

COP4530 – Data Structures, Algorithms and Generic Programming
Recitation 2
Date: 9th September, 2011

Objective:

1. Using a debugger (DDD)
2. Discuss Assignment 1

Setup tasks:

- Under the Start menu, find the folder named X-Win32 and open the Xutils32.
- Start the SSH client. When the shell opens, go to *Edit Settings* and then go to *Tunneling*.
- Check the box that says *Tunnel x11 con* and click *OK*. Now it is possible to run text editors in separate windows from the SSH console. For emacs for example, just type “xterm emacs” followed by the file name to be edited, and a separate emacs window will appear.

Data Display Debugger

A debugger allows us to run other program executables and examine the behavior of these programs as they are running. Debuggers are especially helpful when there is a mysterious bug in the program that is not apparent at first glance.

One of the more popular debuggers for UNIX would be the GDB (the GNU debugger). A graphical front-end version of GDB is known as the DDD.

Areas to be covered:

1. Starting the debugger.
2. Setting and clearing breakpoint
3. Using step and next for stepwise execution
4. Checking the value of program variables
5. Checking the contents of an array

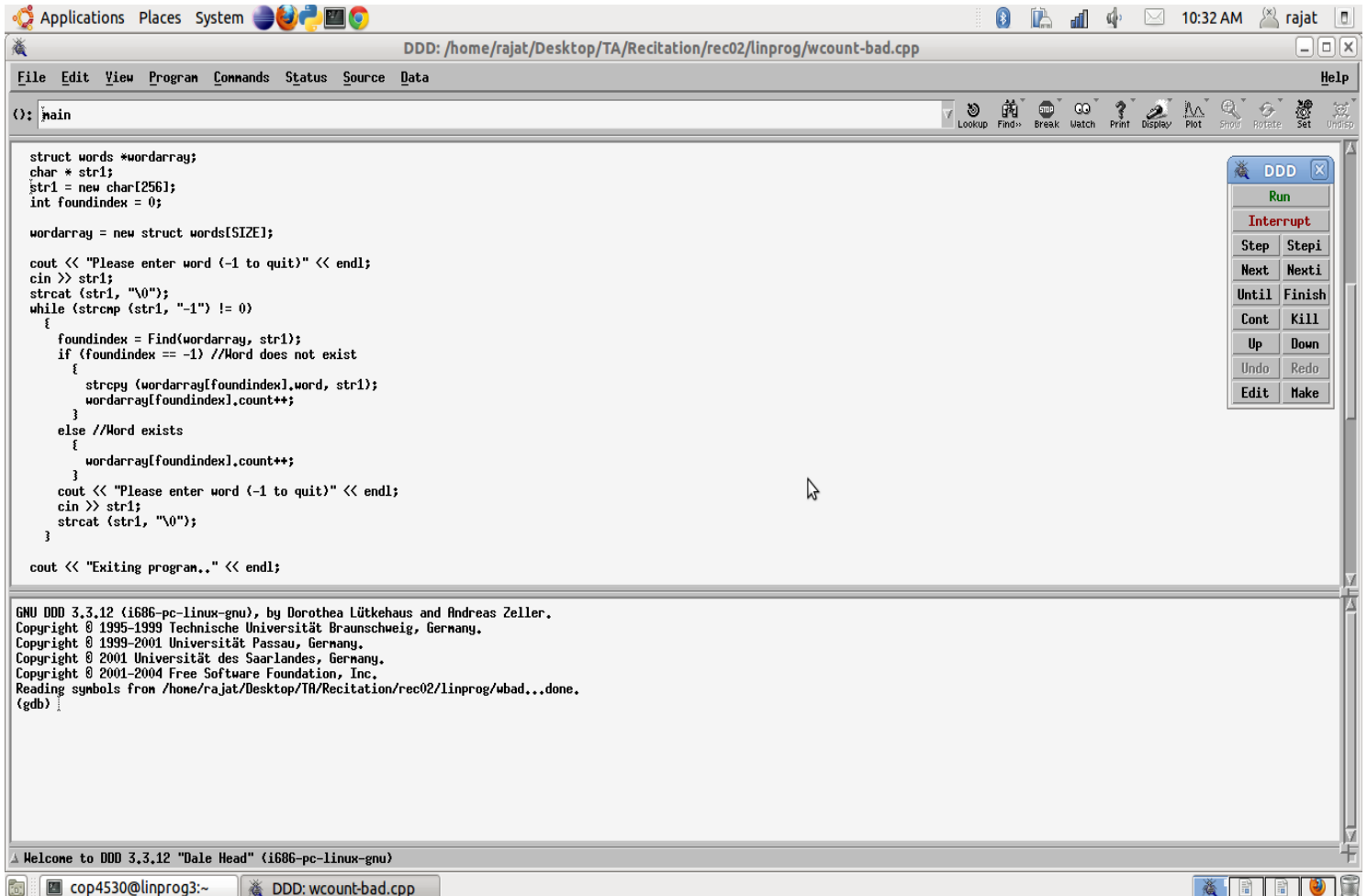
1. Starting the debugger:

