files

Monday, 28 October 2024 4:21 PM

What is fopen? What does it do?
FILE *fopen(**const** char *pathname, **const** char *mode);
Opens a file for you, returns a file pointer

Mode - what we want to do with the file
"r" read
"w" write to the file
"a" add to an existing file

When will fopen fail? What does it return?

How to print what specifically made fopen fail?

File input/output functions

Function	Returns	What it does	When to use
`fprintf(FILE *stream, "%d\n", x)`		Prints the format string with variables inserted to file `stream`	Formatted output
`fscanf(FILE * stream, "0x%x\n", &x)`	Number of items read	Reads variables in the format from the string, stores them in addresses.	When you know the format of the input.
`Fgetc(FILE * stream)`	Character read from stream, EOF on error	Read in a single byte from `stream`	Read one byte/character at a time
`Fputc(int c, FILE * stream)`		Prints `c` (as a byte) to `stream`.	Write one byte/character at a time
`Fgets(char string[], int size, FILE *stream)`	NULL on error, pointer to string on success	Reads a line at a time into the `string` array, up to `size` bytes.	Read one line at a time
`Fputs(char string[], FILE *stream)`		Prints `string` to the `stream`.	Write one line at a time

Should you use fgets/fputs with binary data?

What does the following printf statement display?
```
printf ("%c%c%c%c%c%c", 72, 101, 0x6c, 108, 111, 0x0a);
```

How many values can fgetc return?
Read in byte -> 00000000, 11111111 (0, 255)
Int, so that it can return EOF = -1
256 values, 0-255, -1

Why are the names of fgetc, fputc, getc, putc, putchar, and getchar misleading?
They actually deal with bytes, fgetb, fputb etc.

For each of the following calls to the `fopen()` library function, give an open() system call that has equivalent semantics relative to the state of the file.

```
a. fopen(FilePath, "r")
b. fopen(FilePath, "a")
c. fopen(FilePath, "w")
d. fopen(FilePath, "r+")
e. fopen(FilePath, "w+")
```