(Lab 10) Programming across files, Makefiles, Unit Testing

CS2013 Systems Programming

Department of CSE, IIT Palakkad

IIT Palakkad

Oct 09, 2025

Quiz 9 (15 minutes, **Do not copy the question**)

1. In a 64bit machine, consider the following code

   ```
   char* array = malloc(sizeof(char)* 10);
   ```

   Write down the value obtained on running `sizeof(array)` in bytes. Also give a short justification for your answer.

2. Is the following definition of `main()` to get command line arguments correct ?

   ```
   int main(int argc, char* argv){
        ...
   }
   ```

   If yes, justify. Otherwise, write the corrected code.

3. True or False: A doubly linked list allows traversing the list only in one direction. Justify your answer.

Plan

- Programming in multiple files
- Testing multi-file programs
- Compilation for multi-file programs using `make`.
- Lab exercise - Implementing python lists as library (`realloc()`)

Programming in multiple files

- Simple program (demo)

Programming in multiple files

- Simple program (demo) Steps: (1) Compile and (2) Link

Programming in multiple files

- Simple program (demo) Steps: (1) Compile and (2) Link
- **Step 1.** Compiling induvidual files:
  - Use `-c` flag to **compile** induvidual file
  - Example: `gcc -Wall -c add.c -o add.o`
  - Example: `gcc -Wall -c main.c -o main.o`
  - Declare functions not in the file

Programming in multiple files

- Simple program (demo) Steps: (1) Compile and (2) Link
- **Step 1.** Compiling induvidual files:
  - Use `-c` flag to **compile** induvidual file
  - Example: `gcc -Wall -c add.c -o add.o`
  - Example: `gcc -Wall -c main.c -o main.o`
  - Declare functions not in the file
- **Step 2.** Link the object files
  - Use without `-c` flag
  - Example: `gcc add.o main.o -o main`

Programming in multiple files

- Simple program (demo) Steps: (1) Compile and (2) Link
- **Step 1.** Compiling induvidual files:
  - Use `-c` flag to **compile** induvidual file
  - Example: `gcc -Wall -c add.c -o add.o`
  - Example: `gcc -Wall -c main.c -o main.o`
  - Declare functions not in the file
- **Step 2.** Link the object files
  - Use without `-c` flag
  - Example: `gcc add.o main.o -o main`
- Task of linker: ensure ...
  - ...no missing symbols
  - ...all symbols are defined only once
  - ...`main` symbol is included

Need for header files

- Want to include `add()` function in multiple places

Need for header files

- Want to include `add()` function in multiple places
- One solution: copy paste code !
    - Issue: Modifying `add()` – difficult !

Need for header files

- Want to include `add()` function in multiple places
- One solution: copy paste code !
  - Issue: Modifying `add()` – difficult !
- Preferred solution: Use header file (Demo)

Need for header files

- Want to include `add()` function in multiple places
- One solution: copy paste code !
  - Issue: Modifying `add()` – difficult !
- Preferred solution: Use header file (Demo)
- Issue due to multiple inclusion

```
#ifndef MYADD_H
#define MYADD_H


...
#endif
```

- Adding ifndef endif block is very standard ! Should start doing it automatically !

Why write in multiple files ?

- Separation of concerns
- Allow for code reuse (without copy-paste)
- Makes code managable (all edits in one place)
- Ability to make changes and test faster (Why ? Will see next)

Testing multi-file program

- Done via **Unit tests**

Testing multi-file program

- Done via **Unit tests**
- Why ?

Testing multi-file program

- Done via **Unit tests**
- Why ?
  - NASA Mars Climate Orbiter:
    - 127 lines of code lacked unit testing . . .

Testing multi-file program

- Done via **Unit tests**
- Why ?
  - NASA Mars Climate Orbiter:
    - 127 lines of code lacked unit testing . . .
    - $327 million satellite burned up in the Mars atmosphere !

Testing multi-file program

- Done via **Unit tests**
- Why ?
  - NASA Mars Climate Orbiter:
    - 127 lines of code lacked unit testing . . .
    - $327 million satellite burned up in the Mars atmosphere !
  - Ariane 5 Flight 501:
    - Unchecked 64-bit integer overflow . . .

Testing multi-file program

- Done via **Unit tests**
- Why ?
  - NASA Mars Climate Orbiter:
    - 127 lines of code lacked unit testing . . .
    - $327 million satellite burned up in the Mars atmosphere !
  - Ariane 5 Flight 501:
    - Unchecked 64-bit integer overflow . . .
    - rocket self-destruct 40 seconds after launch, (lost $500 million payload) !

Testing multi-file program

- Done via **Unit tests**
- Why ?
  - NASA Mars Climate Orbiter:
    - 127 lines of code lacked unit testing . . .
    - $327 million satellite burned up in the Mars atmosphere !
  - Ariane 5 Flight 501:
    - Unchecked 64-bit integer overflow . . .
    - rocket self-destruct 40 seconds after launch, (lost $500 million payload) !
  - Therac-25 Radiation Therapy Machine
    - Poor unit testing . . .

