**Terminal Commands and Basics of C**

### **Useful Timestamps**

Basic Overview - 0:00

Shell commands - 1:50

Hello World in C - 12:33

Compilation and execution in C - 19:35

Primitive Datatypes in C - 24:48

Loops and Conditionals - 32:07

### **Quick Notes**

#### **Shell commands**

Every shell command is of the form: program option(s) argument(s).

In some commands, option(s) and/or argument(s) are absent.

Every option is preceded by a hyphen symbol “-”.

Directory - to crudely put it, a directory is a folder in your computer.

1. **pwd** - stands for *print working directory*. Will print the path of the current directory with respect to the root directory. For me, the root directory was *rootfs*, and thus my path was *home/prashanthi*.
2. **ls** - stands for *list*. Will print a list of files and directories present in the current working directory.
3. **ls -l** - This is the list command with an added option. The option, -l stands for *long* format. Will print a list of files and directories present in the current working directory, but in a long format.
4. **cd** - stands for *change directory*. Will help you move in and out of directories and change your current working directory.

E.g. if Blockchain is a directory in my current working directory,

*cd Blockchain* will change my current directory to Blockchain. To go back to the previous directory - *cd ..* (basically, cd dot dot).

If SastaICP is a directory within Blockchain, this is how I can cd into it from the home directory: *cd Blockchain/SastaICP*

E.g.: *cd ../../* will take me back to the home directory from SastaICP

1. **mkdir** - stands for *make directory*. Will help you create directories**.**

E.g.: *mkdir SastaICP* will create a new directory inside the current working directory.

1. **clear**: will *clear* your console. Very satisfying, no?
2. **rmdir**: stands for *remove directory* (surprise, surprise). Will delete a directory if it is empty.

E.g.: *rmdir SastaICP* will remove/delete the directory SastaICP if it is empty.

1. **rm -r** or **rm - rf**  - program (rm) followed by options (-r or -rf). rm stands for remove. It can be used to remove files and directories.

E.g.: *rm -r(f) SastaICP* will remove the directory SastaICP regardless of whether it is empty or not.

1. **touch** - create files in your current working directory

E.g.: touch hello.c

touch test.txt

touch test.py

1. **nano** - edit files on your terminal

E.g.: nano hello.c

Some other useful basic shell commands here:

<https://www.geeksforgeeks.org/basic-shell-commands-in-linux/>

#### **C vs Python**

#### More like why is C still relevant with all its disadvantages? (Hint: high performance, among other things)

|  |  |
| --- | --- |
| C | Python |
| Low-level language  (closer to the computer language\* than human language) | High-level language  (closer to natural languages than computer language) |
| You need to declare variables with their types. | No need to declare variables. |
| It is a compiled language  (it run in two steps: compilation and execution) | It is an interpreted language  (It executes code line by line and terminates when it finds an error; no extra compilation step) |
| Readability: very low for people not familiar with the syntax. | Very readable, accessible, user-friendly, short learning curve. |
| High performance; consumes less energy and is very, very fast (since it’s easily understood by the computer). | Poor performance; consumes a lot of energy and is very slow (not easily understood by your computer; has to be broken down into a simpler language for the computer to understand). |
| Preferred for complex computations and hardware-related applications.  (computations that involve large numbers and complex operations, e.g.: cryptography) | Not preferred for complex computations because of its poor performance.  (preferred for machine learning etc. because of its ease of use and abstraction+it’s many useful libraries) |
| Limited number of libraries available. You may have to implement most of the functionalities yourself sometimes. | A large repository of libraries available, a lot of data structures; basically makes life easy for programmers. |
| You need to be very specific with regards to datatypes of variables. | Not very stringent when it comes to datatypes of variables. |

\* By computer language, I mean 0s and 1s (binary).

Conclusion: As you may have guessed, both are important and very relevant in different areas of Computer Science and Tech! Your choice of language depends on what you want to do.

#### **Hello World in C**

.c is the extension for C codes, just as .py is for Python.

Code attached: testhello.c

<https://www.programiz.com/c-programming/examples/print-sentence>

Some resources to understand stdio.h and other headerfiles:

<https://fresh2refresh.com/c-programming/c-function/stdio-h-library-functions/>

<https://www.tutorialspoint.com/cprogramming/c_header_files.htm>

<https://www.tutorialspoint.com/standard-header-files-in-c>

#### **Compilation and Execution**

The only things you need to know about the compilation process:

1. The C compiler spots the syntax or semantic errors that you might have made in your code.
2. The C compiler also allocates space in the memory for the variables you have created in your code if there are no errors.
3. Your code can run into errors during execution itself even if there are no syntax errors in your code. Such errors are called run time errors.

