CS240 Comprehensive Review

Theo Park

3 May 2022

1 Compiling and Linking

1.1 Gcc Flags

- *-c* Compile file into object file
- *-g* Debugging symbols
- ***-Wall*** Include ALL Warning
- ***-Werror*** Turn wanings into errors
- ***-O1, -O2, -O3*** Optimize output code
- ***-o filename*** Output to filename
- *-ANSI* Adhere to ANSI std
- *-std=C99* Adhere to C99 std

1.2 Linking

Object file contains binary code, symbol tables, and is a compiled form of a C module. To make it a complete executable, one must link object files, with one of them containing main().

2 File I/O

2.1 Essentials

- FILE *fopen(char *file_name, char *mode);
  Modes are "r", "w", and "a" (append). Returns file ptr on success, NULL on unsuccess, so one must check the return val of fopen().
- int fclose(FILE *file_pointer);
  It does not set the file ptr to NULL, so you have to manually set it to NULL. Return val check isn't necessary in this class.
- **int fprinf(FILE *stream, const char *format, . . .);**
- **int fscanf(FILE *stream, const char *format, . . .);**
- int access(char *file_name, int mode);
  Used to check if file can be accessed in "R_OK", "W_OK", or "F_OK" (check for existence) mode.
- int feof(FILE *file_pointer);
  Returns non-zero if EOF reached.
- int ferror(FILE *file_pointer);
  Returns 0 if error occurs (e.g disk space full).

2.2 Notes with fscanf()

- Utilize %[] (%[0-9A-z] %[Â-z])
- **Field width specifier (e.g %49s %49[A-z]). Always one less than the buffer size to account for NUL terminator.**
- Assigns variables to pointers; use & symbol for non-strings.
- Returns number of successfully read variables; Check for error using the return value.

2.3 Random Access File I/O

- **int ftell(FILE *file_pointer);** Returns current offset from the beginning of the file (SEEK_SET) or -1 in case of error.
- **int fseek(FILE *fp, long int offset, int whence);**
  Whence values include
  - SEEK_SET: Offset relative to beginning of the file
  - SEEK_CUR: Offset relative to the current position
  - SEEK_END: Offset relative to the end of the file
- Example of finding how long the file is:
  ```
  fseek(fp, 0, SEEK_END);
  int len = ftell(fp);
  fseek(fp, 0, SEEK_SET);
  ```

3 Struct and Typedef

3.1 Some Syntax

- Typedef and struct definition:
  ```
  typedef struct my_data {
          int age;
  } my_data_t;
  ```
- Struct definition and declaration:
  ```
  struct my_data {
          int age;
  } my_var = { 19 };
  ```

3.2 Declaration vs Definition

Declaration is announcing the properties of var (no memory allocation), definition is allocating storages for a var.

- Declaration:
  ```
  struct my_data {
          int age;
  };
  ```
- Definition and initialization:
  ```
  struct my_data my_var = { 19 };
  ```

Put pure declaration (struct, func prototype, extern) outside of the func, put definition inside func.

3.3 Arrays in Struct