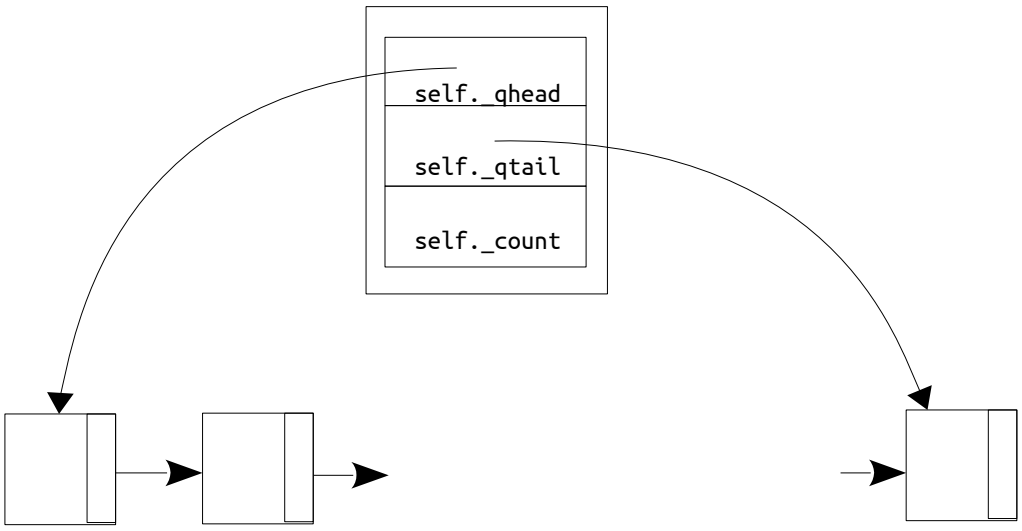
INF239 - Sistemas Operativos
Laboratorio 3

Tema: Simulando algoritmos de
Planificación de procesos

El objetivo del presente laboratorio es implementar métodos que simulen los algoritmos de planificación estudiados en clase de Sistemas Operativos.

La implementación está desarrollada en Python, se ha tomado como base la clase Queue.. Esta clase define un objeto compuesto por los siguientes campos:

```
def __init__(self):
    self._qhead = None
    self._qtail = None
    self._count = 0
```



El nodo. Que forma la lista enlazada. tiene la siguiente estructura:

```
class _QueueNode(object):
    def __init__(self, pid, burst = 0, turnaround=0):
        self.pid = pid
        self.burst = burst
        self.turnaround = turnaround
        self.next = None
```

Esta clase tiene una serie de métodos propios del TDA Colas (Queue) y se se ha añadido dos métodos: schedulingFCFS y schedulingRR

Las implementaciones no consideran tiempos de llegadas, o mejor, puede considera que todos los procesos llegan casi al mismo tiempo y el orden en la cola, corresponde a cómo han ido llegando.

Fisrt Come First Server (FCFS)

Es el algoritmo más fácil de implementar: se toma el nodo y el tiempo de su ráfaga se irá acumulando en la variable *t*. De esta forma el tiempo de retorno corresponderá al tiempo acumulado que le ha tomado a cada proceso hasta acabar con la ráfaga del propio proceso que se encuentra en la CPU.

Para no mezclar los procesos que son atendidos con los que ya se encuentran terminados, se ha considerado una cola aparte, donde la ráfaga es 0 y el tiempo de retorno es el acumulado en *t*. A continuación el código:

```
def schedulingFCFS(self):
        q = Queue()
        t = 0
        ptr = self._qhead
        while ptr is not None:
            t += ptr.burst
            ptr.turnaround = t
            ptr.burst = 0
            q.enqueue(ptr.pid, ptr.burst, ptr.turnaround)
            self.dequeue()
            ptr = ptr.next
        q.printStatistics()
```

El método printStatistics() es el encargado de imprimir los datos de lo procesos terminados, junto con sus tiempos de retorno.

Round Robin (RR)

Este algoritmo tiene como dato el quantum. En cada turno servido hay que considerar dos casos, el primero es cuando la ráfaga actual es menor que el quantum, y el segundo, en que no lo sea. Para el primer caso, significa que el proceso termina, pero el tiempo acumulado no se le debe sumar el quantum, sino la ráfaga, y este será su tiempo de retorno. Adicionalmente hay que eliminarlo de la cola de listos y hay que pasarlo a una segunda cola, para las estadísticas. En el segundo caso, al tiempo acumulado se le agrega un quantum y a la ráfaga se actualiza, restándole el quátum consumido. Luego hay que ponerlo al final y quitarlo de la cabeza de la cola, a continuación el código:

```
def schedulingRR(self, quantum):
        q = Queue()
        t = 0
        ptr = self._qhead
        while True:
            if ptr is None:
                break
            else:
                if ptr.burst <= quantum:
                    t += ptr.burst
                    ptr.turnaround = t
                    ptr.burst = 0
                    q.enqueue(ptr.pid, ptr.burst, ptr.turnaround)
                    self.dequeue()
                else:
                    t += quantum
                    ptr.burst -= quantum
                    self.enqueue(ptr.pid, ptr.burst, ptr.turnaround)
                    self.dequeue()
            ptr = ptr.next
        q.printStatistics()
```

El código completo se le proporciona junto con este archivo.

TAREA:

1) Bajo las mismas condiciones escriba métodos para simular los siguientes algoritmos:

a) Shortest Job First (SJF).
b) Prioridad: 1: menor prioridad y 5 mayor prioridad.

2) Modifique la definición del nodo para que considere el tiempo de llegada. Teniendo en cuenta esta nueva condición escriba (en base a los métodos ya escritos) los métodos que simulen las siguientes algoritmos:

a) Fisrt Come First Server (FCFS)
b) Round Robin (RR)
c) Shortest Job First (SJF).
d) Prioridad: 1: menor prioridad y 5 mayor prioridad.

El desarrollo de la tarea arriba solicitada le asegurará un buen desempeño en el laboratorio.

Para el laboratorio no será necesario traer código adicional, en el laboratorio se le proporcionará código que usted tendrá que completar según las indicaciones que se le proporcione.

Prof. Alejandro T. Bello Ruiz