Debugging with Ptkdb

Andrew E. Page

1.0 Introduction

Any real programming language and programming environment provides a debugger, a tool for executing your code one step at a time and examining variables as it runs. The best environments provide you with a 'visual debugger' with a graphic user interface, that supports such nice features as a list of variables or expressions, breakpoints that can be set with a click of a mouse, and variables whose value will pop-up when you put the cursor over them.

Until now there have been only two visual debuggers for perl, one from Solution Soft and another from Active state. You have to pay for both, and neither of them work on unix.

Now there is a third alternative, Ptkdb(PerlTk based Debugger). It's written in perl and uses the PerlTk graphic toolkit. As such it will not only run on Windows, but any unix environment that supports perl and perlTk. It supports most of the features that the other debuggers have, plus one very important one. It's Free.

2.0 Perl Debuggers

There is a built in debugger with perl. It is part of the standard distribution. It is a line oriented debugger. Commands are entered through a command line that is typically in the same terminal screen that the perl script was started from. You see the command prompt each time you take a step or hit a breakpoint.

You invoke the debugger by issuing the command as:

perl -d myscript.pl

When you do instead of running your program right away the debugger stops

right at the first line of code to be executed and prompts you for what

to do next.

Ptkdb is a debugger that uses the perlTk graphic user interface(GUI) library. Instead of having to enter commands to start, stop or set breakpoints through a command line, ptkdb allows you to do it with the click of a mouse.

3.0 Getting and Installing Ptkdb

Ptkdb can be downloaded and installed from CPAN or http://www.perl.com/CPAN/authors/id/A/AE/AEPAGE/. The archive contains all the files needed to install ptkdb. You can use either the Makefile.PL to generate a makefile, or you can use the install_ptkdb.pl script which uses a perlTk user interface for the installer.

1 perl Makefile.PL

2 make

3 make install

If for some reason you can't install ptkdb into the system's perl library you can run ptkdb from the directory where you are invoking perl.

1 cd myscriptdir

2 mkdir Devel

3 cp ptkdb.pm Devel

4 perl -d:ptkdb myscript.pl

The current version of ptkdb is written entirely in perl. As such it is portable to any system that supports the perlTk library. This includes Windows 95/98/NT as well as unix.

4.0 Running Ptkdb

Once ptkdb has been installed, you can debug your programs with it like this:

perl -d:ptkdb myScript.pl

When you execute this command the ptkdb window will appear and the first executable line of your script will be highlighted. At this point the debugger has stopped the script and is waiting for your command.

If you use #! /usr/local/bin/perl or some similar method of making perl scripts executable you can add the -d:ptkdb option to invoke the debugger when the script is started.

DO NOT INVOKE it like this:

perl -d:ptkdb.pm myScript.pl

This causes an error.

5.0 Code Pane

The left pane of the window is the code pane. It allows you to see the perl script that you are debugging. Each time the debugger stops when it hits a breakpoint this window automatically opens to the file and highlights the line of code. If you click on the line numbers along the side you can add or remove breakpoints. By default a line with a breakpoint will have its line shown red. Breakpoints can be disabled in the Brkpts control panel by pushing the appropriate checkbox. A disabled breakpoint will be colored green. Lines that are not breakable will be 'struck-through'.

