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 8. Suspendidos”

2023A

21/04/2023

Flores Estrada Abraham Miguel Angel

Olguín Hernández Jair Benjamín

Índice.

Objetivo.

Recordando la práctica donde vimos la paginación simple, esta es una técnica básica de gestión de memoria que se utiliza en los sistemas operativos. Dónde 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. Donde en el programa anterior logramos hacer la simulación de la paginación simple, por lo que ahora planeamos agregar una función principal, la de procesos suspendidos, estos son aquellos procesos que son detenidos pero sin ser terminados o eliminados, por lo que ahora al presionar la tecla S el proceso en bloqueados se agregara en un archivo que irá guardando hasta que se presione la tecla R para poder regresar.

Lenguaje utilizado.

Haremos uso de python debido a que continuaremos en base al programa anterior, haciendo solamente las modificaciones necesarias, debido a que es una continuación directa del programa anterior.

Desarrollo del programa.

Como se mencionó anteriormente, solo tuvimos que agregar una función principal para los procesos suspendidos, por lo que no tuvimos que realizar cambios en las interfaces sino que fueron más en cuestiones de código, donde primeramente cambiamos la ejecución del programa, antes el programa estaba en ejecución hasta que la cantidad de procesos sea igual a la cantidad de terminados, pero ahora tenemos que ver que no haya suspendidos. Después de esto simplemente tomamos el primer proceso y lo sacamos de la lista.

```
1  while (len(self.procesos) != len(self.done) or (self.susp != 0)):
2      if (len(listos)>0):
3          j=listos[0]
4          listos.pop(0)
```

Así mismo ahora tenemos dos banderas que se activará cuando presionamos la tecla S y R, por lo que verificamos que por lo menos haya un proceso en la lista de bloqueados, en dado caso tomamos el primer proceso y lo mandamos a una función para suspender el proceso, eliminamos el proceso suspendido de la lista de bloqueados, procesos y lo eliminamos de memoria.

```
if self.FSuspend:
    if len(self.bloq)!=0:
        p=self.bloq[0];
        p.Tb=0;
        self.suspender(p);
        self.bloq.pop(0);
        self.procesos.remove(p);
        self.delProcMar(p);
        act-=1;
```

Esta función se encarga de agregar el procesos a una lista auxiliar de las instancias, y después escribirlo en el documento. Usamos una lista para poder conservar el orden de los procesos suspendidos, donde si no hemos agregado ningún proceso suspendido simplemente agregamos el proceso a la lista y lo escribimos en el archivo. En dado caso que anteriormente ya hemos agregado algún otro proceso a suspendidos, lo que hacemos es leer todos los procesos del archivo y después escribimos.

```
def suspender(self,j):
    ins=[]
    if self.susp==0:
        ins.append(j);
        self.escribr_Instancias(ins);
    else:
        ins = self.leer_instancias();
        ins.append(j);
        self.escribr_Instancias(ins);
    self.susp+=1;
```

Después tenemos la bandera para retornar donde primeramente checamos que se hayan enviado a suspendidos, de ahí lo que hacemos es guardar en una variable el primer proceso con la función retornar, de ahí verificamos que exista el espacio en memoria para acomodar el proceso retornado, en caso de que si le asignamos valores como el tiempo de llegada y si este ha sido atendido.

```
if self.FReturn:
    if self.susp !=0:
        p = self.retornar();
        self.procesos.append(p);
        numProc = len(self.procesos)
        if(act<numProc) and ((self.Dispon*5) >= self.procesos[act].sz) and ((numProc-act)-1 ==0):
            if self.procesos[act].atendido == False:
                self.procesos[act].tLlegada=self.tg;
                self.procesos[act].atendido=True;
            listos.append(self.procesos[act])
            self.addProcMar(self.procesos[act]);
            act+=1;
        self.actualizarListos(listos);
    self.FReturn=False;
```

