2510 : Lecture 9 January 30, 201

- scanf / fscanf / sscanf returns the number of assignment

- Examples

```
int a = ∅, b = 1, c = 2;
double f;
char   s[1∅∅];
1. a = sscanf ("␣ 12 ␣ 34a", "%d %d", &b, &c);
                ↑     ↑   ↑
```

skips whitespace + non-integers

a = 2 (# of assignmen
b = 12
c = 34

```
2. a = sscanf ("␣ 12ab34", "%d %d ", &b, &c);
                    ↑
```

stops b/c it sees ab

a = 1
b = 12
c = unchanged

```
3. a = sscanf ("␣ 12.34 xy", "%d %lf", &b, &f);
```

a = 2
b = 12
f = 0.34

```
4. a = sscanf ("12xyz 34", "%d %s %d ", &b, &s, &c);
```

s doesn't stop until spaces

a = 2
s = xyz 34
c = unchanged

Examples cont.

5. a = sscanf ("hello", "%d", &b);

a : ∅

can't scan chars b : unchanged

6. a = sscanf ("␣␣", "%d", &b);

a : EOF

if there is nothing to scan the function returns EOF b : unchanged

- Don't use scanf for interactive input
  → easy to get into infinite loop
  ∴ Use fgets together with sscanf instead

- Examples : Summing integers obtained by user interactively

```
int n, sum=∅;                    #define LINESIZE 1024
char line [LINESIZE];
while (1) {
    printf ("Enter an integer:  ");
    if (! fgets (line, LINESIZE, stdin)) {
        clearerr(stdin);
        break;
    }
    if (sscanf (line, "%d", &n) ==1)
        sum + =n;
}
printf ("%d\n", sum);
```

breaks out of loop when end of file!

if we can read #, then add numbers

...Lecture 9

▷ Reading Block by Block

fread / fwrite
  └ for writing

```
ex. int a [1ØØ];
    size_t n;
    n = fread (a, size of (a [Ø]), 1ØØ, fp);   used to read binary data
```
stream

/* returns the # of elements read */

```
    fwrite (a, sizeof (a [Ø]), 1ØØ, fp);
```

- size of
  - the ANSI C standard does not specify the exact size of each type
  - it provides the sizeof operator that can be used to find the size of different objects + types
  ↳ by definition, size of (char) = 1 byte

```
ex. size of (int)
    int n;
    size of (n)
```

- file I/O
  We needs 3 steps when dealing w/ files
  ① open the file
  ② perform I/O on the stream return from opening the file
  ③ close the file

...Lecture 9||

• File I/O

1. Opening a file
   ↳ when we open a file, we associate a stream with it.

Standard Idiom to open a file

pointer

or NULL

```
FILE *fp;
if (fp = fopen(filename, mode)) == ∅ )
   perror ("fopen");
   /* additional error -handling */
```

- a stream has type FILE * in C
  eg. stdin is a FILE

- perror is used to print a system error message to stderr
  ↳ It can only be used if a standard library function fails and it sets ~~error~~ errno

Feb 1st

We can think of errno as a global integer variable
- when certain functions in the standard library fails, they store an error code into ERRNO
- perror looks at the error code in errno