CS 45: Operating Systems

CLab 6: memory segmentation and `efence`

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).

Electric Fence (`efence`) is a tool to detect memory issues with our `C` programs.

1. Compile and run a version of `memory-user.c`. Use `pmap` to see how much heap space is used.

```
$ gcc -g  -o memory-user memory-user.c
$ ./memory-user 1 &
$ pmap PID
$ kill PID
```

2. Compile a program `concat.c` that simply combines all the command-line arguments into one big string.

```
$ gcc -g  -o concat concat.c
$./concat a b
ab
$./concat abc def ghi
abcdefghi
```

3. What happens when you run a larger example?

```
$ ./concat abcdefghij abcdefghij abcdefghij
```

4. Does using `gdb` help find the bug?

```
$ gdb ./concat
$ run abcdefghij abcdefghij abcdefghij
$ bt
```

5. Let's run the program when linked against electric fence.

```
$ LD_PRELOAD=libefence.so ./concat abcdefghij abcdefghij abcdefghij
```

6. Let's try again to use `gdb` to find the bug!

```
$ gdb --args env LD_PRELOAD=libefence.so ./concat abcdefghij abcdefghij abcdefghij
$ run
$ bt
```

7. How does the process memory map look different when using electric fence?

```
$ gcc -g  -o memory-user memory-user.c
$ LD_PRELOAD=libefence.so ./memory-user 1 &
$ pmap PID
$ kill PID
```

8. What are the main functions `efence` exports (look for non-static functions in `efence.c`?

9. Alternatively, check the library for its symbols marked with `T`:

```
$ nm -gD /usr/lib/libefence.so
```

10. What system calls is electric fence relying on to do it's job?

11. The examples in `concat2.c` and `concat3.c` manifests this bug in slightly different ways, but have the same underlying problem. `efence` and `valgrind` both come to the rescue in both cases. For example, I wasn't able to make `concat3.c` actually crash, but valgrind and efence told me there was a problem. Create a `concat_fixed.c` that works correctly and reports no errors when run with `valgrind`.

NOTE: Always use the `n` versions of the string functions in `C` that require an explicit buffer size: `strncpy`, `strncat`, and `fgets`, etc.