Excerpts from GDB Manual
Richard Stallman, et al.

Debugging programs with multiple processes

On most systems, GDB has no special support for debugging programs which create additional processes using the `fork` function. When a program forks, GDB will continue to debug the parent process and the child process will run unimpeded. If you have set a breakpoint in any code which the child then executes, the child will get a SIGTRAP signal which (unless it catches the signal) will cause it to terminate.

However, if you want to debug the child process there is a workaround which isn't too painful. Put a call to sleep in the code which the child process executes after the fork. It may be useful to sleep only if a certain environment variable is set, or a certain file exists, so that the delay need not occur when you don't want to run GDB on the child. While the child is sleeping, use the ps program to get its process ID. Then tell GDB (a new invocation of GDB if you are also debugging the parent process) to attach to the child process (see section "Debugging an already-running process"). From that point on you can debug the child process just like any other process which you attached to.

On some systems, GDB provides support for debugging programs that create additional processes using the `fork` or `vfork` functions. Currently, the only platforms with this feature are HP-UX (11.x and later only) and GNU/Linux (kernel version 2.5.60 and later).

By default, when a program forks, GDB will continue to debug the parent process and the child process will run unimpeded.

If you want to follow the child process instead of the parent process, use the command set `follow-fork-mode`.

Commands:

`set follow-fork-mode` *mode*

Set the debugger response to a program call of fork or vfork. A call to fork or vfork creates a new process. The *mode* can be:

`parent`

The original process is debugged after a `fork`. The child process runs unimpeded. This is the default.

`child`

The new process is debugged after a `fork`. The parent process runs unimpeded.

`show follow-fork-mode`

Display the current debugger response to a `fork` or `vfork` call.

If you ask to debug a child process and a `vfork` is followed by an `exec`, GDB executes the new target up to the first breakpoint in the new target. If you have a breakpoint set on main in your original program, the breakpoint will also be set on the child process's main.

When a child process is spawned by `vfork`, you cannot debug the child or parent until an exec call completes.

If you issue a run command to GDB after an exec call executes, the new target restarts. To restart the parent process, use the file command with the parent executable name as its argument.

You can use the catch command to make GDB stop whenever a `fork`, `vfork`, or `exec` call is made. See section "Setting catchpoints".

Debugging an already-running process

Command:

`attach` *process-id*

This command attaches to a running process--one that was started outside GDB. (The command "`info files`" shows your active targets.) The command takes as argument a process ID. The usual way to find out the process-id of a Unix process is with the ps utility, or with the 'jobs -l' shell command. attach does not repeat if you press RET a second time after executing the command.

To use `attach`, your program must be running in an environment which supports processes; for example, attach does not work for programs on bare-board targets that lack an operating system. You must also have permission to send the process a signal.

When you use attach, the debugger finds the program running in the process first by looking in the current working directory, then (if the program is not found) by using the source file search path (see section "Specifying source directories"). You can also use the file command to load the program. See section "Commands to specify files".

The first thing GDB does after arranging to debug the specified process is to stop it. You can examine and modify an attached process with all the GDB commands that are ordinarily available when you start processes with run. You can insert breakpoints; you can step and continue; you can modify storage. If you would rather the process continue running, you may use the continue command after attaching GDB to the process.

Command:

`detach`

When you have finished debugging the attached process, you can use the detach command to release it from GDB control. Detaching the process continues its execution. After the detach command, that process and GDB become completely independent once more, and you are ready to attach another process or start one with run. detach does not repeat if you press RET again after executing the command.

If you exit GDB or use the run command while you have an attached process, you kill that process. By default, GDB asks for confirmation if you try to do either of these things; you can control whether or not you need to confirm by using the set confirm command (see section "Optional warnings and messages").