Lecture – 22

Consider the following function prototypes:

```
int feof( FILE *fp);
int ferror( FILE *fp);
```

We can recognize the end of file by using the function **feof(. . .)** which returns non zero if the file associated with *fp* has reached the end of the file. Otherwise it returns zero. The **ferror(. . .)** function returns non zero if the file associated with *fp* has experienced an error; otherwise it returns zero.

C provides four functions which make file operations easier. The first two are called **fputs(. . .)** and **fgets(. . .)** which write a string to and read a string from a file, respectively. Their prototypes are :

```
int fputs( cahr *str, FILE *fp);
char *fgets(char *str, int num, FILE *fp);
```

The **fputs(. . .)** function writes the string pointed to by *str* to the file associated with *fp.* It returns **EOF** if an error occurs and a **non-negative** value if successful. It doesn't automatically append a new carriage-return/linefeed sequence. The **fgets(. . .)** function writes the string pointed to by *str* to the file associated with *fp* into the string pointed to by *str* until (num-1) characters have been read, a new line character is encountered.

Example 1:

```
void main( ){

        FILE *fp1, *fp2;
        char ch[100];



        if( fp1 = fopen("t1.txt", “r”)) = = NULL) {
                printf(“Cannot open file\n”);
                exit(1);
        }

        if( fp2 = fopen("t2.txt", “w”)) = = NULL) {
                printf(“Cannot open file\n”);
                exit(1);
        }
        while( ! feof(fp1)){
                fgets( ch, 99, fp1);
                fputs( ch, fp2);
        }

        fclose(fp1);
        fclose(fp2);
        printf(“One file copied”);
}
```

Another two very powerful functions for file operations in C are **fprintf(. . .)** and **fscanf(. . .)**. These functions operate exactly like **printf(. . .)** and **scanf(. . .)** except that they work with files. Their prototypes are :

```
int fprintf( FILE *fp, char *control-string, . . . );
int fscanf( FILE *fp, char *control-string, . . . );
```

These functions operate on the file specified by *fp*. Advantage of **fprintf(. . .)** and **fscanf(. . .)** is that they make it very easy to write variety of data to a file using a text format.

Example 2:

```
void main(){
        FILE *fp;
        int x ;
        float y;
        char str[80];
        gets(str);
        scanf(“%d%f ”,&x,&y);

        fp = fopen(“C:\\tc\\bin\\test.txt”, “wb”);
        fprintf(fp, “ %s \t %d \t %f ”, str, x, y);
        fclose(fp);

        fp = fopen(“C:\\tc\\bin\\test.txt”, “rb”);
        fscanf(fp, “ %s %d %f ”, &str, &x, &y );
        fclose(fp);
}
```

When a C program begins execution, three streams called *standard input* (**stdin**), s*tandard output* (**stdout**) and *standard error* (**stderr**).are automatically opened and available for use. Normally, **stdin** inputs from the keyboard; **stdout** and **stderr** write to the screen.

C file system includes another two important functions: **fread(. . .)** and **fwrite(. . .)** which can read and write any type of data, using any kind of representation. Their prototypes are :

```
size_t fread(void *buff, size_t size, size_t num, FILE *fp);
size_t fwrite(void *buff, size_t size, size_t num, FILE *fp);
```

The **fread(. . .)** function reads from the file associated with *fp*, *num* number of objects, each object *size* bytes long, into the buffer pointed to by *buff*. It returns the number of objects actually read. The **fwrite(. . .)** function writes to the file associated with *fp*, *num* number of objects, each object s*ize* bytes long from the buffer pointed to by *buff*. It returns the number of object written. This value will be less than *num* only if an output error has occurred.

Example 3:

```
void main(){
        FILE *fp;
        int k ;
        float p[5], d[5] = {5.67, 10.23, 12.45, 50.87, 98. 32};

        fp = fopen(“C:\\testfile.txt”, “wb”)
        fwrite(d, sizeof(d), 1, fp);
        fclose(fp);

        fp = fopen(“C:\\testfile.txt”, “rb”);
        fread(p, sizeof(p), 1, fp);
        fclose(fp);

        for( k = 0; k < 5; k ++)
                printf(“%f”, p[k]);
}
```

We can access any point in a file at any time by using the function **fseek(. . .)**. It’s prototype is :

```
int fseek( FILE *fp, long offset, int origin);
```

Here, *fp* is associated with the file being accessed. The value of *offset* determines the number of bytes from *origin* to make the new current position, *origin* must be one of these macros, shown here with their meaning.

Value of Origin	**Meaning**
SEEK_SET	Seek from start of file
SEEK_CUR	Seek from current location
SEEK_END	Seek from end of the file

You can determine your current location of a file using **ftell(. . .)**, another of C’s file system functions. Its prototype is :

```
        long ftell( FILE *fp);
```

It returns the location of the file position indicator within the file associated with *fp*. If a failure occurs, it returns -1.

Example 4:

```
        void main(){
                long loc;
                char ch;
                FILE *in, *out;

                in = fopen("f1.txt", “rb”);
                out = fopen("f2.txt", “wb”);

                fseek( in, 1, SEEK_END);
                loc = ftell(in);

                while(loc > = 0 ){
                        fseek(in, loc, SEEK_SET ) ;
                        ch = fgetc(in);
                        fputc(ch, out);
                        loc - - ;
                }

                fclose(in);
                fclose(out);
        }
```

You can erase a file using **remove(. . .)** . Its prototype is :

```
        int remove( char *file_name);
```

It returns 0, if successful and non - 0, if an error occurs. You can position a file’s current location to the start of the file using **rewind(. . .)**. Its prototype is :

```
        void rewind( FILE *fp);
```

You can cause a file’s disk buffer to be flushed by using **fflush(. . .)**. Its prototype is :

```
        int fflush( FILE *fp);
```

The function returns 0 if successful, otherwise it returns **EOF**.