CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Materia: Seminario de Solución de Problemas de Sistemas Operativos

Sección: D03

Profesor: Becerra Velázquez Violeta del Rocío

Alumno: Pérez Flores Eduardo Rafael

Código: 219747492

Carrera: Ingeniería en Computación

Actividad: 6

Fecha: 25-septiembre-2022

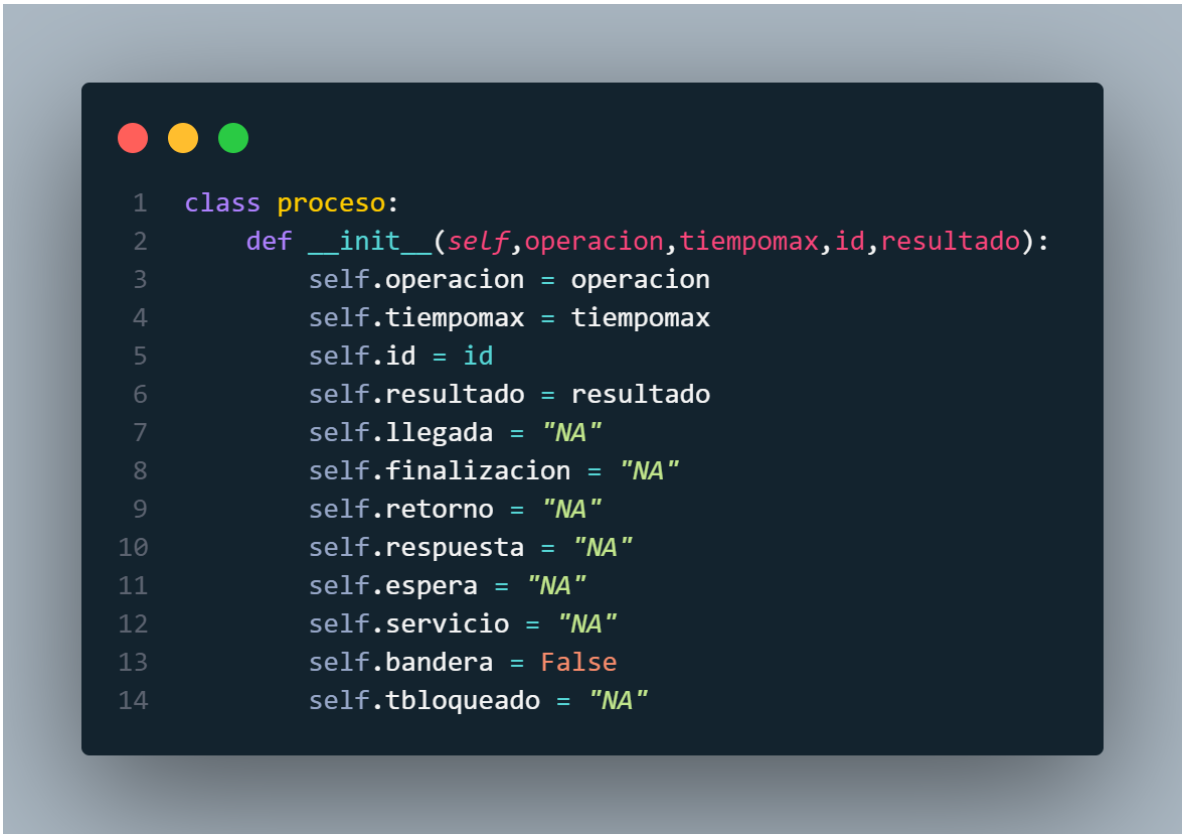
Objetivo

El objetivo de esta actividad es desarrollar el algoritmo First Come First Served, se contemplará el diagrama de 5 estados:

- Nuevo: Procesos que se acaban de crear, pero aún no han sido admitidos por el sistema operativo en el grupo de procesos ejecutables.
- Listos: Procesos que están preparados para ejecutarse, en cuanto sea su turno.
- Ejecución: Proceso que está actualmente en ejecución.
- Bloqueado: Proceso que no puede ejecutar hasta que se produzca cierto suceso, como la terminación de una operación de E/S.
- Terminado: Un proceso que ha sido excluido por el sistema operativo del grupo de procesos activos, bien porque se detuvo o porque fue abandonado por alguna razón.

