BIRZEIT UNIVERSITY

Working with Files

Computer Science Department

Files vs. File Variables

A file variable is a data structure in the C program which represents the file

- Temporary exists only when program runs
- There is a struct called **FILE** in <stdio.h>
- Details of the struct are private to the standard C I/O library routines

What's in *stdio.h?*

Prototypes for I/O functions.

Definitions of useful *#define* constants
Example: EOF for End of File

Definition of *FILE struct* to represent information about open files.

File variables in C programs are **pointers** to a FILE *struct*.

FILE *myfile;

V-3

Opening A File

"Opening" a file: making a connection between the operating system (file name) and the C program (file variable)

Files must be opened before they can be used

V-4

Opening A File

To open a disk file in C:
- library function ***fopen***
- specify "r" (read, input) or "w" (write, output)
  - NB String "r", not char 'r' !

Files stdin/stdout (used by scanf/printf) are automatically opened & connected to the keyboard and display

V-5

File Open Example

```
/*usually done only once in a program*/
/*usually done near beginning of program*/

FILE *infilep, *outfilep;  /*file variables*/
char ch;

/* Open input and output files */
infilep   = fopen ("Student_Data.txt",        "r" ) ;
outfilep = fopen ("New_Student_Data.txt", "w") ;
```

V-6

File I/O: *fscanf* and *fprintf*

Once a file has been opened...

use *fscanf* and *fprintf* to read or write data from/to the file

Use the file variable returned by *fopen* to identify the file to be read/written

File must already be open before *fscanf* or *fprintf* is used!

V-7

File I/O: *fscanf* and *fprintf*

fscanf: works just like *scanf*, but 1st parameter is a *file variable*

```
fscanf (filepi, "%...", &var, ... ) ;
```

fprintf: works just *printf*, but 1st parameter is a file variable

```
fprintf (filepo, "%...", var, ... ) ;
```

V-8

Files - Summary

- Declare a file pointer variable
  - FILE *ftp_in , /* pointer to input file */
  - FILE *ftp_out; /* pointer to output file */
- The calls to function fopen
  - ftp_in = fopen("distance.dat", "r") ;
  - ftp_out = fopen("distance.out", "w") ;
- Use of the functions
  - fscanf(ftp_in, "%lf", &miles);
  - fprintf(ftp_out, "The distance in miles is %.2f. \n", miles);
- End of use
  - fclose(ftp_in);
  - fclose(ftp_out);

Files (Examples)

1. Write a program to read two integers from a file (input.txt), find the **sum** of them and save the result into another file (output.txt).

2. Repeat the above example, but *print the result on the screen instead of saving the result of the file*.

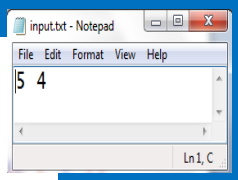
Files (Example 1 solution)

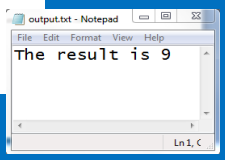
```c
#include <stdio.h>
int main()
{
    FILE *fpt_in, *fpt_out;
    int num1,num2;
    int sum;

    fpt_in = fopen ("input.txt","r");
    fpt_out = fopen ("output.txt","w");

    fscanf (fpt_in,"%d%d",&num1,&num2);
    sum=num1+num2;

    fprintf(fpt_out,"The result is %d",sum);
    fclose(fpt_in);
    fclose(fpt_out);
    return 0;
}
```





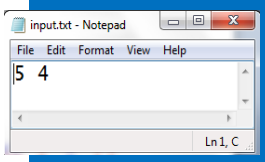
Files (Example 2 solution)

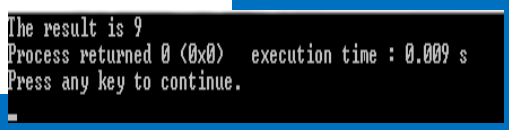
```c
int main()
{
    FILE *fpt_in;
    int num1,num2;
    int sum;

    fpt_in = fopen ("input.txt","r");

    fscanf (fpt_in,"%d%d",&num1,&num2);
    sum=num1+num2;

    printf("The result is %d",sum);
    fclose(fpt_in);
    return 0;
}
```





End-file-Controlled Loops

End-file-Controlled Loops

Repetition statement is very similar to the sentinel controlled loop that uses the status value returned by the scanning function to control repetition rather than using the values scanned.

1. Get the first *data value* and save *input status*
2. while *input status* does not indicate that end of file has been reached
    3. Process *data value*
    4. Get next *data value* and save *input status*

The loop repetition condition: **input_status != EOF**

input_status = scanf("%d%d%lf", &part_id, &num_avail, &cost);

scanf function returns as its value the number of data items scanned Here 3

Example: Write a C program that reads the integers stored in a text file

```c
#include <stdio.h>
int
main()
{
    int m = 0, n, k = 0;
    FILE *fptr;
    fptr = fopen("c:\\Code\\numbers.dat", "r");
    if (fptr != NULL)
    {
        printf("\nFile numbers.dat is opened successfully.");
        printf("\nContents of file numbers.dat:");
        m = fscanf(fptr, "%d", &n);

        while(m != EOF)
        {
            printf("%d ", n);
            m = fscanf(fptr, "%d", &n);
        }
        printf("\n");
        k = fclose(fptr);
        if(k == -1)
            printf("\nFile-closing failed");
        if(k == 0)
            printf("\nFile is closed successfully.");
    }
    else
        printf("\nFile-opening failed");
    return(0);
}
```