EASTERN
WASHINGTON UNIVERSITY
start something **big**

File I/O

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Outline for Today

- Text file Input/output

EASTERN
WASHINGTON UNIVERSITY
start something **big**

File Type

- A file represents a sequence of bytes, does not matter if it is a text file or binary file.
  - In text file, each character is represented with its ASCII code.
    - Human readable.
  - In Binary file, data are represented with a sequence of binary digits.
    - E.g. int I = 10; ➔ 00000000 00000000 00000000 00001010
    - Not Human Readable

File Type

- C programming language provides access on high level functions,
- as well as low level (OS level) calls to handle file on your storage devices.
- Let us first look at the high level functions.
  - We will do low level IO later.

File Open

- You can use the **fopen( )** function to create a new file or to open an existing file, this call will initialize an pointer of the type **FILE.**

  ```
  FILE * fopen( const char * filename, const char * mode );
  ```

  – **filename** is string literal, which you will use to name your file.

  – How about he mode?

File Open

- access **mode** can have one of the following values:

Mode	**Description**
r	Opens an existing text file for reading purpose.
w	Opens a text file for writing, if it does not exist then a new file is created. Here your program will start writing content from the beginning of the file.
a	Opens a text file for writing in appending mode, if it does not exist then a new file is created. Here your program will start appending content in the existing file content.
r+	Opens a text file for reading and writing both.
w+	Opens a text file for reading and writing both. It first truncate the file to zero length if it exists otherwise create the file if it does not exist.
a+	Opens a text file for reading and writing both. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

EASTERN WASHINGTON UNIVERSITY
start something big

File Open

- If you are going to handle binary files then you will use below mentioned access modes instead of the above mentioned:
  - "rb", "wb", "ab", "ab+", "a+b", "wb+", "w+b", "ab+", "a+b”

File Close

- int fclose(FILE *fp);
- The **fclose( )** function returns zero on success, or **EOF** if there is an error in closing the file.
- The EOF is a constant defined in the header file **stdio.h**

File Write

- int fputc(int **c**, FILE ***fp**);
- The function fputc() writes the character value of the argument **c** to the output stream referenced by **fp**.
- It returns the written character written on success otherwise EOF if there is an error.

File Write

- int fputs(const char *s, FILE *fp);
- The function fputs() writes the string **s** to the output stream referenced by **fp**.
- It returns a non-negative value on success, otherwise EOF is returned in case of any error.
- You can use **int fprintf(FILE *fp,const char *format, ...)** function as well to write a string into a file.

File Write

```
// Correction: before the first write IO, test.txt will be truncate to zero.
// Then system appends  the continuous writes to the end of the
// same file, as long as the file keeps open.
#include <stdio.h>

int main()
{
   FILE *fp = NULL;

   fp = fopen("/tmp/test.txt", "w+");
   fprintf(fp, "This is testing for fprintf...\n");
   fputs("This is testing for fputs...\n", fp);
   fclose(fp);

   return 0;
}
```

File Read

- Following is the simplest function to read a single character from a file:
- int fgetc(FILE * fp);
- The fgetc() function reads a character from the input file referenced by fp.
- The return value is the character read, or in case of any error it returns EOF.

File Read

- char *fgets(char *buf, int n, FILE *fp);
- It stops when either **(n-1)** characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.
  - The read-in line includes the newline feed.
- It copies the read string into the buffer **buf**, appending a null character to terminate the string.

File Read

- If fgets() encounters a newline character '\n' or the end of the file EOF before they have read the maximum number of characters,
  - then it returns only the characters read up to that point including new line character.
- You can also use
  - int fscanf(FILE *fp, const char *format, ...)
  - to read strings from a file but it stops reading after the first space character encounters.

File Read

```
#include <stdio.h>
main()
{
   FILE *fp;
   char buff[255];
   fp = fopen("/tmp/test.txt", "r");
   fscanf(fp, "%s", buff);
   printf("1 : %s\n", buff );

   fgets(buff, 255, (FILE*)fp);
   printf("2: %s\n", buff );

   fgets(buff, 255, (FILE*)fp);
   printf("3: %s\n", buff );
   fclose(fp);
}
```

Binary File I/O

- There are following two functions, which can be used for binary input and output:

```
size_t fread(void *ptr, size_t size_of_elements,
             size_t number_of_elements, FILE *a_file);
size_t fwrite(const void *ptr, size_t size_of_elements,
             size_t number_of_elements, FILE *a_file);
```

- Both of these functions should be used to read or write blocks of memories - usually arrays or structures.

Summary

  - FILE pointer, FILE *fp;
  - How to read and write from or to a text file?
- Next Class
  - Structures