FILE HANDLING IN C

CS F111: COMPUTER PROGRAMMING

DR. NIKUMANI CHOUDHURY

BITS PILANI HYDERABAD CAMPUS

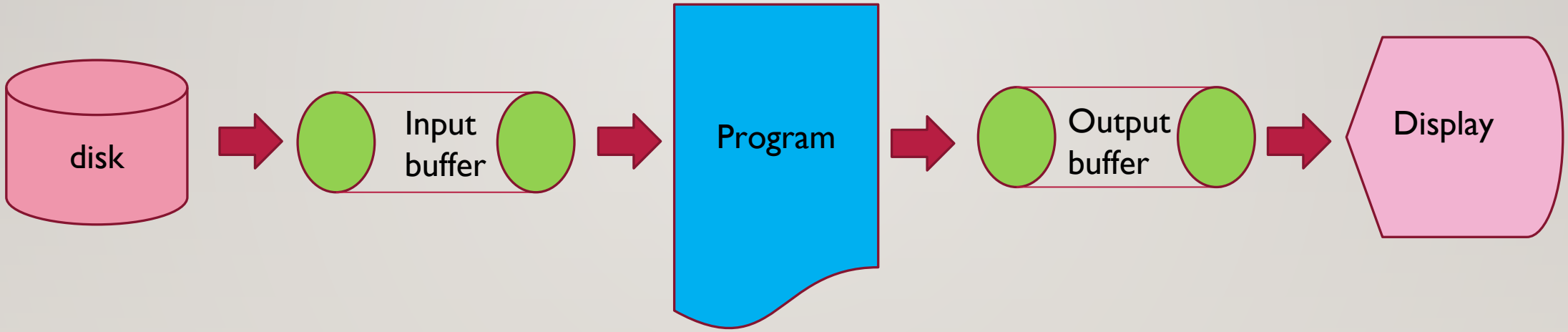
WHAT IS A FILE?

- A file is nothing but a source of storing related data in the form of a sequence of bytes on the disk/ tape.
- Analogy: Assuming that this lecture recording is not available, what some of you might have done? (Reusability)
- Types: Text (.txt) files (characters) and Binary files (integers/ floats/complex types) (.bin)
- Different ways of treating: Eg. '\n' is converted to cr+lf before being written to a text file on the disk, A text file has EOF as the last character, where as binary flie does not have.

FILE HANDLING FUNCTIONS IN C

- Creating a new file: fopen()
- Opening an existing file: fopen()
- Closing a file: fclose()
- Reading a character from a file: getc()
- Writing a character to a file: putc()
- Reading formatted input from file:fscanf()
- Writing formatted data to file: fprintf()
- Reading an integral value from a file: getw()
- Writing an integral value to a file: putw()
- Setting a desired position: fseek(), fsetpos()
- Getting the current position: ftell(), fgetpos()
- Setting the position at the beginning: rewind()
- Finding end of file: feof()
- R/Wing with Binary files: fread(), and fwrite()

THE FILE BUFFERS



FILE STRUCTURE

- C uses a structure called FILE (defined in stdio.h) to store the attributes of a file.
- File-pointer:
  - Is a pointer variable which can store the address of a structure.
- Declaration of a file pointer (fp):

  FILE *fp;

- fp will point to beginning of FILE struct.

```
typedef struct {
int level; /* fill or empty buffer */
unsigned flags; /* File status flags */
char fd; /* File descriptor */
unsigned char hold; /* if no buffer */
int bsize; /* Buffer size */
unsigned char *buffer; /*Data trans. buffer */
unsigned char *curp; /* Current active pointer */
unsigned istemp; /* Temporary file indicator */
short token; /* Used for validity checking */
}FILE;
```

(Attributes of a File)

