

# Linux System Guide for CS230 Students

Original Version by Taekyung Heo  
Updated by Soojin Hwang

# 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

---

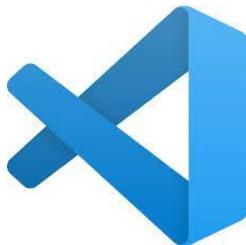
- How can I use a Linux machine?
  - You have to establish a connection to it.
- How can I connect to a machine?
  - Use ssh clients like PuTTY or VSCode.



PuTTY

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

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

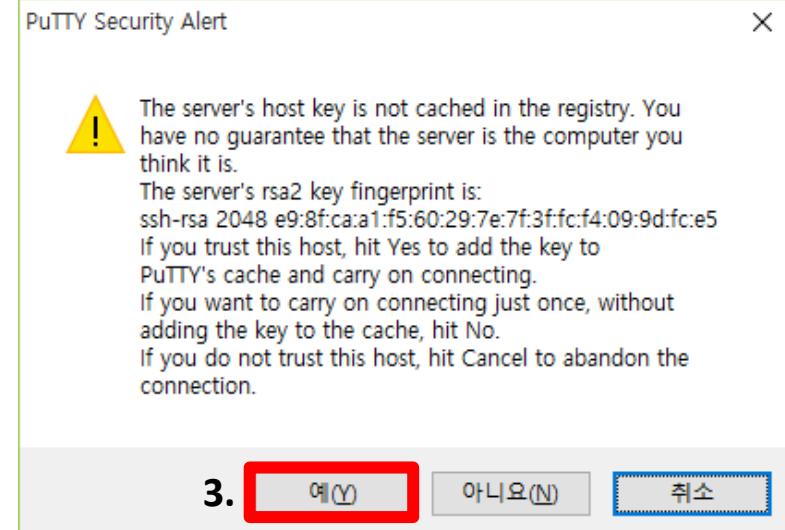
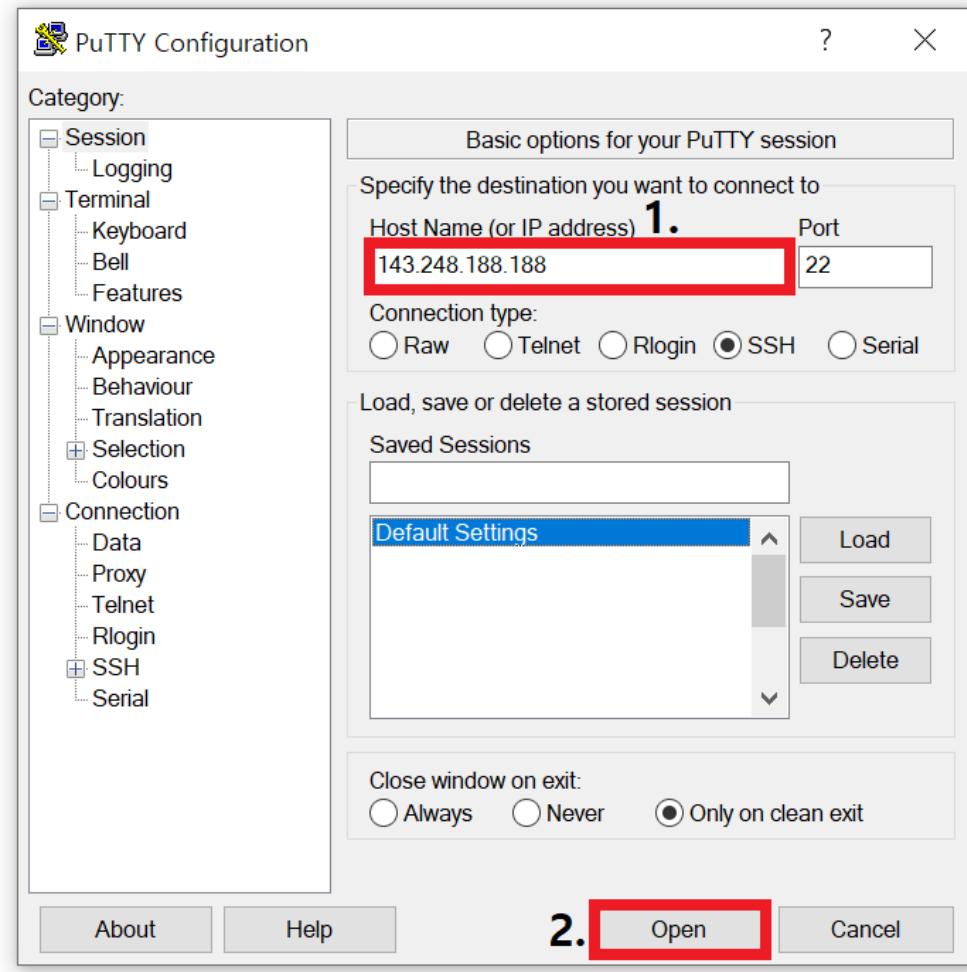


VSCode

## **VSCode (Visual Studio Code)**

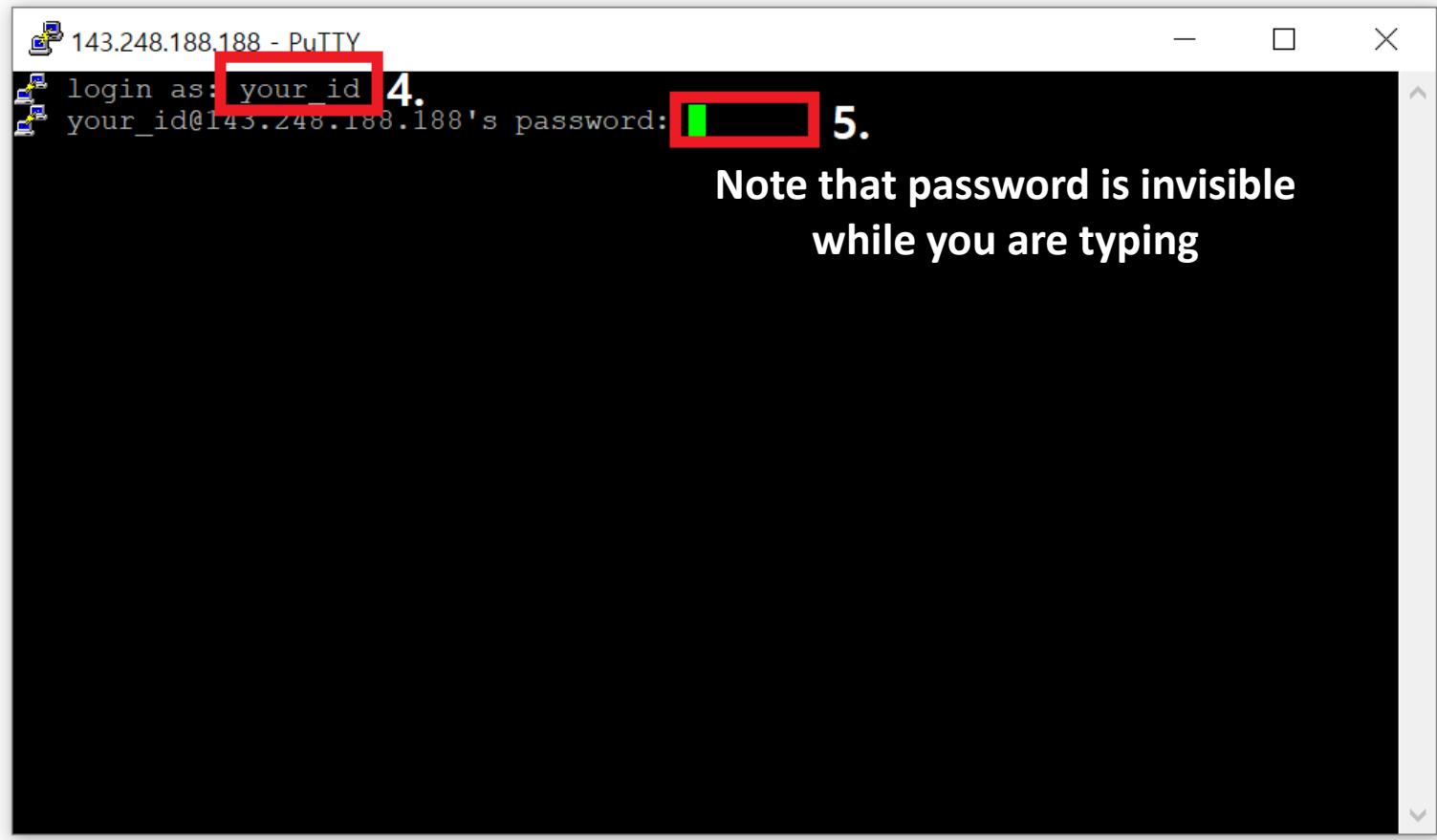
- Needs installation & setup
- User-friendly interface

