CMSC 257: Computer Systems - Spring 2020

Instructor:

Ahmet E Sonmez

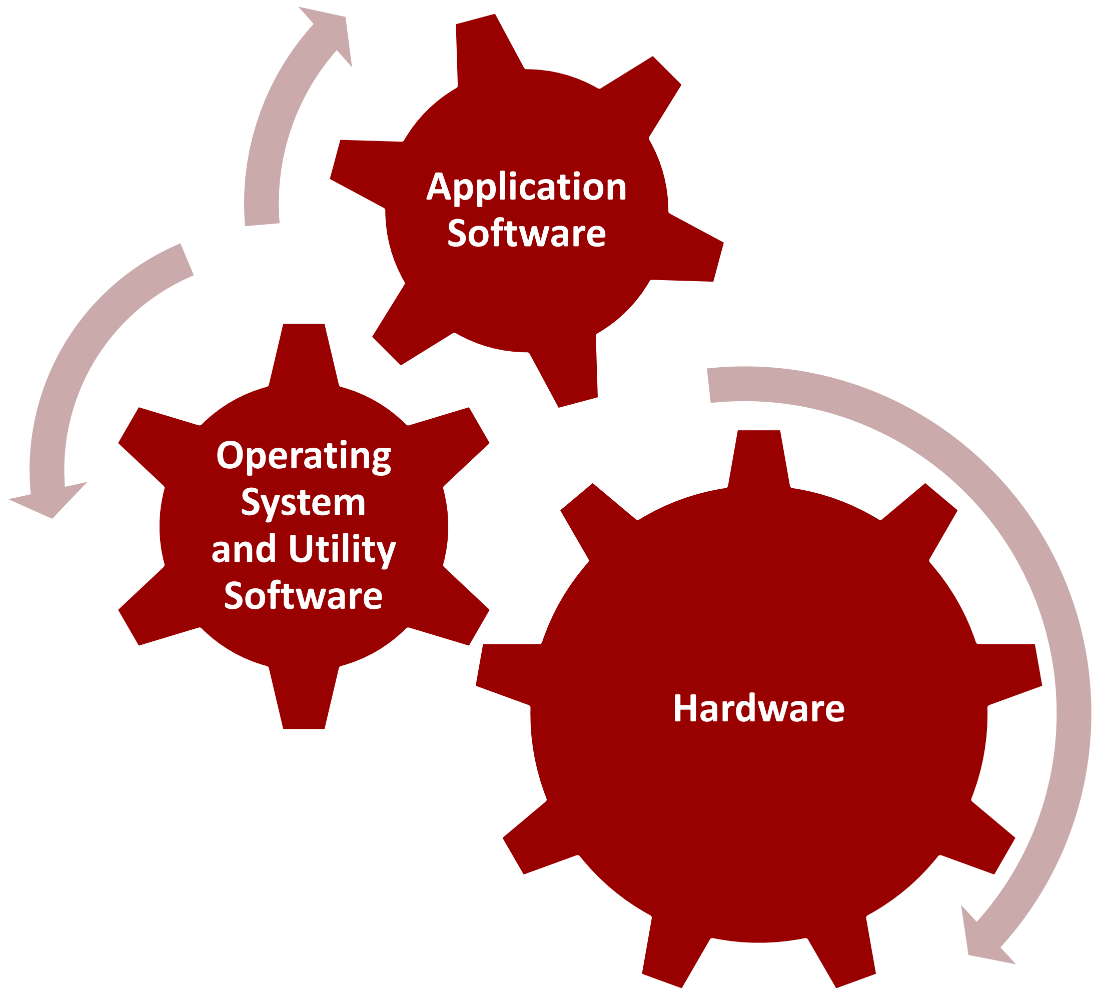
Course Description

- **What is a system?**

Dictionary Definition:

- A set of things working together as parts of a mechanism.

What is computer system?

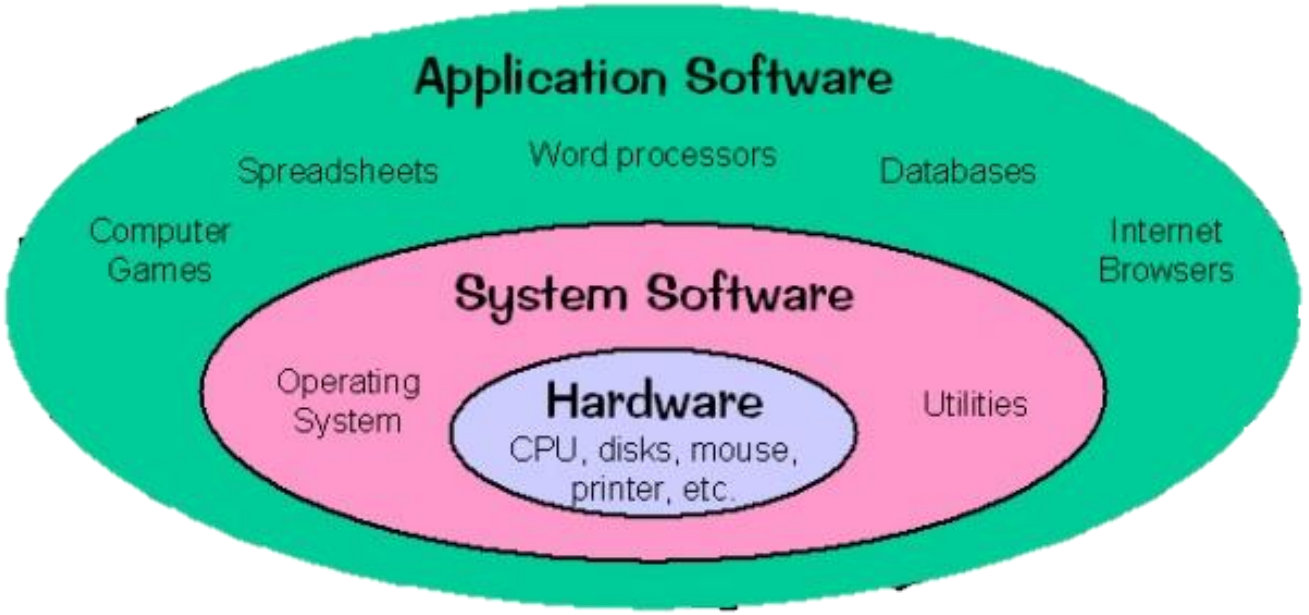


Systems Programming

- **Application programming aims to produce software which provides services to the user directly**
  - Word Processor
  - Games

- **Systems programming aims to produce software and software platforms which provide services to other software**
  - Operating Systems
  - Game Engines : Unreal, Unity

- **System programming requires a great degree of hardware awareness!**



System Programming

- **Usually low level programming language is used so that:**
  - Programs can operate in resource-constrained environments
  - Programs written to be efficient with little runtime overhead.
  - Programs may use direct and "raw" control over memory access
  - The programmer may write parts of the program directly in assembly language

- **An emerging high level alternative is Rust Programming Language**
  - https://www.rust-lang.org/

- **System Software Examples**
  - OS Kernels
  - Device Drivers
  - Compilers, Debuggers

Course Theme

- **We will focus mostly on the System Programming aspect of the Computer Systems**
- **What is not covered:**
  - Instruction Set architecture is covered in Computer Organization.
  - Memory management is covered in Operating Systems, but we will cover how a program uses virtual memory
- **What is covered:**
  - You will get familiar to Linux environment
  - You will learn C programming language, elevate your programming skills beyond an introductory level
  - You will be introduced to some operating system calls

Course Theme

- **We will use system tools during programming development**
  - Using system libraries saves programmer time and effort
  - Using debugger saves time in fixing errors
  - Using profiling tools help us identify program bottlenecks
  - Using System calls provide access to the core functions of the operating system
    - Memory management, file access, process management, interposes communication
  - Using shell environment provides rich capabilities, options and configurability

Outline

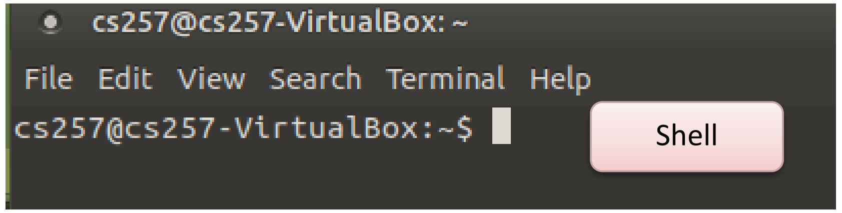
- ➢ Course Content
- ■ Realities you will be introduced
- ■ Unix/Linux and variants
- ■ Command Line Interface
- ■ Text Editors

Course Content

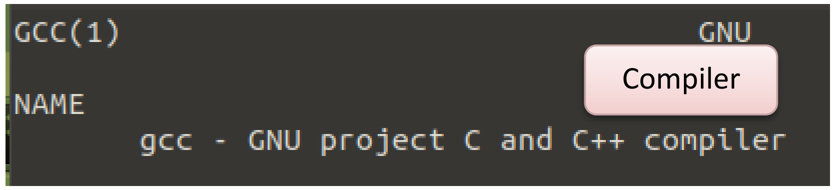
- **Systems Programming Toolset and Basics**
- **Data Representation**
- **Introduction to C**
  - Arrays and Pointers in C
  - Types, Structs and Unions in C
  - Strings
  - Memory Management
- **Processes and UNIX Signals**
- **Concurrency**
- **Input/Output**

Toolset of a System Programmer

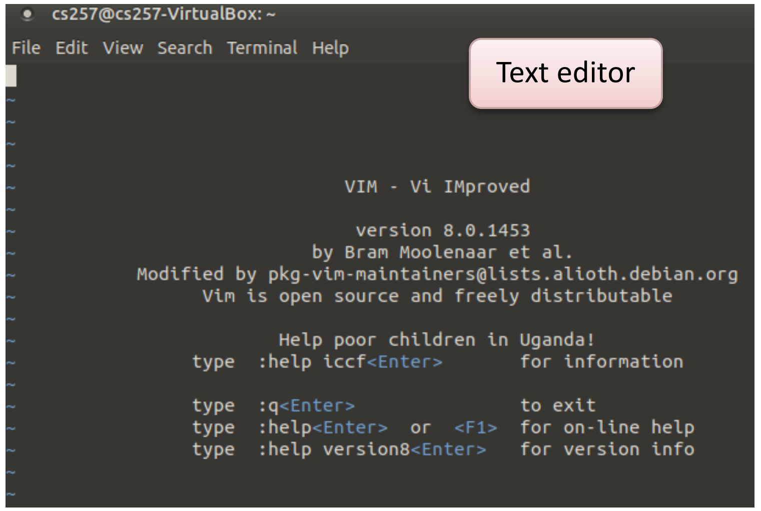
- Shell
- Text Editor
- Compiler
- Debugger


cs257@cs257-VirtualBox: ~
File Edit View Search Terminal Help
Shell

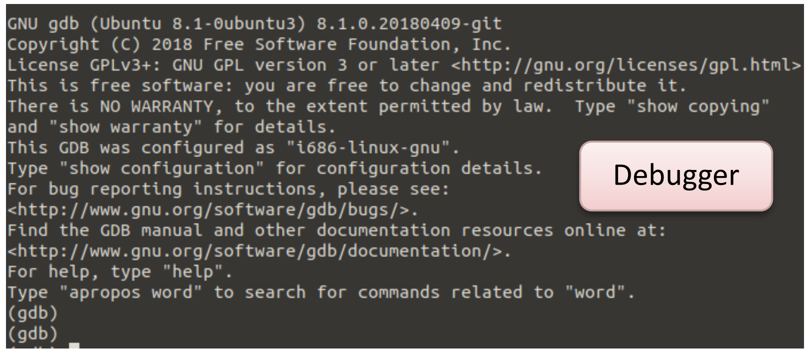
```
cs257@cs257-VirtualBox:~$
```



```
GCC(1)                                                    GNU

NAME
       gcc - GNU project C and C++ compiler
```