Desarrollo

Para esta actividad usé gran parte el código utilizado en prácticas anteriores, eliminé la función que dividía la lista de procesos principal en lotes para utilizarla como el estado de Nuevos, cree otra lista llamada listos que es a donde pasan los procesos nuevos una vez que un proceso finaliza, para el estado en ejecución usé otra lista que: si hay procesos en listos toma el que está en la primera posición y si no hay se añade un proceso nulo y este despliega los datos “NULL”, para la de terminado en esta se le añaden los procesos que están en ejecución si terminan de forma correcta o por una interrupción, y para la de bloqueados los procesos que entran tienen un contador de 7 segundos y cuando termina se añade a la cola de listos, también añadí los atributos correspondientes a cada proceso para indicar los tiempos, los cuales se calculan dentro del código.

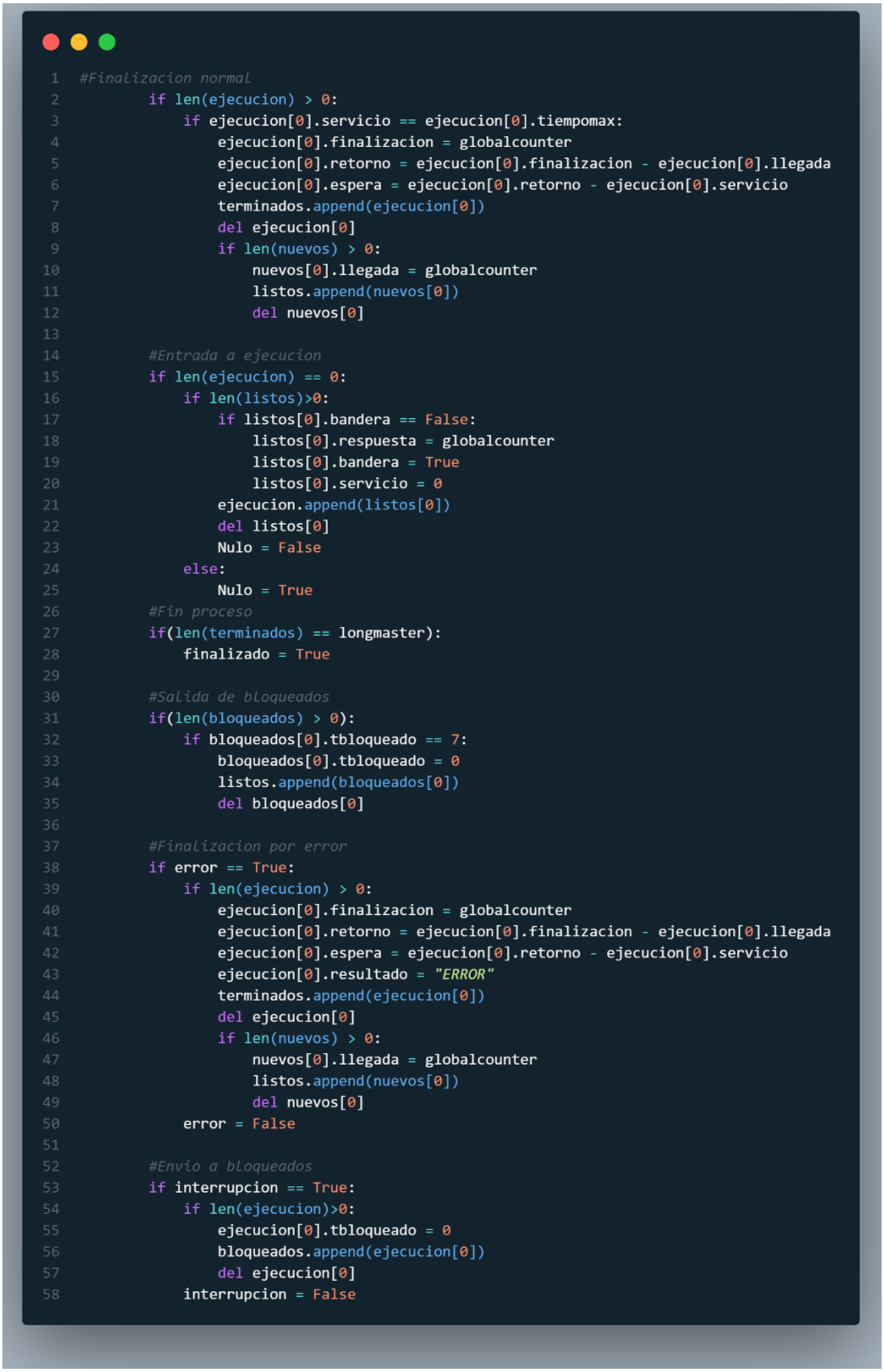


```python
class proceso:
    def __init__(self,operacion,tiempomax,id,resultado):
        self.operacion = operacion
        self.tiempomax = tiempomax
        self.id = id
        self.resultado = resultado
        self.llegada = "NA"
        self.finalizacion = "NA"
        self.retorno = "NA"
        self.respuesta = "NA"
        self.espera = "NA"
        self.servicio = "NA"
        self.bandera = False
        self.tbloqueado = "NA"
```

