

DAILY ASSESSMENT FORMAT

Date:	20/06/2020	Name:	Nichenametla Bhargavi
Course:	C Programming	USN:	4AL17EC061
Topic:	1: Files & Error Handling 2: The Processors	Semester & Section:	6th Sem A Sec
Github Repository:	Bhargavi_Nichenametla		

SESSION

Image of the session

The screenshot shows a mobile interface for a Sololearn session. At the top, there's a navigation bar with a home icon, a 'Lite' badge, the URL 'vw.sololearn.com', a notification icon with '23', and a menu icon. Below this is a blue progress bar with several question marks and a '1' at the end. The main content area is titled 'Binary File I/O' and contains text explaining binary file operations. It lists mode options for `fopen()`: `rb` (read), `wb` (write), `ab` (append), `rb+` (read/write), `wb+` (write/overwrite), and `ab+` (write/append). It also defines `fwrite()`, `fread()`, `fclose()`, and `feof()`. A yellow highlight box contains the text: `feof(fp)` Returns 0 when the end of the file stream has been reached. At the bottom, there's a 'Q&A' section with a '16' badge and a green arrow button.

Binary File I/O

Writing only characters and strings to a file can become tedious when you have an `array` or `structure`. To write entire blocks of memory to a file, there are the following binary functions:

Binary file mode options for the `fopen()` function are:

- `rb` open for reading (file must exist)
- `wb` open for writing (file need not exist)
- `ab` open for append (file need not exist)
- `rb+` open for reading and writing from beginning
- `wb+` open for reading and writing, overwriting file
- `ab+` open for reading and writing, appending to file

`fwrite(ptr, item_size, num_items, fp)` Writes `num_items` items of `item_size` size from `pointer ptr` to the file pointed to by file `pointer fp`.

`fread(ptr, item_size, num_items, fp)` Reads `num_items` items of `item_size` size from the file pointed to by file `pointer fp` into memory pointed to by `ptr`.

`fclose(fp)` Closes file opened with file `fp`, returning 0 if close was successful. `EOF` is returned if there is an error in closing.

`feof(fp)` Returns 0 when the end of the file stream has been reached.

33 COMMENTS

Q&A

Report – Report can be typed or hand written for up to two pages.

Exception Handling:

- Central to good programming practices is using error handling techniques.
- Even the most solid coding skills may not keep a program from crashing should you forget to include exception handling.
- An exception is any situation that causes your program to stop normal execution.
- Exception handling, also called error handling, is an approach to processing runtime errors.
- C does not explicitly support exception handling, but there are ways to manage errors:-
 1. Write code to prevent the errors in the first place. You can't control user input, but you can check to be sure that the user entered valid input.
 2. When performing division, take the extra step to ensure that division by 0 won't occur.

Use the exit statement to gracefully end program execution. You may not be able to control if a file is available for reading, but you don't need to allow the problem to crash your program.

Use `errno`, `perror()`, and `strerror()` to identify errors through error codes.

The "perror and strerror" Functions:

When a library function sets `errno`, a cryptic error number is assigned. For a more descriptive message about the error, you can use `perror()`. You can also obtain the message using `strerror()` in the `string.h` header file, which returns a pointer to the message text.

`perror()` must include a string that will precede the actual error message. Typically, there is no need for both `perror()` and `strerror()` for the same error, but both are used in the program below for demonstration purposes:

```
FILE *fptr;
errno = 0;

fptr = fopen("c:\\nonexistantfile.txt", "r");
if (fptr == NULL) {
    perror("Error");
    fprintf(stderr, "%s\n", strerror(errno));
    exit(EXIT_FAILURE);
}
```

File Handling in C:

- So far the operations using C program are done on a prompt / terminal which is not stored anywhere. But in the software industry, most of the programs are written to store the information fetched from the program.
- One such way is to store the fetched information in a file. Different operations that can be performed on a file are:
 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 fgets).
 4. Writing to a file (fprintf or fputs).
 5. Moving to a specific location in a file (fseek, rewind).
 6. Closing a file (fclose).

Certificate for completion of C Programming:

