# Scripting & Computer Environments

*An Introduction*

IIIT-H

July 31, 2013


WELCOME!

# Course Specifics

- **LTPC**: 3-0-2-4
- **Prerequisites**: Little/no exposure to Unix-like systems or prior programming experience

## Course Description

*This course aims to provide and equip you with the necessary technical skills so as to make you reasonably comfortable with various Unix-like computing environments. Throughout, you will be exposed to some useful utilities/tools and languages to help you in the journey.*

At the end of the course, you should be:

- Comfortable enough working with various unix-like computing environments.
- Able to write simple to complex scripts to automate tasks/solve problems.
- In a position to design and implement database-driven web apps.

1. *Shell Programming*
   - Linux basics, commands and file systems, shell scripting, swiss-army-knife tools (`vi`, `grep`, `sed`, `awk`, ...)
2. *Web Programming*
   - Intro to basic concepts of the World Wide Web (WWW) and tools used to develop web apps.
   - Client-side *&* server-side scripting (`HTML`, `XHTML`, `CSS`, `Javascript`, `Python`, ...)
3. *Database Programming*

- Evaluation methods [1]

| | |
|---|---|
| 2 mid exams | *30%* |
| Assignments/lab activities/quizzes/project [2] | *40%* |
| Final exam | *30%* |

- Reference materials

  TBA throughout

[1] Subject to change

[2] The heart of the course

- Check out the various policies and guidelines (e.g. attendance, computer usage, examination, etc) of the institute from here.
- Deadlines
- Code plagiarism (A or B?)

**A**

- Sharing code verbatim: copying, retyping, submitting copy of a file.
- Coaching: helping your friend to write a code line-by-line.

**B**

- Helping others use tools, clarify or help understand a concept.
- Helping others debug their code.

# Course Specifics: Help!


- Approach/mail the TAs or me. Don't shy away!
- Assignment-specific queries on the Courses Portal.
- Separate threads will be created for each.
- Any Qn?

# Brainstorm: The Genesis

- Computer
- Program
- Script

# Brainstorm: The Genesis

- Computer
- Program
- Script

# Brainstorm: The Genesis

- Computer
- Program
- Script

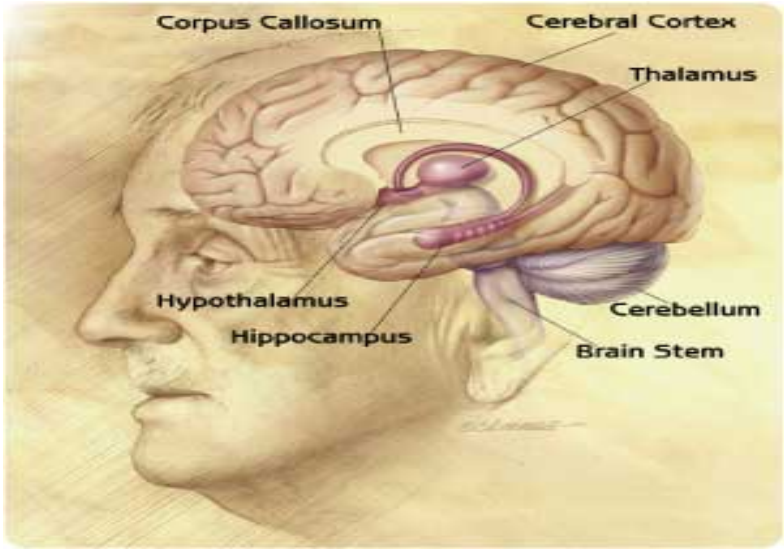
# Brainstorm: Computing

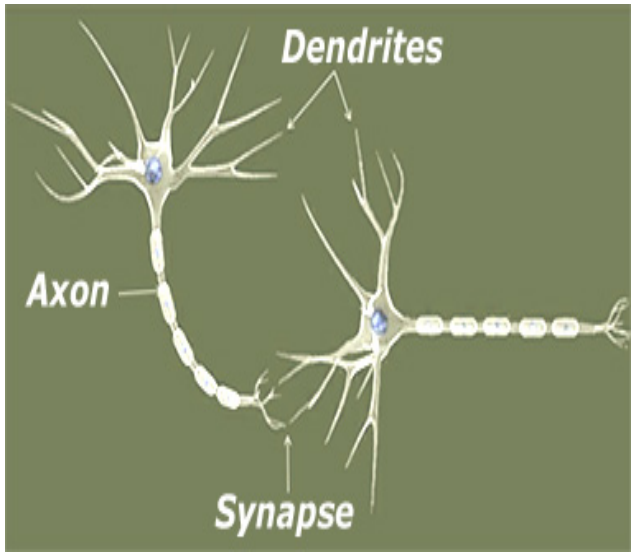
- Computers were 'born' for this.
- The whats?
- The whys?
  - "Necessity is ..."
  - "To make life easier." Agree/disagree?
  - What else?

# Brainstorm: Computing

- Computers were 'born' for this.
- The whats?
- The whys?
  - "Necessity is ..."
  - "To make life easier." Agree/disagree?
  - What else?

