TATA ELXSI
engineering creativity

TATA

C Programming

Presenter : Learning & Development Team

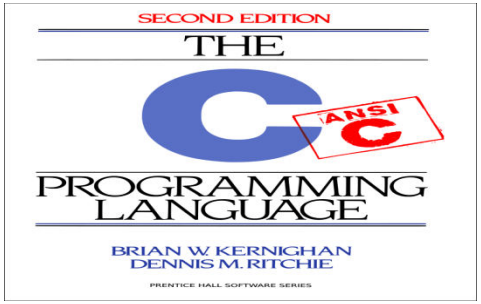
Day - 6

Disclaimer

- This material is developed for in-house training at TATA ELXSI.
- The training material therefore was solely developed for educational purposes. Currently, commercialization of the prototype does not exist, nor is the prototype available for general usage. TATA ELXSI, its staff, its students or any other participants can use it as a part of training. This material should not be found on any website.
- For the preparation of this material we have referred from the below mentioned links or books. Excerpts of these material has been taken where there is no copy right infringements. Some examples and concepts have been sourced from the below links and are open source material

  http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

ANSI C K&R



Day -6 Agenda

- File Handling
- fopen, fprintf, fscanf
- fseek, ftell, rewind
- fclose
- fread, fwrite

Introduction

- It is often useful tool, handling the files which are resided in the OS.
- ‘C’ Library provides library functions to access files.
- Through file handling, one can perform operations like create, modify, delete etc on system files.
- We shall look into couple of file handling functions

C Stream

- The file system transforms each into a logical device called a stream.

- There are two types of streams.
  - Text stream
  - Binary streams

- Text stream: it is sequence of characters. Standard C allows a text stream to be organized into lines terminated by a new line character.

- In a text stream certain character translations may occur as required by the host environment.

fopen()

- Syntax:
  - **FILE *fopen(const char *path, const char *mode);**

- The fopen() function is used to open a file.

- The first argument is a pointer to a string containing name of the file to be opened .

- The second argument is the mode in which the file is to be opened.

- The fopen() function returns a FILE stream pointer on success, while it returns NULL in case of a failure.

Example: fopen()

```
# include <stdio.h>
main(int argc, char  *argv[])
{
        FILE    *fp;
        char    ch;
        if(argc!= 2){
                printf("usage: a.out <file name>\n");
                _exit(1);
        }
        if((fp=fopen(argv[1],"w")) == NULL){
                printf("Can't open a file");
                _exit(2);
        }
}
```

The file opening modes are:

- ‘r’ : Open text file for reading. The stream is positioned at the beginning of the file.
- ‘r+’ : Open for reading and writing. The stream is positioned at the beginning of the file.
- ‘w’ : Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
- ‘w+’ : Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
- ‘a’ : Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.
- ‘a+’ : Open for reading and appending (writing at end of file). The file is created if it does not exist. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

fclose()

- Syntax:
  - int fclose(FILE *fp);
- The fclose() function first flushes the stream opened by fopen() and then closes the underlying descriptor.
- Upon successful completion this function returns 0 else end of file (eof) is returned.
- In case of failure, if the stream is accessed further then the behavior remains undefined.

Function:**fputc**

- **int fputc (int character, FILE * stream);**
- **Write character to stream** and advances the position indicator.

- **Parameters**
- Character : The int promotion of the character to be written (The value is internally converted to an unsigned char when written.)

- Stream : Pointer to a FILE object that identifies an output stream.

- **Return Value**
- On success, the *character* written is returned.

- If a writing error occurs, EOF is returned and the *error indicator* (ferror) is set.

Example : fputc

```
if((fp=fopen(argv[1],"w")) == NULL){
        printf("Can't open a file");
        exit(2);
}
do{
        fputc(ch=getchar(),fp);
}while(ch!=EOF); //EOF is ctrl D
fclose(fp);
```

Function fgetc() : Get character from stream

- **int fgetc (FILE * stream);**
- **Return Value:** Returns the character currently pointed by the internal file position indicator of the specified *stream*.
- The internal file position indicator is then advanced to the next character.
- If a read error occurs, the function returns EOF and sets the *error indicator* for the stream (ferror).
- fgetc and getc are equivalent, except that getc may be implemented as a macro in some libraries.

