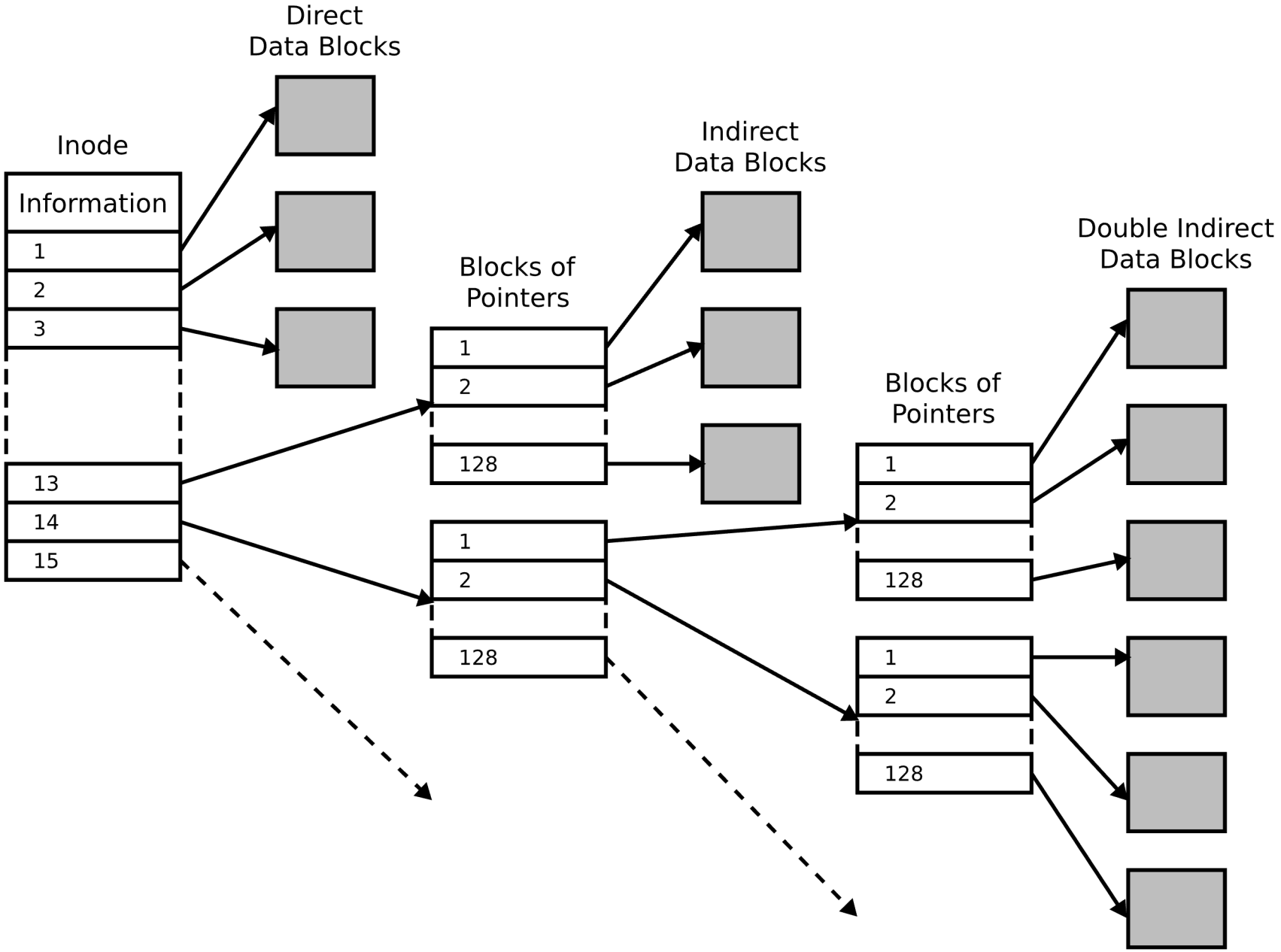
ECE 220 Computer Systems & Programming

File I/O

ECE ILLINOIS

ILLINOIS


Direct
Data Blocks
Inode
Information
1
2
3
13
14
15
Indirect
Data Blocks
Blocks of
Pointers
1
2
128
1
2
128
Double Indirect
Data Blocks
Blocks of
Pointers
1
2
128
1
2
128

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**.

