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 7. Paginación simple”

2023A

11/04/2023

Flores Estrada Abraham Miguel Angel

Olguín Hernández Jair Benjamín

Índice.

Objetivo.

Recordemos que la paginación simple es una técnica básica de gestión de memoria que se utiliza en los sistemas operativos. En la paginación simple la memoria se divide en bloques de tamaño fijo llamados páginas y se divide en marcos de página de tamaño idéntico al de la memoria física. Cuando se carga un proceso en la memoria, se divide en páginas de tamaño fijo. Si una página del proceso no está en la memoria, se produce un fallo de página y la página se carga desde el disco a un marco de página libre en la memoria. Si no hay marcos de página libres en la memoria, se selecciona uno de los marcos existentes y se sobrescribe. Por lo que buscamos crear un programa que simule el uso de la paginación simple junto al algoritmo round robin, pero ahora tendremos un espacio definido de memoria y cada proceso tendrá su tamaño definido, que puede ser un número aleatorio entre 6 y 25. Haciendo que tengamos que tener en cuenta que podemos tener diferente cantidad de procesos en memoria y que ahora tenemos 38 marcos disponibles para los diferentes procesos.

Lenguaje utilizado.

Haremos uso de python debido a la facilidad que nos da para crear las interfaces y así mismo poder hacer uso del algoritmo round robin que anteriormente habíamos creado, donde solamente tengamos que agregar las funcionalidades solicitadas.

Desarrollo del programa.

Para empezar el desarrollo del programa tuvimos que cambiar la interfaz principal donde ahora agregamos nuestra tabla que demuestra la paginación de todos los procesos donde más adelante podremos ver que muestra los procesos que entraron en memoria, así como cuantos marcos ocupa cada proceso contando con los reservados para el sistema operativo. Donde uno de los primeros cambios que tuvimos que hacer fue al momento de tener nuestros procesos listos, ya que anteriormente solamente teníamos 4, y ahora podemos tener hasta 38 si tuvieran un tamaño de 5 cada uno. Lo que hacemos es tener una variable de “disponibles” que es la cantidad de páginas disponibles actualmente, por lo que irá agregando procesos mientras que la cantidad de marcos disponibles por 5 sea mayor al tamaño del proceso.

```
while(act<numProc):
    if (self.Dispon*5) >= self.procesos[act].sz:
        self.addProcMar(self.procesos[act]);
        listos.append(self.procesos[act])
    else:
        break;
    act+=1;
    i+=1
self.actTablePage();
```

De ahí mandamos el proceso actual a nuestra función “addProcMar”, que la función encargada de dividir el proceso en sus correspondientes marcos. Lo que hace es iterar a través de la lista de páginas de memoria, si encuentra un marco libre actualiza su estado al estado del proceso actual y le asigna el ID del proceso. Luego, la función actualiza el espacio disponible en la página de memoria. Si el tamaño del proceso no es divisible por 5, actualiza el espacio disponible en la página de memoria con la cantidad restante y resta esa cantidad del tamaño del proceso. Si el tamaño del proceso es divisible por 5, actualiza el espacio disponible en la página de memoria a "5/5" y resta 5 del tamaño del proceso. Y por último disminuye la cantidad de marcos disponibles.

```
def addProcMar(self, proc):
        sz=proc.sz
        for page in self.Paginacion:
            if sz<=0:
                break;
            elif page.status=="Libre":
                page.status=proc.status;
                page.proceso=str(proc.id)
                if (sz % 5) != 0:
                    page.Espacio= str(proc.sz % 5)+"/5";
                    sz -= (sz % 5);
                else:
                    page.Espacio= "5/5";
                    sz -= 5;
                self.Dispon-=1;
```

