

# 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

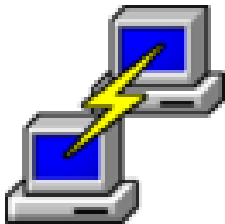
- **Why do we need Linux?**

- 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

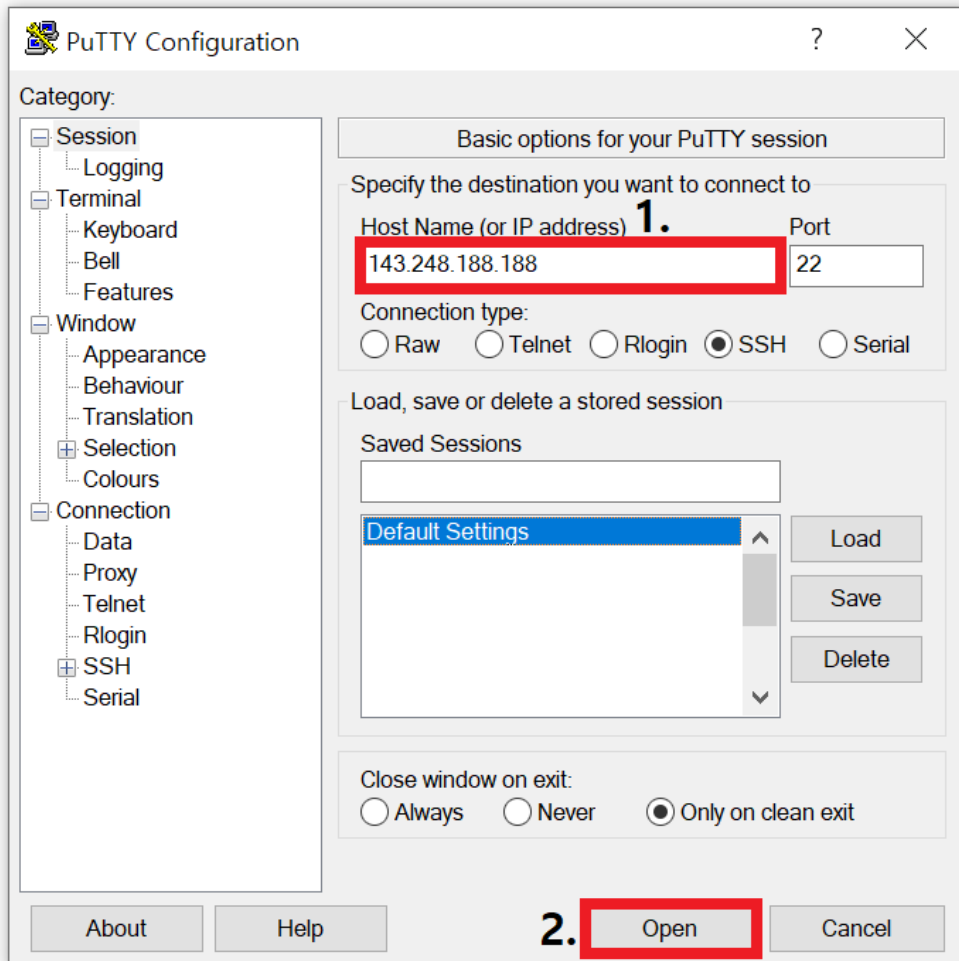


VSCode

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

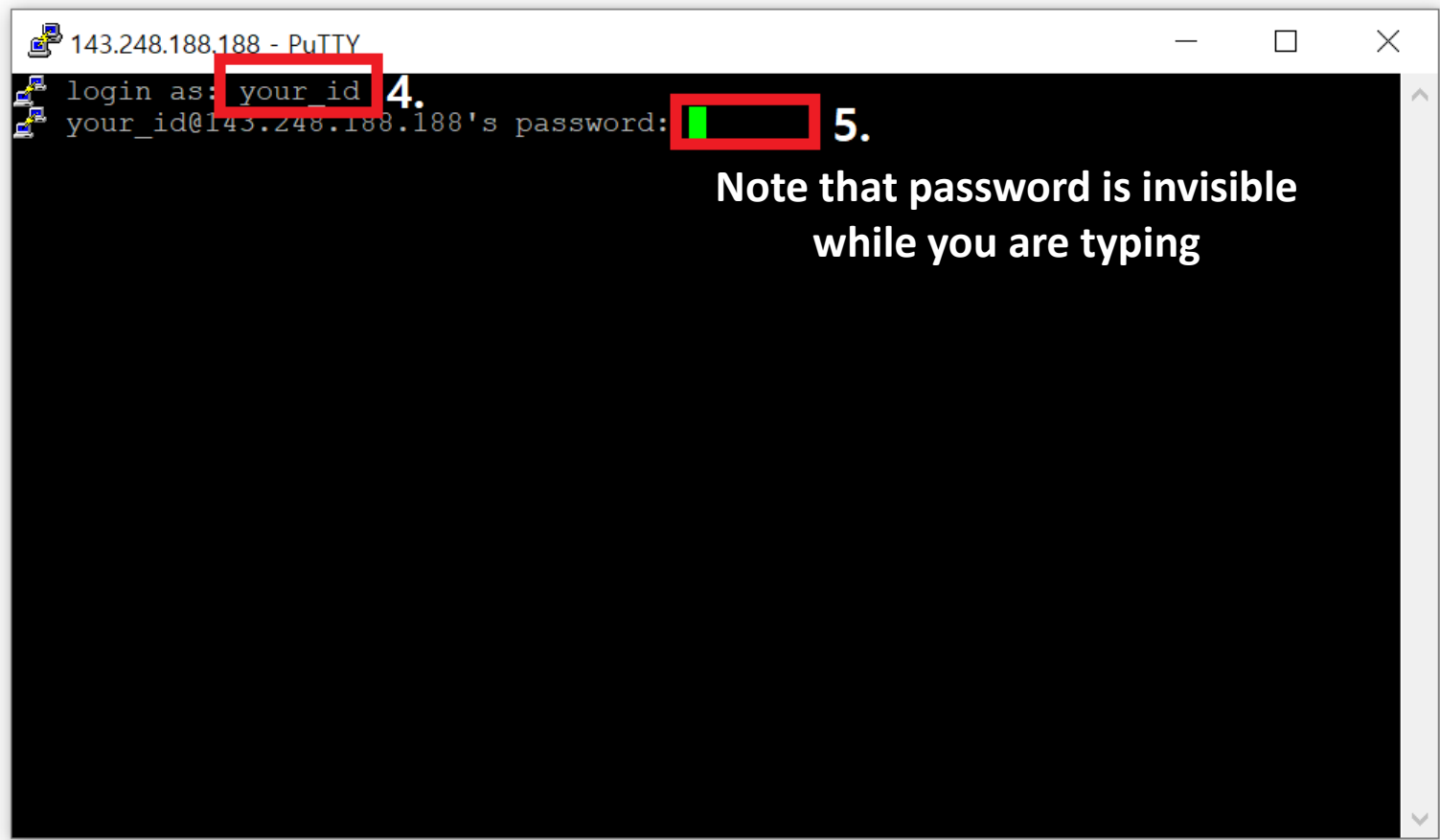
- Needs installation & setup
- User-friendly interface