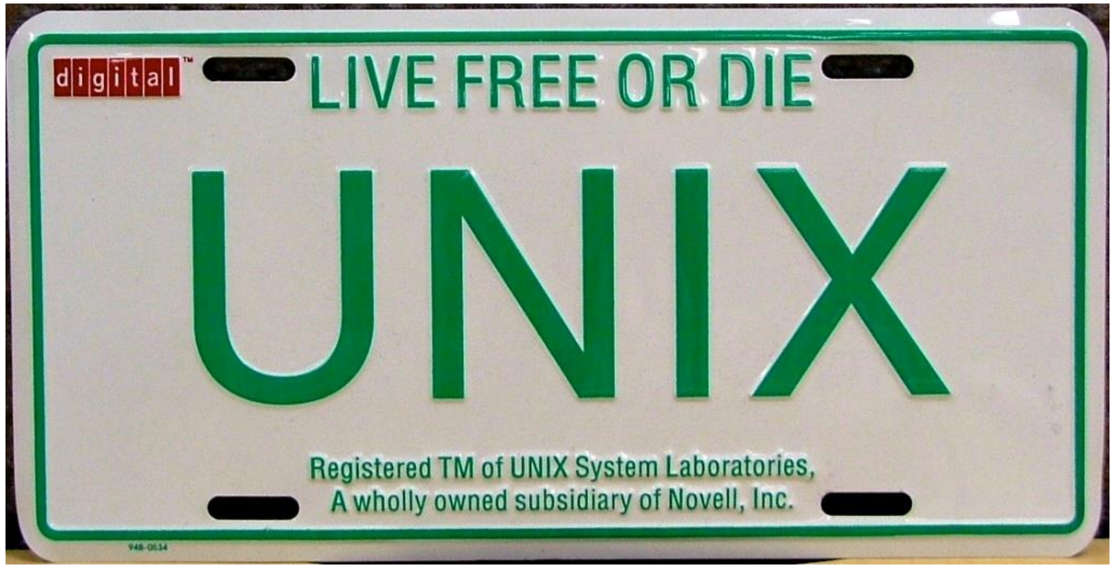

cs257@cs257-VirtualBox: ~
File Edit View Search Terminal Help
Text editor

```
~
~
~
~
~                         VIM - Vi IMproved
~
~                          version 8.0.1453
~                     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 version8<Enter>   for version info
~
~
```



```
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)
(gdb)
```

Unix Operating System


digital™
LIVE FREE OR DIE
UNIX
Registered TM of UNIX System Laboratories,
A wholly owned subsidiary of Novell, Inc.

Promotional license plate by Digital Equipment Corporation
Borrowed from Wikipedia.org

The operating system was originally written in assembly language, but in 1973, Version 4 Unix was rewritten in C

C Programming

- Java vs C
- Bit/Byte Operations
- Arrays and Pointers in C
- Types, Structs and Unions in C
- Strings
- Memory Management

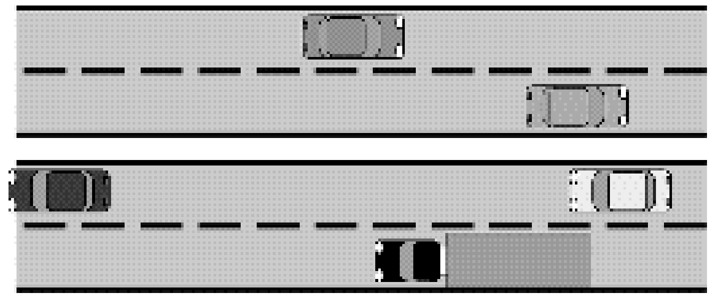
Unix Signals

- Signals are software interrupts sent to a process to indicate that an important event has occurred.
- The events can vary from user requests to illegal memory access errors. Some signals, such as the interrupt signal, indicate that a user has asked the program to do something that is not in the usual flow of control.
- Signals are also used for inter-process communication

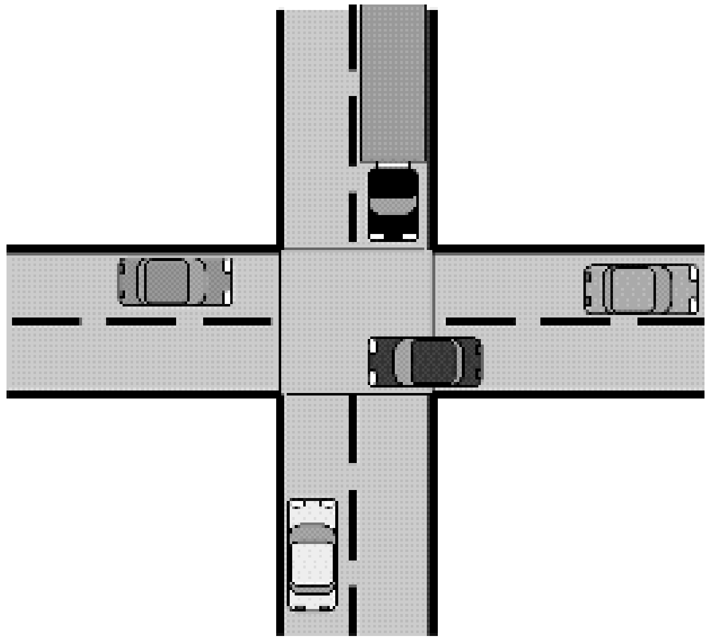
Concurrency

- Processes and Threads

No Interaction



Interaction



https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/process/workflow/ana_desi/co_cncry.htm

I/O

- Keyboard, processes, devices, storage, network
- Can be buffered or unbuffered
- Can be blocking or nonblocking
- I/O can be redirected
- Pipes take output from one program and use it as input for another
- Unix Access Policy

Shell Programming

- **When using an operating system we are directly/indirectly interacting with shell**
- **When you use terminal in Linux you can directly interact with shell**
- **Shell accept human readable commands from user and convert them something which kernel can understand**
- **Uses:**
  - Automating
  - System Administration
  - Prototyping

Outline

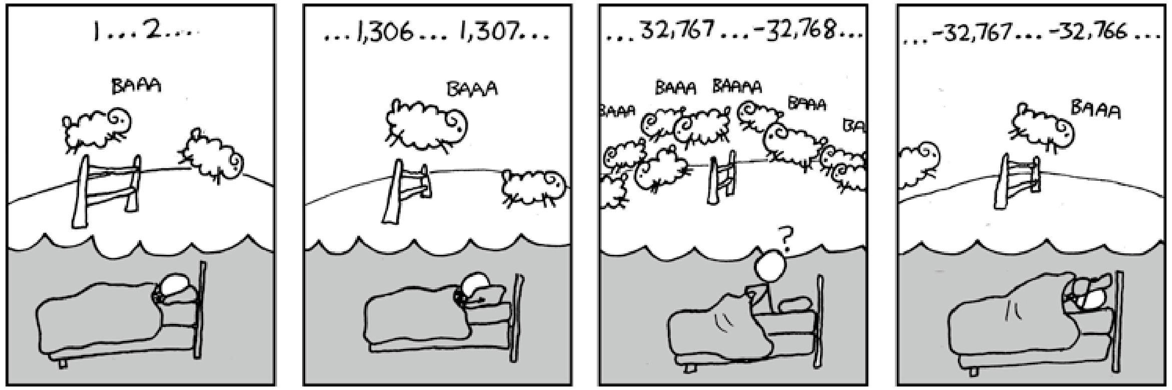
- Course Content
- ➢ Realities you will be introduced
- Unix/Linux and variants
- Command Line Interface
- Text Editors

Abstraction Is Good But Don’t Forget Reality

- **Useful outcomes from taking 257**
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
      - By understanding what is going on under the hood
    - Able to understand and tune for program performance,
      - Using profiling tools
  - Prepare for later “systems” classes
    - Compilers, Operating Systems, Networks, Computer Organization, etc.

Great Reality #1: ints are not Integers, floats are not Reals

- **Example 1: Is $x^2 \geq 0$?**
  - Float's: Yes!
  - Int's:
    - 40000 * 40000 = 1,600,000,000
    - 50000 * 50000 = ??



- **Example 2: Is (x + y) + z = x + (y + z)?**
  - Unsigned & Signed Int's: Yes!
  - Float's:
    - (1e20 + -1e20) + 3.14 --> 3.14
    - 1e20 + (-1e20 + 3.14) --> ??

Source: xkcd.com/571

Engineering Failures due to number representation

- **Patriot missile failure, Dhahran, 1991**
  - Reason: You cannot represent 1/10 in binary
  - Result: 28 Soldiers dead
  - More at:
    - http://www-users.math.umn.edu/~arnold//disasters/patriot.html

- **Ariane 5 failure, French Guiana, 1996**
  - Reason: number overflow (64 bit floating point to 16 bit integer type casting)
  - Result a $7 billion worth of fireworks
  - More at:
    - http://www-users.math.umn.edu/~arnold//disasters/aria

Great Reality #2: Memory Matters

Random Access Memory Is an Unphysical Abstraction

- **Memory is not unbounded**
  - It must be allocated and managed
  - Many applications are memory dominated
- **Memory referencing bugs are harmful and hard to detect**
  - It is possible to overwrite data
- **Memory performance is not uniform**
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements

Memory Referencing Bug Example

```c
typedef struct {
  int a[2];
  double d;
} struct_t;

double fun(int i) {
  struct_t s;
  s.d = 3.14;
  s.a[i] = 1073741824; /* Possibly out of bounds */
  return s.d;
}
```

```
fun(0)   →     3.14
fun(1)   →     3.14
fun(2)   →     3.1399998664856
fun(3)   →     2.00000061035156
fun(4)   →     3.14
fun(6)   →     Segmentation fault
```