Recordemos que estamos utilizando el algoritmo de round robin, por lo que estamos trabajando con quantums. por lo que al igual que en ese programa, lo que hacemos

es iterar sobre el tiempo restante del proceso en ejecución de ahí lo que haremos es ir disminuyendo el valor del quantum hasta que sea menor que 0, después cambiará el status del proceso nuevamente a listo y el status de los marcos con la función actStatusPage, donde le tenemos que restar al ID uno para ajustar el índice de la lista. Así mismo volvemos a asignar el valor del quantum al contador. Y tenemos una bandera la cual nos ayuda a saber cuando el proceso ha sido terminado.

```
tp = int(j.Tr)
for k in range(tp):
    if(self.QuantumContador < 1):
        j.status="Listo"
        self.procesos[(j.id - 1)].status="Listo";
        listos.append(j);
        self.actStatusPage(j.status, j);
        self.QuantumContador=self.QuantumVal;
        flag=False;
        break;
```

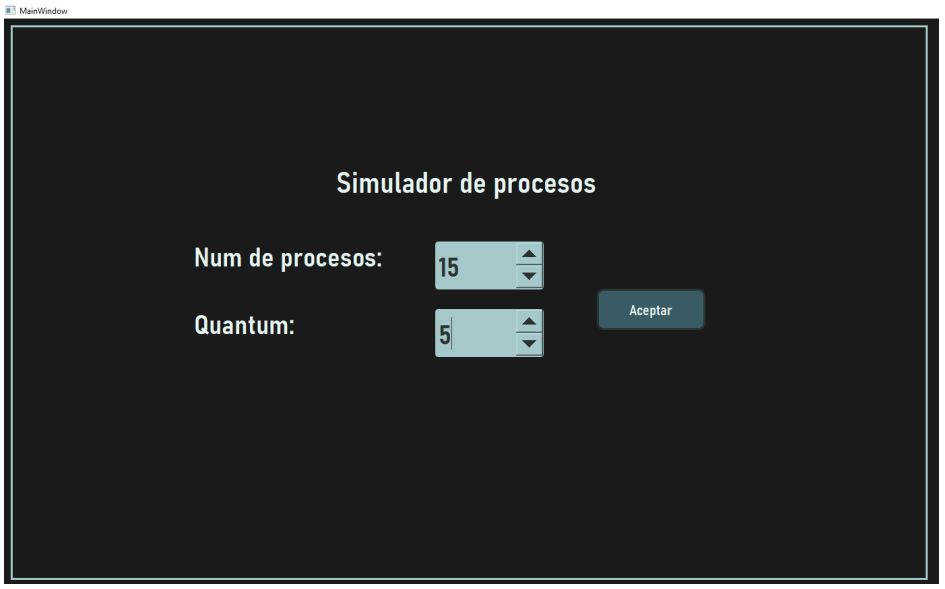
La función itera a través de la lista de páginas de memoria mientras que sz sea menor o igual a 0, la función no sale del bucle. Si la página actual está asignada al proceso p, la función actualiza su estado al valor especificado en status. Luego, la función actualiza la variable sz para reflejar el tamaño restante de la memoria que se asignará al proceso. Si el tamaño restante no es divisible por 5, la función reduce el tamaño a su módulo 5. De lo contrario, la función resta 5 del tamaño. Esta función es muy similar a addProcMar, solamente que esta última se llama cada vez que el proceso sale de ejecución una vez que se termina su tiempo del quantum, guardando el tiempo que pasó en ejecución y lo actualiza en la tabla de marcos.

```
def actStatusPage(self,status,p):
    sz = p.sz;
    for page in self.Paginacion:
        if sz<=0:
            break;
        elif page.proceso == str(p.id):
            page.status=status;
            if (sz % 5) != 0:
                sz -= (sz % 5);
            else:
                sz -= 5;
    self.actTablePage();
```

