601.220 Intermediate Programming

Summer 2024, Meeting 1 (June 3th)

Welcome!

Today's agenda:

- course overview
- ugrad account, ssh
- get started with Linux
- get started with C
- Exercises 1 and 2

Announcements

- HW0 is due on Thursday, June 6th
- HW1 is due on Thursday, June 13th
- You can (and should) start working on both of them right away

Goals for today

- By the end of class today, you should
  - Be able to log into your ugrad account
  - Have your Github account connected to the `jhu-ip` org
  - Be signed up for Piazza and Gradescope
  - Have joined the Slack workspace
- We will help you complete any of these that you still need to do (ask!)

Today will be a bit different than usual

- Since it's not 100% certain that everyone will have done the pre-class prep, today will be slightly more lecture-oriented than usual

Important resources

- Course website: https://jhu-ip.github.io/cs220-summer24
  - All public information about the course is here!
  - Links to videos, slides, exercises, assignments, etc.
- Piazza: we'll use for announcements and Q&A, hopefully you are signed up already
- Gradescope: assignment submission and exams, you should already be registered
- Slack: you should have received an invitation to a Slack workspace. This is the best way to get the attention of the CAs and me when we're doing an in-class exercise.

Typical class meeting

- Each class meeting will cover what would be two days worth of content in the Fall or Spring
- Typical format:
  - Lecture/review/discussion (first set of topics) - 10:00_10:25am
  - In-class exercise - 10:25_11:10am
  - Lecture/review/discussion (second set of topics) - 11:10_11:35am
  - In-class exercise - 11:35_12:15pm
- Time shown above are aproximate.

What you should do before you come to class

- Watch the lecture videos
- Review the slides and the recap questions
- Bring your questions about the day's topics to class and ask them using `slido.com` during the lecture/review/discussion portions of class. (Will be available from next session!)

Participation

Make it a priority to participate fully!

- Complete the in-class exercises
- Ask questions!
  - Using Zoom chat during lecture time.
  - Using the Slack workspace.
  - Using Zoom (in breakout rooms, during in-class exercises or other in-class work sessions)

Homeworks

- There will be 4.5 *individual* coding homeworks
  - HW0 is a "half" homework
- Typically you will have 1 week to work on each

Projects

- Two team projects in groups of 2 or 3
- You'll have about 1 week to work on each

Exams

- Two exams, *in class*
- Using Gradescope

Grade calculation

- Coding homework assignments (done individually) - 36%
- Midterm coding project (in teams) - 14%
- Midterm exam - 17%
- Final coding project (in teams) - 16%
- Final exam - 17%
- Participation - 0% (strongly recommended to fully participate in class sessions)
- In class exercises - 0% (strongly recommended to complete them all)

ugrad accounts, Linux

- For all coding (homeworks, projects), you will use Linux on the ugrad computing cluster
- Hopefully you have a ugrad account already
- You will log into your ugrad account using `ssh`
  - Windows: can use either PuTTY or a WSL terminal
  - Mac or Linux: open a terminal window and use the `ssh` command
- See Resources page on course website for some Linux tutorials
- Make it a priority to become comfortable using the Linux command line
  - We're here to help you if you have questions

Day 1 recap questions

1. What is the difference between short-term and long-term lazy?
2. What is the ssh command to connect to a ugrad machine?
3. What are the commands to move, copy, and remove a file on a Linux machine?
4. What should you do to learn C and C++ faster?
5. What will we do during the class time?

1. What is the difference between short-term and long-term lazy?

Short-term lazy: take a shortcut now, but make the program more difficult to debug or modify.

Long-term lazy: invest time and thought up front, make the program easier to debug and modify. Overall, this will save you time.

2. What is the ssh command to connect to the ugrad machine?

`ssh ugradx.cs.jhu.edu`

or

`ssh ugrad1.cs.jhu.edu` (could replace 1 with any number 1–24)

3. What are the commands to move, copy, and remove a file on a Linux machine?

Move or rename a filename: `mv filename1 filename2`

Copy a filename: `cp filename1 filename2`

Remove a file: `rm filename`

Warning: overwritten files are not (easily) recoverable. *Think* before your run any of these commands.

4. What should you do to learn C and C++ faster?

Practice.

Complete all of the in-class exercises.

Take the individual programming homeworks seriously.

Ask questions! Ask for help!

5. What will we do during the class time?

See earlier slide (“Typical class meeting”).

Day 2 recap questions

1. The command to compile a C program is `gcc <source file> -std=c99 -pedantic -Wall -Wextra`. Use man or Google to find out the meaning of the four flags, i.e. `-std=c99`, `-pedantic`, `-Wall` and `-Wextra`.
2. Briefly describe what a preprocessor, compiler and linker do when transporting C code into executable?
3. What does an undefined behavior mean in programming? Do we need to care about it? Why or why not?
4. What does the modifier const mean?
5. What are the primitive types in C and what are their byte sizes?
6. What is the value of `7 / 2` (a division of two integers) in a C program?

