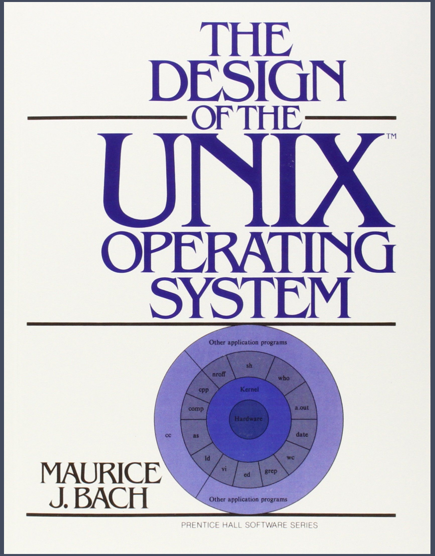
Intro to BASH

Gavin Guinn for CSUS

Defining Terms

- Unix
  - Operating system developed in the 1970’s
  - Unix is the parent of MacOS and all variants of Linux
    - These OS’s share similar functionality
    - Collectively referred to as **nix*
- Shell
  - Program used to interface with an operating system’s functionality via text commands
- BASH (**B**ourne **A**gain **SH**ell)
  - Used to interface with **nix* environments
  - BASH is not the only shell that can be used with **nix*
    - MacOS uses zsh (**zSh**ell) which extends the functionality of BASH


THE DESIGN OF THE UNIX™ OPERATING SYSTEM
MAURICE J. BACH
PRENTICE HALL SOFTWARE SERIES

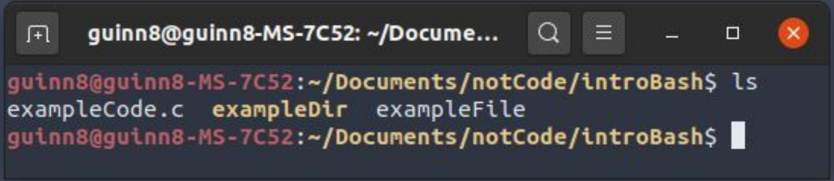
Getting Started

- You should have a BASH or ZSH (Linux or MacOS) shell open
  - This is not Windows command prompt or Powershell
    - Windows shells have an entirely different set of commands
  - You can use BASH with Windows (really good idea to have)
    - WSL (Windows Subsystem for Linux) on Windows 10
    - git BASH for any Windows version

```
guinn8@guinn8-MS-7C52:~$
```

COMMAND: ls

> ls (lists files and folders in the present directory)
  ✱ Present directory == current folder you are in



```
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash$ ls
exampleCode.c  exampleDir  exampleFile
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash$ 
```

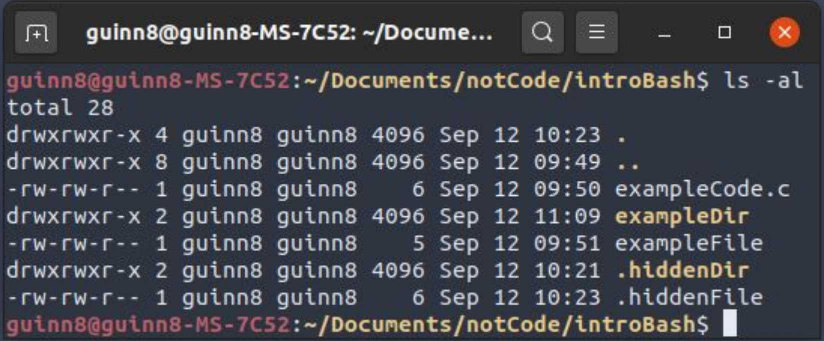
> Both exampleCode.c and exampleFile are files
  ✱ Note that exampleFile does not have a file extension

> The terminal changes the text style between directories and files

COMMAND: ls

> `ls -al` (list all long)

