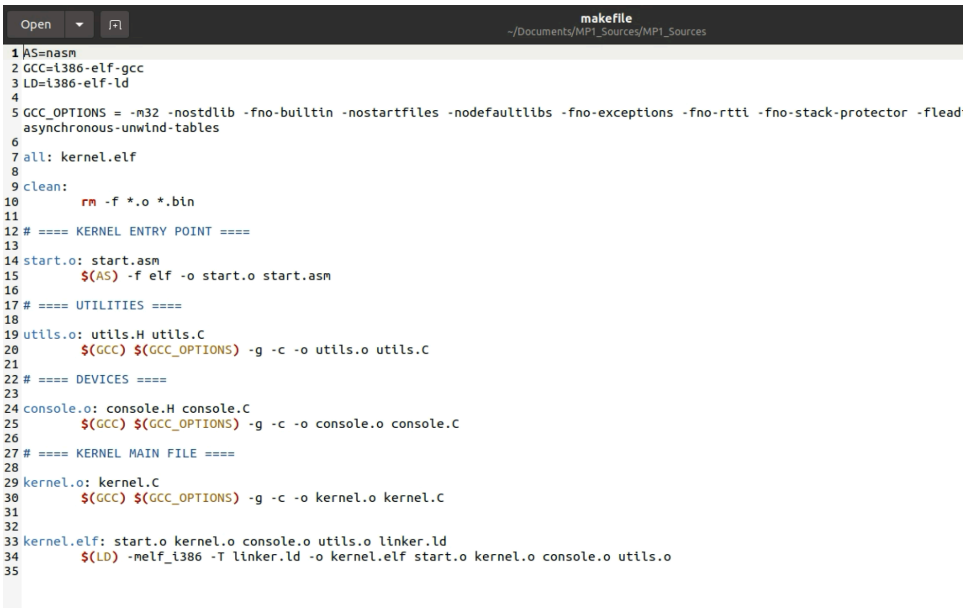
Integrating Bochs Environment with GDB: CSCE 611: MP1

To use GDB with Bochs, we need to build Bochs with gdb-stub enabled.

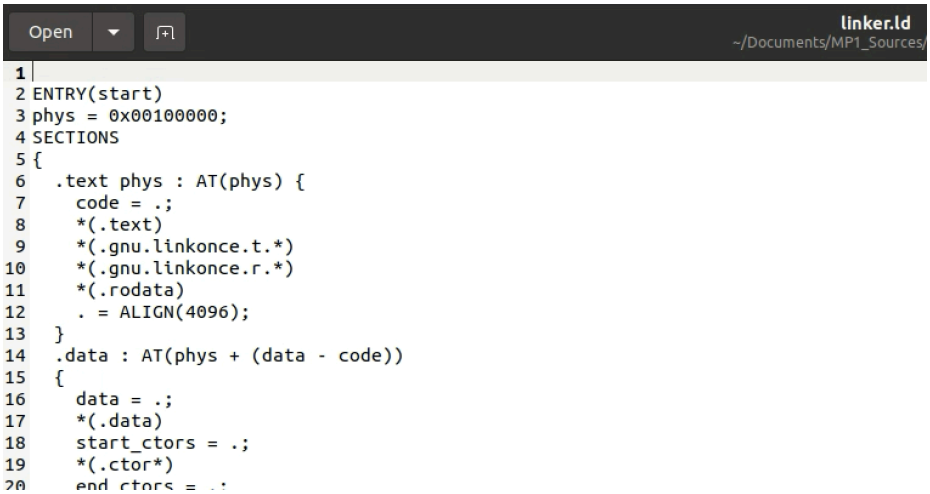
1. \Download Bochs with gdb sourcecode from the following location: https://sourceforge. net/projects/bochs/files/bochs/2.6.8/
2. Configure Bochs with gdb stub enabled, under the directory of the Bochs source code:
    a. sudo ./configure --enable-gdb-stub
3. Install Debian package libXrandr - provides an X Window System client interface to the RandR extension to the X protocol.
    sudo apt install libxrandr-dev
4. After the configuration, to make it and move it to /usr/local/bin, run
    a. sudo make
    b. sudo make install