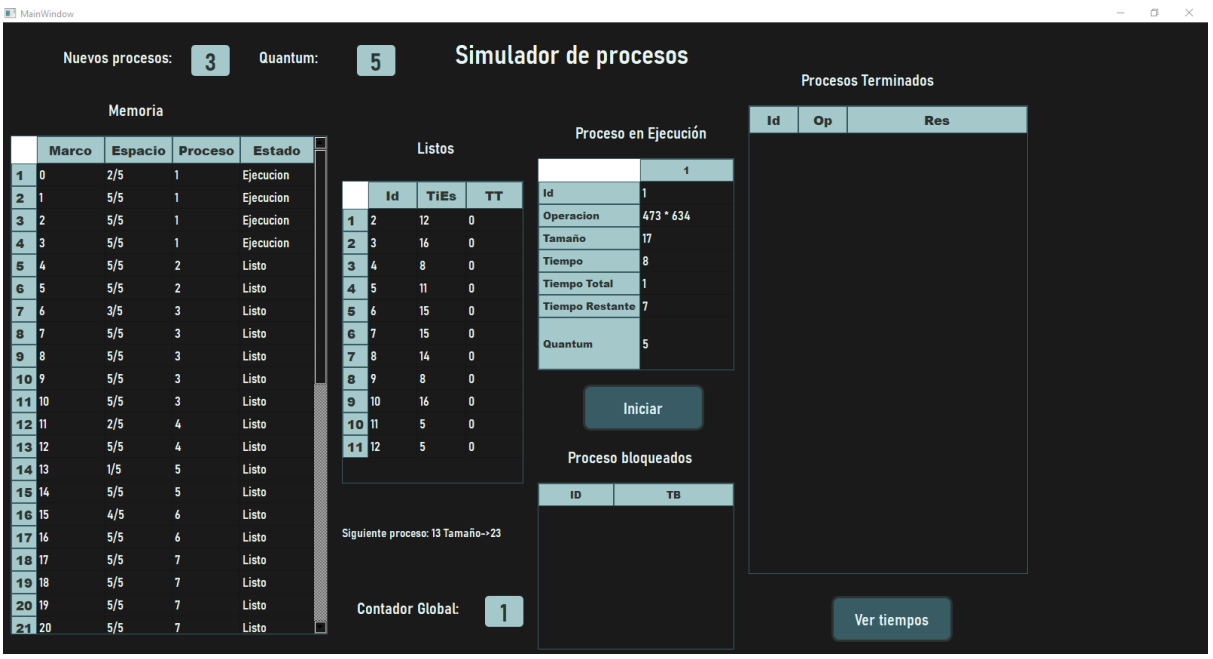
Por último tenemos la función delProcMar que esta elimina el proceso una vez que haya terminado, donde iteramos sobre la lista de páginas de memoria mientras que sz sea menor o igual a 0, de ahi vemos si la página actual está asignada al proceso, y si es asi actualizamos el estado de la página a libre y el espacio a “0/5”, es decir indicamos que esta pagina está libre o vacía. Después simplemente actualizamos el tamaño restante de la memoria que se liberará del proceso y sumamos 1 a la cantidad de páginas disponibles.

```
def delProcMar(self, p):
    sz = p.sz;
    for page in self.Paginacion:
        if sz<=0:
            break;
        elif page.proceso == str(p.id):
            page.proceso = "Libre"
            page.status ="Libre"
            page.Espacio= "0/5";
            if (sz % 5) != 0:
                sz -= (sz % 5);
            else:
                sz -= 5;
            self.Dispon+=1;
    self.actTablePage();
```

