CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 24

File Management in C (2)

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
www.mijanrahman.com

Contents

File Management in C

I/O Operations on Files

- One a file is opened, reading out of or writing to it is accomplished the standard I/O functions, listed below:

Function name	*Operation*
fopen()	* Creates a new file for use. * Opens an existing file for use.
fclose()	* Closes a file which has been opened for use.
getc()	* Reads a character from a file.
putc()	* Writes a character to a file.
fprintf()	* Writes a set of data values to a file.
fscanf()	* Reads a set of data values from a file.
getw()	* Reads an integer from a file.
putw()	* Writes an integer to a file.
fseek()	* Sets the position to a desired point in the file.
ftell()	* Gives the current position in the file (in terms of bytes from the start).
rewind()	* Sets the position to the beginning of the file.

I/O Operations on Files

- **Reading From a File:** The file read operation in C can be performed using functions fscanf() or fgets(). Both the functions performed the same operations as that of scanf and gets but with an additional parameter, the file pointer. There are also other functions we can use to read from a file. Such functions are listed below:

Function	Description
fscanf()	Use formatted string and variable arguments list to take input from a file.
fgets()	Input the whole line from the file.
fgetc()	Reads a single character from the file.
fgetw()	Reads a number from a file.
fread()	Reads the specified bytes of data from a binary file.

I/O Operations on Files

- **Write to a file:** The file write operations can be performed by the functions fprintf() and fputs() with similarities to read operations. C programming also provides some other functions that can be used to write data to a file such as:

Function	Description
fprintf()	Similar to printf(), this function use formatted string and varible arguments list to print output to the file.
fputs()	Prints the whole line in the file and a newline at the end.
fputc()	Prints a single character into the file.
fputw()	Prints a number to the file.
fwrite()	This functions write the specified amount of bytes to the binary file.

getc and putc Functions

- The getc() and putc() functions in C are used for character-based input and output operations on files. They are part of the standard I/O library (stdio.h).
- Both getc() and putc() are simple and efficient for reading and writing characters to files. They can be useful for handling character-based input and output operations when dealing with files in C.

- **int getc(FILE *stream):**
  - Reads a character from the specified input stream.
  - The stream parameter is a pointer to a FILE object representing the stream from which to read the character.
  - Returns the character read as an unsigned char cast to an int, or EOF (defined in stdio.h) if an error occurs or if the end of the file is reached.

getc and putc Functions

- **int getc(FILE *stream):**

```
int ch;
FILE *fp;
fp = fopen("input.txt", "r");
if (fp != NULL) {
    while ((ch = getc(fp)) != EOF) {
        printf("%c", ch);
    }
    fclose(fp);
}
```

getc and putc Functions

- **int putc(int character, FILE *stream):**
  - Writes a character to the specified output stream.
  - The character parameter is the character to be written, specified as an int. It's typically cast to unsigned char before being written.
  - The stream parameter is a pointer to a FILE object representing the output stream to which the character will be written.
  - Returns the character written as an unsigned char cast to an int, or EOF if an error occurs.

getc and putc Functions

- **int putc(int character, FILE *stream):**

```
int ch;
FILE *fp;
fp = fopen("output.txt", "w");
if (fp != NULL) {
    for (ch = 'A'; ch <= 'Z'; ch++) {
        putc(ch, fp);
    }
    fclose(fp);
}
```

getc and putc Functions

- **C Program using getc and putc functions:**

- Here's a simple C program that reads from one file character by character using getc() and writes the content to another file using putc().

```c
#include <stdio.h>
int main() {
    FILE *inputFile, *outputFile;
    int ch;

    inputFile = fopen("input.txt", "r");
    if (inputFile == NULL) {
        perror("Error opening input file");
        return 1;
    }
    outputFile = fopen("output.txt", "w");
    if (outputFile == NULL) {
        perror("Error opening output file");
        fclose(inputFile);
        return 1;
    }

    while ((ch = getc(inputFile)) != EOF) {
        putc(ch, outputFile);
    }
    fclose(inputFile);
    fclose(outputFile);
    printf("File copied successfully!\n");
    return 0;
}
```

getw and putw Functions

- The getw() and putw() functions in C are used for reading and writing binary data (integers) to files. They are typically used for binary file I/O.