Example : fgetc

```
if((fp=fopen(argv[1],"r")) == NULL){
        printf("Can't open a file");
         _exit(2);
}
ch = fgetc(fp);
while(ch!=EOF)
{
                putchar(ch);
                ch = fgetc(fp);
}
fclose(fp);
```

Function **fputs**

- **int fputs (const char * str, FILE * stream);**
- Writes the *C string* pointed by *str* to the *stream*.
- **Parameters**
  - str*C string* with the content to be written to *stream*.
  - streamPointer to a FILE object that identifies an output stream.
- **Return Value**
  - On success, a non-negative value is returned.
  - On error, the function returns EOF and sets the *error indicator*

Example: fputs

```
if((fp = fopen("Test","w")) == NULL){      //open a file with Write mode
                printf("Can't open a file\n");
                exit(1);
        }
        do{
                printf("Enter a string, <CR> to quit      >");
                gets(str);
                strcat(str,"\n");          //add a new line
                fputs(str,fp);
        } while(*str!='\n');
```

Function **fgets**

- **char * fgets (char * str, int num, FILE * stream);**

- Reads characters from *stream* and stores them as a C string into *str* until (*num*-1).
- Characters have been read or either a newline or the *end-of-file* is reached, whichever happens first.

- **Parameters**
  - str: Pointer to an array of chars where the string read is copied.
  - num : Maximum number of characters to be copied into *str* .
  - stream : Pointer to a FILE object that identifies an input stream.
- **Return Value**
  - On success, the function returns *str*.
  - If a read error occurs, the *error indicator* (ferror) is set and a null pointer is also returned

Example: fgets

```
if((fp = fopen("Test","w+")) == NULL){
        printf("Can't open a file\n");
        _exit(1);
}
do{
        printf("Enter a string, <CR> to quit      >");
        gets(str);
        strcat(str,"\n");//add newline at the end of string
        fputs(str,fp);
} while(*str!='\n');
```

function: **fprintf**

- **Write formatted data to stream**
- **int fprintf (FILE * stream, const char * format, ...);**

- **Parameters**
  - Stream: Pointer to a FILE object that identifies an output stream.
  - Format:C string that contains the text to be written to the stream.

- **Return Value**
- On success, the total number of characters written is returned.

- If a writing error occurs, the *error indicator* (ferror) is set and a negative number is returned.

Example: fprintf()

```
fp=fopen("stud.dat","ab+");
if(fp==NULL){
        perror("can't open the file\n");
        _exit(0);
}
scanf("%s",name);
scanf("%d%d%d",&d,&m,&y);
fprintf(fp,"%s %d %d %d\n",name,d,m,y);
printf("name is: %s\n",name);
printf("DOB is :%d/%d/%d\n",d,m,y);
```

Function : **fscanf**

- **int fscanf (FILE * stream, const char * format, ...);**
- **Read formatted data from stream**
  - Reads data from the *stream* and stores them according to the parameter *format* into the locations pointed by the additional arguments.
- **Parameters**
  - Stream: Pointer to a FILE object that identifies the input stream to read data from.
  - Format: string that contains a sequence of characters that control how characters extracted from the stream are treated:

Example : fprintf()

```
fp=fopen("stud.dat","rb+");
     if(fp==NULL){
               perror("can't open the file\n");
               exit(0);
     }
     fscanf(fp,"%s %d %d %d",name,&d,&m,&y);   // Reading from file
    do{
               printf("name is: %s\n",name);
               printf("DOB is :%d/%d/%d\n",d,m,y);
               fscanf(fp,"%s %d %d %d",name,&d,&m,&y);        // Reading from file
     }     while(!feof(fp)) ;
     fclose(fp);
```

fread() and fwrite()

- **size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);**
- **size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);**
- The functions fread/fwrite are used for reading/writing data from/to the file opened by fopen function.
- The first argument is a pointer to buffer used for reading/writing the data.
- The data read/written is in the form of ‘nmemb’ elements each ‘size’ bytes long.
- In case of success, fread/fwrite return the number of bytes actually read/written from/to the stream opened by fopen function.
- In case of failure, a lesser number of byes (then requested to read/write) is returned.

