Debugging C
(What to do when you make the mistakes from the last lecture)

David Chisnall

February 20, 2011
Buggy C Code

$ ./a.out
Segmentation fault: 11 (core dumped)

C gives very helpful feedback: you did something wrong. Somewhere.
What’s this ‘core’ thing?

$ ls *.core
a.out.core
$ gdb a.out a.out.core

• Core files contain a memory dump of the crashed program
• You can use them to inspect a program that’s crashed
Step 0: Compiler Warnings

$ gcc broken.c
$ clang -Wall broken.c

broken.c:9:12: warning: using the result of an assignment as a condition without parentheses [-Wparentheses]
    if (array = 0)
         ~~~~~~~~~~

broken.c:9:12: note: use ‘==’ to turn this assignment into an equality comparison
    if (array = 0)
         ~

    ==

broken.c:9:12: note: place parentheses around the assignment to silence this warning
    if (array = 0)
         ~

         (    )
Fix the Warnings First!

- If you really meant the assignment, add the brackets
- If you meant a comparison, fix the code
- Then fix any other warnings.
Step 1: Where is the bug?

- Bugs in valid C programs often cause crashing
- The first step is to find where the crash occurred
$ gdb a.out
"...(no debugging symbols found)...
(gdb) r
Starting program: /tmp/a.out
(no debugging symbols found)...
(no debugging symbols found)...
Program received signal SIGSEGV, Segmentation fault.
0x080485f6 in main ()

So, it crashed somewhere in main()?
Debugging Symbols

- The debugger needs information to map memory addresses back to the source code
- This is not provided by default

```
$ c99 -g broken.c
$ gdb ./a.out
(gdb) r
Starting program: a.out
Program received signal SIGSEGV, Segmentation fault.
0x08048515 in initialise () at broken.c:24
24 array[i] = random();
(gdb) bt
#0 0x08048515 in initialise () at broken.c:24
#1 0x08048592 in main (argc=1, argv=0xbfbfe72c) at broken.c:40
```
What Could Be Wrong Here?

array[i] = random();

- Memory at array[i] is not valid?
- Array out of bounds error?
Try Another Tool: Valgrind

- Valgrind emulates all load / store instructions
- Tracks memory accesses
- Logs useful errors
- *Much* slower than running the code normally
$ valgrind ./a.out
Memcheck, a memory error detector
Command: ./a.out

Invalid write of size 4
  at 0x8048515: initialise (broken.c:24)
  by 0x8048591: main (broken.c:40)
Address 0x1d8ab0 is 0 bytes after a block of size 400,000 alloc’d
  at 0x5A138: malloc
  by 0x80484B9: resize (broken.c:11)
  by 0x804858C: main (broken.c:39)
Invalid write of size 4
at 0x8048515: initialise (broken.c:24)
by 0x8048591: main (broken.c:40)

• Assigning a 32-bit integer value to memory is a write of size 4
• This shows where the crash was
• But we knew that already
The Interesting Bit

Address 0x1d8ab0 is 0 bytes after a block of size 400,000 alloc’d
at 0x5A138: malloc
by 0x80484B9: resize (broken.c:11)
by 0x804858C: main (broken.c:39)

- We allocated 400,000 bytes
- The stack trace tells us where
The Bug

// The allocation
array = malloc(count);

// The use:
for (int i=0 ; i<count ; i++)
{
    array[i] = random();
}

• What’s the error?
• Allocating 400,000 bytes
• Expecting 400,000 integers
The Fix

// Wrong:
array = malloc(count);
// Correct:
array = malloc(count * sizeof(int));
// Also correct:
array = calloc(count, sizeof(int));

Always remember that malloc() does not know the size of the variable, it just expects a size in bytes.
Heisenbugs

- Sometimes running in the debugger makes bugs vanish
- In this case, resort to older techniques
- Add log statements to find what’s wrong
printf("About to use array %p\n", array);

• Looks right?
• printf() writes to the standard output
• Standard output is buffered
• Buffer might not be flushed before crash
Better Logging

```c
fprintf(stderr,"About to use array %p\n",array);
```

- Standard error is unbuffered
- Output is written to the terminal immediately
Even Better Logging

```c
#ifdef DEBUG
#define LOG(msg, ...) \   fprintf(stderr, msg, ## __VA_ARGS__)
#else
#define LOG(...)
#endif

$c99 -g -DDEBUG code.c
Other Tools

- Clang static analyser
- DTrace (Solaris / Mac / FreeBSD)
- ...
Demo / Questions