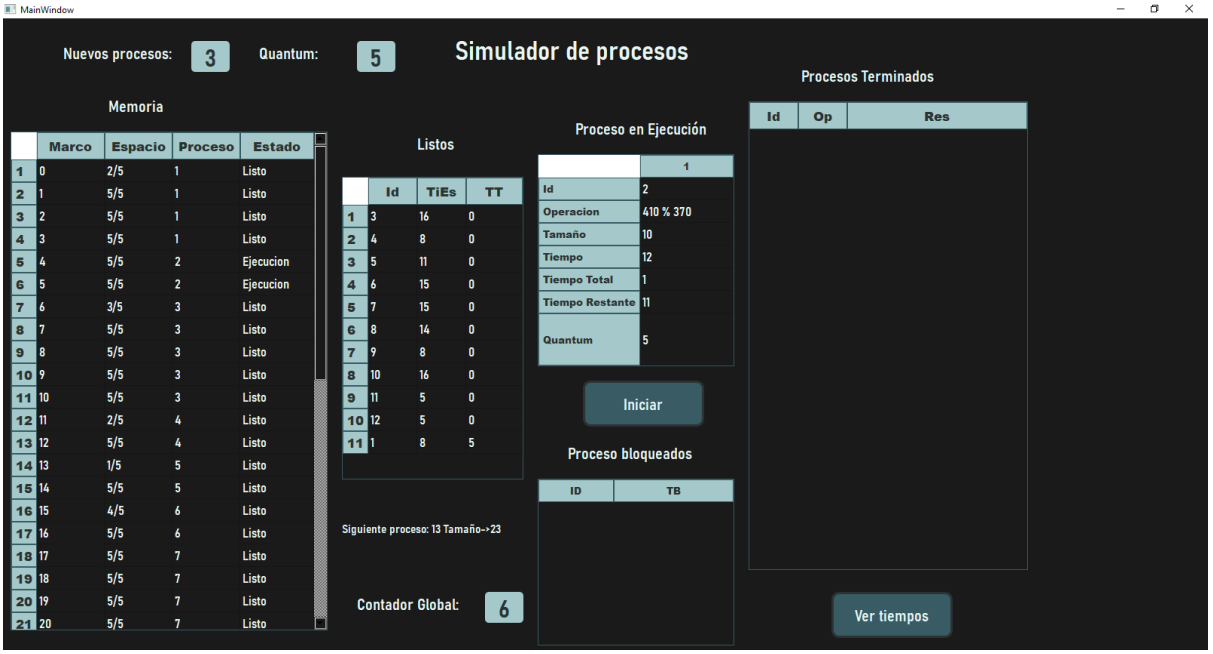
Veamos la ejecución del programa, donde priemeramente nos pide el numero de procesos y el tamaño que tendra el quantum, en este caso pondremos 15 procesos y un valor de 5.



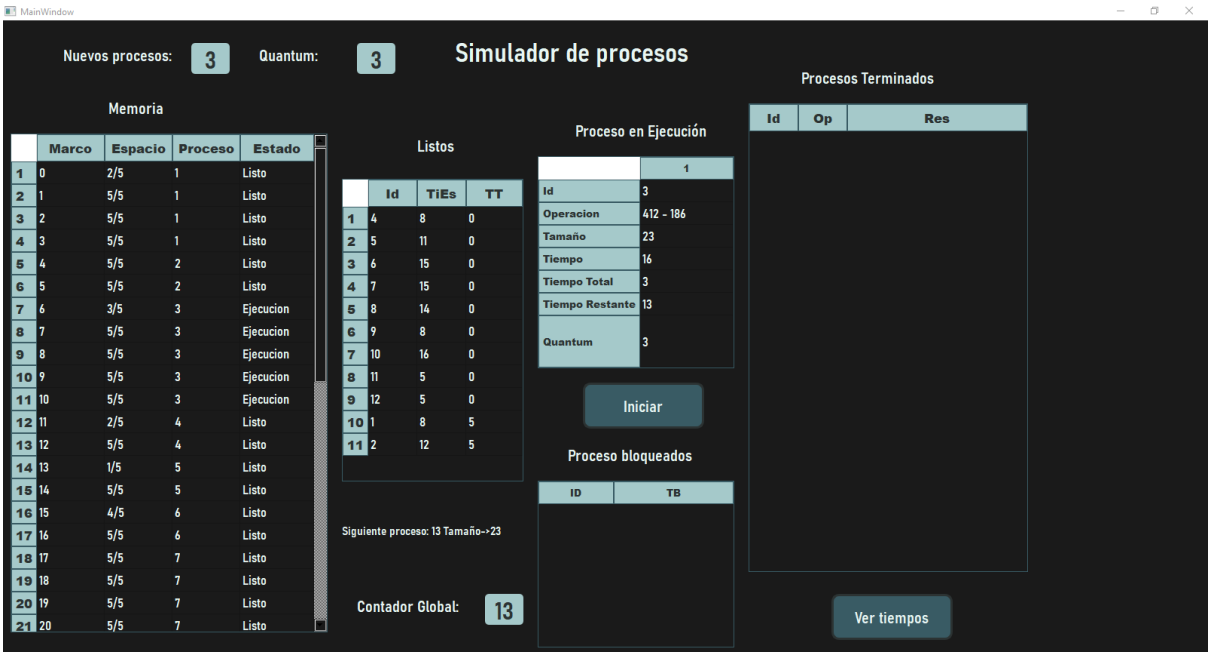
Una vez iniciemos podremos ver primeramente toda la tabla de páginas donde cada proceso estará divididos en n marcos, el proceso en ejecución donde se muestra el tiempo que lleva del quantum, los nuevos procesos que aún no entran en memoria y la información del siguiente proceso. De igual forma podemos ver que el proceso 1 tiene un tamaño de 17 y tiene los marcos del 0 al 3, donde el marco 0 tiene un espacio de 2, y los 1,2,3 tienen un espacio de 5.



