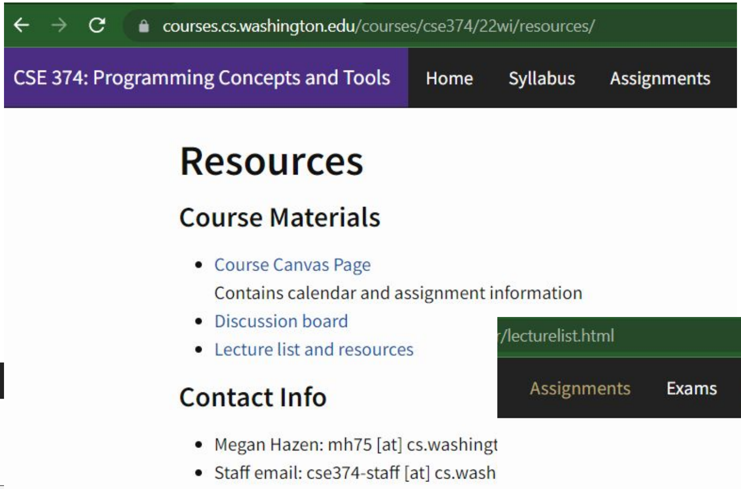
CSE 374 Lecture 2

Notes


courses.cs.washington.edu/courses/cse374/22wi/resources/

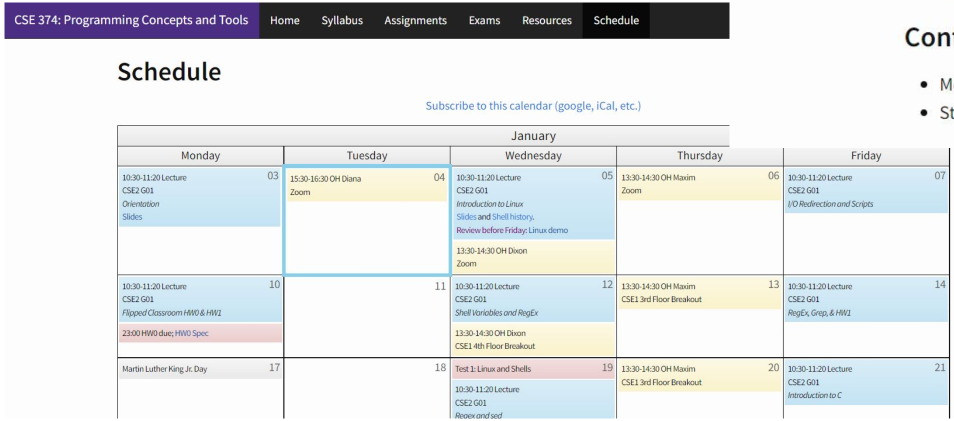
Resources

Course Materials

- Course Canvas Page
  Contains calendar and assignment information
- Discussion board
- Lecture list and resources

Contact Info

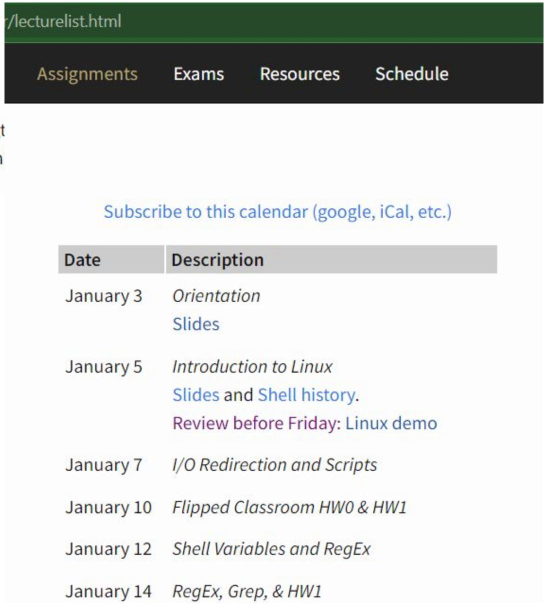
- Megan Hazen: mh75 [at] cs.washingt
- Staff email: cse374-staff [at] cs.wash



Schedule

Subscribe to this calendar (google, iCal, etc.)