Ilustración 1 Clase proceso

```
#LLENAR LISTOS POR PRIMERA VEZ
    for i in range(3):
        if nuevos:
            nuevos[0].llegada = globalcounter
            listos.append(nuevos[0])
            del nuevos[0]
    #Mandar primer proceso a ejecución
    if listos:
        if listos[0].bandera == False:
            listos[0].respuesta = globalcounter
            listos[0].bandera = True
            listos[0].servicio = 0
        ejecucion.append(listos[0])
        del listos[0]
```

Ilustración 2 Transiciones de estados y cálculos de tiempos parte 1



```
#Finalizacion normal
        if len(ejecucion) > 0:
            if ejecucion[0].servicio == ejecucion[0].tiempomax:
                ejecucion[0].finalizacion = globalcounter
                ejecucion[0].retorno = ejecucion[0].finalizacion - ejecucion[0].llegada
                ejecucion[0].espera = ejecucion[0].retorno - ejecucion[0].servicio
                terminados.append(ejecucion[0])
                del ejecucion[0]
                if len(nuevos) > 0:
                    nuevos[0].llegada = globalcounter
                    listos.append(nuevos[0])
                    del nuevos[0]

        #Entrada a ejecucion
        if len(ejecucion) == 0:
            if len(listos)>0:
                if listos[0].bandera == False:
                    listos[0].respuesta = globalcounter
                    listos[0].bandera = True
                    listos[0].servicio = 0
                ejecucion.append(listos[0])
                del listos[0]
                Nulo = False
            else:
                Nulo = True
        #Fin proceso
        if(len(terminados) == longmaster):
            finalizado = True

        #Salida de bloqueados
        if(len(bloqueados) > 0):
            if bloqueados[0].tbloqueado == 7:
                bloqueados[0].tbloqueado = 0
                listos.append(bloqueados[0])
                del bloqueados[0]

        #Finalizacion por error
        if error == True:
            if len(ejecucion) > 0:
                ejecucion[0].finalizacion = globalcounter
                ejecucion[0].retorno = ejecucion[0].finalizacion - ejecucion[0].llegada
                ejecucion[0].espera = ejecucion[0].retorno - ejecucion[0].servicio
                ejecucion[0].resultado = "ERROR"
                terminados.append(ejecucion[0])
                del ejecucion[0]
                if len(nuevos) > 0:
                    nuevos[0].llegada = globalcounter
                    listos.append(nuevos[0])
                    del nuevos[0]
            error = False

        #Envio a bloqueados
        if interrupcion == True:
            if len(ejecucion)>0:
                ejecucion[0].tbloqueado = 0
                bloqueados.append(ejecucion[0])
                del ejecucion[0]
            interrupcion = False
```

Ilustración 3 Transiciones de estados y cálculos de tiempos parte 2

Conclusiones

Este programa no me pareció tan complicado porque ya no se hacía el uso de los lotes, de hecho utilicé esta actividad para corregir algunos problemas que tenía mi código de actividades pasadas y también para hacer una interfaz en la terminal que no se viera tan amontonada, solo tuve que tener cuidado en donde hacía los cálculos de los tiempos para que al final la información mostrada correspondiera a las formulas para obtenerlos.

Enlace al vídeo en Drive

https://drive.google.com/file/d/1DxoKYNALB4sJdkm3w8CsYEKsXno6M7cd/view?usp=sharing