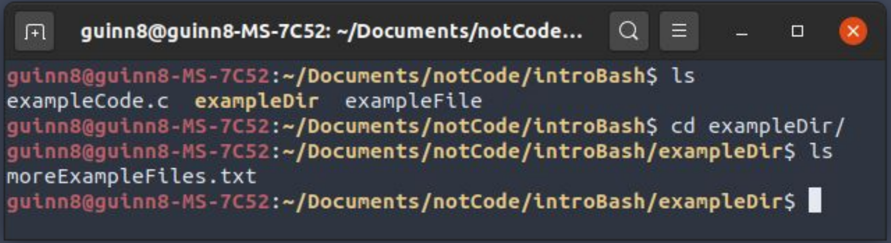
- Lists all files and folders in the present directory descriptive manner, also lists hidden files
- `-al` is an argument (instruction) to the program `ls`
- In *nix systems any file or directory prefixed with `.` is hidden



```
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash$ ls -al
total 28
drwxrwxr-x 4 guinn8 guinn8 4096 Sep 12 10:23 .
drwxrwxr-x 8 guinn8 guinn8 4096 Sep 12 09:49 ..
-rw-rw-r-- 1 guinn8 guinn8    6 Sep 12 09:50 exampleCode.c
drwxrwxr-x 2 guinn8 guinn8 4096 Sep 12 11:09 exampleDir
-rw-rw-r-- 1 guinn8 guinn8    5 Sep 12 09:51 exampleFile
drwxrwxr-x 2 guinn8 guinn8 4096 Sep 12 10:21 .hiddenDir
-rw-rw-r-- 1 guinn8 guinn8    6 Sep 12 10:23 .hiddenFile
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash$
```

COMMAND: cd

> cd (change directory)
  - ✱ Takes the directory you would like as an argument



```
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash$ ls
exampleCode.c  exampleDir  exampleFile
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash$ cd exampleDir/
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash/exampleDir$ ls
moreExampleFiles.txt
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash/exampleDir$ 
```

> Notice how ~/Documents/notCode/introBash changes to ~/Documents/notCode/introBash/exampleDir
  - ✱ This is the path of your current directory

COMMAND: cd

1. Try `cd ..` and note how your pwd has changed
2. Try `cd` (without arguments) and note how your directory has changed
3. Try `cd .` and see what happens

COMMAND: cd

1. Try `cd ..` and note how your pwd has changed
   - The pwd changes to the parent of the last directory
   - `..` is the path to the parent directory
2. Try `cd` (without arguments) and note how your pwd has changed
   - The pwd changes to the user's home directory
   - The home directory is represented by `~`
3. Try `cd .` and see what happens
   - Nothing should happen
   - `.` is the path to the current working directory (this is actually useful)

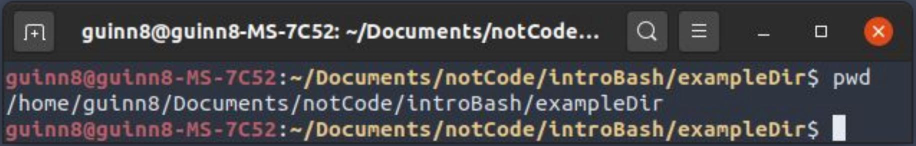
CONCEPT: Directory Organization

- The directory `/` is the file systems root folder
  - All directories on the system are contained by the root
- A pathname is a sequence of directories or file names separated by `/`
  - `~/Documents/notCode/introBash` is the path to the `introBash` directory
  - `~/Documents/notCode/introBash/exampleCode.c` is the path to the `exampleCode.c` file
- The `~` directory is your home folder
  - When you log into a terminal or `cd` (without arguments) you will be taken to the home directory

CONCEPT: Absolute Path

> A pathname starting / is an absolute path

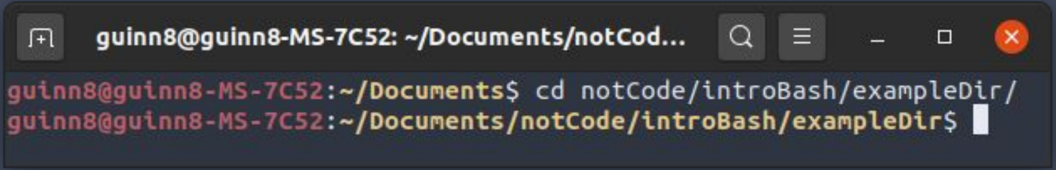
- /home/guinn8/Documents is the absolute path to the Documents directory
- How to navigate from the root (/) to a file or directory
- ANALOGY: Like giving directions by stating a locations address
  - “The address of the Dennys by campus is 2450 16 Ave NW”
- The pwd command produces the absolute path to the current working directory



```
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash/exampleDir$ pwd
/home/guinn8/Documents/notCode/introBash/exampleDir
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash/exampleDir$
```

