

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de ciencias computacionales.

Ingeniería en computación



Seminario de Solución de Problemas de Sistemas Operativos - D01.

Violeta del Rocío Becerra Velázquez

“Programa 2. Simular el procesamiento por lotes con Multiprogramación.”

2023A

17/02/2023

Flores Estrada Abraham Miguel Angel

Olguín Hernández Jair Benjamín

Índice.

Índice.	2
Objetivo.	3
Lenguaje utilizado.	3
Desarrollo del programa.	3
Video del programa:	9

Objetivo.

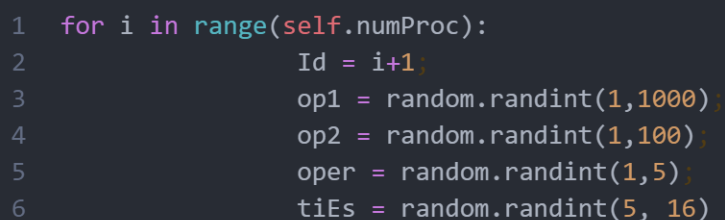
El procesamiento por lotes con multiprogramación es una técnica utilizada en los sistemas informáticos para ejecutar varios trabajos de manera eficiente en la unidad central de procesamiento sin la necesidad de intervención humana. Permite que varios lotes de trabajos se carguen en la memoria del sistema al mismo tiempo y que se ejecuten en paralelo. Por lo que buscamos crear un programa que pueda hacer la simulación del procesamiento por lotes con multiprogramación. Ya pudimos simular el procesamiento por lotes, pero ahora tendrá que hacer que todas se carguen al mismo tiempo y así mismo pueda hacer la simulación de qué pasaría si llegara a tener una interrupción, pausa o error.

Lenguaje utilizado.

Continuamos con el uso de python, debido a que esta actividad se puede desarrollar sobre el código pasado, y contamos con la facilidad de utilizar una de las propias funciones de qt para la detección de las teclas. A pesar de que nunca habíamos hecho uso de esta función, no fue tan complicado el aprender a usarla y el hacer que funcionara para nuestro programa. Ya que dentro de pyqt5 existe mucha documentación y diferentes ejemplos que utilizamos para guiarnos durante el desarrollo.

Desarrollo del programa.

Debido a los requerimientos del programa, se cambiaron tanto interfaces como funciones del programa. Uno de los mayores cambios fue el de la información de cada trabajo, ya que estas se deben generar de manera aleatoria y automática, donde solo tenemos que pedir el número de procesos. Lo que hacemos en código es utilizar la función de random que nos da dos números para los dos operadores, un número del 1 al 5 que nos ayudará a escoger una operación y el tiempo esperado que puede ser un número de 5 a 16.

A screenshot of a code editor with a dark background and syntax-highlighted Python code. The code is a loop that iterates over a range of processes. Each iteration initializes an ID, two random values for operators (op1 and op2), a random operation (oper), and a random time (tiEs).

```
1  for i in range(self.numProc):  
2      Id = i+1;  
3      op1 = random.randint(1,1000);  
4      op2 = random.randint(1,100);  
5      oper = random.randint(1,5);  
6      tiEs = random.randint(5, 16)
```

Uno de los cambios en las interfaces fue dentro de la primera interfaz, donde ahora solo pediremos el número de procesos. Está validado para que el número sea mayor que 0, para después de esto entre en el for anterior y genera en automático todos los procesos. Y al igual que el programa anterior, este se encarga de separarlos en lotes de 4 procesos y agregarlos a lista de procesos pendientes.

The screenshot shows a window titled 'MainWindow' with the title 'Simulador de procesos'. It features a label 'Num de procesos:' followed by a text input field containing the number '0' and a small up/down arrow control. To the right of the input field is a button labeled 'Aceptar'.

Una vez con nuestra lista de procesos separadas por lotes hará el mismo procedimiento de ir recorriendo nuestra lista de procesos primero por cada lote para después ir agregándolo a una lista auxiliar que guarda los procesos terminados. Esta es nuestra siguiente pantalla, una vez aquí podemos ver que están los lotes pendientes, así como los procesos del lote en ejecución que están pendientes. Y los distintos procesos que vayan terminando. Así mismo ya se ve el tiempo total, tiempo en ejecución y tiempo restante del proceso en ejecución.

The screenshot shows the 'Simulador de procesos' window with the following components:

- Lotes pendientes:** A label with a value of '2' in a light blue box.
- Procesos Terminados:** A label above a table with columns: Lote, Id, Op, Res.
- Lote en Ejecución:** A table with columns: Id, TR, TT. It contains three rows of data.

	Id	TR	TT
1	2	10	10
2	3	4	4
3	4	6	6
- Proceso en Ejecución:** A table with columns: Id, Operacion, Tiempo Total, Tiempo Ejecucion, Tiempo Restante. It contains one row of data.

	Id	Operacion	Tiempo Total	Tiempo Ejecucion	Tiempo Restante
1	1	472 * 81	13	1	12
- Botones:** An 'Iniciar' button at the bottom center and a 'Contador Global:' label with a value of '1' in a light blue box at the bottom left.

