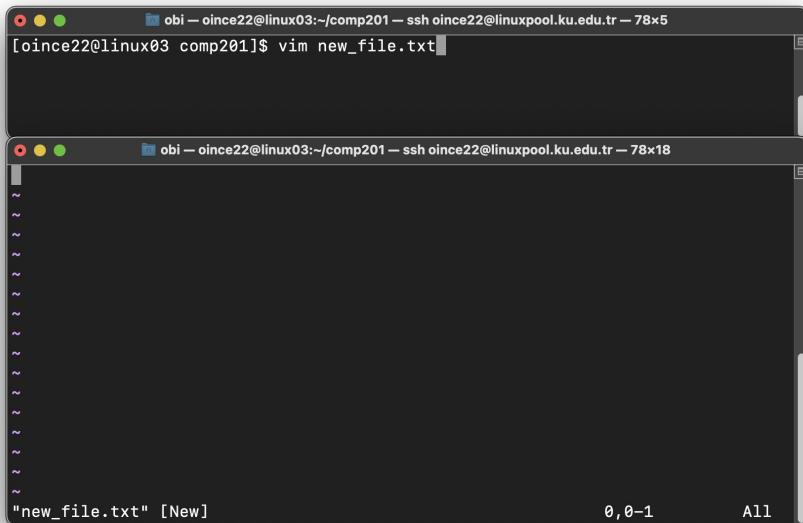


A screenshot of a terminal window titled "obi" showing a Linux command-line session. The session starts with creating a file "test.sh", then listing it with "ls -l" to show it has read and write permissions for the owner. It then uses "chmod 775 test.sh" to change the permissions to "rwxrwxr-x", which is explicitly highlighted in green. Finally, it lists the file again to show the new permissions.

```
[oince22@linux03 comp201]$ touch test.sh
[oince22@linux03 comp201]$ ls -l
total 0
-rw-r--r-- 1 oince22 domainusers 0 Oct 11 18:31 test.sh
[oince22@linux03 comp201]$ chmod 775 test.sh
[oince22@linux03 comp201]$ ls -l
total 0
-rwxrwxr-x 1 oince22 domainusers 0 Oct 11 18:31 test.sh
[oince22@linux03 comp201]$
```

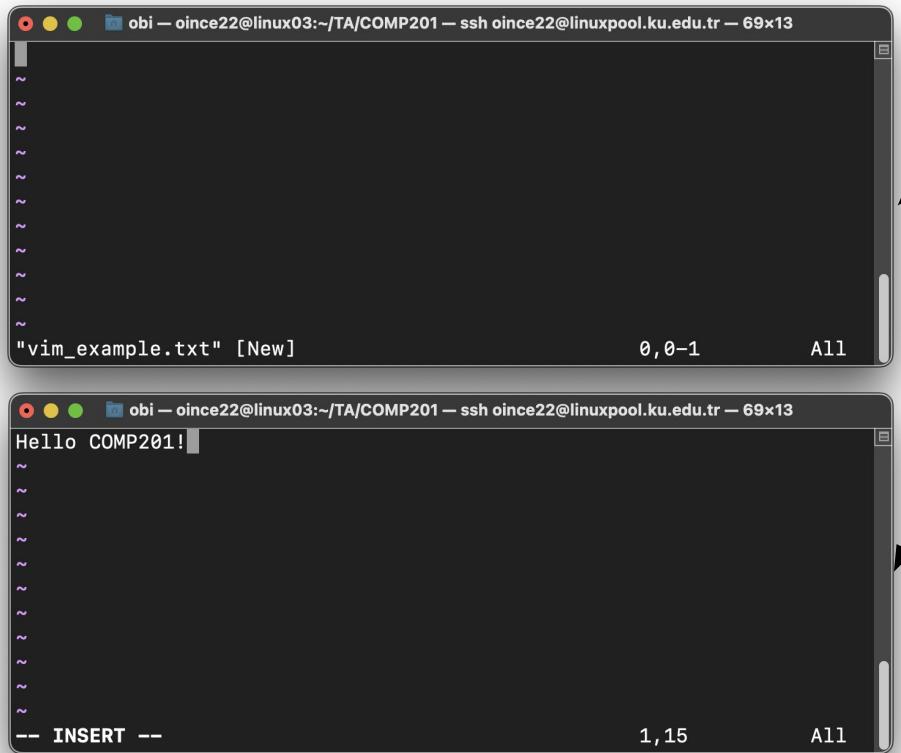
Initially, `test.sh` cannot be executed, to grant `-rwx rwx r-x` permission to `test.sh` file execute `chmod 775 test.sh` command.

