FIT ICT Software Development

Lecture 11

Files

Lecturer : Charles Tyner

FIT ICT Software Development

Handling Files

All standard functions for file i/o contained in ***stdio.h*** header file

File pointer is a special data type for handling files

Used to open, read from, write to and close files

Syntax: **FILE *fp** where *fp* is name of pointer

A file pointer points to a structure in **stdio.h** file that contains information about the file

This includes whether file is being read or written (mode) and current character

FIT ICT Software Development

Handling Files

File is opened with *fopen()* function

Takes two arguments:

- name and location of file
- mode in which to open file

Function returns file pointer if successful, or NULL if not

Once opened, file can be read, overwritten or added to, depending on mode

When completed, file must be closed with *fclose()*, which takes file pointer as its only argument

FIT ICT Software Development

File Modes – a Table

File mode:	Operation:
r	Open an existing file to read
w	Open an existing file to write. Creates a new file if none exists or opens an existing file and discards all its previous contents
a	Appends text. Opens or creates a text file for writing at the end of the file
r+	Opens a text file to read from or write to
w+	Opens a text file to write to or read from
a+	Opens or creates a text file to read from or write to at the end of the file

Where the mode includes a **b** after any of the file modes listed above, the operation relates to a binary file rather than a text file. For example, **rb** or **w+b**

FIT ICT Software Development

Handling Files

```
#include <stdio.h>

int main()
{
 FILE *file_ptr;
 file_ptr = fopen( "c:\\documents\\data.txt", "w" );
 if(file_ptr != NULL )
 {
   printf( "File created\n" );
   fclose( file_ptr );
   return 0;
  }
  else
  {
    printf( "Unable to create file\n");
    return 1;
  }
}
```

FIT ICT Software Development

Handling Files

Header file ***stdio.h*** is included and Main function opened

File pointer declared and then ***fopen()*** is used with arguments to create a named file in a particular location and prepare it for writing

N.B. Both name/location and mode must be in double quotes

Code then checks whether attempt was successful and appropriate message output using ***printf()*** function and a zero or one returned

Save, compile, execute and study. Check to see if file created.

FIT ICT Software Development

File Reading & Writing Functions

Standard input function ***scanf()*** and standard output function ***printf()*** are simplified versions of functions ***fscanf()*** and ***fprintf()*** respectively

Simplified versions do not require argument specifying filestream, but assume keyboard (scanf) and monitor (printf)

scanf(…) = fscanf(stdin, …)

printf(…) = fprintf(stdout, …)

Full versions require filestream identifier as argument

FIT ICT Software Development

File Reading & Writing Functions

	Read Function	Write Function
One character at a time	fgetc()	fputc()
One line at a time	fgets()	fputs()
Entire Filestream	fread()	fwrite()
Filestream with strings and numbers	fscanf()	fprintf()

FIT ICT Software Development

Reading & Writing Characters

```
#include <stdio.h>

int main()
{
 FILE *file_ptr; int i;
 char text[50] = { "Text, one character at a time." };
 file_ptr = fopen( "c:\\documents\\chars.txt", "w" );
 if(file_ptr != NULL )
 {
   printf( "File chars.txt created\n" );
   for( i = 0; text[i]; i++)
   {
     fputc( text[i], file_ptr ) ;
   }
   fclose( file_ptr );
   return 0;
 }
 else
 {
   printf( "Unable to create file\n");
   return 1;
 }
}
```

FIT ICT Software Development

Reading & Writing Characters

Header file ***stdio.h*** is included and Main function opened

File pointer, an integer counter and a character array are declared and then ***fopen()*** is used with arguments to create a named file in a particular location and prepare it for writing

Code checks whether attempt was successful and appropriate message output using ***printf()*** function and a zero or one returned

If attempt was successful, code loops through the character array and uses ***fputc()*** to write each character to the created file

Function ***fclose()*** then deployed to close the file

Save, compile, execute and study. Check to confirm file created and check its contents.

FIT ICT Software Development

Reading & Writing Lines

Functions ***fgetc()*** and ***fputc()*** require repeated calls

Not very efficient

Better to use ***fgets()*** and ***fputs()***

Function ***fgets()*** takes 3 arguments: char array (or pointer) to which text will be assigned; int specifying maximum number of characters per line; and file pointer, saying where to read from

Function ***fputs()*** takes 2 arguments: text to write; and file pointer for target file. Newline character automatically added

FIT ICT Software Development

Reading & Writing Lines

As preparation for the next program, create a text file containing the following:

> I WILL arise and go now, and go to Innisfree,
> And a small cabin build there, of clay and wattles made:
> Nine bean-rows will I have there, a hive for the honey-bee,
> And live alone in the bee-loud glade.

(Copy and paste from online source to save typing but make sure there is a newline character at the end)

Save to c:\documents as inisfree.txt

N.B. Depending on your version of Windows, you may have trouble specifying location for the file and finding it for editing. Experiment!