In C you will not get array out of bounds exception

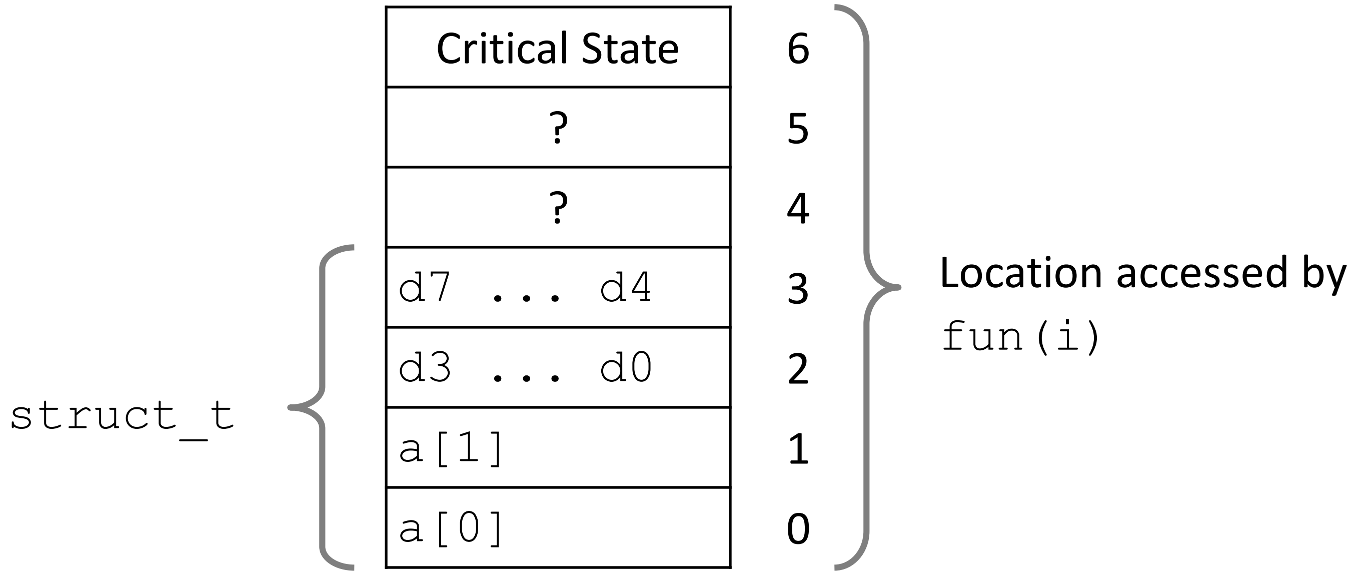
- Result is system specific

Memory Referencing Bug Example

```
typedef struct {
  int a[2];
  double d;
} struct_t;
```

```
fun(0)  ⟿  3.14
fun(1)  ⟿  3.14
fun(2)  ⟿  3.1399998664856
fun(3)  ⟿  2.00000061035156
fun(4)  ⟿  3.14
fun(6)  ⟿  Segmentation fault
```

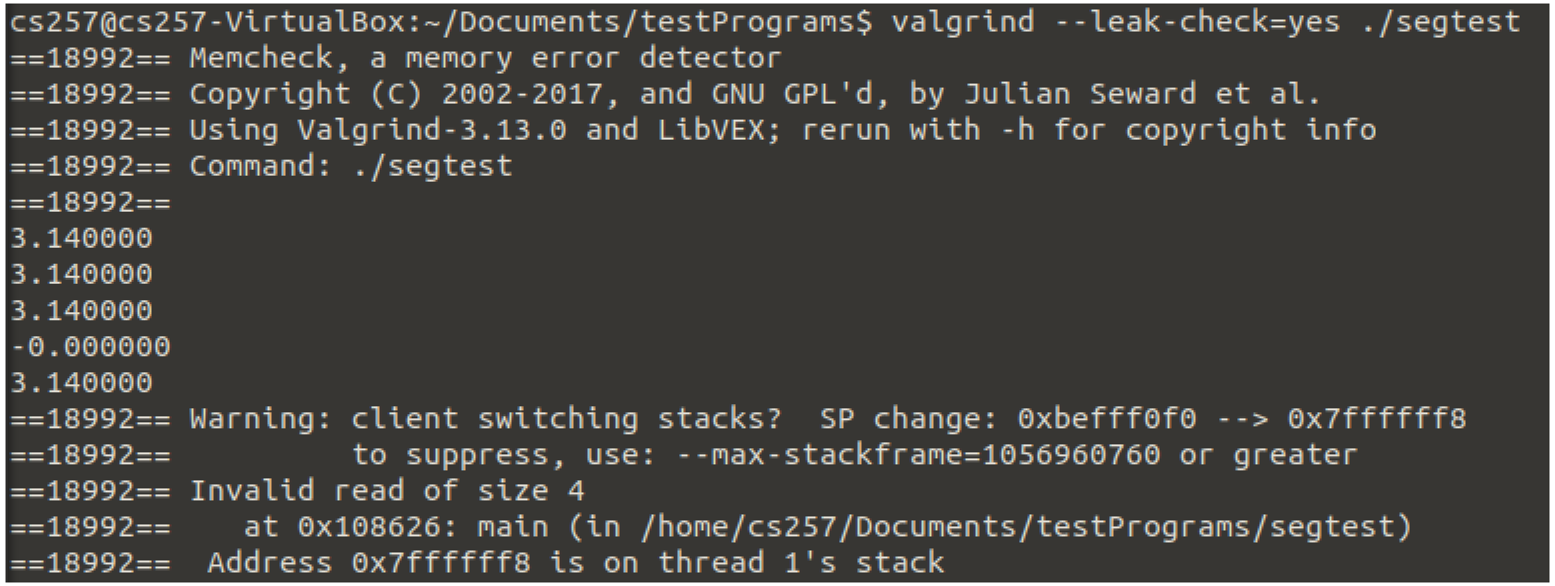
Explanation:



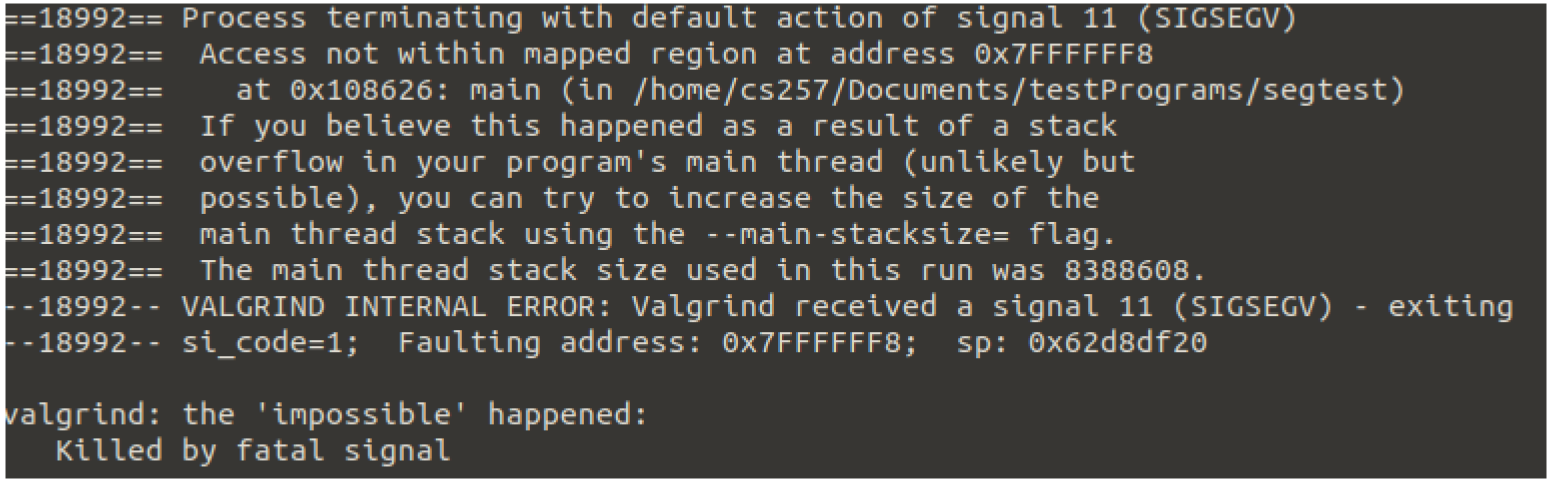
Memory Referencing Errors

- **C and C++ do not provide any memory protection**
  - Out of bounds array references
  - Invalid pointer values
  - Abuses of malloc/free
- **Can lead to nasty bugs**
  - Whether or not bug has any effect depends on system and compiler
  - Action at a distance
    - Corrupted object logically unrelated to one being accessed
    - Effect of bug may be first observed long after it is generated
- **How can I deal with this?**
  - Program in Java, Ruby, Python, ML, …
  - Understand what possible interactions may occur
  - Use or develop tools to detect referencing errors (e.g. Valgrind)

Valgrind example



```
cs257@cs257-VirtualBox:~/Documents/testPrograms$ valgrind --leak-check=yes ./segtest
==18992== Memcheck, a memory error detector
==18992== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==18992== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==18992== Command: ./segtest
==18992==
3.140000
3.140000
3.140000
-0.000000
3.140000
==18992== Warning: client switching stacks?  SP change: 0xbefff0f0 --> 0x7ffffff8
==18992==          to suppress, use: --max-stackframe=1056960760 or greater
==18992== Invalid read of size 4
==18992==    at 0x108626: main (in /home/cs257/Documents/testPrograms/segtest)
==18992==  Address 0x7ffffff8 is on thread 1's stack
```



```
==18992== Process terminating with default action of signal 11 (SIGSEGV)
==18992==  Access not within mapped region at address 0x7FFFFFF8
==18992==    at 0x108626: main (in /home/cs257/Documents/testPrograms/segtest)
==18992==  If you believe this happened as a result of a stack
==18992==  overflow in your program's main thread (unlikely but
==18992==  possible), you can try to increase the size of the
==18992==  main thread stack using the --main-stacksize= flag.
==18992==  The main thread stack size used in this run was 8388608.
--18992-- VALGRIND INTERNAL ERROR: Valgrind received a signal 11 (SIGSEGV) - exiting
--18992-- si_code=1;  Faulting address: 0x7FFFFFF8;  sp: 0x62d8df20

valgrind: the 'impossible' happened:
   Killed by fatal signal
```

Great Reality #3: There's more to performance than asymptotic complexity

- **Constant factors matter too!**
- **And even exact op count does not predict performance**
  - Easily see 10:1 performance range depending on how code written
  - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- **Must understand system to optimize performance**
  - How programs compiled and executed
  - How to measure program performance and identify bottlenecks

Memory System Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

4.3ms **2.0 GHz Intel Core i7 Haswell** **81.8ms**

- **Hierarchical memory organization**
- **Performance depends on access patterns**
  - Including how step through multi-dimensional array

Outline

- Course Content
- Realities you will be introduced
- ➢ Unix/Linux and variants
- Command Line Interface
- Text Editors

What is an OS?

