THEORY OF INFORMATION, ARCHITECTURE OF COMPUTERS AND OPERATING SYSTEMS (TIACOS)
Bioinformatics, ESCI
Computer Architecture Dept., UPC
2020/2021 – 3th term

Midterm exam **Wed 12 May**

- This exam is **closed book**, **closed Internet**. You cannot use your laptop.
- Time **64 minutes**

1. (2 points) Definitions. What is?

   (a) an Operating System?

   The startup program that manages everything. The OS is a software that manages hardware resources. It acts like intermediary between applications and hardware. Provides a runtime environment between convenient and efficient to run programs.

   (b) a process?

   A process is the OS representation of a program during its execution.

2. (2 points) Filling the gaps. Write the following numbers in **decimal**, **binary** and **hexadecimal**. Show all calculation steps.

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
25	**00011001**	19
82	01010010	52
167	**10100111**	$A7$
172	10101100	**AC**

$$0x19 = 0b\underbrace{0001}_{1}\underbrace{1001}_{9}$$

$$0b0\overset{6}{\mathbf{1}}0\overset{4}{\mathbf{1}}00\overset{1}{\mathbf{1}}0 = 2^6 + 2^4 + 2^1 = 64 + 16 + 2 = 82$$

$$0xA7 = 0b\underbrace{1010}_{A}\underbrace{0111}_{7}$$

$$0b10101100 = 0b\underbrace{1010}_{A}\underbrace{1100}_{C} = 0xAC$$

3. (2 points) Floating point representation. Suposse we are using the standard IEEE 754, 32b, normalized. Which is the real number represented by the number $0x40D80000$? Show all calculation steps.

$$0x40D80000 = 0b\ \overbrace{0}^{sign}\ \overbrace{10000001}^{exponent}\overbrace{10110000000000000000000}^{mantissa}$$

$$Sign : positive, e = \overbrace{129}^{exponent} - \overbrace{127}^{E_{max}} = 2, frac = \overbrace{1}^{normalized} \ .\ \overbrace{1011}^{mantissa}$$

$$real : 1.1011 * 2^2 = 110.11_2 = 2^2 + 2^1 + 2^{-1} + 2^{-2} = \mathbf{6.75}_{10}$$

4. (2 points) Bash. In a Linux system, the password file (`/etc/passwd`) has the following contents:

```
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
john:x:1000:1000:john,,,:/home/john:/bin/bash
alice:x:1001:1000:Alice,,,:/home/alice:/bin/zsh
bob:x:1002:1000:Bob:/home/bob:/bin/tcsh
```

What's the output after execute the following command line? Explain what each command does.

```
cat /etc/passwd | awk -F: '{print $7}' | xargs -I shell ls -la shell | grep rw-
```

1. `cat` reads the file `/etc/passwd`, writing its content to the standard output (the first pipe).
2. `awk -F: 'print $7'`, prints to the standard output (the second pipe) the 7th column of each line read from standard input (the first pipe). The 7th column contains a path to a shell binary. For instance, `/bin/bash`.
3. `xargs -I shell ls -la shell`, runs in parallel `ls -la shell`, that is, list all the information of the file named 'shell'; where 'shell' is the 7th column of each line in `/etc/passwd`
4. `grep rw-` searches the standard input (last pipe) selecting the lines that match the pattern 'rw-', ie, files with read and write permissions, but not execute.

Hence, the output of this command line is **empty** because all the files in `/bin` have execute permission.

5. (2 points) Integer representation. What's the value, in decimal, of `a, b, c, d` variables after execute the following Python lines of code?

```
a = 8>>3
b = ~a + 1
c = a & b
d = a | b | c
```

`a =` **1**, because `8>>3` returns 8 with the bits shifted to the right by 3 places . So, `a` is equal to **1**.

$$\begin{array}{rl} & 0\ldots01000_2 \\ >> & 3 \\ \hline & 0\ldots00001_2 \end{array}$$

`b =` **-1**, because $\sim a{+}1$ returns the complement of `a` (the number you get by switching each 1 for a 0 and each 0 for a 1) plus 1. As it is codified in 2's complement, $\sim a + 1$ is equal to $-a = -\mathbf{1} = b$. In binary: $1\ldots1$

$\sim a + 1 = \sim 0\ldots01 + 1 = 1\ldots10 + 1 = 1\ldots1$

`c =` **1**, because `a & b` does a "bitwise and". Each bit of the output is 1 if the corresponding bit of `a` AND of `b` is 1, otherwise it's 0. So, `c` is equal to **1**.

$$\begin{array}{rl} & 0\ldots01 \\ \& & 1\ldots11 \\ \hline & 0\ldots01 \end{array}$$

`d =` **-1**, because `a | b | c` does a "bitwise or". Each bit of the output is 1 if the corresponding bit of `a` OR of `b` OR of `c` is 1, otherwise it's 0. So, `d` is equal to $-\mathbf{1}$, because `b` is $1\ldots1$.

$$\begin{array}{rl} & 0\ldots01 \\ | & 1\ldots11 \\ | & 0\ldots01 \\ \hline & 1\ldots11 \end{array}$$