Hello Lab

Thoth Machine

- What is it?
  - Linux machine in the CS department of Pitt
  - Hosted at **thoth.cs.pitt.edu**
  - Will be used by you to compile and run your programming labs

- Why?
  - Needed softwares, packages etc. are already installed in this machine

1st thing to do: INCREASE your UNIX quota

- To ensure you have **sufficient space** to run the assignments/projects of this class

- Steps:
  - Login to https://accounts.pitt.edu/
  - Click “EMAIL & MESSAGING” menu
  - Click “UNIX QUOTA”
  - Click “Increase Quota”

Accessing Thoth

- Windows
  - Use Putty
    - Source: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
  - Use Powershell/Bash
    - Command Line
- Mac OS/Linux
  - Open the terminal
    - Type `ssh pittusername@thoth.cs.pitt.edu`
    - Enter yes if prompted to accept the new ssh key
    - Enter Pitt password (note: terminal will not show anything you type)

What next??

- You'd probably want to
  - Create a new folder
  - Delete an existing folder
  - Run a C program
  - Transfer files from your laptop to the thoth machine, etc., etc.

- How?
  - By running Bash commands

Common Bash Commands

Make Directory : *mkdir directory_name*

```
ded59@thoth:~$ ls
no_segfault_ex.c  segfault_ex.c
ded59@thoth:~$ mkdir test
ded59@thoth:~$ ls
no_segfault_ex.c  segfault_ex.c  test
ded59@thoth:~$
```

Change Directory : *cd directory_name*

```
ded59@thoth:~/test2$ pwd
/home/PITT/ded59/test2
ded59@thoth:~/test2$ cd /home/PITT/ded59/test
ded59@thoth:~/test$ pwd
/home/PITT/ded59/test
ded59@thoth:~/test$
```

Delete Directory : *rmdir directory_name*

```
ded59@thoth:~$ ls
no_segfault_ex.c  segfault_ex.c  test  test2
ded59@thoth:~$ rmdir test
ded59@thoth:~$ ls
no_segfault_ex.c  segfault_ex.c  test2
ded59@thoth:~$
```

Show file content: *cat filename*

```
ded59@thoth:~$ ls
no_segfault_ex.c  segfault_ex.c  test2
ded59@thoth:~$ cat segfault_ex.c
#include <stdio.h>
int main() {
    int a[5] = {1, 2, 3, 4, 5};
    unsigned total = 0;
    for (int j = 0; j < sizeof(a); j++) {
        total += a[j];
    }
    printf("sum of array is %d\n", total);
}
ded59@thoth:~$
```

More Bash Commands

- Some resources you can refer to:
  - https://www.educative.io/blog/bash-shell-command-cheat-sheet
  - https://hackernoon.com/top-10-bash-file-system-commands-you-cant-live-without-4cd937bd7df1

Transferring files to and from thoth

1. Download and install a user-friendly file transferring software
   - Windows
     - WinSCP
       - https://winscp.net/eng/download.php
   - Mac OS/Linux
     - FileZilla
       - https://filezilla-project.org/download.php?show_all=1

Transferring files to and from thoth

2. Command line

- `scp local.txt user@thoth.cs.pitt.edu:remote.txt`
  (copies local.txt from your PC to remote.txt in your home directory of thoth)
- `scp user@thoth.cs.pitt.edu:remote.txt local.txt`
  (copies remote.txt on thoth to local.txt on your PC)
- `scp user@thoth.cs.pitt.edu:path/to/file .`
  (copies file with path after colon to local machine with same filename as on thoth)

Where to write your C code? **Text Editors**

- Text Editors built in Thoth
  - nano – super basic
  - emacs - full featured
  - vim – full-featured, modal
    - Tutorial - https://www.openvim.com/

Text Editors in your local machine

- atom - Use remoteftp extension to remote edit on Thoth (`https://code.visualstudio.com/docs/remote/remote-overview`)
- vscode - Use remote extension to remote edit on Thoth (`https://atom.io/packages/remote-ftp`)
- Notepad++ (Windows only) - use WinSCP to upload files to Thoth (`https://winscp.net/`)

