Inlab 2: Using the Debugger

In this in-lab we will:

- Review some shell related issues from last time
- Make a change to your prompt so that you can more easily "see" where you are
- Introduce the debugger we will be using.

Review:

Last time around we introduced a number of different commands. Talking to a lot of you, there was a fair bit of confusion about what the various commands *did*. Some of the GSIs had a handout from last semester which is very useful for getting around Linux/Unix. It may be helpful to took at it (if your GSI didn't already give it to you) http://www.eecs.umich.edu/courses/engin101/Inlab/linux_intro.pdf.

Also, another GSI posted the following list of commands that you might find useful:

'pwd' = print the path to your current working directory
'ls' = list the contents of your current directory
'cd ABC' = change into directory ABC
'cd' = change into your home directory
'cd ~' = another way to change to your home directory
'cd ..' = change into the parent of your current directory
'cd -' = returns to the directory you were in previously
'mkdir ABC' = make a new directory named ABC
'rm abc' = delete the file named ABC.
'rmdir ABC' = delete the directory named ABC

Now, pull up nedit (or your favorite editor) and put your name (first and last) and uname at the top. Skip and few lines and start in on the questions:

1. Look at the man page for lpr. What does "lpr –Pbob foo.txt" do?
2. What does "pwd" do? What is pwd an abbreviation for?
3. If you wanted to compile a program named "bob.cc" into an executable named "coolprogram" what would be the command you would use?
4. tcsh supports "tab completion." What is that?

More fun:

In your home directory ("cd" will get you there) there *should* be a file called .cshrc. Because it starts with a period it is a *hidden* file in unix. "ls –a" should let you see it. We want you to copy it into a file named '.tcshrc'. So "cp .cshrc .tcshrc" is the command you want. Now use nedit to edit the .tcshrc file. Look in the file to see if your prompt has been set to something already. (The like should look something like set prompt="XXXXX" where XXXXX could be just about anything.) If it has been set, comment out that line (put a "#" as the first character of that line). Either way, then add the following line:

```
set prompt="<\! `uname -n`: %c3>"
```

The spaces (and lack there of) are very important. Once you have done that, type "tcsh" (to start shell we used last time). Your prompt should now look something like this:

```
<8 bree: ~>
```

where "bree" will be replaced by your computer name. The ~ indicates that you are currently in your home directory. If you cd to a directory named 101 your prompt will become something like

```
<11 bree: ~/101>
```

The first number (8 in the first example, 11 in the one above) is the command number. One neat thing you can do is see all of the things you have done so far by number. Type "history" and you will see something like this:

```
   7  15:08   cd
   8  15:25   cd 101
   9  15:25   mkdir 101
  10  15:25   cd 101
  11  15:26   history
```

This tells me the last few commands I've typed and when I typed them (military time). You can "redo" a command by typing "!X" where X is the command number. So in the above case "!9" would be the same as "mkdir 101". You *should* also be able to see (and edit) your old commands by using the up and down arrows (and left and right)

5. Copy your prompt as the answer to this question.
6. Type "history" and paste the last two lines as your answer to this question.

Now, take a look at http://www.eecs.umich.edu/courses/engin101/Inlab/debugging.pdf Read that document. After you've done that, download the program lab2.cc from the course page (under in-lab2). Compile the program "g++ -o inlab2 lab2.cc" at this time you should get a compiler error (as described in the debugging handout above).

