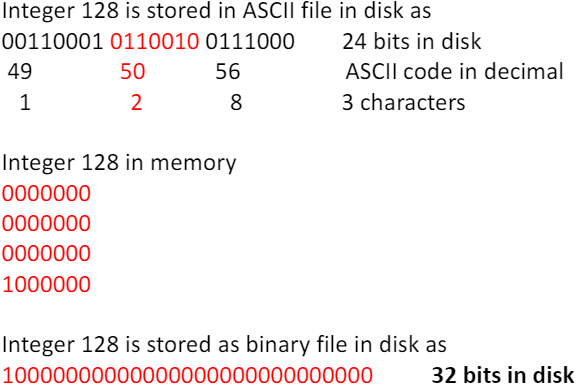
Lecture 11 File IO

**Concept**

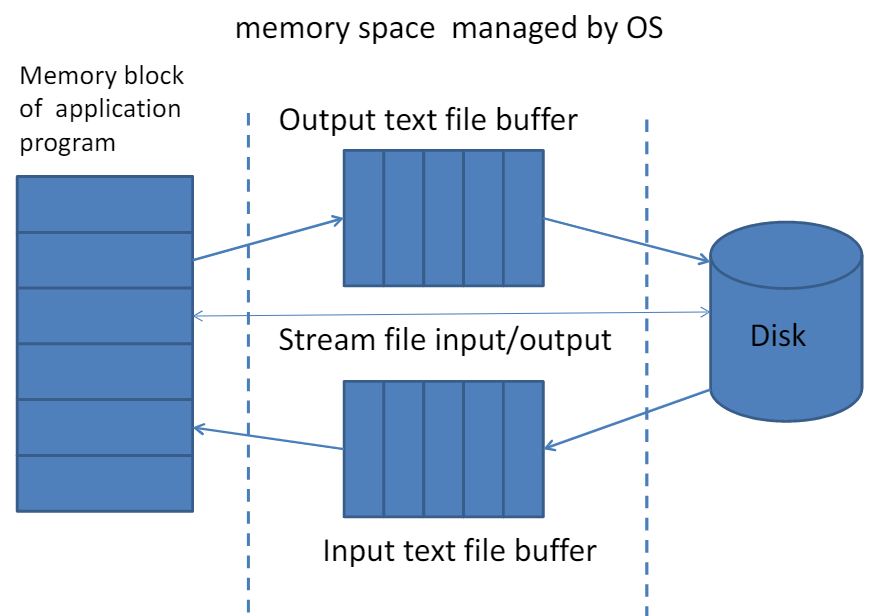
* File output is to save data from memory to files in disk or other media for long term storage
* File input is to open files in disk and load file data into memory for processing
* File I/O and operating system
  + File IO involves kernel operations of operating system. Storing data as file in disk and accessing file data stored in disk needs to access ad control disk, which is managed by OS
  + OS provides services for file IO as **system calls**
  + C’s stdio library provides functions for file IO, which are implemented using the OS’s system calls.

**File storage formats**

* There are two types of file storage formats, ASCII file and Binary file
  + The ASCII (or text) file stores data as a sequence of characters
  + The binary files store data as it is in memory



* Saving int 128 in ASCII file needs to convert int 128 to char sequence ‘1’, ‘2’, ‘8’ and then save the sequence to a file. When inputting char sequence ‘1’, ‘2’, ‘8’ to memory as int 128, it needs converting the ‘1’, ‘2’, ‘8’ to int 128.
* Saving int 128 in binary file dumps the memory chunk to disk as stream, and inputting directly read the stream to memory without converting.



**File IO in C programming**

* C provides library functions for file IO in stdio.h
* All stdio file IO functions use the following FILE structure (defined in stdio.h) to hold the information and pointers of input/output buffer of an IO file

Typedef struct {

Int level; //file/empty level of buffer

Unsigned flags; //file status flags

Char fd; //file descriptor

Unsigned char hold; // ungetc char if no buffer

Int bsize; // buffer size

Unsigned char \*buffer // data transfer buffer

Unsigned char \*curp //current active pointer

Usinged istemp; //temporary file indicaor

Short token; //used for validity checking

} FILE

**Fopen() and fclose() functions**

* FILE \*fopen(const char \*filename, const char \*mode);
* This function requests OS to create a stream buffer, and a FILE structure with info of the I.O buffer. It returns the address of FILE structure, or NULL of the operations fail.
* Mode:
  + R or rb -open file for reading
  + W or wb -open file for writing
  + A or ab -open file for appending
  + R+ or rb+ or r+b – open file for update (reading and writing)
  + W+ or wb+ or w+b – truncate to zero length or create file for update
  + A+ or ab+ or a+b – open or create file for update, writing at end of file
* Int fclose(FILE \*stream) closes the stream buffer, returns zero if the stream is successfully closed. On failure, EOF (-1) is returned.