- **Software that:**
- **Directly interacts with the hardware**
  - OS is trusted to do so; user-level programs are not
  - OS must be ported to new hardware so user-level programs become portable
- **Manages (allocates, schedules, protects) hardware resources**
  - Decides which programs can access which files, memory locations, screen, data etc., and when
- **Abstracts away messy hardware devices**
  - Provides high-level, convenient, portable abstractions
  - e.g., files vs. disk blocks
- **Monitors system activity**
  - Performance and Security

- **UNIX is a classical example of OS**

UNIX

- **Developed in 1969 at Bell Labs**
  - Originally intended for use as a programmer environment for developing multi-platform code and written in Assembly
  - In 1973 it is rewritten in C
  - Then adopted by universities and industry

Ken Thompson (sitting) and Dennis Ritchie working together - Wikipedia.org

- **C as a side product**

  Originally developed at Bell Labs by Dennis Ritchie between 1972 and 1973 to make utilities running on Unix

  First Ken Thompshon developed a new language B, it was slow. Then Richie started to improve B and developed C

Variations

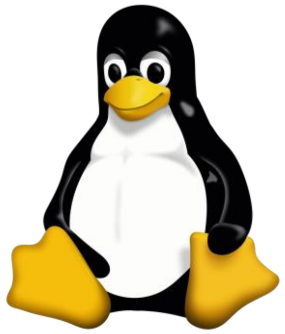
- **Today UNIX is a standard.**
  - There isn't a UNIX, but multiple implementations
    - Linux, BSD, Solaris, OS X, Android

- **Each UNIX usually develops its own niche or specialization**
  - Mac OS X: "User-friendly" desktop
  - Red Hat Linux: "Easy" workstation, simple server
  - Debian Linux: "Advanced" workstation and server
  - Ubuntu Linux: "User Friendly" and powerful server
  - Android: "Mobile" computing

LINUX History

- Linus Torvalds was a student in Helsinki, Finland, in 1991, when he started a project: writing his own operating system kernel
- Developed his kernel and combined with other system components from GNU project.
- Unix was propriety, GNU Linux project aimed to develop a UNIX-like OS that comes with source code that can be
  - Copied
  - Modified
  - Redistributed

Linus Torvalds


Linux mascot

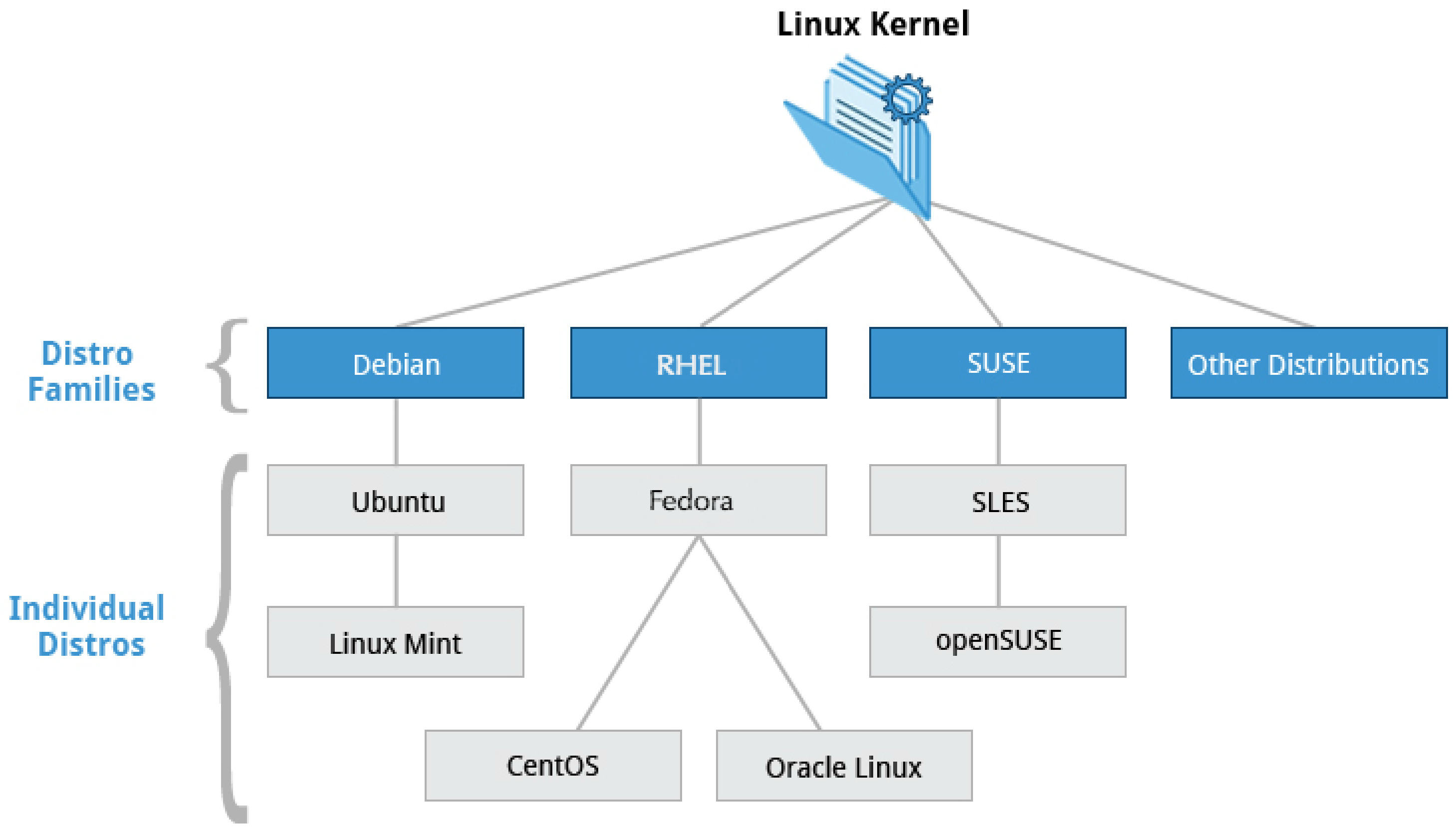
GNU mascot

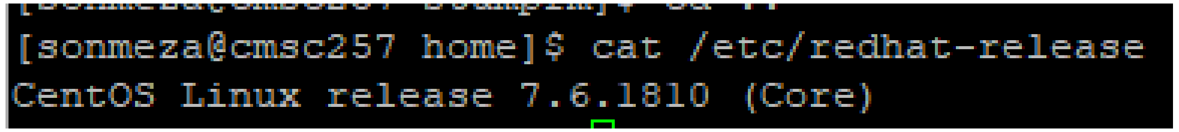
LINUX History

- Today, Linux powers more than half of the servers on the Internet, the majority of smartphones (via the Android system, which is built on top of Linux), and all of the world's most powerful supercomputers.
- It was written to be a free and open source system to be used in place of UNIX, which at the time was designed for computers much more powerful than PCs and was quite expensive.
- Linux was inspired by UNIX, but it is not UNIX.

Linux Distributions (Distros)

- There are 3 major distros
- Software package managers differ.





```
[sonmeza@cmsc257 home]$ cat /etc/redhat-release
CentOS Linux release 7.6.1810 (Core)
```

Desktop environments (flavors)

- **There are also desktop environments to choose from**
  - Gnome
    - https://ubuntugnome.org/
  - KDE
    - https://kubuntu.org/
  - Mate
    - https://ubuntu-mate.org/
  - More flavors for ubuntu at: https://ubuntu.com/download/flavours
    - Multimedia creation
    - Chinese users

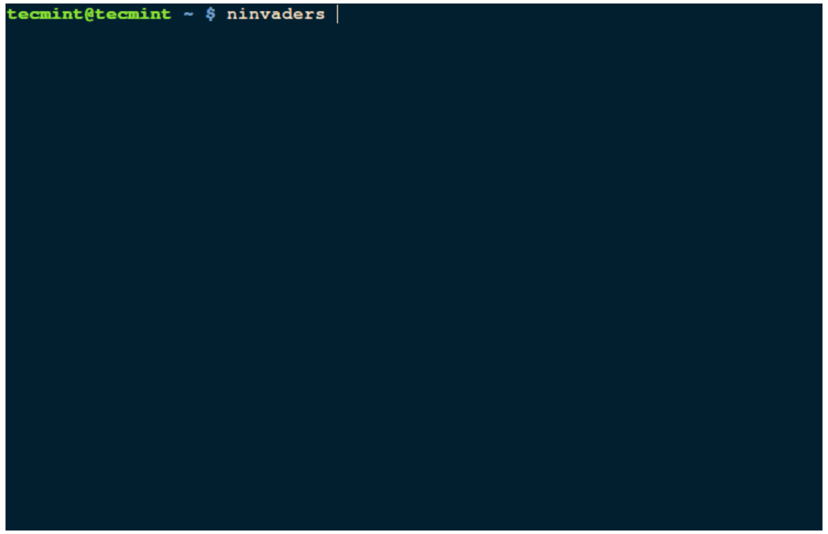
Outline

- Course Content
- Realities you will be introduced
- Unix/Linux and variants
- ➢ Command Line Interface
- Text Editors

Command line interface

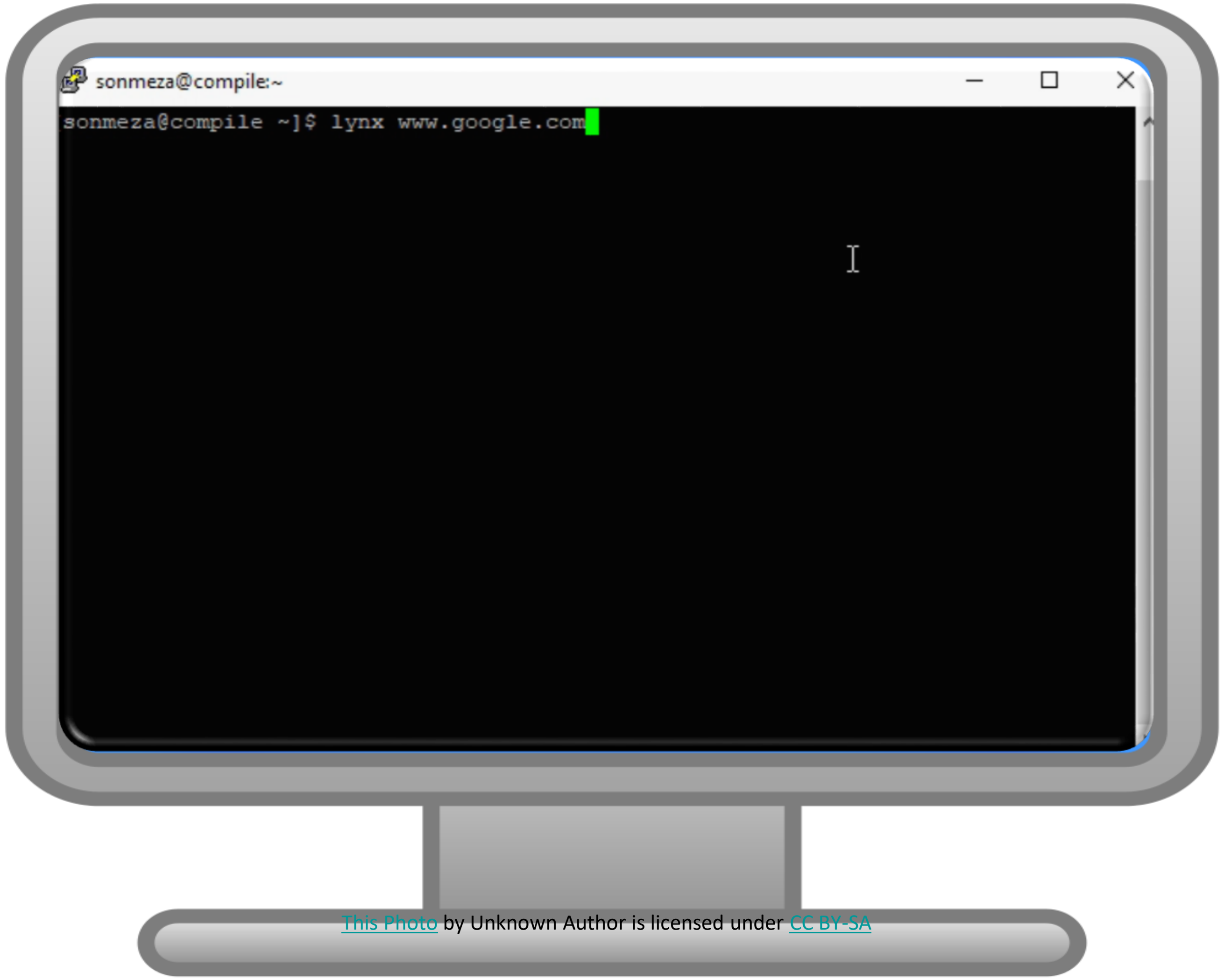
- **Command line? Why?**
  - Efficient and powerful
  - Scriptable
  - Simple and Reliable
    - Always works, even if everything is broken

