ECE 220 Computer Systems & Programming

Lecture 16 – File I/O

ECE ILLINOIS

ILLINOIS

Input / Output Streams


```
scanf("%d", &x)
```

I/O Device operates using I/O protocol (such as memory mapped I/O)

In C, we abstract away the I/O details to an I/O function call

Stream Abstraction for I/O

All character-based I/O in C is performed on **text streams**.

A stream is a **sequence of ASCII characters**, such as:

- the sequence of ASCII characters printed to the monitor by a single program
- the sequence of ASCII characters entered by the user during a single program
- the sequence of ASCII characters in a single file

Characters are processed in the order in which they were added to the stream.

- e.g., a program sees input characters in the same order as the user typed them.

Standard Streams:

Input (keyboard) is called **stdin**.

Output (monitor) is called **stdout**.

Error (monitor) is called **stderr**.

Buffering

- Every value that goes into the stream is captured by the low-level OS software and kept in a **buffer** (a small array)

Input Buffering

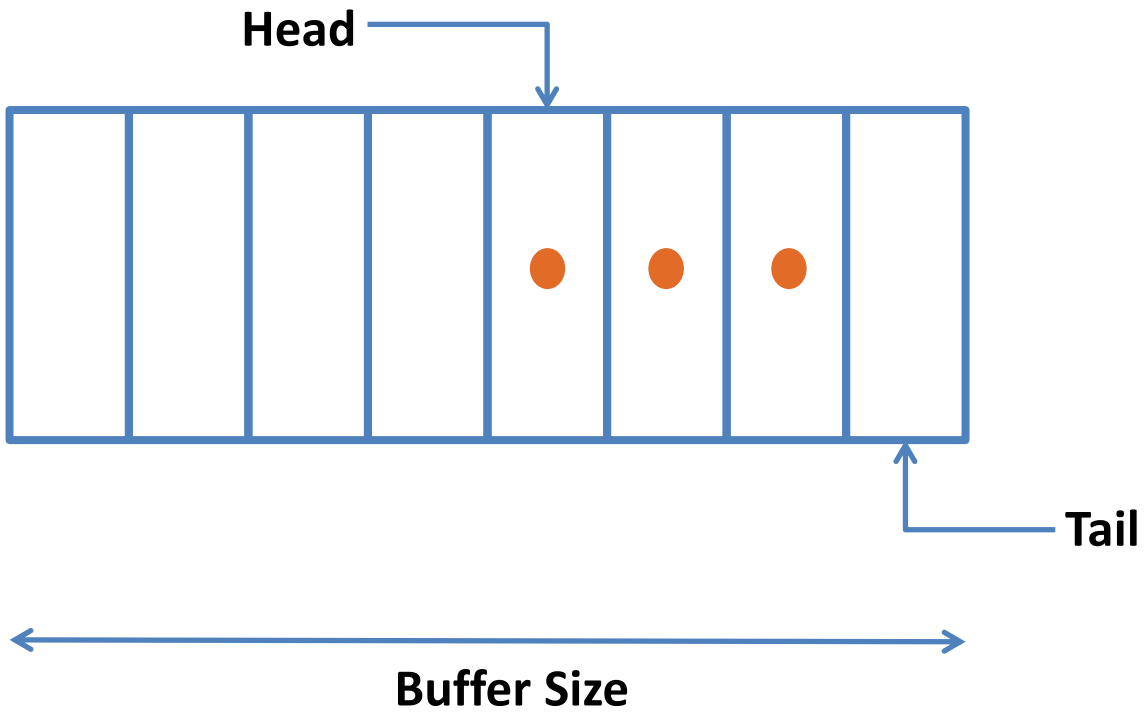
The buffer is released when the user presses Enter key.

Output Buffering

The buffer is released when the program submits a newline character ('\n')

- Buffer allows to decouple the producer from the consumer.

Simple Buffer



- Producer adds data at Tail
- Consumer removes data from Head
- Buffer Full?
- Buffer Empty?
- Concept of circular buffer
- Also called First in, First Out (FIFO) or Queue

Basic I/O Functions

- Creating I/O streams
  - `fopen`: open/create a file for I/O
  - `fclose`: close a file for I/O
- I/O one character at a time
  - `fgetc`: Reads an ASCII character from stream
  - `fputc`: Writes an ASCII character to stream
  - `getchar`: Reads an ASCII character from the keyboard
  - `putchar`: Writes an ASCII character to the monitor
- I/O one line at a time
  - `fgets`: Reads a string (line) from stream
  - `fputs`: Writes a string (line) to stream
- Formatted I/O
  - `fprintf`: Writes a formatted string to stream
  - `fscanf`: Reads a formatted string to stream

Creating I/O stream

```
FILE* fopen(char* filename, char* mode)    //mode: “r”, “w”, “a”, …
```

success-> returns a pointer to FILE

failure-> returns NULL

```
int fclose(FILE* stream)
```

success-> returns 0

failure-> returns EOF (Note: EOF is a macro, commonly -1)

```
FILE *myfile;
myfile = fopen("test.txt", "w");
if(myfile == NULL){
    printf("Cannot open file for write.\n");
    return -1;
}

fclose(myfile);
return 0;
```

I/O one character at a time

```
int fgetc(FILE* stream)
```

success-> returns the next character

failure-> returns EOF and sets end-of-file indicator

```
int fputc(int character, FILE* stream)
```

success-> write the character to file and returns the character written

failure-> returns EOF and sets end-of-file indicator

Formatted I/O

```
int fprintf(FILE* stream, const char* format, …)
```

success-> returns the number of characters written

failure-> returns a negative number

```
int fscanf(FILE* stream, consta char* format, …)
```

success-> returns the number of items read; 0, if pattern doesn't match

failure-> returns EOF

```c
/* File I/O Example */
#include <stdio.h>
int main(){
    FILE *file;
    char buffer[100];

    //
    file = fopen("intro.txt", "w");

    //
    printf("Write a self introduction with less than 100 characters: ");
    fgets(buffer, 100, stdin);

    //
    fputs("Your self introduction: ", file);
    fputs(buffer, file);

    fclose(file);

    //
    fputs(buffer, stdout);

    return 0;
}
```

Exercise: *Read an mxn matrix from file in_matrix.txt and write its transpose to file out_matrix.txt.* ***The first row of the file specifies the size of the matrix.***

Hint: *use fscanf to read from a file and use fprintf to write to a file.*

```
#include <stdio.h>
int main(){
    FILE *in_file;
    FILE *out_file;

    //
    in_file = fopen("in_matrix.txt", "r");
    if(in_file == NULL)
        return -1;

    //
    int m, n;
    fscanf(in_file, "%d %d", &m, &n);
    int matrix[m][n];
```

in_matrix.txt

```
2 3
1 2 3
4 5 6
```

out_matrix.txt

```
3 2
1 4
2 5
3 6
```

```
    //
    out_file = fopen("out_matrix.txt", "w");
    if(out_file == NULL)
        return -1;
    //
    fprintf(out_file, "%d %d\n", n, m);









    return 0;
}
```