Example: fread and fwrite

```
struct student
{
        unsigned int reg_no;
        char name[50];
        unsigned int mark;
} ;
       struct student rec;
        FILE *fp;
        fp=fopen("class.rec","wb");
        printf("\n Enter the register number, name and mark\n");
        printf("Type ctrl+d to stop\n");
```

```
// read from keyboard and write to file
        while(scanf("%u %s %u",&rec.reg_no,rec.name,&rec.mark)!=EOF)
                fwrite(&rec,sizeof(rec),1,fp);
        fclose(fp);

// Read from file and write on screen
        fp=fopen("class.rec","rb");
        while(fread(&rec,sizeof(rec),1,fp))
                printf("%5u %10s %3u\n",rec.reg_no,rec.name,rec.mark);
        fclose(fp);
```

fseek()

- Syntax:
- **int fseek(FILE *stream, long offset, int whence);**

- The fseek() function is used to set the file position indicator for the stream to a new position.
- The first argument is the FILE stream pointer returned by the fopen() function.
- The second argument ‘offset’ tells the amount of bytes to seek.
- The third argument ‘whence’ tells from where the seek of ‘offset’ number of bytes is to be done.
- SEEK_SET : the start of the file
- SEEK_CUR : the current position
- SEEK_END : the end of the file.

- Upon success, this function returns 0, otherwise it returns -1.

Function: **rewind**

- **void rewind (FILE * stream);**
- **Set position of stream to the beginning**
  - Sets the position indicator associated with *stream* to the beginning of the file.

- Example rewind()

```
pFile = fopen ("myfile.txt","w+");
  for ( n='A' ; n<='Z' ; n++)
    fputc ( n, pFile);
  rewind (pFile);
  fread (buffer,1,26,pFile);
  fclose (pFile);
  buffer[26]='\0';
  puts (buffer);
```

Function : ftell()

ftell() is used to determine the current location of a file.

ftell(FILE *fp);

It returns the location of the current position of the file associated with fp. If a failure occurs, it returns −1.

On failure, -1L is returned, and errno is set to a system-specific positive value.

Example: ftell()

```
/* ftell example : getting size of a file */
int main ()
{
  FILE * pFile;
  long size;
  pFile = fopen ("myfile.txt","rb");
  if (pFile==NULL) perror ("Error opening file");
  else  {
    fseek (pFile, 0, SEEK_END);   // non-portable
    size=ftell (pFile);
    fclose (pFile);
    printf ("Size of myfile.txt: %ld bytes.\n",size);
  }
}
```

Function:feof()

- **int feof (FILE * stream);**

- **Check end-of-file indicator**
  - Checks whether the *end-of-File indicator* associated with *stream* is set, returning a value different from zero if it is.

- **Parameters**
  - streamPointer to a FILE object that identifies the stream.

- **Return Value**
  - A non-zero value is returned in the case that the *end-of-file indicator* associated with the stream is set. Otherwise, zero is returned.

Example: feof()

```
pFile = fopen ("myfile.txt","rb");
    if (pFile==NULL) perror ("Error opening file");
    else    {
        while (fgetc(pFile) != EOF) {
                ++n;
     }
     if (feof(pFile)) {
            puts ("End-of-File reached.");
            printf ("Total number of bytes read: %d\n", n);
     }
     else puts ("End-of-File was not reached.");
     fclose (pFile);
}
```

Function: ferror

- int ferror (FILE * stream);

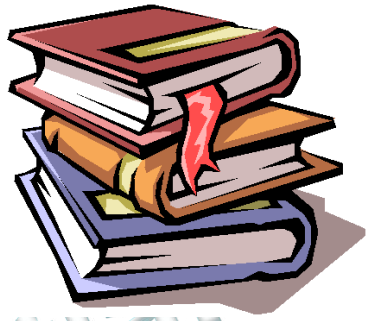
- **Check error indicator**
  - Checks if the *error indicator* associated with *stream* is set, returning a value different from zero if it is.
  - This indicator is generally set by a previous operation on the *stream* that failed
- **Parameters**
  - streamPointer to a FILE object that identifies the stream.

- **Return Value**
- A non-zero value is returned in the case that the *error indicator* associated with the stream is set.
- Otherwise, zero is returned.

Function: ferror()

```
/* ferror example: writing error */
int main ()
{
    FILE * pFile;
    pFile=fopen("myfile.txt","r");
    if (pFile==NULL) perror ("Error opening file");
    else {
        fputc ('x',pFile);
        if (ferror (pFile))
            printf ("Error Writing to myfile.txt\n");
        fclose (pFile);
  }
  return 0;
}
```

references



- The GNU C Reference Manual : https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

- The GNU C Reference Manual : http://www.cprogramming.com/

- C Programming Language

by Brian W. Kernighan , Dennis Ritchie

- C: The Complete Reference

by Herbert Schildt

Thank you

TATA ELXSI

ITPB Road Whitefield
Bangalore 560 048 India
Tel +91 80 2297 9123
Fax +91 80 2841 1474
e-mail info@tataelxsi.com

www.tataelxsi.com