- **What is it?**
  - Shell program (“bash” on Linux)
  - Interprets built-in commands
  - Runs other programs
  - Runs shell scripts



```
tecmint@tecmint ~ $ ninvaders
```

Command line is everything for some Linux admins

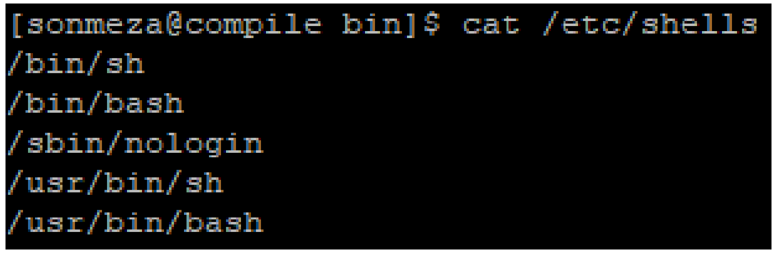


```
sonmeza@compile ~]$ lynx www.google.com
```

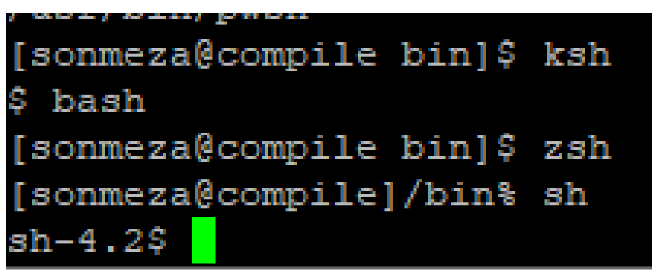
- A command line based browser:
  - lynx www.google.com

Shell

- An interface between the UNIX system and the user
- Used to call commands and programs
- An interpreter
- Powerful programming language
- Many available (bsh; ksh; csh; bash; tcsh; zsh)
- In the terminal you can run the shell you want by typing the shell name



```
[sonmeza@compile bin]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
```

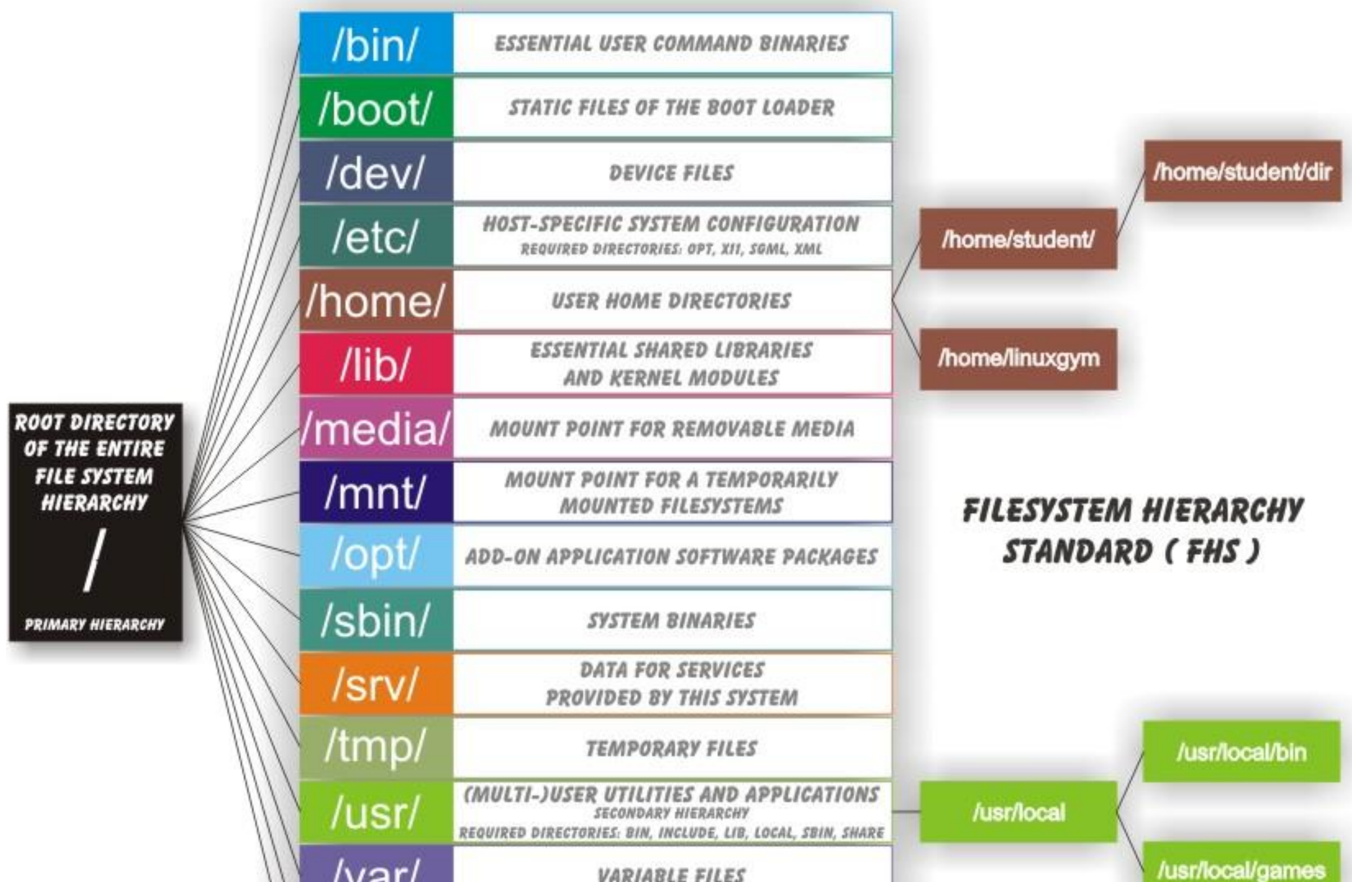
- 

```
[sonmeza@compile bin]$ ksh
$ bash
[sonmeza@compile bin]$ zsh
[sonmeza@compile]/bin% sh
sh-4.2$
```

- To see which one you are running type echo "$SHELL"

UNIX Directory Structure

- Unix uses a hierarchical directory structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.



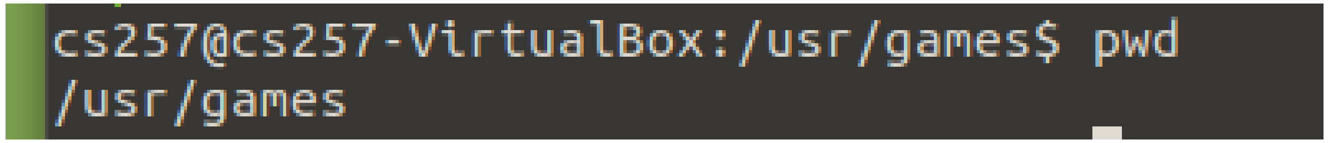
ROOT DIRECTORY OF THE ENTIRE FILE SYSTEM HIERARCHY
/
PRIMARY HIERARCHY
/bin/ ESSENTIAL USER COMMAND BINARIES
/boot/ STATIC FILES OF THE BOOT LOADER
/dev/ DEVICE FILES
/etc/ HOST-SPECIFIC SYSTEM CONFIGURATION
REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/ USER HOME DIRECTORIES
/lib/ ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/ MOUNT POINT FOR REMOVABLE MEDIA
/mnt/ MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/ ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/ SYSTEM BINARIES
/srv/ DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/ TEMPORARY FILES
/usr/ (MULTI-)USER UTILITIES AND APPLICATIONS
SECONDARY HIERARCHY
REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/ VARIABLE FILES
/home/student/
/home/student/dir
/home/linuxgym
FILESYSTEM HIERARCHY STANDARD (FHS)
/usr/local
/usr/local/bin
/usr/local/games

UNIX Directory Structure

Directory & Description
/ The root directory
/bin Where the executable files are located.
/dev Device drivers
/etc valid user lists, groups, ethernet, hosts,
/lib Contains shared library files and sometimes other kernel-related files
/boot Contains files for booting the system
/home Contains the home directory for users and other accounts
/mnt Used to mount other temporary file systems, such as CD, USB stick
/usr includes administrative commands, shared files, library files, and others
/var variable-length files such as log and print files

Directory Structure

- **Files are named by naming each containing directory starting at the root**
  - E.g   /etc/passwd
  - This is called pathname
- **Current Directory**
  - There is one current working directory
  - If you omit the leading / then path name is relative to the current working directory
  - Use pwd to find out where you are (prints current working directory)



```
cs257@cs257-VirtualBox:/usr/games$ pwd
/usr/games
```

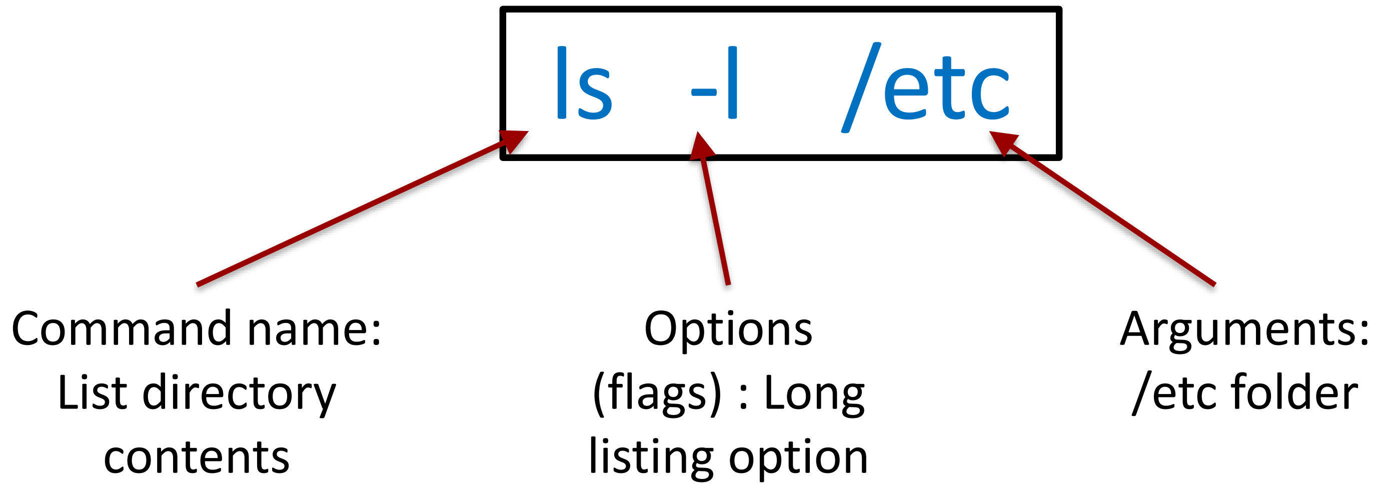
Naming

- **Case sEnsItive**
- **Reserved characters: *.|>< and more**
- **Spaces are a pain (use single quotation)**
- **Numbers are allowed, even at the beginning**
- **Special file names**
  - / The root directory (not to be confused with the root user)
  - . The current directory
  - .. The parent (previous) directory
  - ~ My home directory
- **Examples**

