File Pointers

File Pointers

- The ability to read data from and write data to files is the primary means of storing **persistent data**, data that does not disappear when your program stops running.

- The abstraction of files that C provides is implemented in a data structure known as a `FILE`.
  - Almost universally when working with files, we will be using pointers to them, `FILE*`.

File Pointers

- The file manipulation functions all live in `stdio.h`.
  - All of them accept `FILE*` as one of their parameters, except for the function `fopen()`, which is used to get a file pointer in the first place.

- Some of the most common file input/output (I/O) functions that we'll be working with are:

`fopen()`	`fclose()`	`fgetc()`	`fputc()`	`fread()`	`fwrite()`

File Pointers

- `fopen()`
  - Opens a file and returns a file pointer to it.
  - Always check the return value to make sure you don't get back NULL.

```
FILE* ptr = fopen(<filename>, <operation>);
```

File Pointers

- `fopen()`
  - Opens a file and returns a file pointer to it.
  - Always check the return value to make sure you don’t get back NULL.

```
FILE* ptr1 = fopen(“file1.txt”, “r”);
```

File Pointers

- `fopen()`
  - Opens a file and returns a file pointer to it.
  - Always check the return value to make sure you don’t get back NULL.

```
FILE* ptr2 = fopen(“file2.txt”, “w”);
```

File Pointers

- `fopen()`
  - Opens a file and returns a file pointer to it.
  - Always check the return value to make sure you don’t get back `NULL`.

```
FILE* ptr3 = fopen(“file3.txt”, “a”);
```

File Pointers

- `fclose()`
  - Closes the file pointed to by the given file pointer.

```
fclose(<file pointer>);
```

File Pointers

- `fclose()`
  - Closes the file pointed to by the given file pointer.

```
fclose(ptr1);
```

File Pointers

- `fgetc()`
  - Reads and returns the next character from the file pointed to.
  - Note: The operation of the file pointer passed in as a parameter must be “r” for read, or you will suffer an error.

```
char ch = fgetc(<file pointer>);
```

File Pointers

- `fgetc()`
  - Reads and returns the next character from the file pointed to.
  - Note: The operation of the file pointer passed in as a parameter must be “r” for read, or you will suffer an error.

```
char ch = fgetc(ptr1);
```

File Pointers

- The ability to get single characters from files, if wrapped in a loop, means we could read all the characters from a file and print them to the screen, one-by-one, essentially.

```
char ch;
while((ch = fgetc(ptr)) != EOF)
   printf(“%c”, ch);
```

- We might put this in a file called `cat.c`, after the Linux command “`cat`” which essentially does just this.

File Pointers

- The ability to get single characters from files, if wrapped in a loop, means we could read all the characters from a file and print them to the screen, one-by-one, essentially.

```
char ch;
while((ch = fgetc(ptr)) != EOF)
   printf("%c", ch);
```

- We might put this in a file called `cat.c`, after the Linux command "`cat`" which essentially does just this.

File Pointers

- The ability to get single characters from files, if wrapped in a loop, means we could read all the characters from a file and print them to the screen, one-by-one, essentially.

```
char ch;
while((ch = fgetc(ptr)) != EOF)
   printf(“%c”, ch);
```

- We might put this in a file called `cat.c`, after the Linux command “`cat`” which essentially does just this.

File Pointers

- `fputc()`
  - Writes or appends the specified character to the pointed-to file.
  - Note: The operation of the file pointer passed in as a parameter must be “`w`” for write or “a” for append, or you will suffer an error.

```
fputc(<character>, <file pointer>);
```

File Pointers

- `fputc()`
  - Writes or appends the specified character to the pointed-to file.
  - Note: The operation of the file pointer passed in as a parameter must be “`w`” for write or “a” for append, or you will suffer an error.

```
fputc(‘A’, ptr2);
```

File Pointers

- `fputc()`
  - Writes or appends the specified character to the pointed-to file.
  - Note: The operation of the file pointer passed in as a parameter must be “w” for write or “a” for append, or you will suffer an error.

```
fputc(‘!’, ptr3);
```

File Pointers

- Now we can read characters from files and write characters to them. Let’s extend our previous example to copy one file to another, instead of printing to the screen.

```
char ch;
while((ch = fgetc(ptr)) != EOF)
   printf(“%c”, ch);
```

File Pointers

- Now we can read characters from files and write characters to them. Let’s extend our previous example to copy one file to another, instead of printing to the screen.

```
char ch;
while((ch = fgetc(ptr)) != EOF)
   fputc(ch, ptr2);
```

- We might put this in a file called `cp.c`, after the Linux command “cp” which essentially does just this.

