HP WDB Quick Start Guide

This guide introduces you to the basic commands of the HP WDB debugger, an HP-supported implementation of the GDB debugger.

Abbreviations: You can abbreviate any command to its shortest unambiguous form.

Getting help: Use the `help` command to get online information about commands.

To Start the Debugger

Enter the `gdb` command at a shell prompt, with the executable name as argument (you must have `/opt/langtools/bin` in your path):

```
% gdb a.out
```

HP provides a Visual Interface with both graphical and terminal modes based on Vim 5.7 and WDB. To start the graphical interface:

```
% /opt/langtools/bin/vdb a.out
```

To start the Visual Interface in terminal user interface mode:

```
% /opt/langtools/bin/vdb -tui a.out
```

To start the debugger with the original terminal user interface (TUI), use the `-tui` option. To start the debugger in XDB compatibility mode, which allows you to use many XDB commands, use the `-xdb` option. You may use both:

```
% gdb -tui -xdb a.out
```

To debug a core file, type the executable name, then the core file name:

```
% gdb a.out core
```

To attach to a running process, specify the executable name, then the process ID:

```
% gdb a.out process_id
```

To Start the Target Program

Set a breakpoint on the main program (or the location where you wish to start debugging), then use the `run` command to run the program up to that point:

```
(gdb) b main
(gdb) r
```

You can specify command-line arguments to `run` as well as redirect input or output. The syntax is as follows:

`r [`*args*`] [< `*infile*`] [> `*outfile*`]`

To Restart the Target Program

Use the `run` command:

```
(gdb) r
```

To Interrupt the Target Program

Use Control-C.

To View Source Code

Use the `list` command to display the source code surrounding the current location.

```
(gdb) l
```

Subsequent `list` commands show subsequent sections of source code. A minus sign (`-`) after the command shows previous sections of code. Use `l` *linenum* or `l` *filename*`:`*linenum* to display source around a given line number, and `l` *func* to display source around a given function.

To Fix and Continue Debugging Source Code

Fix and Continue allows you to see the results of changes you make to a program without having to re-compile and re-link the entire program. To use Fix and Continue, you must set your terminal type to hpterm. To edit a program and see the results, stop your program at a breakpoint and type:

```
(gdb) edit
```

This command opens a new terminal window with the source file ready for editing. Make any changes to your code, save the changes and exit the editor.

Use the fix command to re-compile your program and see the results of your changes:

```
(gdb) fix
```

Note: You must rebuild your program after you use the fix command because the changes you make a temporarily patched into the executable image. The changes are lost when you exit the debugger or you load a different executable.

To View Assembly Code

Use the `disassemble` command to get a disassembly display of the current function or a named function:

```
(gdb) disas
```

```
(gdb) disas sum
```

In the TUI, use the `list` command to return to the source display from the disassembly display.

To Step Through Code

Use the `step` command to step *into* called functions:

```
(gdb) s
```

Use the `next` command to step *over* called functions:

```
(gdb) n
```

Use `stepi` (`si`) and `nexti` (`ni`) to step by instruction.

To Continue Execution

Use the `continue` command to run the program until it completes or until a breakpoint or watchpoint is reached:

```
(gdb) c
```

Use `finish` to continue to the end of the current function. Use `until` *location* to continue to a particular location.

To Set a Breakpoint

Use the `break` command.

On a function:

```
(gdb) b sum
```

On a line number:

```
(gdb) b 25
```

On an offset from the current line:

```
(gdb) b +9
```

```
(gdb) b -1
```

On a line number in a given file:

```
(gdb) b myfile.c:45
```

On a memory address (use `*`):

```
(gdb) b *0x2324
```

To Set a Watchpoint

Use the `watch` command to set a watchpoint on a variable:

```
(gdb) wat x
```

To List Breakpoints and Watchpoints

Use `info break` or `info watch` to get a list of all breakpoints and watchpoints:

```
(gdb) i b
```

```
(gdb) i wat
```

To Delete Breakpoints and Watchpoints

Use the `delete` command with the breakpoint or watchpoint number (obtained from `info break` or `info watch`):

```
(gdb) d 7
```

Use `d` with no arguments to delete all breakpoints and watchpoints.