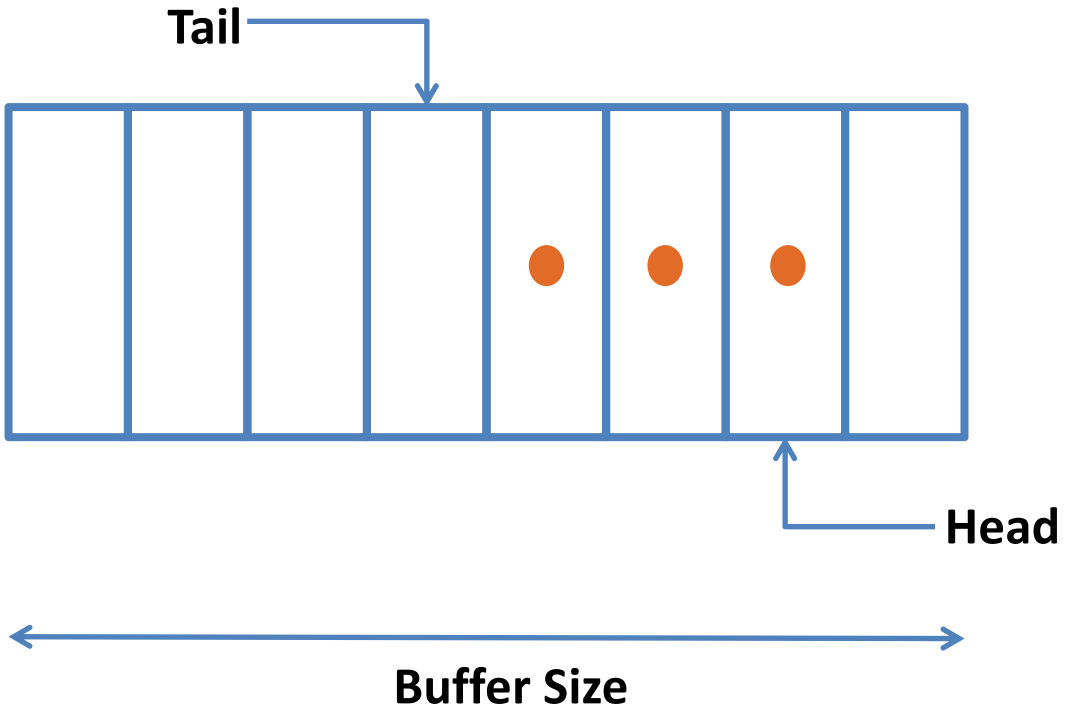
Output (monitor) is called **stdout**.

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

Stream Buffering

- **Input device is the producer; Program is the consumer**
- **We want producer and consumer to be operating independently**
- **Why??? Think Netflix over spotty internet connection**
- **We can accomplish that via buffering**

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

The standard I/O functions are declared in the <stdio.h> header file.

Function	*Description*
putchar	Displays an ASCII character to the screen.
getchar	Reads an ASCII character from the keyboard.
printf	Displays a formatted string.
scanf	Reads a formatted string.
fopen	Open/create a file for I/O.
fclose	Close a file for I/O.
fprintf	Writes a formatted string to a file.
fscanf	Reads a formatted string from a file.
fgetc	Reads next ASCII character from stream.
fputc	Writes an ASCII character to stream.
fgets	Reads a string (line) from stream.
fputs	Writes a string (line) to stream.
EOF & feof	End of file

How to use these I/O functions

FILE* fopen(const char* filename, const 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)

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

success-> returns the number of characters written

failure-> returns a negative number

int fscanf(FILE* stream, const char* format, …)

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

failure-> returns EOF

int fgetc(FILE* stream)

success-> returns the next character

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

int fputc(int c, FILE* stream)

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

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

char* fgets(char* string, int num, FILE* stream)

success-> returns a pointer to string

failure-> returns NULL and sets the end-of-file indicator

int fputs(const char* string, FILE* stream)

success-> writes string to file and returns a non-negative value

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

int feof(FILE* stream) //checks end-of-file indicator

if at the end of file-> returns a non-zero value

if not -> returns 0

int fseek(FILE *stream, long offset, int whence)

success-> 0

failure-> -1

whence: SEEK_SET, SEEK_CUR, SEEK_END

long ftell(FILE *stream)

success-> current offset

failure-> -1

int sprintf(char *str, const char *format, …)

int sscanf(const char *str, const char *format, …)

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;
}
```