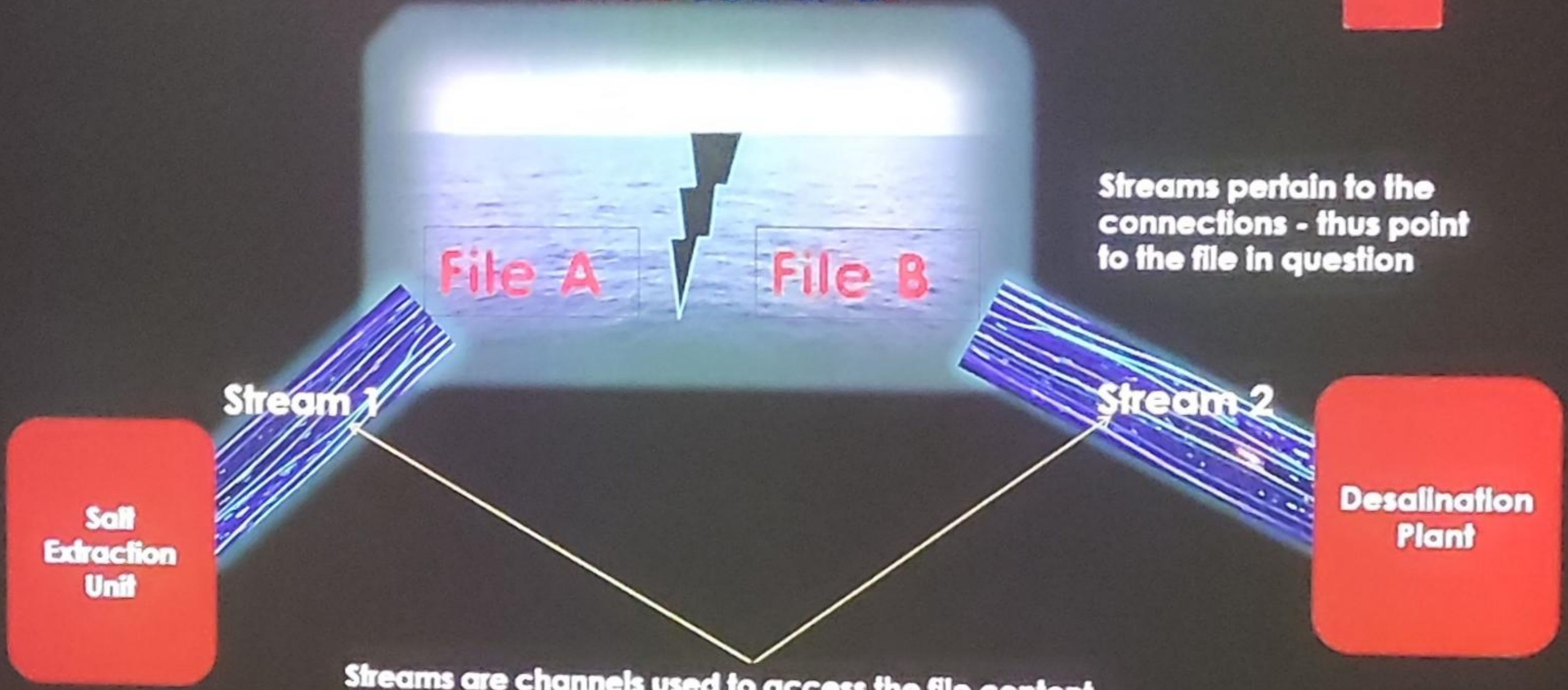
# Files and Streams: An Approximate Analogy In the Sea of Cs



Streams pertain to the connections - thus point to the file in question

Streams are channels used to access the file content

## New functions ahead!

```
FILE * fopen ( const char * filename, const char * mode )
```

Opens the file "filename" (string) and associates it with a stream identified by the FILE pointer

"mode" denotes how the file operations are carried out (writing, reading, appending, etc.)