Think of an executable this way: You want to share your program, let’s say a computer game, with A. But you don’t want to share your code with them, because you fear they might misuse your code. You can just share your code’s executable with A. A won’t be able to trace back to your original code (unless, of course, they know how your game works and they write their own code for your game).

Compile your C code hello.c -

1. gcc hello.c // in this case, your executable is stored in the default executable a.out
2. gcc hello.c -o hello // here, you are naming your executable. It won’t be stored in a.out. Now, your executable is hello.
3. gcc -o hello hello.c //another way to do the above command. They mean the same things

[Note: In case you get a “permission denied” error, use “sudo” before gcc in the above commands.]

Run/Execute your C code hello.c -

1. If you used method 1 above to compile, run your code as:

./a.out

1. If you used methods 2 or 3 above to compile, run your code as:

./hello

#### **Primitive Datatypes in C**

Int, char, float, double.

<https://www.geeksforgeeks.org/data-types-in-c/>

Be very careful about datatypes of variables - C is very strict about types.

Code attached: basic.c

#### **Input and Output in C**

stdio.h has two important functions among others:

printf: a function to print to the console

scanf: a function that is used to take input from the user.

<https://www.programiz.com/c-programming/c-input-output>

<https://codeforwin.org/2015/05/list-of-all-format-specifiers-in-c-programming.html>

#### **Loops and Conditionals**

Code attached: loops.c

Loops:

Very beautifully explained here:

For Loops: <https://www.programiz.com/c-programming/c-for-loop>

Do while and while loops: <https://www.programiz.com/c-programming/c-do-while-loops>

Conditionals:

If-else in C: <https://www.programiz.com/c-programming/c-if-else-statement>

Read about other stuff (break, continue, switch statements) under C - Flow Control section here: <https://www.programiz.com/c-programming>

I did not cover **makefiles** in this class, but that is something important to know:

<https://opensource.com/article/18/8/what-how-makefile>

<https://dev.to/nerfur/easy-make-and-makefile-for-beginners-476a>

<https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/>

### **Some other important things to know**

(Feel free to contact me with questions regarding this section)

#### **Operators: Unary, Binary and Ternary**

<https://tutorials.webencyclop.com/c-language/c-operators/>

#### **Functions in C**

<https://www.programiz.com/c-programming/c-functions>

### **Some corrections in the recording**

11:38: I meant “not empty” instead of “empty”

18:23: scanf is an inbuilt function present in the headerfile stdio.h function to take input from the user - I realise I said “keyword”.

18:30: Yes, I meant percent (%), and not dollar ($), jeez.

44:38: My enthu south Indian parents and guests :)

### **Practice problems**

### [Source: <https://www.w3resource.com/c-programming-exercises/>. You can also access more practice problems from this link.]

Compile and run all your programs

[standard input output]

1. Write a C program to print the following characters in a reverse way. Take the characters as input from the user.

*Test Characters*: 'X', 'M', 'L'

*Expected Output*:

The reverse of XML is LMX

1. Write a C program to compute the perimeter and area of a rectangle taking the height and width of the rectangle as input.
2. Write a C program to convert specified days into years, weeks and days (take input from the user).
3. Write a C program that accepts two item’s weight (floating points' values ) and calculate the average weight of the items.
4. Write a program in C that reads a first name, surname and year of birth from the user and displays the names and the year one after another sequentially.

*Expected Output* :

Input your first name: Tom (your input)

Input your last name: Davis (your input)

Input your year of birth: 1982 (your input)

Tom Davis 1982

[If-else Conditional statements]

1. Write a C program to read the age of a candidate and determine whether it is eligible for casting his/her own vote (using if-else statements).
2. Write a C program to find the largest of three numbers. (take the three numbers as input from the user)

[Operations]

1. Write a C program to perform addition, subtraction, multiplication and division of two numbers.

*Expected Output* :

Input any two numbers separated by comma : 10,5

The sum of the given numbers : 15

The difference of the given numbers : 5

The product of the given numbers : 50

The quotient of the given numbers : 2.000000

[Loops]

1. Write a program in C to display n terms of natural numbers and their sum.

Enter input : 7

*Expected Output* :

The first 7 natural number is :

1 2 3 4 5 6 7

The Sum of Natural Number upto 7 terms : 28

1. Write a C program to calculate the factorial of a given number.

Input the number : 5

*Expected Output* :

The Factorial of 5 is: 120

Don’t hesitate to contact me if you have any questions or doubts regarding this content or anything else: [prashanthi.r@alumni.ashoka.edu.in](mailto:prashanthi.r@alumni.ashoka.edu.in) (+91 9500662485)