CS211: Programing For Operating Systems

Robert Davidson

Contents

1 Intro to C

C is a compiled language, not an interpretive language. Meaning we need a program called a compiler to convert the code into machine code. The compiler is called **gcc**

It is a **very small language and relies heavily on libraries**. The compiler must be told in advance how these functions should be used. So before the compilation process, the **preprocessor** is run to include the function prototypes The compiler then compiles the code into an object file.

1.1 Variables

In C, variables must be declared before they're used. Declarion should have a type to tell compiler what data the variable will hold

- **int** : Integer (1, 2, 3, 4, 5, ...)
- **float** : Floating-point number (7 decimal digits)
- **double** : Double-precision floating-point number (15 decimal digits)
- **char** : Character (a, b, c, ...)
- **void** : No type (used for functions that do not return a value)

1.2 Operators

Operator	Description	Example
+	Addition	`a + b`
-	Subtraction	`a - b`
*	Multiplication	`a * b`
/	Division	`a / b`
%	Modulus	`a % b`

Table 1: Arithmetic Operators

Operator	Description	Example
=	Assignment	`a = b`
+=	Add and assign	`a += b`
-=	Subtract and assign	`a -= b`
*=	Multiply and assign	`a *= b`
/=	Divide and assign	`a /= b`
%=	Modulus and assign	`a %= b`
++	Increment	`a++`
–	Decrement	`a-`

Table 2: Assignment and Arithmetic Assignment Operators

Operator	Description	Example
==	Equal	`a == b`
!=	Not Equal	`a != b`
>	Greater	`a > b`
<	Less	`a < b`
>=	Greater or Equal	`a >= b`
<=	Less or Equal	`a <= b`

Table 3: Relational Operators

Operator	Description	Example
&&	Logical AND	`a && b`
||	Logical OR	`a || b`
!	Logical NOT	`!a`

Table 4: Logical Operators

1.3 Control Flow

1.4 If Else

Listing 1: If-Else

```
int a = 10;
if(a > 10){
        printf("a is greater than 10\n");
    }else if(a == 10){
        printf("a is equal to 10\n");
    }else{
printf("a is less than 10\n");
}
```

Logical opeators, `&&` and `||` can be used to make more complex conditions.

Listing 2: Complex If-Else

```
if(a > 10 && a < 20){
    printf("a is between 10 and 20\n");
}
```

For loop

`for(initial val; continuation condition; increment/decrement){...}`

Listing 3: Print numbers from 0 to 9

```
for(int i = 0; i < 10; i++){
    printf("i is %d\n", i);
}
```

While loop

`while(expression){...}`

Listing 4: Print numbers from 0 to 9

```
int i = 0;
while(i < 10){
    printf("i is %d\n", i);
    i++;
}
```

Do While loop

`do{...}while(expression);`

Listing 5: Print numbers from 0 to 9

```
int i = 0;
do{
    printf("i is %d\n", i);
    i++;
}while(i < 10);
```

1.5 Output

`printf()` is used to print formatted output to the screen. It is a variadic function, meaning it can take any number of arguments. The first argument is a format string, followed by the values to be printed.

The format string may contain a number of escape characters, represented by a backslash. Some of the most common escape characters are:

Sequence	Description
`\a`	Produces a beep or flash
`\b`	Moves cursor to last column of previous line
`\f`	Moves cursor to start of next page
`\n`	New line
`\r`	Carriage return
`\t`	Tab
`\v`	Vertical tab
`\\`	Prints a backslash
`\'`	Prints a single quote

A conversion character is a letter that follows a `%` and tells `printf()` to display the value stored in the corresponding variable. Some of the most common conversion characters are:

Specifier	Description
`%c`	Single character (char)
`%d` or `%i`	Decimal integer (int)
`%e` or `%E`	Floating-point (scientific notation)
`%f`	Floating-point value (float)
`%g` or `%G`	Same as %e/%E or %f, whichever is shorter
`%s`	String (char array)
`%u`	Unsigned int
`%x`	Hexadecimal integer
`%p`	Pointer (memory address)
`%%`	Prints the % character

1.6 Input

`scanf()` reads input from standat input, format it, as directed by a conversion character and store the address of a specified variable.

Listing 6: Reading an integer

```
1    int number;
2    char letter;
3    printf("Enter a number and a char: ");
4    scanf("%d %c", &number, &letter);
5
6    printf("You entered: %d and %c\n", number, letter);
```

- The scan `scanf()` returns an integer equal to the number of successfull conversions made.
- There is related function `fscanf()` that reads from a file. `scanf()` is really just a wrapper for `fscanf()` that treats the keyboard as a file.
- There are other useful functions for readint the standard input stream: `getchar()` and `gets()`.

Listing 7: Check for no input

```
1    int number;
2    printf("Enter a number between 1 and 30: ");
3    scanf("%d", &number);
4
5    while ((number<1) || (number>30))
6    {
7        printf("Invalid number. Please enter a number between 1 and 30: ");
8        scanf("%d", &number);
9    }
```

2 Functions

2.1 Protype and Definition

Protype : A function prototype is a declaration of a function that tells the compiler what the function looks like. It includes the function name, return type, and parameter types. The prototype must be declared before the function is called.

Definition : A function definition is the actual implementation of the function. It includes the function name, return type, parameter types, and the body of the function. The definition must be declared after the function is called.