|  |  |  |
| --- | --- | --- |
| Bochs User Manual | | |
| [Prev](http://docs.google.com/internal-debugger.html) | Chapter 8. Tips and Techniques | [Next](http://docs.google.com/serial-port.html) |

8.14. Using Bochs and the remote GDB stub

This section covers how you can use Bochs with a remote GDB stub to debug your kernel.

8.14.1. Configuring Bochs

The GDB stub is not active in standard Bochs binary package. So you must recompile Bochs. Download the Bochs source package, unpack it and run the configure script with the --enable-gdb-stub argument.

$ ./configure --enable-gdb-stub

After that, just run make and you should have a Bochs binary that contain a GDB stub in your directory.

8.14.2. Running Bochs

Enable the [gdbstub option](http://docs.google.com/bochsrc.html#BOCHSOPT-GDBSTUB) in bochsrc, then just start Bochs as normal. Bochs will stop and wait for GDB to connect to the stub.

8.14.3. Running GDB

Bochs GDB stub waits for a connection on TCP port 1234. Just start GDB like this:

$ gdb YOUR-KERNEL  
 .  
 .  
 .  
 (gdb) target remote localhost:1234  
 Remote debugging using localhost:1234  
 0x0000fff0 in ?? ()  
 (gdb)

You are now connected to the remote GDB stub in Bochs. You are now able to set breakpoints. Use the continue (c) command to continue the simulation. Hitting ^C works. Example:

Program received signal 0, Signal 0.  
 syscall\_testsuite\_result (aux=0x1f11fe4) at ../rtmk/syscalls.c:33  
 33 {  
 (gdb)

|  |  |  |
| --- | --- | --- |
| [Prev](http://docs.google.com/internal-debugger.html) | [Home](http://docs.google.com/index.html) | [Next](http://docs.google.com/serial-port.html) |
| Using Bochs internal debugger | [Up](http://docs.google.com/howto.html) | Using the serial port |