1. The command to compile a C program is `gcc <source file> -std=c99 -pedantic -Wall -Wextra`. Use man or Google to find out the meaning of the four flags, i.e. `-std=c99`, `-pedantic`, `-Wall` and `-Wextra`.

- `-std=c99`: Use the C99 version of the C language
- `-pedantic`: Strictly adhere to the language specification
- `-Wall`: enable (almost) all warnings
- `-Wextra`: enable extra compiler warnings

2. Briefly describe what a preprocessor, compiler and linker do when transporting C code into executable?

- Preprocessor: handles include files (`#include`), macros (`#define`), conditional compilation (`#ifdef`, `#ifndef`, `#endif`)
- Compiler: translates preprocessed source code into machine-specific assembly language
- Assembler: translates assembly language files into "object code" (machine language)
- Linker: joins object files together into an executable

3. What does an undefined behavior mean in programming? Do we need to care about it? Why or why not?

Example:

```c
#include <stdio.h>
int main(void) {
  int x;
  printf("%d\n", x);
  return 0;
}
```

Undefined behavior means that the behavior of the program can't be predicted. Programs with undefined behavior can't be relied on to do anything useful!

4. What does the modifier `const` mean?

`const` means "read-only".

E.g.:

```
const float PI = 3.14159;

PI = 3.0; // not allowed, compile error
```

5. What are the primitive types in C and what are their byte sizes?

Data type	Typical size in bytes
`char`	1
`int`	4
`long`	8
`float`	4
`double`	8

Note that C mandates a minimum range of values for each data type, but in practice that range could be larger. For example, `int` is guaranteed to allow a range of at least $-32,768$ to $32,767$ (i.e., 2 bytes), but supports a much larger range on most modern systems.

6. What is the value of `7 / 2` (a division of two integers) in a C program?

`7 / 2` = 3. This is because `7` and `2` are both integer (`int`) values, and a division of two integer values is an *integer division* where

- the result is an integer, and
- the fraction is discarded

Another example: `19 / 4` = `4`

The C language

- The first half of the course will focus on programming in C
- It is a *low-level*, "systems" programming language
  - Very close to the machine
  - Directly exposes machine-level concepts like
    - hardware-supported numeric data types
    - memory addresses

Hello world in C

```c
// hello_world.c:

#include <stdio.h>

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

Compiling and running the program:

```
$ gcc hello_world.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out
Hello, world!
```

How to try this out on ugrad?

Use `ssh` (or PuTTY) to log into your ugrad account.

Use `mkdir` to create a directory to put your code in.

Use `nano` to edit the source file. (By Wednesday you will know how to use a better editor, `emacs`.)

Use `gcc` to compile the source code into an executable.

Run the executable.

Reading input, computation, printing a computed value

```c
// add.c:
#include <stdio.h>

int main(void) {
  int a, b, sum;
  printf("Enter two integers: ");
  scanf("%d", &a);
  scanf("%d", &b);
  sum = a + b;
  printf("Sum is %d\n", sum);
  return 0;
}
```

Compiling and running the program:

```
$ gcc add.c -std=c99 -pedantic -Wall -Wextra
$ echo "2 3" | ./a.out
Enter two integers: Sum is 5
```

Some C numeric data types

Data type	Description
`char`	Character data type, typical range $-128 \ldots 127$
`int`	Integers, typical range $-2^{31} \ldots 2^{31} - 1$
`long`	Integers, typical range $-2^{63} \ldots 2^{63} - 1$
`float`	Floating point (approximate real number), 32 bit
`double`	Floating point, 64 bit

printf and scanf placeholders

Use these in `printf` and `scanf` format strings to designate output values (`printf`) or variables in which to store input values (`scanf`)

Data type	Placeholder
`char`	`%c`
`int`	`%d`
`long`	`%ld`
`float`	`%f`
`double`	`%lf`

In-class activities

Now we will switch to working on the in-class exercises. (Today: Exercise 1 and Exercise 2.)

Choose as breakout room as follows:

- Breakout rooms 1–10: the "social" breakout rooms. Join one of these if you would like to be able to chat with other students (which we very much encourage!)
- Breakout rooms 11 and above: the individual and small group breakout rooms. Join one of these if you prefer to work by yourself, or if you are planning to work with specific fellow students (who would then join the same room as you.)

How to ask for help!

Zoom features for communication between breakout rooms and the main meeting are very limited.

So, we will use our Slack workspace ("Intermediate Programming Summer 2022 Discussion") for messaging during in-class exercises.

If you need help, post to the `#general` channel and say which breakout room you're in. I or one of the CAs will join you when we're available.

Things that most definitely need to get done today!

- Have your ugrad account and be able to use it
- Be signed up for Piazza and Gradescope
- Have a Github account
- Submit the Google form to register your Github account (see recent emails from me for link)
- Accept the invitation to the `jhu-ip` Github organization (check the email account you used to sign up for Github)

By tomorrow you should have access to two repositories on Github: your personal repository, and the class "public" repository.

Notes

Notes

Notes

Notes

Notes