CS10003: Programming & Data Structures

Dept. of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Autumn 2020

File Handling

Writing to a file: fprintf( )

- fprintf() works exactly like printf(), except that its first argument is a file pointer. The remaining two arguments are the same as printf
- The behaviour is exactly the same, except that the writing is done on the file instead of the display

```
FILE *fptr;
fptr = fopen ("file.dat","w");
fprintf (fptr, "Hello World!\n");
fprintf (fptr, “%d %d”, a, b);
```

Reading from a file: fscanf( )

- fscanf() works like scanf(), except that its first argument is a file pointer. The remaining two arguments are the same as scanf
- The behaviour is exactly the same, except
  - The reading is done from the file instead of from the keyboard (think as if you typed the same thing in the file as you would in the keyboard for a scanf with the same arguments)
  - The end-of-file for a text file is checked differently (check against special character EOF)

Reading from a file: fscanf( )

```
FILE *fptr;
fptr = fopen (“input.dat”, “r”);
/* Check it's open */
if (fptr == NULL)
  {
    printf(“Error in opening file \n”);
    exit(-1);
  }
fscanf (fptr, “%d %d”,&x, &y);
```

EOF checking in a loop

```
char ch;
while (fscanf(fptr, “%c”,
&ch) != EOF)
{
  /* not end of file; read */
}
```

Reading lines from a file: fgets()

- Takes three parameters
  - a character array str, maximum number of characters to read size, and a file pointer fp
- Reads from the file fp into the array str until any one of these happens
  - No. of characters read = size - 1
  - \n is read (the char \n is added to str)
  - EOF is reached or an error occurs
- '\0' added at end of str if no error
- Returns NULL on error or EOF, otherwise returns pointer to str

Reading lines from a file: fgets()

```
FILE *fptr;
char line[1000];
/* Open file and check it is open */
while (fgets(line,1000,fptr) != NULL)
{
   printf ("Read line %s\n",line);
}
```

Writing lines to a file: fputs()

- Takes two parameters
  - A string str (null terminated) and a file pointer fp
- Writes the string pointed to by str into the file
- Returns non-negative integer on success, EOF on error

Reading/Writing a character: fgetc(), fputc()

- Equivalent of getchar(), putchar() for reading/writing char from/to keyboard
- Exactly same, except that the first parameter is a file pointer
- Equivalent to reading/writing a byte (the char)

```
int fgetc(FILE *fp);
int fputc(int c, FILE *fp);
```

- Example:

```
char c;
c = fgetc(fp1); fputc(c, fp2);
```

Formatted and Un-formatted I/O

- Formatted I/O
  - Using fprintf/fscanf
  - Can specify format strings to directly read as integers, float etc.
- Unformatted I/O
  - Using fgets/fputs/fgetc/fputc
  - No format string to read different data types
  - Need to read as characters and convert explicitly

Closing a file

- Should close a file when no more read/write to a file is needed in the rest of the program
- File is closed using fclose() and the file pointer

```
FILE *fptr;
char filename[]= "myfile.dat";
fptr = fopen (filename,"w");
fprintf (fptr,"Hello World of filing!\n");
…. Any more read/write to myfile.dat….
fclose (fptr);
```

Thank You!