DEBUGGING MULTI-THREADED PROGRAMS WITH GDB
PREREQUISITES

• Breakpoints (break 5)
• Single-stepping (step; next)
• Watchpoints (watch i)
• All above is implemented with low-level breakpoints
  • Software breakpoints
    • Instruction injection
  • Hardware breakpoints
    • Debugging registers in CPU
AUTOMATIC DETECTION

• GDB auto-detects new threads (and their death)
  • Implemented by breakpoints in thread library

• Message in the form [New systag]

• Systag is a thread identifier whose form depends on the OS

• The systag can vary a lot
  • [New process 35 thread 27]
  • [New Thread 0x41e02940 (LWP 25582)]

• GDB associates its own thread ID (a small integer) with each thread
INFO ABOUT EXISTING THREADS

- `info threads [thread-id-list]`
- Displays info about threads
  1. Asterisk indicating the current thread
  2. Thread ID
  3. Systag
  4. The thread's name, if one is known
  5. The thread's stack frame summary

```
(gdb) info threads
  Id  Target ID  Frame
  3   process 35 thread 27 “myname” 0x34e5 in sigpause ()
  2   process 35 thread 23 “a.out” 0x34e5 in sigpause ()
* 1   process 35 thread 13 “a.out” main (argc=1, argv=0x7fffffff8) at threadtest.c:68
```
SWITCHING BETWEEN THREADS

• Switching the current thread changes the debugging focus
• Command `thread thread-id`
• Automatic when GDB takes control from different thread
  • All-Stop mode only
• Message `[Switching to systag]` plus the current stack frame summary

(gdb) thread 2
[Switching to process 35 thread 23]
0x34e5 in sigpause ()
THREAD-SPECIFIC BREAKPOINTS AND WATCHPOINTS

• **break** location thread thread-id
  • break 6 thread 2
  • *Breaks only when the specified thread hits the breakpoint*

• **watch** expr thread thread-id
  • watch a+b thread 2
  • *Breaks only when the specified thread changes the result of the expression*
  • *Must be implemented with hardware breakpoints*
WATCHPOINTS

• Hardware watchpoints work as expected

• Software watchpoints have limitations, you must ensure that
  • The expression can only change due to the current thread’s activity
  • No other thread can become the current thread
DEBUGGING

• Two modes
  • All-Stop
    • GDB stops all threads when it takes control
    • Default
  • Non-Stop
    • Commands apply only to the current thread
    • Auto-switching is disabled

• Mode must be set before running the program
SINGLE-STEPPING

• Commands like `step` or `next`
• GDB single-steps only the current thread
• Steps are not atomic
  • Another thread may hit a breakpoint in the middle of the step
INTERRUPTED SYSTEM CALLS

• A system call may return prematurely if another thread stops (in GDB)
• It is allowed to do that!
  • Good programming style is to check its return value
• Same when another thread is created/destroyed

```c
int unslept = sleep(10); //may not sleep for 10 sec
//you should check return value
int unslept = 10;
while(unslept > 0)
{
    unslept = sleep(unslept);
}
```
SOME EXTRA COMMANDS

• thread apply [thread-id-list | all {-ascending}] command

• thread find [regexp]