Detailed GDB Debugging Session Example

Scenario

You are debugging an Arduino UNO program where an LED connected to pin 13 is supposed to blink every second, but it blinks too fast due to incorrect delay logic.

Objective

1. Use **GDB commands** to debug the program step-by-step.
2. Identify and fix the issue.

Setup

1. **Tool Requirements:**
   - Arduino IDE (to write and compile the code).
   - `AVR-GDB` (GNU Debugger for AVR microcontrollers).
   - Simulated environment like **SimAVR** or **Wokwi**.

Sample Arduino Code (LED Blink):

```
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // Turn LED on
  delay(500);            // Incorrect delay (should be 1000)
  digitalWrite(13, LOW);  // Turn LED off
  delay(500);            // Incorrect delay (should be 1000)
```

2. }
3. **Compile Code for Debugging:**
   - Export compiled binaries with debug symbols enabled:**GDB Debugging Session**

Step 1: Start GDB

Run GDB with the compiled binary:

```
avr-gdb blink.elf
```

Step 2: Load the Program

Set up the simulator or microcontroller emulator (e.g., SimAVR) to run with GDB. For example:

```
target sim
load
```

This initializes the target and loads the compiled program into memory.

Step 3: Set Breakpoints

Set a breakpoint at the start of the `loop()` function:

```
break loop
```

This will halt the program when execution reaches the `loop()` function.

Step 4: Run the Program

Start the program execution:

```
run
```

The program will stop at the `loop()` function.

Step 5: Inspect Variables and Registers

While paused at the breakpoint, inspect the state of the program:

- **View Register Values**:
  info registers

Check Variables:
print pin

- print delay

Step 6: Step Through the Code

Execute one line of code at a time:

- **Step Into Functions**:
  step
- **Step Over Functions**:
  next

For example:

- Step into `digitalWrite(13, HIGH)` and observe its behavior.
- Verify the delay value when stepping into `delay(500)`.

Step 7: Modify Variables at Runtime

Change the delay value dynamically to test the program:

set delay = 1000

Step 8: Continue Execution

Resume execution until the next breakpoint:

continue

Step 9: Debugging Example

Identify the incorrect delay value in the following sequence:

1. Program halts at the `loop()` function.
2. You step through `digitalWrite(13, HIGH)` and `delay(500)`.
3. Inspect the value of the delay and confirm it's incorrect:
   print delay
   Output:
   $1 = 500
4. Change the delay value to `1000`:
   set delay = 1000

Step 10: Exit GDB

Once debugging is complete, exit GDB:

quit

Outcome

By following the GDB session:

- You identified the incorrect delay value causing fast blinking.
- You dynamically fixed the value during runtime using `set delay = 1000`.
- Verified the fix by stepping through the program and observing the corrected LED behavior.