CONCEPT: Relative path

> A pathname that <u>does not</u> start with / is a relative path
  - notCode/introBash/exampleDir is a relative path to the exampleDir directory
  - How to navigate from the current directory to the desired file or directory
  - Remember that .. is the path to the parent directory
  - ANALOGY: Like giving directions based on your current location
    - "The Denny's is a couple of blocks south-west of campus"



```
guinn8@guinn8-MS-7C52:~/Documents$ cd notCode/introBash/exampleDir/
guinn8@guinn8-MS-7C52:~/Documents/notCode/introBash/exampleDir$ 
```

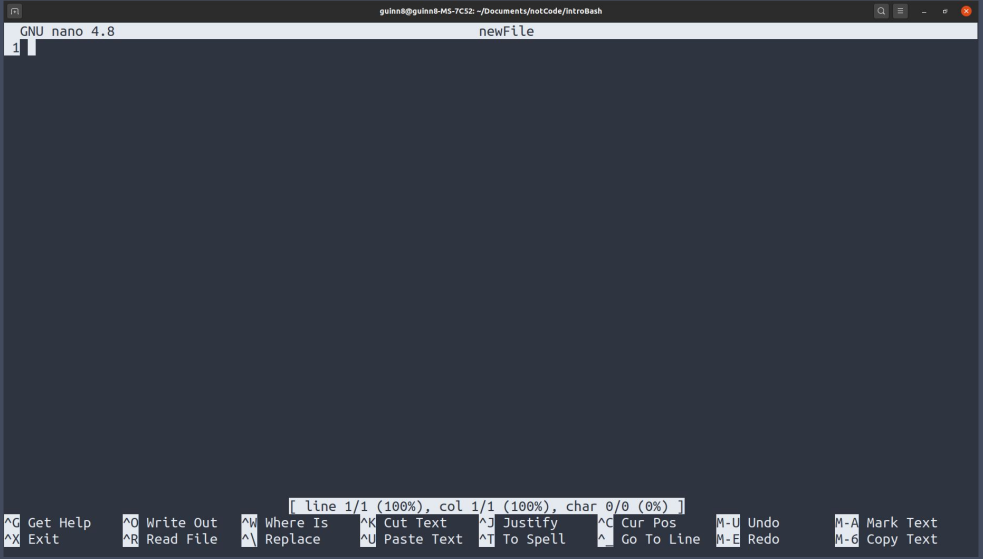
EXERCISE: Paths

- Utilize an absolute path
  1. `cd` (without arguments)
  2. Use `pwd` to generate an absolute path to your current location
  3. Create a absolute path to the Documents directory using the results of `pwd`
  4. Use `cd` with that path to change to the Documents directory
- Utilize an relative path
  1. `cd ~/Documents`
  2. From Documents devise a relative path to Downloads
  3. Use `cd` with that path to change to the Downloads folder

COMMAND: nano

> nano is a command-line text editing program

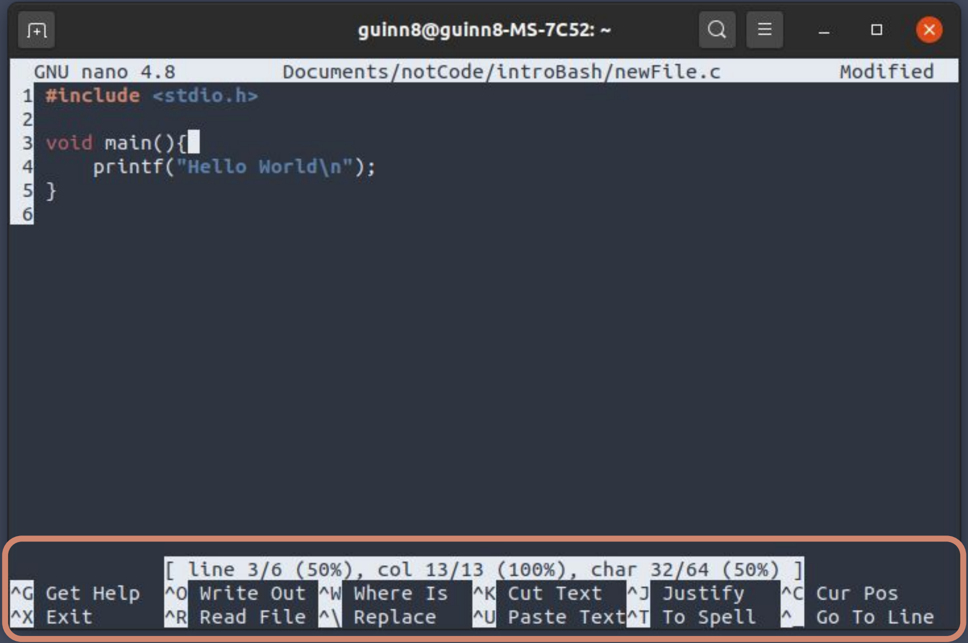
- Easier to use than other text editors
- Invoke with nano *filename*
- Hotkeys are listed
  - ^*key* means ctrl-*key*
  - M-*key* means alt-*key*
- ^X then follow prompts to save and exit



```
guinn8@guinn8-MS-7C52: ~/Documents/notCode/introBash
  GNU nano 4.8                                newFile





                  [ line 1/1 (100%), col 1/1 (100%), char 0/0 (0%) ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo      M-6 Copy Text
```

