File I/O

Arash Rafiey

November 7, 2017

Files

File is a place on disk where a group of related data is stored.

Files

File is a place on disk where a group of related data is stored.

C provides various functions to handle files on the storage devices.

Files

File is a place on disk where a group of related data is stored.

C provides various functions to handle files on the storage devices.

Function	description
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
getw()	reads a integer from a file
putw()	writes a integer to a file
fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

File Pointer

File pointer is used to let the program keep track of the file being accessed.

File Pointer

File pointer is used to let the program keep track of the file being accessed.

Example:

FILE *filepointer_name;

File Pointer

File pointer is used to let the program keep track of the file being accessed.

Example:

FILE *filepointer_name;

File pointer points to a structure that contains information about the file, such as:

File Pointer

File pointer is used to let the program keep track of the file being accessed.

Example:

FILE *filepointer_name;

File pointer points to a structure that contains information about the file, such as:

- current character position in the buffer
- whether file is being read or written
- whether errors or end of file have occurred .

File Pointer

File pointer is used to let the program keep track of the file being accessed.

Example:

FILE *filepointer_name;

File pointer points to a structure that contains information about the file, such as:

- current character position in the buffer
- whether file is being read or written
- whether errors or end of file have occurred .

The header file <**stdio.h**> includes **FILE** structure declaration.

File Access

Opening a file:

fopen( ) function can be used to create a new file or to open an existing file.

File Access

Opening a file:

fopen( ) function can be used to create a new file or to open an existing file.

Syntax:

FILE *fopen(char * filename, char * mode);

File Access

Opening a file:

fopen( ) function can be used to create a new file or to open an existing file.

Syntax:

FILE *fopen(char * filename, char * mode);

It returns a pointer to structure **FILE**.

File Access

Opening a file:

fopen( ) function can be used to create a new file or to open an existing file.

Syntax:

FILE *fopen(char * filename, char * mode);

It returns a pointer to structure **FILE**.

If the file is successfully opened then fopen()returns a **pointer** to file and if it is unable to open a file then it returns **NULL**.

File Access

Opening a file:

fopen( ) function can be used to create a new file or to open an existing file.

Syntax:

FILE *fopen(char * filename, char * mode);

It returns a pointer to structure **FILE**.

If the file is successfully opened then fopen()returns a **pointer** to file and if it is unable to open a file then it returns **NULL**.

Filename is a string that holds the name of the file on disk (including a path like /cs/course if necessary).

File Access

Modes of fopen() can be the following values:

File Access

Modes of fopen() can be the following values:

mode	description
r	opens an text file for reading mode
w	opens or create a text file for writing mode.
a	opens a text file in append mode
r+	opens a text file in both reading and writing mode
w+	opens a text file in both reading and writing mode
a+	opens a text file in both reading and writing mode
rb	opens an binary file for reading mode
wb	opens or create a binary file for writing mode
ab	opens a binary file in append mode
rb+	opens a binary file in both reading and writing mode
wb+	opens a binary file in both reading and writing mode
ab+	opens a binary file in both reading and writing mode

File Access

Example:

```
FILE *fp;
fp = fopen( "xyz.txt", "r" );
```

File Access

Example:

```
FILE *fp;
fp = fopen( "xyz.txt", "r" );
//This will open the file xyz.txt in read mode.
```

File Access

Example:

```
FILE *fp;
fp = fopen( "xyz.txt", "r");
//This will open the file xyz.txt in read mode.
```

Closing a file:

After reading/writing of a file, the file should be closed.

File Access

Example:

```
FILE *fp;
fp = fopen( "xyz.txt", "r" );
//This will open the file xyz.txt in read mode.
```

Closing a file:

After reading/writing of a file, the file should be closed.

Closing of a file is performed using library function **fclose()**.

File Access

Example:

```
FILE *fp;
fp = fopen( "xyz.txt", "r" );
//This will open the file xyz.txt in read mode.
```

Closing a file:

After reading/writing of a file, the file should be closed.

Closing of a file is performed using library function **fclose()**.

Syntax:

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

File Access

Example:

FILE *fp;

fp = fopen("xyz.txt", "r");

//This will open the file xyz.txt in read mode.

Closing a file:

After reading/writing of a file, the file should be closed.

Closing of a file is performed using library function **fclose()**.

Syntax:

int fclose(FILE *file_pointer);

fclose() returns zero on success and returns **EOF** (end of file) if there is an error in closing the file.

EOF is a constant defined in the header file **stdio.h**.

File Access

Example:

FILE *fp;

fp = fopen("xyz.txt", "r");

//This will open the file xyz.txt in read mode.

Closing a file:

After reading/writing of a file, the file should be closed.

Closing of a file is performed using library function **fclose()**.

