Table of Contents

Debugging Your Applications

Introduction

Compile your code in debug mode and use the debugger to trace problems in a program at the source code level.

A debugger enables you to control a program's execution, symbolically monitoring program control flow, variables, and memory locations. You can also use the debugger to trace the logic and flow of control to acquaint yourself with a program written by someone else.

Compiling your Code for Debugging

Command-line debuggers

Command-line debuggers are generally best to use to track down basic coding errors as they are fast to use, and so can be easily used when logged in remotely, as well as being much faster in general for debugging than graphical debuggers, which can be more useful for complex problems.

Debugging on Linux: gdb

The debugger for Linux machines is called gdb. gdb allows you to see what is going on inside a program while it executes -- or what the program was doing at the moment it crashed. gdb can do can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

The basic syntax for gdb is one of the following:

- gdb program - To debug program.
- gdb program core - To debug using the core file, produced when program was core dumped.
- gdb program PID - To debug a running process with process ID number PID.

For more information about gdb, you can look at the man page:

```
man gdb
```

or the info page

```
info gdb
```

The info page in particular contains a lot of information and even a sample gdb session. The info page looks like a text document, but in fact it has links that you can follow to other pages. To navigate the info page, put the cursor on the menu item that you are interested in, and press enter. To exit, press q for quit.

gdb commands

Here are some of the most frequently needed gdb commands:

Command Description
*print [x]*Print the object *xbreak [file:]function*Set a breakpoint at function (in file).*run [arglist]*Start your program (with arglist, if specified).

*bt*Backtrace: display the program stack.*print expr*Display the value of an expression.*c*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.*edit [file:]function*Look at the program line where it is presently stopped.*list [file:]function*type the text of the program in the vicinity of where it is presently stopped.*step*Execute next program line (after stopping); step into any function calls in the line.

*help [name]*Show information about gdb command name, or general information about using gdb.*quit*Exit from gdb.

GUI Debuggers

GUI debuggers have different interfaces on each platform. They are generally a wrapper on the debugger in use. They let you see the code in context as you step through it.

GUI debuggers are nice because they provide a graphical user interface to confusing debugging programs. On the other hand, they can be CPU-intensive, so they can be very slow to run.

ddd

ddd is a nice graphical debugger which is available on Linux. To use it, basically follow the commands for *workshop* below.

This topic: SPI > DebuggingApplications
Topic revision: r2 - 2005-07-27 - AlbertoAimar