Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales.

Ingeniería en Computación

Actividad 12.

Programa 6. Algoritmo de Planificación RR

Materia: Sistemas Operativos ***Sección:*** D04

Profesor: Becerra Velázquez Violeta del Rocío.

Equipo:

- Carriola Navarro Aldo Fernando | 221978167
- Cervantes Araujo Maria Dolores | 217782452

Fecha: 12/11/2023

Índice

Tabla de Imágenes

Programa 6. Round-Robin

Objetivos

Para este sexto programa, asignado como la actividad número 12, es una implementación de Round Robin al Programa 4, anteriormente realizado. Recordando que dicho algoritmo de planificación consta de un número de Quantum el cual genera ráfagas con ese tiempo generando, siendo así, que una vez completado el Quantum en el proceso actual, en caso de no haber cumplido con su TME, este abandonará el espacio de ejecución para volver a colocarse en los procesos listos.

Teniendo en cuenta lo anterior, para este programa se busca justamente hacer la implementación de RR, además, que ahora al inicio de nuestro programa, aparte de pedir al usuario el Número de Procesos Iniciales por crear, se preguntará Cantidad de Tiempo para el Quantum deseado.

Notas Sobre el Lenguaje

El lenguaje utilizado para llevar a cabo el programa fue *Python*, dado que nos permite experimentar con diversas funciones que apoyan en la creación del programa, permitiéndonos que sea orientado a objetos, estructural, con interfaz, a consola, con color e incluso animación.

Dado que es un lenguaje de nuestro total conocimiento y dominio para ambos, optamos por crear el programa de forma estructural, es decir, mediante funciones. Buscando siempre mediante acuerdos, que ambos nos sintiéramos familiarizados con el lenguaje y lógica del programa, de esta manera optimizamos tiempos y nos fue más sencillo modelar y visualizar el programar conforme lo esperado.

TDA o Estructuras Utilizadas en el Programa

Realmente para este programa solo fue necesario crear una función para pedir el valor del Quantum al usuario, siendo este el que determine las ráfagas de tiempo de servicio entre cada proceso y dado que es continuación de programas previos, pero con la actualización del Round Robin, en nuestra función “main” solo implementamos excepciones en el algoritmo para realizar este proceso dentro del ciclo while.

```
def quantum():
    global nQuantum
    while True:
        try:
            nQuantum = int(input("\nValor de Quantum: "))
            if 0 < nQuantum <= 500:
                break
            else:
                print("\nEl valor del quantum excedio el limite.\n")
        except ValueError:
            print("\nQuantum Inválido.\n")
```

Ilustración 1. Función Quantum

Anteriormente, teníamos una condición para cuando el proceso llegara a su tiempo transcurrido establecido y de esta manera pasaba de proceso ejecutado a proceso terminado, dado este panorama agregamos dos condicionales más que nos ayudan a determinar si el Quantum designado por el usuario llego al limite y en caso de ser cierto, pasa el proceso de ejecutado a listos y en caso de que el Quantum y el tiempo transcurrido hayan concluido el ciclo que pase el proceso de ejecutados a terminados.

```
if proceso_ejecucion:

    if proceso_ejecucion[0][15] == nQuantum and proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break

    elif proceso_ejecucion[0][15] == nQuantum:
        robin = proceso_ejecucion.pop(0)
        robin[15] = 0
        procesos_actuales.append(robin)

        if not proceso_ejecucion:
            actual = procesos_actuales.pop(0)
            proceso_ejecucion.append(actual)
        break

    elif proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break
```

Ilustración 2. Algoritmo RR.

Se decidió de la siguiente manera, dado que puede haber 3 casos en cada proceso, que termine su Quantum y pase a la cola de listos, que termine su tiempo estimado antes de terminar el quantum y pase procesos terminados o que terminen ambos tiempos y se requiera pasar a terminados.

Funcionamiento.

