CS 121 **Lab Assignment #8**
10 Points
Bruce Bolden **Date:** March 26, 2015

1 What is a Debugger?

A debugger is a program that runs programs and has the power to suspend its execution to examine and change memory values. Most debuggers know about the data structures you are using and can display them in a useful manner.

2 gdb

`gdb` is the GNU project's debugger and was originally released in 1988. Since that time, a number of graphical interfaces have been developed for it, but many people still use the command line version for a multitude of reasons.

2.1 Using `gdb` From The Command Line

2.1.1 A Sample `gdb` Session

Compile code with debugging symbols `-g` option and rename the executable `-o` option.

```
% gcc -g -o memerror memerror.c
```

Run the program:

```
% ./memerror blarg
the reversed string is '(null)'
```

Run the program inside `gdb`:

```
% gdb ./memerror
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-50.el6)
Copyright (C) 2010 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from memerror...done.
(gdb) run
Starting program: memerror
usage: usage: %s string.  This reverses the string given
on the command line string.  This reverses the string
given on the command line

Program exited with code 0377.
(gdb) break main
Breakpoint 1 at 0x4006c3: file memerror.c, line 28.
(gdb) run
Starting program: memerror

Breakpoint 1, main (argc=1, argv=0x7fffffffe398)
    at memerror.c:28
28          if( argc != 2 )
(gdb) next
30              fprintf( stderr, "usage: %s string.
This reverses the string "
(gdb) print argc
$1 = 1
(gdb) next
usage: usage: %s string.  This reverses the string given
on the command line string.  This reverses the string
given on the command line
32              exit( -1 );
(gdb)
```

Recompile from inside `gdb`, delete first breakpoint, and run with a command-line argument:

```
(gdb) shell gcc -g -o memerror memerror.c
(gdb) run blarg
```

```
(gdb) print argc
$1 = 2
(gdb) print argv
$2 = (char **) 0xbffffc04
(gdb) print argv[0]
$3 = 0xbffffc9c "/Users/bruceb/Teaching/CS_121/Labs/memerror"
(gdb) print argv[1]
$4 = 0xbffffcc8 "blarg"
(gdb)
```

Things are looking better. Single-step to continue

```
(gdb) next
37          stringbuffer = malloc (strlen(argv[1]));
(gdb) call (int) strlen(argv[1])
$5 = 5
```

Single step over the allocation (commands may be abbreviated if they are not ambiguous).

```
(gdb) n
```

Strings are nul-terminated—we need an extra byte!

```
(gdb) p stringbuffer
$1 = 0x100140 "blarg"
(gdb) n
45          printf ("the reversed string is '%s'\n", *stringbuffer);
(gdb) n
the reversed string is '(null)'
47          return (0);
```

3 Deliverables

Documentation of at least six errors found and a two or three sentence proposal for how to fix each one. Be specific about what code you would change and why for each bug.

4 References

Video: GDB Debugger (11 minutes)
`http://www.youtube.com/watch?v=k-zAgbDq5pk`

Video: Intro to Debugging with GDB (18 minutes)
`http://vimeo.com/27422628`

GDB: The GNU Project Debugger `http://www.gnu.org/software/gdb/gdb.html`

GDB Documentation `http://www.gnu.org/software/gdb/documentation/`
366 pages

Debugging with `gdb` `http://www.delorie.com/gnu/docs/gdb/gdb_toc.html`

GDB and LLDB Command Examples
`https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/gdb_to_lldb_transition_guide/document/lldb-command-examples.html`