./a	same as a
../jane/x	go one level up then look in directory jane for x

Basic UNIX/Linux

- To execute a command, type its name and arguments at the command line



```
ls   -l   /etc
```

Command name: List directory contents
Options (flags) : Long listing option
Arguments: /etc folder

Common command syntax

- **Command:**

  ls

- **Command with an argument:**

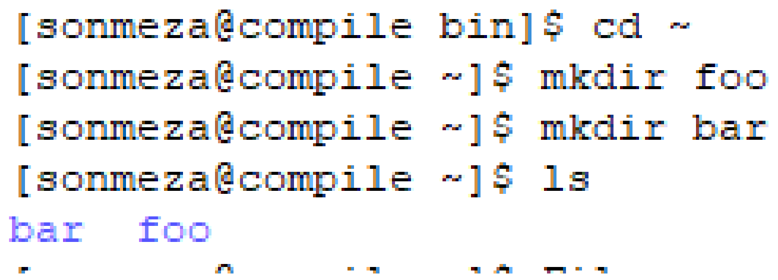
  ls /home

- **Command with an option (long listing):**

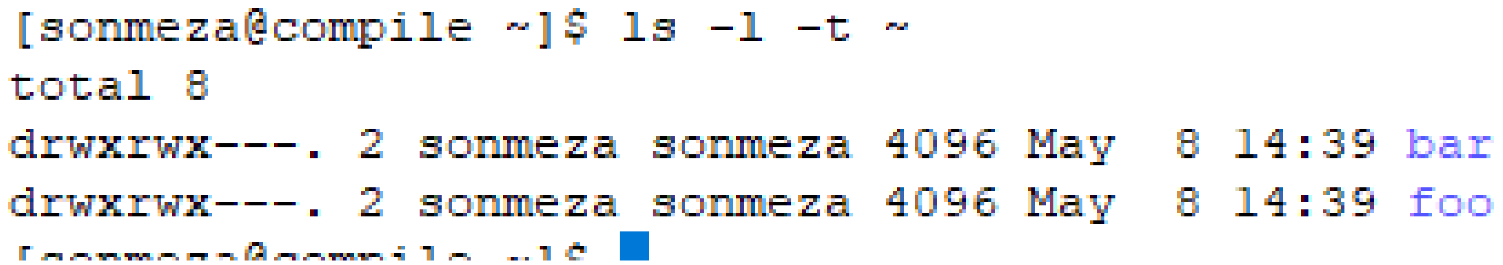
  ls -l

- **Command with options(Long listing and sort by time) and an argument:**

  ls -l -t /home



```
[sonmeza@compile bin]$ cd ~
[sonmeza@compile ~]$ mkdir foo
[sonmeza@compile ~]$ mkdir bar
[sonmeza@compile ~]$ ls
bar  foo
```



```
[sonmeza@compile ~]$ ls -l -t ~
total 8
drwxrwx---. 2 sonmeza sonmeza 4096 May  8 14:39 bar
drwxrwx---. 2 sonmeza sonmeza 4096 May  8 14:39 foo
```

Redirecting output

The output of a command may be sent (directed) to a file

```
ls  -l  >output
```

ls is the linux command to list the contents of the current directory

“>” is used to specify the output file

Redirecting input

The input of a command may be directed from a file

```
wc <input
```

WC is the Linux command for word count

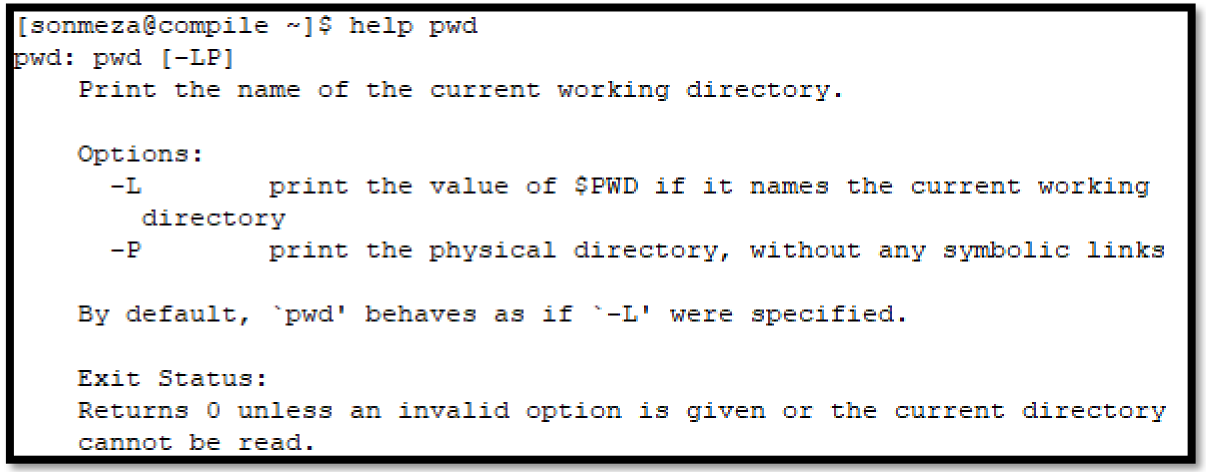
“<” is used to specify an input file

Common Linux Commands

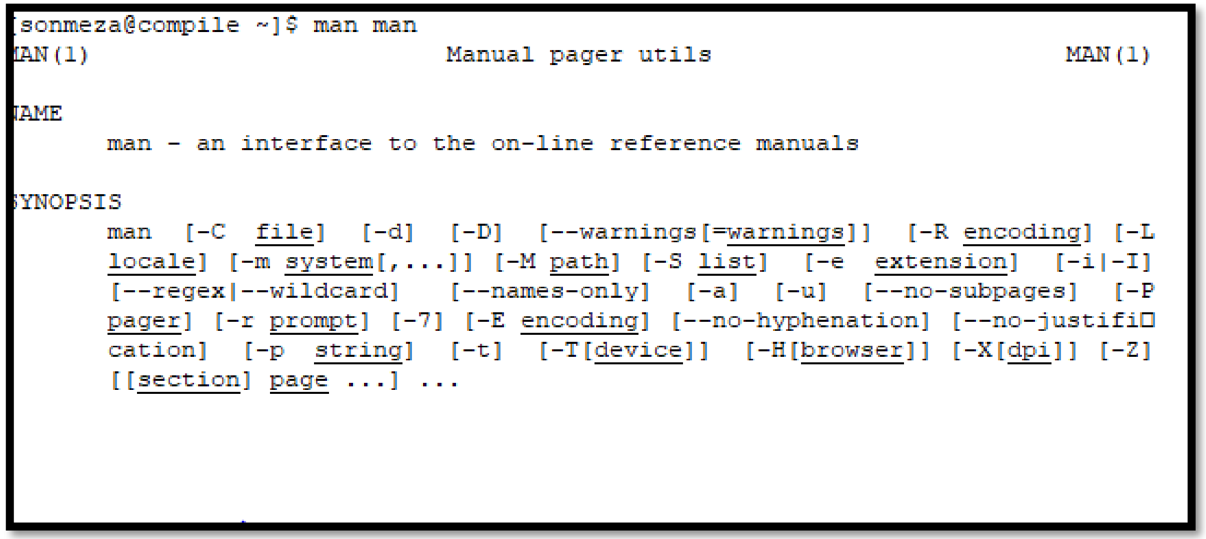
Command	Description
cp <fromfile> <tofile>	Copy from the *<fromfile>* to the *<tofile>*
mv <fromfile> <tofile>	Move/rename the *<fromfile>* to the *<tofile>*
rm <file>	Remove the file named *<file>*
mkdir <newdir>	Make a new directory called *<newdir>*
rmdir <dir>	Remove an (empty) directory *<dir>*
who	List who is currently logged on to the system
whoami	Report what user you are logged on as
ps	List your processes on the system
ps aux	List all the processes on the system
pwd	You are here

Getting Help

- **For built-in commands: help**
  - Commands that belongs to the shell
    - cd, exit, pwd
- **Everything else: man**
  - man ascii
  - man operator
  - man gcc
  - man man
- **Also use Google!**



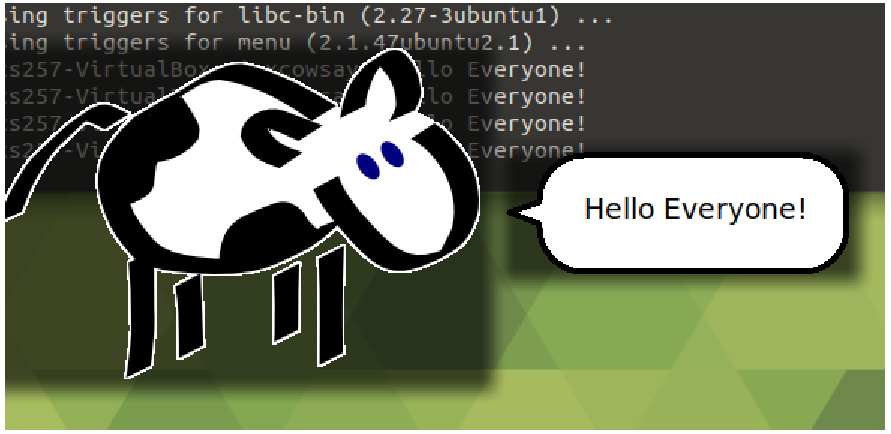
```
[sonmeza@compile ~]$ help pwd
pwd: pwd [-LP]
    Print the name of the current working directory.

    Options:
      -L        print the value of $PWD if it names the current working
        directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory
    cannot be read.
```