# Connection Example: PuTTY



# Connection Example: PuTTY

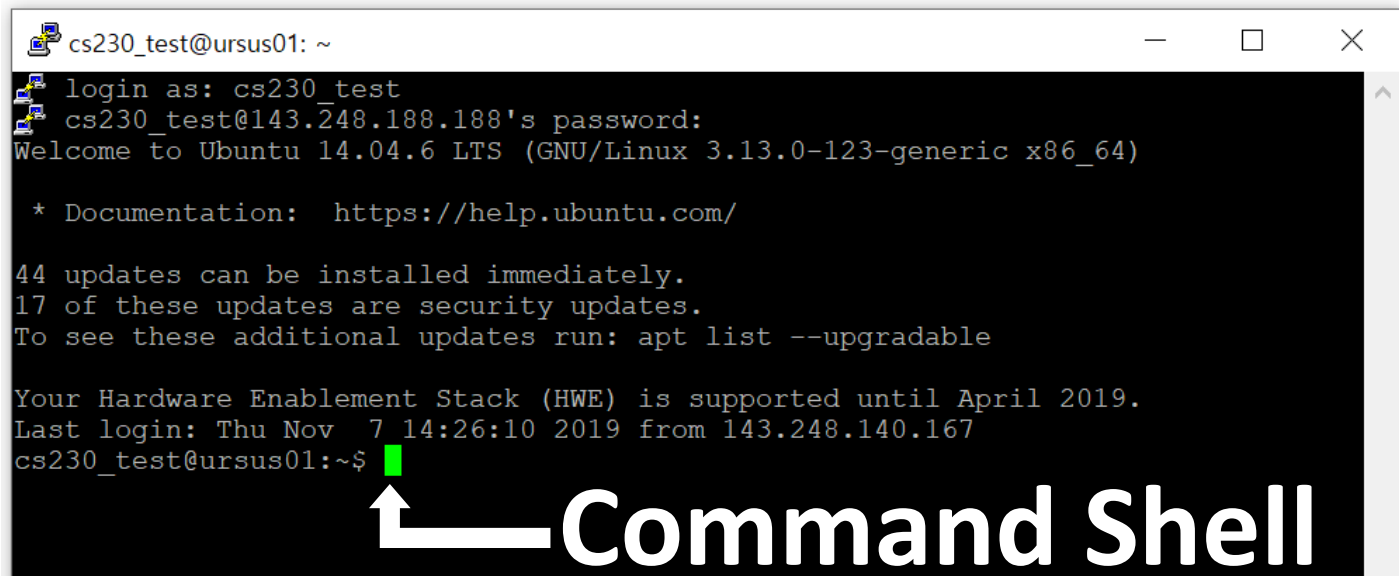
---



# Linux Shell

---

## Connected!



```
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?

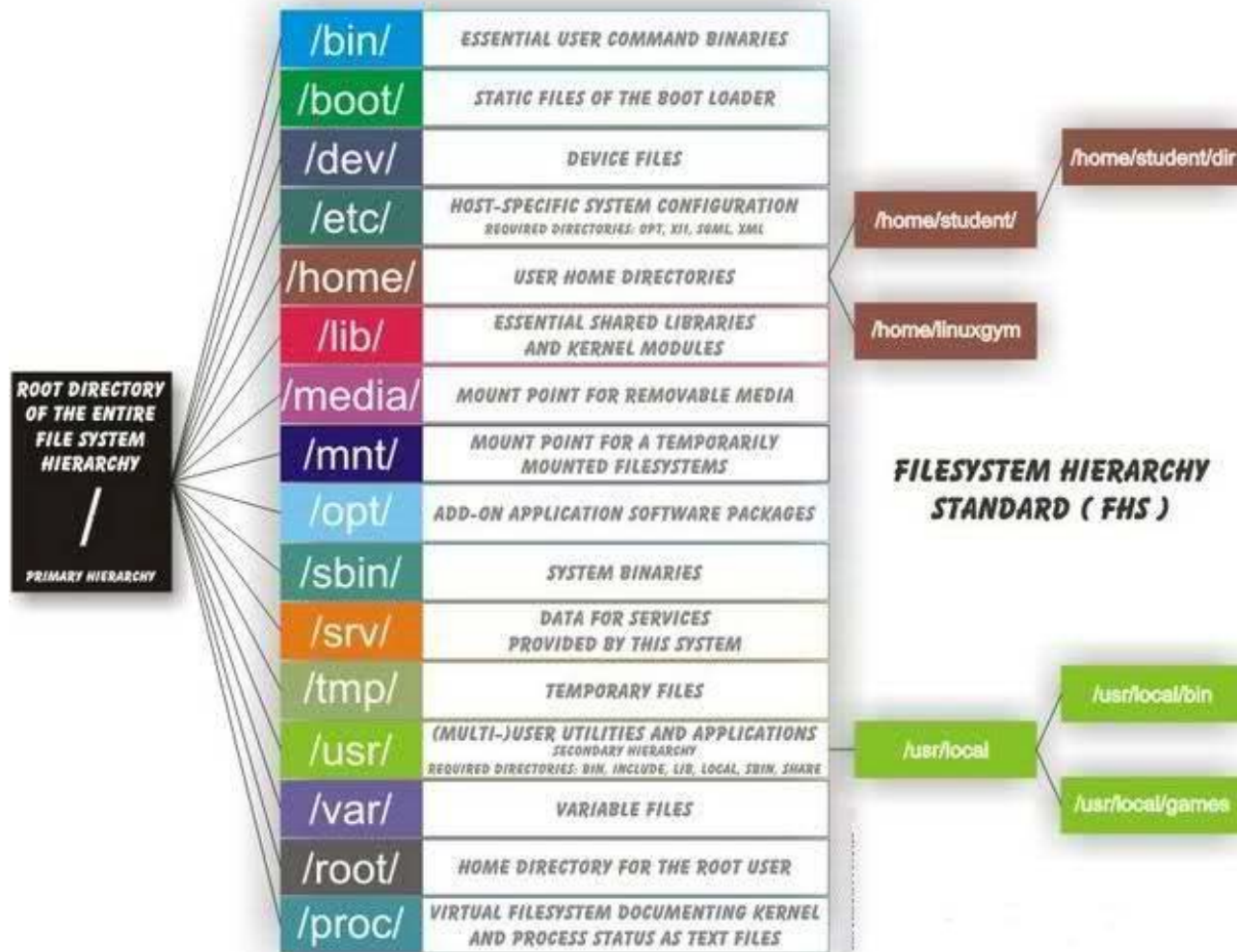
- a command line interface between a user and Linux machine

# Linux Directory

---

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

# Linux Directory Structure





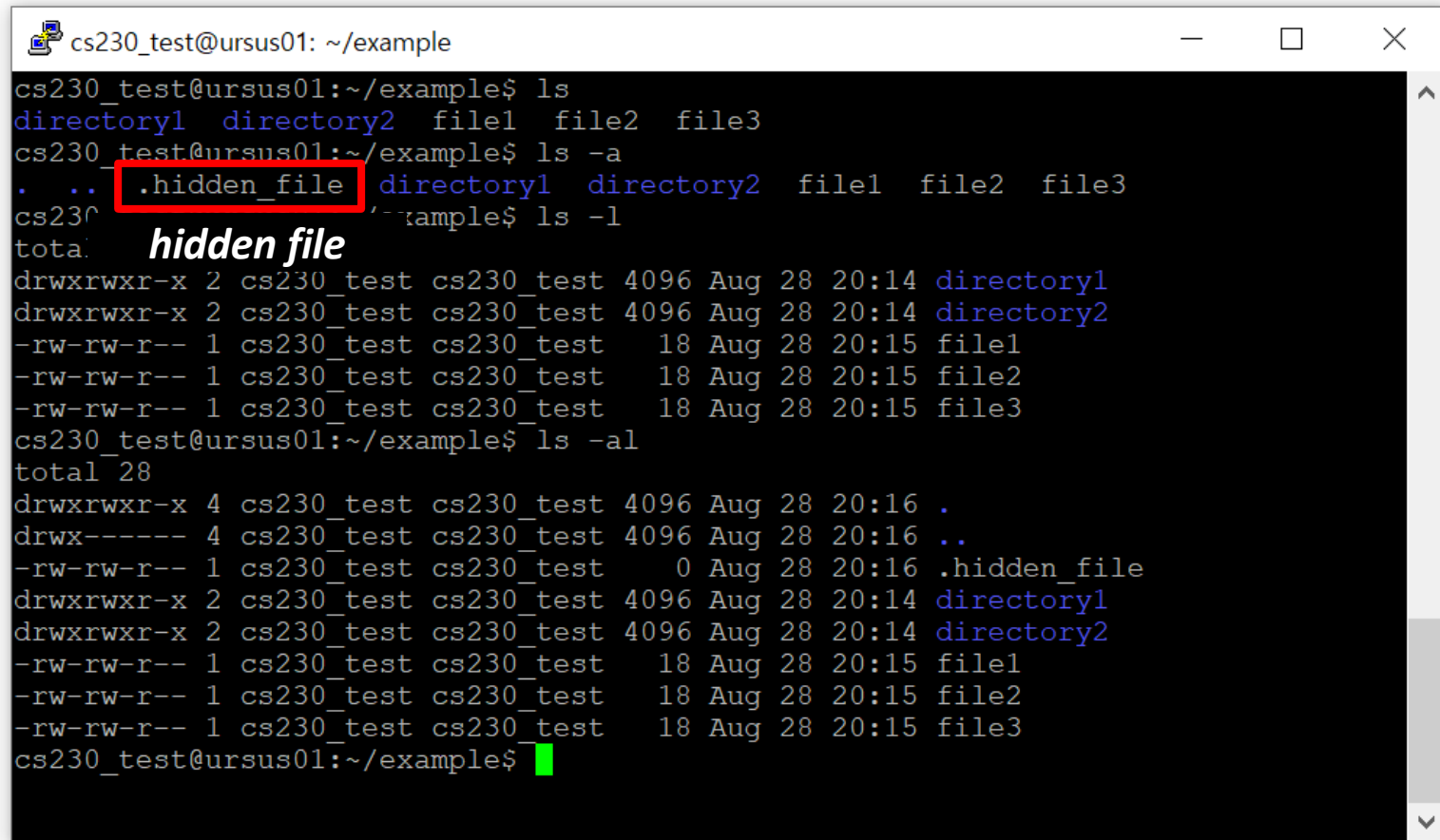
# Linux Commands

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