Returns a FILE pointer if successful else returns NULL

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

Closes the file associated with "stream" and its resources

"stream" is a File pointer

Returns 0 if stream is closed otherwise EOF is returned on failure

End of File (-1 generally)
EOF is present at the end of the file

## Creating a File

```
#include <stdio.h>
int main()
{
FILE *f1, *f2;
f1 = fopen("CS101_attendance.txt", "w");
f2 = fopen("CS101_marks.txt", "w");
FILE *f3 = fopen("CS101_subjects.txt", "w");
fclose(f1);
fclose(f2);
fclose(f3);
}
```

Always close a file after usage!!

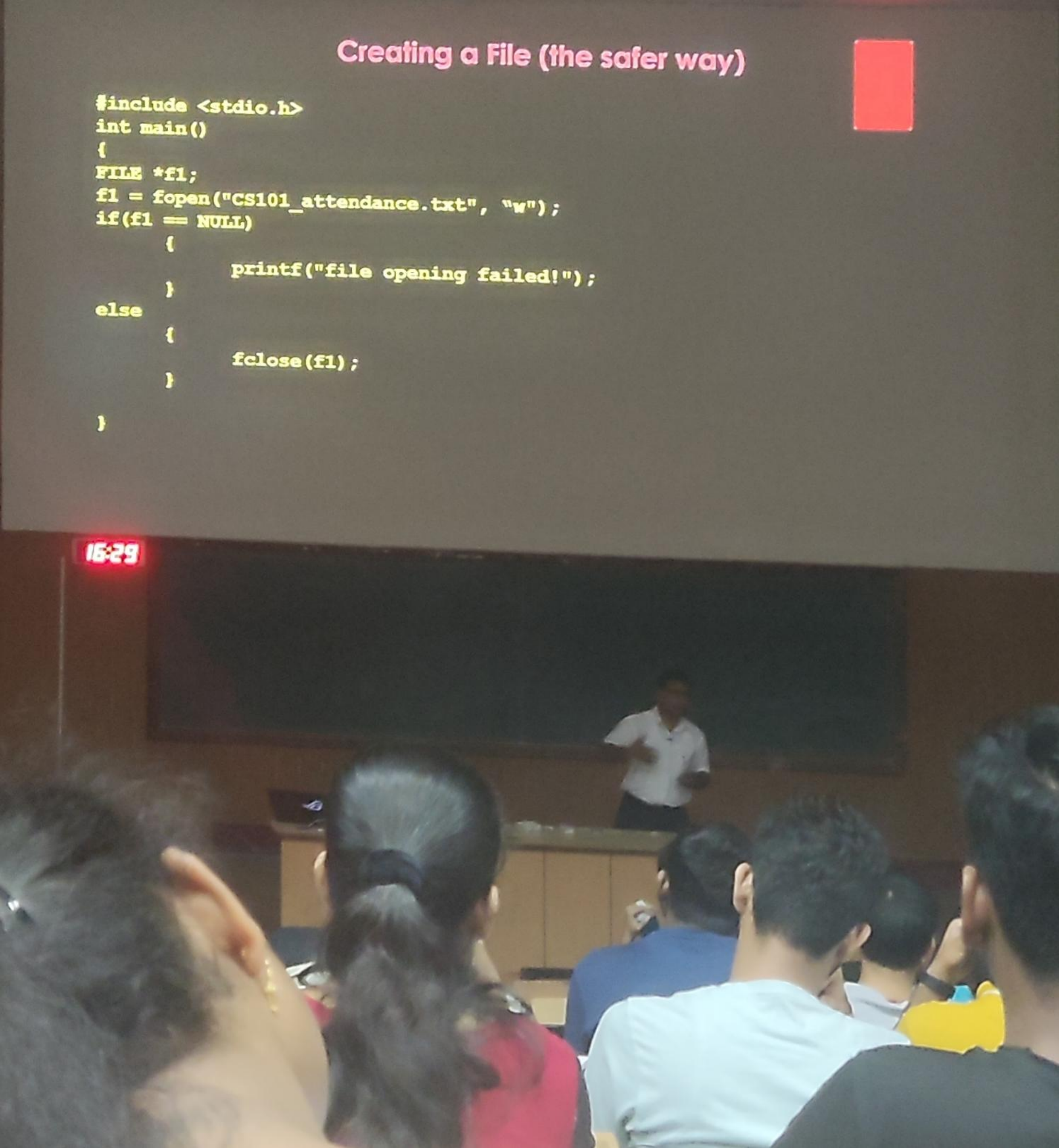
"f1" is a pointer of type "FILE" (a pre-defined structure)
f1 is thus a pointer to a file

