printf

<table>
<thead>
<tr>
<th>Character</th>
<th>Printed as...</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%c</td>
<td>Character (문자)</td>
<td>A</td>
</tr>
<tr>
<td>%d, %i</td>
<td>Decimal integer</td>
<td>-123</td>
</tr>
<tr>
<td>%u</td>
<td>Unsigned decimal integer</td>
<td>4294967173</td>
</tr>
<tr>
<td>%o</td>
<td>Unsigned octal integer</td>
<td>37777777605</td>
</tr>
<tr>
<td>%x, %X</td>
<td>Unsigned hexadecimal integer</td>
<td>ffffffff85, FFFFFFF85</td>
</tr>
<tr>
<td>%e, %E</td>
<td>Floating point number with e or E</td>
<td>5.284100e-01, 5.284100E-01</td>
</tr>
<tr>
<td>%f</td>
<td>Floating point number (precision default : 6)</td>
<td>0.528410</td>
</tr>
<tr>
<td>%g, %G</td>
<td>Shorter format between %f and %e</td>
<td>0.52841</td>
</tr>
<tr>
<td>%s</td>
<td>String (문자열) (hexadecimal)</td>
<td></td>
</tr>
<tr>
<td>%p</td>
<td>Pointer address</td>
<td></td>
</tr>
<tr>
<td>%%</td>
<td>Print “%”</td>
<td></td>
</tr>
</tbody>
</table>
char str[] = "Blue moon!"

<table>
<thead>
<tr>
<th>Character</th>
<th>Printed as...</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%12s</td>
<td>Field width 12, right-adjusted</td>
<td>&quot;__Blue moon!&quot;</td>
</tr>
<tr>
<td>%.7s</td>
<td>Precision 5</td>
<td>&quot;Blue moo&quot;</td>
</tr>
<tr>
<td>%-11.8s</td>
<td>Precision 8, Field width 11, left-adjusted</td>
<td>&quot;Blue moo__&quot;</td>
</tr>
<tr>
<td>%05d</td>
<td>Padded with zeroes</td>
<td>00123</td>
</tr>
<tr>
<td>%-#9x</td>
<td>0x is attached to hexadecimal number</td>
<td>0x7b___</td>
</tr>
<tr>
<td>%10.5f</td>
<td>Precision 5, Field width 10</td>
<td>&quot;__0.12346&quot;</td>
</tr>
<tr>
<td>%-12.5e</td>
<td>Precision 5, Field width 12, e-format</td>
<td>&quot;1.23457e-01_&quot;</td>
</tr>
</tbody>
</table>
**scanf**

- `int scanf(const char * format, void * input);`
- Return the number of successfully read characters. Return EOF for read error.
- Basically, scanf ignores whitespace characters.
  
  *Whitespace character: space (" "), \n, \t...*

- To let scanf receive whitespace characters, change format.
  Ex) `scanf("%[^\n]", str);` Read until the function meets \n
This is the end of my input toward stdin.

```c
int age;
scanf("%d, &age");

int age = 22;
```

Fetch data before WS!

```bash
root@compute21:/home/seokha/SSE# ./week7
age? 22
grade? age = 22, grade =
```

```bash
root@compute21:/home/seokha/SSE# ```
scanf

```c
int main()
{
    char grade;
    int age;
    printf("age? ");
    scanf("%d", &age);
    printf("grade? ");
    scanf("%c", &grade);
    printf("age = %d, grade = %c\n", age, grade);
}
```

int age = 22;    char grade = ‘\n’;

fetch only one character!
stdin, stdout, stderr

- Standard stream input/output
- Defined in stdio.h
- stdin: connected to the keyboard
- stdout, stderr: connected to the screen
**fopen, fclose**

- FILE * fopen (const char *filename, const char * mode);
- Returns File pointer

<table>
<thead>
<tr>
<th>Mode Character</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>Open for read-only</td>
</tr>
<tr>
<td>w</td>
<td>Create for writing, if exists, overwrite.</td>
</tr>
<tr>
<td>a</td>
<td>Open for appending (Write from the end-of-file)</td>
</tr>
<tr>
<td>r+</td>
<td>Open for update</td>
</tr>
<tr>
<td>w+</td>
<td>Open for update (overwrite if already exists)</td>
</tr>
<tr>
<td>a+</td>
<td>Open for appending</td>
</tr>
</tbody>
</table>

- int fclose(FILE * stream);
- Return 0 if succeeded.
File pointer

- FILE *fp = fopen("./input.txt", "r");

Pointing location can be changed via various functions.
(fscanf, fgetc, fgets, fseek, fread....)

- ftell(fp) : return current offset of fp.
- rewind(fp) : rewind fp to the beginning

Stdin, stdout, stderr : special file pointers.
File seek: fseek

- fseek(fp, 6, SEEK_CUR)

SEEK_SET : From the beginning of the file
SEEK_END : From the end of the file

- ftell(fp) : return current offset of fp.
- rewind(fp) : rewind fp back to the beginning.
fprintf, fscanf

- In stdio.h
  int fprintf(FILE *ofp, const char *format, ...);
  int fscanf(FILE *ifp, const char *format, ...);