La función de retornar es totalmente la contraria a suspender, ya que de igual forma tenemos un arreglo auxiliar de las instancias, y ahora en vez de escribir en el archivo, leeremos lo que haya en este, por lo que en este caso podemos leer el archivo vacío y no afectará al funcionamiento. Por lo que sacamos el primer proceso de la lista auxiliar y regresará a la lista de procesos.

```
def retornar(self):
    ins=[]
    ins=self.leer_instancias();
    p=ins[0];
    p.status="Nuevo"
    ins.pop(0);
    self.susp-=1;
    self.escribr_Instancias(ins);
    return p;
```

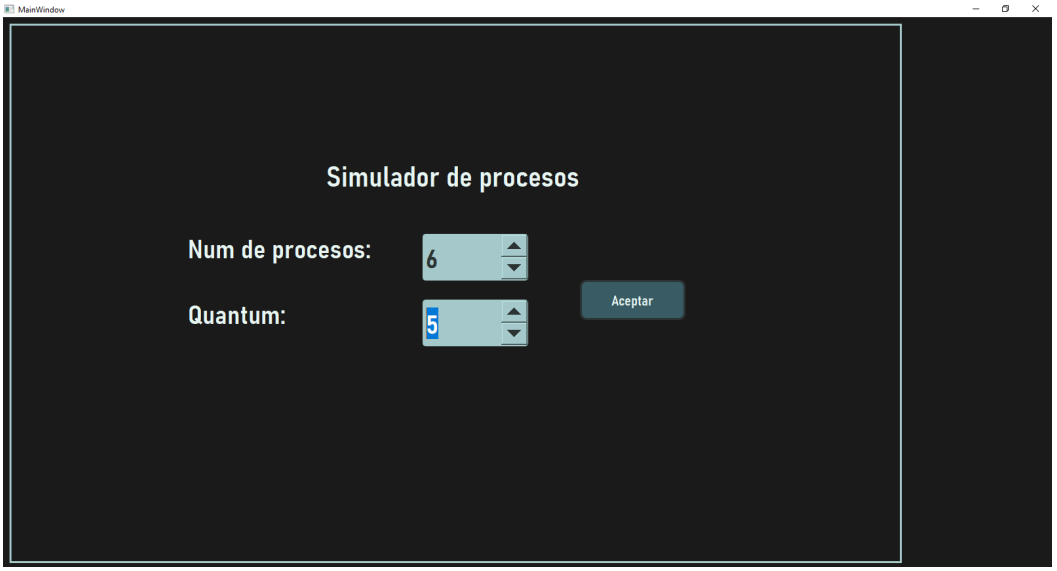
Por último tenemos nuestras funciones que nos ayudan a escribir y leer las instancias de los archivos. Primero tenemos la función para escribir, donde abrimos el archivo suspendidos.txt en modo de escritura, y en caso de que no exista el archivo lo creara. Después con el for itera sobre la lista de instancias y escribiendo sobre cada uno de los elementos y agrega un salto de línea al final.

```python
def escribr_Instancias(self,instancias):
    with open('suspendidos.txt', 'w') as archivo:
        for instancia in instancias:
            archivo.write(json.dumps(instancia.__dict__) + '\n')
```

