2510 : Lecture 7

Some other output functions :

```
putchar(c)          ex. putchar('a');
fputc(c,file)       ex. fputc('a', stderr);
puts(s)             ex. puts("hello");
                        /* note:
                           this prints a newline char */
```

Input

↳ more complicated than output as it may fail
↳ [ALWAYS CHECK THE RETURN VALUE OF AN INPUT FUNCTION IMMEDIATELY AFTER CALLING]

4 ways of performing input :

1. character by character
2. line by line
3. data item by data item
4. block by block

▷ Reading Character By Character

```
int c;
while ((c = getchar()) != EOF)
    putchar(c);
```

- (c = getchar()) != EOF) bracket are necessary
  b/c = lower than
  != than =

... Lecture 7

▷ Char by char example cont.

c = getchar() != EOF is equivalent to

c = (getchar() != EOF) /* c can be 0 or 1 */

generally, a + b * c = a + (b * c)
* higher than +

- EOF is the special value returned by getchar on read error or on end-of-file
  - ↳ end-of-file is a condition; it becomes true when we attempt to read past the last byte of the file
    (Note: after reading last byte, end-of-file is not true yet)

  - ↳ from the keyboard, we can generate the end-of-file condition CTRL-D

    not ctrl-c
    ↳ kills program

- c is declared as int because getchar needs to be able return ALL possible characters values plus EOF to handle/indicate failure

(type must be bigger than char)

...Lecture 7 ||

- getchar() returns an unsigned char as an int

↳ note: keyboard input is typically line buffered. C program doesn't get the input until a line is entered

- the loop can be used to copy a file using I/O redirection

Standard idiom to process stdch char by char

```
int c;
while ((c = getchar()) != EOF)
    /* process c */
```

Examples

① displaying a file in all uppercase

```
int main() {
    int c;
    while ((c = getchar()) != EOF)
        putchar(toupper(c));
    return 0;
}
```

② counting the number of lines in a file

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

int main(void) {
    int c;
    size_t nlines = 0;

alex

    while ((c = getchar()) != EOF)
        if (c = '\n')
            nlines++;
    printf("%lu\n", (unsigned long) nlines);
        return 0;
}
```

< getchar: c OUTPUTS # of char

↳ this is incorrect if the very last line doesn't have a newline

SOLUTION: count the START of the lines