In order to use the debugger, the code must be compiled with the -g flag. For example if the original file is myprogram.cpp you can use the following command to generate the executable:

`c++ -g -o myprogram myprogram.cpp`

Now if you have DDD installed then you can start the debugger with the following command:
ddd myprogram

Here is a screenshot of the window:



2. Setting and clearing breakpoints:

A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point. When execution is suspended at a breakpoint, your program is said to be in break mode. Entering break mode does not stop or end the execution of your program; execution can be resumed at any time.

To set the breakpoint take the mouse pointer to the beginning of the line before which you want to set a breakpoint. Then do a rightclick which will provide you with the menu to set a breakpoint. To clear the breakpoint you can do a right click on that breakpoint and use the menu.

Here are the screenshots for setting and clearing a breakpoint:

Applications Places System 10:42 AM rajat

DDD: /home/rajat/Desktop/TA/Recitation/rec02/linprog/wcount-bad.cpp

File Edit View Program Commands Status Source Data Help

(~): wcount-bad.cpp:34

```

struct words *wordarray;
char * str1;
str1 = new char[256];
int foundindex = 0;

wordarray = new struct words[SIZE];

cout << "Please enter word (-1 to quit)" << endl;
Set Breakpoint
Set Temporary Breakpoint ) != 0)
Continue Until Here darray, str1);
Set Execution Position //Word does not exist
    strcpy (wordarray[foundindex].word, str1);
    wordarray[foundindex].count++;
}
else //Word exists
{
    wordarray[foundindex].count++;
}
cout << "Please enter word (-1 to quit)" << endl;
cin >> str1;
strcat (str1, "\0");
}

cout << "Exiting program.." << endl;

```

GNU DDD 3.3.12 (i686-pc-linux-gnu), by Dorothea Lütkehaus and Andreas Zeller.
 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.
 Reading symbols from /home/rajat/Desktop/TA/Recitation/rec02/linprog/wbad...done.
 (gdb) break wcount-bad.cpp:34
 Breakpoint 1 at 0x80487b7: file wcount-bad.cpp, line 34.
 (gdb) delete 1
 (gdb)

Welcome to DDD 3.3.12 "Dale Head" (i686-pc-linux-gnu)

cop4530@linprog3:~ DDD: wcount-bad.cpp

Applications Places System 10:44 AM rajat

DDD: /home/rajat/Desktop/TA/Recitation/rec02/linprog/wcount-bad.cpp

File Edit View Program Commands Status Source Data Help

(~): wcount-bad.cpp:38

```

struct words *wordarray;
char * str1;
str1 = new char[256];
int foundindex = 0;

wordarray = new struct words[SIZE];

cout << "Please enter word (-1 to quit)" << endl;
cin >> str1;
strcat (str1, "\0");
while (strcmp (str1, "-1") != 0)
Properties...
Disable Breakpoint darray, str1);
Delete Breakpoint //Word does not exist
Set Execution Position y[foundindex].word, str1);
    else //Word exists
    {
        wordarray[foundindex].count++;
    }
    cout << "Please enter word (-1 to quit)" << endl;
    cin >> str1;
    strcat (str1, "\0");
}

cout << "Exiting program.." << endl;

```

GNU DDD 3.3.12 (i686-pc-linux-gnu), by Dorothea Lütkehaus and Andreas Zeller.
 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.
 Reading symbols from /home/rajat/Desktop/TA/Recitation/rec02/linprog/wbad...done.
 (gdb) break wcount-bad.cpp:34
 Breakpoint 1 at 0x80487b7: file wcount-bad.cpp, line 34.
 (gdb) delete 1
 (gdb) break wcount-bad.cpp:37
 Breakpoint 2 at 0x80487ef: file wcount-bad.cpp, line 37.
 (gdb) [

Welcome to DDD 3.3.12 "Dale Head" (i686-pc-linux-gnu)

