# HHt hhhhh

**Project1:The ELF format**

**Due Friday, June 26th, 11:59 PM**

**git, personal and public repositories**

The primary objective of this homework is to get you familiarized with the ELF file format and the process of linking executables. The secondary goal of this assignment is to give you some experience using the Gradescope autograder and turnin system.

We covered using git in the first lab, but if you’re still not that familiar with it, this assignment is a second chance to get up to speed. Remember: git gives you warnings and errors *for good reason*, if it complains at the command line when you run a command, don’t just assume it’s completed correctly!

Setting git to the simple push is a helpful quality of life improvement, this is recommended but not required:

git config --global push.default simple

You should also tell git who you are, otherwise it will complain that you are some anonymous coder. Do this, but use your own name and email. Note that if you do not change your username and email properly, you won’t get credit.

$ git config --global user.name "Copypasting Carol"

$ git config --global user.email [copypasta@domain.invalid](mailto:copypasta@domain.invalid)

The skeleton code for this assignment is available at [this link](https://www.google.com/url?q=https%3A%2F%2Fclassroom.github.com%2Fa%2FDF06Jgr2&sa=D&sntz=1&usg=AFQjCNH7Ys6avt1SeehrH25QYwOM87Cr0w). You must use GitHub classroom to manage your code (the repository created through that link). You will need to submit the code via [Gradescope](https://www.google.com/url?q=https%3A%2F%2Fwww.gradescope.com%2F&sa=D&sntz=1&usg=AFQjCNEUgDFDytIKRdE_pnhnQ-yLwQnnbw).

Now that you have the skeleton code, you can start coding. You should commit early and often, and push to your remote repository whenever is convenient to back up your work!

**Programming environment**

This assignment was created in a Debian-derived Linux environment. This assignment is simple enough that any Linux environment with an up to date gcc will be sufficient, including systems{1,2,3,4}.cs.uic.edu. The autograder is written to the correct specification for the autograding environment. Feel free to complete this on a lab machine, a local Linux VM, or elsewhere.

**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 and hw1.h with code which will:

1. cause your UIC netID (and nothing else) to be printed on the first line of output when the program is run.
2. cause gcc -Wall hw1.c to issue zero warnings and zero errors.
3. cause the output of readelf -s hw1.o to have an identical number of symbol table entries as shown below (25 entries).
4. cause the output of readelf -s hw1.o to have identical values in the bold sections of the output below (order does not matter):

You can perform the symbol table check by running make test within the hw1 directory. A full credit solution will have output that looks like:

Symbol table ‘.symtab’ contains 25 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 11 **FUNC** **LOCAL** DEFAULT **1** **I\_have\_written**

6: 0000000000000000 **12** **OBJECT** **LOCAL** DEFAULT **3** **the\_code**

7: 000000000000000b 26 **FUNC** **LOCAL** DEFAULT **1** **that\_you\_needed**

8: 0000000000000018 **4** **OBJECT** **LOCAL** DEFAULT **3** **to\_compile**.2213

9: 0000000000000000 0 SECTION LOCAL DEFAULT 5

10: 0000000000000058 78 **FUNC** **LOCAL** DEFAULT **1** **and\_which**

11: 000000000000001c **4** **OBJECT** **LOCAL** DEFAULT **3** **has\_a\_bunch\_of**.2220

12: 0000000000000020 **4** **OBJECT** **LOCAL** DEFAULT **3** **ridiculous**.2221

13: 0000000000000000 **8** **OBJECT** **LOCAL** DEFAULT **4** **symbols**.2222

14: 0000000000000000 0 SECTION LOCAL DEFAULT 7

15: 0000000000000000 0 SECTION LOCAL DEFAULT 8

16: 0000000000000000 0 SECTION LOCAL DEFAULT 6

17: 0000000000000000 7 **FUNC** **GLOBAL** DEFAULT **1** **sides\_and**

18: 0000000000000025 61 **FUNC** **GLOBAL** DEFAULT **1** **main**

19: 0000000000000000 **0** **NOTYPE** **GLOBAL** DEFAULT **UND** **\_\_GLOBAL\_\_OFFSET\_\_TABLE\_\_**

20: 0000000000000000 **0** **NOTYPE** **GLOBAL** DEFAULT **UND** **printf**

21: 00000000000000a6 47 **FUNC** **GLOBAL** DEFAULT **1** **Forgive\_me**

22: 0000000000000004 **4** **OBJECT** **GLOBAL** DEFAULT **COM** **they\_are\_arbitrary**

23: 0000000000000010 **8** **OBJECT** **GLOBAL** DEFAULT **3** **so\_random**

24: 00000000000000d5 41 **FUNC** **GLOBAL** DEFAULT **1** **and\_so\_varied**

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.
* The four digit numbers do not need to made identical, but you do need to make them show up. Keep experimenting with different variable types until you find how to create variables with periods and numbers on them.
* 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).