```
#include <stdio.h>

int main(){
    FILE * fp;
    char str[20];
    fp = fopen("test.txt", "w+".Cos
    fprintf(fp, "PEN PINEAPPLE
APPLE PEN")..
    rewind(fp);
    fscanf(fp, "%s", str);
    printf("%s\n", str);
    fclose(fp);
}
```

<table>
<thead>
<tr>
<th>Return</th>
<th>If failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of printed characters</td>
<td>-1</td>
</tr>
<tr>
<td>The number of data the function read</td>
<td>EOF</td>
</tr>
</tbody>
</table>

fprintf(stdout, "..."); = printf("...");
fscanf(stdin, ...); = scanf(...);
fgetc, fputc, fgets, fputs

- For one character,
  int fputc (int chr, FILE *stream);
  int fgetc (FILE *stream);

- For the string
  int fputs(const char *str, FILE *stream);
  char *fgets(char *str, int size, FILE *stream)

✓ Make a program recording the sentence you entered through the screen (Using fputs, fgets)

No need do specify the format

<table>
<thead>
<tr>
<th>Return</th>
<th>If failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>character (ASCII)</td>
<td>EOF</td>
</tr>
<tr>
<td>character (ASCII)</td>
<td>EOF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return</th>
<th>If failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive number</td>
<td>EOF</td>
</tr>
<tr>
<td>str</td>
<td>null (str not changed) * at the end of file</td>
</tr>
</tbody>
</table>
fgets vs fscanf

fgets(str[0], 100, fp);
fgets(str[1], 100, fp);

fgets(fp, "%s", str);
fscanf(fp, "%s", str);

Includes whitespace characters.
Stops at \n
Omit whitespace characters
Stop at whitespace characters
fgets vs fscanf

fgets(fp, "%s", str);
fscanf(fp, "%s", str);

fgets(str[1], 15, fp);
fscanf(fp, "%s", str);

fgets(str[1], 15, fp);
fscanf(fp, "%s", str);
## fgets vs fscanf

<table>
<thead>
<tr>
<th>fgets vs fscanf</th>
<th>(\WSn) PEN PINEAPPLE APPLE PEN</th>
</tr>
</thead>
</table>

```c
fscanf(fp, "\%s", str[0]);
```

<table>
<thead>
<tr>
<th>PEN W0</th>
</tr>
</thead>
</table>

```c
fscanf(fp, "\%s", str[1]);
```

<table>
<thead>
<tr>
<th>PINEAPPLE APPLE W0</th>
</tr>
</thead>
</table>

```c
fgets(str[2], 100, fp);
```

<table>
<thead>
<tr>
<th>Wn W0</th>
</tr>
</thead>
</table>

```c
fscanf(fp, "\%s", str[1]);
```

<table>
<thead>
<tr>
<th>APPLE W0</th>
</tr>
</thead>
</table>
```
EOF, feof

- How to recognize we have reached the end of file?
- EOF : End Of File
- fgets returns 0 when they meet the end.

```c
while(fgets(str, sizeof(str), fp) != 0){
    ..... 
}
// do something until the program reads the end of file.
```
EOF, feof

- `feof(FILE * stream);`
- Returns 1 if fp exceeds end of file (!= EOF), 0 if not.

```c
while(!feof(fp)){
    fgets(str, sizeof(str), fp);
    printf("%s", str);
}
```

- What happens?

✓ Make a program adding all the numbers in the file.
fread, fwrite

- `size_t fread (void * ptr, size_t size, size_t count, FILE * stream);`
- `size_t fwrite (const void * ptr, size_t size, size_t count, FILE * stream);`
- Returns successfully conducted number of elements
  Error if count!=return, or EOF
- Not interrupted by whitespace characters.
File access

✓ Make a program double spacing a file.

```
Everywhere that I go, everywhere that I be
If you are not surrounding me with your energy
I don't wanna be there, don't wanna be anywhere
Any place that I can't feel you, I just wanna be near you
And yes, I'm a mess but I'm blessed
To be stuck with you
Sometimes it gets unhealthy
We can't be by ourserlves
We'll always need each other, and
Yes I'm a mess but I'm blessed
To be stuck with you
I just want you to know that
If I could swear I'll go back
Make everything all better
```
Exercise: Censorship

- Censor bad words.
fgets vs fscanf

fgets(str[0], 15, fp);

fgets(str[1], 15, fp);

fscanf(fp, "%^[\n], str[2]);

fscanf(fp, "%s", str[1]);
fgets vs fscanf

fgets(str[0], 15, fp);

fgets(str[1], 15, fp);

fscanf(fp, "%[^\n], str[2]);

fscanf(fp, "%s", str[1]);

fscanf(fp, "%s", str[0]);

fgets(str[1], 15, fp);

fscanf(fp, "%s", str[2]);

fgets(str[3], 15, fp);

(Wn)
PEN PINEAPPLE
APPLE PEN