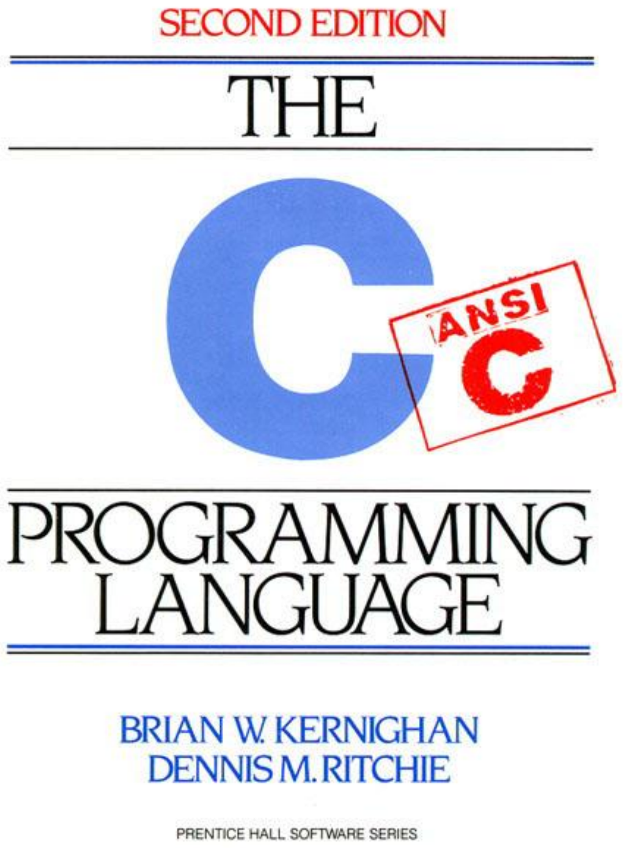
Lab: Computer Concepts & C Programming (ECS-159)

Rashmikiran Pandey
Former Research Fellow @ IIIT-A
Received Russian Gov. International Scholarship
MS, PhD(Pursuing, MIPT Russia)

About ‘the’ Course

- An assignment oriented course
- More emphasis on problem solving

Text



Kernighan, Ritchie. Second Edition

About C

- *GNU : GNU's Not Unix*
  - *GNU C: gcc is a standard compiler*

- *C is non portable*
  - *Terms: Compiler (human -> machine [once]), Interpreter (instructions -> machine [each time the program is run])*
- *C is a high level language*
  - *One line in c maps to many lines of assembly code*

C: History

- Developed in the 1970s – in conjunction with development of UNIX operating system
  - – When writing an OS kernel, efficiency is crucial

    This requires low-level access to the underlying hardware:
    - e.g. programmer can leverage knowledge of how data is laid out in memory, to enable faster data access
  - – UNIX originally written in low-level assembly language – but there were problems:
    - No structured programming (e.g. encapsulating routines as "functions", "methods", etc.) – code hard to maintain
    - Code worked only for particular hardware – not

C: Characteristics

- C takes a middle path between low-level assembly language…
  - Direct access to memory layout through pointer manipulation
  - Concise syntax, small set of keywords
- … and a high-level programming language like Java:
  - Block structure
  - Some encapsulation of code, via functions
  - Type checking (pretty weak)

C: Dangers

- C is not object oriented!
  - Can’t “hide” data as “private” or “protected” fields
  - You can follow standards to write C code that looks object-oriented, but you have to be disciplined – will the other people working on your code also be disciplined?
- C has portability issues
  - Low-level “tricks” may make your C code run well on one platform – but the tricks might not work elsewhere
- The compiler and runtime system will rarely stop your C program from doing stupid/bad things
  - Compile-time type checking is weak
  - No run-time checks for array bounds errors, etc. like in Java

My first C program!

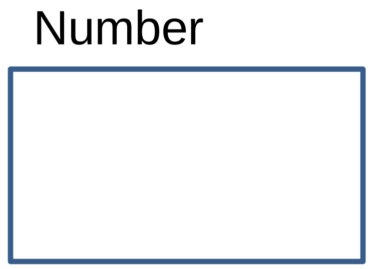
```
/* thou shall begin from
somewhere*/
#include <stdio.h>

// program prints hello world
int main() {
   printf ("Hello world!\n");
   return 0; //indicated program ended //
   successfully
} // end of main function
```

More..

```
#include <stdio.h>
// program reads and prints the same thing

int main() {
    int number;
    scanf(“%d”, &number);
    printf (“%d\n”, number);
    return 0;
}
```



1. Programming on Linux

- Linux command line: GNU-C
  - Use console based editors: vi, emacs, nano
  - Or text based editors: kwrite, gedit, kate
- IDE
  - Eclipse *
    http://www.eclipse.org/cdt/downloads.php

* = available on windows too.

Linux Familiarization

- Common shell commands
  - Remember, commands are issued to a shell
  - pwd, ls, dir, mkdir, cd, date, whoami
  - touch, cp, mv, rm, chmod, cat, less, more, tail
  - man
  - Commands are programs (usually in /usr/bin, /bin/)
  - Most commands take options and input
    - ls   ls -a   ls -l   ls -lt   ls -ltr
- Everything is case-sensitive
- Tab completion, command history

Programming on Linux contd…

- Writing programs
  - Use any editor (graphical, console)
  - Save file as <filename>.c
- Compiling programs
  - gcc <filename>.c   gcc funnysingh.c –o funnysingh
- Running programs
  - ./a.out   ./funnysingh
    (executable files need to have executable permissions.
    $chmod +x <executable>)

Compilation is not a single stage

- Pre process : cpp (C Preprocessor) gcc –E
  - Removes comments, includes #include files
- Compile : gcc –c (GNU compiler)
  - main step, compilation, change into machine code
- Link : ld (GNU linker)
  - link executables

gcc does all the above steps

2. C on windows

- Use a text editor
  - install notepad++
  - compiler : MinGW
    how to install and work-
    http://csjava.occ.cccd.edu/~gilberts/mingw/
- IDE
  - Eclipse *
  - Microsoft Visual C++ Express Edition 2008

Or 3. Work on windows, yet use gcc

- Install SSH Secure Shell or Putty
  - Connect to cc servers: webhome.cc.iitk.ac.in or linserv.cc.iitk.ac.in etc.
- Want to see GUI too?
  - Install Xming
    - And then, enable X11 tunnelling

- Why doesn’t my windows binary run on linux?
  - File format: exe and elf
    - man elf
  - In linux, program does system calls.
  - Libraries are different

Good programming practices

Indentation

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

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

Good programming practices contd..

- Variables names
  - Not too short, not too long
  - Always start variable names with small letters
  - On work break
    - Capitalize: myVariable, OR
    - Separate: my_variable

Good programming practices contd...

- Put comments

```
#include <stdio.h>
int main() {
  /* this program adds
  two numbers */
  int a = 4; //first number
  int b = 5; //second number
  int res = 0; //result
  res = a + b;
Printf();
}
```

Good programming practices

- Your code may be used by somebody else
- The code may be long
- Should be easy to understand for you and for others
- Saves lot of errors and makes debugging easier
- Speeds up program development

Programs

1. WAP in C to print your complete Address. [Demonstration of multiline]
2. WAP in C to find out simple interest.
3. WAP in C to calculate area of Rectangle
4. WAP in C to calculate area of circle
5. WAP in C to find out size of integer, character, float, double, long [Hint:Use sizeof() operator]
6. WAP in C to find out division of two numbers. (Denominator may be float)

- Lab Program -1, Date- 28.05.2021