Testing multi-file program

- Done via **Unit tests**
- Why ?
  - NASA Mars Climate Orbiter:
    - 127 lines of code lacked unit testing ...
    - $327 million satellite burned up in the Mars atmosphere !
  - Ariane 5 Flight 501:
    - Unchecked 64-bit integer overflow ...
    - rocket self-destruct 40 seconds after launch, (lost $500 million payload) !
  - Therac-25 Radiation Therapy Machine
    - Poor unit testing ...
    - 5 people dying from massive overdoses !
- Demo

Compiling multi-file programs

- Issue with current compilation ?
  - To many compilations !
  - Need to remember what changed !
  - Recompile correct files !

Compiling multi-file programs

- Issue with current compilation ?
  - To many compilations !
  - Need to remember what changed !
  - Recompile correct files !
- One way: just dump all and compile ! (Demo)

Compiling multi-file programs

- Issue with current compilation ?
  - To many compilations !
  - Need to remember what changed !
  - Recompile correct files !
- One way: just dump all and compile ! (Demo)
- Issue: **Compilation time $\gg$ Time to generating executable**

Compiling multi-file programs

- Issue with current compilation ?
  - To many compilations !
  - Need to remember what changed !
  - Recompile correct files !
- One way: just dump all and compile ! (Demo)
- Issue: **Compilation time $\gg$ Time to generating executable**
- Compiles files not modified also !
- More reasons ? (CSE: Will see in Compiler course)

Compiling multi-file programs

- Issue with current compilation ?
  - To many compilations !
  - Need to remember what changed !
  - Recompile correct files !
- One way: just dump all and compile ! (Demo)
- Issue: **Compilation time $\gg$ Time to generating executable**
- Compiles files not modified also !
- More reasons ? (CSE: Will see in Compiler course)
- Solution ? Write rules in `Makefile` and run `make`.

Using Makefile

Steps

- Split programs to parts
- Obtain the dependency diagram !
- Write rules to compile them separately, produce compiled object files
- Produce the executable from the object files

Advantage

- Smaller parts can be tested well (Unit tests)
- Modifying one part ? Compile only the change (and not all files)

Demo for make

- Dependency diagram

```
myadd.c---------------- main.c
        \___ mymod.c___/
```

- Demo

Demo for make

- Dependency diagram

```
myadd.c---------------- main.c
        \___ mymod.c___/
```

- Demo
- Summary
  - Use rule names same as file names generated
  - `clean` and `all` rule
  - `.PHONY` for rules not files

Lab Ex1. Linked lists as library

- Splitting into files
- Writing unit test cases
- Writing makefile
- **Suggestion** - Complete the exercise from Lab 09 and then do this !

Realloc

- Did `malloc(10*sizeof(int))`.
- Now want space for 20 integers !
- What do we do ?

Realloc

- Did `malloc(10*sizeof(int))`.
- Now want space for 20 integers !
- What do we do ?
- One way, allocate 20 integers, then copy to new memory location and deallocate old memory!

Realloc

- Did `malloc(10*sizeof(int))`.
- Now want space for 20 integers !
- What do we do ?
- One way, allocate 20 integers, then copy to new memory location and deallocate old memory!
- Too much work ! Any easy way ??

Realloc

- Did `malloc(10*sizeof(int))`.
- Now want space for 20 integers !
- What do we do ?
- One way, allocate 20 integers, then copy to new memory location and deallocate old memory!
- Too much work ! Any easy way ??
- Just do `realloc()` :)
- **Note**: `realloc()` is costly. Use it sparingly !
- Demo

Lab Ex 3, 4: Python Lists and Big Integer

- Ex. 3 Goal: Implement list functionality in python.
- List to grow as more data is stored. (Hint: double memory using `realloc()`)
- List to shrink when data is removed. (Hint: halve memory using `realloc()`)
- Library does alloc and dealloc.
- Library user need not worry about it !
- Ex. 4 Goal: Implement python addition
- Support addition of arbitrary number of digits !

Quick Summary

- Writing code in multiple files
- Unit testing
- Using makefile to build and test
- The need and use of `realloc()`.

Lab Exercise

Questions (Do the following in your repo)

- Do `$ git switch lab09`. Commit all the changes.
- Do `$ git push -u origin lab09`

Lab Exercise

Questions (Do the following in your repo)

- Do `$ git switch lab09`. Commit all the changes.
- Do `$ git push -u origin lab09`
- Do `$ git fetch && git merge`
- Do `$ git switch lab10` to see the `questions.md` in `lab10` folder.
- **To push changes: do `$ git push -u origin lab10`**

Class repo (for in-class demo)

- Accessible via
  - `git clone git@gitserver:class_repo`
- To see latest changes, cd to the class_repo and do
  - `git fetch && git merge`

Class has ended

- No more pushes to `gitserver`.
- Complete the exercises during off-lab hours.

Humble Request

Please keep the chairs in position before you leave.
(as a token of respect for our CFET staff)