5.1 Getting Around the File

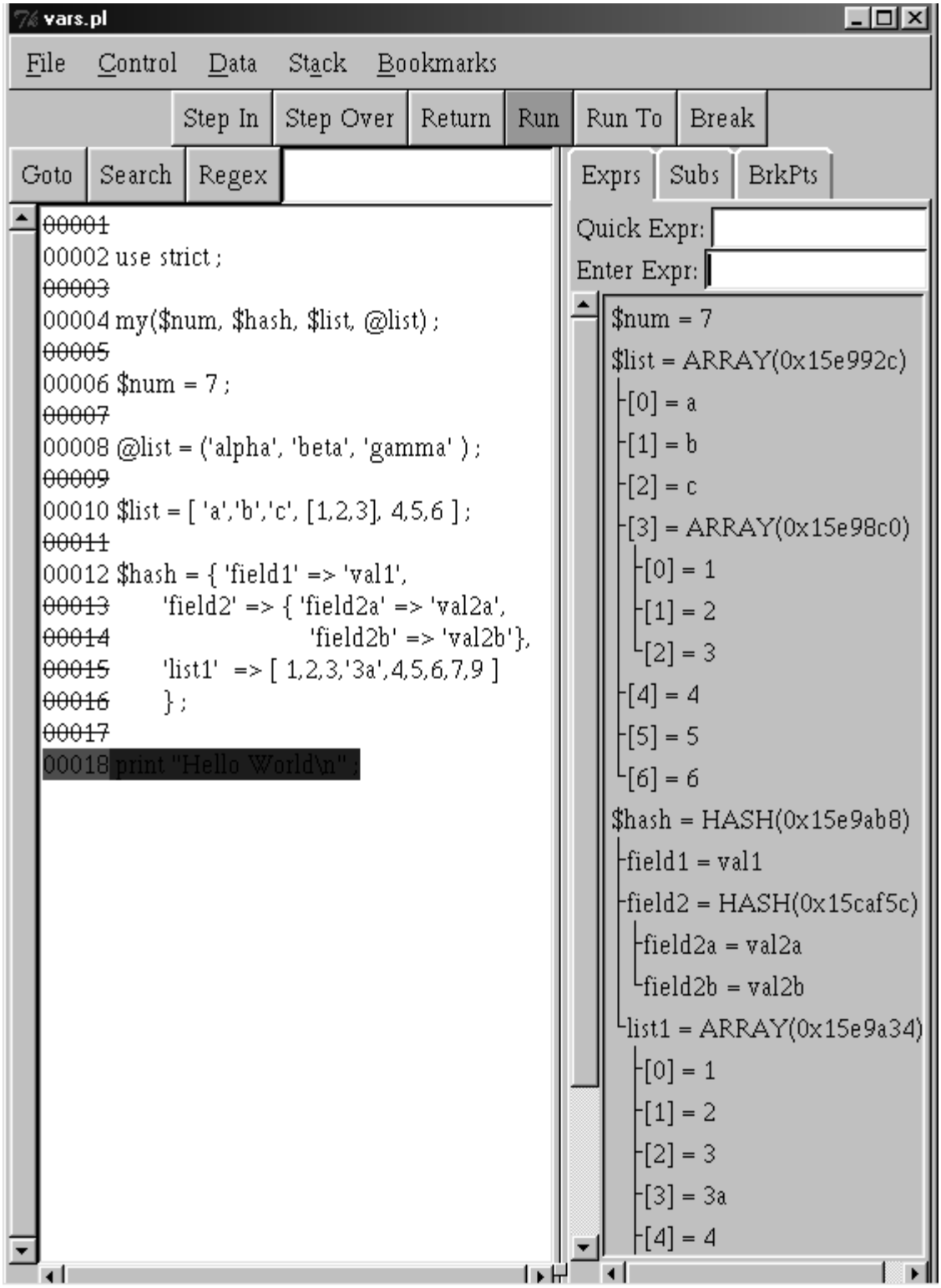
Ptkdb has several tools that allow you to search or move through your code. At the top of the Code window there is a text entry with “Goto”, “Search”, and “Regex” buttons to the left. If you enter line number into text box and press the “Goto” button, the code will scroll to that line. You can search for text by entering text to search for in the entry box and pressing “Search”. You can use regular expressions to search as well by pressing the “Regex” button. These features can be very helpful in finding your way around in pieces of code that others have written that you may not be familiar with. The searching and regular expression features are all part of the Tk::Text widget that comes with the perlTk library.

6.0 Expression Window

The Expression window is the 'Exprs' tab in the notebook in the right pane of the window. This presents a list of expressions. Expressions are entered into the “Enter Expr:” entry and then listed as a hierarchical list. Every time the debugger stops this list of expressions will be reevaluated and displayed. Arrays are broken down by their index, and hashes are broken down by their keys. Any array within the array, or list within a hash will also be decomposed in the same way. Double clicking on any entry will cause it to hide its members. If it is double clicked again it will redisplay it's contents. Entries can be deleted with “Delete Expression” item in the Data menu or Ctrl-d.