# Connection Example: PuTTY



# Connection Example: PuTTY

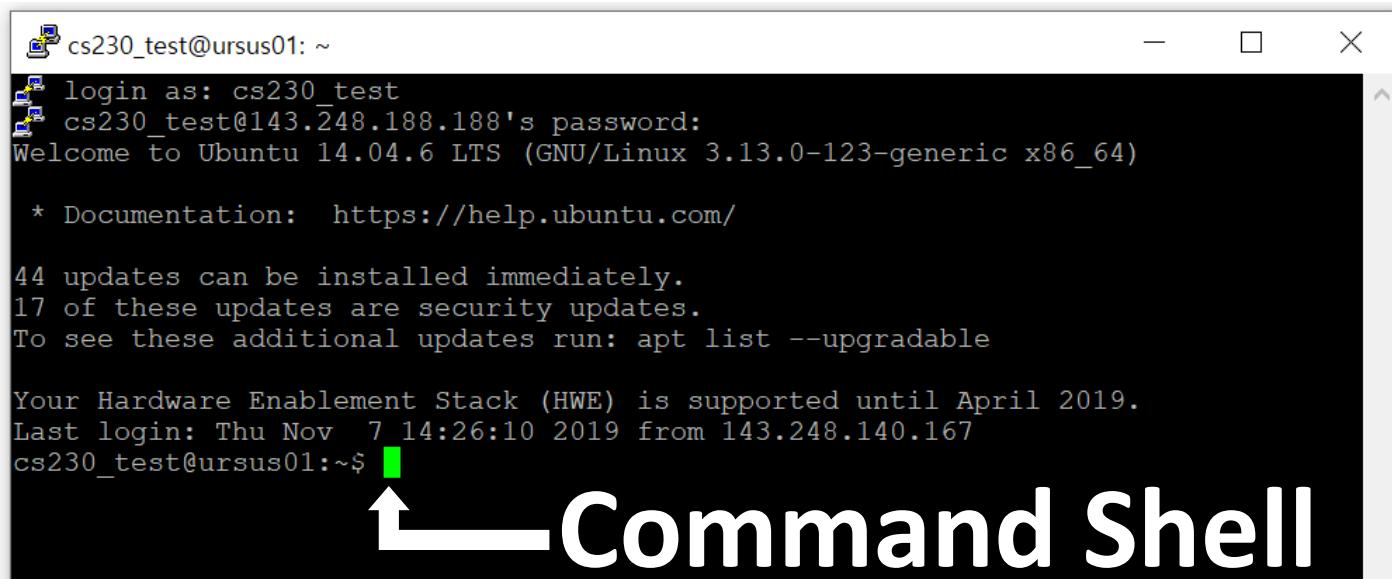
---



# Linux Shell

---

## Connected!



A screenshot of a terminal window titled "cs230\_test@ursus01: ~". The window shows a standard Ubuntu 14.04 LTS login screen with a black background and white text. It includes the user's name, password prompt, welcome message, documentation link, update information, hardware support details, and a last login timestamp. A green cursor arrow points upwards from the bottom of the terminal window towards the word "Command Shell".

```
cs230_test@ursus01: ~
login as: cs230_test
[...] cs230_test@143.248.188.188's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-123-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

44 updates can be installed immediately.
17 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2019.
Last login: Thu Nov  7 14:26:10 2019 from 143.248.140.167
cs230_test@ursus01:~$
```

↑—Command Shell

## Shell?

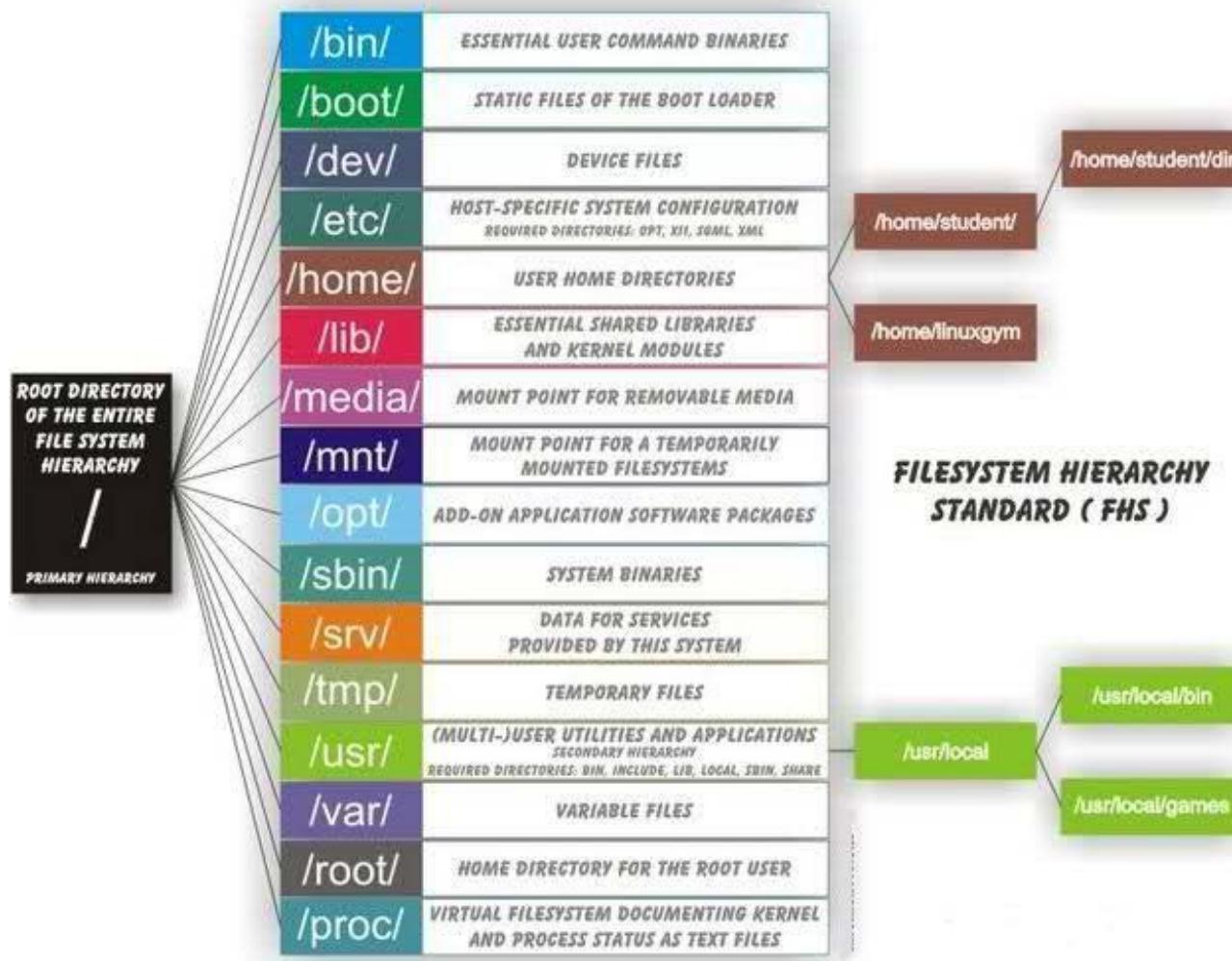
- a command line interface between a user and Linux machine

