



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales.
Ingeniería en Computación



Actividad 14.

Programa 7. Paginación Simple

Materia: Sistemas Operativos

Sección: D04

Profesor: Becerra Velázquez Violeta del Rocío.

Equipo:

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

Fecha: 26/11/2023

Programa 7. Paginación Simple

Índice

Objetivos _____	4
Notas Sobre el Lenguaje _____	4
TDA o Estructuras Utilizadas en el Programa _____	5
Funcionamiento. _____	7
Errores y Soluciones. _____	12
Conclusiones _____	14

Tabla de Imágenes

<i>Ilustración 1. Función nueva. Tecla "T"</i>	<i>5</i>
<i>Ilustración 2. Nueva impresión en pantalla</i>	<i>5</i>
<i>Ilustración 3. Algoritmo de Paginación Simple</i>	<i>6</i>
<i>Ilustración 4. Paso de Estado en Paginación</i>	<i>7</i>
<i>Ilustración 5. Liberar espacio en memoria.....</i>	<i>7</i>
<i>Ilustración 6. Comienzo del Programa.....</i>	<i>7</i>
<i>Ilustración 7. Ejecución del Programa</i>	<i>8</i>
<i>Ilustración 8. Proceso Bloqueado.</i>	<i>8</i>
<i>Ilustración 9. Proceso Interrumpido</i>	<i>9</i>
<i>Ilustración 10. Proceso esperando a entrar en memoria.....</i>	<i>9</i>
<i>Ilustración 11. BCP</i>	<i>10</i>
<i>Ilustración 12. Tabla de Paginación.....</i>	<i>10</i>
<i>Ilustración 13. Tabla Final</i>	<i>11</i>
<i>Ilustración 14. Memoria Liberada.</i>	<i>11</i>
<i>Ilustración 15. Error 1.....</i>	<i>12</i>
<i>Ilustración 16. Causa del Error 1</i>	<i>12</i>
<i>Ilustración 17. Solución Error 1.....</i>	<i>12</i>
<i>Ilustración 18. Error 2.....</i>	<i>13</i>
<i>Ilustración 19. Causa Error 2</i>	<i>13</i>
<i>Ilustración 20. Solución Error 2.....</i>	<i>13</i>

Objetivos

Para este séptimo programa se tomará como base el programa 6, el cual manejaba la entrada y salida de nuestros procesos a ejecución por medio de un Quantum bajo el algoritmo de planificación Round Robin. Sin embargo, para este programa no solamente cabrán cinco procesos dentro de nuestro sistema, si no que, tendrán un tamaño asignado aleatoriamente, entre 6 y 26. Donde nuestro sistema estará dividido en 44 marcos de tamaño 5 cada uno, dando un total de 220 espacio total de memoria.

Asimismo, además de implementar la paginación para este séptimo programa, se debe tener en consideración los siguientes puntos por añadir:

- 4 marcos de nuestra página siempre estarán ocupados por nuestro SO, por lo tanto, estos jamás podrán ser modificados o desocupados
- Si el proceso que desea entrar necesita más marcos de los que hay disponibles en nuestro sistema, este quedará en estado de pendiente, y se mostrarán sus respectivos datos de tamaño (para de esta forma comprobar porque no pudo entrar a la página).
- A la hora de presionar la tecla 'T' esta mostrará la paginación de nuestros procesos actualmente en sistema.

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

En esta ocasión el algoritmo de RR tuvo algunos cambios, como la integración de una nueva tecla/funcionalidad para crear una paginación simple por cada proceso que entre a memoria.

