EECS 678, W 4:00
Rachel J. Morris

1. Explain the difference between a hardware breakpoint and a software breakpoint. [5 Points]

Hardware breakpoints allow you to pause program execution when a memory address is being accessed. A software breakpoint allows you to pause program execution at a specific line of code.

2. Explain the difference between a breakpoint and a watchpoint. [5 Points]

A breakpoint pauses execution when the program flow hits a specific line of the code. A watchpoint pauses execution when the value of a variable changes.

3. Is it possible to change the value of a variable during debug session? If so, mention the gdb command for doing it. [5 Points]

Yes, you can use the set var command.

4. Consider the following piece of code [5 Points]:

```
#define STRING_LENGTH 256 /* Defined in some obscure header file and included in
the source file */

…

int main (void)
{
        char    my_string[STRING_LENGTH] = “This world is a beautiful place!”;

        print (“%s\n”, my_string);

        return 0;
}
```

While debugging the program, you place a breakpoint inside the main function. Once the breakpoint is hit, Your job is to find out the length of the variable “my_string” (i.e. 256) while debugging. How can you do that? Remember that the macro will get pre-processed and you won’t be able to use:
print STRING_LENGTH

You can get the array’s length (in bytes) with sizeof(), and then divide by the size of a char in order to get the amount of elements.

5. Consider the following loop [10 Points]:

```
for (i = 0; i < 1000000; ++i) {
        ...
        read_page (i);
        ...
}
```

During the execution of this loop, you get a segmentation fault for some value of 'i' let's say i = 10000. You want to see what exactly happens inside the loop in that iteration. One way to do it is to place a breakpoint in the loop body, manually wait for it to get hit 9999 times and then begin your investigation in the 10000th iteration. This is certainly undesirable and there should be a better way to do it. Can you figure out an efficient method to resolve this problem?

We can set a watch point for when *i=10000*.

6. Consider the follwing code segment [10 Points]:

```
void scheduler ( … )
{
        /* Get the priority of the highest priority ready task */
        int priority = get_high_priority ( … );

        /* Fetch the actual task */
        TASK* task = fetch_task (priority);

        /* Schedule the task */
        run (task);
}
```

You want to store the priority value when a certain task-x is selected for running inside the scheduler. Can you figure out a way to do that using gdb? (hint: You may want to use 'conditional breakpoint' and 'convenience variables' to solve this problem)

I think that I will have to spend more time with the gdb docs, because I'm not familiar with its functionality at all.