```
cs230_test@ursus01: ~/example
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ ls -la
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 .
drwx----- 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

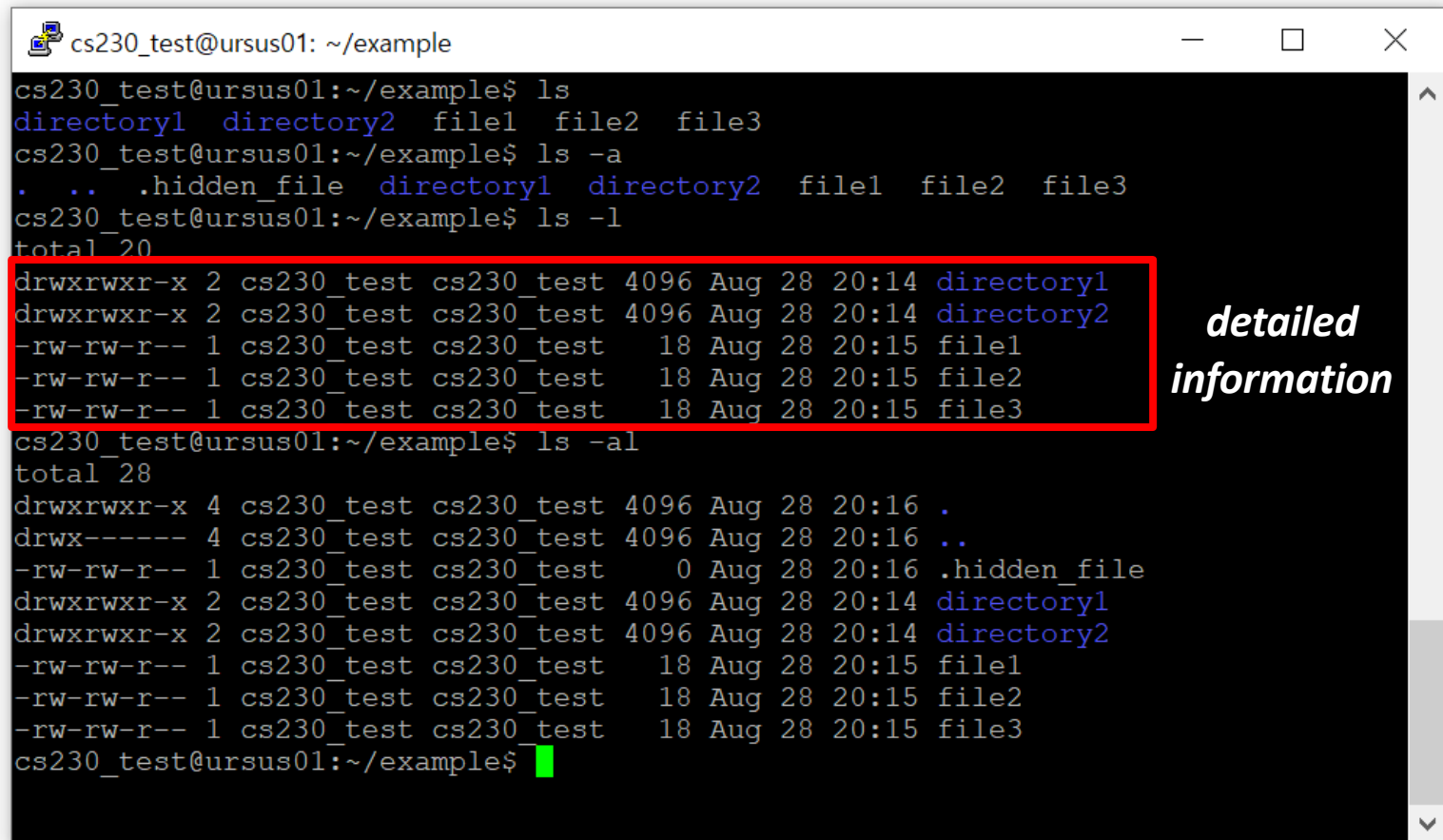
- ls [directory] (empty for working directory)
  - -a : Print the list including hidden contents (. name)



```
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 28
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 .
drwx----- 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)
  - -l : Print the list of files with detailed information
    - Permissions, number of links, owner name, owner group, size, last modification time, name



A terminal window titled 'cs230\_test@ursus01: ~/example' showing the output of the 'ls' command. The output is as follows:

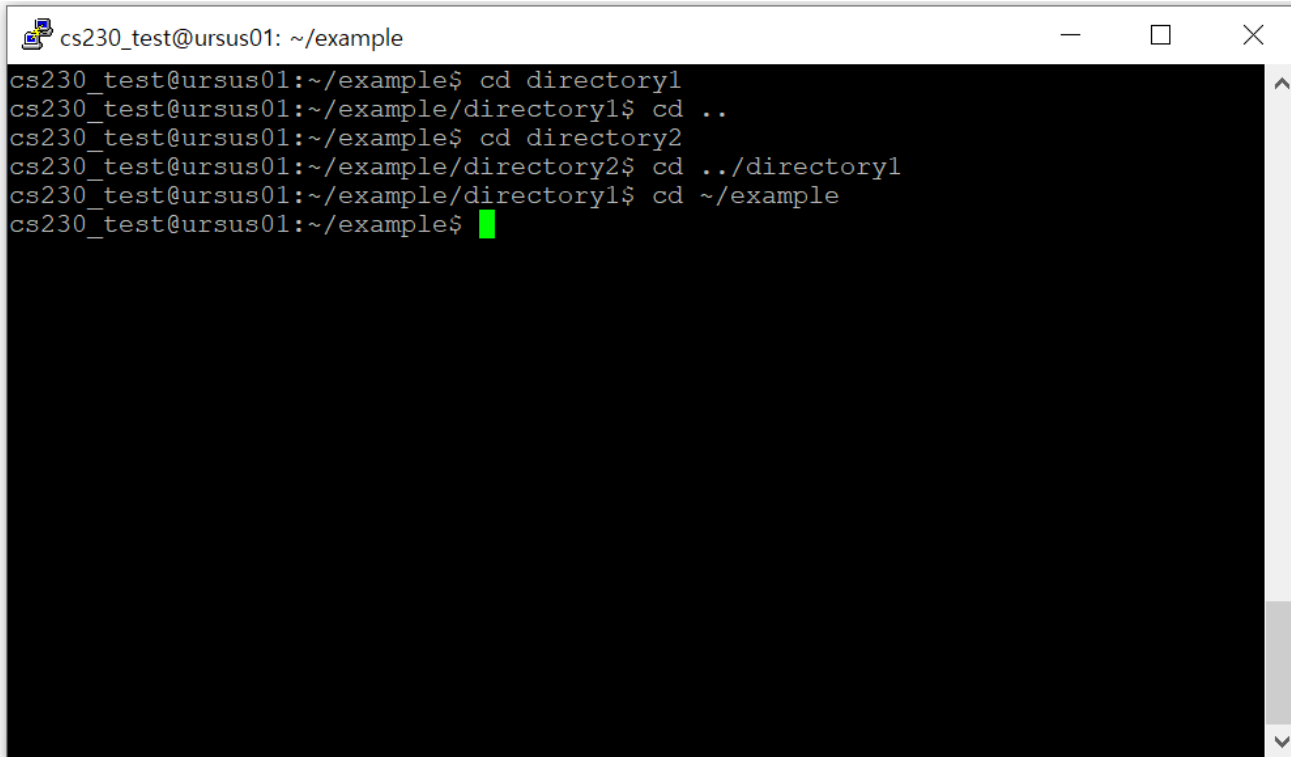
```
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ ls -a
.  .. .hidden_file directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ ls -l
total 20
drwxrwxr-x 2 cs230_test cs230_test 4096 Aug 28 20:14 directory1
drwxrwxr-x 2 cs230_test 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 .
drwx----- 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$
```

The output of the 'ls -l' command is highlighted with a red box. To the right of this box, the text **detailed information** is written in a bold, italicized font.

# Linux Commands

---

- `cd [directory]`
  - change working directory

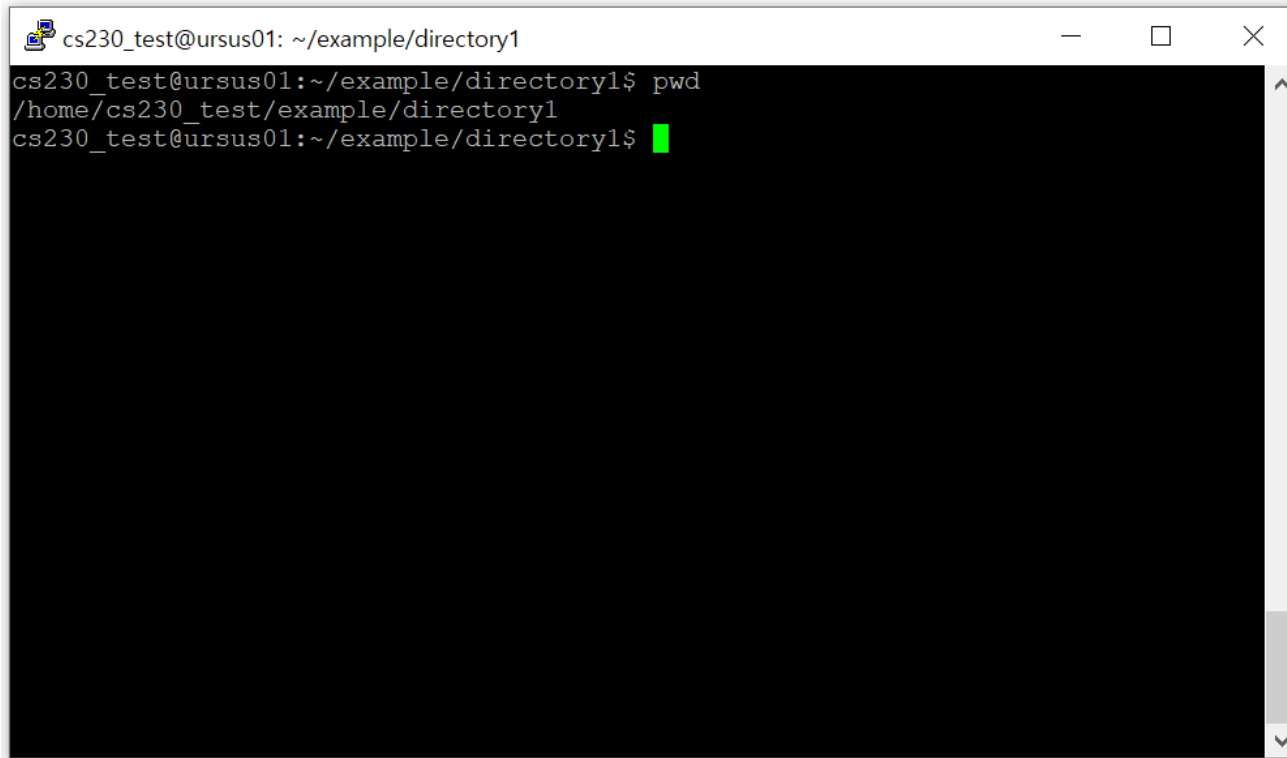
A terminal window titled 'cs230\_test@ursus01: ~/example' with standard window controls. The terminal shows a sequence of 'cd' commands: moving to 'directory1', then to '..' (parent), then to 'directory2', then to '../directory1', and finally back to '~/example'. The prompt returns to the root directory after the final command.

