gdb: example

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#include "show.h"

int main(int argc, char *argv[]) {
    double a[10], sum;
    int i;
    for (i = 0; i <= 10; i++)
        a[i] = sqrt(i);
    for (i = 0; i < 10; i++)
        sum += a[i];
    printResult("sum", sum);
    return EXIT_SUCCESS;
}
```

array-bounds.c

```c
#ifndef SHOW_HDR
#define SHOW_HDR

void printResult(char *name,
                 double r);

#endif
```

show.h

```c
#include <stdio.h>

void printResult(char *name,
                 double r) {
    printf("%s = %f\n",
           name, r);
}
```

show.c

```
$ ./a.out
sum = 22.468278
```

Oops!

$\neq$

```
octave:1> sum(sqrt([0:9]))
ans =  19.306
```

gdb: compiling code & starting gdb

- Compile program for debugging

```
$ gcc  -g2  array-bounds.c show.c
```

- Running under gdb control: start gdb

```
$ gdb  ./a.out
GNU gdb 6.8.0.20080328-cvs (cygwin-special)
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.o …
This is free software: you are free to change and redist …
There is NO WARRANTY, to the extent permitted by law.  T …
and "show warranty" for details.
This GDB was configured as "i686-pc-cygwin"...
(gdb)
```

gdb: listing source code

- Listing code 10 line window: `l(ist)`

```
(gdb) l
1       #include <math.h>
2       #include <stdio.h>
3       #include <stdlib.h>
4
5       #include "show.h"
6
7       int main(int argc, char *argv[]) {
8           double a[10], sum;
9           int i;
10          for (i = 0; i <= 10; i++)
```

- L

```
(gdb) l
11              a[i] = sqrt(i);
12          for (i = 0; i < 10; i++)
13              sum += a[i];
14          printf("sum = %lf\n", sum);
15          return EXIT_SUCCESS;
16      }
17
```

gdb: listing source code

- Listing code 10 line window around line 9:

```
(gdb) l 9
4
5       #include "show.h"
6
7       int main(int argc, char *argv[]) {
8           double a[10], sum;
9           int i;
10          for (i = 0; i <= 10; i++)
11              a[i] = sqrt(i);
12          for (i = 0; i < 10; i++)
13              sum += a[i];
```

- L 13

```
(gdb) l 10,13
10          for (i = 0; i <= 10; i++)
11              a[i] = sqrt(i);
12          for (i = 0; i < 10; i++)
13              sum += a[i];
(
```

gdb: listing source code

- Listing function: `l <func>`

```
(gdb) l main
2       #include <stdio.h>
3       #include <stdlib.h>
4
5       #include "show.h"
6
7       int main(int argc, char *argv[]) {
8           double a[10], sum;
9           int i;
10          for (i = 0; i <= 10; i++)
11              a[i] = sqrt(i);
```

- L
<

```
(gdb) l show.c:printResult
1       #include <stdio.h>
2
3       void printResult(char *name, double r) {
4           printf("%s = %f\n", name, r);
5   }
6
```

gdb: running a program

- Run the code: `r(un)`

```
(gdb) r
Starting program: ./a
[New thread 13704.0x32e0]
[New thread 13704.0x2110]
sum = 22.468278

Program exited normally.
```

- Not so interesting, but one can
  - Set breakpoints
  - Set conditional breakpoints
  - Watch stuff

gdb: breakpoints

- Set breakpoint at line number: `b(reak)`

```
(gdb) b 10
Breakpoint 1 at 0x401199: file array-bounds.c, line 10.
(gdb) r
```

```
8           double a[10], sum;
9           int i;
10  ➔       for (i = 0; i <= 10; i++)
11              a[i] = sqrt(i);
12          for (i = 0; i < 10; i++)
```

array-bounds.c

```
(gdb) b printResult
Breakpoint 1 at 0x40123a: file show.c, line 4.
(gdb) r
```

```
1       #include <stdio.h>
2
3       void printResult(char *name, double r) {
4   ➔       printf("%s = %f\n", name, r);
5       }
```

show.c

gdb: at breakpoints

- Inspect value of variables: `p(rint) <var>`
- Proceed execution by
  - Stepping
  - With descending into subroutines: `s(tep)`
  - Without descending into subroutines: `n(ext)`
  - Until next statement: `u(until)`
  - Continuing to next breakpoint: `c(ontinue)`
- Handle breakpoints
  - List: `i(nfo) b(reakpoints)`
  - Remove: `d(elete) <bn>`
  - Disable/enable: `disable <bn>/enable <bn>`

gdb: example stepping

```
(gdb) b main
Breakpoint 1 at 0x401194: file array-bounds.c, line 7.
(gdb) r
Starting program: /cygdrive/c/Users/lucg5005/Documents/My
Dropbox/Projects/HPC/Samples/DebuggingProfiling/trunk/Gdb/a
[New thread 11372.0x352c]
[New thread 11372.0x322c]

Breakpoint 1, main () at array-bounds.c:7
7       int main(int argc, char *argv[]) {
(gdb) n
10          for (i = 0; i <= 10; i++)
(gdb)
11              a[i] = sqrt(i);
(gdb) p i
$1 = 0
(gdb) n
10          for (i = 0; i <= 10; i++)
(gdb)
11              a[i] = sqrt(i);
(gdb) p a
$2 = {0, 1, 8.0461413318721078e-315, 0, 4.7233218921966505e+192,
4.7151331558996709e+192, 2.9835385156662606e-314,
  4.0742432477169204e-312, 5.0488079539729615e-314,
3.5150220537551154e+159}
```

gdb: counted steps

```
(gdb) b 11
Breakpoint 3 at 0x4011a2: file array-bounds.c, line 11.
(gdb) c
Continuing.