# Linux Directory

---

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

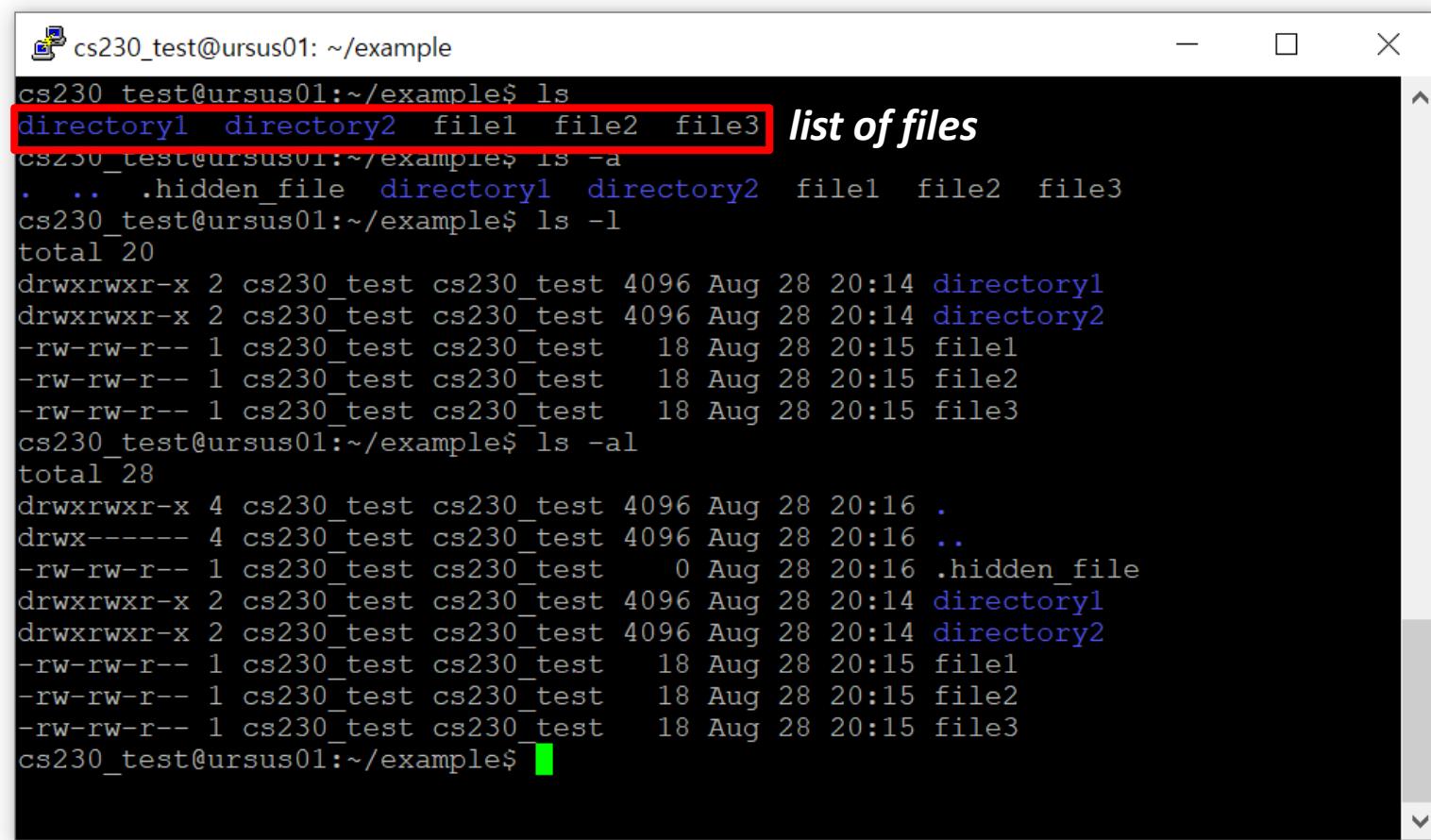
# Linux Directory Structure



# Linux Commands

---

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

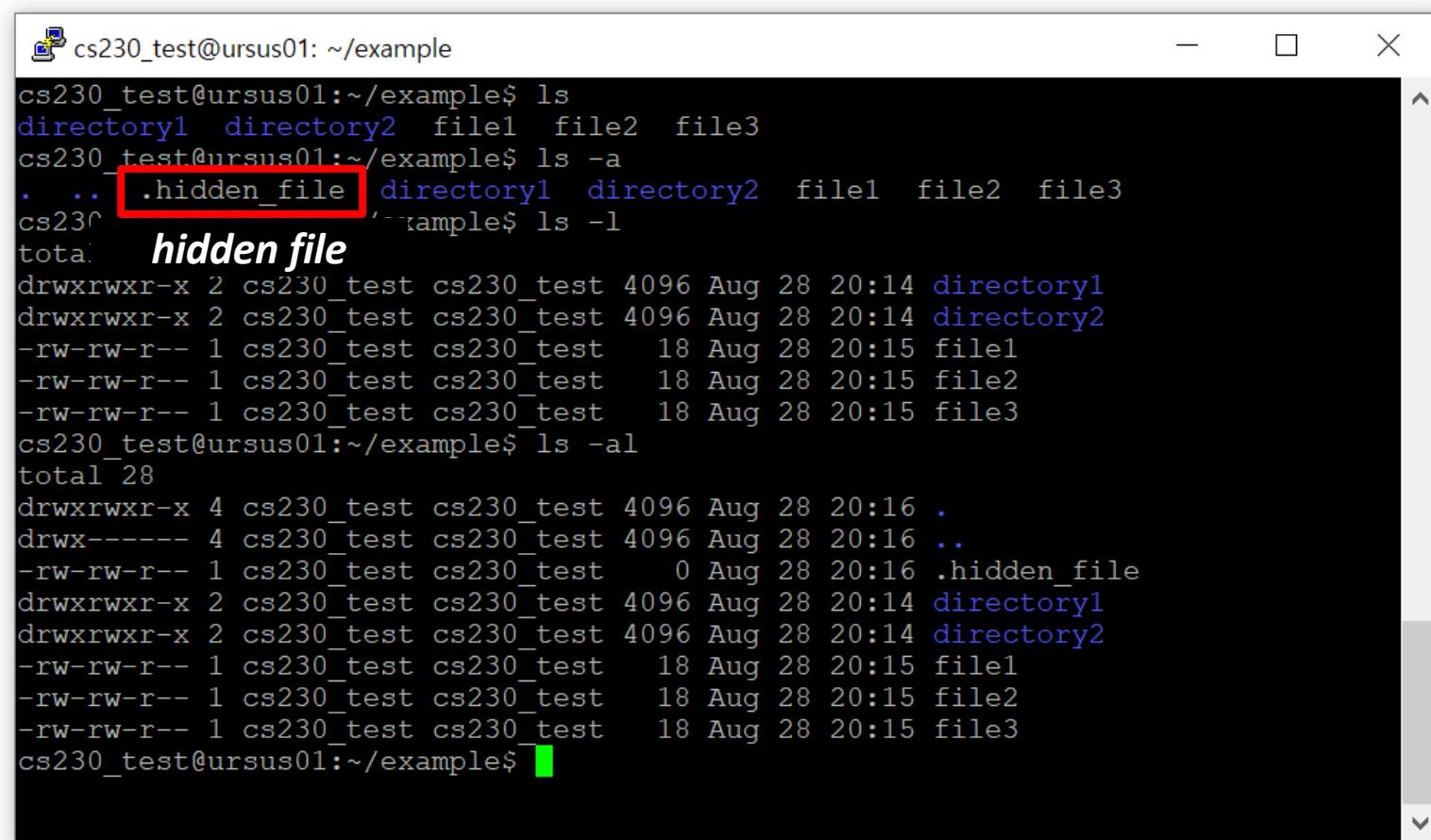


A screenshot of a terminal window titled "cs230\_test@ursus01: ~/example". The window contains a command-line session demonstrating the ls command. The user runs "ls" which lists files "directory1", "directory2", "file1", "file2", and "file3". A red box highlights this output. The user then runs "ls -a" which lists all files including hidden files ".hidden\_file", "..", and ".". Finally, the user runs "ls -l" which provides a detailed listing of file permissions, ownership, and timestamps for all files. The word "list of files" is overlaid in bold italic text on the highlighted ls command output.

```
cs230 test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3 list of files
cs230 test@ursus01:~/example$ ls -a
. .. .hidden_file directory1 directory2 file1 file2 file3
cs230 test@ursus01:~/example$ ls -l
total 20
drwxrwxr-x 2 cs230 test cs230 test 4096 Aug 28 20:14 directory1
drwxrwxr-x 2 cs230 test cs230 test 4096 Aug 28 20:14 directory2
-rw-rw-r-- 1 cs230 test cs230 test 18 Aug 28 20:15 file1
-rw-rw-r-- 1 cs230 test cs230 test 18 Aug 28 20:15 file2
-rw-rw-r-- 1 cs230 test cs230 test 18 Aug 28 20:15 file3
cs230 test@ursus01:~/example$ ls -al
total 28
drwxrwxr-x 4 cs230 test cs230 test 4096 Aug 28 20:16 .
drwxrwxr-x 4 cs230 test cs230 test 4096 Aug 28 20:16 ..
-rw-rw-r-- 1 cs230 test cs230 test 0 Aug 28 20:16 .hidden_file
drwxrwxr-x 2 cs230 test cs230 test 4096 Aug 28 20:14 directory1
drwxrwxr-x 2 cs230 test cs230 test 4096 Aug 28 20:14 directory2
-rw-rw-r-- 1 cs230 test cs230 test 18 Aug 28 20:15 file1
-rw-rw-r-- 1 cs230 test cs230 test 18 Aug 28 20:15 file2
-rw-rw-r-- 1 cs230 test cs230 test 18 Aug 28 20:15 file3
cs230 test@ursus01:~/example$
```