You may also refer to the following document written by Gordon Lu that will walk you through the process of setting up SFTP for VSCode and Atom. How to setup SFTP on Thoth: `https://www.dropbox.com/s/dwknm8i9ce0az9x/SFTP_setup.pdf`

Sample C Program

stdio.h file contains functions such as scanf() and printf() to take input and display output respectively

```
#include <stdio.h>
int main (int argc, char* argv[]){
  // Declare a variable
  int x;
  // Assign a variable
  x = 2;

// Print a string and a variable
  printf("Hello world!  x is currently %d \n", x);

  return 0;
}
```

Preprocessor command that tells the compiler to include the contents of stdio.h

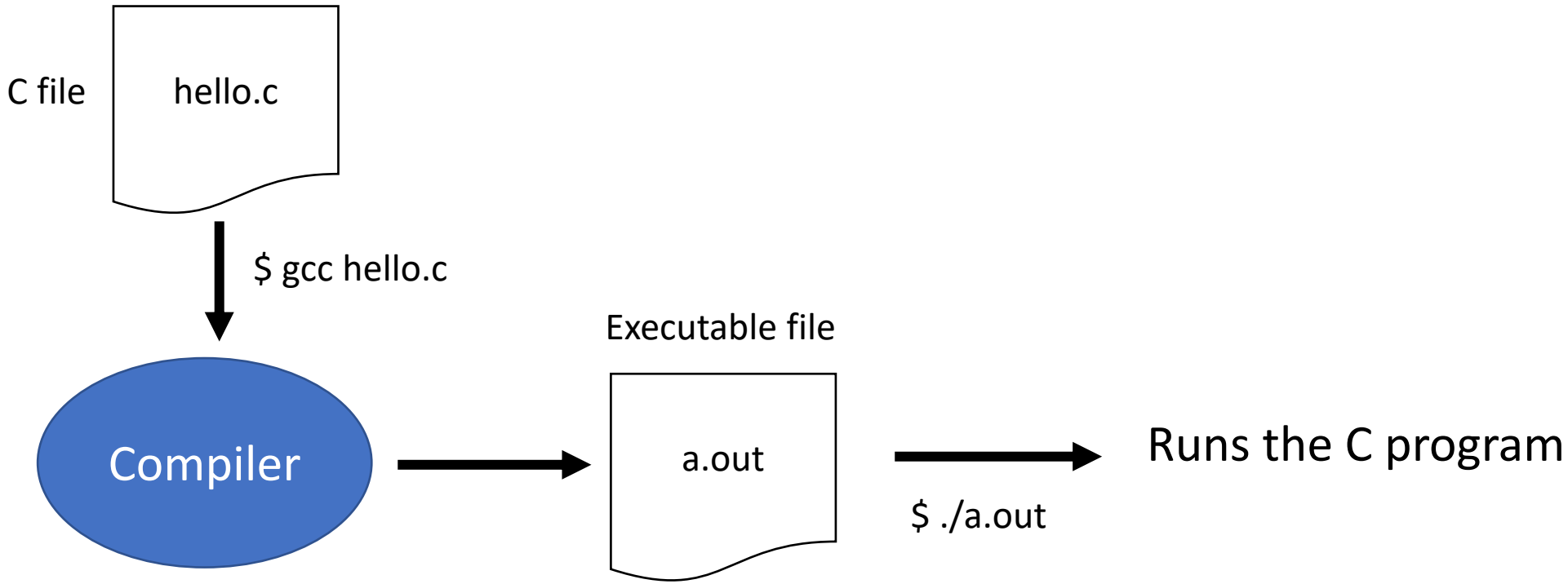
Placeholders:
%d: signed int
%u: unsigned int
%f: float
%s: string
%p: pointer
%x: hexadecimal

printf()

- printf() is a library function to send formatted output to the screen
- printf() without stdio.h will give compilation error

```
printf("I am %d years old", 20);          // prints "I am 20 years old"
printf("My name is %s", "John");          // prints "My name is John"
printf("%d in hex is %x", 2827, 2827);    // prints "2827 in hex is 0xb0b"
```