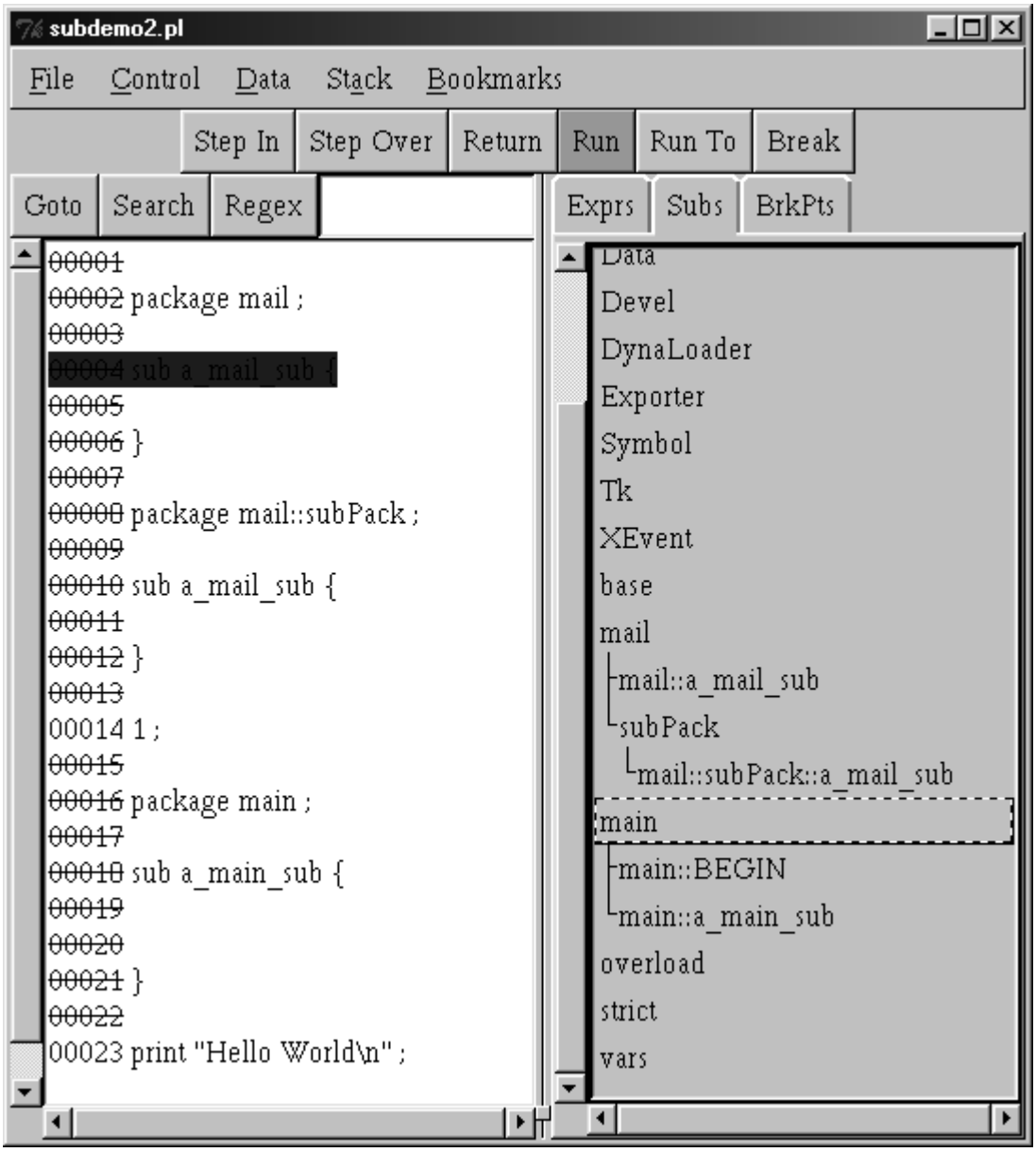
Another way of displaying data(for Tk800 and above) is through the “Hot Variable Balloons”. When the debugger stops and you position the cursor over a variable such as $var, %var or @var and leave it there for a second a text balloon will pop up displaying the vari-

ables value. If you have text selected it will evaluate the selected text as an expression and display the result. If you have Gurusamy Sarathy's Data::Dumper module(included with perl 5.005) installed, Data::Dumper will be used to format the contents of the variable. This is a convenient way to take a quick look at variables without having to have their values constantly listed.