A continuación, se presentan algunas ilustraciones que reflejan el funcionamiento del algoritmo RR:

1. Primeor el programa pregunta los n procesos y el n Quantum:

Ilustración 3. Inicio del Programa

2. Internamente se llenan los datos correspondientes y comienza el programa, en todo momento se evalua que sean 5 procesos entre las tablas de procesos listos, ejecutado y bloqueados:

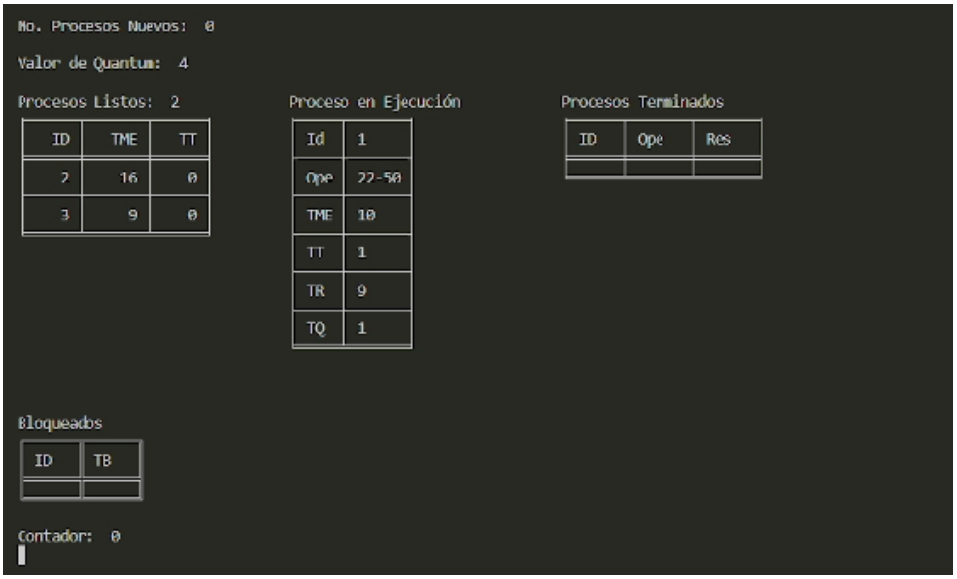


Ilustración 4. Proceso en Tablas.

3. Se realiza el Quantum y una vez que llegue al limite determinado se muestra el carrusel regresando el proceso a la tabla de listos:

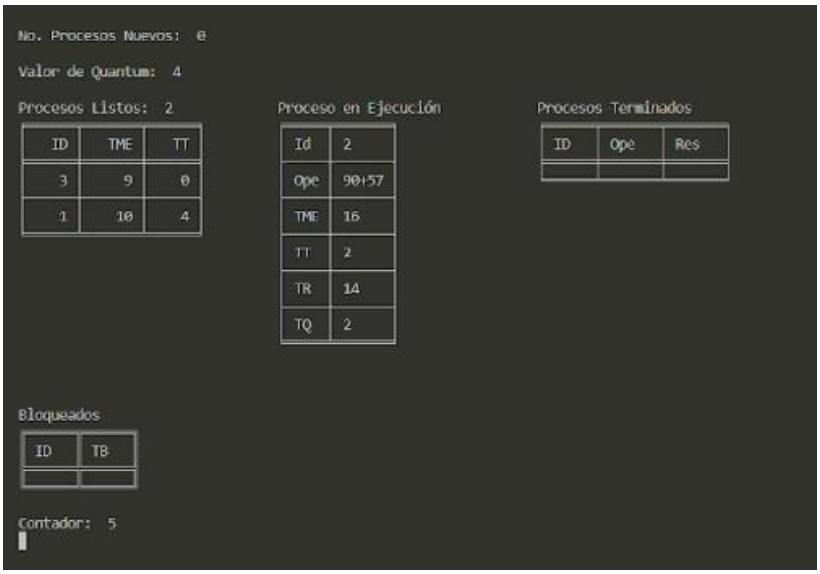


Ilustración 5. Carrusel del Quantum.