Primero creamos una lista que establece los marcos (44) y los espacios en memoria permitidos para cada proceso y que es con lo que estaremos manipulando los datos a lo largo del programa, posteriormente la función de la tecla que nos permite visualizar una tabla de paginación por cada proceso, pero debido a que mostramos la misma en todo momento en el programa, solo damos una pausa para visualizar mejor cada paginación de cada proceso.

```
elif tecla == 't':  
    os.system('cls')  
    imprimir()  
    print("Presiona c para continuar con la ejecución...")  
while tecla != 'c':  
    try:  
        tecla = mv.getch().decode('utf-8')  
        if tecla == 'c':  
            print("Reanudando Programa...")  
            t.sleep(1)  
            os.system('cls')  
    except UnicodeDecodeError:  
        pass
```

Ilustración 1. Función nueva. Tecla "T"

El programa nos pide una adaptación en las tablas que imprimimos en todo momento para el usuario, por lo que adaptamos los espacios y añadimos la tabla que mostrará la paginación de los procesos, unificando las listas previamente creadas y accediendo solo a los valores del tamaño del proceso, paginación y el estado actual de cada proceso.

[illegible]

Ilustración 2. Nueva impresión en pantalla

En la función principal creamos otra lista vacía donde guardaremos copia de los datos de cada proceso, una vez que implementamos la copia, hacemos una condicional para analizar en todo momento que se encuentren procesos pendientes y en caso de no ser así, salir del algoritmo. Calculamos el tamaño de los procesos para verificar si puede entrar o no a memoria

Programa 7. Paginación Simple

para la paginación, esto en comparativa de la cantidad de marcos disponibles que estén también en memoria.

Una vez que logra entrar a la iteración porque hay marcos disponibles, se ira restando el tamaño del proceso para dividirse en las paginaciones pertinentes (de 5 cada una) y que se almacenen en los marcos disponibles en memoria, llenando los valores en la lista de procesos pendientes, donde posteriormente se manipulan a lo largo del programa entre la tabla de datos actuales/listos, bloqueados, ejecutados y finalizados, actualizando el estado.

```
vacio = ['', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '']

while contProcess != numProcess:

    ##### ----- COMIENZO DE LA PAGINACIÓN ----- #####
    copiaPendiente = copy.deepcopy(procesos_pendientes)
    while True:
        if not procesos_pendientes:
            break # Rompe el bucle si no hay procesos pendientes

        marcosDisponibles = sum(1 for p in pagina if p == vacio) # Cuenta los marcos disponibles

        tamanoProceso = (marcosDisponibles * 5) - procesos_pendientes[0][16]
        if tamanoProceso < 0:
            break # Rompe el bucle si el tamaño del proceso es negativo

        while len(copiaPendiente) > 0:
            marcosDisponibles = sum(1 for p in pagina if p == vacio)
            tamanoProceso = (marcosDisponibles * 5) - procesos_pendientes[0][16]

            if tamanoProceso <= 0:
                break # Rompe el bucle interno si el tamaño del proceso es no positivo

            if pagina[pivotePagina] == vacio:
                try:
                    primerPendiente = copiaPendiente.pop(0)
                    restador = primerPendiente[16]
                except IndexError:
                    pass

                while restador != 0:
                    if restador >= 5:
                        pagina[pivotePagina] = primerPendiente
                        pagina[pivotePagina][18] = '5 de 5'
                        restador -= 5
                    elif restador < 5:
                        primerPendiente2 = copy.deepcopy(primerPendiente)
                        pagina[pivotePagina] = primerPendiente2
                        pagina[pivotePagina][18] = f'{restador} de 5'
                        restador = 0
                        break

                    pivotePagina = (pivotePagina + 1) % len(pagina)

                pendiente = procesos_pendientes.pop(0)
                pendiente[7] = contadorGeneral

                for i, marco in enumerate(pagina):
                    if marco[0] == pendiente[0]:
                        pagina[i][17] = 'L'
```

Ilustración 3. Algoritmo de Paginación Simple

Cada que se cambia el estado del proceso se busca mediante el “ID” entre cada marco ocupado, donde se encuentre una coincidencia, es donde se cambia el valor del campo 17,

colocando una letra indicadora del estado, por ejemplo, “E” que significa que el estado se está ejecutando.

```

        procesos_actuales.append(pendiente)

    else:
        pivotePagina = (pivotePagina + 1) % len(pagina)

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

            for marco in pagina:
                if marco[0] == actual[0]:
                    marco[17] = 'E'

            proceso_ejecucion.append(actual)

