Pre-lab Exercises - Worksheet8: Debugging and Testing

Even A. Nilsen 11.10.2016

1. GDB Commands

a) Inserts a breakpoint at the start of the function `deleteBook()`.

b) Runs the program until it hits the first breakpoint. In this case `deleteBook()`.

c) Will print the data type and its memory address.

d) Will print the struct and the fields plus their value at the time.

e) `watch` is like a breakpoint for variables. In this case `gdb` would stop when `book` is equal to `cat->books[i]`.

f) The debugger would continue until the watchpoint is hit and pause.

g) The debugger would skip to the next line and over any functions it hits. Currently on the line `cat->numBooks--;`.

h) Same as above, but it is now inside the while loop.

i) The debugger would continue running until just after the function in the selected stack frame returns. So it would pause just after `deleteBook()` was called.

2. Debugging Tactics

a) There might be a call to `scanf()` inside `readInt()` and the programmer forgot the `&` operator. `getBorrower()` could try to access a part of `blist` that is not allocated.

I would set a breakpoint on `menuReturnBook()` and watchpoint to check when `id` changed. I would then step through the function and try to find the error.

b) There might be a call to `scanf()` inside `readInt()` and the programmer forgot the `&` operator. `cat` might be uninitialized. Dereferencing an uninitialized pointer can cause a segmentation fault.

I would not use the debugger to check the call to `scanf()`, I would check the code directly. To check if `cat` is initialized I would set a breakpoint at `menuReturnBook()` and `print *cat`.

c) `returnBook()` could just be checking if a certain book is registered to a borrower and not actually changing the status of the book.

3. Unit Testing

a) I would create a testing harness that would read input from a file and simulate user input.

b) One unit test function per function implemented in a way so that I could supply them with different arguments to test the different scenarios.

c) One to check the `readInt()`. One for each of the conditionals. I am not sure, but 5.

Debugging Walkthrough

a) The program is aborted due to a segmentation fault after listing the first book.

b) `cat` is a struct containg an array of structs called `books`. Each `book` in `books` has a member named `onLoan`. `cat` is a pointer to a catalog struct. We point to its member named `books` wich is a pointer to an array of pointers to `book` structs. Then we point to book's member `onLoan` by accessing it via index.

c) The root cause of the segmentation fault is that `int eof = TRUE;` on line 19 in `catalogue.c`.

On Your Own

a) The program is not loading the borrower list properly. No names are loaded initially and when you add a new borrower the index is faulty saved as `j`.

b) `createBorrower` does not store the name properly in the `borrower` struct.

   `addBorrower` does not add the `borrower` to the `borrowerlist` properly.

   SOLUTION: In the call to `strncpy`, `name` was set to take on the value of `bor->name` instead of the opposite. That means that `bor->name` never got declared.