CSc 352

C Programming
Pointers, GDB, debugging

Benjamin Dicken

Announcements

- PA 2 grades published
- How was assignment 3?
- PA 4
- Look at the schedule

Activity

```
int x = 1;
int y = 2;
int r1 = 0;
int r2 = 0;

r1 = x++;                       // A
r2 = (x++)++;                   // B
x + y  = x + y;                 // C
*(&x) = (++y) + (r1++);         // D
*(&x) = (++y) + (x++);          // E
*(&x + y) = 10;                 // F
x++ = y++;                      // G
```

What is valid and what is not valid?

Activity

```c
int x = 1;
int y = 2;
int* xp = &x;
int* yp = &y;

*(++xp) = 30;  // A
y = (*xp)++;   // B
y = *(xp++);   // C
y = *((&x)++); // D
```

What is valid and what is not valid?

Uninitialized and Dangling Pointers

Uninitialized Pointer: A pointer that does not get assigned a value

- What happens when you look up a “random” address?

Dangling Pointer: Points to a location that is no longer valid

- Think: Points to a value that *was* on the stack but has been deallocated
- Think: Points to dynamically-allocated memory that has been freed

Activity

What do you think of this code?

```
char * get_name(char* prompt) {
  char buffer[32];
  printf("%s", prompt);
  scanf("%31s", buffer);
  return buffer;
}

int main() {
  char* name = get_name("Enter your name:\n");
  printf("Your name is %s\n", name);
  return 0;
}
```

What do you think of this code?

```c
void get_name(char* prompt, char** name) {
  char buffer[32];
  printf("%s", prompt);
  scanf("%31s", buffer);
  *name = buffer;
}

int main() {
  char* name;
  get_name("Enter your name:\n", &name);
  printf("Your name is %s\n", name);
  return 0;
}
```

What do you think of this code?

```
void get_name(char* prompt, char* name) {
  printf("%s", prompt);
  scanf("%31s", name);
}

int main() {
  char name[32];
  get_name("Enter your name:\n", name);
  printf("Your name is %s\n", name);
  return 0;
}
```

```
$ man gdb
```

Making your executable compatible

Use the -g option when compiling with GCC

Causes the executable to include debugging information

```
$ man gcc
```

Key options for gdb

break - sets a stopping points within the code

run - starts the program running

next / step - walk through the program

bt - backtrace

frame - show information for a stack frame

print - display the value of a variable / expr

Coin Program With GDB

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

int change(int amount) {
  if (amount == 1 || amount == 5 || amount == 10 || amount == 25) { // Base case
    return 1;
  }
  if (amount > 25) {
    return 1 + change(amount - 25); // Quarter
  } else if (amount > 10) {
    return 1 + change(amount - 10); // Dime
  } else if (amount > 5) {
    return 1 + change(amount - 5); // Nickel
  } else {
    return 1 + change(amount - 1); // Penny
  }
}

int main(int argc, char** argv) {
  int x = atoi(argv[1]);
  int number_of_coins = change(x);
  printf("Minimum coins needed: %d\n", number_of_coins);
  return 0;
}
```

Coin Program With GDB

```
lectura:> gcc -Wall -Werror -std=c11 -g coins.c -o coins
lectura:> gdb coins
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from coins...
(gdb)
```

Type commands here

```
(gdb) break 5
Breakpoint 1 at 0x1178: file coins.c, line 5.
(gdb) run 27
Starting program: /home/bddicken/test/coins 27

Breakpoint 1, change (amount=27) at coins.c:5
5      if (amount == 1 || amount == 5 || amount == 10 || amount == 25) { // Base case
(gdb) bt
#0  change (amount=27) at coins.c:5
#1  0x0000555555555224 in main (argc=2, argv=0x7fffffffe8f8) at coins.c:21
(gdb)
```

```
(gdb) step
8	    if (amount > 25) {
(gdb) step
9	        return 1 + change(amount - 25); // Quarter
(gdb) step
change (amount=21845) at coins.c:4
4	int change(int amount) {
(gdb) step

Breakpoint 1, change (amount=2) at coins.c:5
5	    if (amount == 1 || amount == 5 || amount == 10 || amount == 25) { // Base case
(gdb) bt
#0  change (amount=2) at coins.c:5
#1  0x00005555555551aa in change (amount=27) at coins.c:9
#2  0x0000555555555224 in main (argc=2, argv=0x7fffffffe8f8) at coins.c:21
(gdb)
```

```
(gdb) info frame 0
Stack frame at 0x7fffffffe7c0:
 rip = 0x555555555178 in change (coins.c:5); saved rip = 0x5555555551aa
 called by frame at 0x7fffffffe7e0
 source language c.
 Arglist at 0x7fffffffe798, args: amount=2
 Locals at 0x7fffffffe798, Previous frame's sp is 0x7fffffffe7c0
 Saved registers:
  rbp at 0x7fffffffe7b0, rip at 0x7fffffffe7b8
(gdb) info frame 1
Stack frame at 0x7fffffffe7e0:
 rip = 0x5555555551aa in change (coins.c:9); saved rip = 0x555555555224
 called by frame at 0x7fffffffe810, caller of frame at 0x7fffffffe7c0
 source language c.
 Arglist at 0x7fffffffe7b8, args: amount=27
 Locals at 0x7fffffffe7b8, Previous frame's sp is 0x7fffffffe7e0
 Saved registers:
  rbp at 0x7fffffffe7d0, rip at 0x7fffffffe7d8
(gdb) info frame 2
Stack frame at 0x7fffffffe810:
 rip = 0x555555555224 in main (coins.c:21); saved rip = 0x7ffff7de20b3
 caller of frame at 0x7fffffffe7e0
 source language c.
 Arglist at 0x7fffffffe7d8, args: argc=2, argv=0x7fffffffe8f8
 Locals at 0x7fffffffe7d8, Previous frame's sp is 0x7fffffffe810
 Saved registers:
  rbp at 0x7fffffffe800, rip at 0x7fffffffe808
(gdb)
```

Activity

Debug

- Download **`code.c`** and **`makefile`** from the class website
- Without modifying the makefile or C file, determine:
  - What could cause this program to crash?
  - Why?
  - Use GDB
- I'll give you 5-7 minutes to download, test, explore with GDB, then we can discuss