Breakpoint 3, main () at array-bounds.c:11
11              a[i] = sqrt(i);
(gdb) p i
$1 = 0
(gdb) c 5
Will ignore next 4 crossings of breakpoint 3.  Continuing.

Breakpoint 3, main () at array-bounds.c:11
11              a[i] = sqrt(i);
(gdb) p i
$2 = 5
```

gdb: example handling breakpoints

```
(gdb) b 12
Breakpoint 2 at 0x4011e1: file array-bounds.c, line 12.
(gdb) i b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x00401194 in main at array-bounds.c:7
        breakpoint already hit 1 time
2       breakpoint     keep y   0x004011e1 in main at array-bounds.c:12
(gdb) delete 1
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
2       breakpoint     keep y   0x004011e1 in main at array-bounds.c:12
```

```
(gdb) c
Continuing.

Breakpoint 2, main () at array-bounds.c:12
12          for (i = 0; i < 10; i++)
(gdb) p a
$3 = {0, 1, 1.4142135623730951, 1.7320508075688772, 2, 2.2360679774997898,
  2.4494897427831779, 2.6457513110645907, 2.8284271247461903, 3}
```

gdb: conditional breakpoint

- Break conditionally: `b <ln> if <cond>`

```
(gdb) b 11 if i == 4
Breakpoint 2 at 0x4011a2: file array-bounds.c, line 11.
(gdb) c
Continuing.

Breakpoint 7, main () at array-bounds.c:11
11              a[i] = sqrt(i);
(gdb) p i
$1 = 4
(gdb) c
Continuing.
sum = 22.468278

Program exited normally.
```

Only break when condition holds

gdb: dis/enabling breakpoints

- Disable breakpoint: `disable <bn>`

```
(gdb) b 11
Breakpoint 1 at 0x4011a2: file array-bounds.c, line 11.
(gdb) b 13
Breakpoint 2 at 0x4011ea: file array-bounds.c, line 13.
(gdb) r
Breakpoint 1, main () at array-bounds.c:11
11              a[i] = sqrt(i);
(gdb) c
Continuing.

Breakpoint 1, main () at array-bounds.c:11
11              a[i] = sqrt(i);
(gdb) disable 1
(gdb) c
Continuing.

Breakpoint 2, main () at array-bounds.c:13
13              sum += a[i];
(gdb) i b
Num     Type           Disp Enb Address    What
1       breakpoint     keep n   0x004011a2 in main at array-bounds.c:11
        breakpoint already hit 2 times
2       breakpoint     keep y   0x004011ea in main at array-bounds.c:13
        breakpoint already hit 1 time
```

gdb: issuing commands at break

- Commands associated with breakpoint will be executed each

```
(gdb) b 11
Breakpoint 2 at 0x4011a2: file array-bounds.c, line 11.
(gdb) commands 2
Type commands for when breakpoint 2 is hit, one per line.
End with a line saying just "end".
>silent
>printf "i = %d\n", i
>continue
>end
(gdb) c
Continuing.
i = 0
i = 1
i = 2
...
i = 8
i = 9
i = 10
```

don't print breakpoint info

don't stop at breakpoint

gdb: watch

- halt on change: `watch <expr>`

```
(gdb) b 10
Breakpoint 1 at 0x401199: file array-bounds.c, line 8.
(gdb) r
Starting program: /cygdrive/c/Users/lucg5005/Documents/My
Dropbox/Projects/HPC/Samples/DebuggingProfiling/trunk/Gdb/a
[New thread 4556.0x17ec]
[New thread 4556.0x102c]

Breakpoint 1, main () at array-bounds.c:10
10          for (i = 0; i <= 10; i++)
(gdb) watch a[5]
Hardware watchpoint 2: a[5]
(gdb) c
Continuing.
Hardware watchpoint 2: a[5]

Old value = 4.7151331558996709e+192
New value = 2.2360679774997898
main () at array-bounds.c:10
10          for (i = 0; i <= 10; i++)
(gdb) p i
$1 = 5
```

a[5] modified

gdb: more watch

- Types of watch points:
  - Halt on write variable: `watch <expr>`
  - Halt on read variable: `rwatch <expr>`
  - Halt on read/write variable: `awatch <expr>`
- List watchpoints: `i(info) watchpoints`
  - Also shows breakpoints
  - Synomym for `i(nfo) b(reakpoints)`
- Removing, disabling/enabling watchpoints
  - Same as for breakpoints

gdb: watch example

```
(gdb) watch sum
Hardware watchpoint 3: sum
(gdb) c
Continuing.
Hardware watchpoint 3: sum

Old value = 5.2844206111867101e+159
New value = 3.1622776601683795
main () at array-bounds.c:10
10          for (i = 0; i <= 10; i++)
```

Oops, not expected!

