ACTIVIDAD DE APRENDIZAJE 6 06/03/20221

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍA
DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

SEMINARIO DE SOLUCIÓN DE PROBLEMAS DE SISTEMAS OPERATIVOS
BECERRA VELZQUEZ VIOLETA DEL ROCIO
INGENIERIA EN COMPUTACIÓN
SECCION D01

PACHECO ROMERO VICTOR MANUEL
216589519

LINK VIDEO

https://drive.google.com/file/d/1phc-UuE3DsMwZovTNbRScmb_PqfIS-eN/view?usp=sharing

índice

Objetivo

El objetivo principal de esta actividad es implementar el Algoritmo de planificación FCFS (First Come First Server).

Desarrollo

Para el desarrollo de esta actividad utilice Python, inicialmente para esta practica tuve que hacer muchos cambios debido a que lo lotes ya no se mostrarían y en los anteriores trabajos estos eran necesarios para calcular todos los procesos que estaban dentro de un mismo lote, así que comencé agregando los tiempos con los que trabajaría durante el programa, estos tiempos y como calcularlos fueron explicados y vistos en las tareas y clases anteriores, la manera de almacenar esos datos fue mediante la creación de listas.

```
cont_global = 0
nuevos = []
listos = []
terminados = []
ejecucion = []
bloqueados = []


class Lote:
    def __init__(self, id, operacion, tiempo_estimado, num1, num2):
        self.id = id
        self.operacion = operacion
        self.tiempo_estimado = tiempo_estimado
        self.num1 = num1
        self.num2 = num2
        self.error = 0
        self.tiempo_transcurrido_bloqueado  = 0
        self.tiempo_transcurrido = 0
        self.tiempo_llegada = 0
        self.tiempo_finalizacion = 0
        self.espera = 0
        self.tiempo_retorno = 0
        self.tiempo_respuesta = 0

    def __str__(self):
        return (str(self.id) + " " +
            str(self.operacion)+ " " +
            str(self.tiempo_estimado) + " " +
            str(self.num1)  + " " +
            str(self.num2))
```

Seguidamente de esto uno de los cambios más importantes fue en la cuestión de la asignación de procesos, donde los tiempos serian actualizados en tiempo real y mostrados en la pantalla final para simular el algoritmo de planificación FCFS, a su vez agregue campos de longitud fija para que fuera mas fácil visualizar dichos tiempos al finalizar el programa.

```
if (len(terminados) == cantidad_procesos):
    errores=0
    normal=0
    print("Procesos terminados: ", len(terminados))

    for i in range(len(terminados)):
        errores += terminados[i].error
    print("Procesos terminados en error: "+ str(errores))

    print("ID" +("{:10}".format(''))+
    "Llegada" +("{:8}".format(''))+
    "Finalizacion" +("{:6}".format(''))+
    "Espera" +("{:10}".format(''))+
    "Servicio" +("{:8}".format(''))+
    "Retorno" +("{:8}".format(''))+
    "Respuesta" + ("{:8}".format(''))+
    "Terminado"))
    print("**********************************************************************************************
    for i in range(0,len(terminados)):
        terminados[i].tiempo_retorno = terminados[i].tiempo_finalizacion - terminados[i].tiempo_llegada
        terminados[i].espera = terminados[i].tiempo_retorno - terminados[i].tiempo_transcurrido
        print(str(terminados[i].id) +"\t\t"+
            str(terminados[i].tiempo_llegada) +"\t\t" +
            str(terminados[i].tiempo_finalizacion) +"\t\t"+
            str(terminados[i].espera) +"\t\t"+
            str(terminados[i].tiempo_transcurrido) +"\t\t"+
            str(terminados[i].tiempo_retorno)+"\t \t"+
            str(terminados[i].tiempo_respuesta)+"\t \t"+
            "Error" * (terminados[i].error) + "Normal"* (not terminados[i].error))
```

Una de las cosas que tambien cambie fue en la cuestion de las teclas, debido a que la letra “c” que hace la funcion continuar solo funcionaba al presionar la tecla “enter”, asi que en esta ocasión lo que hice fue cambiarlo para que ya no dependiera de esa tecla para continuar, sino que se pudiera presionar la letra “C” y continuar con el programa desde donde se quedó.

```python
def teclas():
    salida = False
    condicion = True
    if(msvcrt.kbhit()):
        tecla = msvcrt.getwch()
        if(tecla == 'i' or tecla == 'I'):
            if len(listos) > 0 and len(bloqueados) < 3:
                bloqueados.append(ejecucion[0])
                ejecucion.remove(ejecucion[0])
                ejecucion.append(listos[0])
                listos.remove(listos[0])
                impresion_de_procesos()
            salida = True

        elif(tecla == 'e' or tecla == 'E'):
            ejecucion[0].tiempo_estimado = 0
            ejecucion[0].error = 1
            salida = True

        elif(tecla == 'p' or tecla == 'P'):
            print("Presiona la letra c para continuar con el proceso: ")
            while(True):
                if(msvcrt.kbhit()):
                    tecla = msvcrt.getwch()
                    if(tecla == 'c' or tecla == 'C'):
                        break
                    else:
                        print("Esa no es la letra c")
                continue
            salida = True

    return salida
```

