MP1: Integrating Bochs with GDB

Jicheng Lu
525004048

Step 1: Download Bochs with gdb sourcecode from the following location:
https://sourceforge. net/projects/bochs/files/bochs/2.6.8/

Step 2: Install the minimum needed packages with:
sudo apt-get install libxcursor-dev libxrandr-dev libxinerama-dev libxi-dev

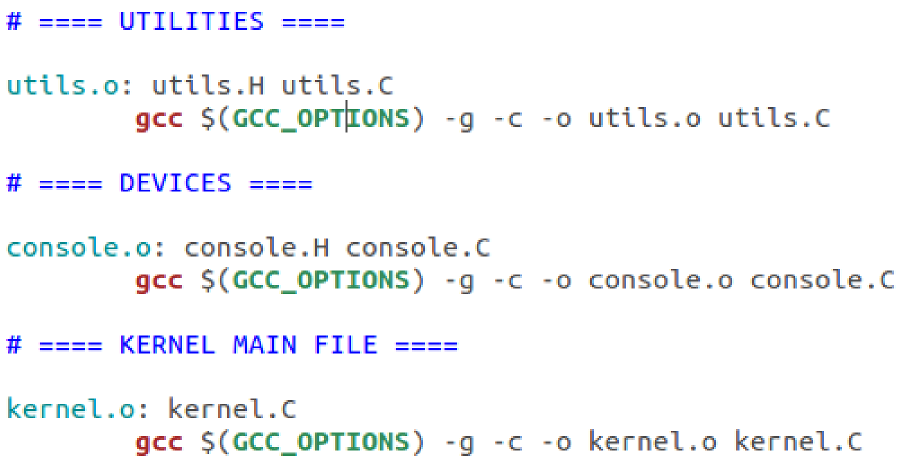
Step 3: Configure Bochs with gdb stub enabled:
sudo ./configure --enable-gdb-stub

Step 4: After the configuration, to make it and move it to /usr/local/bin, run
sudo make

Step 5: Add the following line in bochsrc.brxc file as shown:
gdbstub: enabled=1, port=1234, text_base=0, data_base=0, bss_base=0

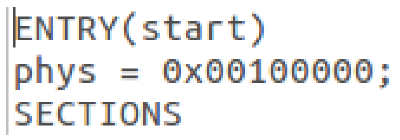
Step 6: Installing GDB debugger:
sudo apt-get install gdb

Step 7: Add "-g" flag to the existing makefile:



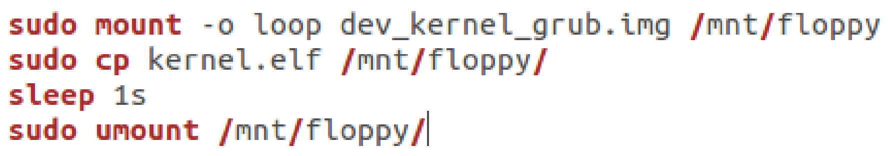
```
# ==== UTILITIES ====

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
```

Step 8: Remove the first line of the linker.ld file:



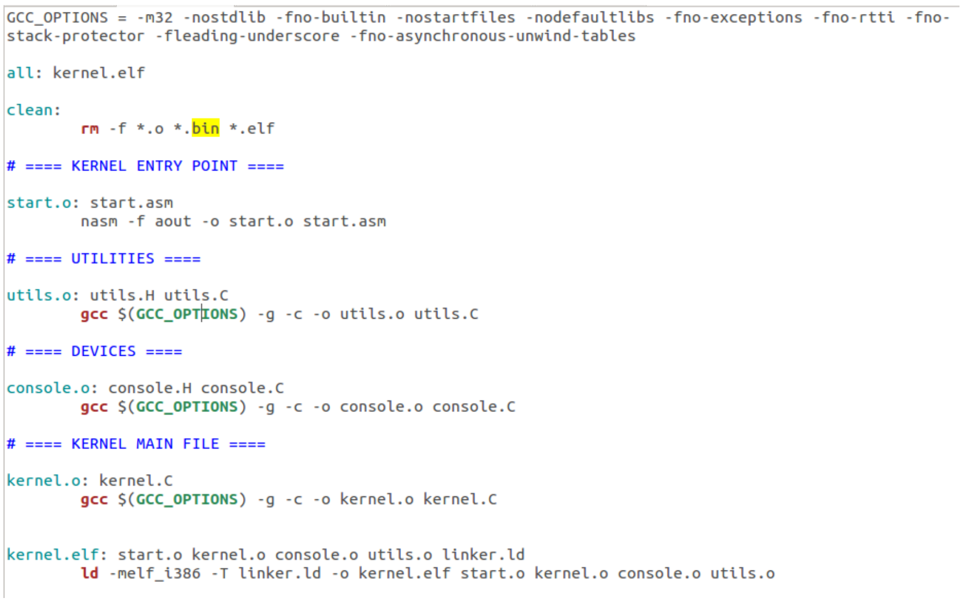
```
ENTRY(start)
phys = 0x00100000;
SECTIONS
```