"fopen()" opens a file for performing operations
If file is not already present, it creates one
"CS101_attendance.txt" is the name of the file actually created

Always check if the file is successfully opened, by looking for NULL

Opening file in write mode

"fclose()" closes a file. It deallocates the file resources.



## Creating a File (the safer way)

```
#include <stdio.h>
int main()
{
FILE *f1;
f1 = fopen("CS101_attendance.txt", "w");
if(f1 == NULL)
        {
                printf("file opening failed!");
        }
else
        {
                fclose(f1);
        }

}
```

## Creating a File (the safer way)

```
#include <stdio.h>
int main()
{
FILE *f1;
f1 = fopen("CS101_attendance.txt", "w");
if(f1 == NULL)
        {
                printf("file opening failed!");
        }
else
        {
                fclose(f1);
        }

}
```

| | |
|---|---|
| r | Opens existing file for reading. |
| w | Creates an empty file for writing. Overwrites the existing one. |
| a | Opens file for writing at the end. File is created if not present. |
| r+ | Opens existing file for reading and writing. |
| w+ | Opens existing file for reading and writing. Overwrites existing one. |
| a+ | Opens files for reading and writing from/at the end. Repositioning operations can work. |

## Writing to a File

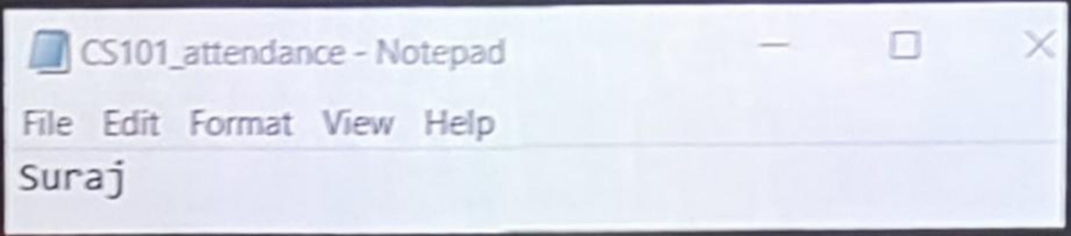
```
#include <stdio.h>
int main()
{
FILE *f1 = fopen("CS101_attendance.txt", "w");
fprintf(f1, "Suraj");
fclose(f1);
}
```

"f1" is a file pointer for the file "CS101_attendance.txt"