4. Todas las funciones anteriores siguen funcionando con normalidad. Mandamos un proceso por error:

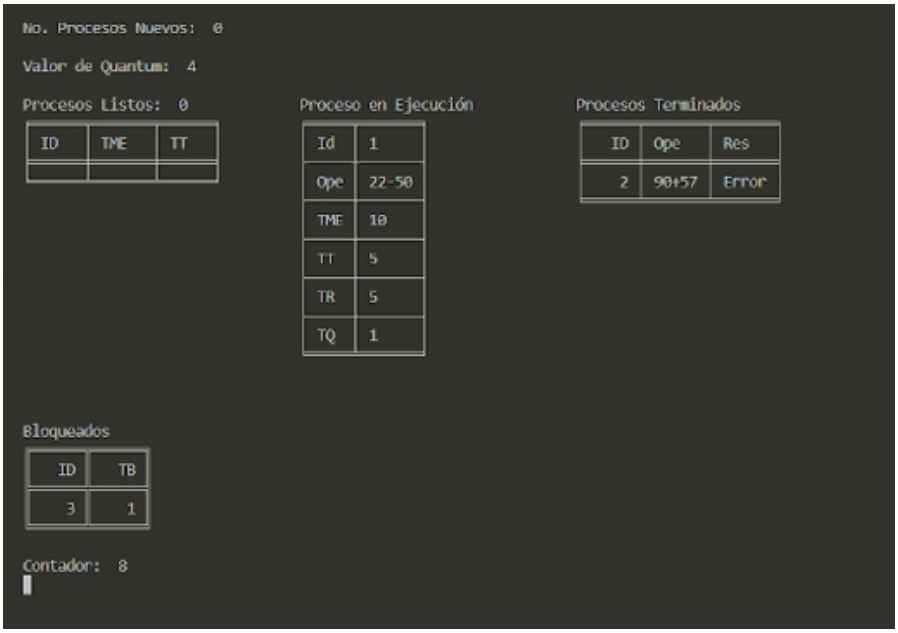


Ilustración 6. Interrupción Error.

5. Mandamos dos a bloqueados:

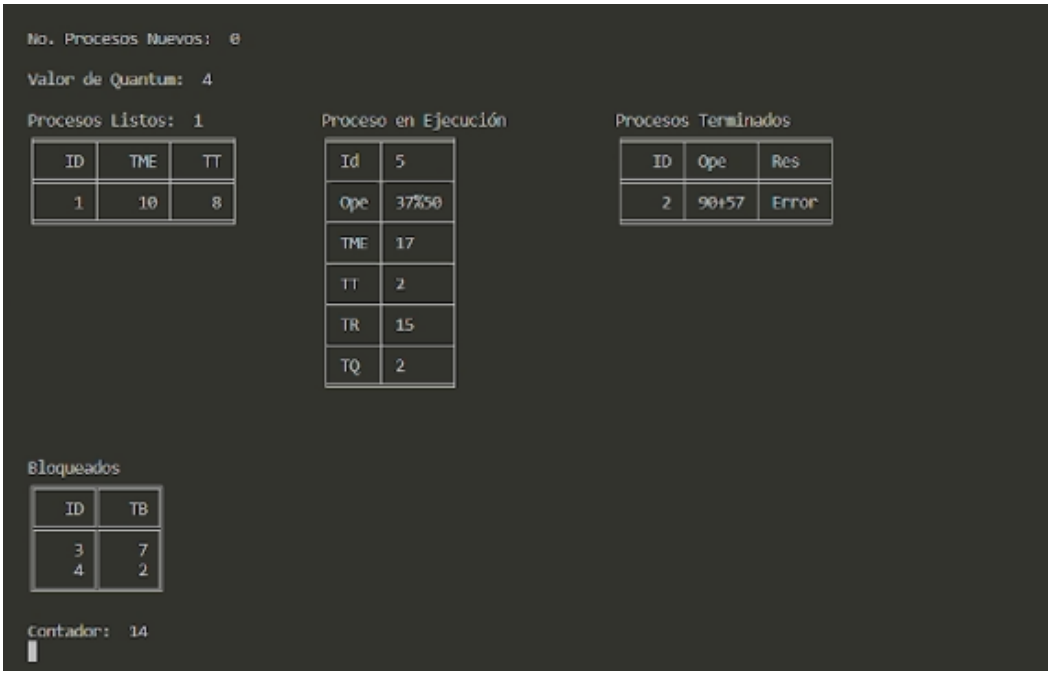


Ilustración 7. Interrupción Bloqueo.