# Linux Commands

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



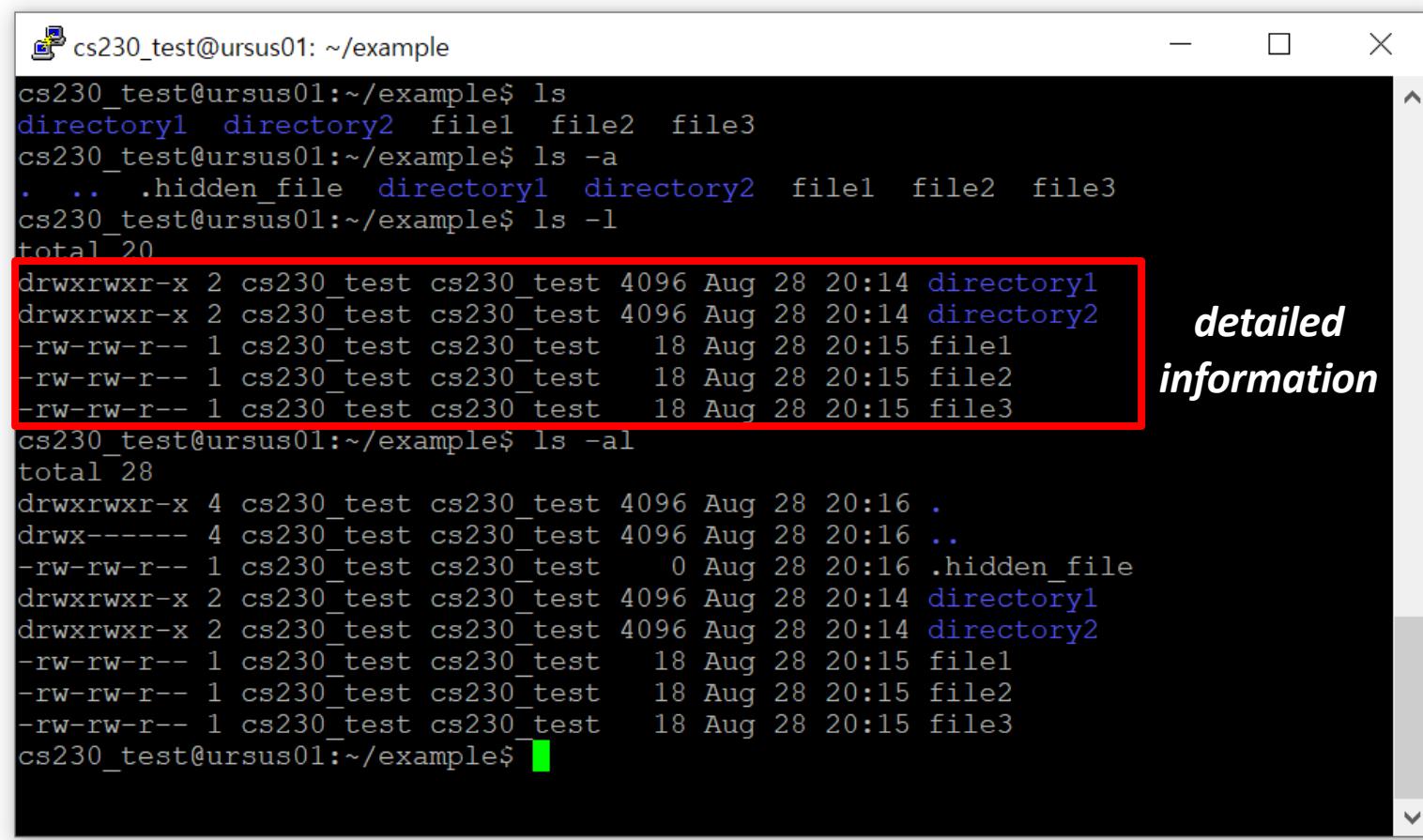
The screenshot shows a terminal window with the following session:

```
cs230_test@ursus01: ~/example
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ ls -a
. .. .hidden_file directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ ls -l
total 1
hidden file
drwxrwxr-x 2 cs230_test cs230_test 4096 Aug 28 20:14 directory1
drwxrwxr-x 2 cs230_test cs230_test 4096 Aug 28 20:14 directory2
-rw-rw-r-- 1 cs230_test cs230_test 18 Aug 28 20:15 file1
-rw-rw-r-- 1 cs230_test cs230_test 18 Aug 28 20:15 file2
-rw-rw-r-- 1 cs230_test cs230_test 18 Aug 28 20:15 file3
cs230_test@ursus01:~/example$ ls -al
total 28
drwxrwxr-x 4 cs230_test cs230_test 4096 Aug 28 20:16 .
drwxrwxr-x 4 cs230_test cs230_test 4096 Aug 28 20:16 ..
-rw-rw-r-- 1 cs230_test cs230_test 0 Aug 28 20:16 .hidden_file
drwxrwxr-x 2 cs230_test cs230_test 4096 Aug 28 20:14 directory1
drwxrwxr-x 2 cs230_test cs230_test 4096 Aug 28 20:14 directory2
-rw-rw-r-- 1 cs230_test cs230_test 18 Aug 28 20:15 file1
-rw-rw-r-- 1 cs230_test cs230_test 18 Aug 28 20:15 file2
-rw-rw-r-- 1 cs230_test cs230_test 18 Aug 28 20:15 file3
cs230_test@ursus01:~/example$
```

A red box highlights the file ".hidden\_file" in the second "ls -a" command output. A green box highlights the word "hidden file" in the third "ls -l" command output.

# 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



The screenshot shows a terminal window with the following session:

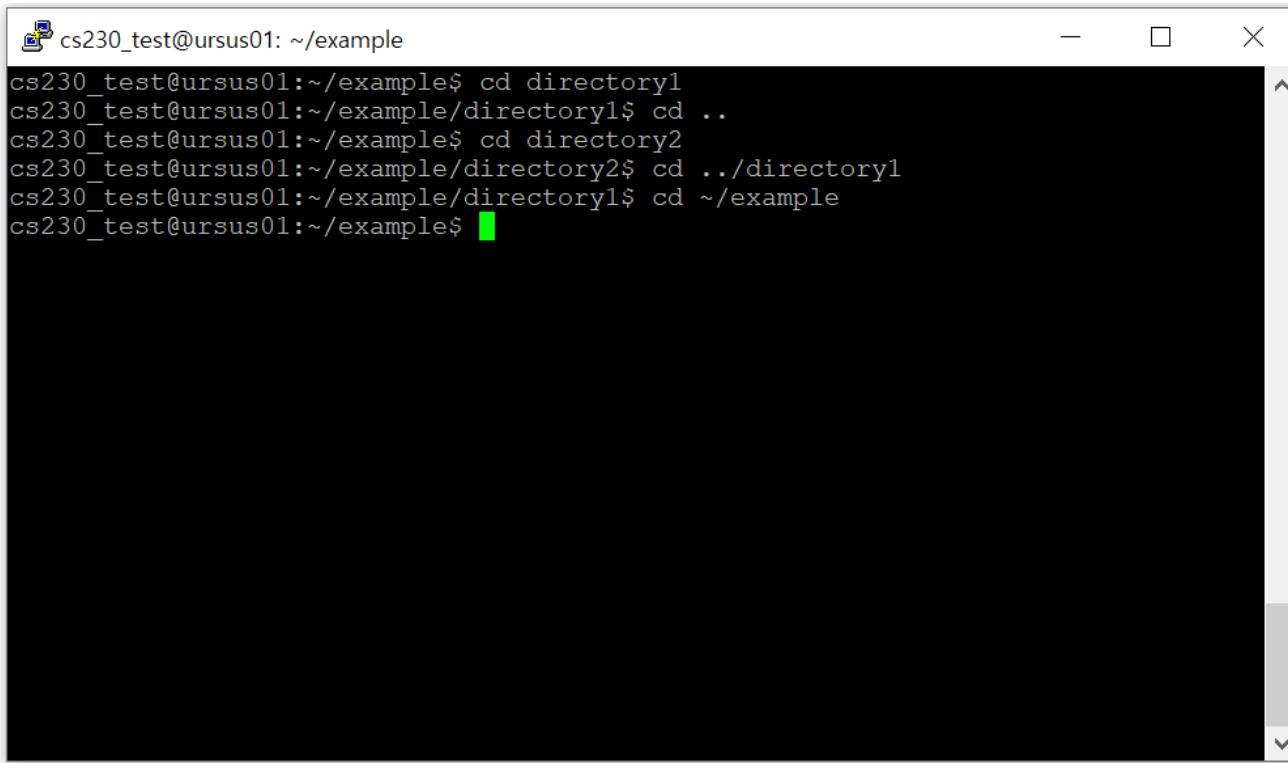
```
cs230_test@ursus01:~/example$ ls
directory1  directory2  file1  file2  file3
cs230_test@ursus01:~/example$ ls -a
.  ..  .hidden_file  directory1  directory2  file1  file2  file3
cs230_test@ursus01:~/example$ ls -l
total 20
drwxrwxr-x  2 cs230_test cs230_test 4096 Aug 28 20:14 directory1
drwxrwxr-x  2 cs230_test cs230_test 4096 Aug 28 20:14 directory2
-rw-rw-r--  1 cs230_test cs230_test    18 Aug 28 20:15 file1
-rw-rw-r--  1 cs230_test cs230_test    18 Aug 28 20:15 file2
-rw-rw-r--  1 cs230_test cs230_test    18 Aug 28 20:15 file3
cs230_test@ursus01:~/example$ ls -al
total 28
drwxrwxr-x  4 cs230_test cs230_test 4096 Aug 28 20:16 .
drwxrwxr-x  2 cs230_test cs230_test 4096 Aug 28 20:16 ..
-rw-rw-r--  1 cs230_test cs230_test     0 Aug 28 20:16 .hidden_file
drwxrwxr-x  2 cs230_test cs230_test 4096 Aug 28 20:14 directory1
drwxrwxr-x  2 cs230_test cs230_test 4096 Aug 28 20:14 directory2
-rw-rw-r--  1 cs230_test cs230_test    18 Aug 28 20:15 file1
-rw-rw-r--  1 cs230_test cs230_test    18 Aug 28 20:15 file2
-rw-rw-r--  1 cs230_test cs230_test    18 Aug 28 20:15 file3
cs230_test@ursus01:~/example$
```

A red box highlights the first five lines of the `ls -l` output, which are the permissions, owner, group, size, and last modified time for the two directories and three files. To the right of this box, the text "detailed information" is written in a bold, italicized font.

