CSE 142 Programming I

File I/O

What is a File

- Persistent storage
  - ➤Memory goes away when the program ends (or the computer crashes, etc.)
  - ➤Files must be actively deleted

Why Not Always Use Files

- Slow
  - ➤Thousands (hundreds of thousands?) of times slower than memory
- Sequential access
  - ➤It's hard to jump around to find the data we're looking for

Why Would We Use Files?

- We often have to store data to use later
  - ➤text files
  - ➤programs
  - ➤web pages
  - ➤images
  - ➤etc.

Basic File Operations

- Open a file
- Write to a file
- Read from a file
- Close a file

(There's more, but this is enough for now)

#include?

- Everything we need to use files is in stdio.h
  - ➤In fact, we've been using files all along—more about this later

Opening a File

- "Opening" a file makes a connection between the file (on disk) and your program
- Once the file is open, we can read or write to it

The FILE Structure

- There's a special structure called FILE which contains information for the computer to access your file
- You never have to look inside one of these, just pass them to functions

Opening a File

```
int main(void){
  FILE *input;

  input = fopen("foo.txt", "r");
  if (input == NULL) {
     printf("Couldn't open the file!\n");
     exit -1;
  }
  ...
}
```

A FILE pointer

fopen()

- fopen() takes a filename and a mode string as arguments
  - ➤The filename says which string to open
  - ➤The mode strings tells the computer what you are going to do with the file
- fopen() returns a pointer to a FILE structure

Mode Strings

- "r" means read
- "w" means write
- "a" means append (write at the end)

NULL

- NULL is a special pointer that does not point to anything
- If fopen() returns NULL, then we know that there was an error opening the file—it didn't work

Closing a File

- Closing a file removes the connection between the file on disk, and your program
  - ➤If you do not close your files, you may lose changes that were made to them!

Closing Our File

- To close our the file associated with the FILE * input, we just need to write:

```
fclose(input);
```

- After this command we can no longer access the file!

I/O and Files

- We've been doing file I/O all along!
- C has two special files: stdin and stdout
- stdin is a file that reads from the keyboard
- stdout is a file that writes to the console

Writing to a File

```
FILE *output;

output = fopen("bar.txt", "w");
if (output == NULL) exit(-1);

fprintf(output, "Hello!");
```


fprintf()

- Works exactly like printf, except it has an argument telling which file to print to
- If we specify **`stdout`** as the file, then it is the same as using printf

```
printf("Apple\n");
fprintf(stdout, "Apple\n");
```


Reading from a File

```
FILE *input;
int x;

input = fopen("foo.txt", "w");
if (input == NULL) exit(-1);

fscanf(input, "%d", &x);
```

fscanf()

- Works exactly like scanf, except it has an argument telling which file to print to
- If we specify **`stdin`** as the file, then it is the same as using scanf

```
scanf("%d", &age);
fscanf(stdin, "%d", &age);
```


End of File (EOF)

- fscanf returns the number of items assigned (just like scanf) unless it reaches the end of the file
- If fscanf reaches the end of the file, then it returns the constant EOF
  - ➤We'll see how to use this

File I/O Application

- Copy a file.
- What do we need to do?
  - ➤Get filenames from the user
  - ➤Open the files
  - ➤Copy one file to the other
  - ➤Close the files

The Copy Function

```
void fcopy(FILE *to, FILE *from){
   char temp;
   int status;

   status = fscanf(from, "%c", &temp);
   while(status != EOF){
      fprintf(to, "%c", temp);
      status = fscanf(from, "%c", &temp);
   }

   return;
}
```


The Rest of the Program

```
int main(void){
  char infile[100];
  char outfile[100];
  FILE *in, *out;

  printf("Copy from: ");
  scanf("%s", infile);

  printf("Copy to: ");
  scanf("%s", outfile);
   in = fopen(infile, "r");
  out = fopen(outfile, "w");
    if ((in == NULL)
     || (out == NULL)){
       printf("Ack!\n");
       exit(-1);
    }

    fcopy(out, in);
    fclose(out);
    fclose(in);

    return 0;
}
```