6. Creamos un proceso nuevo:

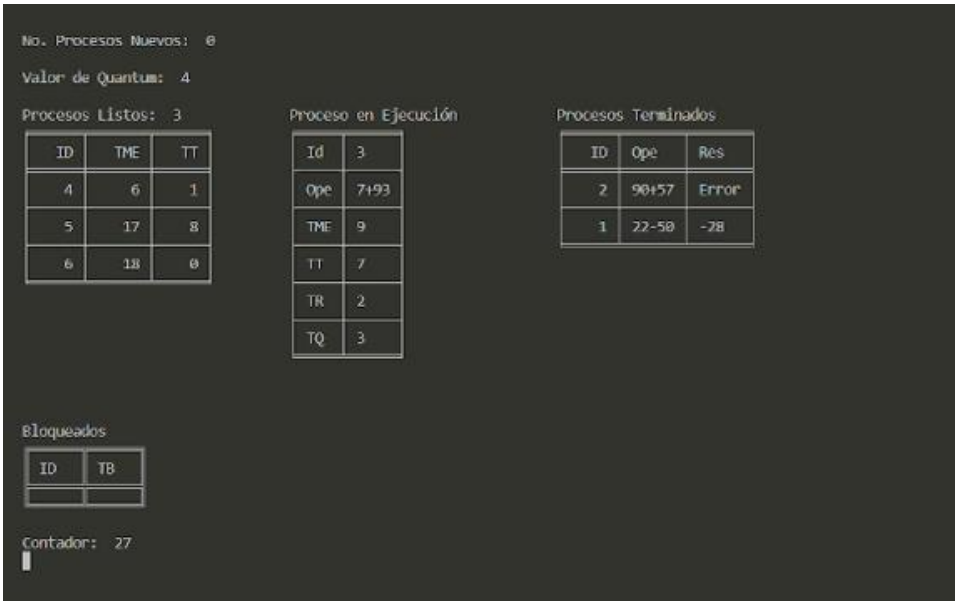
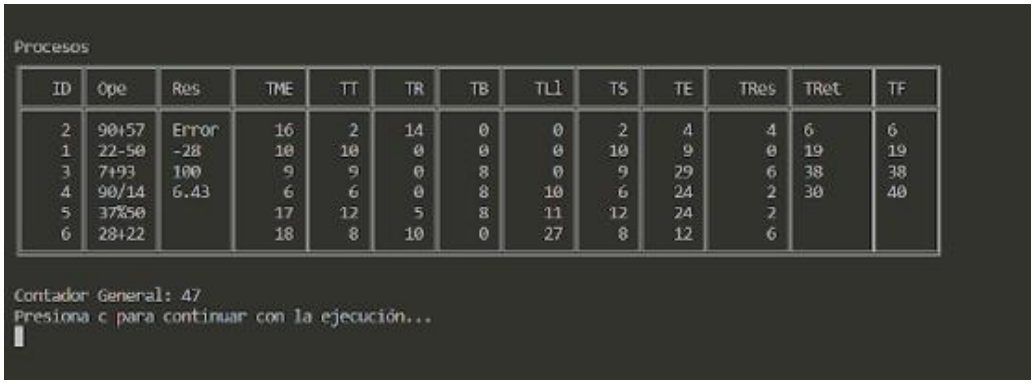


Ilustración 8. Proceso Nuevo.