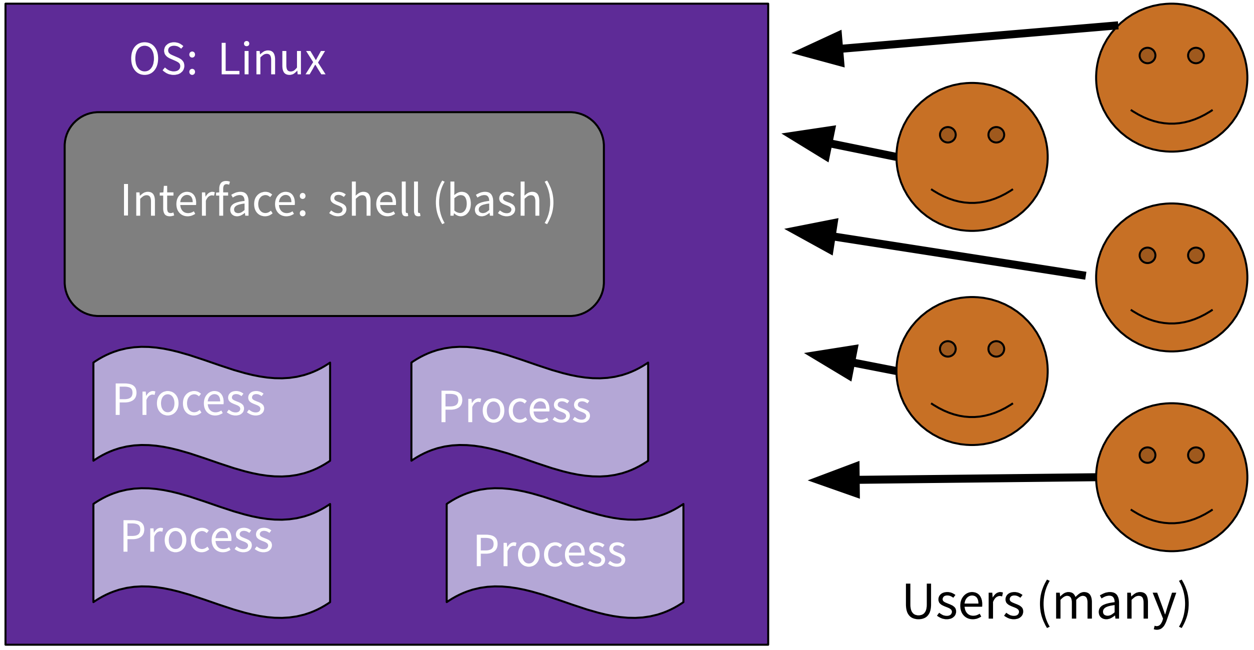
January				
Monday	Tuesday	Wednesday	Thursday	Friday
03 10:30-11:20 Lecture CSE2 G01 *Orientation* Slides	04 15:30-16:30 OH Diana Zoom	05 10:30-11:20 Lecture CSE2 G01 *Introduction to Linux* Slides and Shell history. Review before Friday: Linux demo; 13:30-14:30 OH Dixon Zoom	06 13:30-14:30 OH Maxim Zoom	07 10:30-11:20 Lecture CSE2 G01 *I/O Redirection and Scripts*
10 10:30-11:20 Lecture CSE2 G01 *Flipped Classroom HW0 & HW1*; 23:00 HW0 due; HW0 Spec	11	12 10:30-11:20 Lecture CSE2 G01 *Shell Variables and RegEx*; 13:30-14:30 OH Dixon CSE1 4th Floor Breakout	13 13:30-14:30 OH Maxim CSE1 3rd Floor Breakout	14 10:30-11:20 Lecture CSE2 G01 *RegEx, Grep, & HW1*
17 Martin Luther King Jr. Day	18	19 Test 1: Linux and Shells; 10:30-11:20 Lecture CSE2 G01 *Regex and sed*	20 13:30-14:30 OH Maxim CSE1 3rd Floor Breakout	21 10:30-11:20 Lecture CSE2 G01 *Introduction to C*


/lecturelist.html

Subscribe to this calendar (google, iCal, etc.)