```

Ilustración 4. Paso de Estado en Paginación

Cuando el proceso pasa a terminados por cualquiera de las condiciones permitidas en el sistema, se libera memoria mediante el “ID” como en el caso de cambio de estado, pero lo reemplazamos por una copia vacía en la tabla de paginación, dejando espacio para que otro proceso entre a la tabla de listos.

```

#####
for i, marco in enumerate(pagina):
    if marco[0] == finalizado[0]:
        pagina[i] = vacio

procesos_terminados.append(finalizado)
contProcess += 1
break

```

Ilustración 5. Liberar espacio en memoria.

Funcionamiento.

El programa al igual que en las versiones anteriores, comienza preguntándole al usuario la cantidad de procesos que desea en memoria y el valor del quantum.

```

Dame un número de procesos: 4
Valor de Quantum: 4

```

Ilustración 6. Comienzo del Programa

Una vez que entra al sistema se reflejan las siguientes tablas, las 4 que ya teníamos previamente y añadimos la tabla de paginación, mostrando que procesos están en espera pero

Programa 7. Paginación Simple

listos en memoria y que proceso esta en ejecución al instante, así como los 4 marcos que son ocupados por el sistema operativo, en este caso son los últimos que se muestran en la tabla.

No. Procesos Nuevos: 0
Valor de Quantum: 4 Contador: 4

Tabla de Paginación:

No. Marco	ID	Tamaño	Páginas	Estado
0	1	23	5 de 5	L
1	1	23	5 de 5	L
2	1	23	5 de 5	L
3	1	23	5 de 5	L
4	1	23	3 de 5	L
5	2	25	5 de 5	E
6	2	25	5 de 5	E
7	2	25	5 de 5	E
8	2	25	5 de 5	E
9	2	25	5 de 5	E
10	3	19	5 de 5	L
11	3	19	5 de 5	L
12	3	19	5 de 5	L
13	3	19	4 de 5	L
14	4	21	5 de 5	L
15	4	21	5 de 5	L
16	4	21	5 de 5	L
17	4	21	5 de 5	L
18	4	21	1 de 5	L
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40	50	50	50	50
41	50	50	50	50
42	50	50	50	50
43	50	50	50	50

Procesos Listos: 3

ID	TME	TT
3	18	0
4	11	0
1	14	4

Proceso en Ejecución

Id	2
Ope	20+18
TME	18
TT	1
TR	17
TQ	1

Bloqueados

ID	TB

Procesos Terminados

ID	Ope	Res

Ilustración 7. Ejecución del Programa

Mismo caso cuando interrumpimos un proceso, se muestra en la tabla de paginación que se encuentra en estado “B”, es decir, bloqueado y observando la tabla de bloqueados, notamos que dura un lapso de 8 tiempos en ese estado y después se vuelve a colocar en la cola de listos.

No. Procesos Nuevos: 0
Valor de Quantum: 4 Contador: 25

Tabla de Paginación:

No. Marco	ID	Tamaño	Páginas	Estado
0	1	23	5 de 5	B
1	1	23	5 de 5	B
2	1	23	5 de 5	B
3	1	23	5 de 5	B
4	1	23	3 de 5	B
5	2	25	5 de 5	L
6	2	25	5 de 5	L
7	2	25	5 de 5	L
8	2	25	5 de 5	L
9	2	25	5 de 5	L
10	3	19	5 de 5	E
11	3	19	5 de 5	E
12	3	19	5 de 5	E
13	3	19	4 de 5	E
14	4	21	5 de 5	L
15	4	21	5 de 5	L
16	4	21	5 de 5	L
17	4	21	5 de 5	L
18	4	21	1 de 5	L
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40	50	50	50	50
41	50	50	50	50
42	50	50	50	50
43	50	50	50	50

Procesos Listos: 2

ID	TME	TT
4	11	4
2	18	8

Proceso en Ejecución

Id	3
Ope	29/7
TME	18
TT	8
TR	10
TQ	4

Bloqueados

ID	TB
1	8

Procesos Terminados

ID	Ope	Res

Ilustración 8. Proceso Bloqueado.