To Print a Variable or Expression

Use the `print` command:

```
(gdb) p i
```

```
(gdb) p i*5
```

Use the format specifier `x` to print the value in hexadecimal:

```
(gdb) p/x i
```

Other format specifiers include `d` (decimal), `t` (binary), `c` (character), and `f` (float).

To Show the Data Type of a Variable

Use the `ptype` command:

```
(gdb) pt i
```

To Change the Value of a Variable

Use the `print` command with an assignment operator to change the value of a variable:

```
(gdb) p i = 2
```

```
(gdb) p i*=4
```

To Examine Registers

Use the `info registers` command to see all integer registers.

```
(gdb) i r
```

Use `info all-registers` to see all registers, including floating-point registers.

```
(gdb) i all
```

Use `info reg` with an argument to see a specific register:

```
(gdb) i r $sp
```

To Examine Memory

Use the `x/i` command to get a disassembly of a limited area of memory. For example, to look at the next 10 instructions after the program counter:

```
(gdb) x/10i $pc
```

Use `x` with other `print` format specifiers to display memory in other formats.

To Obtain a Stack Traceback

Use the `backtrace` command:

```
(gdb) bt
```

To Traverse the Call Stack

Use the `up` and `down` commands, with or without an argument, to move up and down the call stack. The default is to move up or down one level:

```
(gdb) up
```

```
(gdb) down 2
```

Use the `frame` command to move to a specific stack frame:

```
(gdb) f 2
```

Use `frame` with no arguments to find out your current location.

To Exit the Target Program

Use the `kill` command:

```
(gdb) k
```

To Exit the Debugger

Use the `quit` command:

```
(gdb) q
```

To Configure vi Editing Commands

To make command history understand your vi key bindings you need to create a ~/.inputrc file with the following contents:

```
set editing-mode vi
```

The readline interface uses the .inputrc file to control the settings.

Terminal User Interface and XDB Compatibility Commands

The following commands are available when you invoke `gdb` with the `-tui` option:

`disassemble` *addr*	Redirect `disas` command output to disassembly window.
`focus` *win* | `next` | `prev`	Set focus to `next`, `prev` or named window, to allow scrolling commands to take place without a window specification.
`info win`	List the active windows.
`layout` {`prev` | `next` | `split` | *layout_name*}	`next` or `prev` cycles through the available layouts. *layout_name* is one of the following: `src`, `asm`, `regs`, `split`.
`list`	Redirect `list` command output to source window.
`refresh`	Refresh the display.
`tabset` *n*	Set hard tabs in source file to *n* number of spaces.
`update`	Update screen to current execution point
`winheight` *win* [+|-]*n*	Set the height of a window.

Use the `+`, `-`, `<`, `>`, Page Up, Page Down, and arrow keys to scroll the windows.

The following additional commands are available when you invoke `gdb` with both the `-xdb` and `-tui` options:

`fr`, `gr`, `sr`	Show floating-point, general, special registers.
`td`	Toggle between source and disassembly display.
`tf`	Toggle floating-point register display precision.
`ts`	Toggle between split (Source/Disassembly/Command) and Source/Command or Disassembly/Command.
`u`	Update screen to current execution point.
`U`	Refresh the display.
`w` *n*	Set the height of a window.

Many more XDB commands are available when you invoke `gdb` with the `-xdb` option (with or without `-tui`): `am`, `ba`, `bc`, `bu`, `bx`, `D`, `g`, `l`, `L`, `lb`, `lc`, `ld`, `lf`, `lg`, `lr`, `lz`, `Q`, `R`, `S`, `sm`, `t`, `T`, `v`, `V`, `va`, `z`, `/`, `?`, `!`.