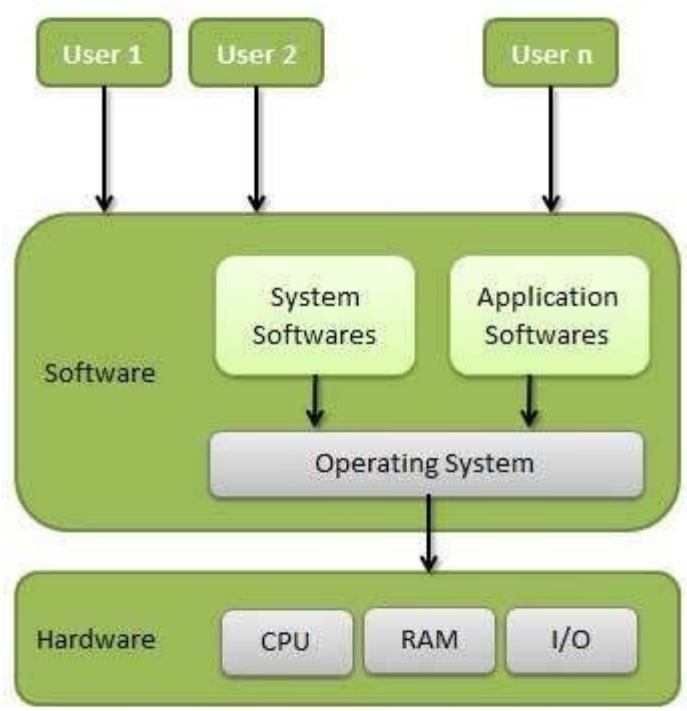
Date	Description
January 3	*Orientation* Slides
January 5	*Introduction to Linux* Slides and Shell history. Review before Friday: Linux demo
January 7	*I/O Redirection and Scripts*
January 10	*Flipped Classroom HW0 & HW1*
January 12	*Shell Variables and RegEx*
January 14	*RegEx, Grep, & HW1*

Computer Model



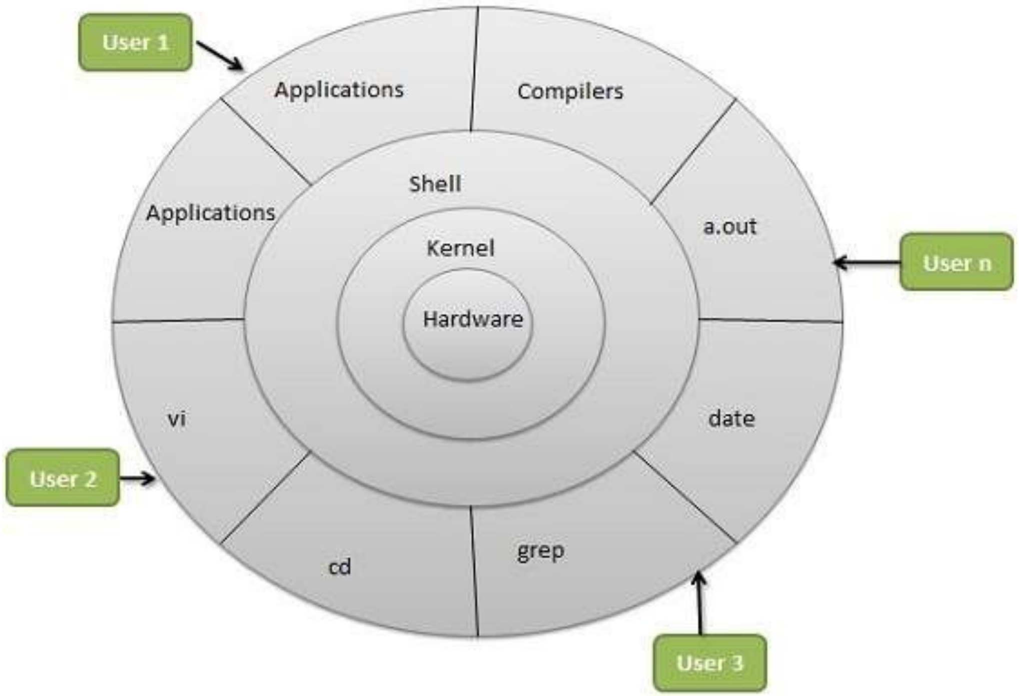
- One OS (CentOs) controls the computer.
- One filesystem stores data.
- Many processes are run. (A program runs one or many processes.)
- A shell is one process that allows for command line interface.
- Many users

What is the OS?



- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Linux Model



Linux -
Portable; multi-user

Includes

- Hardware layer (drivers, etc.)
- Kernel (does all the hardware interaction)
- Shell (provides user friendly interface to kernel)
- Processers (various programs)
- Users - multiple users run processes

http://www.tldp.org/LDP/intro-linux/html/chap_01.html

Linux & Shells

Text is efficient - typing is fast, and there aren't big image objects to pass around

Scripting makes it easy to automate text based interfaces

Linux *does* have a graphical interface

Windows and MacOS *do* have shell interfaces

Most power users use BOTH

You could use any distribution of Linux that is up-to-date. Using CSE machines ensures consistency.

(What a distribution? Something like a 'flavor', or a branded implementations. Distributions vary somewhat.)

Processes & the Shell

"On a UNIX system, everything is a file; if something is not a file, it is a process."

Shell essentially runs programs, or processes. Shell *is* a process, and has a state.

Usually launch a process, and return to shell when done.

Each process has own memory stream and I/O

Stdin (keyboard), stdout (console), stderr

Many processes have options

‘&’ runs process in the background

‘fg’, ‘bg’, top, kill

Step through a script with built-in ‘source’

Can redirect input and output (‘<’, ‘>’)

Getting Started with Linux

Use a virtual machine

Or

Log on to Cancun

(CSE 374's 'flavors' of Linux)

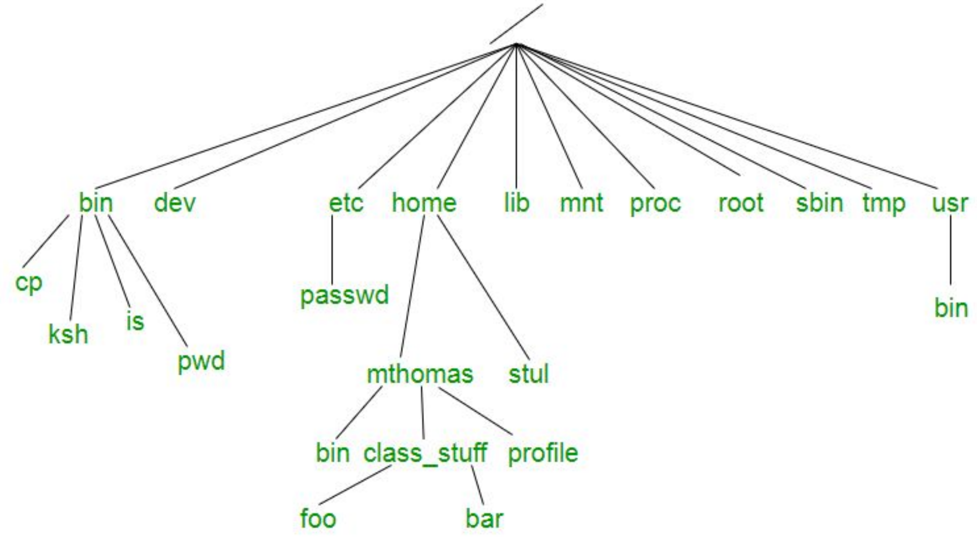
https://courses.cs.washington.edu/courses/cse374/22wi/resources/linux.html

Log-in and get a 'shell'

- Shell - text based interface
- Specifically 'bash'

Everyone should have a cancun account - send email to cse374-staff if you can not access yours.

File Systems
(Processes interact with data, stored in a file system)



- ❑ File systems are trees
- ❑ (or directed acyclic graphs)
- ❑ A file (or directory) is specified by its path from the top (‘/’)
- ❑ Can be specified absolutely or
- ❑ Relatively (from current location)
  - ❑ This directory ‘./’
  - ❑ One directory up ‘../’
- ❑ You have access to your ‘home’ directory (‘~’)

More: https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html
Also true on Windows, btw, although the structure and some notation is different.
Demo - whoami, pwd, ls, mkdir, cd, cp, mv, rm, less, more
http://www.tldp.org/LDP/intro-linux/html/sect_03_01.html