7. Mostramos la tabla temporal de datos:



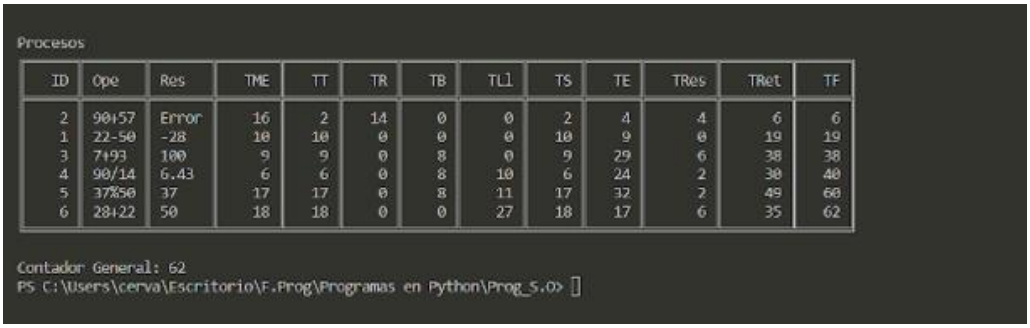
Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
2	90+57	Error	16	2	14	0	0	2	4	4	6	6
1	22-50	-28	10	10	0	0	0	10	9	0	19	19
3	7+93	100	9	9	0	8	0	9	29	6	38	38
4	90/14	6.43	6	6	0	8	10	6	24	2	30	40
5	37%50		17	12	5	8	11	12	24	2		
6	28+22		18	8	10	0	27	8	12	6		

Contador General: 47
Presiona c para continuar con la ejecución...

Ilustración 9. Tabla BCP.

8. Dejamos que finalicen los procesos y mostramos la tabla final:



Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
2	90+57	Error	16	2	14	0	0	2	4	4	6	6
1	22-50	-28	10	10	0	0	0	10	9	0	19	19
3	7+93	100	9	9	0	8	0	9	29	6	38	38
4	90/14	6.43	6	6	0	8	10	6	24	2	30	40
5	37%50	37	17	17	0	8	11	17	32	2	49	60
6	28+22	50	18	18	0	0	27	18	17	6	35	62

Contador General: 62
PS C:\Users\cerva\Escritorio\F.Prog\Programas en Python\Prog_5.0>

Ilustración 10. Tabla BCP Final.

Errores y Soluciones.

Acceder a un valor inválido dentro de la lista: Al implementar el quantum para los procesos nos encontramos de un dilema para encontrar la mejor manera de hacer que el proceso una vez que terminara el tiempo establecido volviera a entrar en ese ciclo recursivo, por lo que se refrescaba el valor una vez que terminaba, pero estábamos intentando acceder como si fuera el valor directo de la lista siendo que lo asignamos previamente a una variable.

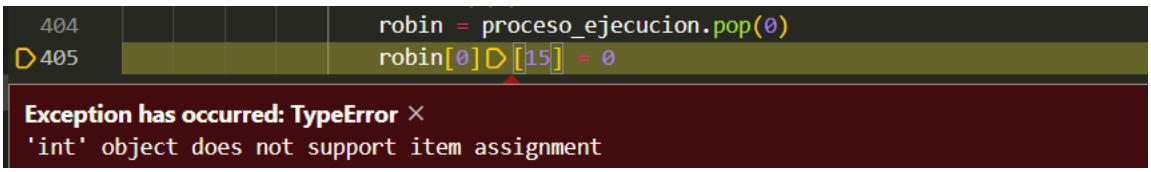


Ilustración 11. Error 1.

Así que, se puede decir que estábamos accediendo de manera incorrecta, no era necesario colocar "[0]", por lo que generaba el error en el fragmento de código mostrado en la ilustración 12.

```
if proceso_ejecucion:
    if proceso_ejecucion[0][15] == nQuantum and proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break

    elif proceso_ejecucion[0][15] == nQuantum:
        # t.sleep(1)
        robin = proceso_ejecucion.pop(0)
        robin[0][15] = 0
        procesos_actuales.append(robin)

        if not proceso_ejecucion:
            actual = procesos_actuales.pop(0)
            proceso_ejecucion.append(actual)
        break

    elif proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break
```

Ilustración 12. Causa del Error 1.

