COP4342 - Fall 2018
Assignment 12
Document Generation

Objectives: Learn how to use latex, bibtex, import, xfig, tar, and gzip. Within latex and bibtex, learn how to create tables, generate a list, include figures, and create and cite references.

Instructions: Your assignment is to write a document using latex about the five most interesting/useful items that you have learned in this class. These items might be Unix commands, Unix utilities, features of Perl, or other tools discussed in the class. Your document should be in a file called paper.tex and your references should be in a file called refs.bib. You should use the article document class for this paper. You should use 11 point font, single spaced (which is the default), and the paper should be at least three pages long. Your document should include at least one table, one list, one figure drawn with xfig, one image captured with import, and at least two cited references along with a bibliography using bibtex. This document should be written as a real paper, not just a collection of items that are forced together to meet the requirements. These items should be naturally placed in the paper rather than just forcing in an item. In addition, you should ensure that there are no misspellings in the paper. An example of a latex paper that is attached to this assignment is on the course website.

Submission: You should create a tar file called paper.tar containing the files paper.pdf, paper.tex, and refs.bib. You should compress the tar file by gzipping it. You should submit the file paper.tar.gz.

Introduction to the GNU Debugger

John E. Student
Florida State University

November 2012

Abstract

This paper goes over the basics of the GNU debugger, also known as *GDB* [1].

1 Introduction

The purpose of a source level debugger, such as GDB, is to be able to see *inside* another program while it is running and examine the state of the machine at specific moments. It is also helpful to determine why crashes may occur.

1.1 Invoking GDB

GDB can be called with the following options:

```
GDB [-help] [-nx] [-q] [-batch] [-cd=dir] [-f] [-b bps]
    [-tty=dev] [-s symfile] [-e prog] [-se prog] [-c core]
    [-x cmds] [-d dir] [prog[core|procID]]
```

2 Main Activities

GDB can do four main kinds of activities, (along with support for these main features).

1. Start your program, specifying anything that might affect its behavior.
2. Make your program stop on specified conditions.
3. Examine what has happened when your program has stopped.

4. Change the values of variables in your program so you can experiment with the effects of a bug.

3 Main Commands

Table 1 shows the main commands that *GDB* accepts. Once you start using these commands, they will quickly become familiar and you won't have to think about the syntax when using them.

Command	Arguments	Explanation
break	[file:]function	Set a breakpoint at function (in file).
run	[arglist]	Start your program (with arglist, if specified).
where		display the backtrace of program stack frames.
print	expr	Display the value of an expression.
cont		Continue running your program (after stopping, e.g. at a breakpoint).
next		Execute next program line (after stopping); step over any function calls in the line.
step		Execute next program line (after stopping); step into any function calls in the line.
help	[name]	Show information about a GDB command name, or general information about using GDB.
quit		Exit from GDB.

Table 1: Common GDB commands

4 Data Display Debugger

The Data Display Debugger is a graphical user interface that can call other debuggers (although the standard is GDB) [2]. It makes debugging so much nicer than using the gdb line oriented interface. Figure 1 shows a picture of *ddd* so you can see how nice it looks.

References

[1] R. Stallman, R. Pesch, and S. Shebs. *Debugging with GDB*. Free Software Foundation, January 2002.

[2] A. Zeller. *Debugging with DDD*, January 2004.

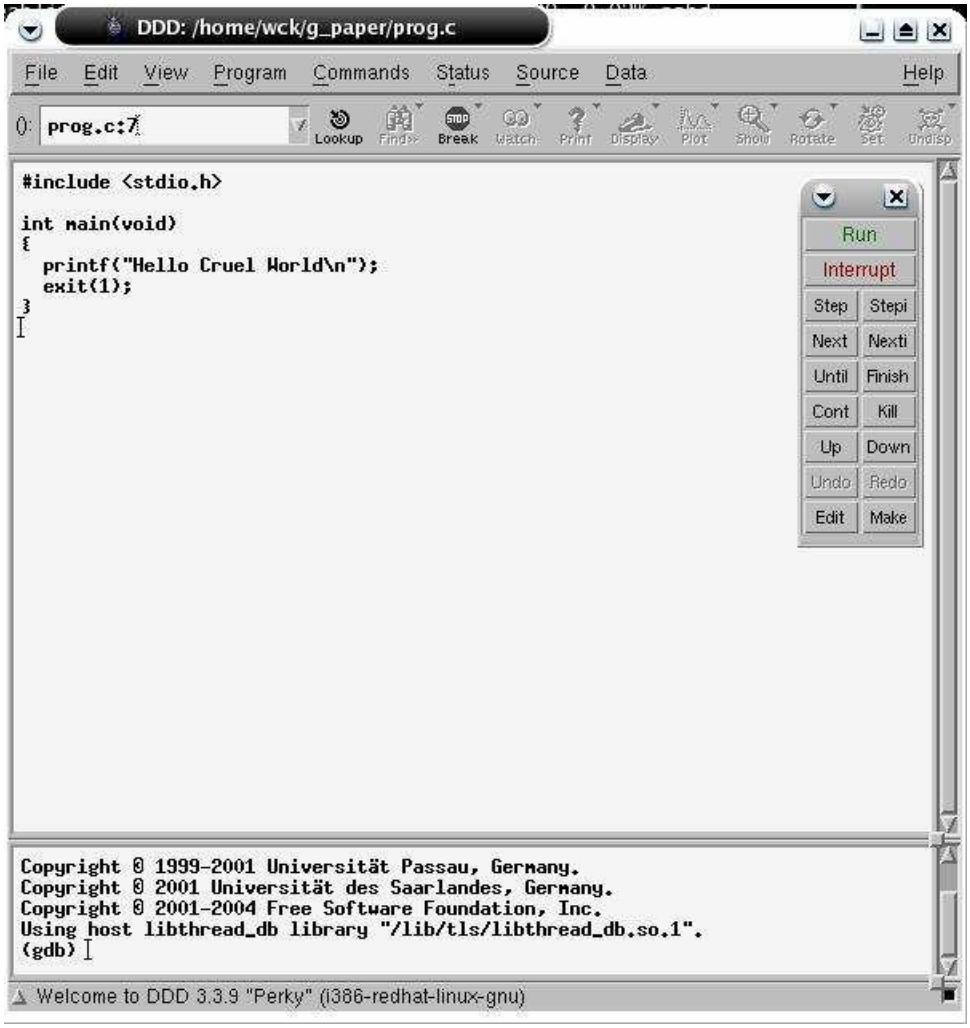

Figure 1: A Screenshot of DDD