cop4530@linprog3:~ DDD: wcount-bad.cpp

3. Using step and next for stepwise execution

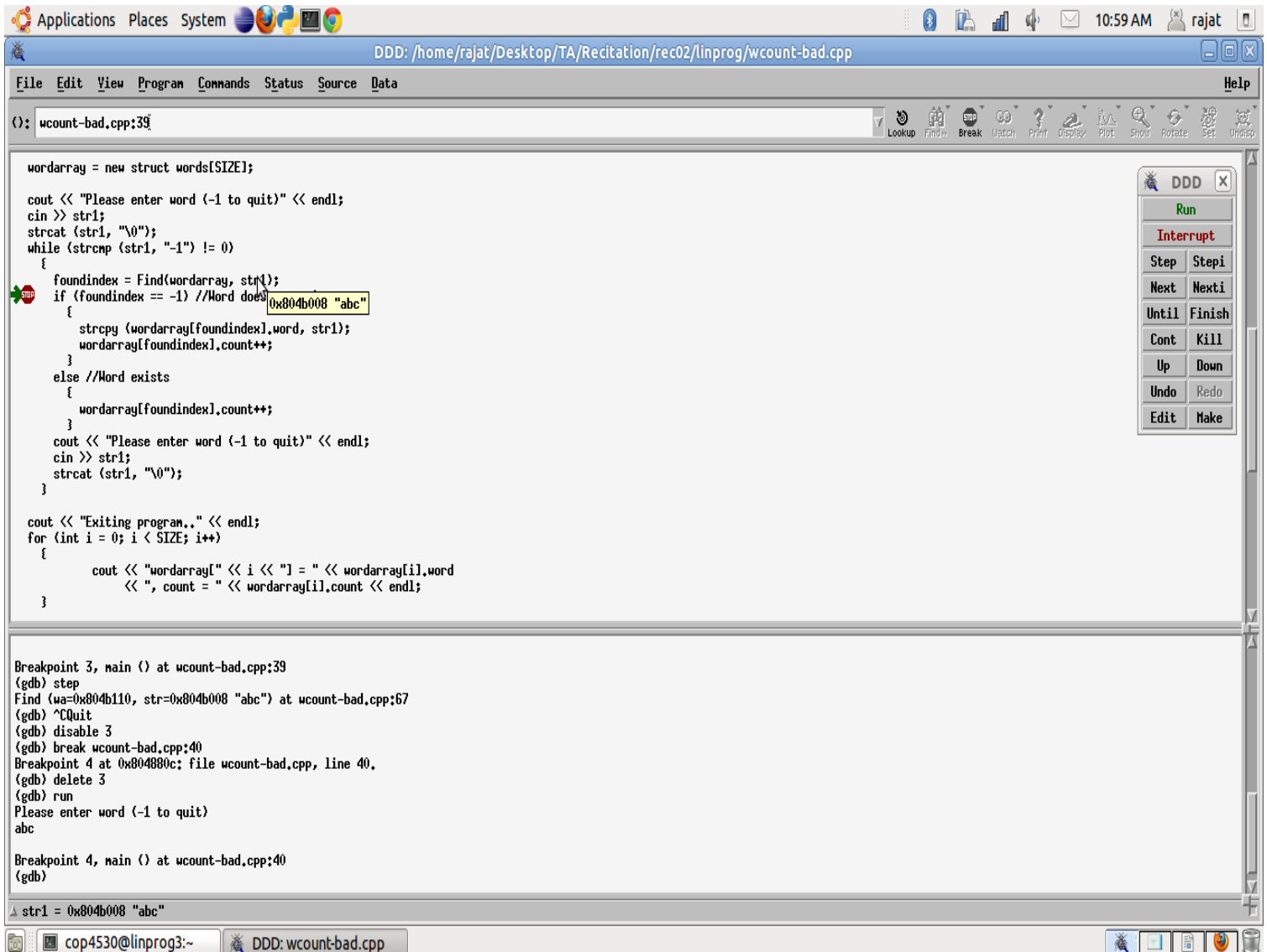
To resume execution of the program step or next can be used. Both this buttons appear on the GUI panel.

- Step executes the current line of source code entering called function if any.
- Next does the same but does not enter the called function.
- Also the 'Cont' button means continue execution without single stepping.

4. Checking the value of program variables

The main purpose of suspend, resume and stepwise execution is to check the value or state of the variables which are used to implement the program logic.

