Notes for Programming in C Lab Session #9

25 October 2017

1 Introduction

The purpose of this lab session is to experiment with programs exhibiting undefined behaviour.

2 Overview

Unlike the earlier labs, there is very little code for you to write. Instead, the focus of the lab is on looking at a small collection of C programs exhibiting undefined behaviour, and (a) seeing if you can compiler flags to change the behaviour of the program, and (b) understanding where and what undefined behaviour is being used.

3 Instructions

1. Download the `lab9.tar.gz` file from the class website.
2. Extract the file using the command `tar xvzf lab9.tar.gz`.
3. This will extract the `lab9/` directory. Change into this directory using the `cd lab9/` command.
4. In this directory, there will be files `problem1.c`, `problem2.c`, `problem3.c`, `problem4.c`, `problem5.c`, and `problem6.c`.
5. There are **NO** makefiles for these programs — just compile them with an explicit invocation of gcc or clang:

```
$ gcc <flags> -o problemN problemN.c
```

4 Problems

Each of the problems in this list is also a file in the lab package. Note that undefined behaviour, by its very nature, is unpredictable. Depending on your platform and compiler, each of these programs may exhibit some surprising behaviour or not – you will need to experiment with optimization levels (and switching between gcc and clang, if you have both installed).

1. Consider the following program:

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

int main(int argc, char **argv) {
    char a[2];

    a[0] = 'a';

    if(!strcmp(a, "a")) {
        puts("a is \"a\"");
    }

    return 0;
}
```

- What does this program do when compiled with no optimization?
- What does this program do when compiled with optimizations (i.e., with the -O2 or -O3 flag)?
- What is the undefined behaviour in this program?

2. Consider the following program:

```
#include <stdio.h>

void foo(int i) {
  while (i) {
    /* loop? */
  }
}

int main(void) {
  foo(1);
  printf("Done!?\n");
  return 0;
}
```

   - What does this program do when compiled with no optimization?
   - What does this program do when compiled with optimizations (i.e., with the -O2 or -O3 flag)?
   - What is the undefined behaviour in this program?

3. Consider the following program:

```
void foo(int i) {
  foo(i+1);
}


int main(void) {
  foo(0);
  return 0;
}
```

   - What does this program do when compiled with no optimization?
   - What does this program do when compiled with optimizations (i.e., with the -O2 or -O3 flag)?
   - Why did the behaviour change?

4. Consider the following program:

```c
#include <stdio.h>

int *G;

int f(void) {
  int l = 1;
  int res = *G;
  G = &l;
  return res;
}

int main(void) {
  int x = 2;
  G = &x;
  f();
  printf("%d\n", f());
}
```

- What does this program print when compiled with no optimization?
- What does this program print when compiled with optimizations (i.e., with the -O2 or -O3 flag)?
- What is the undefined behaviour in this program?

5. Consider the following program:

```c
int main(void) {
  int i = 1;
  while (i > 0){
    i *= 2;
  }
  return 0;
}
```

6. Consider the following program:

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

typedef struct {
  int a,b,c;
} st;

int main(void) {
  st obj = {10,12,15};
  st *p1 = &obj;
  st *p2 = NULL;
  intptr_t n = (intptr_t) &p2->b;

  printf("%ld\n", *(intptr_t *) ((char*)p1 + n));
  return 0;
}
```

- Where is the undefined behaviour in this program?

- Was there a segmentation fault?
- Why or why not?