Step 9: Rename our output file from “kernel.bin” to “kernel.elf” in makefile and copykernel files:



```
sudo mount -o loop dev_kernel_grub.img /mnt/floppy
sudo cp kernel.elf /mnt/floppy/
sleep 1s
sudo umount /mnt/floppy/
```

Copykernel file



```
GCC_OPTIONS = -m32 -nostdlib -fno-builtin -nostartfiles -nodefaultlibs -fno-exceptions -fno-rtti -fno-
stack-protector -fleading-underscore -fno-asynchronous-unwind-tables

all: kernel.elf

clean:
        rm -f *.o *.bin *.elf

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

start.o: start.asm
        nasm -f aout -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
```

Make file

Step 10: Compile and copy:

$ make

$ sh copykernel.sh

Step 11: Load Bochs:

$ bochs -f bochsrc.bxrc

Step 12: Open a new terminal and run:

gdb kernel.elf

Step 13: Set architecture:

(gdb) set architecture i386:x86-64:intel

Step 14: Connect to Bochs target:

(gdb) target remote localhost:1234

Step 15: Set breakpoint:

(gdb) b main()

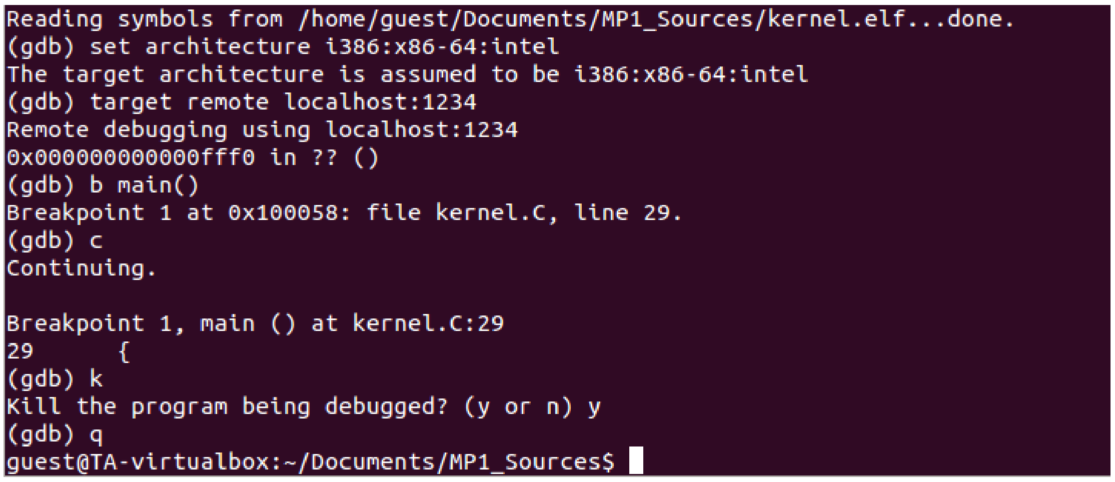
Step 16: Continue the Bochs simulation:

(gdb) continue

Step 17: Kill the debugging process and exit gdb:

(gdb) kill

(gdb) quit



```
Reading symbols from /home/guest/Documents/MP1_Sources/kernel.elf...done.
(gdb) set architecture i386:x86-64:intel
The target architecture is assumed to be i386:x86-64:intel
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x000000000000fff0 in ?? ()
(gdb) b main()
Breakpoint 1 at 0x100058: file kernel.C, line 29.
(gdb) c
Continuing.

Breakpoint 1, main () at kernel.C:29
29	  {
(gdb) k
Kill the program being debugged? (y or n) y
(gdb) q
guest@TA-virtualbox:~/Documents/MP1_Sources$ 
```