File Pointers

- `fread()`
  - Reads `<qty>` units of size `<size>` from the file pointed to and stores them in memory in a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be "r" for read, or you will suffer an error.

```
fread(<buffer>, <size>, <qty>, <file pointer>);
```

File Pointers

- `fread()`
  - Reads `<qty>` units of size `<size>` from the file pointed to and stores them in memory in a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be “r” for read, or you will suffer an error.

```
int arr[10];
fread(arr, sizeof(int), 10, ptr);
```

File Pointers

- `fread()`
  - Reads `<qty>` units of size `<size>` from the file pointed to and stores them in memory in a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be “r” for read, or you will suffer an error.

```
double* arr2 = malloc(sizeof(double) * 80);
fread(arr2, sizeof(double), 80, ptr);
```

File Pointers

- `fread()`
  - Reads `<qty>` units of size `<size>` from the file pointed to and stores them in memory in a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be “r” for read, or you will suffer an error.

```
char c;
fread(&c, sizeof(char), 1, ptr);
```

File Pointers

- `fread()`
  - Reads `<qty>` units of size `<size>` from the file pointed to and stores them in memory in a buffer (usually an array) **pointed to by `<buffer>`.**
  - Note: The operation of the file pointer passed in as a parameter must be "r" for read, or you will suffer an error.

```
char c;
fread(&c, sizeof(char), 1, ptr);
```

File Pointers

- `fwrite()`
  - Writes `<qty>` units of size `<size>` to the file pointed to by reading them from a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be “`w`” for write or “`a`” for append, or you will suffer an error.

```
fwrite(<buffer>, <size>, <qty>, <file pointer>);
```

File Pointers

- `fwrite()`
  - Writes `<qty>` units of size `<size>` to the file pointed to by reading them from a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be “w” for write or “a” for append, or you will suffer an error.

```
int arr[10];
fwrite(arr, sizeof(int), 10, ptr);
```

File Pointers

- `fwrite()`
  - Writes `<qty>` units of size `<size>` to the file pointed to by reading them from a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be “w” for write or “a” for append, or you will suffer an error.

```
double* arr2 = malloc(sizeof(double) * 80);
fwrite(arr2, sizeof(double), 80, ptr);
```

File Pointers

- `fwrite()`
  - Writes `<qty>` units of size `<size>` to the file pointed to by reading them from a buffer (usually an array) pointed to by `<buffer>`.
  - Note: The operation of the file pointer passed in as a parameter must be "`w`" for write or "`a`" for append, or you will suffer an error.

```
char c;
fwrite(&c, sizeof(char), 1, ptr);
```

File Pointers

- `fwrite()`
  - Writes `<qty>` units of size `<size>` to the file pointed to by reading them from a buffer (usually an array) **pointed to by `<buffer>`**.
  - Note: The operation of the file pointer passed in as a parameter must be “w” for write or “a” for append, or you will suffer an error.

```
char c;
fwrite(&c, sizeof(char), 1, ptr);
```

File Pointers

- Lots of other useful functions abound in stdio.h for you to work with. Here are some of the ones you may find useful!

File Pointers

- Lots of other useful functions abound in stdio.h for you to work with. Here are some of the ones you may find useful!

Function	Description
`fgets()`	Reads a full string from a file.
`fputs()`	Writes a full string to a file.
`fprintf()`	Writes a formatted string to a file.
`fseek()`	Allows you rewind or fast-forward within a file.
`ftell()`	Tells you at what (byte) position you are at within a file.
`feof()`	Tells you whether you’ve read to the end of a file.
`ferror()`	Indicates whether an error has occurred in working with a file.