(Lab 9) C Programming - Files, GDB (debug), Valgrind (mem leaks), Linked lists

CS2013 Systems Programming

Department of CSE, IIT Palakkad

IIT Palakkad

Sep 26, 2025

Quiz 8 (15 minutes, **Do not copy the question**)

1. Write a command to compile `phaltoo.c` displaying all compiler warnings, and create an output named `phaltoo`.

2. Variables declared in a .c file gets allocated in the memory of a program by the C compiler. Is it always allocated in stack or always in heap ? Justify your answer.

3. Function `void branch_change()` modifies the roll number of the student given as argument to reflect branch change.

```
struct student {
    long rollno ;
    ....
};
```

Write an appropriate function *declaration* for such a `branch_change` function.

Plan (Demo)

- Dynamic memory allocation details (two dim array)
- File Handling
- Print buffer and handling them
- Debugging using GDB
- Memory leak detection using Valgrind
- Singly and Doubly linked lists

File handling

- Operating on files – same interface

  (OS independent wrappers)

  - `fopen()`, `fprintf()`, `fscanf()`, `fclose()`
  - Appear in section 3 of man
  - Example: man 3 fopen

  (Linux system calls)

  - `open()`, `write()`, `read()`, `close()`
  - Appear in section 2 of man
  - Example: man 2 open

File handling

- Operating on files – same interface

  (OS independent wrappers)

  - `fopen()`, `fprintf()`, `fscanf()`, `fclose()`
  - Appear in section 3 of man
  - Example: man 3 fopen

  (Linux system calls)

  - `open()`, `write()`, `read()`, `close()`
  - Appear in section 2 of man
  - Example: man 2 open

- Program to read, write using files (Demo).

Summary (File handling)

- FILE * file - file pointer
- `fopen` - open file for read/write/creation. Returns file pointer
- `fscanf` - same as scanf, first arg is file pointer
- `fprintf` - same as printf, first arg is file pointer
- `fclose` - closes the file, necessary for OS book keeping
- `feof` - has end of file reached
- Add error handling code – always help !

Print buffer

- Buffering - print initiated only when it is filled or `\n` seen.

Print buffer

- Buffering - print initiated only when it is filled or `\n` seen.
- Program to show non-standard print order (Demo).

Print buffer

- Buffering - print initiated only when it is filled or `\n` seen.
- Program to show non-standard print order (Demo).
- **Summary**
  - `stdout` is buffered. Prints may not come in the order.
  - Can be a cause for confusion !

Print buffer

- Buffering - print initiated only when it is filled or `\n` seen.
- Program to show non-standard print order (Demo).
- **Summary**
  - `stdout` is buffered. Prints may not come in the order.
  - Can be a cause for confusion !
- Workaround
  1. Use fprintf to `stderr` (which is not buffered) instead of `printf`
  2. Still want to use `printf` ? Use `fflush(stdout)`
  3. Debug with printf ? Bad idea !
- Wrap code around `#ifndef ... #endif` to easily enable/disable debug

Debugging using GDB

Basic principles

1. Clearly know what your program is supposed to do.
2. Detect when it does not happen.
3. Fix the issue.

- Avoid temptation to skip step 1, by arbitrarily tweaking the code.
- Do not guess what could be going wrong.
- Allow the computer to show you what is going inside.

Memory leak detection

- Scenarios: Program compiles - no error, runs - no errors
- However, it can leak memory !
- Valgrind - Useful in detecting memory leaks
- Illustration of usage (Demo)

Valgrind summary

- *definitely lost*: Program leaking memory. Fix it!
- *indirectly lost*: Program crashed. Couldn't clean up memory.
- *suppressed*: Can be ignored (memory not managed by program)
- *possibly lost*: Program leaking memory. Check pointers !

Linked lists (Main exercise)

Quick Summary

- Dynamic memory allocation for 2 dim arrays
- File handling + handling possible errors
- Issues with printf and Debugging using GDB
- Memory leak detection using Valgrind
- Linked list

Push existing code

- Do `$ git switch lab08`. Commit all the changes.
- Do `$ git push -u origin lab08`.

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 lab09` to see the `questions.md` in `lab09` folder.
- **To push changes: do `$ git push -u origin lab09`**

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)