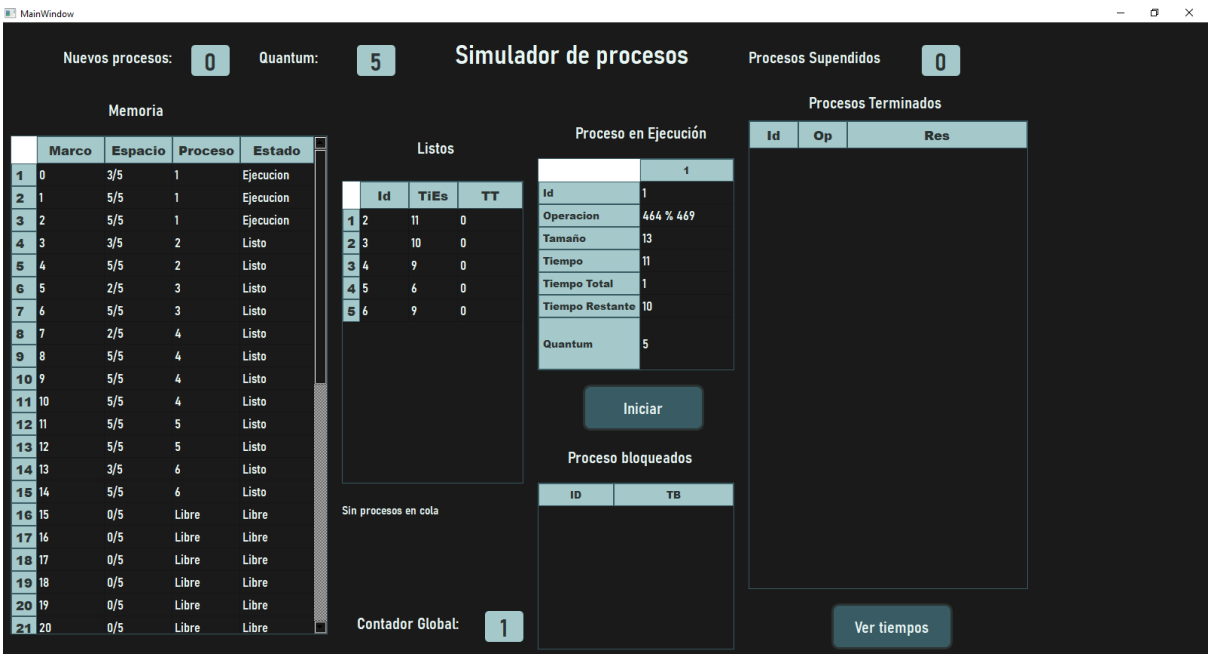
Después tenemos la función para leer, donde de igual forma abrimos el archivo suspendidos.txt pero esta vez será en modo lectura para después iterar sobre cada línea del archivo y asignar cada uno de los atributos al proceso. Para finalmente agregar a la lista y regresar a la lista final.

```python
def leer_instancias(self):
    instancias = []
    with open('suspendidos.txt', 'r') as archivo:
        for linea in archivo:
            instancia_dict = json.loads(linea)
            instancia = proceso(instancia_dict['id'],instancia_dict['operacion'],instancia_dict['res'],instancia_dict['tiEs'])
            instancia.atendido = instancia_dict['atendido']
            instancia.sz = instancia_dict['sz']
            instancia.status = instancia_dict['status']
            instancia.tLlegada = instancia_dict['tLlegada']
            instancia.tFinali = instancia_dict['tFinali']
            instancia.tRetorno = instancia_dict['tRetorno']
            instancia.tRespuesta = instancia_dict['tRespuesta']
            instancia.tEspera = instancia_dict['tEspera']
            instancia.Tservicio = instancia_dict['Tservicio']
            instancia.Tt = instancia_dict['Tt']
            instancia.Tr = instancia_dict['Tr']
            instancia.Tb = instancia_dict['Tb']
            instancias.append(instancia)
    return instancias
```

Veamos la ejecución del programa, donde de igual manera nos pedirá ingresar el número de procesos a simular y el valor que tendrá el quantum. En este caso ingresamos 6 procesos, ya que con el programa anterior vimos el funcionamiento de la paginación.

