C Programming

File Input/Output (I/O)

Reading a string of characters from a file

The function **fgets()** reads a string of characters from a file. It is the file equivalent of gets().

fgets (string_array, max_characters_read, file_pointer);

fgets() reads characters from the file into an array (i.e., a char array) until one of the following occurs:

1. a newline character is read, i.e., '`\n`'

2. the end of the file is reached

3. `max_characters` - 1 are read

In all cases, the NULL character, i.e., '\0', is placed in the array after the last character is read.

```
/*
Program to read a string from a file and display to standard
output
*/
#include <stdio.h>

#define MAX_CHARS 81

int main()
{
   //Create a file pointer
   FILE *fp_in;

   // Array to store the string from the file
```

```
    char one_line[MAX_CHARS];

    //Open the file called file.txt for reading
    //and check if this is successful
    if ( (fp_in = fopen("file.txt", "r")) == NULL )
    {
        printf("\nError opening file");
    } // end if
    else
    {
        /*
        Read at most (i) MAX_CHARS - 1 characters from the file
        or (ii) until a new line character is read or (iii) the
end of the file is reached
        */
        while( fgets(one_line, MAX_CHARS, fp_in) != NULL)
        {
            printf("%s", one_line);

        } // end while

        // Close the file once finished
        fclose(fp_in);

    } // end else

    return 0;

} // end main()
```

Repl 25.1: https://replit.com/@michaelTUDublin/251-Read-a-string-from-File#main.c

Writing a string of characters to a file

The function **fputs()** writes a string of characters to a file opened for writing. The format is:

fputs (string_array, file_pointer);

The NULL character ('\0') is not written to the file and unlike the keyboard equivalent of `puts()`, `fputs()` does not add the newline character ('\n') to the end of the string.

Let's take a look at fputs():

```
/*
Program to read and write one line at a time, i.e., a string at a
time from one file to another file. In essence, copying a file to
another one string at a time
*/
#include <stdio.h>

#define MAX_CHARS 81

int main()
{
   //Create a file pointer
   FILE *fp_in, *fp_out;

   // Array to store the string from the file
   char one_line[MAX_CHARS];

   //Open the file called file.txt for reading
   //and check if this is successful
   if ( (fp_in = fopen("file.txt", "r")) == NULL )
   {
       printf("\nError opening file");
   } // end if
   else if( (fp_out = fopen("new.txt", "w") ) != NULL )
   {
       /*
       Read at most (i) MAX_CHARS - 1 characters from the file
       or (ii) until a new line character is read or (iii) the
end of the file is reached. Write the string to a new file called
"new.txt"
       */
       while( fgets(one_line, MAX_CHARS-1, fp_in) != NULL)
       {
           fputs(one_line, fp_out);

           // Used simply to display the copied string to
standard output, which has been written to new.txt
           printf("%s", one_line);

       } // end while

       // Close the file once finished
       fclose(fp_in);
       fclose(fp_out);

   } // end else if
   else
   {
       printf("\nError opening/writing to new file");
   } // end else

   return 0;

} // end main()
```

Repl 25.2: https://replit.com/@michaelTUDublin/252-Write-a-string-to-a-File#main.c

Note: you can use the fseek() function to move the file pointer to a specific location in the file and read/write from that point.

fseek (file_pointer, offset_no_of_bytes, SEEK_CUR/SEEK_END);