# Linux Commands

---

- cd [directory]
  - change working directory



A screenshot of a Linux terminal window titled "cs230\_test@ursus01: ~/example". The window contains the following command history:

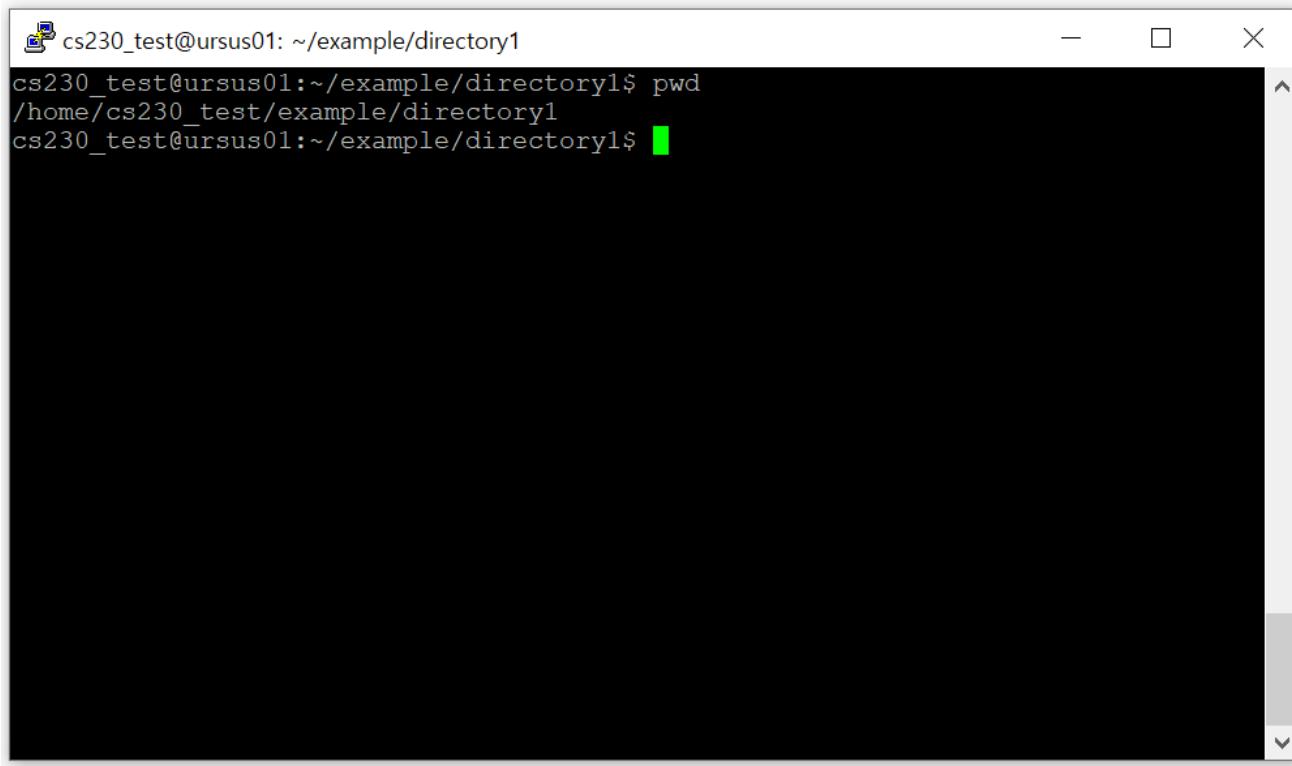
```
cs230_test@ursus01:~/example$ cd directory1
cs230_test@ursus01:~/example/directory1$ cd ..
cs230_test@ursus01:~/example$ cd directory2
cs230_test@ursus01:~/example/directory2$ cd ../directory1
cs230_test@ursus01:~/example/directory1$ cd ~/example
cs230_test@ursus01:~/example$
```

The terminal window has a dark background and light-colored text. A small green square cursor is visible at the end of the last command line.

# Linux Commands

---

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



A screenshot of a Linux terminal window titled "cs230\_test@ursus01: ~/example/directory1". The window contains the following text:

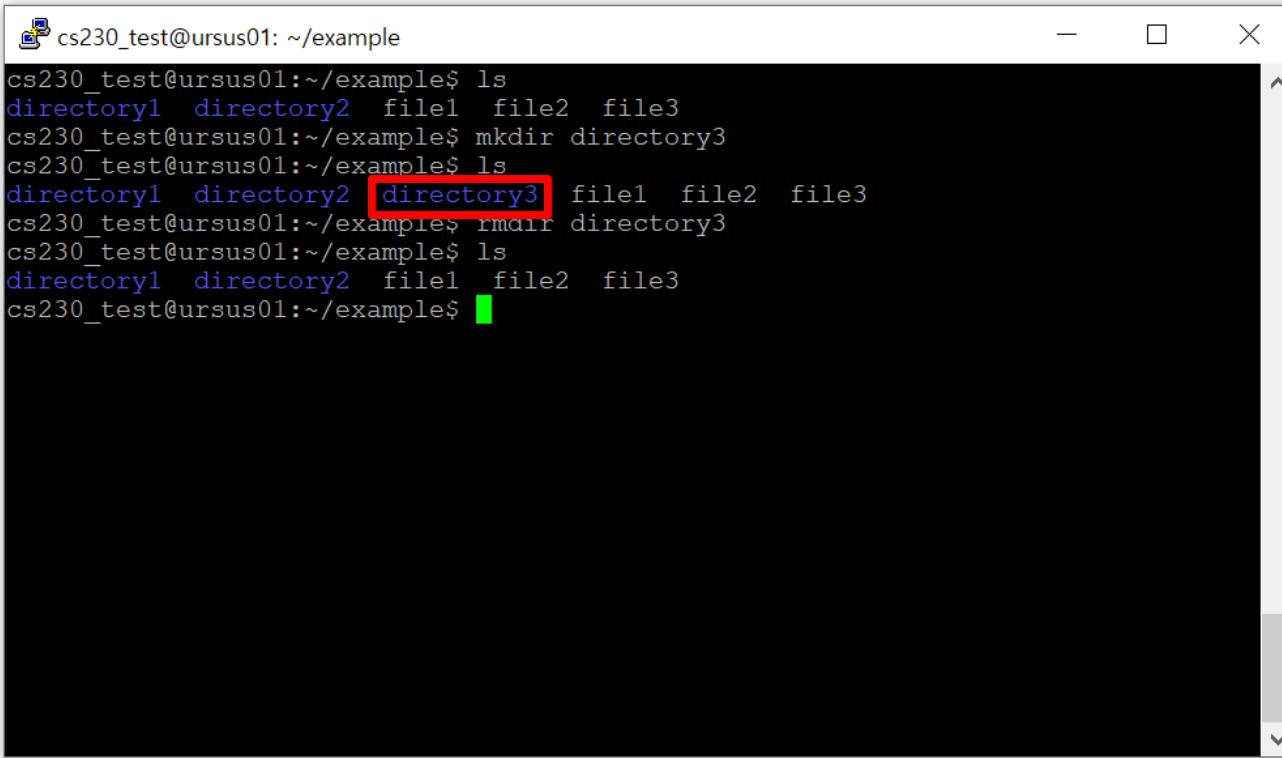
```
cs230_test@ursus01:~/example/directory1$ pwd  
/home/cs230_test/example/directory1  
cs230_test@ursus01:~/example/directory1$ █
```

The terminal has a dark background and light-colored text. The cursor is represented by a green square at the end of the command line.

# Linux Commands

---

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

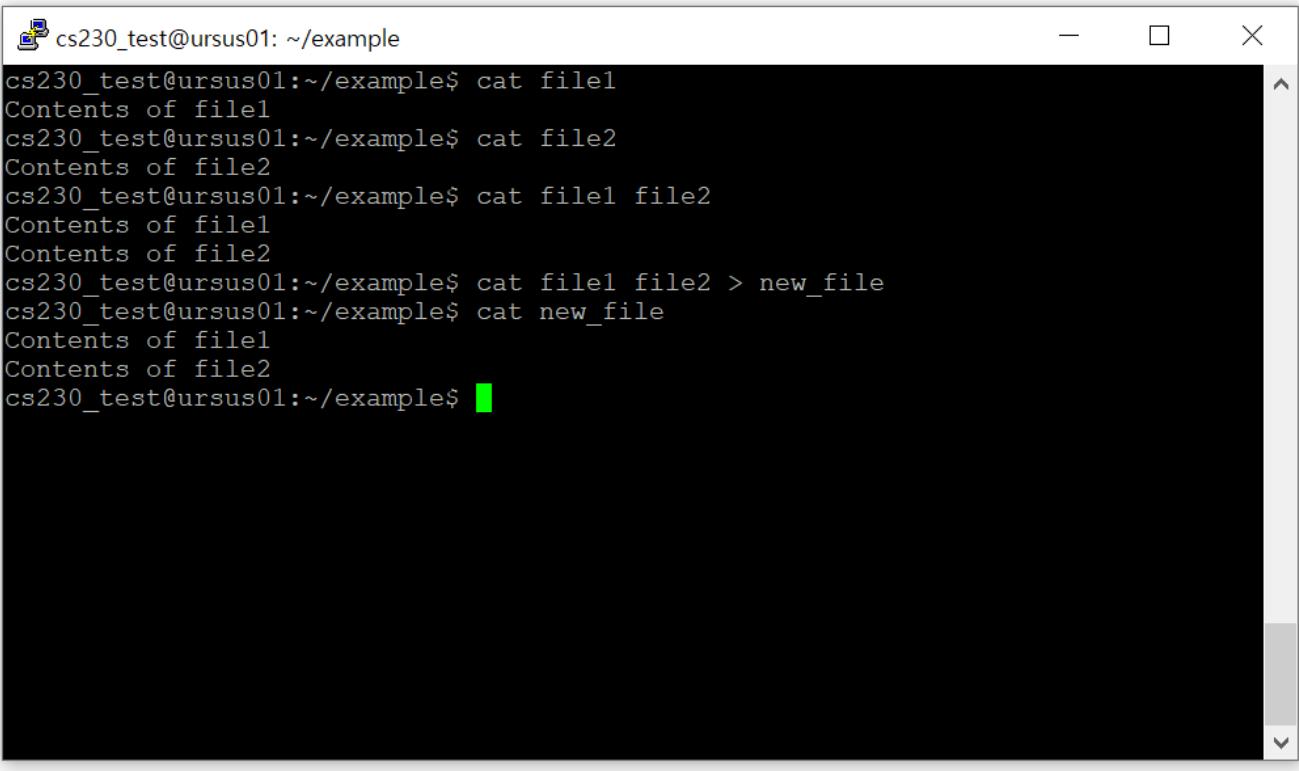


```
cs230_test@ursus01:~/example
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ mkdir directory3
cs230_test@ursus01:~/example$ ls
directory1 directory2 directory3 file1 file2 file3
cs230_test@ursus01:~/example$ rmdir directory3
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$
```