Syntax:

int fclose(FILE *file_pointer);

fclose() returns zero on success and returns **EOF** (end of file) if there is an error in closing the file.

EOF is a constant defined in the header file **stdio.h**.

Example: fclose(fp);

Writing to a file

Writing to a file:
When a file is opened for writing, it is created if it does not already exist.

Writing to a file

Writing to a file:

When a file is opened for writing, it is created if it does not already exist.

If the file already exists then its old contents are discarded.

Writing to a file

Writing to a file:

When a file is opened for writing, it is created if it does not already exist.

If the file already exists then its old contents are discarded.

The function **fprintf** can be used to write to a file.

It is similar to printf except that it takes a file pointer as its first argument.

Writing to a file

Writing to a file:

When a file is opened for writing, it is created if it does not already exist.

If the file already exists then its old contents are discarded.

The function **fprintf** can be used to write to a file.

It is similar to printf except that it takes a file pointer as its first argument.

Syntax:

```
int fprintf(FILE *fp,char *format, ...);
```

Writing to a file

Writing to a file:

When a file is opened for writing, it is created if it does not already exist.

If the file already exists then its old contents are discarded.

The function **fprintf** can be used to write to a file.

It is similar to printf except that it takes a file pointer as its first argument.

Syntax:

```
int fprintf(FILE *fp,char *format, ...);
```

Example:

```
FILE *fp;
fp = fopen("xyz.txt", "w");
fprintf(fp, "HELLO \n");
```

Writing to a file

fputc() is used to write a single character at a time.

Writing to a file

fputc() is used to write a single character at a time.

Syntax:

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

Writing to a file

fputc() is used to write a single character at a time.

Syntax:

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

It writes a character to the file and returns success, or EOF if an error occurs.

Writing to a file

fputc() is used to write a single character at a time.

Syntax:

int fputc (char c, FILE *fp);

It writes a character to the file and returns success, or EOF if an error occurs.

Example:

```
FILE *fp;
char c;
fp = fopen("xyz.txt", "w");
for (c = 'A' ; c <= 'Z' ; c++)
  fputc ( c , fp );
```

Writing to a file

The function **fputs()** writes a string to the file.

Writing to a file

The function **fputs()** writes a string to the file.

Syntax:

```
int fputs (char *s, FILE *fp);
```

Writing to a file

The function **fputs()** writes a string to the file.

Syntax:

```
int fputs (char *s, FILE *fp);
```

It returns a non-negative value on success or EOF in case of any error.

Writing to a file

The function **fputs()** writes a string to the file.

Syntax:

```
int fputs (char *s, FILE *fp);
```

It returns a non-negative value on success or EOF in case of any error.

Example:

```
FILE *fp;
char c[] = "Hello";
fp = fopen("xyz.txt", "w");
fputs ( c , fp );
```

Example:

```
FILE *fp;
int n = 2017;
fp = fopen("xyz.txt", "w");
fprintf(fp, "%s %s %s %d ", "We", "are", "in", n);
```

Reading a file

Reading a file:

A file can be opened for reading, only if it already exist.

Reading a file

Reading a file:

A file can be opened for reading, only if it already exist.

Reading a file that does not exist is an error.

Reading a file

Reading a file:

A file can be opened for reading, only if it already exist.

Reading a file that does not exist is an error.

fscanf can be used to read from a file.

It is similar to scanf except that it takes a file pointer as its first argument.

Reading a file

Reading a file:

A file can be opened for reading, only if it already exist.

Reading a file that does not exist is an error.

fscanf can be used to read from a file.

It is similar to scanf except that it takes a file pointer as its first argument.

Syntax:

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

Reading a file

Reading a file:

A file can be opened for reading, only if it already exist.

Reading a file that does not exist is an error.

fscanf can be used to read from a file.

It is similar to scanf except that it takes a file pointer as its first argument.

Syntax:

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

Example:

```
FILE *fp;
char buff[100];
fp = fopen("xyz.txt", "r");
fscanf(fp, "%s", buff);
```

Reading a file

fgetc() is used to read a single character from the file.

Reading a file

fgetc() is used to read a single character from the file.

Syntax:

```
char fgetc (FILE *fp);
```

Reading a file

fgetc() is used to read a single character from the file.

Syntax:

```
char fgetc (FILE *fp);
```

It returns the character that is read from the file, or EOF if an error occurs.

Reading a file

fgetc() is used to read a single character from the file.

Syntax:

```
char fgetc (FILE *fp);
```

It returns the character that is read from the file, or EOF if an error occurs.

Example:

```
FILE *fp;
char c;

fp = fopen("xyz.txt", "r");

while(c!=EOF) {

  c = fgetc(fp);

  printf("%c",c);
}
```

Reading a file

The function **fgets()** is used to read a line from the file.

