**MP1**

**CSCE611: Operating System Fall 2022**

**Steps to integrate Bochs environment with GDB.**

**Step1**: Download the Bochs package from the below link. We have to download the Bochs 2.6.8 version tar file and unzip it.

<https://sourceforge.net/projects/bochs/files/bochs/2.6.8/bochs-2.6.8.tar.gz/download>

We have now the source code. Note that if you are working on mac, the .files are hidden. So we can look for them using ls -al command on the terminal and we will find all the files in the source code.

**Step2**: Remove the existing Bochs in the Ubuntu by

*sudo apt-get remove bochs*

**Step 3**: Under the directory of bochs source code, configure Bochs with gdb stub enabled.

*sudo ./configure --enable-gdb-stub*

**Step 4**: After configuration, to compile bochs ,run

*sudo make*

and to move it to /usr/local/bin, run

*sudo make install*

**Note:**

* While compile, (after sudo make), if error related to plugin occurs, uncomment ‘which\_config = normal’ and comment out ‘which\_config = plugins’ in the file .conf.linux

we can find this file in the bochs source code.

* Also we might get some errors related to X11. For that , install libraries:

*sudo apt-get install xorg-dev*

Don’t forget to reconfigure ( sudo ./configure --enable-gdb-stub) before doing *sudo make* again.

We have now installed the bochs environment successfully. Now we have to integrate it with GDB.

**Step: 5** Install GDB debugger. It might be installed already.

*sudo apt-get install gdb*

**Step 6**: Add the following line in the bochsrc.brcx file (in the MP source code):

*gdbstub: enabled=1, port=1234, text\_base=0, data\_base=0, bss\_base=0*

**Step 7**: To enable debugging symbols, edit file makefile in MP source code by adding -g in all the GCC options(at 3 positions).

For example: *gcc $(GCC\_Optios) -g -c -o utils.o utils.C*

**Step 8:** Edit the linker.ld file in the MP source code. Remove the first line of the linker file:

OUTPUT\_FORMAT(“binary)”.

This is done because a flat binary file is usually stripped off its debugging informstion and thus only contains its data part. This makes it impossible to debug from GDB.

Note: Don’t change the name of the kernel.bin file. Leave it as it is.

**Step 9**: Recompile the project and copy it(usual steps).

We are ready to run the project with debugger:

**Step 10**: We open the bochs tool and wait for gdb connection on port.

*bochs -f bochsrc.bxrc*

it will wait for the gdb connection on port 1234.

**Step 11:** Open a new terminal and run.

*gdb kernel.bin*

We need to connect to the bochs target . For that use command:

*(gdb) target remote localhost:1234*

**Step 12:** Set the breakpoint.

(gdb) *b main*

after that continue:

*(gdb) c*

Kernel output will popup. Hit n in the debugger terminal for next line till your reach your desired location.

*(gdb) n*

We can end the debugging session by :

*(gdb) kill*

And exit the GDB by:

*(gdb) quit*

*Screenshot of GDB tool in action:*

Text

Description automatically generated