**Experiment No. 4**

**Aim**: Write C program that copies the 25th character of each line in a file to a new file. All extracted characters are to be on same line. If a line has fewer than 25 characters, write the last character. If a line is blank, copy nothing. At the end of file, write a newline to the new file along with total lines scanned.

Steps:

1. Create a C file and write text to that file.
2. Copy 25th character from the existing file.
3. Create a new file and copy contents to it provided:
   1. All copied characters are on same line
   2. If line is blank, copy nothing.
   3. At the EOF, add total no. of lines scanned.

**WHAT IS FILE?**

File is a collection of bytes that is stored on secondary storage devices like disk. There are two kinds of files in a system. They are,

1. Text files (ASCII): Text files contain ASCII codes of digits, alphabetic and symbols
2. Binary files: Binary file contains collection of bytes (0’s and 1’s). Binary files are compiled version of text files.

**Basics of File Handling in C**

1. Creation of a new file (fopen with attributes as “a” or “a+” or “w” or “w++”)
2. Opening an existing file (fopen)
3. Reading from file (fscanf or fgetc)
4. Writing to a file (fprintf or fputs)
5. Moving to a specific location in a file (fseek, rewind)
6. Closing a file (fclose)

Opening or creating a file

**fopen()** – To open a file

Declaration: FILE \***fopen**(const char \*filename, const char \*mode)

**Eg:**

FILE \*fp;  
fp=**fopen**(“filename”, ”‘mode”);

Modes:

* **“r” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
* **“w” –** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
* **“a” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
* **“r+” –** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
* **“w+” –** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.
* **“a+” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

**fclose()**– To close a file

Declaration: int **fclose**(FILE \*fp);

**Eg:**

**fclose**(fp);

**/ \* Open, write and close a file : \*/**

# include <stdio.h>

# include <string.h>

int main( )

{

    FILE \*fp ;

    char data[50];

    // opening an existing file

    printf( "Opening the file test.c in write mode" ) ;

    fp = fopen("test.c", "w") ;

    if ( fp == NULL )

    {

        printf( "Could not open file test.c" ) ;

        return 1;

    }

    printf( "\n Enter some text from keyboard” \

             “ to write in the file test.c" ) ;

    // getting input from user

    while ( strlen ( gets( data ) ) > 0 )

    {

        // writing in the file

        fputs(data, fp) ;

        fputs("\n", fp) ;

    }

    // closing the file

    printf("Closing the file test.c") ;

    fclose(fp) ;

    return 0;

}

**OUTPUT:**

|  |
| --- |
| Opening the file test.c in write mode  Enter some text from keyboard to write in the file test.c Hai, How are you? Closing the file test.c |

**/\* Open, Read and close a file: reading string by string \*/**

# include <stdio.h>

int main( )

{

         FILE \*fp ;

         char data[50] ;

         printf( "Opening the file test.c in read mode" ) ;

         fp = fopen( "test.c", "r" ) ;

         if ( fp == NULL )

         {

                 printf( "Could not open file test.c" ) ;

                 return 1;

         }

         printf( "Reading the file test.c" ) ;

         while( fgets ( data, 50, fp ) != NULL )

         printf( "%s" , data ) ;

         printf("Closing the file test.c") ;

         fclose(fp) ;

         return 0;

}

**OUTPUT:**

|  |
| --- |
| Opening the file test.c in read mode  Reading the file test.c  Hi, How are you?  Closing the file test.c |

