Lab 2

Unix tools for binaries

Preliminaries

Use an editor to write the following program

func.c:

```c
#include <stdlib.h>
#include <stdio.h>

int foo(void) {
    return rand();
}

int main(int argc, char* argv[]) {
    int m1, m2 = 0;
    int (*fptr)(void) = NULL;

    m1 = foo();

    fptr = foo;
    m2 = fptr();

    fprintf(stderr, "Got: %d %d\n", m1, m2);
}
```

Compile and run.

```
$ gcc -Wall func.c -o func
$ ./func
Got: 1804289383 846930886
```

Compile also just the code, without transforming it to an executable.

```
$ gcc -Wall -c func.c -o func.o
```

Steps

1) Try to disassemble the executable and the object file. Notice that the executable contains much more machine code than the object file. Where does this code come from?
2) List the sections of both files. Filter all the executable sections.
3) List all symbols that are stored in the symbol table or the dynamic symbol table for both files and observe their values. Now filter only symbols associated with a function name.
4) Disassemble the object file and spot the direct and indirect branch.
5) Use gdb to infer how the indirect branch is resolved in the executable.