Logrando resolverlo simplemente colocando la posición directa del valor que deseábamos modificar, dado que ya teníamos ese fragmento de la lista, era más sencillo de esta manera.

```
elif proceso_ejecucion[0][15] == nQuantum:
    # t.sleep(1)
    robin = proceso_ejecucion.pop(0)
    robin[15] = 0
    procesos_actuales.append(robin)

    if not proceso_ejecucion:
        actual = procesos_actuales.pop(0)
        proceso_ejecucion.append(actual)
    break
```

Ilustración 13. Solución del Error 1.

Error al realizar el carrusel: Creando el algoritmo, principalmente no se nos ocurrió reiniciar el valor del quantum una vez que haya cumplido con el tiempo límite, por lo que una vez que terminaba el tiempo la primera vez, regresaba a la tabla de listos, pero cuando entraba de nuevo a ejecutados se convertía en un proceso infinito:

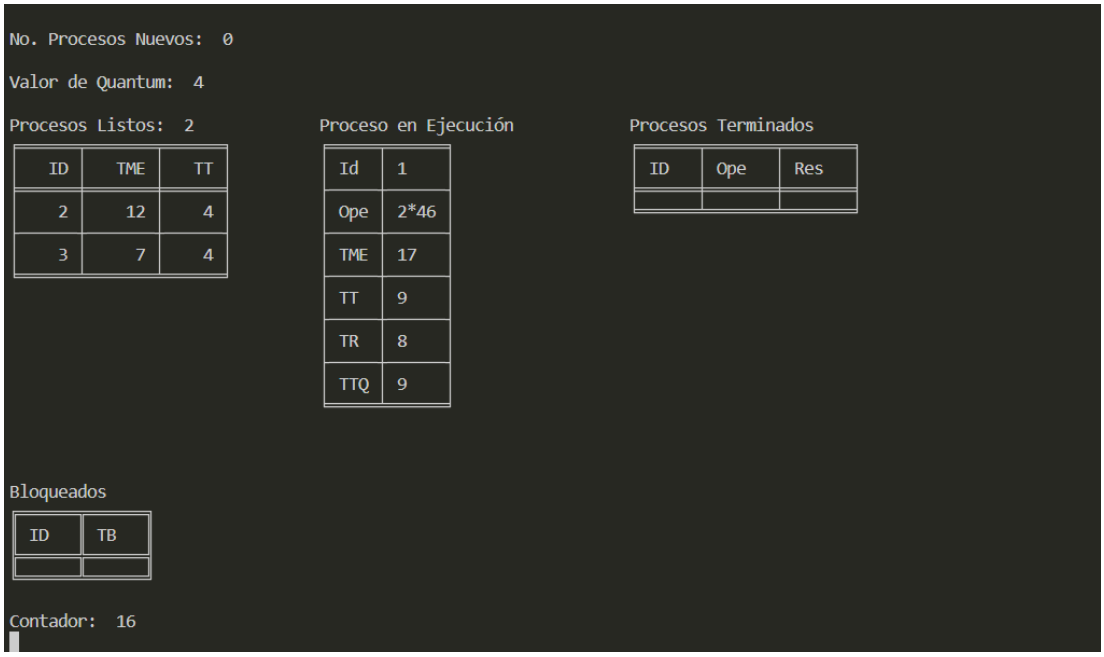


Ilustración 14. Error 2

Esto pasaba debido a que no determinamos el reinicio en el contador para el quantum por lo que la primera vez funcionaba bien, pero después ocasionaba el error en esta parte del código:

Programa 6. Round-Robin

```
if proceso_ejecucion:
    if proceso_ejecucion[0][15] == nQuantum:
        # t.sleep(1)
        robin = proceso_ejecucion.pop(0)
        procesos_actuales.append(robin)

        if not proceso_ejecucion:
            actual = procesos_actuales.pop(0)
            proceso_ejecucion.append(actual)
        break

    elif proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break
```