5. Then we produce an ELF (Executable and Linkable Format) output that keeps the debugging information.
6. We add the “-g” flag to our existing makefile at each of the compile steps to produce an object file containing the debug information.




```
AS=nasm
GCC=i386-elf-gcc
LD=i386-elf-ld

GCC_OPTIONS = -m32 -nostdlib -fno-builtin -nostartfiles -nodefaultlibs -fno-exceptions -fno-rtti -fno-stack-protector -fleadasynchronous-unwind-tables

all: kernel.elf

clean:
	rm -f *.o *.bin

# ==== KERNEL ENTRY POINT ====

start.o: start.asm
	$(AS) -f elf -o start.o start.asm

# ==== UTILITIES ====

utils.o: utils.H utils.C
	$(GCC) $(GCC_OPTIONS) -g -c -o utils.o utils.C

# ==== DEVICES ====

console.o: console.H console.C
	$(GCC) $(GCC_OPTIONS) -g -c -o console.o console.C

# ==== KERNEL MAIN FILE ====

kernel.o: kernel.C
	$(GCC) $(GCC_OPTIONS) -g -c -o kernel.o kernel.C


kernel.elf: start.o kernel.o console.o utils.o linker.ld
	$(LD) -melf_i386 -T linker.ld -o kernel.elf start.o kernel.o console.o utils.o

```

7. In the linker.ld file, we remove the first line to get the ELF output



8. Now we look for any references to kernel.bin and change the name of kernel.bin file in makefile and in script file copykernel to kernel.elf.

9. gdb is used to debug the Bochs Environment, where the Bochs emulator. For this to work, we need to enable the gdb stub in the Bochs configuration file. This can be done by adding the following line in bochsrc.brxc file as shown:

gdbstub: enabled=1, port=1234, text_base=0, data_base=0, bss_base=0

10. Then we run the commands:

Sudo make
Sudo make install
./copykernel.sh

11. Loading Bochs, I run the command:

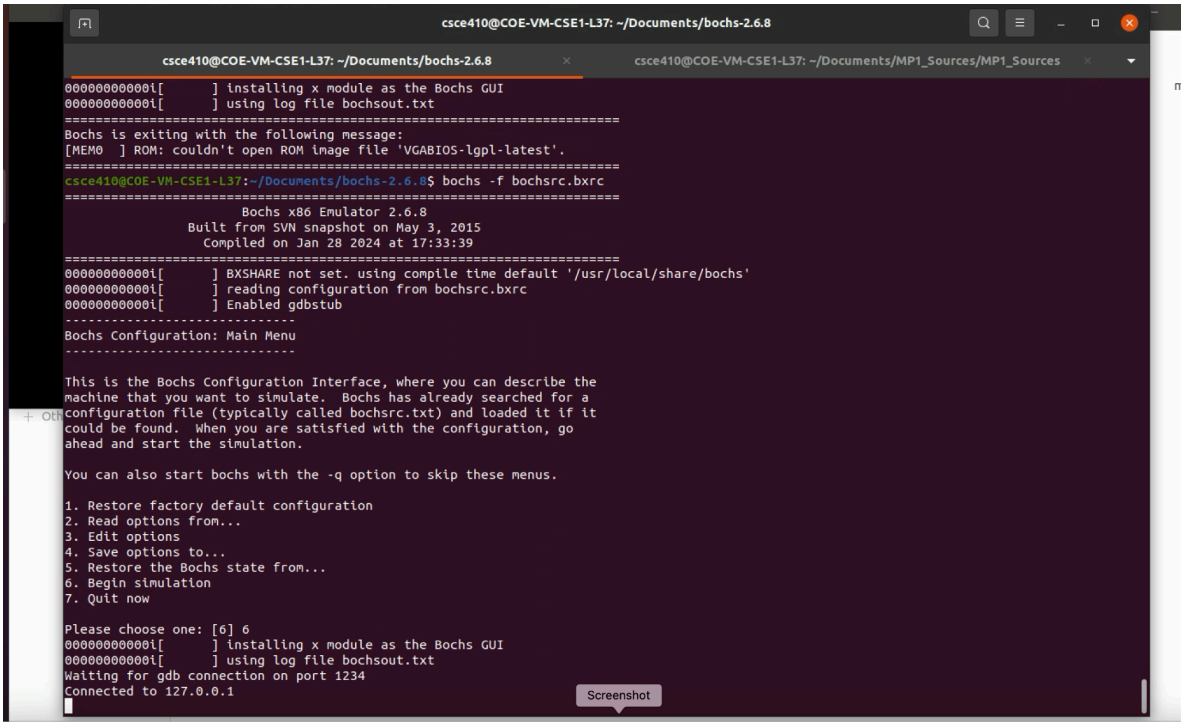
bochs –f bochsrc.bxrc

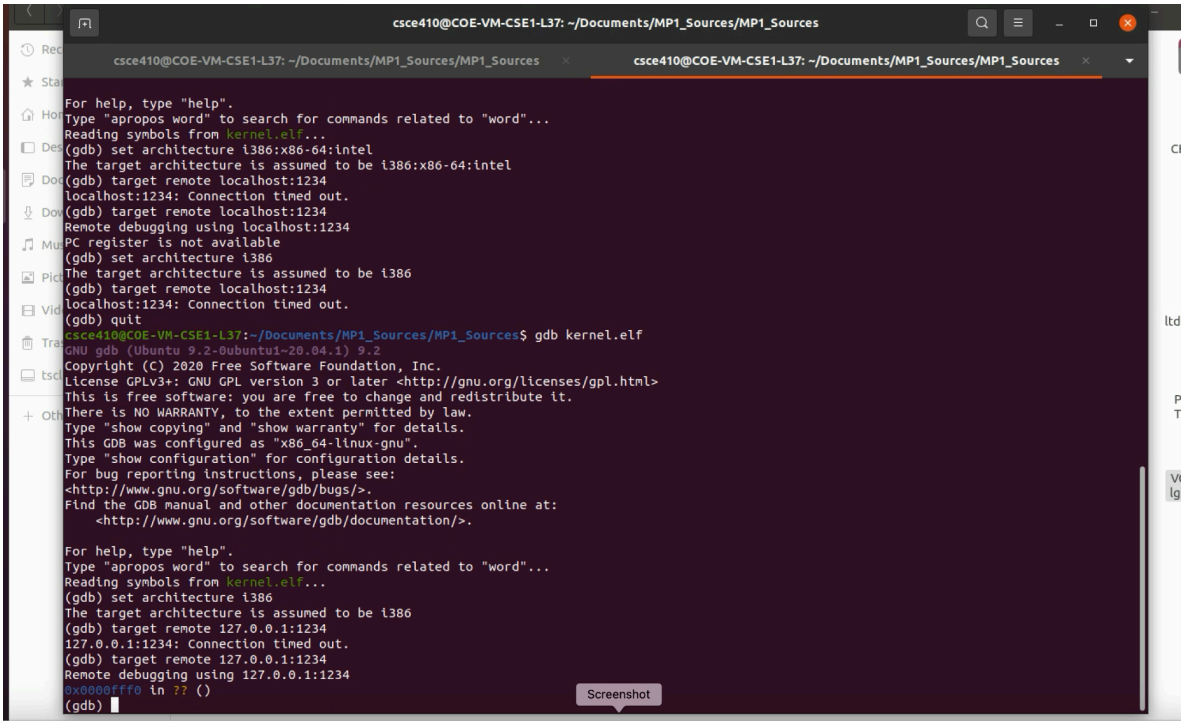
12. To connect from GDB, open a new terminal and run:

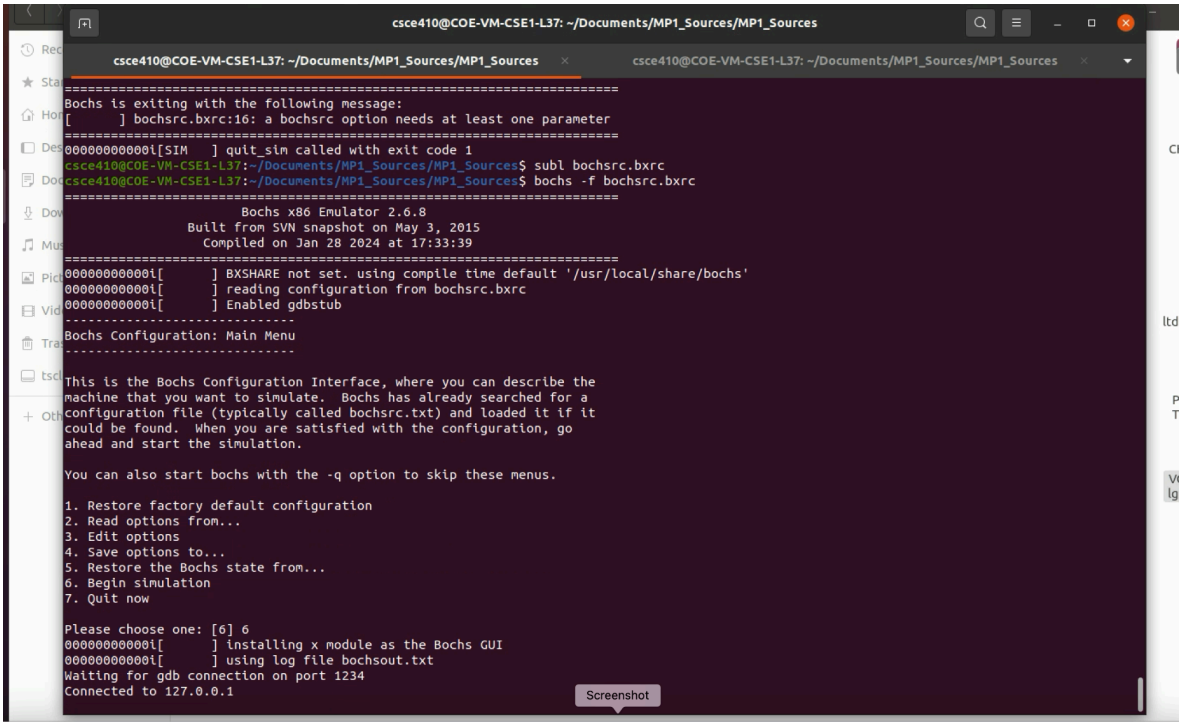
gdb YOUR-KERNEL where YOUR-KERNEL is the ELF output file.

We run the commands

gdb kernel.elf
(gdb) target set architecture i386
(gdb) target remote 127.0.0.1:1234


csce410@COE-VM-CSE1-L37: ~/Documents/bochs-2.6.8
csce410@COE-VM-CSE1-L37: ~/Documents/bochs-2.6.8
csce410@COE-VM-CSE1-L37: ~/Documents/MP1_Sources/MP1_Sources
00000000000i[ ] installing x module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt
Bochs is exiting with the following message:
[MEM0] ROM: couldn't open ROM image file 'VGABIOS-lgpl-latest'.
csce410@COE-VM-CSE1-L37:~/Documents/bochs-2.6.8$ bochs -f bochsrc.bxrc
Bochs x86 Emulator 2.6.8
Built from SVN snapshot on May 3, 2015
Compiled on Jan 28 2024 at 17:33:39
00000000000i[ ] BXSHARE not set. using compile time default '/usr/local/share/bochs'
00000000000i[ ] reading configuration from bochsrc.bxrc
00000000000i[ ] Enabled gdbstub
Bochs Configuration: Main Menu
This is the Bochs Configuration Interface, where you can describe the
machine that you want to simulate. Bochs has already searched for a
configuration file (typically called bochsrc.txt) and loaded it if it
could be found. When you are satisfied with the configuration, go
ahead and start the simulation.
You can also start bochs with the -q option to skip these menus.
1. Restore factory default configuration
2. Read options from...
3. Edit options
4. Save options to...
5. Restore the Bochs state from...
6. Begin simulation
7. Quit now
Please choose one: [6] 6
00000000000i[ ] installing x module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt
Waiting for gdb connection on port 1234
Connected to 127.0.0.1
Screenshot