Reading a file

The function **fgets()** is used to read a line from the file.

Syntax:

```
char *fgets (char *s, int n, FILE *fp);
```

Reading a file

The function **fgets()** is used to read a line from the file.

Syntax:

```
char *fgets (char *s, int n, FILE *fp);
```

fgets reads a line until (n-1) characters are read or new line character is read, and stores it in the string pointed by s.

It returns the parameter s on success or a null pointer in case of any error.

Reading a file

The function **fgets()** is used to read a line from the file.

Syntax:

```
char *fgets (char *s, int n, FILE *fp);
```

fgets reads a line until (n-1) characters are read or new line character is read, and stores it in the string pointed by s.

It returns the parameter s on success or a null pointer in case of any error.

Example:

```
FILE *fp;
char c[100];

fp = fopen("xyz.txt", "r");

if( fgets(c,100,fp) != NULL )

  printf("%s",c);
```

Example:

```
FILE *fp;
int n;
fp = fopen("xyz.txt", "r");
fscanf(fp, "%d", &n); // reading an integer n from xyz.txt
fclose(fp);
```

Example

Program to copy contents of one file to another (Using fgetc and fputc).

Example

Program to copy contents of one file to another (Using fgetc and fputc).

```
# include <stdio.h>
void main() {
  FILE *fp1, *fp2;
  char a;
  fp1 = fopen( "xyz.txt", "r" );
  if ( fp1 == NULL ) {
    printf("cannot open this file");
    exit(1);
  }
  fp2 = fopen( "abc.txt", "w" );
```

Example

```
if ( fp2 == NULL ) {
   printf("Not able to open this file");
   fclose(fp1);
   exit(1);
   }
  do {
   a = fgetc(fp1);
   fputc(a, fp2);
   } while (a != EOF);
  fcloseall();
}
```

Example

Program to open a file and count the occurrence of the word "system" in the file.

Example

Program to open a file and count the occurrence of the word "system" in the file.

```
# include <stdio.h>
# include <string.h>
int main() {
  FILE *fp;
  char w[ ] = "system";
  char ch[100];
  int i, count=0;
  fp = fopen( "xyz.txt", "r" );
  if ( fp == NULL ) {
    printf("cannot open this file");
    return(1);
  }
```

Example

```
while ( !feof(fp) ) { //while end of file is not reached
   fscanf(fp,"%s",ch);
   if( strcmp(w,ch)==0 )
    count++;
}
printf("%s occurs %d times in the file \n",w,count);
fclose(fp);
return(0);
}
```

fseek function

char fseek (FILE * stream, long int offset, int origin)
origin could be :
SEEK_SET : Beginning of file
SEEK_CUR : Current position of the file pointer
SEEK_END : End of file

fseek function

char fseek (FILE * stream, long int offset, int origin)
origin could be :
SEEK_SET : Beginning of file
SEEK_CUR : Current position of the file pointer
SEEK_END : End of file

```
# include <stdio.h>
int main ()
{
  FILE * pf;
  pf = fopen ( "example.txt" , "w" );
  fputs ( "This is a test" , pf );
  fseek ( pf , 10 , SEEK_SET );
  fputs ( "book " , pf );
  fclose (pf);
  return 0;
}
```

Example

Program to open a file and modify the third line to “Hello World”

Example

Program to open a file and modify the third line to “Hello World”

```
# include <stdio.h>
# include <string.h>
int main() {
  FILE *fp;
  int i, len, count=0;
  fp = fopen( "xyz.txt", "r" );
  if ( fp == NULL ) {
    printf("cannot open this file");
    return(1);
  }
```

Example

```
char Buffer[100], *buffpointer;
while ( !feof(fp) ) { //while ! end of file and not three lines read
  getline(&Buffer, &len, fp);
  buffpointer= Buffer; counter++;
  if (counter == 3) {
    fseek( fp , -strlen(buffpointer) , SEEK_CUR) ;
    fputs ("Hello World \n", fp);
    fclose(fp);
    break;
  }
}
```

Changing a particular word in a file

Write a program that reads a file and change all the words “book” to “cook”.

```
# include <stdio.h>
# include <string.h>
int main() {
  FILE *fp;
  fp = fopen( "example.txt", "r+" );
  if ( fp == NULL ) {
    printf("cannot open this file");
    return(1);
  }
```

```
char oneword[100];
while ( !feof(fp) ) {
   fscanf(fp,"%s",oneword);
   if ( strcmp(oneword, "book") == 0 ) {
    fseek( fp , -4 , SEEK_CUR) ;
    fputs ("cook", fp);
   }
}
fclose(fp);
}
```