Una vez que interrumpimos por error un proceso haciendo que pasara a la tabla de terminados, podemos notar que se libera memoria en la tabla de paginación.

No. Procesos Nuevos: 0
Valor de Quantum: 4
Contador: 47

Tabla de Paginación:

No. Marco	ID	Tamaño	Páginas	Estado
0	1	23	5 de 5	E
1	1	23	5 de 5	E
2	1	23	5 de 5	E
3	1	23	5 de 5	E
4	1	23	3 de 5	E
5	2	25	5 de 5	L
6	2	25	5 de 5	L
7	2	25	5 de 5	L
8	2	25	5 de 5	L
9	2	25	5 de 5	L
10				
11				
12				
13				
14				
15				
16				
17				
18	5	26	5 de 5	L
19	5	26	5 de 5	L
20	5	26	5 de 5	L
21	5	26	5 de 5	L
22	5	26	5 de 5	L
23	5	26	1 de 5	L
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40	50	50	50	50
41	50	50	50	50
42	50	50	50	50
43	50	50	50	50

Procesos Listos: 2

ID	TME	TT
2	18	16
5	12	0

Proceso en Ejecución

Id	1
Ope	73/58
TME	14
TT	12
TR	2
TQ	4

Bloqueados

ID	TB
----	----

Procesos Terminados

ID	Ope	Res
3	29/7	Error
4	0*89	0

Ilustración 9. Proceso Interrumpido

Llenamos la memoria con nuevos procesos hasta que muestra algunos como que están listos para entrar en memoria, pero no entrar debido a la memoria llena, viendo el primer proceso que espera entrar a la primera oportunidad de espacio libre.

No. Procesos Nuevos: 1
Valor de Quantum: 4
Contador: 103

Proceso Pendiente a Entrar:
ID: 18 Tamaño: 16

Tabla de Paginación:

No. Marco	ID	Tamaño	Páginas	Estado
0	11	8	5 de 5	L
1	11	8	3 de 5	L
2	12	10	5 de 5	L
3	12	10	5 de 5	L
4	13	14	5 de 5	L
5	13	14	5 de 5	L
6	13	14	4 de 5	L
7	14	11	5 de 5	L
8	14	11	5 de 5	L
9	14	11	1 de 5	L
10	15	19	5 de 5	L
11	15	19	5 de 5	L
12	15	19	5 de 5	L
13	15	19	4 de 5	L
14	16	22	5 de 5	L
15	16	22	5 de 5	L
16	16	22	5 de 5	L
17	16	22	5 de 5	L
18	16	22	2 de 5	L
19	17	12	5 de 5	L
20	17	12	5 de 5	L
21	17	12	2 de 5	L
22				
23				
24	6	11	5 de 5	L
25	6	11	5 de 5	L
26	6	11	1 de 5	L
27	7	15	5 de 5	L
28	7	15	5 de 5	L
29	7	15	5 de 5	L
30	8	25	5 de 5	E
31	8	25	5 de 5	E
32	8	25	5 de 5	E
33	8	25	5 de 5	E
34	8	25	5 de 5	E
35	9	13	5 de 5	L
36	9	13	5 de 5	L
37	9	13	3 de 5	L
38	10	7	5 de 5	L
39	10	7	2 de 5	L
40	50	50	50	50
41	50	50	50	50
42	50	50	50	50
43	50	50	50	50

Procesos Listos: 11

ID	TME	TT
16	23	0
17	23	0
11	23	3
6	25	5
14	10	4
12	24	1
7	9	4
13	23	3
9	11	5
15	10	4
10	24	8

Proceso en Ejecución

Id	8
Ope	22**84
TME	25
TT	3
TR	22
TQ	3

Bloqueados

ID	TB
----	----

Procesos Terminados

ID	Ope	Res
3	29/7	Error
4	0*89	0
2	20*18	38
1	73/58	1.26
5	39/24	1.62

Ilustración 10. Proceso esperando a entrar en memoria

Programa 7. Paginación Simple

Si pausamos el programa podemos notar la tabla de procesos con los tiempos temporales de cada proceso.

