CSE 220: Systems Programming

2 – Introduction to C
Karthik Dantu
Ethan Blanton
Computer Science and Engineering
University at Buffalo
kdantu@buffalo.edu
Why C?

• Dozens of programming languages – why C?  
  C is “high level” – but not very  
  C provides functions, structured programming, complex data types  
  and many other powerful abstractions

• It also exposes many architectural details

• Most system software including OS kernels are written in  
  C/C++

• C influences many other languages
Effective C

- Effective programming in C requires that you master the machine
- You must be aware of the system architecture and details of operation
- We will be using C in Linux on x86-64
- The compiler we will use is gcc
- The dialect of C we will use is C99
CSE 220 and C

- That said, CSE 220 is not about learning C (only)
- CSE 220 teaches you systems concepts, and you will learn to implement them in C
- We will not cover all details of C syntax
- We will cover ideas, and some syntax when we feel necessary
- You should consult:
  - K&R book
  - Unix man pages
  - Given code
A Simple Computer Model

- Data in memory is stored at accessible addresses
- CPU is able to manipulate data stored in memory and access I/O
- Program code is executed as a series of instructions
  That manipulate memory
  Interact with input/output devices
  Display results to the user
- Program code is also stored in memory – possibly not accessible
Modern Multi-Tasking OS

• Most modern OSes (including *NIX) provide a particular model

• Each process has its own dedicated resources, i.e., each process appears to have:
  A dedicated CPU
  Private, dedicated memory
  Private I/O

• OS provides mechanisms to share existing resources among all active processes
Program Execution

- C programs (all programs) are translated into machine instructions
- Computer executes these instructions in order
- Instructions are things like:
  Add two numbers together (and other arithmetic operations)
  Store a number to a location in memory
  Retrieve a sensor reading
  Display a result
- Its all numbers!
Imperative Programming

- C is an imperative language
- It consists of a list of statements
- Each statement is an instruction to the computer to do something
- Statements can be grouped into functions
- The computer executes the program from beginning to end (roughly) – i.e., imperative
- Modern systems (especially interactive systems such as smartphones/robots) allow for event-driven programming
Every C program starts with the function `main()`

```c
int main() {
    return 0;
}
```

Every C function takes zero or more arguments

Every C function can return a single value

Every statement ends with a semi-colon (;)

C programs are stored in files that end with .c extension

Let us examine `main()` in more detail
• The main function takes two arguments:

```c
int main(int argc, char *argv[])
```
The main function takes two arguments:

```
int main(int argc, char *argv[])
```

- First argument
- Second argument
- delimiter
The main function takes two arguments:

```c
int main(int argc, char *argv[])
```
The main function takes two arguments:

```c
int main(int argc, char *argv[])
```

- Pointer type
- Argument is an array
Aside on slide syntax

```bash
$ gcc program.c -o program
```

Terminal
prompt

- `$` sign indicates the terminal prompt
- Please do not type this – you will get an error
- You should type everything that follows the `$` sign
- Good time to brush up on Linux basics


[2] Comprehensive set: [https://ryanstutorials.net/linuxtutorial/](https://ryanstutorials.net/linuxtutorial/)

Karthik Dantu and Ethan Blanton
Assume you saved our earlier program as trivial.c:
```c
int main() {
    return 0;
}
```
We can compile it into an executable program as follows:
```
gcc trivial.c
```
This produces a file a.out, which is a native binary
```
ls
a.out  trivial.c
```
You can run the binary as follows:
```
./a.out
```

Karthik Dantu and Ethan Blanton
First Real Program

- "Hello World" is a classic first program when learning a language
- Objective is to print "Hello, world!" in the terminal