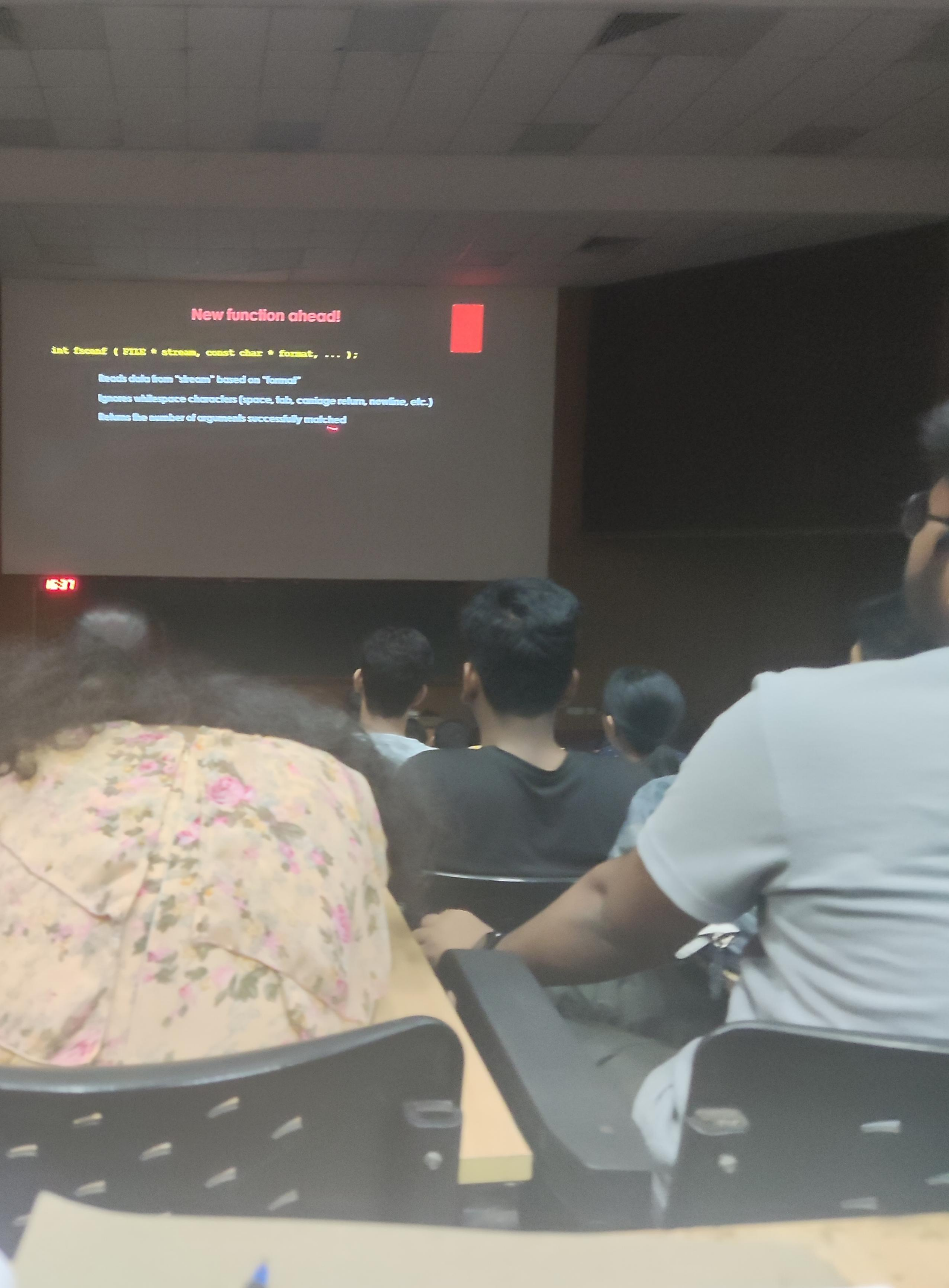
File is opened in write mode

"fprintf()" writes to the file addressed by the file pointer

"Suraj" is written into the file



```
Suraj
```



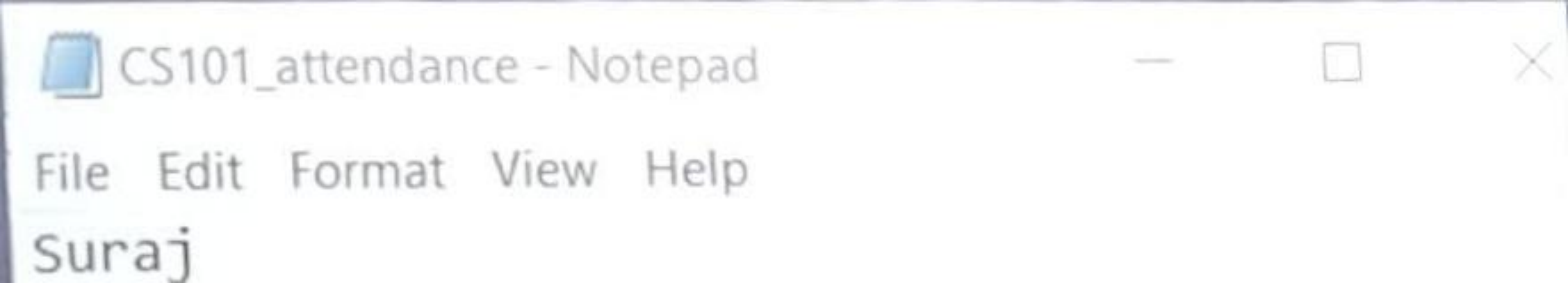
## New function ahead!

```
int fscanf ( FILE * stream, const char * format, ... );
```