# 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)



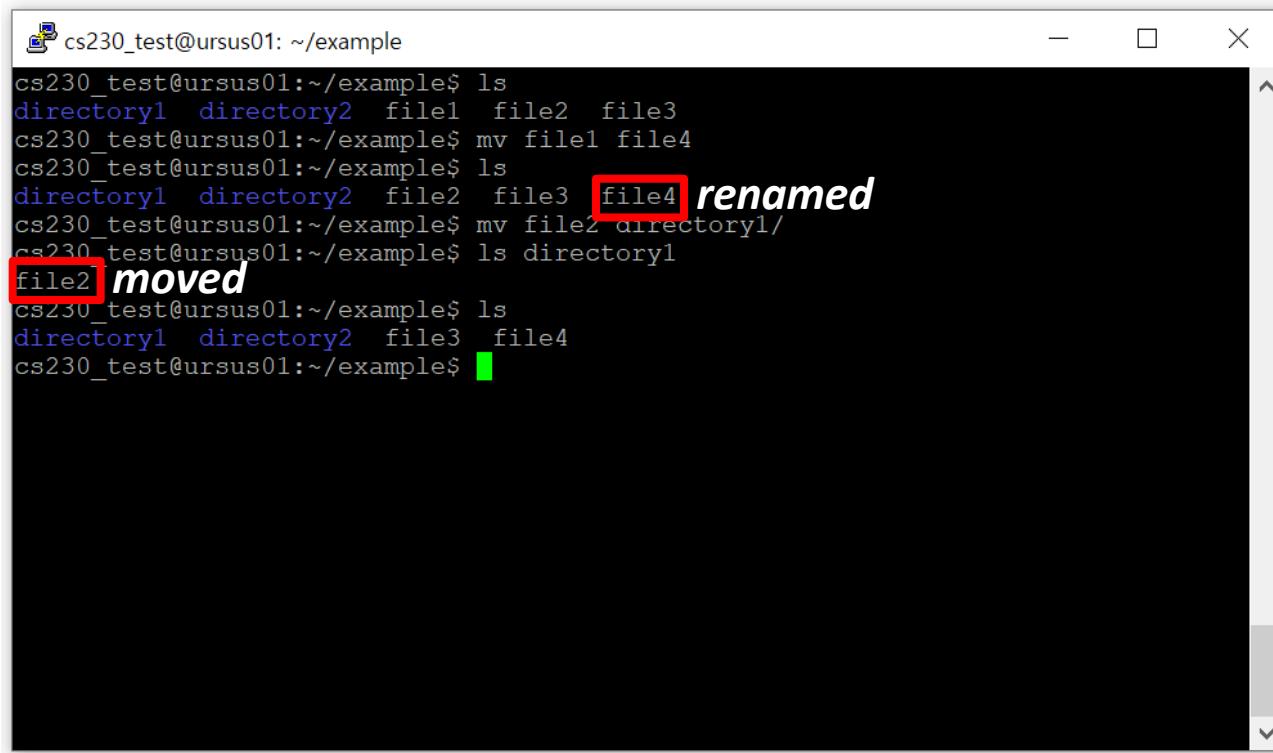
The screenshot shows a terminal window with a black background and white text. The window title is "cs230\_test@ursus01: ~/example". The terminal displays the following commands and their outputs:

```
cs230_test@ursus01:~/example$ cat file1
Contents of file1
cs230_test@ursus01:~/example$ cat file2
Contents of file2
cs230_test@ursus01:~/example$ cat file1 file2
Contents of file1
Contents of file2
cs230_test@ursus01:~/example$ cat file1 file2 > new_file
cs230_test@ursus01:~/example$ cat new_file
Contents of file1
Contents of file2
cs230_test@ursus01:~/example$
```

# Linux Commands

---

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



The screenshot shows a terminal window with the following session:

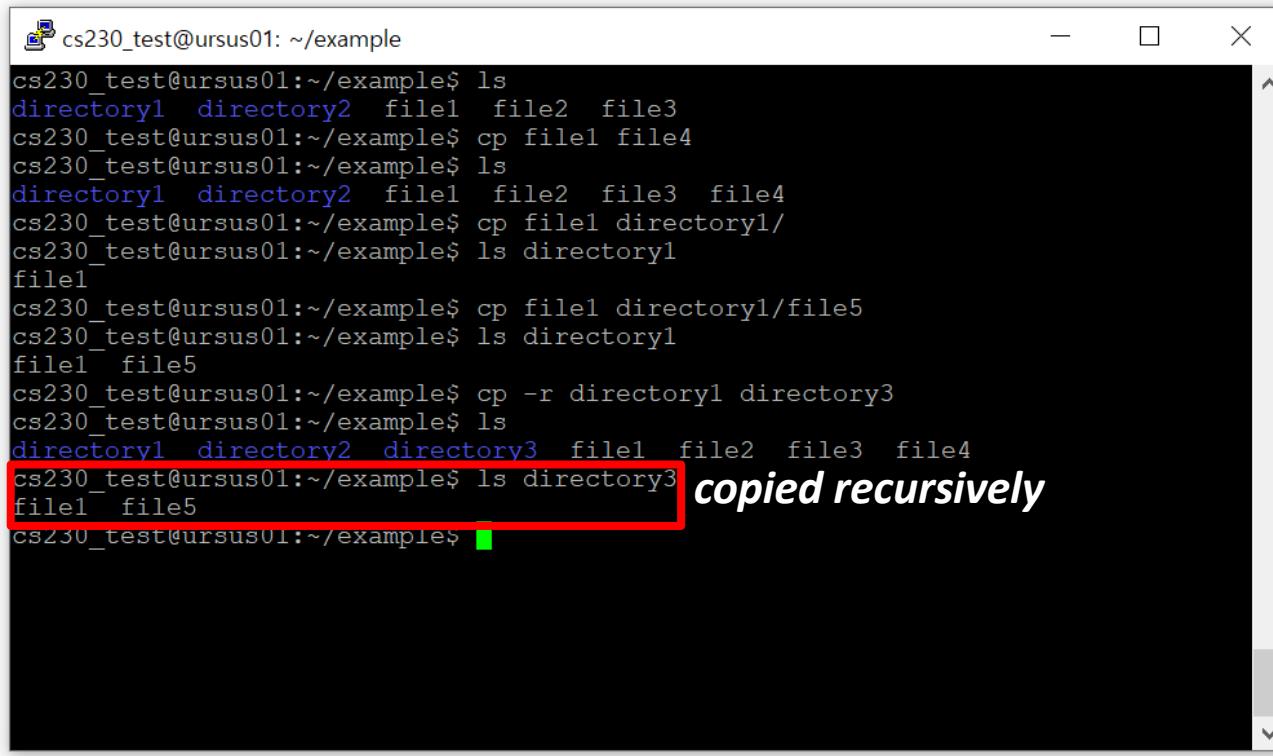
```
cs230_test@ursus01: ~/example
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ mv file1 file4
cs230_test@ursus01:~/example$ ls
directory1 directory2 file2 file3 file4 renamed
cs230_test@ursus01:~/example$ mv file2 directory1/
cs230_test@ursus01:~/example$ ls directory1
file2 moved
cs230_test@ursus01:~/example$ ls
directory1 directory2 file3 file4
cs230_test@ursus01:~/example$
```

The terminal window has a red box around "file4 renamed" and another red box around "file2 moved". A green box highlights the final command prompt at the bottom.