# What is Vim?



- **Vim** is the default text editor in the UNIX operating system.
  - Using **vim**, we can create a new file, read, and edit an existing file.
  - To open **vim**, type **vim** or **vim FILENAME**. If the file FILENAME doesn't exist, it will be created when you save it.

# Operation Modes in Vim



```
"vim_example.txt" [New]
```

```
Hello COMP201!
```

```
-- INSERT --
```

## Normal mode

- The default mode in **vim**.
- Every character you type is interpreted as a command.

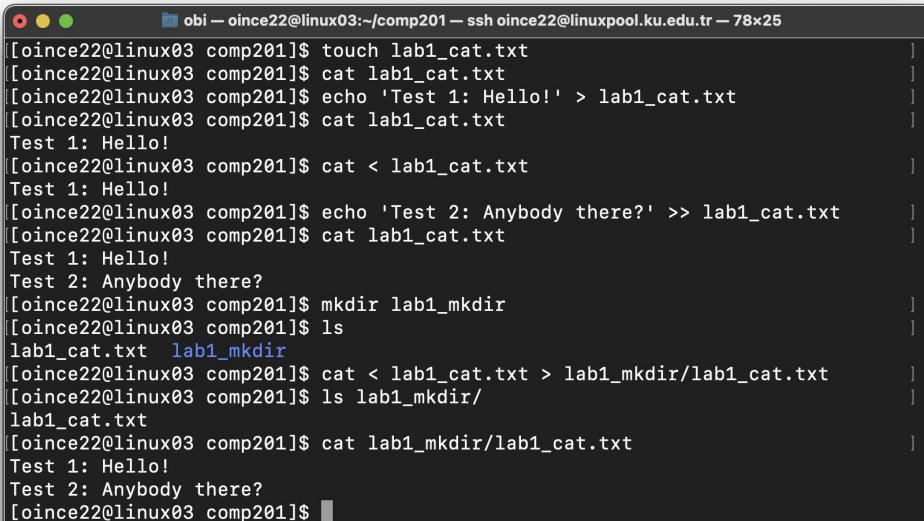
## Insert mode

- To switch from normal mode to insert mode, type **i** in the normal mode.
- Every character you type is put to the file.
- To switch back to normal mode, press **<Esc>**

# Operation Modes in Vim

- **Exit with saving**
    - To save and exit a file, go to the Normal mode by pressing `<Esc>` then type `:wq`
  - **Exit without saving**
    - To exit from a file without saving it, go to the Normal mode by pressing `<Esc>` then type `:q!`
  - After typing `:wq` or `:q!`, press `<Enter>`

# Redirection



```
[oince22@linux03 comp201]$ touch lab1_cat.txt
[oince22@linux03 comp201]$ cat lab1_cat.txt
[oince22@linux03 comp201]$ echo 'Test 1: Hello!' > lab1_cat.txt
[oince22@linux03 comp201]$ cat lab1_cat.txt
Test 1: Hello!
[oince22@linux03 comp201]$ cat < lab1_cat.txt
Test 1: Hello!
[oince22@linux03 comp201]$ echo 'Test 2: Anybody there?' >> lab1_cat.txt
[oince22@linux03 comp201]$ cat lab1_cat.txt
Test 1: Hello!
Test 2: Anybody there?
[oince22@linux03 comp201]$ mkdir lab1_mkdir
[oince22@linux03 comp201]$ ls
lab1_cat.txt  lab1_mkdir
[oince22@linux03 comp201]$ cat < lab1_cat.txt > lab1_mkdir/lab1_cat.txt
[oince22@linux03 comp201]$ ls lab1_mkdir/
lab1_cat.txt
[oince22@linux03 comp201]$ cat lab1_mkdir/lab1_cat.txt
Test 1: Hello!
Test 2: Anybody there?
[oince22@linux03 comp201]$
```