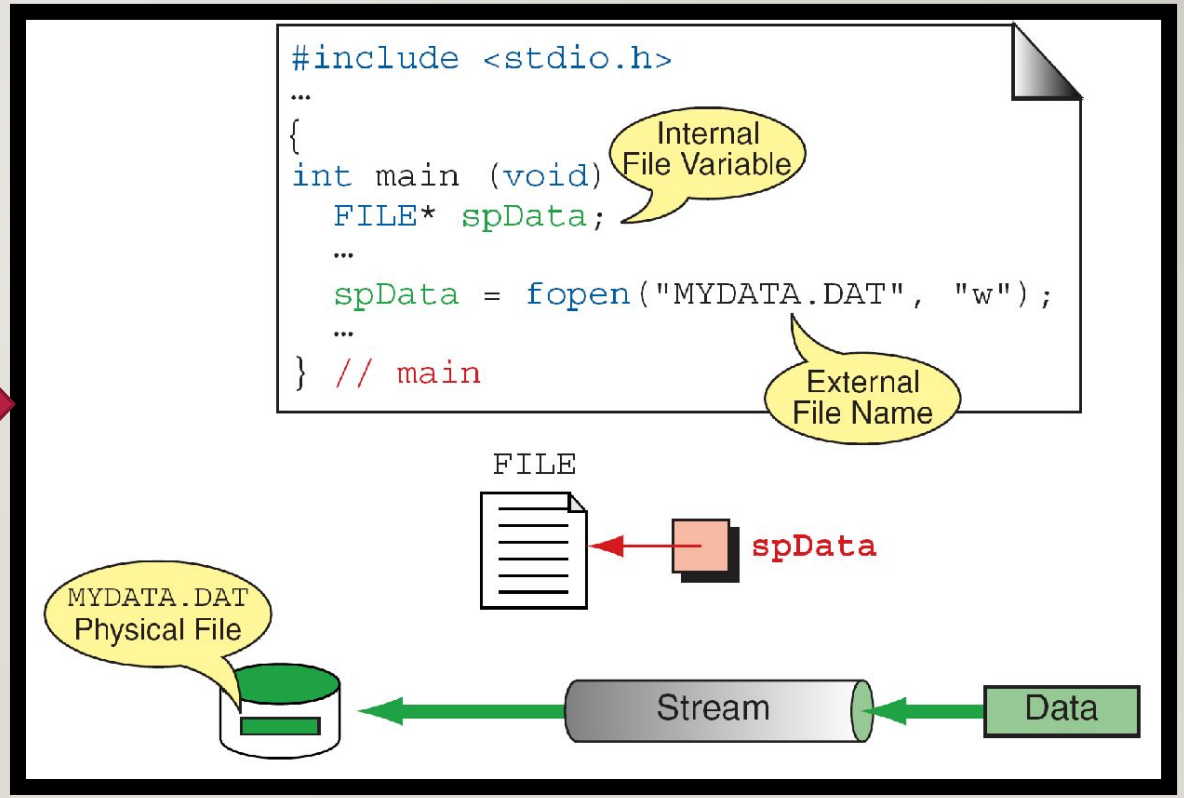
STEPS IN PROCESSING A FILE

1. Create the stream via a pointer variable using the FILE structure: FILE *p;

2. Open the file, associating the stream name with the file name.

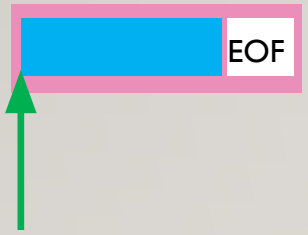
3. Read or write the data.

4. Close the file.



FILE OPENING MODES

FILE *fp;
fp=fopen("aaa", "r");

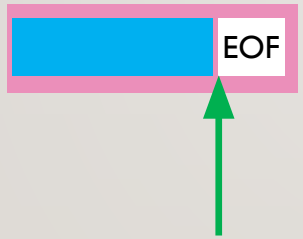


"curp" positioned at the beginning

FILE *fp;
fp=fopen("bbb", "w");

"curp" positioned at the beginning

FILE *fp;
fp=fopen("ccc", "a");



"curp" positioned at the end of the file

r	Open an existing file for reading.
w	Create a file for writing. If the file already exists, *discard* the current contents.
a	Open or create a file for writing at the end of the file—i.e., write operations *append* data to the file.
r+	Open an existing file for update (reading and writing).
w+	Create a file for reading and writing. If the file already 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—i.e., write operations *append* data to the file.
rb	Open an existing file for reading in binary mode.
wb	Create a file for writing in binary mode. If the file already exists, discard the current contents.
ab	Append: open or create a file for writing at the end of the file in binary mode.
rb+	Open an existing file for update (reading and writing) in binary mode.
wb+	Create a file for update in binary mode. If the file already exists, discard the current contents.
ab+	Append: open or create a file for update in binary mode; writing is done at the end of the file.

READING & WRITING ON A FILE

```c
#include <stdio.h>

int main() {

    FILE *fp;  char ch;

   fp = fopen ("file.txt", "w");

   printf ("Enter a string");

   while ((ch =getchar()) != EOF)

        fputc (ch, fp);

   fclose (fp);

return 0; }
```

main.c | file.txt

```
1
2
3 Computer programming
```

```
Enter a string

Computer programming
```

```
File found

Computer programming
```

```c
#include <stdio.h>
#include <stdlib.h>
int main()  {

   FILE *fp;
   char ch;
   fp = fopen ("file.txt", "r");
   if (fp == NULL) {
       printf ("File not found");
       exit (1);
   }
   else
        printf ("File found");

   while ((ch =fgetc(fp)) != EOF)
       printf ("%c", ch);
   fclose (fp);
return 0;
}
```

APPENDING A FILE

```c
#include <stdio.h>
#include <stdlib.h>
int main() {   FILE *fp;   char ch;
  fp = fopen ("file.txt", "a");
  if (fp == NULL) {
    printf ("File not found");
    exit (1);
  }
```

```c
while ((ch =getchar()) != EOF)
        putc (ch, fp);
fclose (fp);
fp = fopen ("file.txt", "r");
while ((ch = fgetc (fp)) != EOF)
    printf ("%c", ch);
fclose (fp);
return 0;
}
```

main.c | file.txt

```c


Computer programming

Appending to earlier string "computer...
```

```
Appending to earlier string "computer...

Computer programming

Appending to earlier string "computer...
```

READING AND WRITING BINARY FILES

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *fp;
    int a = 34563;    int b, i;
    char k[5] = "Hello", m[5];
    short c[5]={100, 200, 300, 400, 500};  short
d[5];
    if ((fp = fopen ("test", "w+b")) == NULL)    {
        printf ("Error in writing");
        exit (1);    }
    fwrite (&a, sizeof(int), 1, fp);
    fwrite (k, sizeof(char), 5, fp);
    fwrite (c, sizeof(short), 5, fp);
```

```
rewind (fp);
fread(&b, sizeof(int),1, fp);
fread(m, sizeof(char),5, fp);
fread(d, sizeof(short),5, fp);

printf ("%s %d\n", m, b);
for (i = 0; i < 5; i++)
      printf ("%hd ", d[i]);
return 0;
}
```