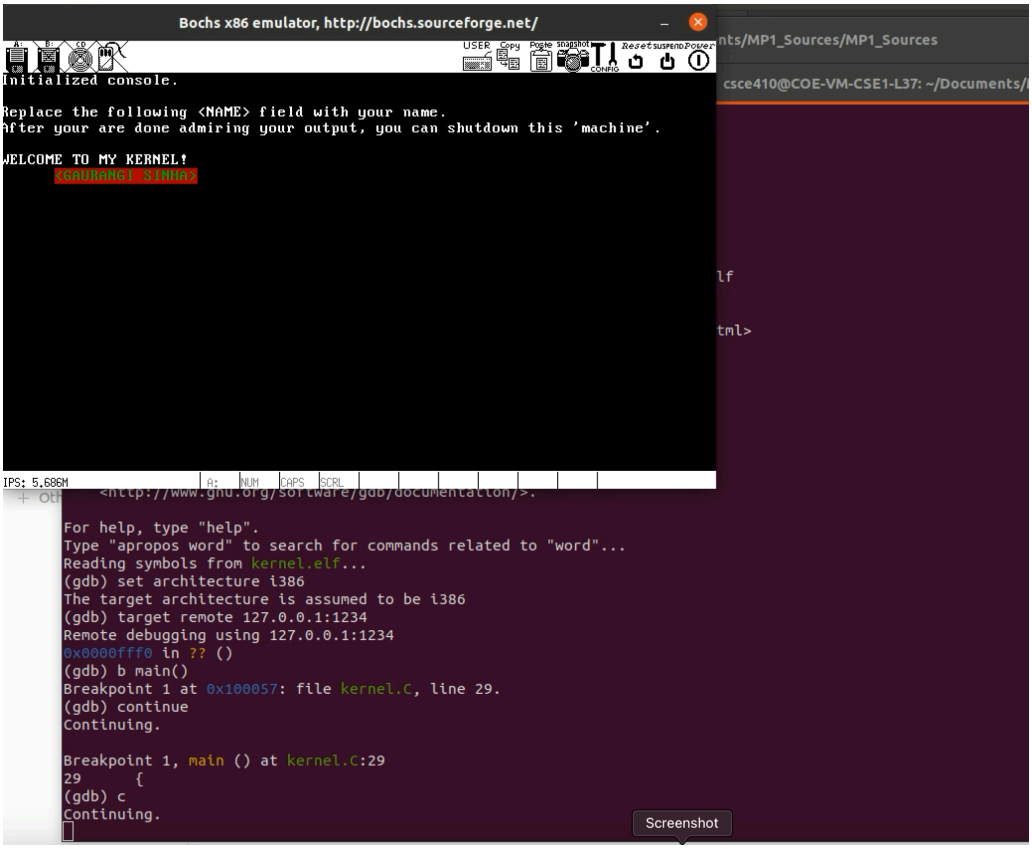

csce410@COE-VM-CSE1-L37: ~/Documents/MP1_Sources/MP1_Sources
csce410@COE-VM-CSE1-L37: ~/Documents/MP1_Sources/MP1_Sources
csce410@COE-VM-CSE1-L37: ~/Documents/MP1_Sources/MP1_Sources
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel.elf...
(gdb) set architecture i386:x86-64:intel
The target architecture is assumed to be i386:x86-64:intel
(gdb) target remote localhost:1234
localhost:1234: Connection timed out.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
PC register is not available
(gdb) set architecture i386
The target architecture is assumed to be i386
(gdb) target remote localhost:1234
localhost:1234: Connection timed out.
(gdb) quit
csce410@COE-VM-CSE1-L37:~/Documents/MP1_Sources/MP1_Sources$ gdb kernel.elf
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 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 kernel.elf...
(gdb) set architecture i386
The target architecture is assumed to be i386
(gdb) target remote 127.0.0.1:1234
127.0.0.1:1234: Connection timed out.
(gdb) target remote 127.0.0.1:1234
Remote debugging using 127.0.0.1:1234
0x0000fff0 in ?? ()
(gdb)
Screenshot



