Data Structures and Algorithms

GDB, Valgrind, Makefiles and more!

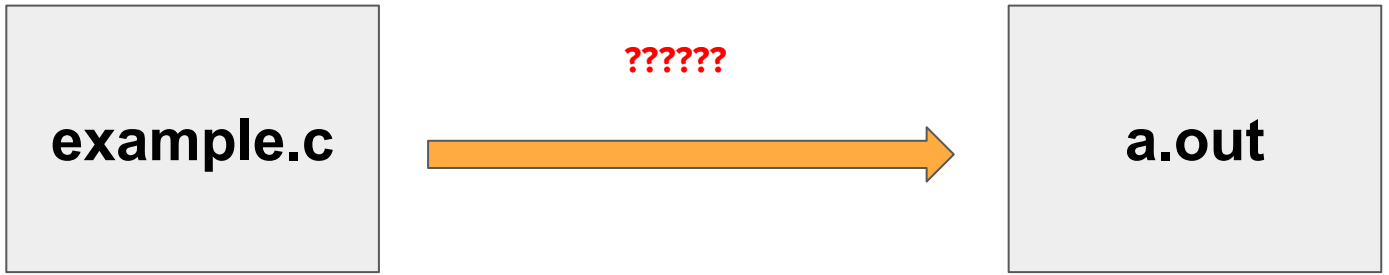
INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Outline

1. Recap of compilation steps.
2. A brief introduction to **Makefiles**!
3. Memory error debugging with **valgrind**.
4. **GDB** - introduction and real-time debugging.

Stages of compilation

How do you go from a .c file to the a.out executable?



Stages of compilation

1. **Preprocessing**
   - **input:** .c file **output:** .i file
2. **Compilation**
   - **input:** .i file **output:** .s file
3. **Assembly**
   - **input:** .s file **output:** .o file
4. **Linking**
   - **input:** .o file **output:** .out file

Stages of compilation (use --save-temps flag)

1. **Preprocessing (-E flag)**
   - **Preprocessor directives #** are interpreted and **commands** are stripped out.
2. **Compilation (-S flag)**
   - Preprocessed code is translated to assembly instructions *specific to the target processor architecture.*
3. **Assembly (-c flag; view file using hexdump)**
   - Translate the assembly instructions to object code (actual instructions).
4. **Linking**
   - The object code generated has missing pieces of instructions. These are filled in during linking.

Makefiles

1. What are makefiles?

2. Why makefiles?

3. How to create makefiles?

Makefiles

1. **What are makefiles?**
   - A makefile is a special file, containing shell commands named makefile.
   - Type make and the commands in the makefile will be executed.
2. **Why makefiles?**
   - Typing gcc for each individual file is not always a good idea;

     ex - files with dependencies, order of compilation matters; many files in project; etc.
3. **How to create makefiles?**
   - The makefile contains a list of rules. These rules tell the system what commands you want to be executed.
   - ```
     DEPENDENCY LINE   TARGET FILES:SOURCE FILES
     ACTION LINE       [tab]ACTION LINE(S)
     ```
     ```

GDB

- GDB stands for the **GNU debugger**.
  - It supports Assembly, C, C++, Fortran, Go, Objective-C, Pascal, Rust and *others*.
  - GDB also has an inbuilt python interpreter.

- Compile using the **-g flag**.
  - This stops the compiler from optimizing and preserves debugging information.
- Run gdb using
  - gdb a.out
  - gdb a.out --tui (my recommendation)

GDB (basic commands)

- run r
- break <line number>/<function name> b
- next n
- list (not necessary if we start in tui mode)
- quit q
- print p
- info locals / breakpoints / frame

GDB (debugging commands)

- continue
- up/down
- backtrace bt
- step
- set <variable id>
- watch <variable id>
- what <variable>
- display <variable>/ undisplay <variable id>

Valgrind

What is valgrind?

It is a tool for checking memory management and threading bugs.

How to use valgrind?

valgrind ./a.out

Let's look at some live demos!