```
cs230_test@ursus01: ~/example
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$
```

# Linux Commands

---

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

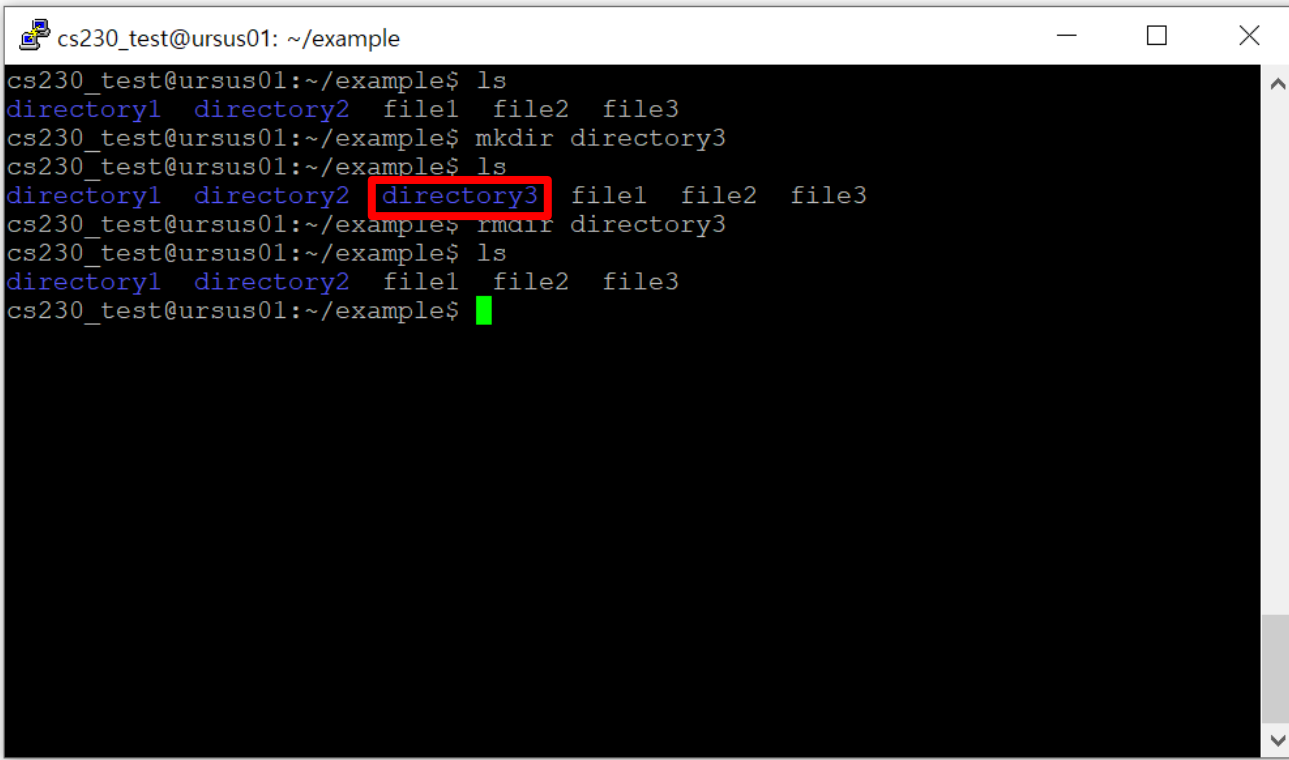
A terminal window with a title bar showing 'cs230\_test@ursus01: ~/example/directory1'. The terminal has a black background with white text. The prompt 'cs230\_test@ursus01:~/example/directory1\$' is followed by the command 'pwd'. The output of the command is '/home/cs230\_test/example/directory1'. Below the output, the prompt 'cs230\_test@ursus01:~/example/directory1\$' is shown again with a green cursor. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

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

# Linux Commands

---

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

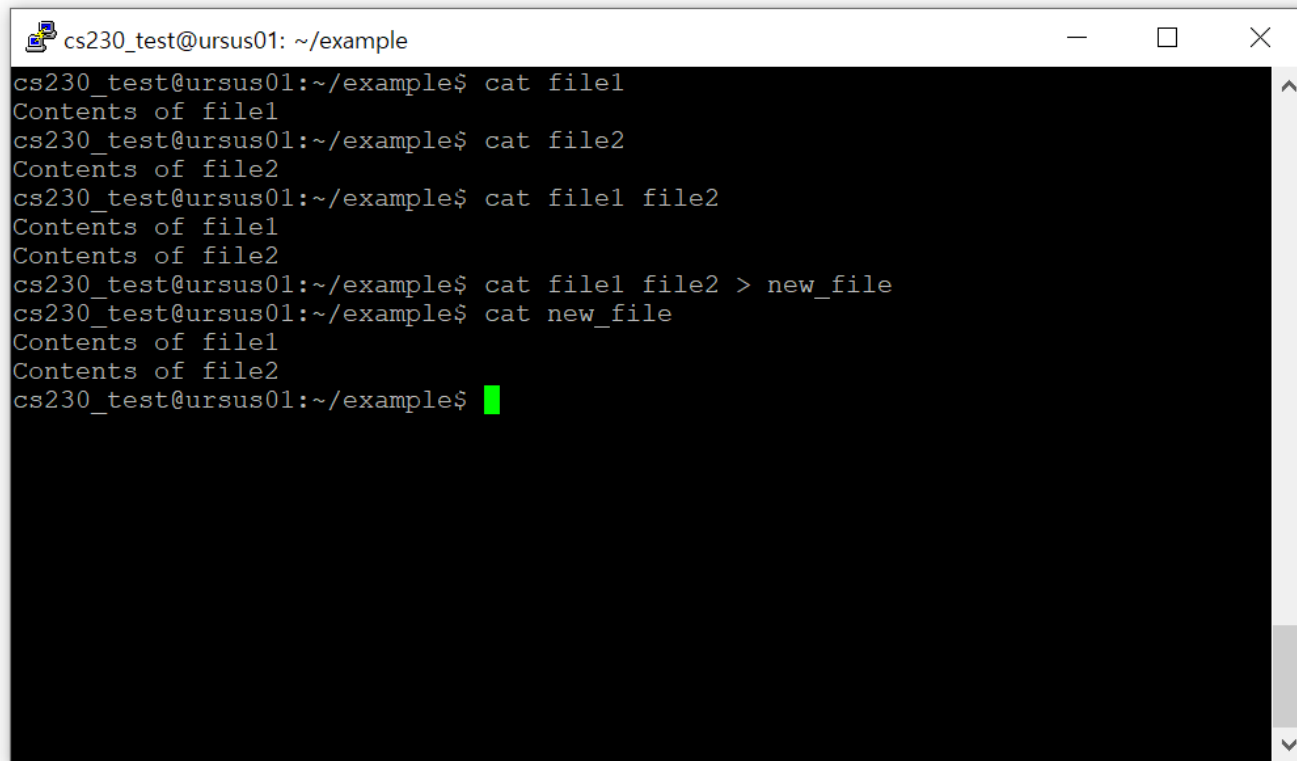
A terminal window titled 'cs230\_test@ursus01: ~/example' with standard window controls. The terminal shows a sequence of commands and their outputs. The 'ls' command is used to list directory contents. The 'mkdir directory3' command is used to create a new directory, and its output 'directory3' is highlighted with a red rectangle. The 'rmdir directory3' command is then used to remove the directory. The terminal text is as follows:

```
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

---

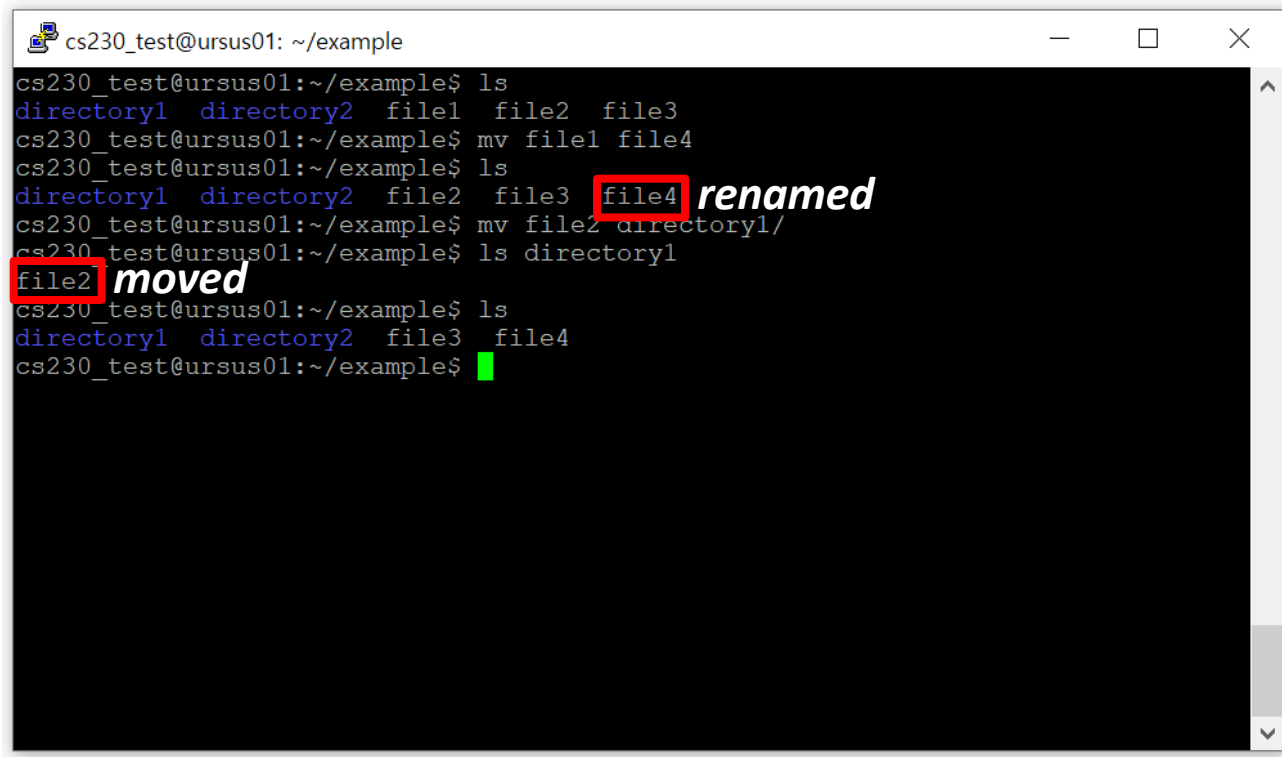
- cat
  - concatenate files and print on the standard output
  - > : redirects standard output to a file (overwrite)
  - >> : redirects standard output to a file (append)

