

DEBUGGING

GENERAL DEBUGGING

- `gdb` = GNU Project debugger
- lets you look at what's going on inside a program as it runs
- helpful for identifying:
 - segfaults
 - logical errors

RUNNING GDB

```
gcc -g program.c  
gdb a.out
```

- inside gdb:
 - `run` - start execution from beginning
 - `run < input_file > output_file` if you want to run with input/output redirection
 - `kill` - stop program execution
 - `quit` - exit gdb

BASIC COMMANDS

- `p variable_name` - print a specific variable
- `p *array@len` - print array with certain length (you specify array and len)
- `backtrace` - how did your program get to this point
 - often helpful after a segfault
 - can also use `bt` (abbreviated command)
- `step` - execute one line of code
 - `step nsteps` - execute a specified number of lines of code

BREAKPOINTS

- Setting breakpoints
 - `break function_name`
 - `break line_number`
 - `break line_number if condition`
- `continue` - **step until next break boint**
- `info break` - **list all breakpoints**

BREAKPOINTS (CONT.)

- Enabling/disabling
 - `disable breakpoint_number`
 - `enable breakpoint_number`
- Deleting breakpoints
 - `delete` - delete all breakpoints
 - `delete breakpoint_number` - delete a specific breakpoint

VALGRIND

```
gcc -g program.c  
valgrind --leak-check=yes ./a.out
```

- tool used to debug memory leaks
- may be other errors at the top -- don't just ignore them