CS 45: Operating Systems

CLab 2: Tearing apart `printf`

DUE by next class

Write your answers in `README.md`. Create a `COLLAB.md` file to keep track of any outside resources you might use. Be sure to push to the repo after class (even if you are not done).

Read the article on `Tearing apart printf`, follow along in your terminal, and answer the following questions.

1. Put the following stages in order: (run time, compile time, design time, load time, link time).
2. What tools are we using (fill in the output)?

```
$ uname --version

$ uname -a

$ gcc --version

$ ldd  --version

$ nm --version

$ ltrace  --version

$ strace  --version

$ objdump  --version
```

3. What does `printf` return?
4. What do the symbols in your object code look like for the `printf` and the more useful `printf` examples (step 3)?
5. Which binary is bigger `printf1` or `printf1_s`? Why?

```
$ ls -lh printf1_*
```

6. Use `ltrace` to see what *library calls* your `printf1` makes.
7. use `strace` to see what system calls both your `printf1` and `printf1_s` make.
8. Show the memory map of both `printf1` programs. (You might at a `getchar` to pause the program long enough to find its PID).

```
$ ./printf1 &
$ cat /proc/__YOUR_PID__/maps
$ ./printf1_s &
$ cat /proc/__YOUR_PID__/maps
```

9. Which system call is actually responsible for sending the string to the kernel to be displayed?
10. We can use gdb to trace the execution path of our program.

```
$ gdb ./printf1_s
set logging enabled on
[...]
(gdb) rbreak .
[...]
# hit `c` to avoid pager
(gdb) commands
```

```
Type commands for breakpoint(s) 1-3, one per line.
End with a line saying just "end".
>silent
>bt 1
>cont
>end
(gdb) r
Starting program: [...]/printf1_s
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[...]
(gdb) quit
$ code gdb.out
```

11. Explain the three parameters to `write`.

12. What system call number is `write`?

```
$ objdump -d printf1_s  > out.asm
$ grep -C 4 "<__write_nocancel>:" out.asm
```

13. `strace` said `execve` was the first system call invoked, using the diassembly in `out.asm`, find what syscall number it is.