vars.pl
File Control Data Stack Bookmarks
Step In Step Over Return Run Run To Break
Goto Search Regex
Exprs Subs BrkPts
Quick Expr:
Enter Expr:
00001
00002 use strict ;
00003
00004 my($num, $hash, $list, @list) ;
00005
00006 $num = 7 ;
00007
00008 @list = ('alpha', 'beta', 'gamma') ;
00009
00010 $list = ['a','b','c', [1,2,3], 4,5,6] ;
00011
00012 $hash = { 'field1' => 'val1',
00013 'field2' => { 'field2a' => 'val2a',
00014 'field2b' => 'val2b'},
00015 'list1' => [1,2,3,'3a',4,5,6,7,9]
00016 } ;
00017
00018 print "Hello World\n" ;
$num = 7
$list = ARRAY(0x15e992c)
[0] = a
[1] = b
[2] = c
[3] = ARRAY(0x15e98c0)
[0] = 1
[1] = 2
[2] = 3
[4] = 4
[5] = 5
[6] = 6
$hash = HASH(0x15e9ab8)
field1 = val1
field2 = HASH(0x15caf5c)
field2a = val2a
field2b = val2b
list1 = ARRAY(0x15e9a34)
[0] = 1
[1] = 2
[2] = 3
[3] = 3a
[4] = 4

7.0 Subs Panel

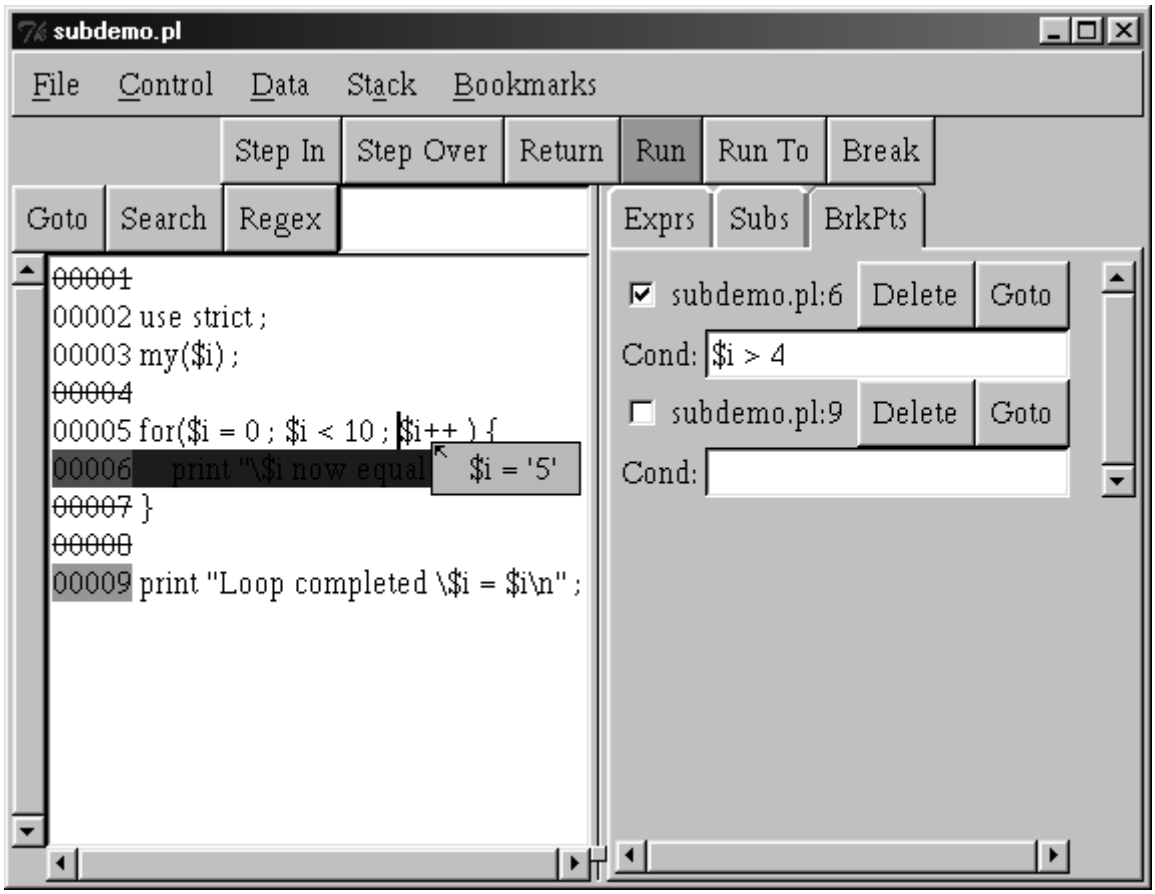
The 'Subs' tab of the notebook brings up the subroutine panel. This panel lists all of your modules and subroutines broken down into a hierarchical list. This tends to be quite lengthy since the debugger includes the perlTk modules. The modules, sub-modules and subroutines within the modules are always arranged alphabetically. You should be able to find your own packages quite easily. There will always be the ‘main’ package. Scripts typically start with “main::BEGIN”. The subroutines are always listed with their fully qualified names, for example: main::mySub().



