Computers and Programming Languages

GBS 746-VT – Introduction to Scientific Computing

February 19th, 2014

Emidio Capriotti
http://biofold.org/emidio

Biomolecules Folding and Disease

Division of Informatics
Department of Pathology

UAB
THE UNIVERSITY OF
ALABAMA AT BIRMINGHAM

Basic Computer Architecture

- A basic computer is composed by CPU, Memory or RAM, storage support and I/O peripherals.

- The CPU (central processing unit) is the hardware that executes the basic arithmetical, logical, and input/output operations of the system.

- RAM (Random Access Memory) is normally associated with volatile types of memory, where its information are stored and lost when the power is removed.

- Secondary memory is the slowest form of memory. It cannot be processed directly by the CPU. It must first be copied into primary storage. It include magnetic disks like hard drives and optical disks.

Test your RAM

Write on your RAM memory

```
> for i in `seq 1 100000`
do
echo $i >> /dev/shm/nums.txt
done
```

Write on your hard-drive memory

```
> for i in `seq 1 100000`
do
echo $i >> ~/nums.txt
done
```

What is the time needed to execute the two commands?

```
> time (for i in `seq 1 100000`; do  echo $i >> /dev/shm/nums.txt; done)

> time (for i in `seq 1 100000`; do  echo $i >> ~/nums.txt; done)
```

How much space?

A **bit** is the basic unit of information in computing and digital communications. It is a binary number that corresponds to two states generally represented with 0 and 1.

The **byte** is most common unit of digital information in computing that consists of eight bits.

How show the disk space in your machine?

```
> df -H —total
```

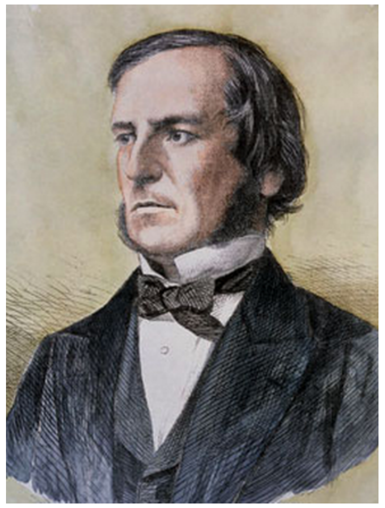
What your computer is doing?

```
> top
```

How show the amount of RAM in your machine?

```
> free -h
```

Boolean Algebra



Boolean algebra is a part of mathematics for the manipulation of binary variables that can assume true and false values, usually denoted 1 and 0 respectively.

Thew basic operations in the Boolean Algebra are:

NOT, AND, OR.

All the calculation performed by the computer can be write using the basic Boolean operations.

Boolean operations and conditional statements are extensively used in programming

The generic structure of conditional statements is:

```
IF (boolean condition) THEN
   (consequent)
ELSE
   (alternative)
END IF
```

Statements in Bash

In the bash shell the syntax of a conditional statement is the following:

```
>if [ statement ];
  then
        (consequent)
  else
        (alternative)
  fi
```

An example: verify that Human PFAM file (9606.tsv) has more than 3 lines.

```
>if [ `zcat 9606.tsv.gz |wc -l` -gt 3 ]
  then
        echo 9606.tsv.gz `zcat 9606.tsv.gz|wc -l`
  else
        echo "ERROR: file 9606.tsv.gz too short" >/dev/stderr
  fi
```

Statements can be combined with operators ! (NOT), && (AND), || (OR)

Exercise 1

Write a bash script that takes in input the name of a fasta file with only one sequence and check

1. if the file exists
2. if the file has the a one line header in the first line
3. if the file has more than one line

Download the following files using wget

http://www.uniprot.org/uniprot/P53_HUMAN.fasta
http://www.uniprot.org/uniprot/BRCA1_HUMAN.fasta

IF option -f to check if the file exists

```
> if [ -f namefile ] …
```

To find the header use the command grep and caret (^) on the first line using head

```
> grep ^\> …..                    # Searches for the character > on the first column
```

Concatenate two expression in the if statement

```
> if [ expression ] && [ expression 2 ] then ….
```

Algorithm and program

Algorithm is a step by step procedure used for to solve a calculation problem.
The number of step should be finite.