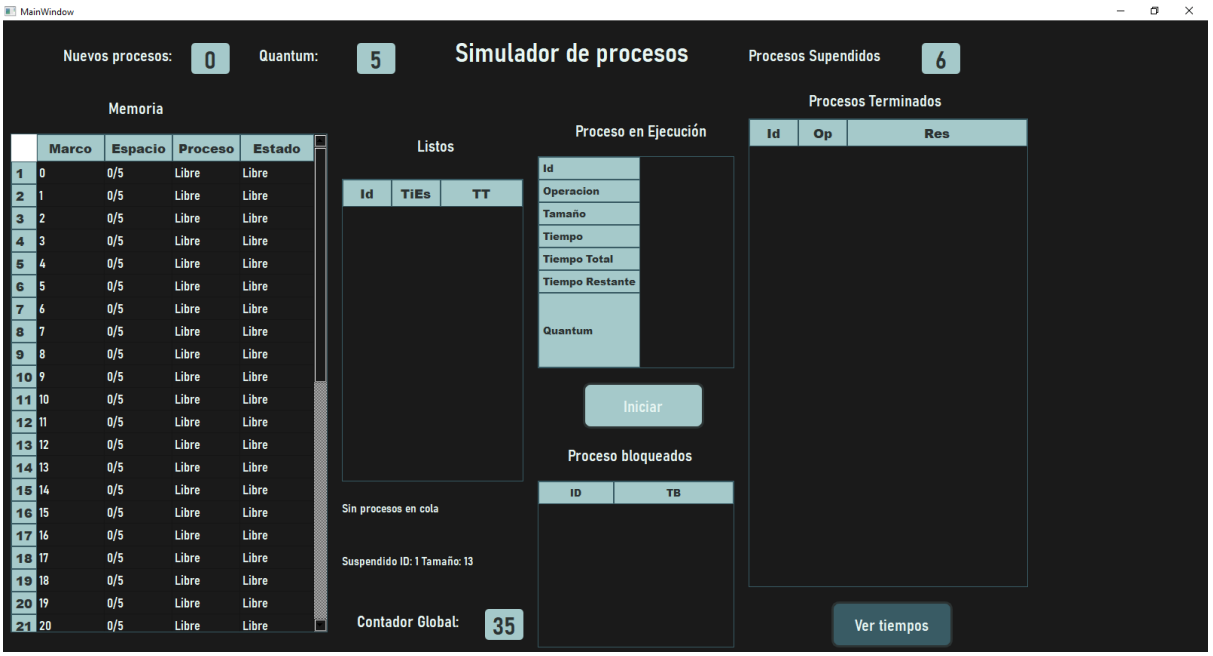


Una vez comencemos los procesos tendrán su espacio en memoria asignada, mostrándonos si están en ejecución o están listos

