Linux Commands

- gcc --version
- ls --version
- which gcc
- touch f1.txt f2.txt f3.txt
- cp demo.c test.c
- cp demo.c dir1/
- cp -r dir1 copy
- clear
- history
- mv f1.txt ECP
- mv dir1 ECP
- f2.txt desd.txt
- rm desd.txt
- rm -r copy
- cd ~
- cd -
- cd ..
- ls -a
- mv f3.txt .f3.txt
- ls -l file1.txt
- chmod 764 file1.txt
- chmod 744 file1.txt
- chmod u-x file1.txt
- chmod g+w file1.txt
- chmod +x file1.txt
- chmod 777 file1.txt
- chmod 664 file1.txt
- chmod 274 file1.txt
- chmod +rwx file1.txt

VI Editor commands

- :w - save/write into file
- :q - quit/exit from editor
- :q! - quit without saving
- :y - copy current line
- yy - copy current line
- nyy - copy n lines from current line
- y^ - copy current line upto cursor position
- y$ - copy current line after cursor position
- yw - copy current word
- nyw - copy n words from cursor position
- :m,ny - copy lines from m to n
- :d - cut current line

- dd - cut current line
- ndd - cut n lines from current line
- :m,nd - cut lines from m to n
- d^ - cut current line upto cursor position
- d$ - cut current line after cursor position
- dw - cut current word
- ndw - cut n words from cursor position
- p - paste copied content
- u - undo
- ctrl + r - redo

C Programming

History

- C language was developed by Dennis Ritchie in 1972 at AT & T Bell Labs on PDP-11 machine.
- It was developed while porting UNIX from PDP-7 to PDP-11.
- Many features of C are inspired from B (Ken Thompson) and BCPL (Martin Richards).
- Initial release of C is referred as K & R C.

Standardization

- C was standardized by ANSI in 1989. This is referred as C89.
- Standardization ensures C code to remain portable.
- C standard is revised multiple times to add new features in the language.
    - C89 – First ANSI standard
    - C90 – ANSI standard adopted by ISO
    - C99 – Added few C++ features like bool, inline, etc.
    - C11 – Added multi-threading feature.
    - C17 – Few technical corrections.

Introduction

- High-level
- Compiled
- Procedural
- Block-Structured (control structures).
- Typed
- Library Functions

Features

- Data types
- Operators
- Control structures
- Functions
- Storage classes

- Pointers
- Arrays
- Strings
- Dynamic memory allocation
- Structures
- Unions
- Enums
- File IO
- Preprocessor directives

Strengths

- Low level memory access (pointers, data structures)
- Effective memory access (bitwise operators, bit-fields, unions)
- Can access OS features (functions/commands)
- Extensive library functions (math, strings, file IO, ...)
- Compilers for different platforms & architectures
- Highly Readable (macros, enum, functions, ...)

Applications

- System programming
    - OS development
    - Device drivers
    - System utilities
- Embedded programming
    - ARM, AVR, PIC, etc.
    - IoT development
- Language development
    - Compiler development
- Achievements (tiobe.com)
    - In top-2 languages in last 40 years.
    - Language of year: 2019, 2017, 2008.

Toolchain & IDE

- Toolchain is set of tools to convert high level language program to machine level code.
    - Preprocessor
    - Compiler
    - Assembler
    - Linker
    - Debugger
    - Utilities

- Popular compiler (toolchains)
  - GCC
  - Visual Studio
- IDE – Integrated development environment
  - Visual Studio
  - Eclipse
  - VS Code (+ gcc)
  - Turbo C
  - Anjuta, KDevelop, Codeblocks, Dev C++, etc.

Hello World Program

- Source Code

```c
// Hello World program
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

- Commands

```
$ gcc -c main.c
$ gcc -o main.out main.c
$ ./main.out
```

Tokens

- C program is made up of functions.
- Function is made up of statements.
- Statement contain multiple tokens.
  - Keywords
  - Data Types
  - Identifiers
  - Variables
  - Constants
  - Operators

Keywords

- Keywords are predefined words used in program, which have special meanings to the compiler.
- They are reserved words, so cannot be used as identifier.
- K & R C has 27 keywords. C89 added 5 keywords. C99 added 5 new keywords.

Identifiers

- Identifiers give names to variables, functions, defined types and pre-processor macros.
- Rules of Identifiers:
  - Should start with alphabet or with _ (underscore)
  - Can include alphabets, _ (underscore),digits
  - Case sensitive
- Examples:
  - Var_1 //Valid
  - 1_var // Not Valid
  - _var //valid
  - Var-1 // invalid
  - Basic Salary //invalid

Data Types, Variables & Constants

- C allows computations to be performed on various types of data.
  - Numerical: Whole numbers, Real numbers
  - Character: Single character, Strings
- Fixed data values are said to be constants.
  - 12, -45, 0, 2.3, 76.9, 1.23456e+2, 'A', "Sunbeam", etc.
  - Constant examples
    - -23, 1L, 34U, 3UL, 0x41, 0101,
    - 1.234f, 1.234567e+2, ...
    - 'A', '\101', '\x41'
    - "SunBeam", "A\101\x41"
- Data is hold in memory locations identified by names called as variables.
  - Variable must be declared before its use in the program.
  - As per need, variable have some data type.
  - Variable examples
    - int number = 10;
    - double basic_salary = 20000.0;
    - char letter = 'A';
    - int roll_number;
    - roll_number = 20;
    - double price = 200.0;
    - price = 300.0;
- Simple C data types are: int, double, char.
  - Data type represents amount of space assigned to the variable.
  - It also defines internal storage of the data.
- Each variable is assigned some memory location.
- Size of data type of given variable or constant is found by sizeof() operator.

printf()

- Arbitrary strings and variable values can be printed using printf() function.
  - int - %d
  - double - %lf
  - char - %c
- Examples:
  - printf("Hello PreCAT @ Sunbeam");
  - printf("%d", roll_number);
  - printf("%d %lf %c", number, basic_salary, letter);
  - printf("Book price is %lf", price);
- Escape sequences
  - \n, \r, \t, \b, \a, \, %%, ", '

printf() and scanf()

- #include <stdio.h> -- function declaration
- printf()
  - Used to print values & string on terminal.
  - Various format specifiers %d, %c, %f, ...
  - Formatting: %5d, %-7d, %8.2f, ...
- scanf()
  - Used to input values from user.
  - Same format specifiers as of printf().
  - Do not use any char other than format specifiers in format string.
  - To skip a char from input use %*c.