El cambio mas significativo fue en la pantalla principal donde se agregarian los procesos “Nuevos, Listos, Bloqueados y terminados” estos procesos tienen una tarea diferente, la funcion de “Nuevo” se encargaria de almancenar los procesos que se encontraran fuera de el prosesador, el procesador solo puede almacenar hasta 5 procesos por lo tanto “Nuevos” esperaria hasta que el proceso en ejecucion pasara a “Terminados” para poder agregar otro nuevo a la cola de listos y asi sucesivamente hasta terminar con todos los procesos. En la parte de “Listos” se encontrarina los procesos que estan dentro de el procesador que estarian esperando para ser atendidos. En la parte de “Bloqueados” se encuentran los procesos bloqueados durante 8 segundos, pasados esos seguntos los procesos pasarian a la cola de “Listos” para ser atendidos. Y finalmente en la parte de terminados se encuentran los procesos ya calculados con sus respectivas operaciones con sus resultados.

```
 print("********************************")
 print("Listos")
 print("ID" +("{:14}".format('')+"TME" +("{:13}".format('')+ "TT")))

 for lote in listos:
     if (lote != ejecucion[0]):
         print(str(lote.id) + ("{:15}".format('') +
         str(lote.tiempo_estimado) + ("{:15}".format('') +
         str(lote.tiempo_transcurrido))))

print("********************************")
print("Bloqueados")
print("ID" +("{:14}".format('')+ "TTB"))
if(len(bloqueados) > 0):
    for lote in bloqueados:
        if lote.tiempo_transcurrido_bloqueado  <= 7:
            print(str(lote.id) + ("{:15}".format('') + str(lote.tiempo_transcurrido_bloqueado )))
            lote.tiempo_transcurrido_bloqueado  += 1
        else:
            lote.tiempo_transcurrido_bloqueado  = 0
            listos.append(lote)
            bloqueados.remove(lote)

tiempo_transcurrido += 1
ejecucion[0].tiempo_transcurrido = tiempo_transcurrido

print("********************************")
print("Terminados")
print("ID" +("{:14}".format('')+ "Operacion" ))
for terminado in terminados:
    if(terminado.error == 0):
        print(str(terminado.id) +("{:15}".format('')+ str(obtener_resultado(terminado))))
    elif(terminado.error == 1):
        print(str(terminado.id) +("{:15}".format('')+ "ERROR "))

time.sleep(1)

if(len(ejecucion) == 1 and tiempo_transcurrido>ejecucion[0].tiempo_estimado and ejecucion[0].error == 0):
    print(str(ejecucion[0].id) + "\t\t\t| " + str(obtener_resultado(ejecucion[0])))

elif((len(ejecucion) == 1 and tiempo_transcurrido>ejecucion[0].tiempo_estimado and ejecucion[0].error != 0)):
    print(str(ejecucion[0].id) +  "\t\t\t| ERROR ")
os.system("cls")
```

Finalmente, se nos muestran todos los tiempos calculados de forma en que fueron llegando, como vemos los datos se encuentran correctos debido a que si hacemos las operaciones correspondientes estos nos muestran que si están bien calculados.

```
Procesos terminados:  12
Procesos terminados en error: 3
```

ID	Llegada	Finalizacion	Espera	Servicio	Retorno	Respuesta	Terminado
1	0	15	0	15	15	0	Normal
2	0	28	15	13	28	15	Normal
3	0	44	28	16	44	28	Normal
8	44	93	39	10	49	80	Normal
4	0	102	88	14	102	44	Normal
5	0	108	93	15	108	49	Normal
6	15	118	87	16	103	108	Normal
7	28	125	84	13	97	74	Normal
9	93	129	32	4	36	125	Error
10	102	136	27	7	34	129	Error
11	108	147	28	11	39	136	Normal
12	108	148	39	1	40	147	Error

Conclusiones

Esta actividad me pareció bastante compleja, debido a la gran cantidad de información que tuvimos que manejar, creo que unos de los problemas que mas tuve fue con los datos debido a que eran muchos, aunque fue de gran ayuda conocer como se calculaban los tiempos requeridos, gracias a eso pude comprobar que mi programa funcionaba correctamente, en lo personal me llevo mucho tiempo concluir esta actividad debido a que tuve que hacer muchos cambios al programa. En lo personal me gusto la actividad porque entendí como es funciona el algoritmo de planificación FCFS y como podemos calcular sus tiempos.