|  |  |
| --- | --- |
| **File handling functions** | **Description** |
| fopen () | fopen () function creates a new file or opens an existing file. |
| fclose () | fclose () function closes an opened file. |
| getw () | getw () function reads an integer from file. |
| putw () | putw () functions writes an integer to file. |
| fgetc () | fgetc () function reads a character from file. |
| fputc () | fputc () functions write a character to file. |
| gets () | gets () function reads line from keyboard. |
| puts () | puts () function writes line to o/p screen. |
| fgets () | fgets () function reads string from a file, one line at a time. |
| fputs () | fputs () function writes string to a file. |
| feof () | feof () function finds end of file. |
| fgetchar () | fgetchar () function reads a character from keyboard. |
| fprintf () | fprintf () function writes formatted data to a file. |
| fscanf () | fscanf () function reads formatted data from a file. |
| fputchar () | fputchar () function writes a character onto the output screen from keyboard input. |
| fseek () | fseek () function moves file pointer position to given location. |
| SEEK\_SET | SEEK\_SET moves file pointer position to the beginning of the file. |
| SEEK\_CUR | SEEK\_CUR moves file pointer position to given location. |
| SEEK\_END | SEEK\_END moves file pointer position to the end of file. |
| ftell () | ftell () function gives current position of file pointer. |
| rewind () | rewind () function moves file pointer position to the beginning of the file. |
| getc () | getc () function reads character from file. |
| getch () | getch () function reads character from keyboard. |
| getche () | getche () function reads character from keyboard and echoes to o/p screen. |
| getchar () | getchar () function reads character from keyboard. |
| putc () | putc () function writes a character to file. |
| putchar () | putchar () function writes a character to screen. |
| printf () | printf () function writes formatted data to screen. |
| sprinf () | sprinf () function writes formatted output to string. |
| scanf () | scanf () function reads formatted data from keyboard. |
| sscanf () | sscanf () function Reads formatted input from a string. |
| remove () | remove () function deletes a file. |
| fflush () | fflush () function flushes a file. |

**Returned values of printf() and scanf()**

In C, printf() returns the number of **characters**successfully written on the output and scanf() returns number of **items**successfully read.

int main()

{

  printf(" %d", printf("%s", "geeksforgeeks"));

  getchar();

}

Irrespective of the string user enters, below program prints **1**.

|  |  |
| --- | --- |
| int main()  {    char a[50];    printf(" %d", scanf("%s", a));    getchar();  }  **EOF, getc() and feof() in C**  In C/C++, [getc()](http://www.cplusplus.com/reference/clibrary/cstdio/getc/)returns EOF when end of file is reached. getc() also returns EOF when it fails. So, only comparing the value returned by getc() with EOF is not sufficient to check for actual end of file.To solve this problem, C provides [feof()](http://en.wikipedia.org/wiki/Feof) which returns non-zero value only if end of file has reached, otherwise it returns 0. For example, consider the following C program to print contents of file test.txt on screen. In the program, returned value of getc() is compared with EOF first, then there is another check using feof(). By putting this check, we make sure that the program prints “End of file reached” only if end of file is reached. And if getc() returns EOF due to any other reason, then the program prints “Something went wrong”   |  | | --- | | #include <stdio.h>    int main()  {    FILE \*fp = fopen("test.txt", "r");    int ch = getc(fp);    while (ch != EOF)    {      /\* display contents of file on screen \*/      putchar(ch);        ch = getc(fp);    }      if (feof(fp))       printf("\n End of file reached.");    else       printf("\n Something went wrong.");    fclose(fp);      getchar();    return 0;  } | |

**What is return type of getchar(), fgetc() and getc() ?**

In C, return type of getchar(), fgetc() and getc() is int (not char). So it is recommended to assign the returned values of these functions to an integer type variable.

|  |
| --- |
| char ch;  /\* May cause problems \*/  while ((ch = getchar()) != EOF)  {     putchar(ch);  } |

Here is a version that uses integer to compare the value of getchar().

|  |
| --- |
| int in;  while ((in = getchar()) != EOF)  {     putchar(in);  } |

**C program to copy contents of one file to another file**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h> // For exit()  int main()  {      FILE \*fptr1, \*fptr2;      char filename[100], c;      printf("Enter the filename to open for reading \n");      scanf("%s", filename);      // Open one file for reading      fptr1 = fopen(filename, "r");      if (fptr1 == NULL)      {          printf("Cannot open file %s \n", filename);          exit(0);      }      printf("Enter the filename to open for writing \n");      scanf("%s", filename);      // Open another file for writing      fptr2 = fopen(filename, "w");      if (fptr2 == NULL)      {          printf("Cannot open file %s \n", filename);          exit(0);      }      // Read contents from file      c = fgetc(fptr1);      while (c != EOF)      {          fputc(c, fptr2);          c = fgetc(fptr1);      }      printf("\nContents copied to %s", filename);        fclose(fptr1);      fclose(fptr2);      return 0;  } |

Output:  
Enter the filename to open for reading

a.txt

Enter the filename to open for writing

b.txt

Contents copied to b.txt