<u>FILE INPUT AND OUTPUT:</u> SPECIAL TUTORIAL SESSION 28/10/16

BY:

MAEGHEL PURI

SHIVAM RICHHARIYA

OUTLINE

- WHY FILE I/O
- FILE I/O IN C
- EXAMPLES
- COMMAND LINE

FILE I/O

- Why File I/O ?
- C supports a number of functions that have the ability to perform basic file operations:
- Creating a file
- Opening a file
- Reading data from a file
- Writing data to a file
- Closing a file

FILE I/O

- Declare a file pointer and open a file using the function
- declaration of file pointer :

- FILE *fp ;   // FILE is a type, as int

- FILE *fp1 ;

- opening a file :

- fp = fopen(“filename”,”mode”);

- filename for ex: “one.txt”

FILE I/O

- Basic modes for opening files
- “r” : open an existing file for reading only.
- “w” : open the file for writing only. if the file already exists, it is truncated to zero length.   otherwise a new file is created.
- “a” : open a file for append access.
- More file modes :
- r+ = open file for read/write
- w+ = create file for read/write
- a+ = append text file for read/write

FILE I/O

- Checking the result of fopen() : (example 2.c)

```
if( fp == NULL)
{
        printf("can't open");
        return 0;
}
```

- Closing Files :

- fclose(fp)

FILE I/O

- functions for read files :

  - fscanf()

  - fgets() , example (3.c)

  - fgetc()

- functions for write files :

  - fprintf()

  - fputs()

  - fputc()

FILE I/O

- Formatted reading and writing :

  - fscanf(*filepointer*, "…", *args*)

  - fprintf(*filepointer,* "…", *args*)

- format string and arguments same as with scanf() and printf()

- Code EXAMPLE: (2.c)

FILE I/O

- **Reading from a file using fgets**
- fgets is a better way to read from a file, we can read into a string using fgets

```
file *fptr;
char line [1000];
while (fgets(line,1000, fptr) != null) {
        printf ("read line %s\n", line);
 }
```

- fgets() takes 3 arguments, a string, maximum
- number of characters to read and a file pointer.
- it returns null if there is an error (such as EOF)

FILE I/O

- **main(int argc, char *argv[])**
- the main program in c is called with two implicit : arguments *argc* and *argv*
- argc is an integer value for the number of arguments in the command line

 - if none then argc =1 (the program name)

- argv is an array of strings, passed by reference

 - argv[0] is the name of the program

 - argv[1] is the first command-line argument

 - argv[2] is the next argument, and so on …

FILE I/O

- Command line input / output as for : (**main(int argc, char *argv[])**)

./filename input1 input2   // input1 input2 input3 more…

- Code EXAMPLE : (5.c arg)
- Full Code Examples using Structures (4.c).