Debugging with gdb

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org

Debugging with gdb

gdb: GNU debugger

gdb helps you run your program in a way that allows you to:

- flexibly *pause* and *resume*
- print out the values of variables mid-stream
- see where severe errors like `Segmentation Faults` happen

When using gdb (or `valgrind`) we compile with `-g`, which packages up the source code ("debug symbols") along with the executable

gdb

Keep this “cheat sheet” nearby as you read on:

bit.ly/gdb_cheat

gdb

Buggy program:

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

void string_reverse(char *str) {
  const int len = strlen(str);
  for(int i = 0; i < len; i++) {
    str[i] = str[len-i-1]; // swap characters
    str[len-i-1] = str[i];
  }
}

int main() {
  char reverse_me[] = "AAABBB";
  string_reverse(reverse_me);
  printf("%s\n", reverse_me);
  return 0;
}
```

gdb

We are trying to reverse a string by starting at the left and right extremes, swapping the characters, then continuing inward, swapping as we go until we've reversed the whole thing.

```
$ gcc -o str_rev str_rev.c -std=c99 -pedantic -Wall -Wextra -g
$ ./str_rev
BBBBBB
```

Oops, I expected output to be BBBAAA

valgrind gives clean report, so likely not an issue with mishandled pointers

gdb

We'll use gdb to investigate

Since the problem would seem to be in the `string_reverse` function, I am going to start my program at the beginning and then take small steps forward until I get to the loop.

gdb

```
(gdb) break main
Breakpoint 1 at 0x4005ad: file str_rev.c, line 13.
(gdb) run
Starting program: /app/str_rev
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.26-15.fc27.x

Breakpoint 1, main () at str_rev.c:13
13    char reverse_me[] = "AAABBB";
```

`break main` because I want to debugger to pause as soon I as get to the beginning of the program, i.e. the `main` function

`run` to start the program, which immediately pauses at top of `main`

After running a command, gdb prints out the next line of code in the program

gdb

```
(gdb) next
14    string_reverse(reverse_me);
(gdb) step
string_reverse (str=0x7fffffffe629 "AAABBB") at str_rev.c:5
5     const int len = strlen(str);
```

`next` executes the statement on the current line and moves onto the next. If the statement contains a function call, gdb executes it without pausing.

`step` begins to execute the statement on the current line. If the statement contains a function call, it *steps into* the function and pauses there. Otherwise, it behaves like `next`.

Now we're at the beginning of `string_reverse`

gdb

```
(gdb) n
6     for(int i = 0; i < len; i++) {
(gdb) print len
$2 = 6
```

n is short for `next`

`print` prints out the value of a variable. `len` is 6 – that's what we expected. So far so good.

We're about to enter the loop.

gdb

```
(gdb) n
7       str[i] = str[len-i-1]; // swap characters
(gdb) p i
$3 = 0
(gdb) p str[i]
$4 = 65 'A'
(gdb) p str[len-i-1]
$5 = 66 'B'
```

p is short for `print`

i's initial value is 0, as expected

The elements we're swapping really are the first `A` and the last `B`, as expected

gdb

Let's execute the swap:

```
(gdb) n
8       str[len-i-1] = str[i];
(gdb) n
6     for(int i = 0; i < len; i++) {
(gdb) p i
$6 = 0
```

Just finished the first iteration; i still equals 0

gdb

Let's see if the swap was successful:

```
(gdb) p str[i]
$7 = 66 'B'
(gdb) p str[len-i-1]
$8 = 66 'B'
```

No – the swap fails because I overwrite `str[i]` with the value of `str[len-i-1]` *before* copying it into `str[len-i-1]`

This explains why the result is BBBBBB

I need to use a temporary variable like we did previously with swap

gdb

Fixed?:

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

void string_reverse(char *str) {
  const int len = strlen(str);
  for(int i = 0; i < len; i++) {
    int temp = str[i];  // swap characters -- FIXED
    str[i] = str[len-i-1];
    str[len-i-1] = temp;
  }
}

int main() {
  char reverse_me[] = "AAABBB";
  string_reverse(reverse_me);
  printf("%s\n", reverse_me);
  return 0;
}
```

gdb

```
$ gcc -o str_rev2 str_rev2.c -std=c99 -pedantic -Wall -Wextra -g
$ ./str_rev2
AAABBB
```

Still not working! I expected output to be BBBAAA

Exercise: use gdb to find lingering bug.

Hint 1: examine results of the swaps through *several* loop iterations

Hint 2: Instead of `break main`, use `break str_rev2.c:7`, replacing `str_rev2.c` with the name of your source file and `7` with the line number of the first statement in the loop body. That way `run` will advance directly there. (If you already set the `main` breakpoint, remove it with `delete`.)

gdb help

Type `help` at the `(gdb)` prompt for help

- `(gdb) help running` – for advancing thru program
- `(gdb) help show` – for printing commands

There are *many* gdb commands, so I prefer brief “cheat sheets”:

- darkdust.net/files/GDB%20Cheat%20Sheet.pdf