Compiling and Running a C Program



Compiling and Running a C Program

```
$ gcc -Wall -g -std=c99 -o hello hello.c
$ ./hello
```

-g : turns on debugging symbols

-W: Sets warnings
-Wall: Enable all warnings

-std: Sets the version of C to be used
-std=c99: Sets the version of C to be c99

-o: sets the name of the resulting executable

Debugging Tools - GDB

- Used to sift through the assembly code of compiled programs in later labs
- Used to locate bugs in your code

Debugging Tools - GDB

```
$ wget https://bit.ly/2ysH6gH -O calculator.c
```

Read through the code in a text editor, then compile and run the program:

```
$ gcc -Wall -g -std=c99 -o calculator calculator.c
$ ./calculator 4 5 +
```

Download, compile and run calculator.c

- `gdb calculator`: opens the binary file and presents you with the command prompt.
- `layout split`: shows C source and assembly instructions together (the C source is available because the program was compiled with the -g flag).
- `b 35`: sets a breakpoint at line 35, the if statement inside main().
- `r 4 5 +`: runs the program with input "4 5 +", which will pause at line 35.
- `p argc`: prints the argc C variable.
- `p $rdi`: prints the `%rdi` register inside the CPU (holding argc).
- `c`: continues the execution of the program.
- `q`: quits GDB.

Use these commands to use gdb for debugging

Debugging Tools - Valgrind

- Helps catch memory bugs
- Emulates CPU and tracks memory access

Valgrind Example

- Download

```
$ wget https://bit.ly/2yyYTmt -O no_segfault_ex.c
$ wget https://bit.ly/3duIIoP -O segfault_ex.c
```

- Compile

```
$ gcc -std=c99 no_segfault_ex.c -o no_segfault_ex
$ gcc -std=c99 segfault_ex.c -o segfault_ex
```

- Run

```
$ ./segfault_ex
...

$ ./no_segfault_ex
...
```

Segmentation fault

- Running segfault_ex results in a segmentation fault.
  - Occurs when a program crashes from trying to access memory that is **not available** to it

- The file segfault_ex.c is very small so you should be able to open the file and understand what is causing the segfaults.
  - But what to do when the file is too large?
    - Use Valgrind
      - Outputs where the illegal access occurred
    - gdb can also be helpful sometimes (more on this later)

Valgrind Example

- Run:
  - $valgrind ./segfault_ex
    - Should cause Valgrind to output where the illegal access occurred
  - $valgrind ./no_segfault_ex
    - Valgrind seems to not be able to tell you exactly where the problem is occurring
    - Use the message provided by Valgrind to determine the issue

Lab Quiz: C Warm Up

- Download `lab0.c`
  - `$ wget https://bit.ly/35LYokU -O lab0.c`
- Navigate to directory where `lab0.c` resides
- Compile
  - `$ clang -g -Wall -std=c99 -o lab0 lab0.c`
- Run
  - `$ ./lab0 <number ranging from 1 to 5>`
    - `Eg: $./lab0 1`
- Go to Gradescope and complete the quiz
  - Open lab0.c in a text editor and go through the comments and code and make any required changes in it
  - Recompile lab0.c after you make any change in it.

Lab Quiz: C Warm Up

10 Checking Your Work

You now should have everything you need to complete the assignment. Follow the instructions found on the associated Gradescope quiz; you will want to work on the different parts of the lab in order (from 1 to 5). Each question can be answered and/or verified by appropriate edits to the source code. Note that every time you want to test a code modification, you will need to use the `clang -g -Wall -std=c99 -o lab0 lab0.c` command to produce an updated `lab0` executable file (Tip: Use the up and down keys to scroll through previous terminal commands you've executed).

11 Submission

You will not be submitting files for this lab. **There is no checkoff meeting for this lab.** Submit your answers to the "Hello Lab" online quiz/assignment on Gradescope. You can submit each question individually in Gradescope. Gradescope will indicate which questions were answered correctly and which were answered incorrectly. You have unlimited tries for each question in this lab until the submission deadline.