

# ECE 220 Computer Systems & Programming

## Lecture 14 – File I/O



# 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.

# Buffered Input

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char inChar1;
6      char inChar2;
7
8      printf("Input character 1:\n");
9      inChar1 = getchar();
10
11     printf("Input character 2:\n");
12     inChar2 = getchar();
13
14     printf("Character 1 is %c\n", inChar1);
15     printf("Character 2 is %c\n", inChar2);
16 }
```

# Buffered output

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main()
5  {
6      putchar('a');
7
8      sleep(5);
9
10     putchar('b');
11     putchar('\n');
12 }
```

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main()
5  {
6      putchar('a');
7      putchar('\n');
8      sleep(5);
9
10     putchar('b');
11     putchar('\n');
12 }
```

# 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

## Write to a file using fputc()

```
1  #include <stdio.h>
2  int main()
3  {
4      int c;
5      FILE *f;
6      /* write to file */
7      f = fopen("out.txt", "w");
8      if (f == NULL)
9      {
10         printf("Unable to open file out.txt for writing\n");
11         return -1;
12     }
13     c = getchar();
14     while (c != '\n')
15     {
16         fputc(c, f);
17         c = getchar();
18     }
19     fclose(f);
```

## Read from a file using fgetc()

```
21  /* read from file */
22  if ((f = fopen("out.txt", "r")) == NULL)
23  {
24      printf("Unable to open file out.txt for reading\n");
25      return -1;
26  }
27  c = fgetc(f);
28  while (c != EOF) /* EOF is a macro defined in stdio.h */
29  {
30      putchar(c);
31      c = fgetc(f);
32  }
33  putchar('\n');
34  fclose(f);
35  return 0;
36 }
```

```
/* 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;
}
```

# 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, const char* format, ...)
```

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

failure-> returns EOF

```
1  #include <stdio.h>
2  int main() {
3      int k;
4      FILE* out;
5      int array[100];
6
7      for(k=0;k<100;++k) {
8          array[k] = k;
9      }
10     out = fopen("data.txt","w");
11
12     for(k=0;k<100;++k) {
13         fprintf(out, "%d ", array[k]);
14     }
15     fclose(out);
16     return 0;
17 }
```

**Exercise:** Read an  $m \times n$  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 *out;

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

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

in\_matrix.txt

<b>2 3</b>
1 2 3
4 5 6



out\_matrix.txt

<b>3 2</b>
1 4
2 5
3 6

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

```
return 0;
```

```
}
```