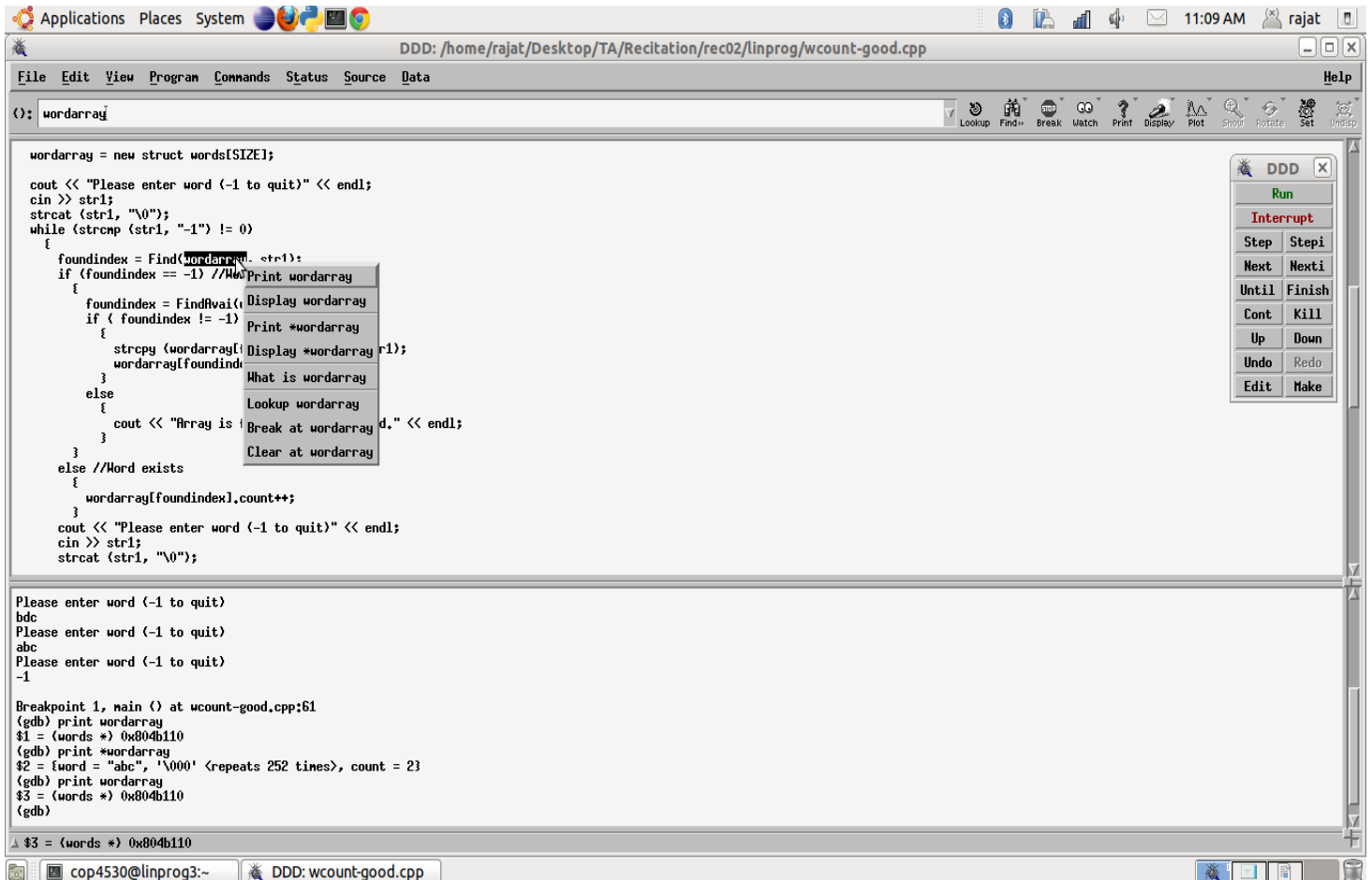
While on a breakpoint we can check the value of a single variable by hovering the mouse over it. Its value will pop up on the screen. Please see the screenshot below:



5. Checking the contents of an array:

To check the contents of an array select the array (in the source file) and do a right click to get the menu to print the value of the contents.

Note: Check the difference between “print array” and “print *array”



```
wordarray = new struct words[SIZE];
cout << "Please enter word (-1 to quit)" << endl;
cin >> str1;
strcat (str1, "\0");
while (strcmp (str1, "-1") != 0)
{
    foundindex = Find(wordarray, str1);
    if (foundindex == -1) //Word does not exist
    {
        foundindex = FindAval(wordarray, str1);
        if ( foundindex != -1)
        {
            strcpy (wordarray[foundindex], str1);
            wordarray[foundindex].count++;
        }
        else
        {
            cout << "Array is full. Word does not exist." << endl;
        }
    }
    else //Word exists
    {
        wordarray[foundindex].count++;
    }
    cout << "Please enter word (-1 to quit)" << endl;
    cin >> str1;
    strcat (str1, "\0");
}
```

```
Please enter word (-1 to quit)
bdc
Please enter word (-1 to quit)
abc
Please enter word (-1 to quit)
-1

Breakpoint 1, main () at wcount-good.cpp:61
(gdb) print wordarray
$1 = (words *) 0x804b110
(gdb) print *wordarray
$2 = {word = "abc", '\000' <repeats 252 times>, count = 23}
(gdb) print wordarray
$3 = (words *) 0x804b110
(gdb)
```

Assignment 1 discussion

Assignment 1 requirements are already discussed in the previous recitation. A few reminders:

- Due on 14th September 2011
- Must use one STL container and the corresponding iterator.
- Make sure your code has all the mentioned features.

References:

A nice introduction to DDD is here: <http://heather.cs.ucdavis.edu/~matloff/Debug/Debug.pdf>