# Linux Commands

---

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



The screenshot shows a terminal window with the following session:

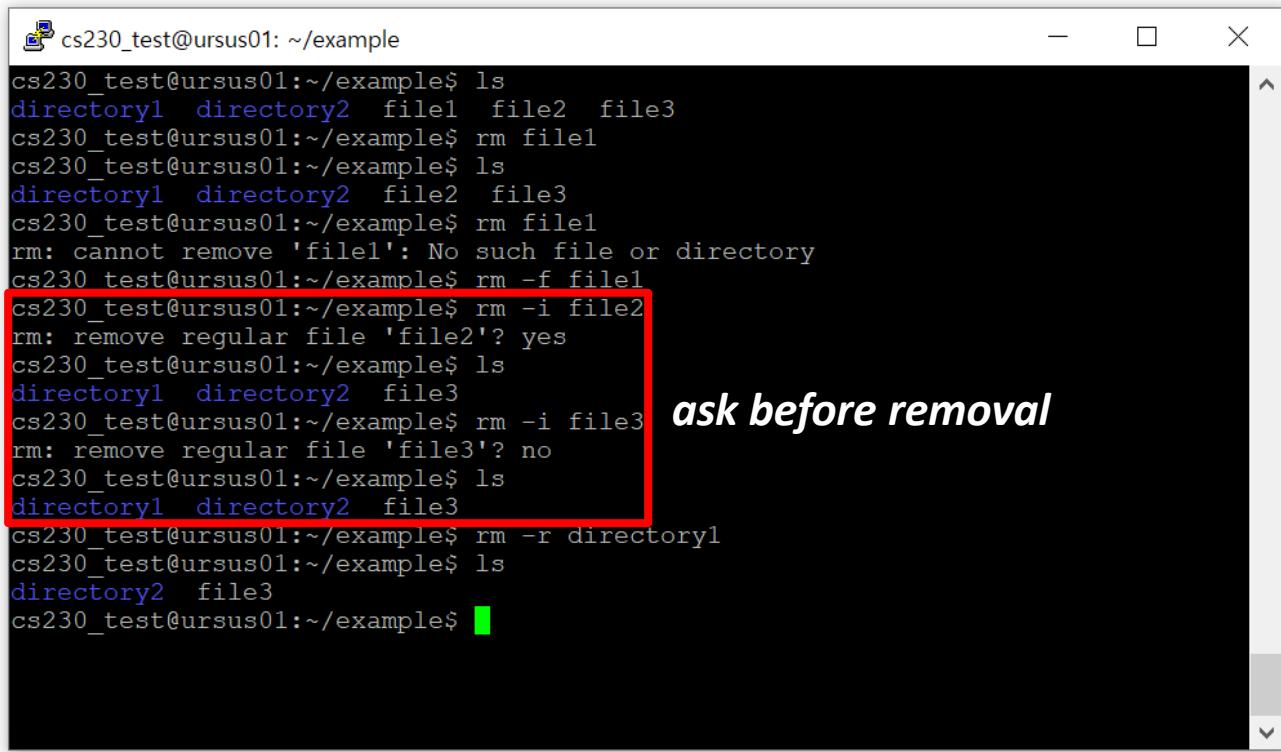
```
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ cp file1 file4
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3 file4
cs230_test@ursus01:~/example$ cp file1 directory1/
cs230_test@ursus01:~/example$ ls directory1
file1
cs230_test@ursus01:~/example$ cp file1 directory1/file5
cs230_test@ursus01:~/example$ ls directory1
file1 file5
cs230_test@ursus01:~/example$ cp -r directory1 directory3
cs230_test@ursus01:~/example$ ls
directory1 directory2 directory3 file1 file2 file3 file4
cs230_test@ursus01:~/example$ ls directory3
file1 file5
cs230_test@ursus01:~/example$
```

A red box highlights the command `cp -r directory1 directory3` and its output. The word "copied recursively" is overlaid in white text on the right side of the highlighted area.

# 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



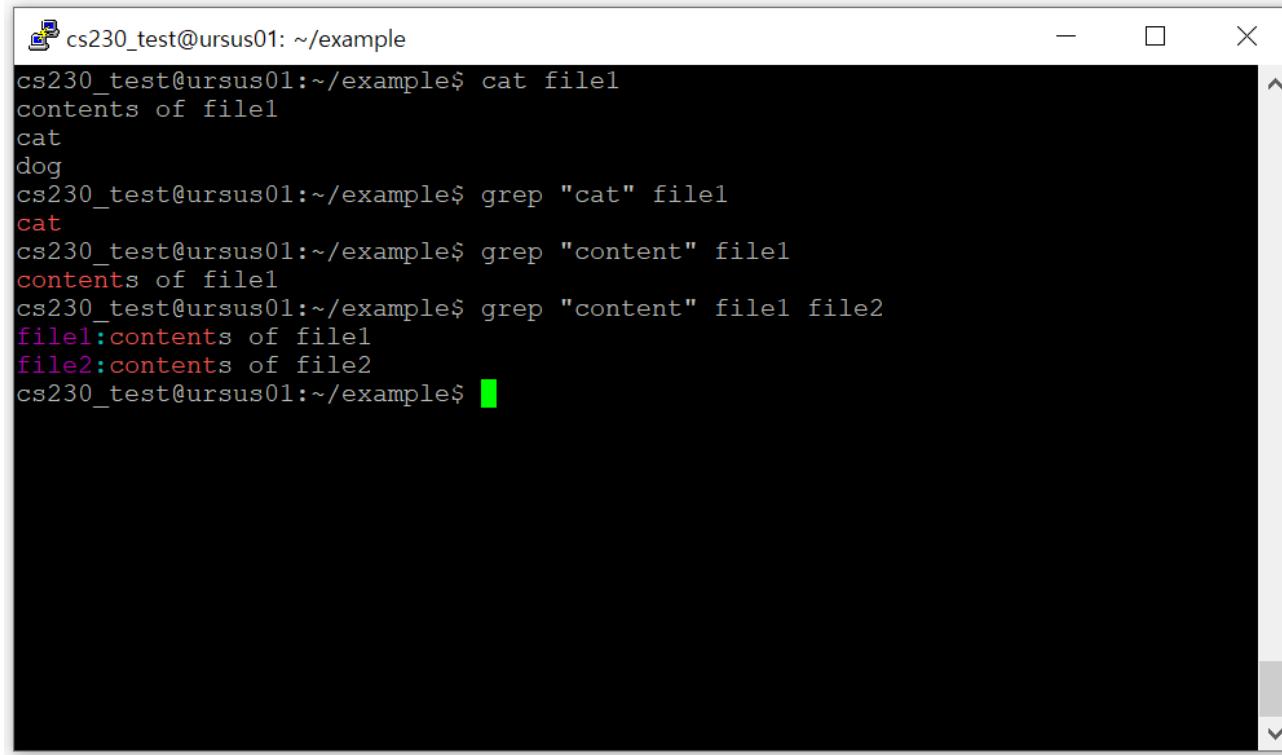
```
cs230_test@ursus01:~/example$ ls
directory1  directory2  file1  file2  file3
cs230_test@ursus01:~/example$ rm file1
cs230_test@ursus01:~/example$ ls
directory1  directory2  file2  file3
cs230_test@ursus01:~/example$ rm file1
rm: cannot remove 'file1': No such file or directory
cs230_test@ursus01:~/example$ rm -f file1
cs230_test@ursus01:~/example$ rm -i file2
rm: remove regular file 'file2'? yes
cs230_test@ursus01:~/example$ ls
directory1  directory2  file3
cs230_test@ursus01:~/example$ rm -i file3
rm: remove regular file 'file3'? no
cs230_test@ursus01:~/example$ ls
directory1  directory2  file3
cs230_test@ursus01:~/example$ rm -r directory1
cs230_test@ursus01:~/example$ ls
directory2  file3
cs230_test@ursus01:~/example$
```

*ask before removal*

# Linux Commands

---

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



A screenshot of a Linux terminal window titled "cs230\_test@ursus01: ~/example". The terminal displays the following command-line session:

```
cs230_test@ursus01:~/example$ cat file1
contents of file1
cat
dog
cs230_test@ursus01:~/example$ grep "cat" file1
cat
cs230_test@ursus01:~/example$ grep "content" file1
contents of file1
cs230_test@ursus01:~/example$ grep "content" file1 file2
file1:contents of file1
file2:contents of file2
cs230_test@ursus01:~/example$ █
```

The terminal window has a dark background and light-colored text. It includes standard window controls (minimize, maximize, close) at the top right.

# Linux Commands

---

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

```
GREP(1)                                General Commands Manual                               GREP(1)

NAME
    grep, egrep, fgrep, rgrep - print lines matching a pattern

SYNOPSIS
    grep [OPTIONS] PATTERN [FILE...]
    grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
    grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

    In addition, three variant programs egrep, fgrep and rgrep are available. egrep is the same as grep -E. fgrep is the same as grep -F. rgrep is the same as grep -r. Direct invocation as either egrep or fgrep is deprecated, but is provided to allow historical applications that rely on them to run unmodified.

OPTIONS
    Generic Program Information
        --help Print a usage message briefly summarizing these command-line options and the bug-reporting address,
                then exit.

        -V, --version
                Print the version number of grep to the standard output stream. This version number should be
                included in all bug reports (see below).

    Matcher Selection
        -E, --extended-regexp
                Interpret PATTERN as an extended regular expression (ERE, see below). (-E is specified by POSIX.)
```

