Operating Systems	**Homework 1**
CS4410 Summer 2011	**Due Sunday, 29 May 2011**

1 System Calls and Interrupts

Suppose we have a computer system with 3 privilege levels and 3 corresponding privilege modes (pm):

User (pm = 0):	User's applications
OS_mid (pm = 1):	Services that provide information about the files of the system (size, creation date, owner, etc).
OS_bottom (pm = 2):	Services that manage files (create, open, read, write, delete, etc) and all the other services that a usual system offers.

We want to add the function *delete_file_if_empty* in the user's application interface. This function will take the *name of a file* as an argument and, first, it will check if the file is empty or not. If the file is empty, the function will delete the file and it will return *1*, else it will return *0*.
As can be understood, for the realization of this function, all the privilege levels will, possibly, take part:

- The User level initiates the call of *delete_file_if_empty*.
- The OS_mid checks the size of the file to see if it is empty.
- The OS_bottom deletes the file if OS_mid decides that it is empty.

We decide to implement this function using system calls.

1. Why do we use system calls?
2. How many system calls do we have to implement and why?
3. Would you wrap the system calls into libraries? Why?
4. Suppose Kathy has a file *"my_summer_grades.txt"* that is empty. She decides to write a program to clean up her disk, through which she scans all the files and executes the function *delete_file_if_empty* for each file. If the CPU state, exactly before the execution of the system call *delete_file_if_empty ("my_summer_grades.txt")*, is $SP = f34, PC = cf32, pm = 0, v_0 = 42$:
   (a) Write the sequence of values of the pm from the start to the end of the execution of the system call *delete_file_if_empty("my_summer_grades.txt")*. Assume that only Kathy's program is executed this moment.
   (b) What is the CPU state at the end of this system call?
   (c) Repeat *a* and *b* if the file *"my_summer_grades.txt"* is not empty.
   (d) It is apparent that this function includes communication with the hard disk, in order the file to be actually deleted. At which level would you place the device driver of the hard disk? Why?
   (e) If we want to guarantee that the file is actually deleted when the system call returns to Kathy's program, then the driver of the hard disk should wait a notification from the hard disk controller before returning to its caller. Describe how the hard disk controller will notify the hard disk driver about the completion of the file deletion (lecture 02, slide 11).

2 Context Switch

1. What is Context Switch?
2. Why do we use it?
3. Does the OS take part in the Context Switch? Why?

3 Process State Transition

Suppose that the process A is going to be executed alone at Ted's computer system. The sequence of commands that A is going to execute is:

- compute_a
- open network connection
- compute_b
- read from disk
- compute_c
- compute_d
- end,

where *compute_** represents a chunk of commands that need CPU cycles. If between the *compute_c* and *compute_d* an interrupt will happen, notifying the CPU that some data have been received and if the initial state of the process A is *New*, describe its state transitions and the kind of waiting queues that A will be found in.

4 Threads

What is the basic reason for the creation of a process and which for a thread?

5 Programming Part

The file serial.py implements a naive way to compare fingerprints and determine their matching degree. More specifically, we have created three matrices (fingerprints); the first belongs to the criminal that we want to find, and the other two belong to two suspects. Thus, we compare the first "fingerprint" with the other two and, then, we calculate and print the matching degree for each of the suspects.
Even though our "fingerprints" are not detailed (6 x 12 pixels), the real one contain much information and usually there are more than two suspects that have to be examined. Thus, we will try to parallelize the program to share the work of comparing among 4 threads.
Please, look at the file parallel.py and try to fill the code-gaps.