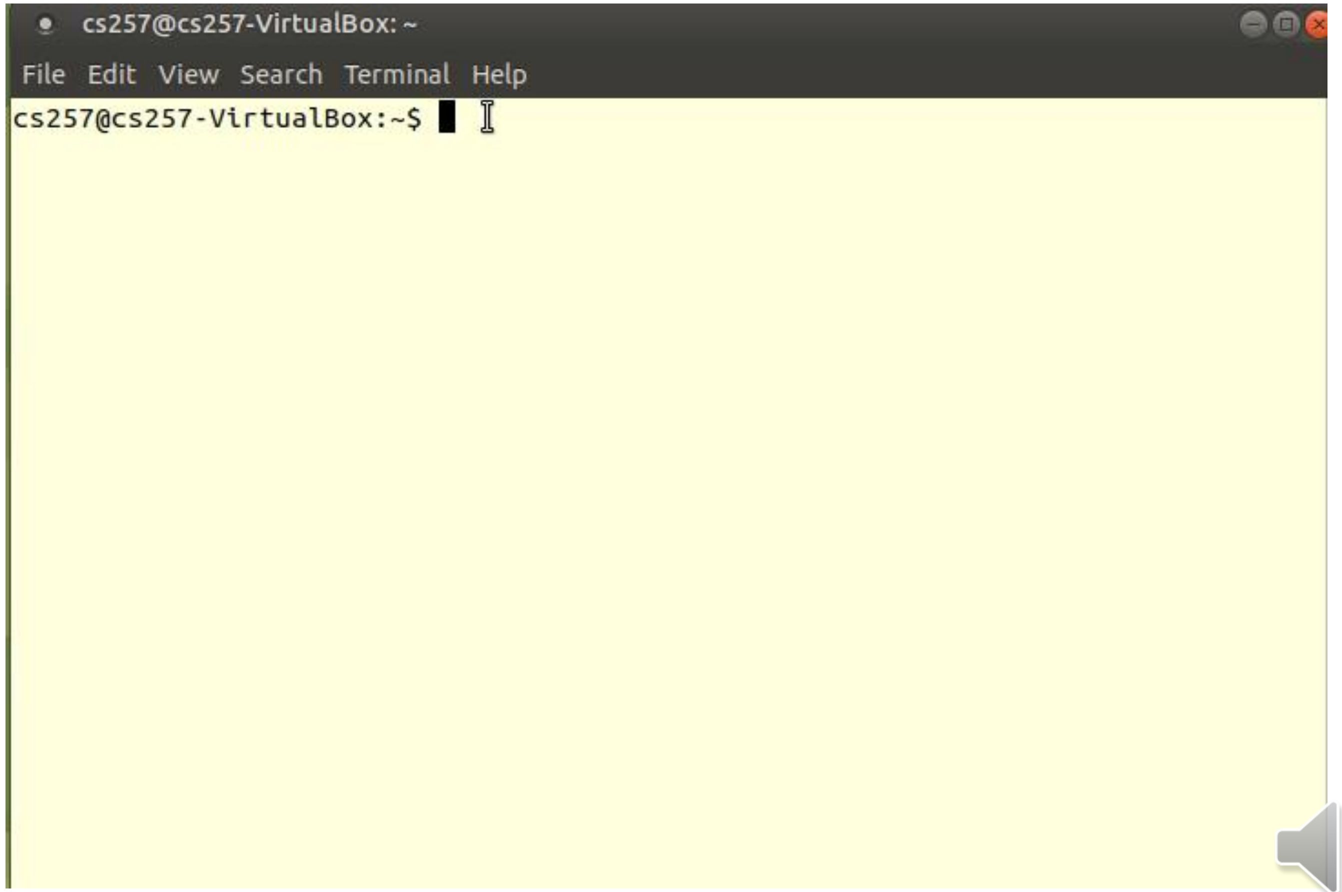
```
sonmeza@compile ~]$ man man
AN(1)                          Manual pager utils                          MAN(1)

AME
       man - an interface to the on-line reference manuals

YNOPSIS
       man  [-C  file]  [-d]  [-D]  [--warnings[=warnings]]  [-R encoding] [-L
       locale] [-m system[,...]] [-M path] [-S list]  [-e  extension]  [-i|-I]
       [--regex|--wildcard]  [--names-only]  [-a]  [-u]  [--no-subpages]  [-P
       pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justifi□
       cation]  [-p  string]  [-t]  [-T[device]]  [-H[browser]] [-X[dpi]] [-Z]
       [[section] page ...] ...
```

There are fun things you can try!

- **You may need to install them using package manager**
  - cmatrix (on ubuntu you can type sudo apt install cmatrix)
  - yes I love VCU
  - sl (reverse of ls)
  - xcowsay Hello Everyone



```
ing triggers for libc-bin (2.27-3ubuntu1) ...
ing triggers for menu (2.1.47ubuntu2.1) ...
s257-VirtualBox   xcowsay   llo Everyone!
s257-Virtual               llo Everyone!
s257                         Everyone!
s2  -Vi                     Everyone!
```



cs257@cs257-VirtualBox: ~
File Edit View Search Terminal Help

```
cs257@cs257-VirtualBox:~$ 
```

File Permissions

Every file has permissions associated with it.

- **Owner – who created it or owns the file**
- **Group – what group has special permissions**
- **Other – everyone else’s permissions**

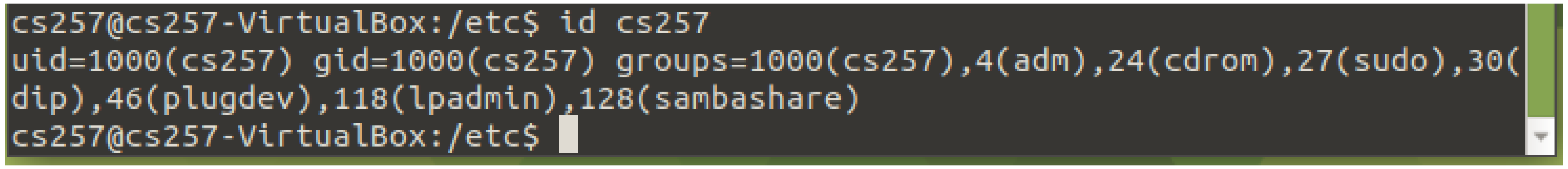
File Permissions

Every user has:

- ➢ **uid (login name)**
- ➢ **gid (login group)**

  membership of a "group" list:

  - ✓ **The uid is who you are (name and number)**
  - ✓ **The gid is your initial "login group" you normally belong to**
  - ✓ **The groups list is the file groups you can access via group permissions**



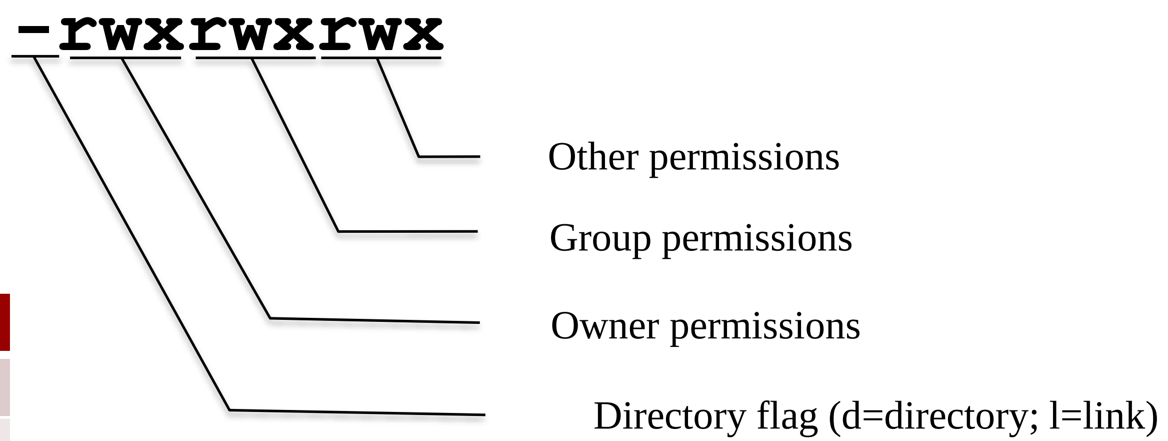
```
cs257@cs257-VirtualBox:/etc$ id cs257
uid=1000(cs257) gid=1000(cs257) groups=1000(cs257),4(adm),24(cdrom),27(sudo),30(
dip),46(plugdev),118(lpadmin),128(sambashare)
cs257@cs257-VirtualBox:/etc$ 
```

UNIX/Linux provides three kinds of permissions

- Read – refers to a user's capability to read the contents of directory or file
- Write – refers to a user's capability to write or modify a file or directory.
- Execute – affects a user's capability to execute a file or view the contents of a directory.

Permission type	Abbreviation	Bit value
Read	r	4
Write	w	2
Execute	x	1

File Permissions



Code	Reference to permissions
u	User that owns file
g	Group owners
o	Others
a	All

Changing File Permissions

- `chmod` command to change file permissions
- The permissions are encoded as an octal number

chmod command	Result of executing command
chmod 755 file	Owner=rwx Group=r-x Other=r-x
chmod 500 file2	Owner=r-x Group=--- Other=--
chmod 644 file3	Owner=rw- Group=r-- Other=r--
chmod +x file	Add execute permission to file for all
chmod o-r file	Remove read permission for others
chmod a+w file	Add write permission for everyone

Permission type	Abbreviation	Bit value
Read	r	4
Write	w	2
Execute	x	1

Outline

- Course Content
- Realities you will be introduced
- Unix/Linux and variants
- Command Line Interface
- ➢ Text Editors

Editing Files

- **Unix philosophy: Text is important!**
  - Configuration? Text files.
  - Scripting? Text files.
  - Programming? Text files.
- **Many editors to choose from**
  - You will learn basic vim, but feel free to use others for your projects

Editing Files with text Editors

Easy to learn

- **nano:** All help you need is displayed at the bottom of the screen. Do not require graphical environment.
- **gedit:** look like notepad in windows. Capable, configure able. Requires graphical environment.

Advanced

- Used by developers and administrators
  - **vi**
  - **emacs**
- Easily available and compatible
- Steep learning curve, very efficient for programming
- Do not require graphical environment.

Editing files without text editors

Using echo

```
$ echo line one > myfile
$ echo line two >> myfile
$ echo line three >> myfile
```

Using cat

```
$ cat << EOF > myfile
> line one
> line two
> line three
> EOF
$
```

Both will produce a file with lines

```
line one
line two
line three
```

vi (visual editor)

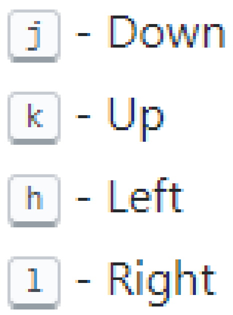
- Universal text editor – comes with Linux
- There may be times there is no other editor available in the system.
- There are extended version with graphical interface: gvim, kvim
- All commands are entered using keyboard

vi (visual editor)

vi has 2 modes:

- **Input mode**
  - To change mode to input mode press i
  - Exit to command mode with <ESC> key
  - Navigate

- **Command mode**
  - Default mode everything except typing
  - When you doubt, ESC will put you back into command mode



j	- Down
k	- Up
h	- Left
l	- Right

vi (visual editor)

After you enter the command: ***vi <filename>***

the following are the options you may have to enter **input mode** to input text.

i – insert before the cursor
I – insert text in the beginning
a – append after the cursor
A – append to the end of line
o – open a line below the current line
O – open a line above the current line
u– undo

vi (visual editor)

Hit <**ESC>** to go back to **command mode**.

- In the command mode, you may use the arrow keys to go left, right, up and down.

vi (visual editor)

- To undo the last modification, use **u** command
- To undo all changes in the current line, use **U** command
- To quit,
  - “**:wq**” write and quit.
  - “**:q**” quit without save.
  - “**:q!**” quit, this is needed if you modify the text and do not want to save.
  - “**ZZ**” quick save.

vi (visual editor)

Deletion commands are:

- **x** – delete (3x deletes 3 characters)
- **dd** – delete line (3dd deletes 3 lines)
- **dw** – delete a word

Try this!