Ilustración 15. Causa del Error 2

Para solucionarlo, antes de agregarlo a la tabla de listos, accedemos al valor del contador quantum para reiniciarlo en cero y que al entrar vuelva a incrementarse hasta el límite designado por el usuario y agregamos una segunda condición previniendo un posible error por si el proceso llegaba a su tiempo límite de servicio al mismo tiempo que el tiempo del quantum, para que en lugar de regresar a listos entrar a ejecutados e inmediatamente pasar a terminados, que de una vez pase a terminados.

```
if proceso_ejecucion:

    if proceso_ejecucion[0][15] == nQuantum and proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break

    elif proceso_ejecucion[0][15] == nQuantum:
        # t.sleep(1)
        robin = proceso_ejecucion.pop(0)
        robin[15] = 0
        procesos_actuales.append(robin)

        if not proceso_ejecucion:
            actual = procesos_actuales.pop(0)
            proceso_ejecucion.append(actual)
        break

    elif proceso_ejecucion[0][5] == 0:
        t.sleep(1)
        finalizado = proceso_ejecucion.pop(0)
        finalizado[12] = contadorGeneral
        finalizado[11] = finalizado[12]-finalizado[7]
        finalizado[9] = finalizado[11]-finalizado[8]
        procesos_terminados.append(finalizado)
        contProcess += 1
        break
```

Ilustración 16. Solución del Error 2.

Activación inconsistente de la bandera de ejecución: Teniendo en cuenta nuestra lógica de la activación de la bandera creada para cuando cada proceso entra en ejecución por primera vez, en esta ocasión, nos empezó a generar fallos con la implementación del RR, ya que, anteriormente, todo nuestra lógica de ejecución se basaba en un 'While True', el cual entraba en 'break' en dos posibles casos:

- Cuando un proceso pasaba de Ejecución a Terminado
- Cuando un proceso pasaba de Ejecución a Bloqueado

Sin embargo, para este programa existía un nuevo caso que no habías previsto:

- Cuando un proceso pasa de Ejecución a Listo

Siendo así, que a la hora de imprimir nuestra Tabla de Procesos con la tecla 'B' esta ya no mostraba correctamente los tiempos de respuesta, y ocultaba tiempos que en teoría deberían mostrarse, así como se observa en la siguiente Ilustración.

Procesos

ID	Ope	Res	TME	TT	TR	TB	TLl	TS	TE	TRes	TRet	TF
5	58+28		11	1	10	0	0	1	12	0		
1	33%60		8	3	5	0	0	3	10	0		
2	39-76		12				0		10			
3	71+38		11				0		10			
4	86*70		9				0		10			

Contador General: 13
Presiona c para continuar con la ejecución...

