File Handling

- Structure FILE is defined in `stdio.h`
- Used for permanent retention of data
- Store files on secondary storage devices
- Data files created, updated and processed by C
- C views file as sequential stream of bytes
- File processing: sequential-access and random-access

Streams

- Stream of communication between files and programs
- Portable way for reading an writing data
  - Files, Printer, Monitor
- Normal streams go to `stdin`, `stdout`, `stderr`
- Data structure FILE represents all streams
- Open to access, Declare a pointer, Close file
- Files end with an end-of-file marker

Data Reading and Writing Character

- `int fgetc(FILE *file);`
- `int getchar(void);`
- `int fputc(int char, FILE *file);`
- `int putchar(int char);`

Data Reading and Writing String

- `char *fgets(char *str, int n, FILE *file);`
- `int fputs(const char *str, FILE *file);`
- `int fscanf(FILE *file, const char *format, ...);`
- `int fprintf(FILE *file, const char *str, ...);`

File Functions

- `FILE *fopen(const char *name, const char *mode);`
- `int fflush(FILE *stream);`
- `int feof(FILE *name);`
- `int ferror(FILE *stream);`
- `void clearerr(FILE *stream);`
- `int fileno(FILE *stream);`
- `int fclose(FILE *name);`

Access Mode for Files

Mode	Description
r	Open existing file for reading.
w	Create a file for writing. If the file exists, discard the current contents.
a	Open or create a file for writing at the end of the file. Write operations append data to the file.
r+	Open existing file for update (reading and writing).
w+	Create a file for reading and writing. If the file exists, discard the current contents.
a+	Open or create a file for reading and updating; all writing is done at the end of the file. Write operations append data to the file.
rb	Open en existing file for reading in binary mode.

Also : wb, ab, rb+, wb+, ab+

Opening a File

```
FILE *file;
/* declare a stream and prototype fopen */
file = fopen("myfile.dat","r");  // read a file

// Error checking file has opened correctly
if ( (stream = fopen( "myfile.dat","r")) == NULL) {
    printf("Can't open %s\n", "myfile.dat");
    exit(1);
}
// or
if ( (stream = fopen( "myfile.dat","r")) != NULL)
{ /* read the file as you intended */ }
else {
    printf("Can't open %s\n", "myfile.dat");
    exit(1);
}
```

```
char *string[80]
FILE *stream;
if ((stream = fopen(...)) != NULL)
{ fscanf(stream,"%s", string); }

fprintf(stderr,"Cannot Compute!!\n");

fprintf(stdout,"Enter a string : ");
fscanf(stdin,"%s",string);

printf("Enter a string : ");
scanf("%s",string);

while (!feof(fp))
{ fscanf(fp,"%s",line); }

int ch;
ch = getchar();
```

Create and enter user inputted data in a txt file

```c
#include <stdio.h>

int main () {
  FILE *clientPtr;
  if ((clientPtr = fopen("client.txt", "w")) != NULL) {
    unsigned int account;
    char name[30];
    double balance;

    printf("Enter the Acct, Name, and Balance.\n");
    printf("Enter EOF to end input\n");
    printf("%s", "? ");
    scanf("%d%29s%lf", &account, name, &balance);
```

```
    while (!feof(stdin)) {
      fprintf(clientPtr, "%d %s %.2f\n",
       account, name, balance);
      printf("%s", "? ");
      scanf("%d%29s%lf", &account, name, &balance);
    }
    fclose(clientPtr);
  }
  else
  { printf("File will not open!!"); }

  return 0;
}
```

- eof : `<ctrl> d` or `<crtl> z`
- Closing a file can free resources

Create and enter user inputted data in a txt file

```
#include <stdio.h>

int main () {
  FILE *clientPtr;
  if ((clientPtr = fopen("client.txt", "r")) != NULL) {
    unsigned int account;
    char name[30];
    double balance;

    printf("%-10s%-13s%s\n", "Account", "Name",
     "Balance");
    fscanf(clientPtr,"%d%29s%lf", &account, name,
     &balance);
```

```
    while (!feof(stdin)) {
      printf("%-10s%-13s%s\n", "Account", "Name",
       "Balance");
      fscanf(clientPtr,"%d%29s%lf", &account, name,
       &balance);
    }
    fclose(clientPtr);
  }
  else
  { printf("File will not open!!"); }

  return 0;
}
```

- eof : `<ctrl> d` or `<crtl> z`
- Closing a file can free resources

Resetting File Position Pointer

- Retrieve data sequentially
  - Start reading from the beginning
  - Stop at desired data or end of file
- May want to process the data several times
- `void rewind(FILE *stream);` -- point to beginning
- File Position Pointer (not a pointer)
  - Integer value a byte in the file
  - Known as the file offset
  - Is member of the FILE structure

Questions