- **int getw(FILE *stream):**
  - Reads a binary integer from the specified input stream.
  - The stream parameter is a pointer to a FILE object representing the stream from which to read the integer.
  - Returns the integer read from the file.

getw and putw Functions

- **int getw(FILE *stream):**

```
int num;
FILE *fp;
fp = fopen("data.bin", "rb");
if (fp != NULL) {
    num = getw(fp);
    printf("Read number: %d\n", num);
    fclose(fp);
}
```

getw and putw Functions

- **int putw(int num, FILE *stream):**
  - Writes a binary integer to the specified output stream.
  - The num parameter is the integer to be written.
  - The stream parameter is a pointer to a FILE object representing the output stream to which the integer will be written.
  - Returns 0 on success or EOF if an error occurs.

getw and putw Functions

- **int putw(int num, FILE *stream):**

```
int num = 42;
FILE *fp;
fp = fopen("data.bin", "wb");
if (fp != NULL) {
    putw(num, fp);
    fclose(fp);
}
```

getw and putw Functions

- **C Program using getw and putw functions:**

- Here's a simple C program that demonstrates the usage of getw() and putw() functions to read and write integers to a binary file.

```c
#include <stdio.h>
int main() {
    FILE *inputFile, *outputFile;
    int num;

    inputFile = fopen("input.bin", "rb");
    if (inputFile == NULL) {
        perror("Error opening input file");
        return 1;
    }

    outputFile = fopen("output.bin", "wb");
    if (outputFile == NULL) {
        perror("Error opening output file");
        fclose(inputFile);
        return 1;
    }
    while ((num = getw(inputFile)) != EOF) {
        putw(num, outputFile);
    }
    fclose(inputFile);
    fclose(outputFile);
    printf("File copied successfully!\n");
    return 0;
}
```

fprintf and fscanf Functions

- The fprintf() and fscanf() functions in C are used for formatted input and output operations with files. They are part of the standard I/O library (stdio.h).
- **int fprintf(FILE *stream, const char *format, ...):**
  - Writes formatted data to the specified output stream.
  - The stream parameter is a pointer to a FILE object representing the output stream to which the data will be written.
  - The format parameter is a format string that specifies how subsequent arguments are formatted and written to the stream, similar to printf().
  - Returns the number of characters written, or a negative value if an error occurs.

fprintf and fscanf Functions

- **int fprintf(FILE *stream, const char *format, ...):**

```
int num = 42;
double pi = 3.14159;
FILE *fp;
fp = fopen("output.txt", "w");
if (fp != NULL) {
   fprintf(fp, "Integer: %d, Pi: %f\n", num, pi);
   fclose(fp);
}
```

fprintf and fscanf Functions

- **int fscanf(FILE *stream, const char *format, ...):**
  - Reads formatted data from the specified input stream.
  - The stream parameter is a pointer to a FILE object representing the input stream from which the data will be read.
  - The format parameter is a format string that specifies how the input data should be interpreted and read from the stream, similar to scanf().
  - Returns the number of input items successfully matched and assigned, or EOF if the end of the file is reached or an error occurs.

fprintf and fscanf Functions

- **int fscanf(FILE *stream, const char *format, ...):**

```
int num;
double pi;
FILE *fp;
fp = fopen("input.txt", "r");
if (fp != NULL) {
    fscanf(fp, "Integer: %d, Pi: %lf\n", &num, &pi);
    printf("Read Integer: %d, Pi: %f\n", num, pi);
    fclose(fp);
}
```

fprintf and fscanf Functions

- **C Program using fscanf and fprintf functions:**

- Here's is a simple C program that demonstrates the usage of fscanf() to read formatted data from a file and fprintf() to write formatted data to another file.

```c
#include <stdio.h>
int main() {
    FILE *inputFile, *outputFile;
    int num1, num2;
    double result;
    inputFile = fopen("input.txt", "r");
    if (inputFile == NULL) {
        perror("Error opening input file");
        return 1;
    }
    outputFile = fopen("output.txt", "w");
    if (outputFile == NULL) {
        perror("Error opening output file");
        fclose(inputFile);
        return 1;
    }
    if (fscanf(inputFile, "%d %d", &num1, &num2) == 2) {
        result = num1 + num2;
        fprintf(outputFile, "Sum: %.2f\n", result);
    } else {
        printf("Failed to read input from file.\n");
    }
    fclose(inputFile);
    fclose(outputFile);
    printf("Sum calculated and written to output.txt.\n");
    return 0;
}
```

THE END