A terminal window titled 'cs230\_test@ursus01: ~/example' with standard window controls. The terminal shows a series of commands and their outputs: 'cat file1' outputs 'Contents of file1', 'cat file2' outputs 'Contents of file2', 'cat file1 file2' outputs the contents of both files, 'cat file1 file2 > new\_file' creates a new file, and 'cat new\_file' outputs the concatenated contents. The prompt is green.

```
cs230_test@ursus01: ~/example
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

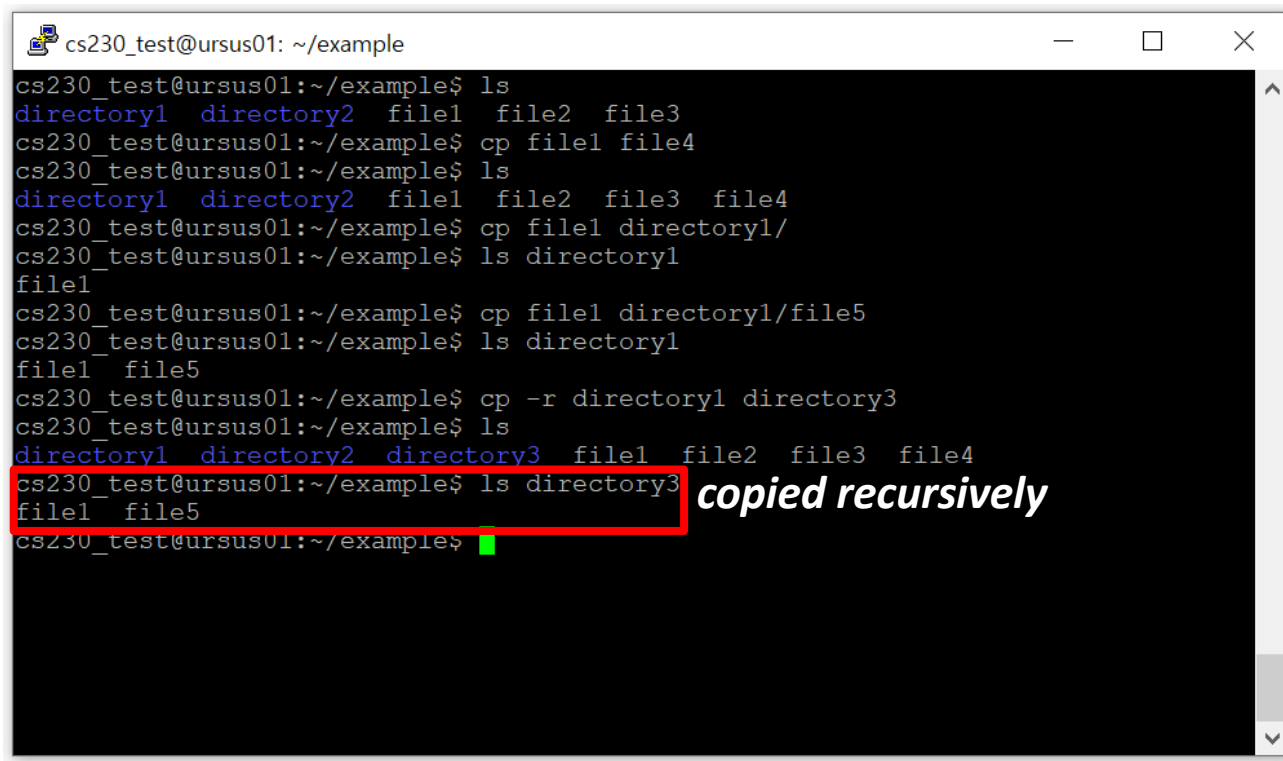


```
cs230_test@ursus01: ~/example
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ mv file1 file4
cs230_test@ursus01:~/example$ ls
directory1 directory2 file2 file3 file4 renamed
cs230_test@ursus01:~/example$ mv file2 directory1/
cs230_test@ursus01:~/example$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