ID	Ope	Res	TME	TT	TR	TB	TL1	TS	TE	TRes	TRet	TF
3	29/7	Error	18	9	9	0	0	9	28	8	37	37
4	0*89	0	11	11	0	0	0	11	29	12	40	40
2	20+18	38	18	18	0	0	0	18	32	4	50	50
1	73/58	1.26	14	14	0	8	0	14	42	0	56	56
5	39/24	1.62	12	12	0	0	46	12	6	4	18	64
14	51-63	Error	10	5	5	0	75	5	38	12	43	118
12	40-94	Error	24	3	21	8	72	3	45	10	48	120
7	40*27		9	7	2	8	62	7	54	6		
13	22**8		23	3	20	8	73	3	47	10		
9	72-5		11	5	6	8	65	5	53	5		
15	69-83		10	4	6	0	77	4	42	17		
10	93**99		24	8	16	0	66	8	49	8		
8	22**84		25	4	21	8	63	4	56	6		
16	1/59		23	4	19	0	79	4	40	26		
17	47+45		23	4	19	0	84	4	35	25		
11	56**36		23	4	19	8	67	4	52	11		
6	46**73		25	8	17	8	61	8	54	3		
18	25%65		13				118		5			

Contador General: 123
Presiona c para continuar con la ejecución...

Ilustración 11. BCP

Tabla de paginación con espacios de memoria liberados.

No. Procesos Nuevos: 0

Valor de Quantum: 4

Contador: 229

Tabla de Paginación:

No. Marco	ID	Tamaño	Páginas	Estado
0	11	8	5 de 5	B
1	11	8	3 de 5	B
2				
3				
4	13	14	5 de 5	E
5	13	14	5 de 5	E
6	13	14	4 de 5	E
7				
8				
9				
10				
11				
12				
13				
14	16	22	5 de 5	L
15	16	22	5 de 5	L
16	16	22	5 de 5	L
17	16	22	5 de 5	L
18	16	22	2 de 5	L
19	17	12	5 de 5	B
20	17	12	5 de 5	B
21	17	12	2 de 5	B
22	18	16	5 de 5	L
23	18	16	5 de 5	L
24	18	16	5 de 5	L
25	18	16	1 de 5	L
26	6	11	1 de 5	B
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38	10	7	5 de 5	L
39	10	7	2 de 5	L
40	50	50	50	50
41	50	50	50	50
42	50	50	50	50
43	50	50	50	50

Procesos Listos: 3

ID	TME	TT
18	13	12
10	24	21
16	23	17

Proceso en Ejecución

Id	13
Ope	22**8
TME	23
TT	16
TR	7
TQ	4

Bloqueados

ID	TB
17	8
11	8
6	7

Procesos Terminados

ID	Ope	Res
3	29/7	Error
4	0*89	0
2	20+18	38
1	73/58	1.26
5	39/24	1.62
14	51-63	Error
12	40-94	Error
8	22**84	Error
7	40*27	1080
9	72-5	67
15	69-83	-14

Ilustración 12. Tabla de Paginación.