Esta es una de las partes fundamentales ya que en esta interfaz podemos poner a prueba las distintas funcionalidades agregadas que son las que nos ayudan a simular los distintos estados.

Interrupción(Tecla I):

En este caso la tecla I nos ayuda a simular una interrupción, donde en caso de que se presione el proceso que esté en ejecución se verá interrumpido y de pasar a estar en ejecución se irá al último lugar del lote. En la imagen podemos ver que el proceso con ID de 1 está en ejecución, por lo que si se interrumpe volverá a la lista del lote y en el proceso en ejecución se pondrá el proceso con ID de 2.

Simulador de procesos

Lotes pendientes: **2**

Lote en Ejecución

	Id	TR	TT
1	3	6	6
2	4	2	2
3	1	10	13

Proceso en Ejecución

	1
Id	2
Operacion	901 - 39
Tiempo Total	3
Tiempo Ejecucion	2
Tiempo Restante	1

Procesos Terminados

Lote	Id	Op	Res
------	----	----	-----

Contador Global: **5**

Iniciar

Como podemos ver, el proceso de ID de 1 vuelve a estar en la cola de ejecución, y continua con el proceso de ID de 2.

Simulador de procesos

Lotes pendientes: **2**

Lote en Ejecución

	Id	TR	TT
1	1	9	13

Proceso en Ejecución

	3
Id	3
Operacion	450 * 68
Tiempo Total	6
Tiempo Ejecucion	1
Tiempo Restante	5

Procesos Terminados

Lote	Id	Op	Res
1	2	901 - 39	862
2	4	975 * 60	58500

Contador Global: **10**

Iniciar

También podemos notar que el tiempo restante (TR) del proceso 1 en la primera imagen es de 12, y cuando se interrumpe y regresa a la cola su tiempo es de 10. Pero el tiempo total (TT) es mayor debido a que cuenta el tiempo que pasó en ejecución. Veamos otro ejemplo con el mismo lote.

Se aplica una interrupción:

Simulador de procesos

Lotes pendientes: **2**

Lote en Ejecución

	Id	TR	TT
1	3	1	6

Proceso en Ejecución

	1
Id	1
Operacion	704 * 34
Tiempo Total	13
Tiempo Ejecucion	5
Tiempo Restante	8

Procesos Terminados

	Lote	Id	Op	Res
1	1	2	901 - 39	862
2	1	4	975 * 60	58500

Contador Global: **15**

Iniciar

Vemos que el proceso en ejecución pasa de ser el 3 a ser el 1, y el único proceso que está en la cola de lote en ejecución es el 3. Debido a que el 2 y 4 han sido terminados.

Error (Tecla E):

En este caso la tecla E es la que nos ayudará a simular que un proceso tuvo un error, por lo que se considera terminado y nos señala que tuvo error. En la imagen podemos ver que mandamos a 3 de los 4 procesos a error, por lo que nos la mandará directamente a la lista de procesos terminados.

Simulador de procesos

Lotes pendientes: **0**

Lote en Ejecución

	Id	TR	TT
1	10	1	1
2	11	5	5

Proceso en Ejecución

	1
Id	9
Operacion	126 * 92
Tiempo Total	9
Tiempo Ejecucion	2
Tiempo Restante	7

Procesos Terminados

	Lote	Id	Op	Res
1	1	2	901 - 39	862
2	1	4	975 * 60	58500
3	1	1	704 * 34	23936
4	1	3	450 * 68	30600
5	2	5	948 * 26	ERROR
6	2	6	342 * 18	ERROR
7	2	7	319 / 34	9.382352941176471
8	2	8	180 / 72	ERROR

Contador Global: **34**

Iniciar

Pausa y Continuar (Tecla P y Tecla C):

En este caso las teclas P y C nos ayudarán a simular una pausa y continuación respectivamente, en este caso nos apoyaremos de la consola para mostrar que se han presionado las teclas. En consola se nos imprimirá la tecla que se ha presionado, donde si presionamos alguna tecla diferente de c o p, no nos imprimirá nada en pantalla.

Simulador de procesos

Lotes pendientes: 0

Procesos Terminados

Lote	Id	Op	Res
1	2	901 - 39	862
2	4	975 * 60	58500
3	1	704 * 34	23936
4	3	450 * 68	30600
5	5	948 + 26	ERROR
6	6	342 * 18	ERROR
7	7	319 / 34	9.382352941176471
8	8	180 / 72	ERROR

