

Debugging with GDB

(Please watch the [video](#), these slides are just a quick recap)

Our Program

```
#define ARRAY_SIZE 100

/* array[slot] = value */
int update_slot(int *array, int slot,
               int value) {
    array[slot] = value;
    printf("Updated index %d to %d\n",
          slot, value);
}

int process_array(int *array, int size) {
    int ii = 1000000;

    for(int i=0; i<size; i++) {
        // If the value is even,
        // change it to 1000000
        if(!(array[i] % 2)) {
            update_slot(array, ii, i);
        }
    }
}
```


```
int fill_array(int *array, int size) {
    for(int i=0; i<size; i++)
        array[i] = rand()%10;
}

int main(int argc, char **argv) {
    int *array = malloc(ARRAY_SIZE *
                       sizeof(int));

    fill_array(array, ARRAY_SIZE);
    process_array(array, ARRAY_SIZE);

    free(array);
    return 0;
}
```

```
//
```

[19-debugging/buggy-program.c](#) 

GDB Essentials

- Compile with `-g` flag to include debug symbols
- To launch GDB on a particular binary : `gdb program` in a shell
- `b test.c:12` places a breakpoint on the instruction at line 18 in `test.c`
- `b my_func` places a breakpoint on the first instruction of `my_func`
- `b test.c:12 if x == 42` is a conditional breakpoint, will pop only if `x` is equal to 42 when the corresponding statement is executed
- `run` starts the program
- In single-step mode:
 - `up` and `down` allows to navigate the function call stack
 - `next` advances the execution, jumping over function calls
 - `step` advances the execution, diving into function calls
 - `continue` exits single step mode

Further Resources

- Official documentation:
<https://sourceware.org/gdb/current/onlinedocs/gdb/index.html>
 - GDB reference card: <https://users.ece.utexas.edu/~adnan/gdb-refcard.pdf>
 - GDB's interface can be heavily customised, for example:
<https://github.com/cyrus-and/gdb-dashboard>
-

Feedback form: <https://bit.ly/3Cn2no5>