# Linux Commands

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

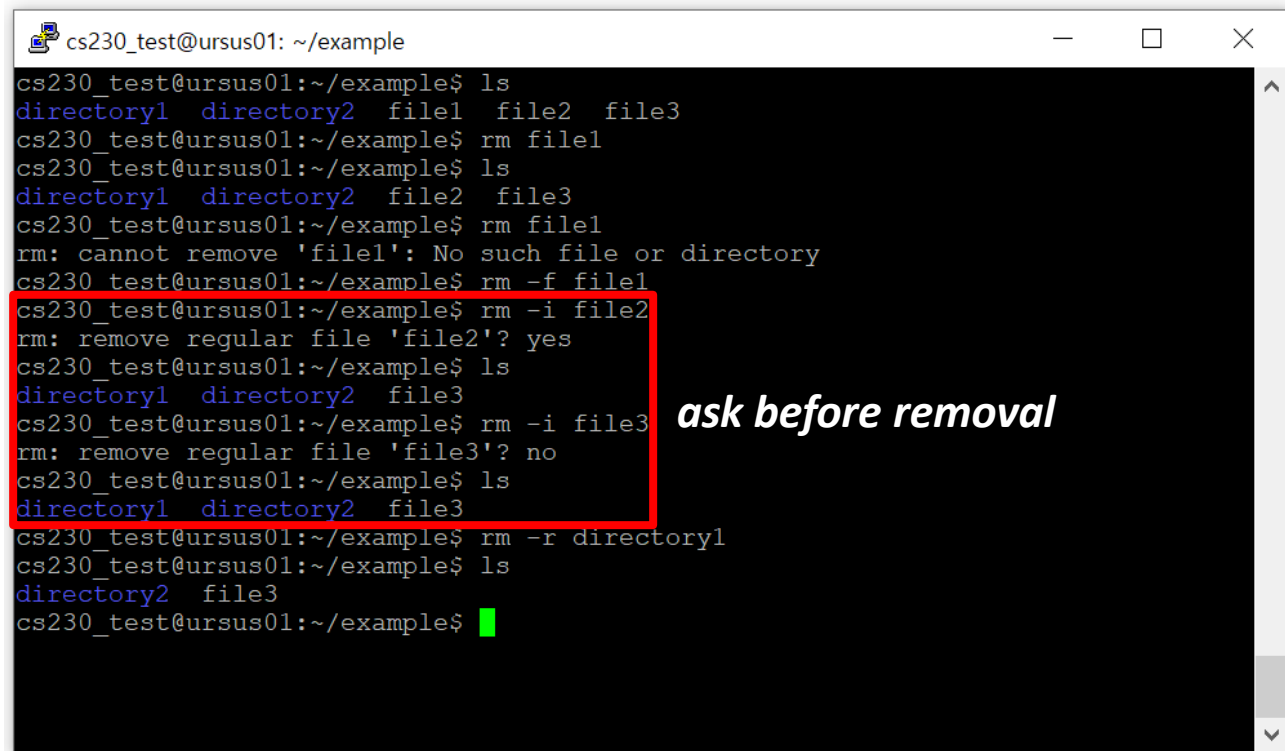


```
cs230_test@ursus01: ~/example
cs230_test@ursus01:~/example$ ls
directory1  directory2  file1  file2  file3
cs230_test@ursus01:~/example$ cp file1 file4
cs230_test@ursus01:~/example$ ls
directory1  directory2  file1  file2  file3  file4
cs230_test@ursus01:~/example$ cp file1 directory1/
cs230_test@ursus01:~/example$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

*copied 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



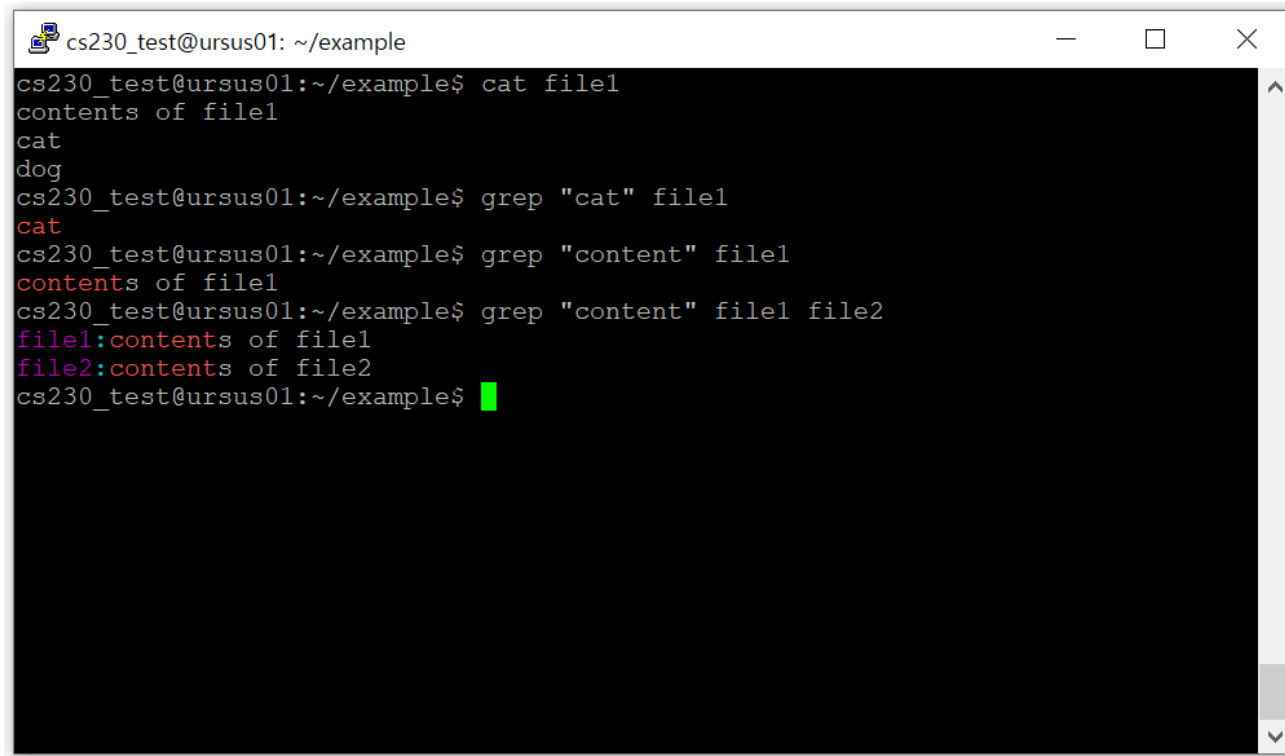
```
cs230_test@ursus01: ~/example
cs230_test@ursus01:~/example$ ls
directory1 directory2 file1 file2 file3
cs230_test@ursus01:~/example$ rm file1
cs230_test@ursus01:~/example$ ls
directory1 directory2 file2 file3
cs230_test@ursus01:~/example$ rm file1
rm: cannot remove 'file1': No such file or directory
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 terminal window titled 'cs230\_test@ursus01: ~/example' with standard window controls. The terminal shows a sequence of commands and their outputs. First, 'cat file1' is run, outputting 'contents of file1', 'cat', and 'dog'. Then, 'grep "cat" file1' is run, outputting 'cat' in red. Next, 'grep "content" file1' is run, outputting 'contents of file1' in red. Finally, 'grep "content" file1 file2' is run, outputting 'file1:contents of file1' and 'file2:contents of file2' in red. The prompt 'cs230\_test@ursus01:~/example\$' is followed by a green cursor.

```
cs230_test@ursus01: ~/example
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$
```

# 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!

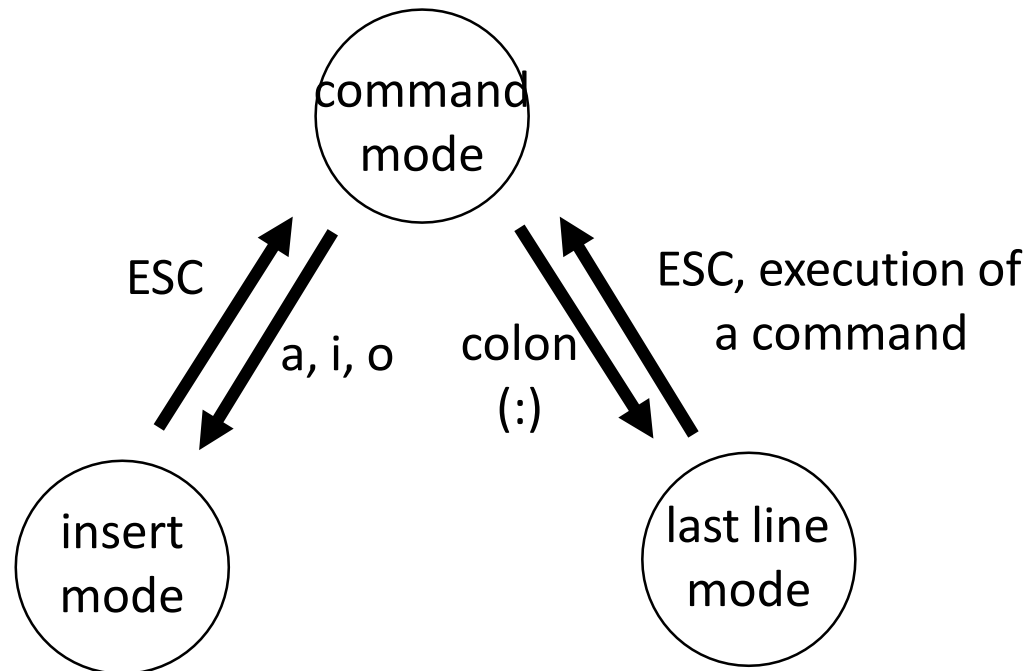
type  :help iccf<Enter>          for information

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



# vi / vim graphical cheat sheet

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 "hard" bol             | - prev line       | = auto <sup>3</sup> 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 <sup>1,3</sup> | 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 <sup>1</sup> | bol/ goto col     |                            |
| a append                  | s subst char      | d delete <sup>1,3</sup> | f find char      | g extra cmds <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.)    |                          |                   |                            |
| Z extra <sup>5</sup> cmds | X delete char     | c change <sup>1,3</sup> | v visual mode    | b prev word               | n next (find)         | m set mark     | , reverse t/T/f/F         | . repeat cmd          | / find           |                          |                   |                            |