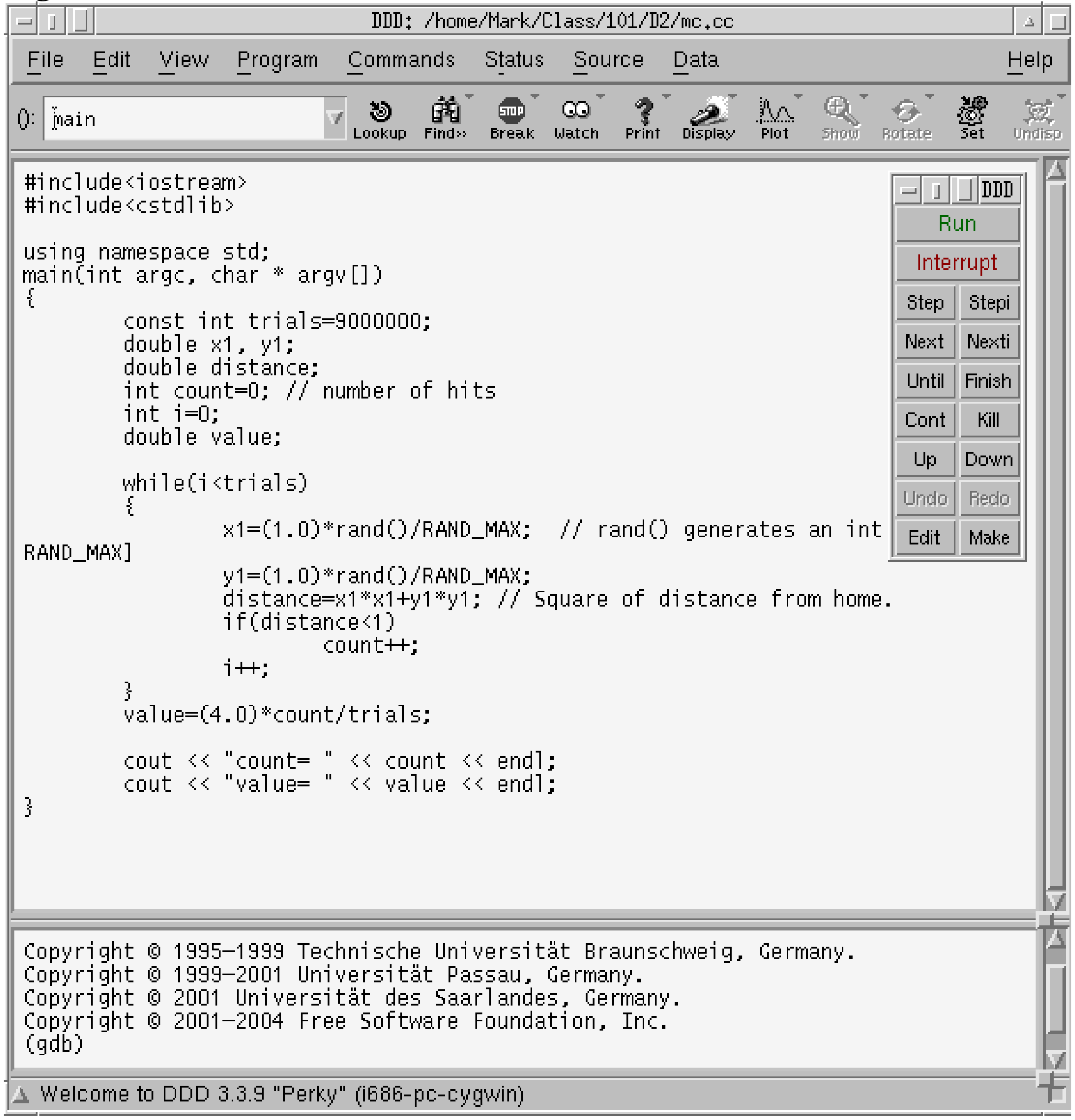
7. What line(s) does the complier claim the error is on?
8. Look at the code and fix the error. What was the error?
9. Why do you think it gave you more than one line with an error?

Now, using the debugging document as a guidepost, recompile the code (remember the –g flag!) and pull it up in the debugger. It should look something like figure 1 (next page) Now set a breakpoint where "distance" gets assigned a value (line 18 if you haven't changed much) and run the program. A "stop sign" should appear. In figure 2 the stop sign is shown, but at a different line. Using the debugger display the values of x1, y1, and distance. You may need to drag some values around so you can see them all.

10. What are the values of x1, y1 and distance?
11. Now press the "step" button once. Did any of those values change? If so, which ones?
12. Add "i" to the list of variables you are watching. What is its value?
13. Now press "cont" (continue). Which values changed if any? What did pressing "cont" do?
14. Now clear the breakpoint. Press "cont" again. What happened?
15. Put the breakpoint back where it was. Hit run and press "cont" 5 times. What is the value of "count" at that point?

That's it folks. Good luck!

Figure 1

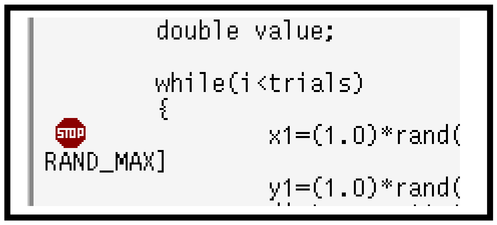

DDD: /home/Mark/Class/101/D2/mc.cc
File Edit View Program Commands Status Source Data Help
(): main
Lookup Find» Break Watch Print Display Plot Show Rotate Set Undisp
DDD
Run
Interrupt
Step Stepi
Next Nexti
Until Finish
Cont Kill
Up Down
Undo Redo
Edit Make

```
#include<iostream>
#include<cstdlib>

using namespace std;
main(int argc, char * argv[])
{
        const int trials=9000000;
        double x1, y1;
        double distance;
        int count=0; // number of hits
        int i=0;
        double value;

        while(i<trials)
        {
                x1=(1.0)*rand()/RAND_MAX;  // rand() generates an int
RAND_MAX]
                y1=(1.0)*rand()/RAND_MAX;
                distance=x1*x1+y1*y1; // Square of distance from home.
                if(distance<1)
                        count++;
                i++;
        }
        value=(4.0)*count/trials;

        cout << "count= " << count << endl;
        cout << "value= " << value << endl;
}
```

```
Copyright © 1995-1999 Technische Universität Braunschweig, Germany.
Copyright © 1999-2001 Universität Passau, Germany.
Copyright © 2001 Universität des Saarlandes, Germany.
Copyright © 2001-2004 Free Software Foundation, Inc.
(gdb)
```

Welcome to DDD 3.3.9 "Perky" (i686-pc-cygwin)

Figure 2



```
        double value;

        while(i<trials)
        {
                x1=(1.0)*rand(
RAND_MAX]
                y1=(1.0)*rand(
```