Procesos

ID	Ope	Res	TME	TT	TR	TB	TL1	TS	TE	TRes	TRet	TF
3	29/7	Error	18	9	9	0	0	9	28	8	37	37
4	0*89	0	11	11	0	0	0	11	29	12	40	40
2	20+18	38	18	18	0	0	0	18	32	4	50	50
1	73/58	1.26	14	14	0	8	0	14	42	0	56	56
5	39/24	1.62	12	12	0	0	46	12	6	4	18	64
14	51-63	Error	10	5	5	0	75	5	38	12	43	118
12	40-94	Error	24	3	21	8	72	3	45	10	48	120
8	22**84	Error	25	5	20	8	63	5	69	6	74	137
7	40*27	1080	9	9	0	8	62	9	87	6	96	158
9	72-5	67	11	11	0	8	65	11	89	5	100	165
15	69-83	-14	10	10	0	0	77	10	80	17	90	167
18	25%65	25	13	13	0	8	118	13	100	35	113	231
10	93**99	7.58e+194	24	24	0	8	66	24	144	8	168	234
11	56**36	Error	23	18	5	16	67	18	160	11	178	245
6	46**73	Error	25	22	3	16	61	22	163	3	185	246
16	1/59	Error	23	21	2	8	79	21	147	26	168	247
17	47+45	92	23	23	0	8	84	23	143	25	166	250
13	22**8	5.49e+10	23	23	0	16	73	23	157	10	180	253

Contador General: 253

Ilustración 13. Tabla Final

No. Procesos Nuevos: 0

Valor de Quantum: 4

Contador: 5

Tabla de Paginación:

No. Marco	ID	Tamaño	Páginas	Estado
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40	50	50	50	50
41	50	50	50	50
42	50	50	50	50
43	50	50	50	50

Procesos Listos: 0

ID	TME	TT

Proceso en Ejecución

Id	
Ope	
TME	
TT	
TR	
TQ	

Bloqueados

ID	TB

Procesos Terminados

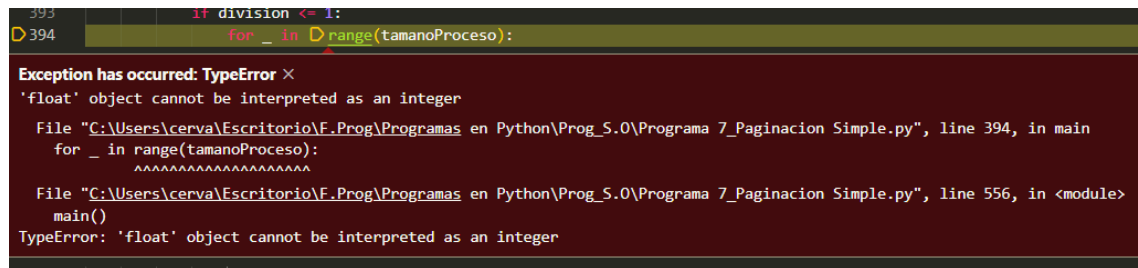
ID	Ope	Res
1	27-74	Error
2	23**6	Error

Ilustración 14. Memoria Liberada.

Programa 7. Paginación Simple

Errores y Soluciones.

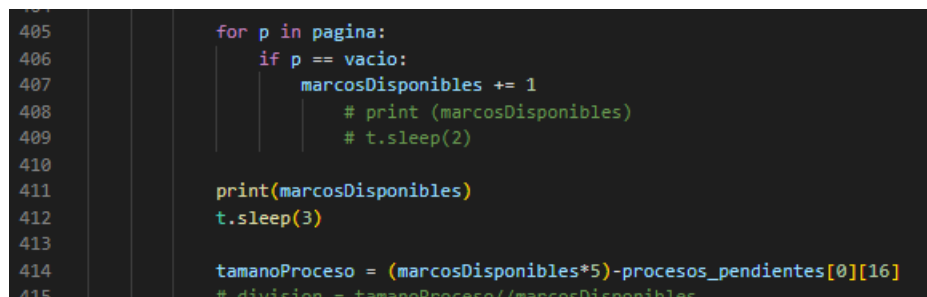
Verificación si el proceso entraba en paginación: Para poder determinar si un proceso podría pasar de pendiente a listo, se optó por usar una formula en la cual se verificaba los marcos disponibles en nuestra página, luego estos se multiplicaban por 5, para así mediante una resta saber si el tamaño que trae nuestro proceso, podía entrar, sin embargo, a la hora de ejecutar el código, obtuvimos el siguiente error.



```
393         if division <= 1:
394             for _ in range(tamanoProceso):
Exception has occurred: TypeError ×
'float' object cannot be interpreted as an integer
File "C:\Users\cerva\Escritorio\F.Prog\Programas en Python\Prog 5.0\Programa 7_Paginacion Simple.py", line 394, in main
    for _ in range(tamanoProceso):
File "C:\Users\cerva\Escritorio\F.Prog\Programas en Python\Prog 5.0\Programa 7_Paginacion Simple.py", line 556, in <module>
    main()
TypeError: 'float' object cannot be interpreted as an integer
```