COMMAND: nano

- Important Hotkeys
  - Highlight text
    - Shift-*arrow_key*
    - M-A to mark spot in text
  - Copy == M-Shift-6
  - Cut == ^K
  - Paste == ^U
  - Undo == M-U
  - Auto complete == M-]
- Most of these shortcuts can be found in the bottom bar



```
  GNU nano 4.8          Documents/notCode/introBash/newFile.c          Modified
#include <stdio.h>

void main(){
    printf("Hello World\n");
}

                [ line 3/6 (50%), col 13/13 (100%), char 32/64 (50%) ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line
```

^ == Control
M == Alt

COMMAND: nano

1. Try the following
   - Type some text into nano
   - Highlight that text (Shift-*arrow_key* or M-A)
   - Copy that text (M-Shift-6)
   - Paste that text (^U)
   - Save and close the file (^X)
2. Copy some text from the system into the terminal (Control-Shift-V)

COMMAND: mv / cp / mkdir

- `mv` (move) is used to move or rename files
  - `mv` *filepath destinationFilepath*
  - If a directory is supplied as the destination the file will be moved but keep its name
  - If a path to a filename is supplied the name will be changed
- `cp` (copy) is used to copy files
  - It is used is pretty much the same way as `mv`
- `mkdir` (make directory)
  - `mkdir` *directory-path*
  - Creates a new directory with the name of the supplied path

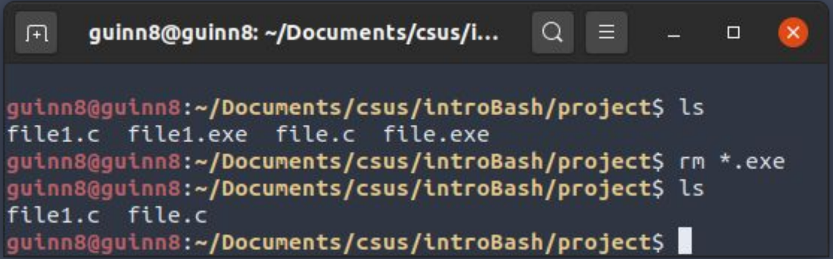
COMMAND: rm

- rm (remove) is used to remove a file or directory
  - rm *file-path* will remove that file
    - **There is no recycle bin, the command is final**
  - rm -r *directory-path* removes a directory recursively (deletes directory and everything in it)
    - **Again, use with care!**
    - **You could literally destroy your OS with** `sudo rm -r /*`

CONCEPT: Regular Expressions (regex)

> Regular expressions are used to define patterns of text (specify a set of strings)

- ✱ Say you wanted to delete a bunch of files with similar names



```
guinn8@guinn8:~/Documents/csus/introBash/project$ ls
file1.c  file1.exe  file.c  file.exe
guinn8@guinn8:~/Documents/csus/introBash/project$ rm *.exe
guinn8@guinn8:~/Documents/csus/introBash/project$ ls
file1.c  file.c
guinn8@guinn8:~/Documents/csus/introBash/project$ 
```

> In the command `rm *.exe` the `*` symbol will *match* with any arbitrary text (0 or more characters) which is followed by `.exe`

> Pretty much all file processing commands with support regex input (where reasonable)

> There is way more to [BASH regex]

CONCEPT: become hackerman

- Use the up/down arrow keys to view commands entered commands in order
- You can use the tab key to complete commands or path-names
  - If you double press tab you will get a list of possibilities
  - Makes typing in long commands bearable
- reverse-i-search (Control-r)
  - Search for previous command history
  - Great for finding long commands

CONCEPT: become hackerman

- man
  - Use `man` *`command`* to view the commands manual page
  - Great for technical explanation, not great for examples
- sudo
  - Some commands require extra permissions to execute,
  - `sudo` *`command`* (super-user do) gives that permission