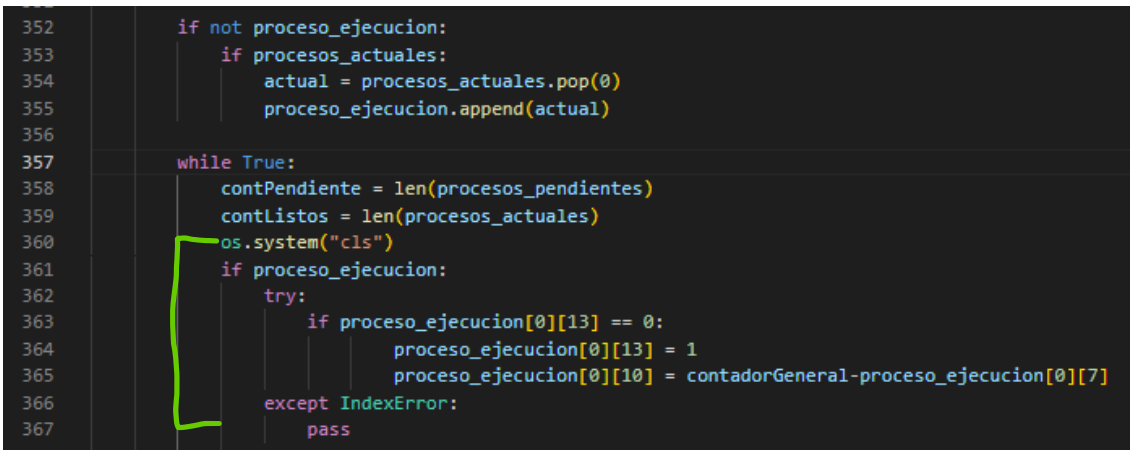
Ilustración 17. Error 3

En el ejemplo anterior, todos los procesos habían entrado ya con una ráfaga de Quantum a ejecución, sin embargo, se observa claramente, como no se muestran completos los datos, y que es ilógico que el proceso 1 y 5, tengan '0' como tiempo de respuesta (TRes) siendo que llegaron al mismo tiempo (TLl), así que, tras analizar nuestro código, dimos con la causa de dicho error.

```
352         if not proceso_ejecucion:
353             if procesos_actuales:
354                 actual = procesos_actuales.pop(0)
355                 if actual[13] == 0:
356                         actual[13] = 1
357                         contResp = contadorGeneral-actual[7]
358                         actual[10] = contResp
359                 proceso_ejecucion.append(actual)
360
361         while True:
```

Ilustración 18. Causa Error 3.

Una vez teniendo en cuenta lo anterior, nos dimos cuenta justamente de lo que comentamos al inicio de la explicación de este error, siendo que lo que hicimos para darle solución, fue bajar el fragmento de código de la activación de bandera (anteriormente señalado) dentro del 'While True' cuando se comprobaba que existiera un proceso en ejecución.



```
352         if not proceso_ejecucion:
353             if procesos_actuales:
354                 actual = procesos_actuales.pop(0)
355                 proceso_ejecucion.append(actual)
356 
357         while True:
358             contPendiente = len(procesos_pendientes)
359             contListos = len(procesos_actuales)
360             os.system("cls")
361             if proceso_ejecucion:
362                 try:
363                     if proceso_ejecucion[0][13] == 0:
364                             proceso_ejecucion[0][13] = 1
365                             proceso_ejecucion[0][10] = contadorGeneral-proceso_ejecucion[0][7]
366                 except IndexError:
367                     pass
```

Ilustración 19. Solución del Error 3.

Conclusiones

Maria Dolores.

En esta ocasión el algoritmo Round Robin, en lo personal, fue sencillo de implementar a excepción del "Error 3" que nos mantuvo atorados por un par de horas, logrando de comprender por qué se daba; para lo que tuvimos que descartar todas las opciones posibles, logrando dar con el problema en la activación de la bandera.

En la parte lógica todo estaba bien implementado, solo fue cuestión de acomodo al momento de crear el algoritmo, pero se logró solucionar satisfactoriamente y optimizamos el código, haciéndolo más preciso y eficiente al momento de la depuración, por ejemplo, trabajar directo con los valores de la lista y no con variables que luego pasas la copia a la lista, lo que hace que a veces falle entre procesos ejecutados.

Finalmente, considero que una buena retroalimentación la práctica de esta semana, dado que cada vez avanzamos más y cambian los requerimientos, vamos optimizando para que los cambios no nos afecten tanto y sea más sencilla la implementación con ayuda de mi compañero, esperamos el próximo programa que complejidades nos traerá.

Aldo Fernando.

En lo personal, me quedo muy satisfecho con los resultados obtenidos en este sexto programa, sin embargo, me atrevería a decir que fue algo frustrante el hecho de ver que al hacer la implementación del Round Robin y encontrarnos con el 'Error 3', explicado anteriormente; ha sido uno de los errores más confusos con los que nos encontramos mi compañera y yo, considerando todos los programas hechos hasta el momento.

Ya que, realmente esa parte lógica de la bandera para conocer el tiempo de respuesta, en esta ocasión no la modificamos, pero afortunadamente supimos darnos cuenta a tiempo de que la validación de la bandera la teníamos mal posicionada en el código. Llegando así exitosamente al resultado deseado.

Por otro lado, realmente este algoritmo de planificación desde la primera ocasión en el que se nos fue impartido en clase, logre comprenderlo correctamente, sin embargo, justo lo mencionado en el párrafo anterior, hizo darme cuenta de las posibilidades de 'transferencia' de un proceso entre estados posibles dentro del sistema, dándome una retroalimentación a mis conocimientos.