- **Make a file hello.c containing:**

```
#include <stdio.h>

int main(void)
{
	printf(“Hello World!\n");
	return 0;
}
```

- **Quick save and quit: ZZ**

Compile and run

- **Run the compiler:**

```
gcc hello.c –o hello
```

- **Execute the program**

```
./hello
```

Shell scripting

- Automating things!
- Getting tired of typing the compile command?
- You can put list of commands in a shell script and run that script.
- Put this in a file called compile.sh

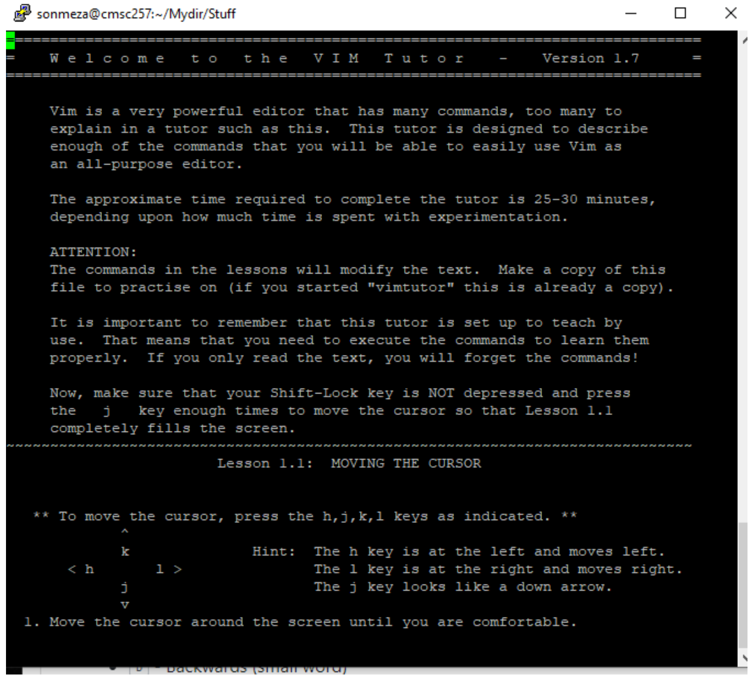
```
gcc hello.c –o hello
```

- Now just run it

```
sh compile.sh
```

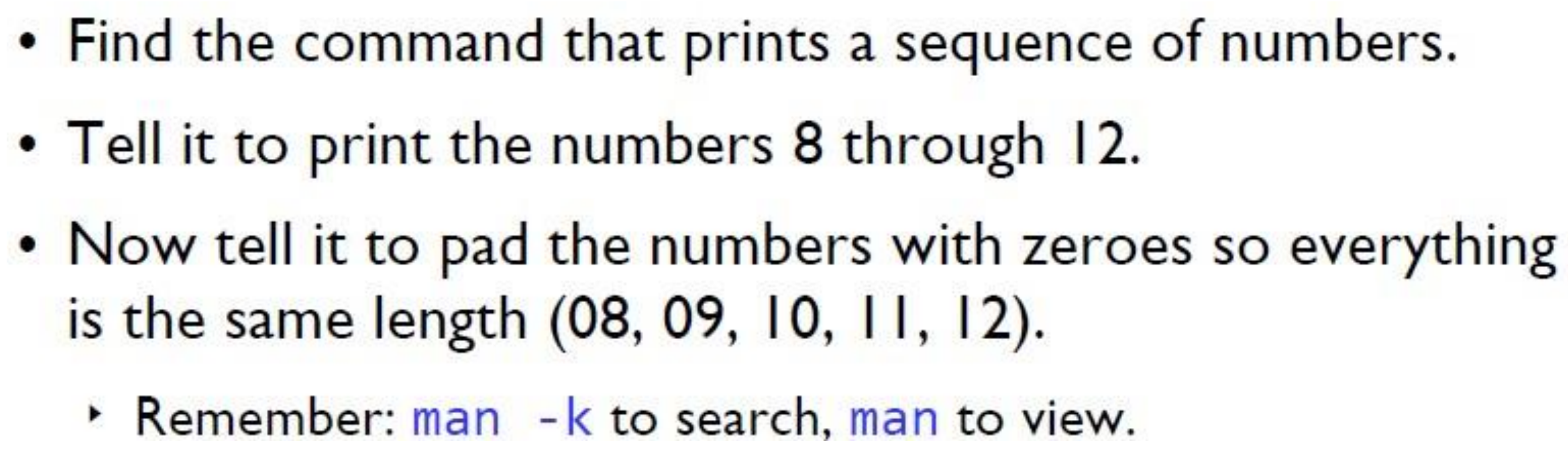
Learn more about vi editor

- Typing vimtutor launches a short but very comprehensive tutorial for those who want to learn their first vi commands. This tutorial is a good place to start learning vi.
- Your fist HW is to complete vimtutor lessons
- If you like learning with games try: https://vim-adventures.com/



```
sonmeza@cmsc257:~/Mydir/Stuff
===============================================================================
=    W e l c o m e   t o   t h e   V I M   T u t o r    -    Version 1.7      =
===============================================================================

     Vim is a very powerful editor that has many commands, too many to
     explain in a tutor such as this.  This tutor is designed to describe
     enough of the commands that you will be able to easily use Vim as
     an all-purpose editor.

     The approximate time required to complete the tutor is 25-30 minutes,
     depending upon how much time is spent with experimentation.

     ATTENTION:
     The commands in the lessons will modify the text.  Make a copy of this
     file to practise on (if you started "vimtutor" this is already a copy).

     It is important to remember that this tutor is set up to teach by
     use.  That means that you need to execute the commands to learn them
     properly.  If you only read the text, you will forget the commands!

     Now, make sure that your Shift-Lock key is NOT depressed and press
     the   j   key enough times to move the cursor so that Lesson 1.1
     completely fills the screen.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                        Lesson 1.1:  MOVING THE CURSOR


   ** To move the cursor, press the h,j,k,l keys as indicated. **
             ^
             k              Hint:  The h key is at the left and moves left.
       < h       l >               The l key is at the right and moves right.
             j                     The j key looks like a down arrow.
             v
  1. Move the cursor around the screen until you are comfortable.
```

man practice



- Find the command that prints a sequence of numbers.
- Tell it to print the numbers 8 through 12.
- Now tell it to pad the numbers with zeroes so everything is the same length (08, 09, 10, 11, 12).
  - Remember: `man -k` to search, `man` to view.

How to approach

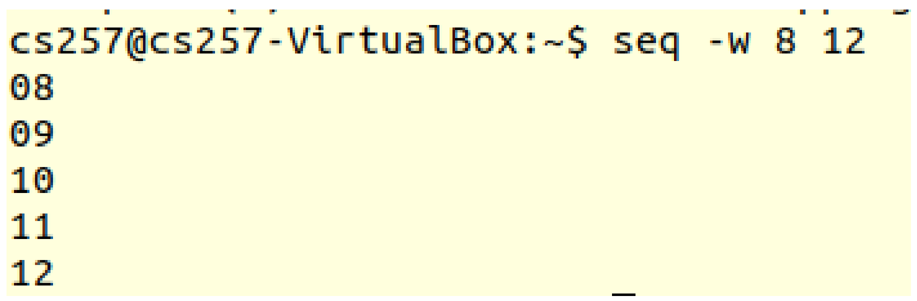
- First look for the word “sequence” in man pages

  man –k sequence

  Or

  apropos sequence

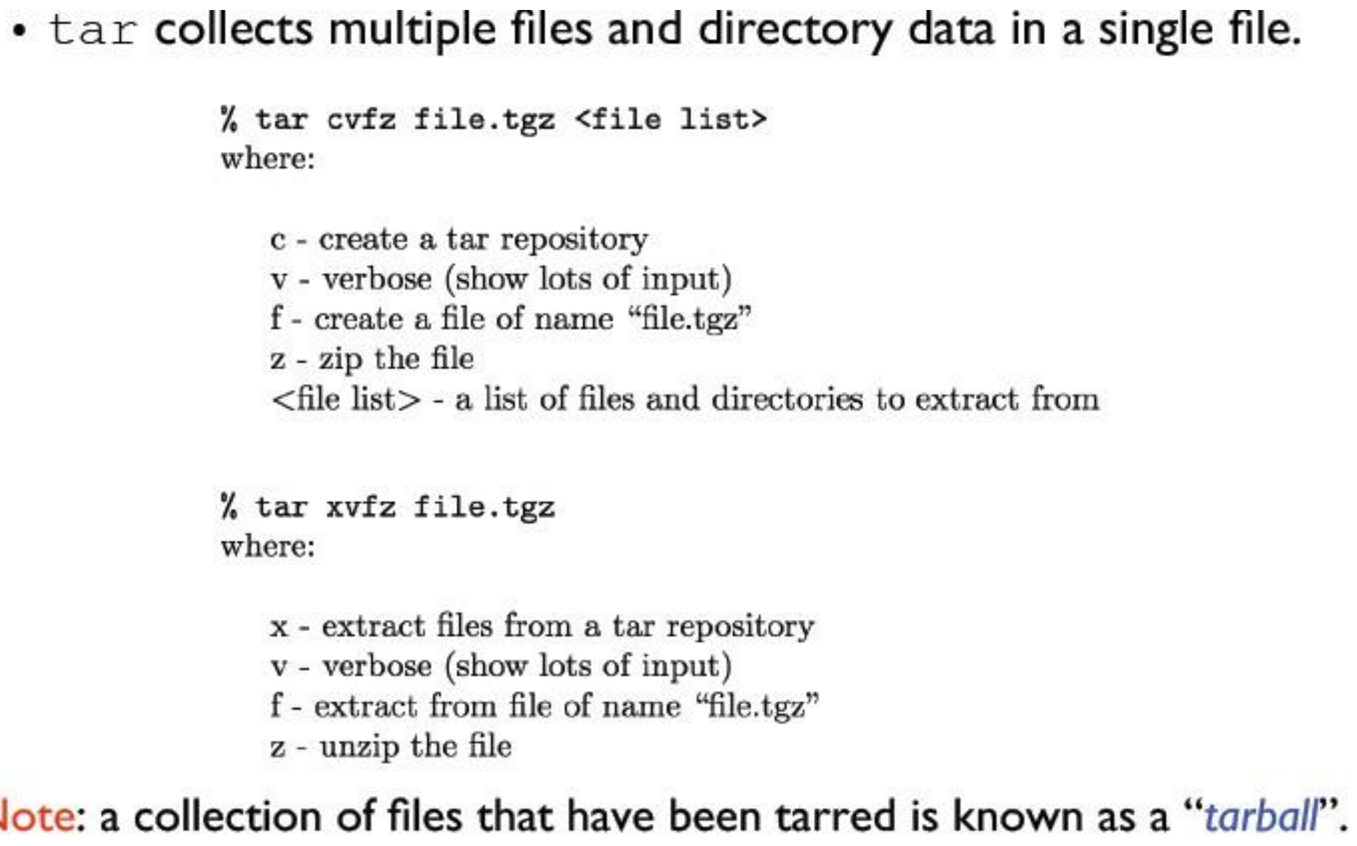
- Find the command that prints numbers in sequence looking at short descriptions
- Find the man pages for that specific command and look for the syntax.



```
cs257@cs257-VirtualBox:~$ seq -w 8 12
08
09
10
11
12
```

Linux Utilities: tar - Backups

Used for backup and recovery as well as compressing and archiving



- `tar` collects multiple files and directory data in a single file.

```
% tar cvfz file.tgz <file list>
```

where:

c - create a tar repository
v - verbose (show lots of input)
f - create a file of name “file.tgz”
z - zip the file
<file list> - a list of files and directories to extract from

```
% tar xvfz file.tgz
```

where:

x - extract files from a tar repository
v - verbose (show lots of input)
f - extract from file of name “file.tgz”
z - unzip the file

Note: a collection of files that have been tarred is known as a “*tarball*”.