FIT ICT Software Development

Reading & Writing Lines

```
#include <stdio.h>
#include <string.h>

int main()
{
 FILE *file_ptr; int i;
 char text[50];
 file_ptr = fopen( "c:\\documents\\inisfree.txt",  "r+a" );
 if(file_ptr != NULL )
 {
   printf( "File inisfree.txt opened\n" );
   while(fgets(text, 50, file_ptr) )
   {
     printf( "%s", text) ;
   }
   strcpy ( text, "…by W.B.Yeats");
   fputs( text, file_ptr ) ;     fclose( file_ptr );
   return 0;
 }
 else
 {
    printf( "Unable to open file\n");
    return 1;
 }
}
```

FIT ICT Software Development

Reading & Writing Lines

Header files ***stdio.h*** and ***string.h*** are included and Main function opened

File pointer and a character array are declared and then ***fopen()*** is used with arguments to create a named file in a particular location and prepare it for reading and appending

Code checks whether attempt was successful and appropriate message output using ***printf()*** function and a zero or one returned

If attempt was successful, code loops through the text line by line until the EOF character is returned to end the loop. On each loop, the code prints to the screen the line read

It uses ***fputs()*** to add a line identifying the author

Function ***fclose()*** then deployed to close the file

Save, compile, execute and study. Check to confirm line is added naming author .

FIT ICT Software Development

Reading & Writing Files

Whole files can be read and written using functions *fread()* and *fwrite()*

Each takes same 4 arguments:

- *char* variable where text can be stored
- *int* specifying size of text to read/write
- *int* specifying total number of characters to read
- file pointer of file to work with

Array storing text must be large enough to hold entire content

FIT ICT Software Development

Reading & Writing Files

Functions *fread()* and *fwrite()* return the number of objects read or written

This is a count of characters, spaces and newlines, each representing a single object

Value returned by *fread()* can be used as third argument in *fwrite()* to ensure that number of objects written is the same as number of objects read

Create a text file called *source.txt* containing text of your choice. Some lines in the following code may have to be adapted depending on file location

FIT ICT Software Development

Reading & Writing Files

```
#include <stdio.h>

int main()
{
 FILE *source_ptr;
 FILE *target_ptr;
 char buffer[1000];
 int num;
 source_ptr = fopen( "c:\\documents\\source.txt", "r" );
 target_ptr = fopen( "c:\\documents\\target.txt", "w" );

 if(source_ptr != NULL ) && (target_ptr != NULL )

 {
```

FIT ICT Software Development

Reading & Writing Files

```
  num = fread( buffer , 1 , 1000 , source_ptr ) ;
  fwrite( buffer , 1 , num , target_ptr ) ;
  fclose(source_ptr);
  fclose(target_ptr);
  printf( "Done: source.txt copied to target.txt" );
  printf( "\n%dobjects copied. \n" , num );
  return 0;
 }
 else
 {
   if( source_ptr == NULL) printf( "Unable to open source.txt\n" );
   if( target_ptr == NULL) printf( "Unable to open target.txt\n" );
   return 1;
 }
}
```

FIT ICT Software Development

Reading & Writing Files

Header file ***stdio.h*** included and Main function opened

Two file pointers, a character array and an integer are declared and then ***fopen()*** is used with arguments to first open an existing file for reading and then create/open another file for writing

Code checks whether attempt was successful

If not, checks which file returned NULL and prints a message for either or both, before returning a one to signify failure

If attempt was successful, code uses ***fread()*** not only to populate the buffer, but also to assign a value to the integer variable

This value is used as third argument of call to ***fwrite()*** and both files are then closed

Messages are output to the monitor and a zero is returned as required

Save, compile, execute and study. Check to confirm file has been copied

FIT ICT Software Development

Scanning Files

As we have seen, *fscanf()* and *fprintf()* allow nomination of a filestream to read from or write to

This is the first argument for each function

Can use **stdin** with scan function to specify keyboard input and **stdout** with print function to specify output to monitor

Otherwise can specify a file

Scan function also allows numbers in a text file to be converted to their numeric type

Create a file called nums.txt containing numbers 1 – 10, each separated by a space

FIT ICT Software Development

Scanning Files

```
#include <stdio.h>
int main()
{
/* Declare 2 file pointers and 3 integer variables, one an array*/
 FILE *nums_ptr, *hint_ptr ;
 int nums[20] , i , j ;
 /* Open files for reading and writing*/
 nums_ptr = fopen( "c:\\documents\\nums.txt", "r" );
 hint_ptr = fopen( "c:\\documents\\hint.txt", "w" );
/* Check whether files have opened correctly */
 if( (nums_ptr != NULL ) && (hint_ptr != NULL ) )
 {/* If opened correctly, loop through the original file. Note use of
   function feof() to check whether end of file has been  reached*/
    for (i = 0 ; !feof( nums_ptr) ; i++ )    { fscanf( nums_ptr , "%d" , &nums[i] ) ;  }
```

FIT ICT Software Development

Scanning Files

```
/* Output array element values*/
   fprintf( stdout, "\nTotal numbers found: %d\n" , i );
   for( j = 0 ; j < i ; j++) {fprintf( stdout, "%d" , nums[j] ); }
/* Print array element values into a file*/
   fprintf( hint_ptr, "fscanf and fprintf are flexible\n" );
   for( j = 0 ; j < i ; j++) {fprintf(hint_ptr, "%d" , nums[j] ); }
/* Close both files on completion*/
   fclose(nums_ptr);
   fclose(hint_ptr);
   return 0;
  }
  else /* If not opened correctly */
  {
    printf( "Unable to open a file\n" );   return 1;
  }
}
```