8.0 Breakpoints

Clicking the "BrkPts" tab brings the breakpoint control panel forward. This panel provides a list of the breakpoints that have been set. There is a check button that will allow you to disable a breakpoint. If the breakpoint is disabled then the debugger will not stop there. There are also control buttons to delete the breakpoint or to goto the file and line where the breakpoint is set. There is an entry box for a controlling expression. Each time the program reaches the breakpoint the debugger will evaluate the expression, if it is 'true', not 'undef' and not 0 then the debugger will stop the program there, otherwise it will continue without stopping until it reaches this point again and rechecks the expression.

TIP: When debugging a program and you want it to stop at a certain point in a loop where X == N, or some other condition, use a condition that will make it stop at the next iteration as well, such as X = N. This way if you to 'go around again' to have another look, you can just press run. For small programs this is not much help, but for large complex scripts, it can save alot of time. Remember, the conditional will be checked each time the breakpoint is hit.



9.0 Customizing Ptkdb

9.1 Startup File

When ptkdb starts up it checks for three startup files. A system startup file, a user's startup file in the $HOME directory, and a startup file that is in the directory in which perl was started. Each of these files can contain a perl script that will be run before the debugger stops at the first line of the program. Ptkdb also provides functions that can be called to set breakpoints, set conditions, enter expressions and customize the text in the code pane. Refer to the ptkdb POD section or man page for documentation.

9.2 Xresources

Many of the ‘look’ and feel aspects of ptkdb can be controlled with Xresources. This can include the fonts that are used and the colors that show breakpoints. See the man page on xrdb for instructions for customizing Xresources. Windows users can now add Xresources for versions Tk800.012 and above. Fonts can be selected with the xfontsel program under Xwindows.

For example, say you wish to change the font that the code is displayed in. Open the .Xdefaults or .Xresources

file in your $ENV{‘HOME’} directory and add the following line:

ptkdb.frame2.frame1.rotext.font: fixed

Change ‘fixed’ to whatever font specification you would like to use. Other fonts can be set individually, or you can use ptkdb*font to specify all of them. The man page contains a commented example section of .Xdefaults that you can use as a guide.

10.0 Advanced Uses of Ptkdb

10.1 Debugging Web/CGI Scripts With Ptkdb

One of the most valuable uses of ptkdb is that it can be used to debug perl CGI pages as they run on a web server. In fact, if you're server is unix based and with a properly configured script it can be used to debug these scripts across a network. It is able to do this by virtue of the fact that it is using a graphic display as its interface. This can not be done with the default debugger since it's interface is the same stream as the http server that is receiving the CGI scripts output.

10.1.1 System Requirements

One of the most valuable features of ptkdb is that it can be used to debug perl CGI pages as they run on a web server. In fact, if you're server is unix based and with a properly con-

figured script it can be used to debug these scripts across a network. It is able to do this by virtue of the fact that it is using a graphic display as its interface. This can not be done with the default debugger since it's interface is the same stream as the http server that is receiving the CGI scripts output.