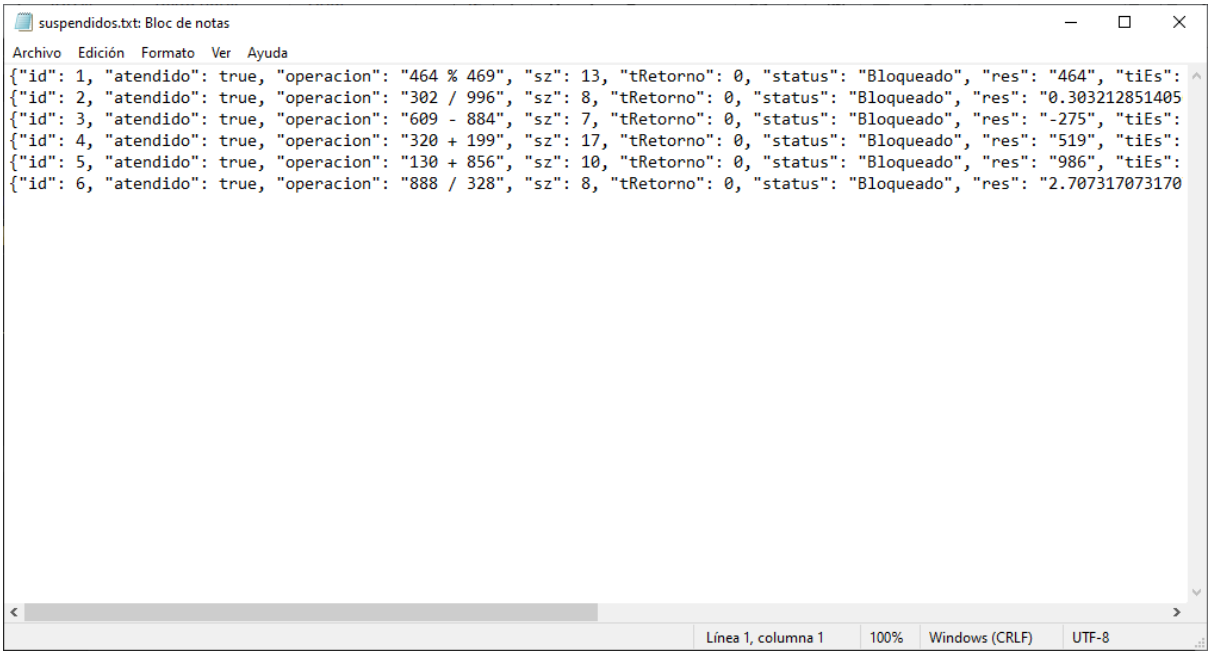


Lo que haremos será mandar todos los procesos a bloqueados, para después usar la función agregada de procesos suspendidos. Como vemos todos los procesos han sido enviados a bloqueados, y en la tabla de memoria aparecen con estado bloqueado.

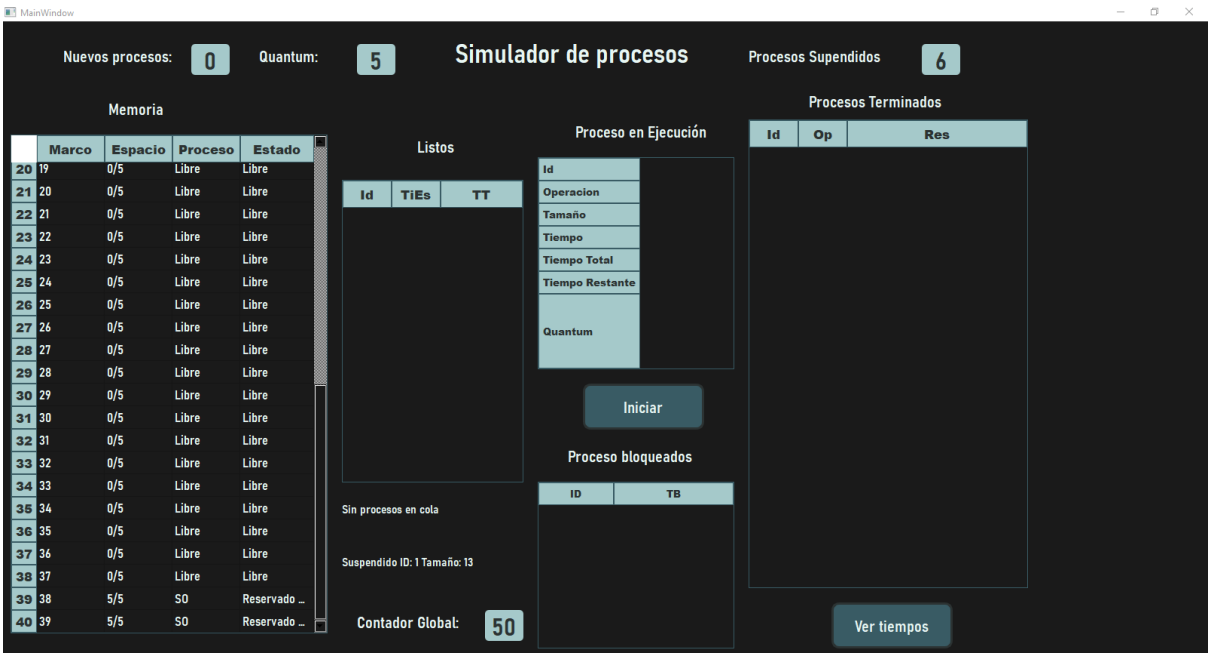
Por lo que al presionar la tecla S mandará el proceso 1 a suspendidos, lo que haremos será dejar el programa vacío, donde podemos ver que el tiempo puede continuar ya que el programa sigue en ejecución, y así mismo la tabla de memoria está totalmente vacía.



