【Linux Programming】Day14

Tags	
Date	@June 6, 2022
Summary	printf and scanf

【Ch3】Work with Files

3.6 Formatted Input and Output

3.6.1 printf, fprintf, and sprintf

The `printf` family output a variable number of arguments of different types.

The way each is represented in the output stream is controlled by the `format` parameter, which is a string that contains ordinary characters to be printed and codes called conversion specifiers.

```
#include <stdio.h>

int printf(const char *format, ...);
int sprintf(char *s, const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
```

The `printf` function produces its output on the standard output.

The `fprintf` function produces its output on a specified stream.

The `sprintf` function writes its output and a terminating null character into the string `s` passed as a parameter. This string must be large enough to contain all of the output.

Conversion characters always start with a % character. Here is an example:

```
printf("Some numbers: %d,m %d, and %d\n", 1, 2, 3);
```

To print `%`, we can use either `\%` or `%%`.

We can gain greater control over the way items are printed by using field specifiers.

Field specifiers are given as numbers immediately after the `%` character in a conversion specifier.

Format	Argument	|Output|
`%10s`	`"Hello"`	`|     Hello|`
`%-10s`	`"Hello"`	`|Hello     |`
`%10d`	`1234`	`|      1234|`
`%-10d`	`1234`	`|1234      |`
`%010d`	`1234`	`|0000001234     |`
`%10.4f`	`12.34`	`|   12.3400|`
`%*s`	`10,"Hello"`	`|     Hello|`

If we try to print a string longer than the field, the field grows

Format	Argument	|Output|
`%10s`	`"HelloTherePeeps"`	`|HelloTherePeeps|`

3.6.2 scanf, fscanf, and sscanf

The `scanf` family of functions read items from a stream and place values into variables at the addresses they're passed as pointer parameters.

```
#include <stdio.h>

int scanf(const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *s, const char *format, ...):
```

Note: It's very important that the variables used to hold the values are of the correct type and that they match the format string precisely. If not, the memory could be corrupted and our program could crash.

The format string for `scanf` contains both ordinary characters and conversion specifiers. However, the ordinary characters are used to specify characters that must be present in the input.

For example:

```
int num;
scanf("Hello: %d", &num);
```

This call to `scanf` will succeed only if the next five characters on the standard input are `Hello:`.

A space in the format string is used to ignore all whitespace(spaces, tabs, form feeds, and newlines).

Use the `%[]` specifier to read a string composed of characters from a set. The format `%[A-Z]` will read a string of capital letters.

If the first character in the set is `^`, then the specifier reads a string that consists of characters not in the set. For example, to read a string with spaces in it, but stopping at the first comma, we can use `%[^,]`.

In general, scanf are not highly regarded. Try using other functions like `fread` or `fgets`.