FILE I/O

MAKE VS CMAKE

- Similar in that both are used to help with compilation of project
- CMake is a cross-platform tool to manage build process
  - Generates build scripts
  - Works with Make, IDEs, Ninja, ...
  - These native tools then used to actually build the project

FILE POINTERS

- Files are really just streams
- All file I/O functions are in `<stdio.h>` (like stdin/stdout functions)
- General file I/O process:
    1. Create file pointer
    2. Open file with correct mode
    3. Read / edit file
    4. Close file

CREATING FILE HANDLE

```
FILE *fin, *fout;
fin = fopen("in_fname", "r");
fout = fopen("out_fname", "w");
fclose(fin);
fclose(fout);
```

OPENING FILES

- General: `FILE *fp = fopen(filename, mode);`
- Modes available:
  - "r" = read
  - "w" = write
  - "a" = append
- Can also add "+" to any to make make read and write (or append and read)
- Can add "b" (e.g. "rb" for binary) - whether or not this makes a difference is system dependent

OPENING FILES (CONT.)

CLOSING FILES

- Actually important
- File I/O is typically buffered, results only written out to the file at certain points
- Closing will make sure the buffer is cleared at the end
    - May work out fine if you don't (but not worth the risk)

READING FROM FILES

- Similar to reading from stdin
- `fscanf(fileptr, formatstr, memaddr1, memaddr2, ...)`
  - `formatstr` and `memaddr1` same as with `scanf`
- `fgets(char *s, int size, FILE *stream)`
  - used `stdin` for the file pointer in the past
  - to use file, just use `FILE*` returned when opening file
- `getline(char **lineptr, size_t *n,`

`FILE *stream)`

- just use `FILE*` other than stdin

OUTPUTTING TO FILES

- `fprintf(FILE *stream, formatstr, other_args)`
  - like printf, but specify file to output to
- `fputc(int c, FILE *stream)`
  - write a single character
  - corresponding `int fgetc(FILE *stream)` to read single character
- `fputs(char *s, FILE *stream)`
  - write string to stream

READING / WRITING BLOCKS

- `fread(void *ptr, size_t size, size_t nitems, FILE *stream)`
  - reads `nitems` objects each `size` bytes long from `stream` into `ptr`
- `fwrite(void *ptr, size_t size, size_t nitems, FILE *stream)`
  - writes `nitem` objects each `size` bytes long to `stream` from `ptr`

IMPORTANT NOTES

- Pay attention to return values
  - Functions treat errors and EOF in different ways
- If you switch between reading and writing in the samme file pointer
  - Must use `fseek()` or `fflush()` before switching
  - Forces buffer to be flushed