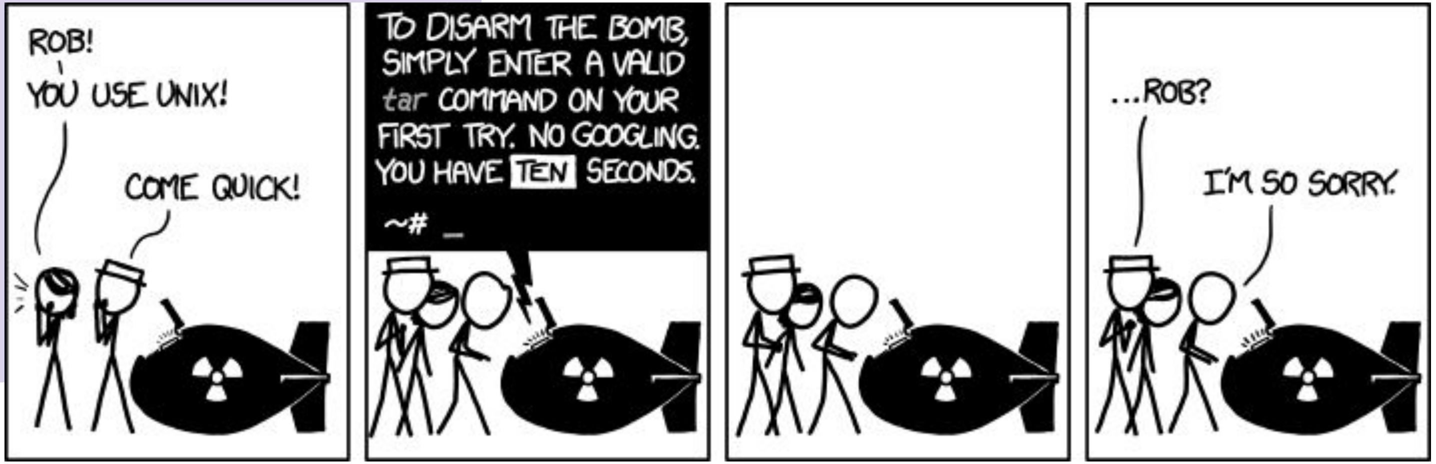
Getting Help

Most commands: ‘man ls’

Also “--help”

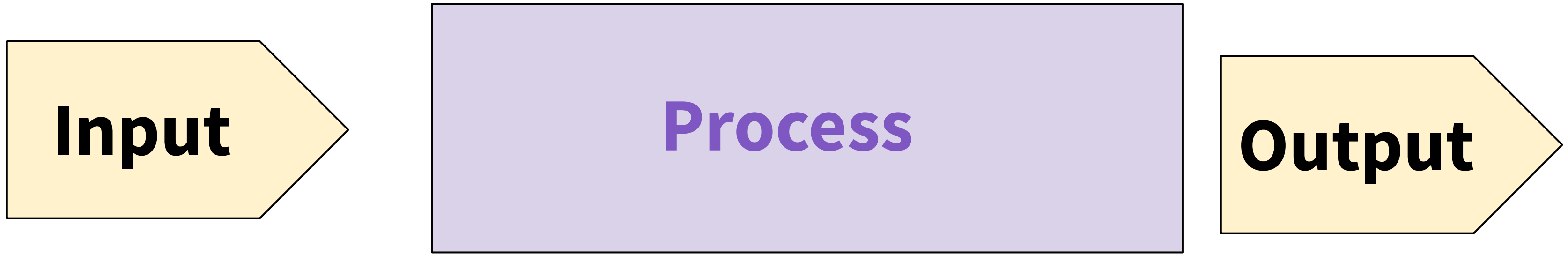
Look for keyword: ‘man -k’

http://www.tldp.org/LDP/intro-linux/html/sect_02_03.html


ROB!
YOU USE UNIX!
COME QUICK!
TO DISARM THE BOMB, SIMPLY ENTER A VALID tar COMMAND ON YOUR FIRST TRY. NO GOOGLING. YOU HAVE TEN SECONDS.
~# _
...ROB?
I'M SO SORRY.

Bash (shell) Language

- Bash acts as a language interpreter
  - Commands are subroutines with arguments
  - Bash interprets the arguments & calls subroutine
  - Bash also has its own variables and logic



BASH applies its own processing to the I/O text - ‘globbing’

Special Characters

- Directory Shortcuts
  - ~uname or ~
  - ./ or ../
- Wildcards - *Globbing*
  - 0 or more chars: *
  - Exactly 1 char: ?
  - Specified chars: [a-f]

History, or ‘!’