Ahora si pasamos con el proceso número 2, podemos ver de igual forma su tamaño es de 10 y tiene 2 marcos de 5 espacios cada uno pero ahora su estado aparece en ejecución.

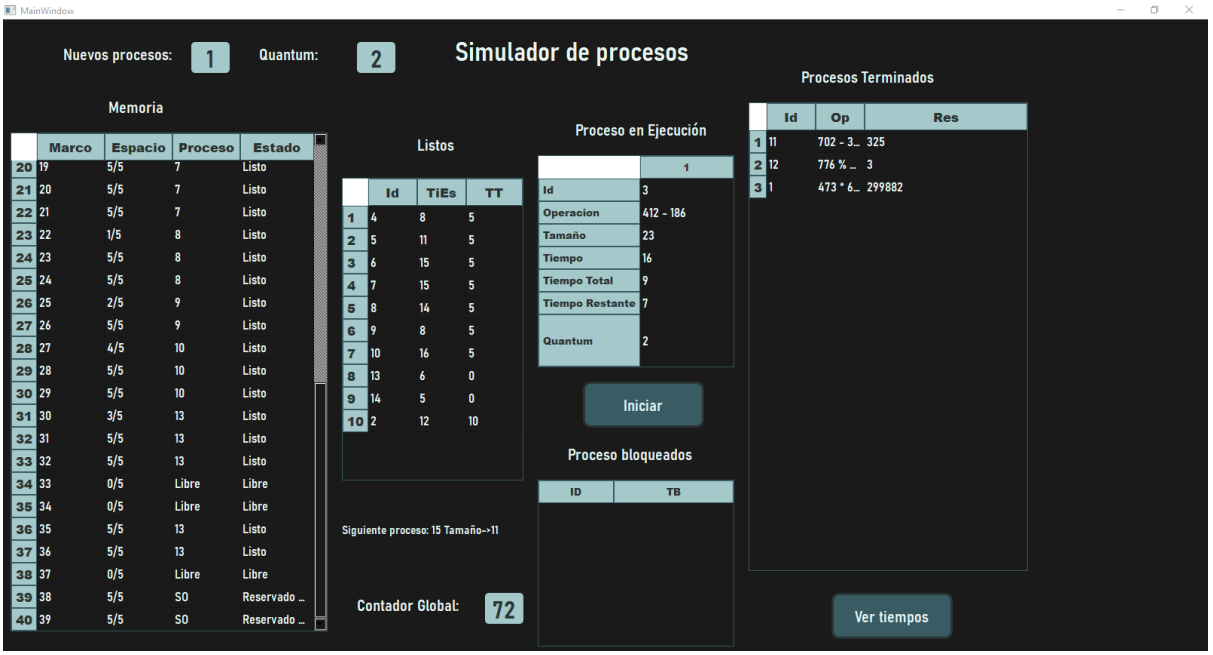


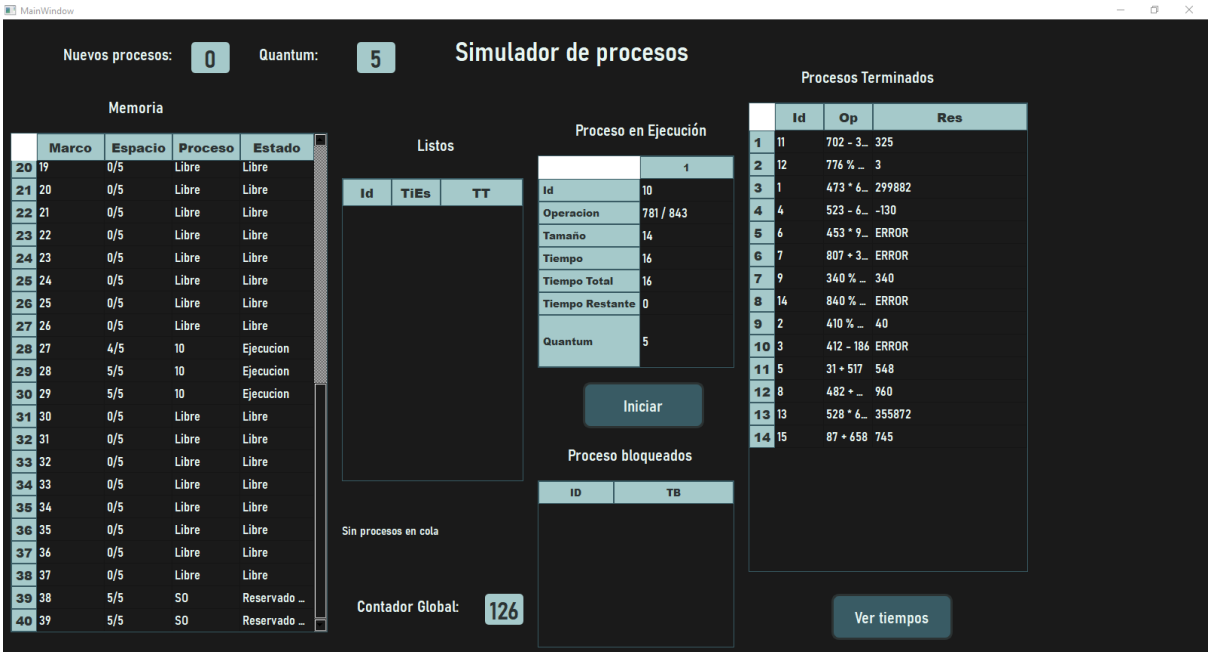
Ahora si pasamos con el proceso 3 podremos ver nuevamente el tamaño de 23, y este tiene 5 marcos, donde son 4 de 5 espacios y uno de 3.



De igual forma si bajamos en la tabla de memoria podemos ver que tenemos los últimos 2 proceso reservados para el sistema operativo y en cuanto avanza el programa se irán agregando procesos a los listos y en cuanto pasan por su tiempo de quantum se van al último lugar de la cola. Si dejamos que el programa continue

podemos ver que los marcos van quedando libres y algunos siguen listos para procesarse.





Conclusiones.

El desarrollo de este programa fue el más complicado hasta el momento, ya que estamos viendo un concepto mucho más especializado para los sistemas operativos, y donde lo teórico puede resultar mucho más sencillo que lo practico. Ya que en este caso podíamos aprovechar el programa anterior, donde utilizamos round robin y de ahí solamente modificarlo a lo que necesitábamos. Por lo que se nos facilitó una parte del trabajo, pero lo complicado fue el que este programa necesitaba de más validaciones, debido a que no todos los procesos tienen el mismo tamaño y no siempre se cuenta con el espacio en memoria suficiente para que entre el proceso, por lo que debimos contemplar varios casos que nos ayudaran a solucionar y controlar estos problemas. Pero aun así pudimos lograr hacer una simulación, donde se ve de manera adecuada el cómo es que funciona la paginación simple.