Ilustración 15.Error 1

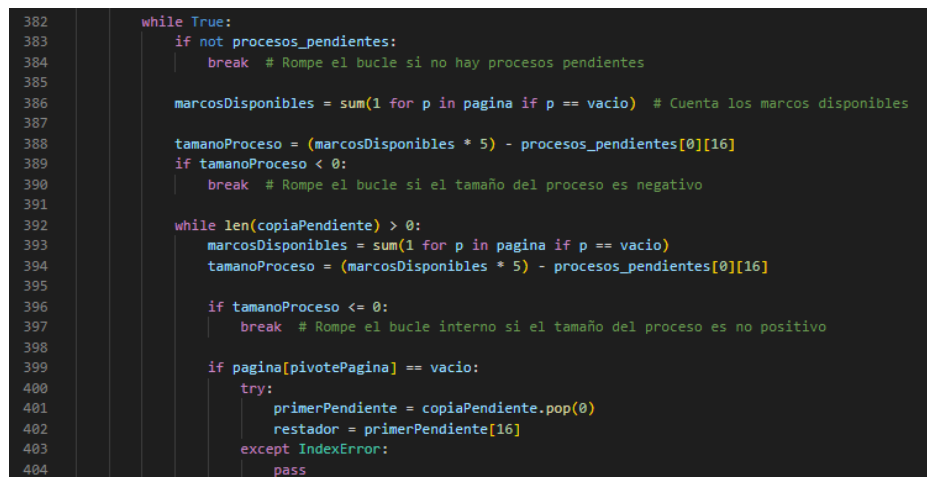
Detectando rapidamente que el error se debía a la creación de un flotante y querer hacer uso del mismo en una estructura for.



```
405         for p in pagina:
406             if p == vacio:
407                 marcosDisponibles += 1
408                 # print (marcosDisponibles)
409                 # t.sleep(2)
410
411         print(marcosDisponibles)
412         t.sleep(3)
413
414         tamanoProceso = (marcosDisponibles*5)-procesos_pendientes[0][16]
415         # division = tamanoProceso/marcosDisponibles
```

Ilustración 16.Causa del Error 1

Solucionando dicho error solamente modificando la formula utilizada para sacar la relación de si el proceso por entrar cabe en la página.



```
382         while True:
383             if not procesos_pendientes:
384                 break # Rompe el bucle si no hay procesos pendientes
385
386             marcosDisponibles = sum(1 for p in pagina if p == vacio) # Cuenta los marcos disponibles
387
388             tamanoProceso = (marcosDisponibles * 5) - procesos_pendientes[0][16]
389             if tamanoProceso < 0:
390                 break # Rompe el bucle si el tamaño del proceso es negativo
391
392             while len(copiaPendiente) > 0:
393                 marcosDisponibles = sum(1 for p in pagina if p == vacio)
394                 tamanoProceso = (marcosDisponibles * 5) - procesos_pendientes[0][16]
395
396                 if tamanoProceso <= 0:
397                     break # Rompe el bucle interno si el tamaño del proceso es no positivo
398
399                 if pagina[pivotePagina] == vacio:
400                     try:
401                         primerPendiente = copiaPendiente.pop(0)
402                         restador = primerPendiente[16]
403                     except IndexError:
404                         pass
```