Why algorithms?

The complexity of a problem is proportional to the number of calculations needed to solve a computational problems and is given as a function of the number of variables.

Problems in computer science can be classified as NP-hard vs NP-Complete. NP means nondeterministic polynomial time. NP-Complete problems can be verified in polynomial time. Most of the problem are NP-hard. One example is the “Traveling Salesman Problem”.

Algorithms are developed to optimize the solution of the problem.

In some cases we need to find heuristics that although the provide a solution close to the exact one in a polynomial time.

A program is computer program is a specific implementation of an algorithm or set of algorithms that running ia a computer provide the solution of a computational problem.

Program Languages

- A programming language is an artificial language to communicate instructions to a machine, particularly a computer.
- Programming languages can be used to develop programs and to express algorithms.
- Programming languages usually relies on imperative and declarative forms.
- A programming language is usually described by two components: the syntax (form) and the semantics (meaning).

Classification

The main categories of program languages are compiled and interpreted.

A compiled language is a programming language whose implementations are typically compilers which generate machine code (object code) from source code.

An interpreted language is a programming language that executes instructions directly, without previously compiling a program into machine-language instructions.

Abstraction level

A low-level programming language provides little or no abstraction from a computer's instruction set architecture. Very machine specific and difficult to use, but can be very fast.

A high-level programming language has strong abstraction from the details of the computer. In comparison to low-level programming languages, it less dependent from computer architecture more easy to understand but needs to be compiled or interpreted.

Assembly

Hello world in assembly

```
# hello.s
        .section .data
hello:
        .ascii    "Hello, world!\n"
hello_len:
        .long     . - hello
        .section .text
        .globl _start
_start:
        ## display string using write () system call
        xorl %ebx, %ebx                 # %ebx = 0
        movl $4, %eax                   # write () system call
        xorl %ebx, %ebx                 # %ebx = 0
        incl %ebx                       # %ebx = 1, fd = stdout
        leal hello, %ecx                # %ecx ---> hello
        movl hello_len, %edx            # %edx = count
        int $0x80                       # execute write () system call
        ## terminate program via _exit () system call
        xorl %eax, %eax                 # %eax = 0
        incl %eax                       # %eax = 1 system call _exit ()
        xorl %ebx, %ebx                 # %ebx = 0 normal program return code
        int $0x80                       # execute system call _exit ()
```

Compile Assembly

1. Generating the Object Code

```
> as -o hello.o hello.s
```

2. Link Object File

```
> ld -o hello hello.o
```

3. Run the program making it executable

```
> chmod a+x hello
> ./hello
> Hello, world!
```

Compile C program

1. Write the code in the hello.c file

```
#include <stdio.h>

int main(void)
{
  printf("Hello world\n");
  return 0;
}
```

2. Generating executable file

```
> gcc hello.c hello_c
```

3. Run the program making it executable

```
> chmod a+x hello
> ./hello
> Hello, world!
```

Run the python script

1. Write the python script in the hello.py file

```
#!/usr/bin/python

print "Hello, world!"
```

2. Run the script making it executable

```
> chmod a+x hello.py
> ./hello.py
> Hello, world!
```

3. Make the script consistent with C

```
#!/usr/bin/python

def main():
        print "Hello, world!”
        return

main()
```

Another command

AWK
data extraction from text file

1. Basic structure

```
BEGIN { do something 1 }

      { do something 2 }

END   { do something 3 }
```

2. Example: extract column 1 from file 9606.tsv.gz

```
> zcat 9606.tsv.gz | awk -F “\t” ‘{print $1}’
```

3. Use if statement to print rows with “Family” in position 8

```
> zcat 9606.tsv.gz | awk -F “\t” ‘{if ($8==“Family”) print $0}’
```

Exercise 2

Write a command line using awk that from 9606.tsv,gz file selects

1. rows with “Family in position 8
2. E-value in position 13 lower than 1e-10

Use the AND operator (&&) to solve this problem

```
> awk ‘{ if (expression1 && expression2 ) do something }’
```

Restrict the search selecting domains with alignment length >=100

Concatenate three expressions in the IF statement calculating the difference between columns 3 and 2

```
> awk ‘{ if ( ($3-$2)>threshold && expression1 && expression2 ) do something }’
```