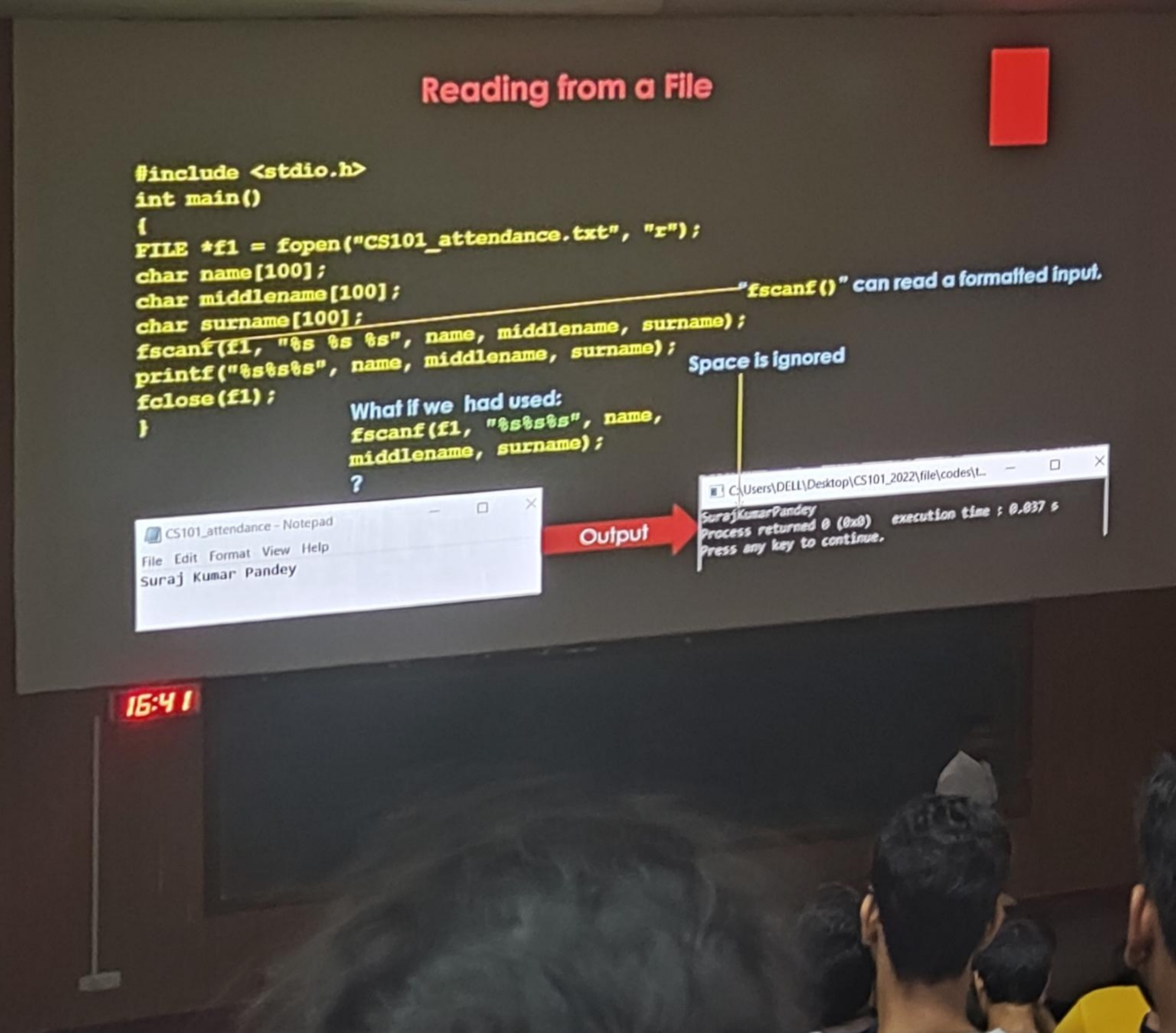
Reads data from "stream" based on "format"

Ignores whitespace characters (space, tab, carriage return, newline, etc.)