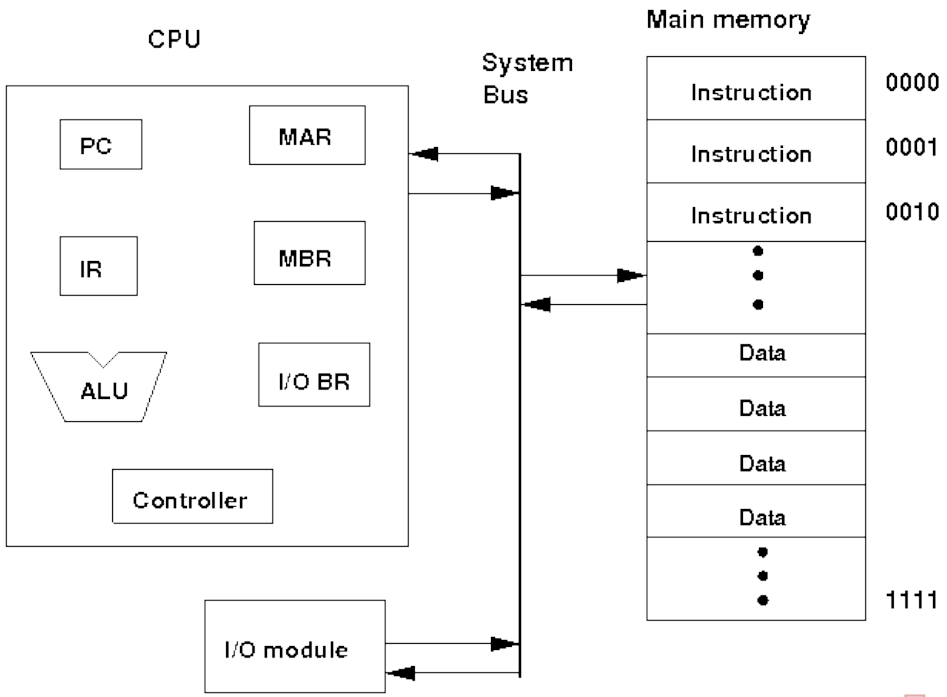
# Computing: The Brain




- Nerve cells/*neurons* are central in human computation.
- *Dendrites* receive signal from other neurons via *synapses*.
- Signal transmission is a *chemical* process resulting in high/low electrical potential inside the cell.

1. I/O Unit
2. Memory Unit (primary, secondary)
3. Processing Unit (ALU, CU, registers)
4. Interconnection Structures (control, address & data buses)



# The Winner Is...

- Organization
- Mode of processing
- Storage

- Speed
- Complexity
- Control mechanism
- Resilience.

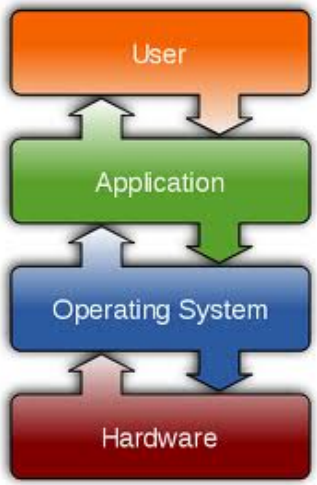
## Software

A set of programs, procedures, algorithms and its documentation.

- Written using *programming languages*
  - High-level languages e.g. `C`, `C++`, `Java`, `scripting languages` ...
  - Assembly language - mnemonic-based e.g. `ADD`, `SUB`, `MOV` ...
  - Low-level language - native language of computer circuitry
- Each language has syntactic + semantic rules.
- Otherwise, syntax, logical and/or runtime errors.

Did you know? Some famous software bugs: Mark II's Moth (1945), Mariner I space probe (1962), Y2K bug (2000)

- Compiled vs Interpreted languages
- Application vs System software




**Example**

translators, `operating systems`, device drivers, etc

# The Operating System

A set of software that *manages* computer hardware resources and *serves* other programs.

**Example**

`Windows, Unix, Linux, OS X, Android, OSBSD, etc`

- Past → Expensive hardware, 'cheap' people!

  *Goal:* maximize hardware utilization.

- Now → Cheap hardware, expensive people!

  *Goal:* make it easy for people to use computer.

  *Characteristics:* Enormous and complex (millions of LOC)

- Future →

- Proprietary vs Open-source
- Single-user vs Multi-user
- Single-tasking vs multi-tasking
- Distributed
- Embedded
- Real-time operating systems (RTOS)

  Multi-tasking OS for apps with fixed deadlines.

  e.g. industrial robots, embedded systems, engine controllers, etc.

**Example**

`VxWorks, QNX, MontaVista Linux`

Generally, coordination and control center

- Abstraction layer (i.e. hides hardware details)
- Resource Manager
  - Processor manager
  - Memory manager
  - File manager
  - Device manager
- User interface
- Networking & Security (e.g. access control)

# Computing Trends & Environments

1. Distributed Computing
   e.g. sensor networks, Internet
2. Mobile Computing
   e.g. ipod, ipad, iphone, etc
3. Ubiquitous/pervasive Computing
   e.g. smart homes, cars, buildings, etc
4. DNA/Biomolecular computing
5. Cloud Computing
6. Social Computing
   SNS and collaborations
7. Green Computing/ Green IT
   "Average Google search releases 7 gm of CO2 " (Google says 0.2 gm)

# Next...