Ilustración 17.Solución Error 1

Mostrar datos del próximo proceso pendiente por entrar: Para poder hacer la impresión correspondiente del siguiente proceso pendiente por entrar al sistema, y así comprobar el porque no logró entrar en la paginación, lo hicimos por medio de una estructura for, sin embargo, así fue como obtuvimos nuestro segundo error.

```

198     for proceso in procesos_pendientes[0]:
199         print("ID: ", proceso[0], "Tamaño: ", proceso[3])

Exception has occurred: TypeError ×
'int' object is not subscriptable
File "C:\Users\cerva\Escritorio\F.Prog\Programas en Python\Prog_S.O\Programa 7_Paginacion Simple.py", line 199, in print("ID: ", proceso[0], "Tamaño: ", proceso[3])
~~~~~
File "C:\Users\cerva\Escritorio\F.Prog\Programas en Python\Prog_S.O\Programa 7_Paginacion Simple.py", line 199, in imprimir()
File "C:\Users\cerva\Escritorio\F.Prog\Programas en Python\Prog_S.O\Programa 7_Paginacion Simple.py", line 199, in main()
TypeError: 'int' object is not subscriptable

```

Ilustración 18.Error 2

Una vez analizado dicho error, pudimos detectar que solo nos interesaba imprimir el primer proceso pendiente, no todos, así como lo estabamos realizando. Detecando de esta manera la causa de dicho error.

```

198     for proceso in procesos_pendientes[0]:
199         print("ID: ", proceso[0], "Tamaño: ", proceso[3])

```

Ilustración 19.Causa Error 2

Finalmente, dicho error logramos resolverlo correctamente, mediante la siguiente forma.

```

204     if procesos_pendientes:
205         proceso = procesos_pendientes[0]
206         print("\nProceso Pendiente a Entrar:\nID: ", proceso[0], "Tamaño: ", proceso[16])
207

```

Ilustración 20.Solución Error 2

Conclusiones

Maria Dolores.

El programa de esta semana tuvo un poco de complejidad debido a que implementar la paginación en cada proceso nos causó diversos errores en el algoritmo primero por la división de procesos por tamaño y querer hacerlo todo en una misma lista, orillándonos por realizar copias, posteriormente por el paso de algoritmos entre las tablas y el cambio de estados porque no realizaba los cambios adecuadamente. Teniendo que acceder directamente al valor por los “ID” de marcos.

Pese a que nos tomó más de lo esperado el resolver este algoritmo, debido a la cantidad de errores presentados en la implementación de los procesos pendientes, antes de que pasaran a los procesos actuales/listos; logramos resolver cada uno de los problemas de la manera óptima y eficiente posible.

La paginación simple fue un tema interesante y complejo, pero nos ayuda a comprender las particiones que se realizan en memoria tras cada tarea solicitada. En compañía de mi compañero se estableció una estructura óptima, trabajamos bien a lo largo del semestre y esperamos la culminación satisfactoria de la materia.

Aldo Fernando.

En lo personal, este he sentido que ha sido el programa de llevar a cabo, ya que, la estructura que tenía nuestro programa se vio muy afectada por la manera en cómo habíamos realizado los anteriores cambios, porque si bien el programa era bueno, contaba con muchas condicionales para rectificar su correcto funcionamiento.

Sin embargo, creo que a pesar de encontrarnos con diversos problemas en la creación y modificación del código se logró llegar al resultado esperado. Además, afortunadamente he logrado comprender en su totalidad el funcionamiento de la paginación simple, y como es que se le van asignando procesos dentro de la misma.

Finalmente, me quedó muy satisfecho del trabajo realizado en compañía de mi compañera, ya que, a base de complementos y lluvias de ideas, logramos obtener el resultado más óptimo posible.