Returns the number of arguments successfully matched



```
Suraj
```



## Reading from a File

```c
#include <stdio.h>
int main()
{
FILE *f1 = fopen("CS101_attendance.txt", "r");
char name[100];
char middlename[100];
char surname[100];
fscanf(f1, "%s %s %s", name, middlename, surname);
printf("%s%s%s", name, middlename, surname);
fclose(f1);
}
```

"fscanf()" can read a formatted input.

Space is ignored

What if we had used:
`fscanf(f1, "%s%s%s", name, middlename, surname);`
?

CS101_attendance - Notepad

File Edit Format View Help

```
Suraj Kumar Pandey
```

Output

C:\Users\DELL\Desktop\CS101_2022\file\codes\t...

```
SurajKumarPandey
Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```

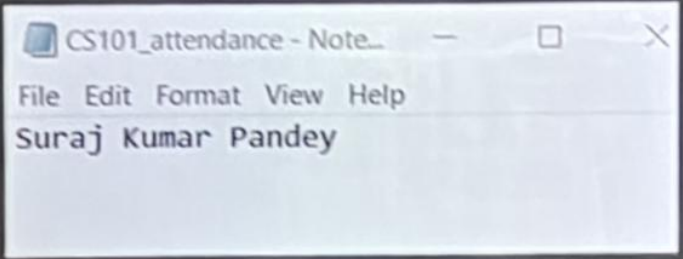
## Reading a character from the file

```
#include <stdio.h>
int main()
{
FILE *f1 = fopen("CS101_attendance.txt", "r");
char ch = NULL;
while(ch != EOF)
    {
        ch = fgetc(f1);
        printf("%c", ch);
    }
fclose(f1);

}
```

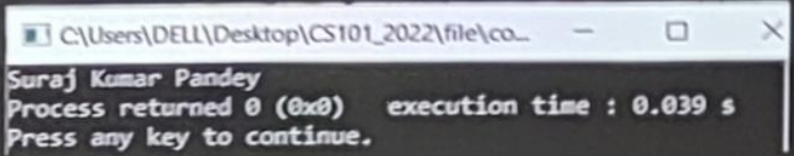
"fgetc()" reads a character from the file

Please note that whitespaces are also read as input here