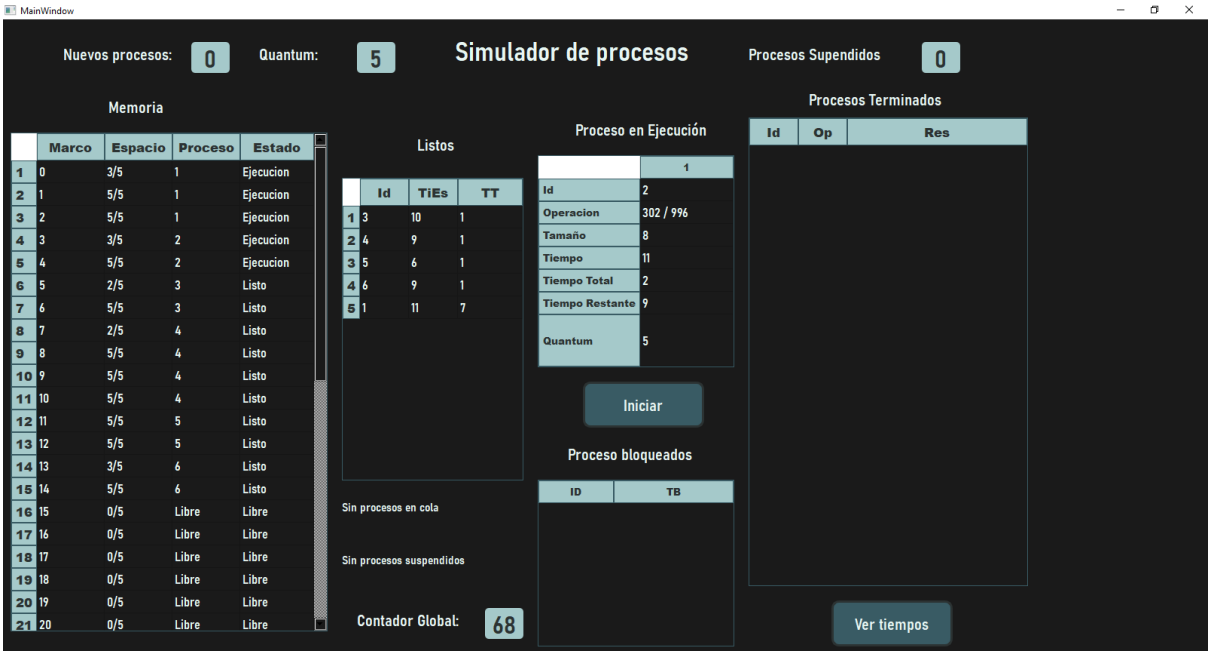
De igual manera, podemos ver el txt donde se guardaron los procesos suspendidos, estos están guardados con un salto de línea cada uno, y separando cada atributo con una coma.



Y también podemos notar que el programa sigue trabajando a pesar de no tener procesos en memoria, ya que estos procesos suspendidos no han sido terminados o eliminados, y que de igual forma tenemos toda la tabla de memoria vacía, únicamente con los espacios reservados para el sistema.



Y al regresarlos estos trabajarán de manera normal, hasta que todos terminen.



Conclusión.

A diferencia de los programas anteriores este es uno de los más sencillos, debido a que es una continuación directa del programa anterior. Donde ahora tiene la función de agregar procesos suspendidos, que podría parecer un poco más complicado ya que al inicio teníamos el concepto de procesos suspendidos como algo muy ambiguo, pero con el desarrollo del programa pudimos entender que estos procesos son aquellos han sido detenidos y se encuentran en espera para ser reanudados. Ya con el programa podemos entender cómo es que llegan a funcionar o suceder dentro de los sistemas operativos que conocemos, ya que se eliminan completamente de memoria siendo ignorados pero no eliminados, porque en este caso el programa sigue en ejecución hasta que estos procesos hayan sido terminados, ya sea por que han pasado su tiempo de ejecución o por algún error.