# 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]

# Vim

---

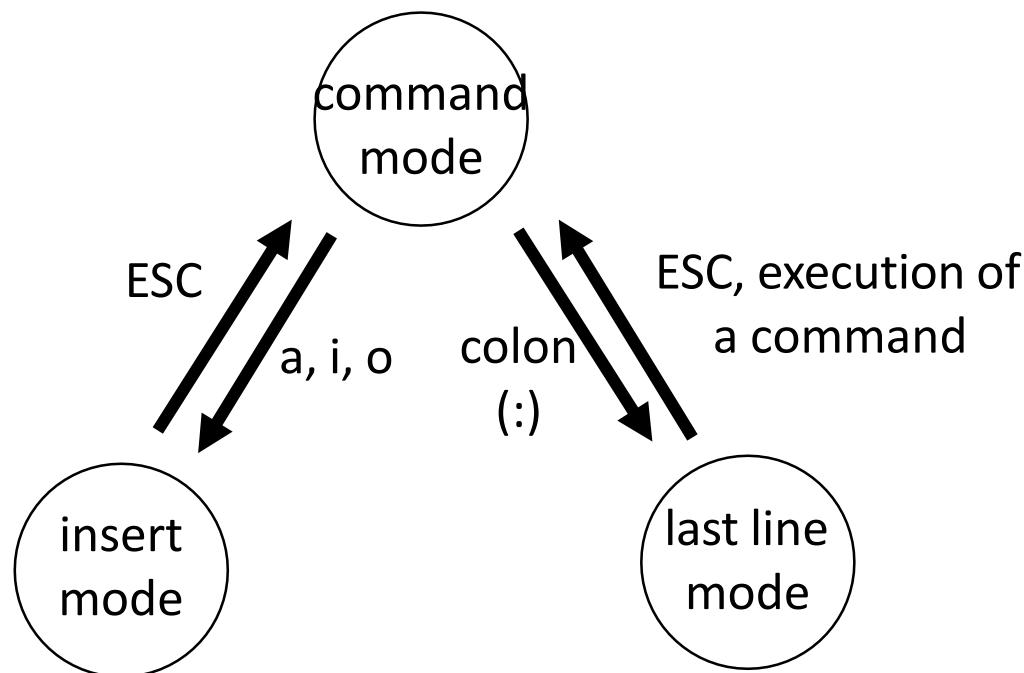
- vim
  - Vi IMproved, a programmer's text editor

```
VIM - Vi IMproved  
version 7.4.52  
by Bram Moolenaar et al.  
Modified by pkg-vim-maintainers@lists.alioth.debian.org  
Vim is open source and freely distributable  
  
Help poor children in Uganda!  
type :help iccf<Enter>      for information  
  
type :q<Enter>              to exit  
type :help<Enter> or <F1> for on-line help  
type :help version7<Enter>   for version info
```

# Vim

---

- vim
  - Vi IMproved, a programmer's text editor
  - Three modes
    - Command mode
    - Insert mode
    - Last line mode



# Vim

---

- vim
  - Vi IMproved, a programmer's text editor
  - Three modes
    - Command mode
    - Insert mode
    - Last line mode
  - Example

- \$ vi test.txt ← command mode
- press “a” or “i” ← insert mode
- input the contents
- press “Esc” ← Return to command mode
- :wq ← last line mode  
: save and quit

**ESC**

normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	( begin sentence	) end sentence	"soft" bol down	+ next line
\. goto mark	1 2 3 4 5 6 7 8 9 0										- prev line	= auto format
Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	W next word	e end word	r replace char	t 'till	y yank	u undo	i insert mode	o open below	p paste after	[ misc	]	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	. ex cmd line	". reg. spec	bol/ goto col	
a append	s subst char	d delete	f find char	g extra cmd <sup>6</sup>	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	'. goto mk. bol	\ not used!	
Z quit <sup>4</sup>	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- <sup>3</sup> indent	> indent <sup>3</sup>	? find (rev.)	/ find		
Z extra <sup>5</sup>	X delete char	C change	V visual mode	b prev word	n next (find)	m set mark	,	,				

**motion** moves the cursor, or defines the range for an operator

**command** direct action command, if red, it enters insert mode

**operator** requires a motion afterwards, operates between cursor & destination

**extra** special functions, requires extra input

q· commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: `guux([foo, bar, baz])`

WORDs: `guux (foo, bar, baz)`

### Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)  
 :e f (open file f),  
 :%s/x/y/g (replace 'x' by 'y' filewide),  
 :h (help in vim), :new (new file in vim),

### Other important commands:

CTRL-R: redo (vim),  
 CTRL-F/-B: page up/down,  
 CTRL-E/-Y: scroll line up/down,  
 CTRL-V: block-visual mode (vim only)

### Visual mode:

Move around and type operator to act on selected region (vim only)

### Notes:

(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,\*)  
 (e.g.: "ay\$ to copy rest of line to reg 'a')

(2) type in a number before any action to repeat it that number of times  
 (e.g.: 2p, d2w, 5i, d4j)

(3) duplicate operator to act on current line (dd = delete line, >> = indent line)

(4) ZZ to save & quit, ZQ to quit w/o saving

(5) zt: scroll cursor to top,  
 zb: bottom, zz: center

(6) gg: top of file (vim only),  
 gf: open file under cursor (vim only)

Esc

명령 모드

# Vim 명령어 단축키

손에 잡히는 Vim 인사이트



## 동작

커서를 이동하거나,  
연산자가 동작할 범위를 지정합니다.

## 명령

바로 동작하는 명령입니다.  
빨간색은 입력 모드로 변경됩니다.

## 오퍼레이션 펜딩 모드

커서 위치부터 목적지까지를 대상으로  
명령을 실행합니다.

## 확장

추가적인 키 입력이 필요합니다.

## q

입력 후 글자를 입력해야 합니다.

## 단어

공백 문자나 특수 문자로 구분된 단어  
test([123], 456, 789);

## 단어 (의미상)

공백 문자로 구분된 단어  
test(123, 456, 789);

## • 명령행 모드의 주요 명령어

- :w 저장
- :q 종료
- :q! 저장없이 종료
- :e f 파일 열기
- :%s/x/y/g 파일 전체에서 'x'를 'y'로 교체
- :h name name 명령에 대한 도움말
- :new 새 파일

## • 일반 모드의 주요 명령어

- CTRL-R 재실행
- CTRL-F/-B 페이지 위로/아래로
- CTRL-E/-Y 스크롤 위로/아래로
- CTRL-V 비주얼 모드

## • 참고

- ① 복사/붙여넣기/지우기 명령을 사용하기 전에  
"a"를 입력하여 레지스터 a를 지정할 수 있습니다.  
(레지스터 이름은 a부터 z까지 사용 가능)  
예를 들어 "ay\$"는 커서 위치부터 행 끝까지의 내용을  
레지스터 a에 저장합니다.
- ② 명령어 입력 전 숫자를 지정하면,  
해당 숫자만큼 명령어가 반복됩니다.
- ③ 연속으로 입력하면, 현재 행에 반영됩니다.  
예를 들어 dd는 현재 행이 지워집니다.
- ④ ZZ는 저장 후 종료, ZQ는 저장 없이 종료입니다.
- ⑤ zt는 커서가 위치한 부분을 화면 상단으로 스크롤합니다.  
zb는 바닥으로, zz는 가운데로 스크롤합니다.
- ⑥ gg는 커서를 파일 처음으로 이동합니다.  
gf는 커서 위치의 파일명을 인식하여 파일을 엽니다.

# Vim 이동 단축키

◀ 앞

**0** 행 처음으로    **^** 행 첫 글자  
행 처음으로    이전 x

, 이전 x

**Fx** 이전 x 문자로  
이전 x 문자로

**Tx** 이전 x 문자 뒤로  
이전 x 문자 뒤로

**B** 이전 단어

**gE** 이전 단어 끝

**b** 앞 단어  
앞 단어

**ge** 앞 단어 끝

**h** 원쪽

) 문장 끝  
문장 끝

**L** 화면 하단으로

**CTRL-d**

아래로 반 페이지 스크롤

**CTRL-f**

아래로 한 페이지 스크롤

## ● 이동키

"	,	#G	%
이전 위치	이전 수정 위치	#행	꽉 말로

\* 출처 : <https://bilbucket.org/lednaleid/vim-shortcut-wallpaper>

**gg** 위 / 이전  
첫 행

**#** 커서 위치  
단어 찾기

**N** 이전 검색

**?text** 위쪽으로 text 찾기

**CTRL-b** 위로 한 페이지 스크롤

**CTRL-u** 위로 반 페이지 스크롤

**H** 화면 상단으로

{ 문단 처음으로  
문장 처음으로

**K** 위쪽

j 아래쪽  
아래쪽

) 문장 끝  
문장 끝

**I** 오른쪽

**e** 단어 끝

**w** 다음 단어

**tx** 다음 x 문자 앞으로

**fx** 다음 x 문자로

**\$** 행 끝으로

E 단어(의미상) 끝

W 다음 단어 (의미상)

;

, 다음 x

▶

/text

아래쪽으로 text 찾기

n

다음 검색

\*

커서 위치

단어 찾기

**G** 아래 / 다음  
마지막 행

손에 잡히는 Vim 인사이트

# Vim

---

- Customization
  - see `~/.vimrc` and vim plugins
  - you can see line numbers in Vim
  - auto indentation is available
  - filetype-aware auto completion is also available  
(snipmate plugin)

# 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
```

...

...

# 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
```

```
start.o: start.c global.h
```

```
        gcc -c start.c
```

```
clean:
```

```
        rm -f main.o end.o inter.o start.o
```

# make

---

- A sample Makefile

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

# Self-study material

---

- Linux : <https://www.youtube.com/watch?v=CpTfQ-q6MPU>
- Vim :  
<https://www.youtube.com/watch?v=ggSyF1SVFr4>