**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: quux(**foo**, **bar**, **baz**);

WORDS: quux(**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

|                          |                     |                          |                    |                          |                     |                    |                             |                         |                                  |                           |                      |                                      |
|--------------------------|---------------------|--------------------------|--------------------|--------------------------|---------------------|--------------------|-----------------------------|-------------------------|----------------------------------|---------------------------|----------------------|--------------------------------------|
| ~ 대소문자<br>전환             | ! 외부<br>명령          | @. 매크로<br>실행             | # 이전<br>검색         | \$ 행 끝으로<br>이동           | % 짝 괄호<br>찾기        | ^ 행의<br>첫 글자       | & :s<br>반복                  | * 다음<br>검색              | ( 문장<br>시작                       | ) 앞으로<br>문장<br>끝          | - 아래<br>행<br>이전<br>행 | + 다음<br>행<br>자동<br>들여쓰기 <sup>3</sup> |
| 1                        | 2                   | 2                        | 3                  | 4                        | 5                   | 6                  | 7                           | 8                       | 9                                | 0                         |                      |                                      |
| Q 실행<br>모드               | W 다음<br>단어<br>(의미상) | E 단어<br>(의미상)<br>끝으로     | R 수정<br>모드         | T. 행에서<br>후방<br>검색       | Y 행 단위<br>복사        | U 행 단위<br>실행<br>취소 | I 행 시작에<br>삽입               | O 행 위에<br>삽입            | P 커서<br>이전에<br>붙여넣기              |                           | { 문단<br>시작           | } 문단<br>끝                            |
| q. 매크로<br>기록             | w 다음<br>단어          | e 단어<br>끝으로              | r. 한 문자<br>교체      | t. 행에서<br>전방<br>검색       | y 복사 <sup>1 3</sup> | u 실행<br>취소         | i 커서 앞에<br>삽입               | o 행 아래에<br>삽입           | p 커서<br>이후에<br>붙여넣기 <sup>1</sup> |                           | [                    | ]                                    |
| A 행 끝에<br>삽입             | S 삭제 후<br>입력 모드     | D 행<br>삭제                | F. 행에서<br>후방<br>검색 | G 파일<br>끝으로<br>이동        | H 화면<br>상단          | J 아래 행<br>합치기      | K 도움말                       | L 화면<br>하단              | : 명령행<br>모드                      | " 레지스터 <sup>1</sup><br>지정 | 열<br>이동              |                                      |
| a 커서 뒤에<br>삽입            | s 삭제 후<br>입력 모드     | d 삭제 <sup>1 3</sup>      | f. 행에서<br>전방<br>찾기 | g. 확장 <sup>6</sup><br>명령 | h ←                 | j ↓                | k ↑                         | l →                     | ; t/T/t/F<br>반복                  | ' . 표식으로<br>이동            | \ 사용하지<br>않음         |                                      |
| Z. 종료 <sup>4</sup>       | X 백스<br>페이스         | C 행<br>끝까지<br>바꾸기        | V 비주얼<br>라인<br>모드  | B 이전<br>단어<br>(의미상)      | N 이전<br>(찾기)        | M 화면<br>가운데        | < 내어<br>쓰기 <sup>3</sup>     | > 들어<br>쓰기 <sup>3</sup> | ? 찾기<br>(위로)                     |                           |                      |                                      |
| z. 확장 <sup>5</sup><br>명령 | x 글자<br>삭제          | c 바 <sup>1 3</sup><br>꾸기 | v 비주얼<br>모드        | b 이전<br>단어               | n 다음<br>(찾기)        | m 표식<br>설정         | , t/T/t/F<br>반대<br>방향<br>검색 | . 명령<br>반복              | / . 찾기<br>(아래로)                  |                           |                      |                                      |

### 동작

커서를 이동하거나, 연산자가 동작할 범위를 지정합니다.

### 명령

바로 동작하는 명령입니다. 빨간색은 입력 모드로 변경됩니다.

### 오퍼레이션 팬딩 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  
:ef f 파일 열기  
:%s/x/y/g 파일 전체에서 'x'를 'y'로 교체  
:h name name 명령에 대한 도움말  
:new 새 파일

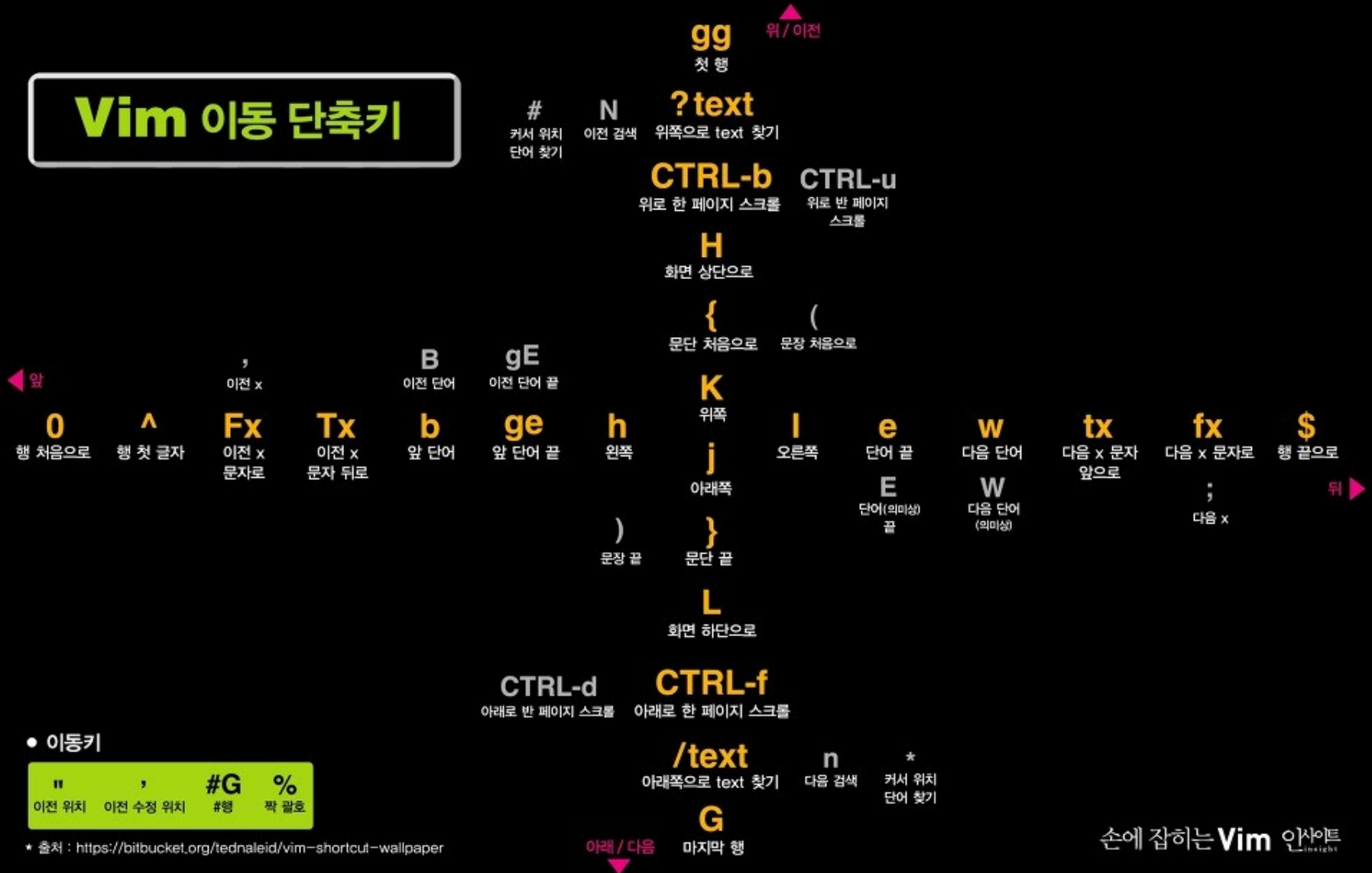
### • 일반 모드의 주요 명령어

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

### • 참고

- 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.
- 명령어 입력 전 숫자를 지정하면, 해당 숫자만큼 명령어가 반복됩니다.
- 연속으로 입력하면, 현재 행에 반영됩니다. 예를 들어 dd는 현재 행이 지워집니다.
- ZZ는 저장 후 종료, ZQ는 저장 없이 종료입니다.
- zt는 커서가 위치한 부분을 화면 상단으로 스크롤합니다. zb는 바닥으로, zz는 가운데로 스크롤합니다.
- gg는 커서를 파일 처음으로 이동합니다. gf는 커서 위치의 파일명을 인식하여 파일을 엽니다.

# Vim 이동 단축키



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**

**tart.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>