2510 · Lecture 10 || February 1/201

☐ assignment 1 - File I/O due friday before finals

- filename is just a string
  ex. "data.txt";
  "c:\newfile.txt"
  ↳ newline char

- mode is also a string → there are 12 modes (divided into 2 groups of 6)

TEXT MODE

"r"	open for reading; file must exist!	≠ w
"w"	Create or truncate file for writing	≠ r
"a"	create or open file for appending	
"rt"	open for updating; file must exist	
"wt"	create or truncate file for updating	
"at"	create or open file for updating, for appending	

"truncate" = delete content

"appending" = writing at the END

UPDATING + both read and write

BINARY MODE

"rb" "wb" "ab" "r+b" "w+b" "a+b"

"rbt" "wbt" "abt"

txt vs binary mode

- no difference in Unix
- in Windows, there is special handling of \n in text mode

↗ Cont.

... Lecture 10

cont... text mode in Windows

In C program		in the file
`'\n'`	⟶	`'\r' '\n'`
any other char, c	⟷	c

only \n char becomes 2 chars

▷ Writing and Reading a stream

fprintf / fscanf
fgets ex. fgets (line, LINESIZE, fp)

ex. Summing integers read from a file

```
int n, sum = 0;
/* code to open file omitted */
while (fscanf (fp, "%d", &n) == 1)
    sum += n;
```

file content: 123␣ 456␣ 11␣

a FILE *

- closing a file

Standard idiom to close a file

```
if (fclose (fp) != 0) {
    perror ("fclose");
    /* additional error-handling if needed */
}
```

↳ actually, we close a STREAM
- this disassociate the stream from the file
- this flushes output buffer associated w/ stream

NOTE: most modern OS automatically closes file when program exits

ex. Program to copy a file [user will specify file in cmd line]

```
$./copy file1 file2
```

```
#include <stdio.h>
int main (int argc, char * argr [ ]) {
    FILE *ifp, *ofp;

    if (argc != 3) {
        fprintf (stderr, "usage: %s [source] [destination] \n",
        return 1;                                    argv [0]);
    }
    if ((ifp = fopen (argv [1], "rb")) == 0 ) {
        perror ("fopen");
        return 2;
    }
    if ((ifp = fopen (argv [2], "wb")) == 0 ) {
        perror ("fopen");
        return 3;
    }
    while ((c = fgetc (ifp)) != EOF)
        fputc (c, ofp);
    if (fclose (ifp) != 0 ) {
        perror ("fclose");
        return 4;
    }
    if (fclose (ofp) != 0 ) {
        perror ("fclose");
        return 5;
    }
    return 0;
}
```

* is repeated

PROGRAM needs to check the cmd args or else it will crash!