Setting up your programming environment:

We will be programming in our own server for the course, systems.cs.uic.edu. If you have previously gotten an account on a CS machine, such as bert.cs.uic.edu, the same username and password should work for our system, and you should have access to your CS home directory. If you do not already have a CS account, email Phil Beltran (pbeltr1@uic.edu) and request one. You should use ssh to connect to the server.

git and github

The first objective of this homework is to get you familiarized with the versioning system we will be using for homework turn-in, called git. Git is a decentralized revision control system. We will be using it with github, a website which allows you to host and share code repositories.

If you do not already have a github account, create a new account on github.com. You can choose any username you like.

Use this link to create a repository for lab 1.

Use the instructions on github to clone your new Lab 1 repository into your home directory on the server.

Create a textfile called netid.md that contains your netid. You will need to add this file to all of your lab repositories, so that we can give you a grade for your lab. Add this file to your repository, commit, and push. If you do these things correctly, you should be able to see this new file via the github web interface for your repository.

If you've never used git before, I'd suggest getting up to speed on it: you can learn some of the basics interactively here.

The Programming Part!

This part will give you a quick introduction to using `readelf` to better understand the linking process.

In this assignment, you must fill hw1.c with code which will:

- cause your username (and nothing else) to be printed on the first line of output when the program is run.
- cause `gcc -Wall hw1.c` to issue zero warnings (and zero errors, duh).
- cause the output of `readelf -s hw1.o` to have identical values in the bolded sections of the output below:

Symbol table '.symtab' contains 18 entries:

Num:	Value	Size	Type	Bind	Vis	Ndx	Name
0:	0000000000000000	0	NOTYPE	LOCAL	DEFAULT	UND	
1:	0000000000000000	0	FILE	LOCAL	DEFAULT	ABS	hw1.c
2:	0000000000000000	0	SECTION	LOCAL	DEFAULT	1	
3:	0000000000000000	0	SECTION	LOCAL	DEFAULT	3	
4:	0000000000000000	0	SECTION	LOCAL	DEFAULT	4	
5:	0000000000000000	0	SECTION	LOCAL	DEFAULT	5	
6:	0000000000000000	0	SECTION	LOCAL	DEFAULT	7	
7:	0000000000000000	0	SECTION	LOCAL	DEFAULT	8	

8:	0000000000000000	0	SECTION	LOCAL	DEFAULT	6	
9:	0000000000000000	81	**FUNC**	**GLOBAL**	**DEFAULT**	**1**	**main**
10:	0000000000000000	**0**	**NOTYPE**	**GLOBAL**	**DEFAULT**	**UND**	**printf**
11:	0000000000000051	38	**FUNC**	**GLOBAL**	**DEFAULT**	**1**	**all**
12:	0000000000000004	**4**	**OBJECT**	**GLOBAL**	**DEFAULT**	**COM**	**your**
13:	0000000000000077	38	**FUNC**	**GLOBAL**	**DEFAULT**	**1**	**cs361**
14:	0000000000000004	**1**	**OBJECT**	**GLOBAL**	**DEFAULT**	**3**	**are**
15:	0000000000000004	**4**	**OBJECT**	**GLOBAL**	**DEFAULT**	**COM**	**belong**
16:	0000000000000000	**4**	**OBJECT**	**GLOBAL**	**DEFAULT**	**4**	**to**
17:	0000000000000008	**37**	**OBJECT**	**GLOBAL**	**DEFAULT**	**COM**	**us**

Hints:

- Are you seeing `puts` instead of `printf`? Check the man pages for what the difference is, and make sure that the compiler can't optimize away your call to printf. Remember, requirement #1 **only** mentions the **first** line of output.

- Function lengths are very difficult to reproduce - note that for every FUNC, you do not have to duplicate the length (the length is the number of bytes of assembly code chosen by the compiler to execute the body of each function).

Template

There is no skeleton code for this assignment, only a Makefile.

Turn-in instructions

When you have completed your assignment, make sure you add, commit and push hw1.c to your repository.

To turn in your assignment, create a new branch in your git repository called "submission" that contains the final version of your code.

Make sure the branch you are submitting contains the following files: netid.md (containing your netid), hw1.c (containing your homework code), and Makefile (Makefile should be identical to the version of the Makefile we provided you).

Testing

We have added two files to allow you to automatically test your code to your repos: Jenkinsfile, and submit.h. If your repo did not contain these files, you can download them from piazza. Running submit.h will both submit your current branch by pushing to the submission branch, and automatically run test scripts which will test your code and display the results on github.