Special Characters

! > < & | * ~ [] “ ‘ ` $ /

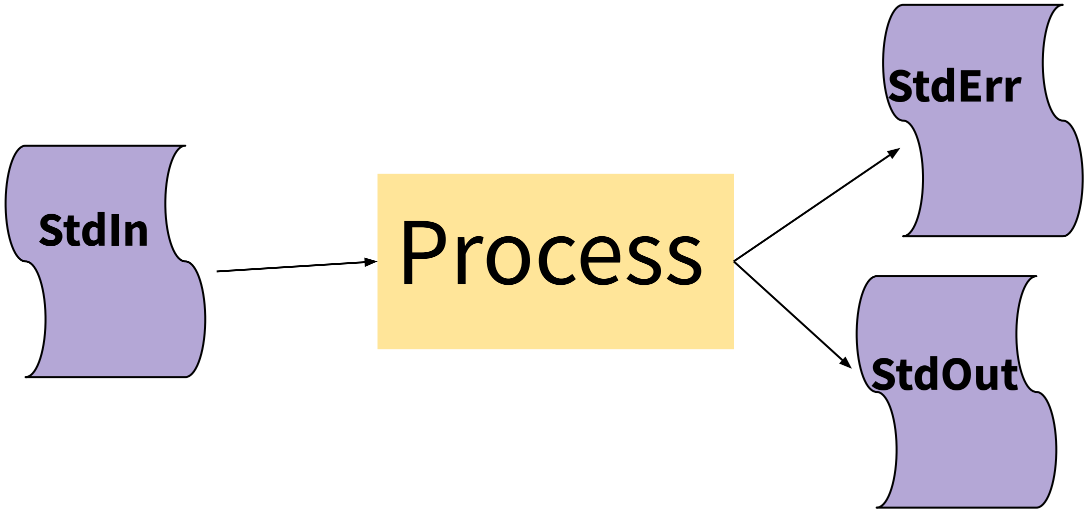
What do they all mean?

\ is escape character

“string” → Would substitute things like $VAR

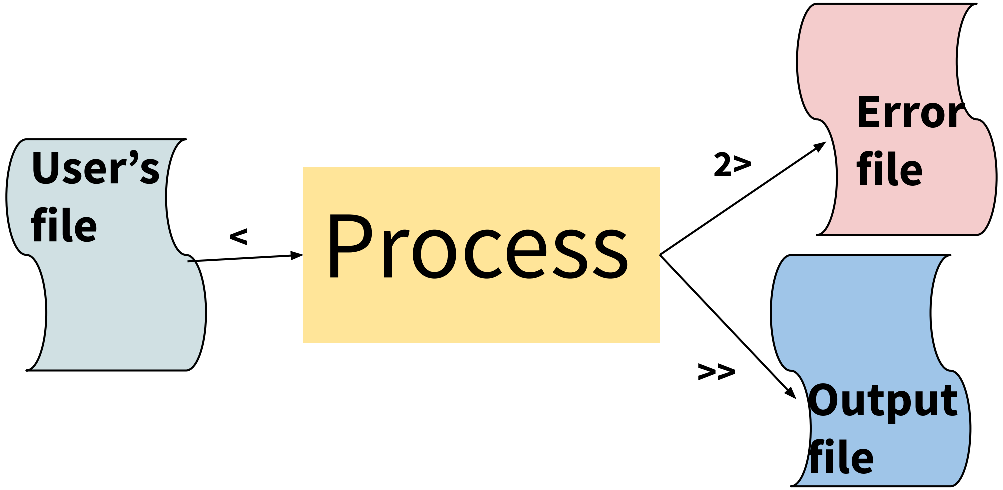
‘string’ → Suppresses substitutions

Processes all can take INPUT from one source, the default being StdIn.



Processes have two OUTPUT destinations, the default being StdOut and StdErr. You can think of these as two potential files to which a processes can write.

But, instead of using StdIn you can use any file, and 'redirect' it in by using the '<' symbol (pointing towards process).



You can also write to different files instead of StdErr or StdOut. The '>' symbol means to put in an new file, while '>>' means to append to the end of a file. The '2' specifies that you want iostream '2', or the error stream.

Shell Behavior

All redirection & string expansion or substitutions are done by the shell, before the command.

Command only sees resulting I/O streams.