**File writing example**

#include <stdio.h>

Int main(){

FILE \*fp = fopen(“test.txt”, “w”);

If (fp==NULL){

Perror(“error opening file”); //printing error message

Return 0;

}

Int x=10;

Fprintf(fp, “hello\n”); // write text to a file

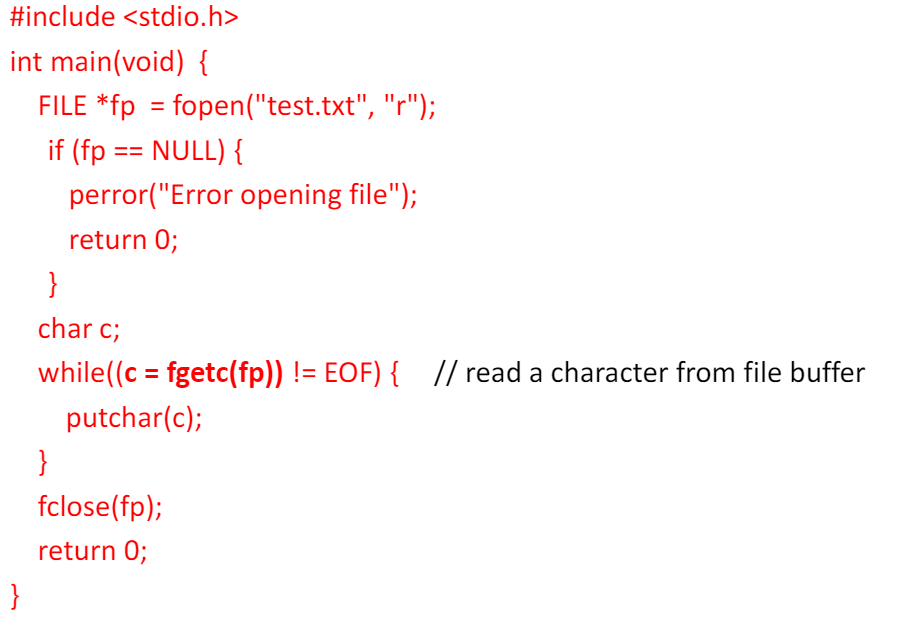
Fprintf(fp, “int x=%d\n”,x); //formatted write

Fclose(fp);

Return 0;

}

**File reading example**



**File IO functions**

* File position controls
  + Int fseek(FILE \*stream, long offset, int whence)
  + Whence is once of three macros
    - SEEK\_SET: offset is relative to the beginning of the file
    - SEEK\_CUR: offset is relative to the current file pointer position
    - SEEK\_END: offset is relative to the end of the file
  + This function resets the file position indicator

Example

Fseek(fp,100,SEEK\_SET);// seek the 100th byte of the file

Fseek(fp,-10,SEEK\_CUR);//seek backward 10 bytes from the current position

Fseek(fp,-10,SEEK\_END);// seek the 10th byte before the end of the file

Void rewind (FILE \*stream);

* This function resets the file position indicator to the beginning of the file
* Rewind (fp); // to the beginning of the opened file fp

**File read**

* Int fgetc(FILE \*stream)
  + This reads and returns one character of reading
* Char \*fgets(char \*buffer, int n, FILE \*stream)
  + This reads n bytes to char array buffer, returns the pointer of string
* Size\_t fread (void \*buffer, size\_t n, size\_t m, FILE \*stream)
  + This reads n\*m bytes to char array buffer, returns number of byte of successful reading
* Int fscan(FILE \*stream, const char \*format, location address)
  + This is formatted reading, returns the number of successful reading

**Example: read a line by fgets()**

#include <stdio.h>

Int main(){

Char buf[100];

FILE \*fp = fopen(“test.txt”, “r”);

If(fp==NULL){

Perror(“error opening file”);

Return 0;

}

Fgets(buf, sizeof(buf), fp); // read a line

Puts(buf);

Fclose(fp);

Return 0;

}

**Example: read all lines by fgets()**

#include <stdio.h>

Int main(int argc, char \*argv[]){

FILE \*fp;

Char buf [128];

If ((fp=fopen(“test.txt”, “r”))==NULL){

Perror(“error opening file”);

Return 0;

}

While (!feof(fp)){ //read line by line till end of file

If(fgets(buf, sizeof(buf), fp))

Printf(“%s”, buf);

}

Fclose(fp);

Return 0;

}

