Team 30 – Milestone 3

OS Operating System: *Payroll System*

The aim of our project is to create a batch operating system that fulfils the role of a customized Payroll system. It's designed to run per user/call and perform multiple jobs that in the end accumulate to provide the salary due to each employee along with the accompanied transactions. It has 2 main resources:- memory and input file.

- **OS Type:** Batch Processing

Our system does not require any real time input from the user.

- **OS Components:**
    a. **Input**
        i. Employees record file from database
    b. **Code**
        i. Module to all loan installment payment
        ii. Module to update monthly salary
            1. Module to compute any deductions (i.e. advance payments, loans, ...)
        iii. Module to add working hours
        iv. Module to print out financial slips
    c. **Output**
        - In case of payment day, checks/payment slips
        - Updated version of employees record
        - Throughout its run time, the system will output status update to help users track the function of the system and offer an insight on the operations performed and in which order

- **OS Methodology:**

1) *Module to compute daily salary*

Cards Input files include info about when the employee logged onto the system (card swiping at entrance) and when they've left, computes the total working hours and accordingly computes the daily salary for each employee depending on job position (manager, secretary,…etc) and department (from employee files). If employee didn't log

in (come that day), will deduct from available vacation days or if no vacation days available, will deduct from salary (i.e. will have a negative value for the daily salary).

2) *Module to update monthly salary*

   Adding the computed daily salary to the cumulative value stored. If current date is the end of month (payment day), will deduct any loan installments due, add bonuses if available and order printing of payment slip/check.

3) *Module to compute bonuses for after hours*

   If employee surpassed the average or set working hours/day, will compute bonus according to extra hours, department and position.

4) *Module to print out checks/payment slips*

   Will format the payment slip print out and retrieve the appropriate information (employee name, current date, amount to be paid,…etc)

- **First Code Implementation:**

Our code is divided into 3 main packages: -

1. Processes --> this package contains the Process class that initiates any program to the NEW state, ready to be activated by the instantiation of a program/application. The process class is abstract with attributes (process ID , memory Address, state) and contains an abstract run() method that each application that will implement the class must have it customized to the specified use of the app/process.
2. Operating System --> this package contains a memory class that's responsible for holding the array simulating the RAM memory of a real-life OS. We have made it of a fixed size to ensure that it always for resource allocation and since real life memory is not infinite. This class also manages the memory and informs CPU of available space needed to store new processes or be used by processes that are already being executed. There is a Scheduler class that contains 3 queues: running queue, blocked queue and ready queue. Since in real life more than one process is expected to run at a time (simulation of parallelism by fast context switching or multicore) we created a running queue that will hold at most 2 processes and only one of them will be allowed to be executed at a time if they need the same resource (ex. Input file) and if they don't, they will be allowed to be run in parallel. The ready queue will store the rest of the processes that have been activated and wait their turn to be deployed (I.e. inserted in the running queue and be executed). The class also contains a schedulerRunning() method that runs

on a separate thread continuously and acts as the main scheduler making sure queues are in order (running queue has max 2 apps) and moves processes in and out of queues as well a passing the process that should currently be run to the CPU. Finally, there is the CPU class that runs the process after using its process to obtain the process itself from the memory and proceeding to execute it.

Notes:

- Since this is a payroll system, we will not be implementing any priority schemes or preemption since all jobs have to be fully executed with no interruptions.