WWU CISS Program Boot Camp Notes

# Goals/Rationale

## Last cohort struggled

### Content of 247

### Adapting to expectations of WWU

## Goal is to prepare you

## Funded by Grant to Corrinne & Dept.

## I did very well

### I knew C

### I knew WWU

### I knew what to expect

### I knew how to take notes

# Stage-setting

## This boot camp is based on the 247 I took

## We’ll do lecture stuff and hands-on

## We’ll implement things every hour or so

## I’ll give you "homework" to tackle

## You don’t have to show up, but later stuff will build on earlier, so you might struggle if you miss sessions

# Git

## What is it?

### Version control system

### Created by Linus Torvalds

### Kind of like snapshots in VMs

### Can easily revert to earlier version, without losing the later versions

### Key difference: can synchronize to remote storehouses of versions

### Lots of other features: branches, forks, …

## Examine Github online

### <https://github.com/davehirsch/>

### Go to SwipeViewSwiftly

### Click on commits

### What is a commit?

#### Click on Version 1.1 to see what changed

### Where did I make those changes?

#### Not on Github

#### Local machine, then pushed (copied) to Github

#### Allows powerful offline use

## Hands-on with Git

Go to Github and create a personal account

Verify the email address

Login

### Create a repository

Call it HelloWorld

Public

Don’t set up a README

Choose "C" for .gitignore type

### Go to Linux

#### Create the program

In ~/Documents make a "bootcamp" directory

In there, make "hello" directory

In there right-click and Open in Terminal

nano hello.c

##### Type in the hello, world program from the text:

#include <stdio.h>

int main() {

printf("Hello, world\n");

}

##### Save and exit

##### Verify it looks right with more hello.c or cat hello.c

#### Set up git for the directory

be sure you are in the hello directory

git init

type ls -al

cd .git

ls -al

Never mess with this stuff!

cd ..

#### You have to tell git which files it should care about! It doesn’t know!!

git add hello.c

#### Now we want to commit this change (adding the file) to the repository

git commit -m "Added hello.c"

Gives warning about username and email and instructions on how to change them.

Follow those instructions

Last line puts you into the editor for the commit message, because you didn’t specify it on the command line. It’s VI. Remember the :wq! is save and exit.

Also: git config --global core.editor "nano"

#### Let’s look at your GitHub - has it changed? No.

Click clone or download. Copy the url to the clipboard. Should end in .git

##### Now tell git on linux where your remote repository is:

###### git remote add origin <url>

##### Before you can push your changes to the remote, you have to incorporate what is already there (the .gitignore file):

git pull origin master

Save the default merge comment, or change it if you like.

##### Now you can push your changes

git push origin master

(enter your credentials)

Go to Github and see your file!

##### Let’s make a change

nano hello.c

Change the hello greeting somehow.

Save and exit

###### How can this get saved to Github?

Commit locally and push to remote?

git commit - gives error

What do we need to do?

Add the changed file first, or put -a in the commit command

git commit -a (-a automatically adds all changes for already known files)

git push origin master

##### Try making two changes, committing each one, then pushing only after the second.

##### Try making a change and typing

git status

git diff

##### Note that there are ways to store your credentials in the local git so you don’t have to type them every time.

Google it.

See if you can get it to work

##### See what all your changes were:

git log

# Let’s run the hello, world program

gcc hello.c -Wall -o hello

should give a warning about "control reaches end of non-void function

(-Wall; try omitting the -Wall)

add a line before the last brace: return 0;

re-run the compilation

new file exists, hello, and it is executable (ls -l)

run it: ./hello

# C programming (vs. Java & Python)

## Much lower-level:

### closer to the machine code

### runs faster

### more flexible

Want to have the bits of the number 71937649201 be interpreted as a string? No problem. (The string is not "71937649201")

### easier to make mistakes

Want to go reading values right past the end of your array? No problem.

### less protection from language

### Harder to write and maintain big sprawling programs

### Can access memory more or less directly

## Must be compiled

### Python is interpreted

### Java is compiled, then interpreted

## C is not object-oriented!

### Both Python and Java have objects

### C++ is the OO extension of C

### C is procedural

## In C, you have to care about memory

### In Java and Python, it all gets taken care of for you

## In C, you have to care about how compilation happens

### like #include <stdio.h>

tells your program how printf() is to be called

### In Java and Python, much less so

## In C, your program crashes if there’s an error

### In others, it throws an exception, which can be handled

# C syntax

## White space rarely matters

Other than preprocessor lines (#), your whole program can be one line

Make hello.c a 2-line program and compile it.

## How do you tell C that you are done with a line and starting another?

Semicolon!

## How do you group a block of statements?

Body of a function

Body of a loop

> {Curly Braces!}

# C variables

## In C, you have to declare a variable to have a type:

int value;

## Cannot change variable types

## You may also initialize the variable in the declaration line:

int value = 0;

# Printf

## This outputs data to standard out

## There’s also scanf(), which takes data from std in

## Also the versions that do i/o with strings:

sprintf()

sscanf()

## printf() takes a format string, and a set of variables.

printf("The number is: %d.\n", 42);

The \n is the new line at the end.

The %d is a code meaning integer (see p. 153-155).

Put this into your hello.c and run it.

Then replace the "42" with an integer variable called value with an initialization to a number of your choice.

# Data types

## There are only a few basic data types:

### char (actually a small integer)

### int

### float

### double

## Notably strings are not really a thing!!

### Just an array of char, terminated by a byte with the value zero (‘\0’ as char);

### ‘x’ != "x";

### Can declare and initialize a string variable with:

char name[] = "David";

char name[] = {‘D’, ‘a’, ‘v’, ‘i’, ‘d’, ‘\0’};

printf("Hello, %s!\n");

## Integer types (char, short, int, long) can be signed or not

unsigned int nonNegativeValue

## Can transform one type into another: casting

doesn’t change the value of the thing; makes a copy in another type

floatval = (float) value;

printf("The float’s value is: %f\n");

float pi = 3.1415926;

int intpi = (int) pi;

printf("Pi as int = %d.\n", intpi);

(Doesn’t work with strings! Strings are not a thing in C.)