Programming in C

15-213 Recitation 6

Weekly Update

Low traffic on mailing list/in office hours. How are people getting help?

Buffer lab due Tuesday (tomorrow), 11:59PM

- Try not to use late days

Cache lab out Tuesday (tomorrow), 11:59PM

- First programming lab
- Due Thursday, February 27

First programming lab

Agenda

Intensive C workshop
and self-diagnosis

Style

GDB and C

Hunting memory bugs

Intensive C workshop

Wednesday, 6:30 PM. Location TBA

Do exercises

Much of what you need to know

Come to office hours for help with exercises

Should you go?

Can you answer these next 3 questions?

Can you answer these next 3 questions *effortlessly?*

Q1: 2D Arrays

```
int A[40][30];

int **B = malloc(sizeof(int*) * 40);
for (size_t i = 0; i < 30; i++)
  B[i] = malloc(sizeof(int) * 30);
```

Does `sizeof(A) == sizeof(B)`?

Q2: Macros

```
#define IS_GREATER(a, b) a > b

int is_greater(int a, int b) {
   return a > b;}

int A = IS_GREATER(1, 0) + 1;
int B = is_greater(1, 0) + 1;
```

What is `A`? What is `B`?

Q3: Types

```
typedef char (*(*arrptr[3])())[10];

arrptr x;
```

A. `x` is a pointer to an array of three pointers
B. `x` is an array of ten character arrays
C. `x` is an array of three function pointers
D. Compiler error

If these gave you trouble,
come to the workshop

Wednesday, 6:30PM. Location TBA

Agenda

Intensive C workshop
and self-diagnosis

Style

GDB and C

Hunting memory bugs

Style

Good documentation

- Header comments, large blocks, tricky bits

Check error/failure conditions

- Must program robustly

80 characters per line

No memory or file descriptor leaks

Style

Use interfaces for data structures

E.g. a linked list should have create/insert/remove/free functions

Modularity of code

No magic numbers

Use `#define`

Consistency and whitespace

Agenda

Intensive C workshop
and self-diagnosis

Style

GDB and C

Hunting memory bugs

GDB still exists

```
prandolp@lemonshark:~$ whereis gdb
gdb: /usr/bin/gdb /usr/share/gdb /usr/share/man/man1/gdb.1.gz
prandolp@lemonshark:~$
```

Recompiling with GDB

Don’t quit GDB

Edit source files in another terminal

In GDB, type `make` then `refresh`

Note: some breakpoints may move after recompiling

gdbtui

stack_smash.c

```
1       #include <string.h>
2       #include <stdio.h>
3
4       int main(int argc, char **argv){
5
B+ 6            int magic = 0x1337;
7               int beef = 0xbeef;
8
9               int magicbeef = magic + beef;
> 10            int beefmagic = magic*beef;
11
12              // what happens when we don't null terminate strings?
13              char small[4];
14              strcpy(small, "hah");
15
16              // what was in small[3] before?
17              small[3] = 'a';
```

child process 6852 In: main Line: 10 PC: 0x804846c

```
(gdb) break main
Breakpoint 1 at 0x804844d: file stack_smash.c, line 6.
(gdb) run
Starting program: /afs/andrew.cmu.edu/usr12/prandolp/private/213/rec5/stack_smash

Breakpoint 1, main (argc=1, argv=0xffffd254) at stack_smash.c:6
(gdb) next
(gdb) next
(gdb)
```

Using `gdbtui`

Compile with `-g` debug flag

```
gcc -g -m32 my_prog.c -o my_prog
```

Use the `gdbtui` wrapper command, not `gdb`

```
gdbtui my_prog
```

Using `gdbtui`: Layout

Many different layouts

Source, Assembly, Source/Assembly, Assembly/Registers…

`layout next/prev`

Display the next or previous layout

`layout src/asm/regs/split`

Display the source/assembly/registers/source & assembly layout

Using `gdbtui`: Focus

Focus controls which window receives scrolling

`focus next/prev`

Make next or previous window active for scrolling

`focus src/asm/regs`

Make the source/assembly/registers window active for scrolling

Demo

Agenda

Intensive C workshop
and self-diagnosis

Style

GDB and C

Hunting memory bugs

gdb

Useful for debugging “easy” segfaults

Run until segfault and evaluate the situation using…

`where` — print function stack and lines

`up/down` — traverse the function stack

`list` — print source code for current location

`display` — analyze the variables in use and see which is incorrectly using memory

Memory leaks

Allocate some memory with `malloc`

Throw away the pointer without using `free`

May cause memory use to grow unboundedly

blued	1.01 GB	4	111	38756	root

(bluetooth daemon using 1GB of memory)

valgrind

A suite of tools for memory debugging and profiling

Track memory leaks

Track possibly lost blocks

Track origin for uninitialized values

Report definitely lost blocks

The verbose `-v` flag is recommended

Finding leaks

```
valgrind --leak-resolution=high --
   track-fds=yes --leak-check=full --
   show-reachable=yes ./my_prog
```

Demo

Agenda

Intensive C workshop
and self-diagnosis

Style

GDB and C

Hunting memory bugs

Questions?