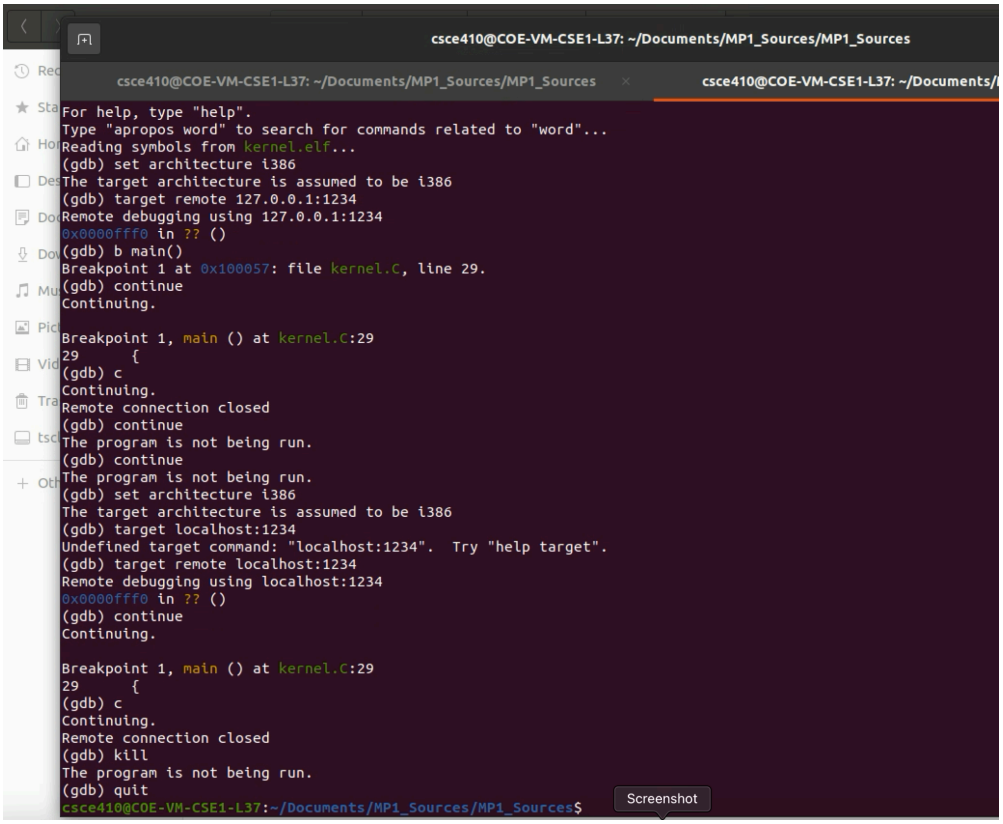
Finally running commands

(gdb) b main()

(gdb) continue



You can end the debugging session by running the kill (k) command to terminate the debugging process and exit GDB with the quit (q) command.


csce410@COE-VM-CSE1-L37: ~/Documents/MP1_Sources/MP1_Sources
csce410@COE-VM-CSE1-L37: ~/Documents/MP1_Sources/MP1_Sources
csce410@COE-VM-CSE1-L37: ~/Documents/
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel.elf...
(gdb) set architecture i386
The target architecture is assumed to be i386
(gdb) target remote 127.0.0.1:1234
Remote debugging using 127.0.0.1:1234
0x0000fff0 in ?? ()
(gdb) b main()
Breakpoint 1 at 0x100057: file kernel.C, line 29.
(gdb) continue
Continuing.
Breakpoint 1, main () at kernel.C:29
29 {
(gdb) c
Continuing.
Remote connection closed
(gdb) continue
The program is not being run.
(gdb) continue
The program is not being run.
(gdb) set architecture i386
The target architecture is assumed to be i386
(gdb) target localhost:1234
Undefined target command: "localhost:1234". Try "help target".
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x0000fff0 in ?? ()
(gdb) continue
Continuing.
Breakpoint 1, main () at kernel.C:29
29 {
(gdb) c
Continuing.
Remote connection closed
(gdb) kill
The program is not being run.
(gdb) quit
csce410@COE-VM-CSE1-L37:~/Documents/MP1_Sources/MP1_Sources$
Screenshot