```
int main(int argc, char *argv[]) {
    double a[10], sum;
    int i;
    for (i = 0; i <= 10; i++)
        a[i] = sqrt(i);
    for (i = 0; i < 10; i++)
        sum += a[i];
    printResult("sum", sum);
    return EXIT_SUCCESS;
}
```

array-bounds.c

gdb: stack frames

```
#include <stdio.h>
#include <stdlib.h>

long fib(long n) {
    if (n < 2)
        return 1;
    else {
        long a = fib(n-1);
        long b = fib(n-2);
        return a + b;
    }
}

int main(int argc, char *argv[]) {
    long n = atol(argv[1]);
    printf("fib(%ld) = %ld\n", n, fib(n));
    return EXIT_SUCCESS;
}
```

fib.c

gdb: backtrace

```
(gdb) b 6
Breakpoint 1 at 0x40118d: file fib.c, line 6.
(gdb) r 5
Starting program: /cygdrive/c/Users/lucg5005/Documents/My
Dropbox/Projects/HPC/Samples/DebuggingProfiling/trunk/Gdb/fib 5
[New thread 7548.0x1ee8]
[New thread 7548.0x2534]

Breakpoint 1, fib (n=1) at fib.c:6
6               return 1;
(gdb) bt
#0  fib (n=1) at fib.c:6
#1  0x004011a4 in fib (n=2) at fib.c:8
#2  0x004011a4 in fib (n=3) at fib.c:8
#3  0x004011a4 in fib (n=4) at fib.c:8
#4  0x004011a4 in fib (n=5) at fib.c:8
#5  0x004011f9 in main (argc=2, argv=0xbd49d8) at fib.c:13
(gdb) frame
#0  fib (n=1) at fib.c:6
6               return 1;
(gdb) frame 1
#1  0x004011a4 in fib (n=2) at fib.c:8
8               return fib(n-1) + fib(n-2);
(gdb) frame 5
#5  0x004011f9 in main (argc=2, argv=0xbd49d8) at fib.c:13
13          printf("fib(%ld) = %ld\n", n, fib(n));
```

gdb: inspecting frames

- Getting information
  - Print local variables: `i(nfo) locals`
  - Print all frame info: `i(nfo) f(rame)`
- Moving to other frame
  - Move to another frame: `f(rame) <fn>`
  - Move up a frame: `up`
  - Move down a frame: `down`

gdb: reverse debugging

- For gdb ≥ 7.3! (introduced in 7.0)
- Allows to "step back in time", i.e., reverse execution (records changes)
- Slow, so
  - Set breakpoint close to (but before) point of interest
  - Run up to breakpoint, use `record`
  - Continue till error
  - Step back by `reverse-next` (`rn`), `reverse-step` (`rs`), `reverse-continue` (`rc`)
  - Breakpoints/watch expression should all work

Not mature yet, use with caution!

gdb: multithreaded programs

- OpenMP code

```
program hello_world
  integer*4 nthreads, tid, &
       &    OMP_GET_NUM_THREADS, OMP_GET_THREAD_NUM
  ! Fork a team of threads giving them their own copies of variables
  !$OMP PARALLEL PRIVATE(nthreads, tid)
  ! Obtain thread number
  tid = OMP_GET_THREAD_NUM()
  print *, 'Hello World from thread = ', tid
  ! Only master thread does this
  if (tid == 0) then
     nthreads = OMP_GET_NUM_THREADS()
     print *, 'Number of threads = ', nthreads
  end if
  ! All threads join master thread and disband
  !$OMP END PARALLEL
end program hello_world
```

Hello-world.f90

gdb: switching threads

- `thread <tn>` to switch

```
(gdb) b 11
(gdb) r
Starting program: hello
[New thread 0xb7fe7b70 (LWP 1432)]
[Swithing to thread 0xb7fe7b70 (LWP 1432)]
Breakpoint 1, MAIN__.omp_fn.0 (.omp_data_i=0x0) at hello-world.f90:11
11 print *, 'Hello World from thread = ', tid
(gdb) info threads
•   2 Thread 0xb7fe7b70 (LWP 1432)  MAAIN__.omp_fn.0 (.omp_data_i=0x0)
        at hello-world.f90:11
  1 Thread 0xb7fe8700 (LWP 1429)  MAAIN__.omp_fn.0 (.omp_data_i=0x0)
        at hello-world.f90:11
(gdb) p tid
$1 = 1
(gdb) thread 1
[Switching to thread 1 (Thread 0xb7fe8700 (LWP 1429)]#0 MAIN__.omp_fn.0
(.omp_data_i=0x0) at hello-world.f90:11
11 print *, 'Hello World from thread = ', tid
(gdb) p tid
$2 = 0
```

breakpoint hit in *every* thread

- Break in specific thread `b 13 thread 1`