```
Suraj Kumar Pandey
```

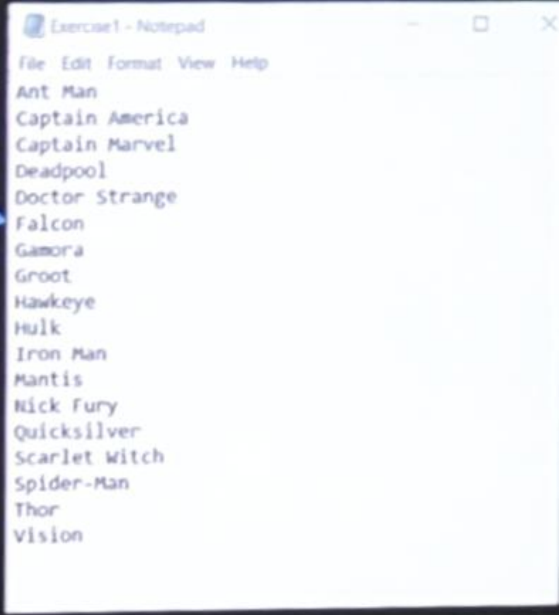
Output



```
Suraj Kumar Pandey
Process returned 0 (0x0)   execution time : 0.039 s
Press any key to continue.
```

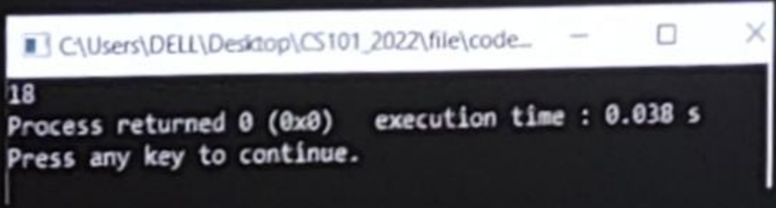
```c
int main()
{
FILE *f1 = fopen("Exercise1.txt", "r");
char ch;
int count = 0;
while(ch != EOF)
      {
            ch = fgetc(f1);
            if(ch == '\n')
            {
                  count++;
            }

      }
printf("%d", count);
fclose(f1);
}
```



```
Ant Man
Captain America
Captain Marvel
Deadpool
Doctor Strange
Falcon
Gamora
Groot
Hawkeye
Hulk
Iron Man
Mantis
Nick Fury
Quicksilver
Scarlet Witch
Spider-Man
Thor
Vision
```

Program Output



```
18
Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

## Exercise 2: What does the following program do?

Input File: Exercise2.txt

```
#include <stdio.h>
int main()
{
FILE *directory = fopen("Exercise2.txt", "r");
char ch = NULL;
while(ch != EOF)
    {

        ch = fgetc(directory);
        if ((ch >= 48) && (ch <= 57))
            {
                printf("%c", ch);
            }
        if( ch == '\n')
            {
                printf("\n");
            }
    }
fclose(directory);

}
```

Program Output

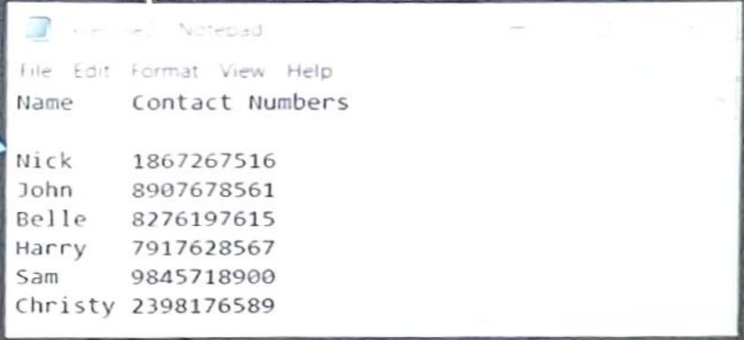
## Exercise 2: What does the following program do?

```c
#include <stdio.h>
int main()
{
FILE *directory = fopen("Exercise2.txt", "r");
char ch = NULL;
while(ch != EOF)
    {

        ch = fgetc(directory);
        if ((ch >= 48) && (ch <= 57))
            {
                printf("%c", ch);
            }
        if( ch == '\n')
            {
                printf("\n");
            }
    }
fclose(directory);

}
```

**Input File: Exercise2.txt**



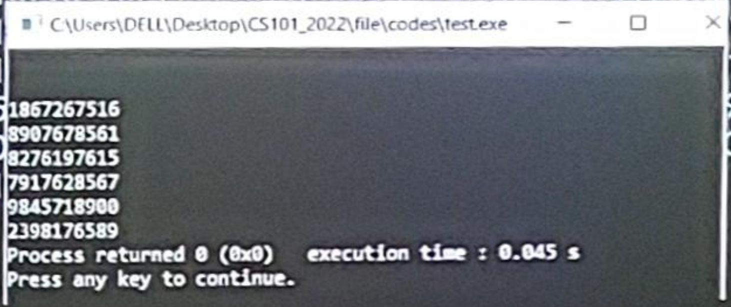
```
Name     Contact Numbers

Nick     1867267516
John     8907678561
Belle    8276197615
Harry    7917628567
Sam      9845718900
Christy  2398176589
```

**Program Output**

**ASCII equivalent of the input file:**

7897109101967111110116979911632781171099810111
41151 11
10411 10
19565 95
55450 06
71041



```
1867267516
8907678561
8276197615
7917628567
9845718900
2398176589
Process returned 0 (0x0)   execution time : 0.045 s
Press any key to continue.
```