- **cat**
  - Print the content of the given file
- **< file and > file**
  - You can write the input and output of a program to a file
  - “>> file” appends to end of file

# Piping

The screenshot shows a terminal window with the following session:

```
[oince22@linux03 comp201]$ cat myfile.txt
BaNanA
apple
BaNanA
orange
Apple
[oince22@linux03 comp201]$ grep apple myfile.txt
apple
[oince22@linux03 comp201]$ grep -i apple myfile.txt
apple
Apple
[oince22@linux03 comp201]$ grep -i a myfile.txt
BaNanA
apple
BaNanA
orange
Apple
[oince22@linux03 comp201]$
```

- Pipe character is |
  - Connects output of a program to input of another one
- grep
  - Searches for a particular information
  - By default it is case sensitive
- Try grep --help and find what does -i option do

# SCP

- **SCP** is a tool in Linux used to transfer files between hosts over a network.
- The syntax for SCP is as follows:
  - `scp [OPTIONS] SOURCE DESTINATION`
- **-r** flag is used to copy directories, stands for **recursive**

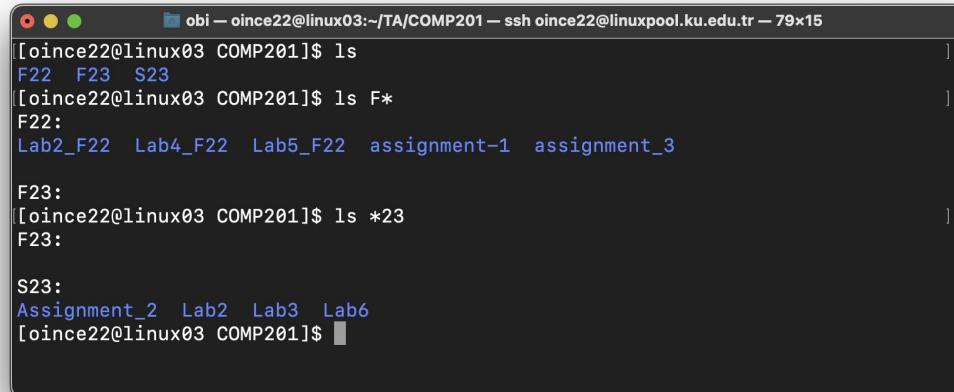
# SCP

- From local machine to Linuxpool:
  - (on local machine): `scp -r FILENAME USERNAME@linuxpool.ku.edu.tr:`
- From Linuxpool to local machine:
  - (on local machine): `scp -r USERNAME@linuxpool.ku.edu.tr:PATH/TO/FILE .`

**Do not forget the colon!!**

# Useful Commands

- **clear**: Clearing the contents of the terminal screen
- **history**: Searching for previously executed commands
- **Tab key**: auto-completion
- **\* (asterisk)**: Used as a wildcard to represent any combination of characters in a command or filename



The screenshot shows a terminal window with the following session:

```
[oince22@linux03 COMP201]$ ls
F22 F23 S23
[oince22@linux03 COMP201]$ ls F*
F22:
Lab2_F22 Lab4_F22 Lab5_F22 assignment-1 assignment_3

F23:
[oince22@linux03 COMP201]$ ls *23
F23:

S23:
Assignment_2 Lab2 Lab3 Lab6
[oince22@linux03 COMP201]$
```

# Other Resources

- MIT MS [The Shell](#)
- Stanford [CS107 Unix videos](#) 1-15, 24, 25
- [UNIX Tutorial for Beginners](#)