Contador Global: 38

Simulador de procesos

Lotes pendientes: 0

Procesos Terminados

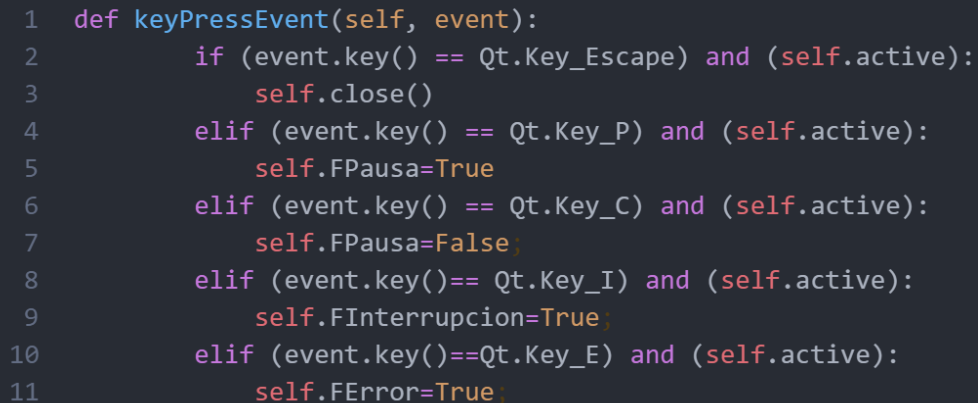
Lote	Id	Op	Res
1	2	901 - 39	862
2	4	975 * 60	58500
3	1	704 * 34	23936
4	3	450 * 68	30600
5	5	948 + 26	ERROR
6	6	342 * 18	ERROR
7	7	319 / 34	9.382352941176471
8	8	180 / 72	ERROR
9	9	126 * 92	218
10	10	205 - 22	183

Contador Global: 42

Console 2/A X

```
Presionaste la tecla: c
True
Presionaste la tecla: p
Presionaste la tecla:
Presionaste la tecla:
Presionaste la tecla:
Presionaste la tecla:
False
Presionaste la tecla: c
True
Presionaste la tecla: p
Presionaste la tecla:
Presionaste la tecla:
```

Como se mencionó al principio, dentro de las funciones de PyQt5 tenemos una que nos ayuda a detectar las teclas, esta se llama "KeyPressEvent" la cual es una función que detecta algún evento, en este caso el evento es cualquier tecla presionada. Donde tenemos una bandera para cada tecla (I,E,P,C) que estas nos ayudaran a saber si un evento debe ocurrir o no.

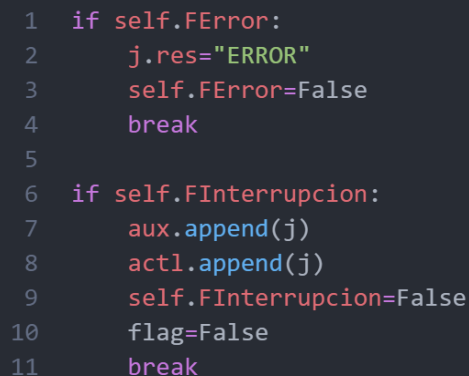


```

1  def keyPressEvent(self, event):
2      if (event.key() == Qt.Key_Escape) and (self.active):
3          self.close()
4      elif (event.key() == Qt.Key_P) and (self.active):
5          self.FPausa=True
6      elif (event.key() == Qt.Key_C) and (self.active):
7          self.FPausa=False;
8      elif (event.key()== Qt.Key_I) and (self.active):
9          self.FInterrupcion=True;
10     elif (event.key()==Qt.Key_E) and (self.active):
11         self.FError=True;

```

Donde los casos más sencillos son los de pausa y continuar, ya que estas solamente se encargan de parar o reanudar todo. Estas dos opciones utilizan la misma bandera si se presiona P la bandera se pone en verdadero y pausara el programa, caso contrario si se presiona C, la bandera se pone en falso y el programa continua normal.



```

1  if self.FError:
2      j.res="ERROR"
3      self.FError=False
4      break
5
6  if self.FInterrupcion:
7      aux.append(j)
8      act1.append(j)
9      self.FInterrupcion=False
10     flag=False
11     break

```

En caso del error, si se activa el resultado del proceso en curso que en este caso es j pasa a ser Error, vuelvo a cambiar la bandera a falso y se termina la ejecución de ese proceso. Y en el caso de la interrupción si se activa la se agrega el proceso con la información que tiene en ese momento a la misma lista donde se guarda el lote en proceso. Y con la segunda flag nos ayuda a que el proceso no se guarde en los procesos terminados.

Conclusiones.

A pesar de que al comienzo del desarrollo del programa teníamos dudas acerca de cómo íbamos a capturar las teclas, no tuvimos tantos problemas durante el desarrollo del programa. Ya que una vez que encontramos sobre la función que nos ofrecía PyQt5 para la captura de las teclas, lo demás fue más sencillo, ya que como se mencionó existe mucha documentación y ejemplos sobre esta biblioteca. Y en comparación del desarrollo del programa anterior, en este caso ya teníamos conocimientos de cómo podíamos hacer las funcionalidades en código. Y sentimos que realmente se cumplió el poder simular un programa de lotes con multiprogramación que pueda pasar por distintos estados como error o llegar a tener un interrupción.

Video del programa:

En YouTube: <https://youtu.be/dvm4ZpD9U0k>

En drive:

<https://drive.google.com/file/d/1yzAsNSF3KfzYLn9cEA0nSc27eD5ZlhmP/view?usp=sharing>