UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL RESISTENCIA

SISTEMAS OPERATIVOS

SIMULADOR DE ASIGNACIÓN DE MEMORIA Y PLANIFICACIÓN DE PROCESOS

2022

Grupo N° 2

INTEGRANTES

- Ballesteros Franco
- Bolo Luciana Agustina
- Gaczynski Mauro
- Leguizamón Sofía Violeta
- Llopi Ulises Alan

ÍNDICE

INTRODUCCIÓN

Implementar un simulador de asignación de memoria y planificación de procesos según los siguientes requerimientos.
El simulador deberá brindarla posibilidad de cargar N procesos. Para facilitar la implementación se permitirán como máximo 10 procesos y la asignación de memoria se realizará con particiones fijas. El esquema de particiones será el siguiente:
·100K destinados al Sistema Operativo
·250K para trabajos los más grandes.
·120K para trabajos medianos.
·60K para trabajos pequeños.
El programa debe permitir ingreso de nuevos procesos cuando sea posible (manteniendo en grado de multiprogramación en 5) La política de asignación de memoria será Worst-Fit, por cada proceso se debe ingresar o leer desde un archivo el Id de proceso, tamaño del proceso, tiempo de arribo y tiempo de irrupción. La planificación de CPU será dirigida por un algoritmo SJF.

El simulador deberá presentar como salida la siguiente información:

·El estado del procesador (proceso que se encuentra corriendo en ese instante)
·La tabla de particiones de memoria, la cual deberá contener (Id de partición, dirección de comienzo de partición, tamaño de la partición, id de proceso asignado a la partición, fragmentación interna)
·El estado de la cola de procesos listos.
Listado de procesos que no se encuentran en estado de listo ni ejecución (informar el estado en que se encuentran)

IMPLEMENTACION

Ejecución:

Para la ejecución del código no es necesario tener ninguna librería en especial, la versión de Python que se utilizo fue la 3.10.6, se recomienda usar de la versión 3.10 en adelante.

Inicio del programa:

Al comenzar la ejecución deberemos elegir entre dos opciones para la carga de procesos:

La opción 1 carga en la lista de nuevos unos procesos por default, que se muestran al escoger la opción en la cola de nuevos.

La opción 2 carga los procesos de un archivo

Carga de procesos por un archivo (opción 2):

1_ el archivo que se puede cargar debe ser un .txt

2_el archivo .txt tiene que tener el nombre: ArchivoDeProcesos y deberá estar ubicado en la misma carpeta que está ubicado el código .py

3_el archivo que se desee ingresa para cargar procesos en el programa debe tener la siguiente estructura:

a_ una primera línea que se va a saltear el código, esta línea se saltea para que quede como índices de las columnas

b_ posterior a la primera línea, cada una de las siguientes será un proceso

c_ los atributos de cada proceso seguirán un orden el cual se verá en el ejemplo, dato importante, el id del proceso no se deb
era ingresar, ya que se asigna automáticamente.

d_ no tendrá que tener saltos de líneas innecesarios.

ejemplo del archivo de proceso:

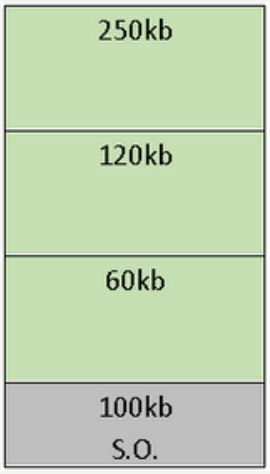
```
tamaño  ta   ti
300      0    3
200      1    4
100      1   10
.
```

SIMULACION

- TABLA DE PROCESOS

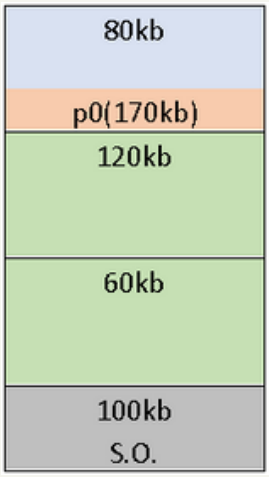
id	tamaño	Ta	ti
0	170	0	2
1	220	0	2
2	140	1	3
3	70	1	2
4	100	1	4
5	30	2	3
6	30	2	5
7	120	3	4
8	60	4	5

t=0, mult=0



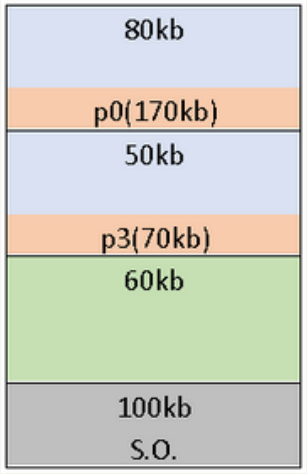
listos=[]
ejecutando=[]
nuevos=
[p0,p1,p2,p3,
p4,p5,p6,p7,p8]
lis/sus=[]
terminados=[]

t=0, mult=2



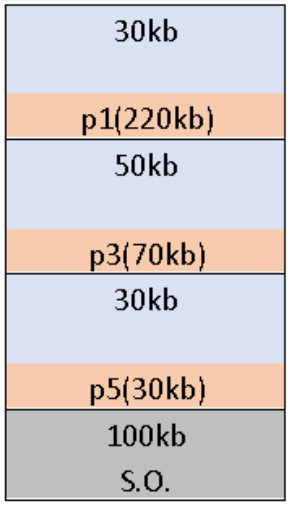
listos=[p0]
ejecutando=[p0]
nuevos=[p2,p3,
p4,p5, p6,p7,p8]
lis/sus=[p1]
terminados=[]

t=1, mult=5



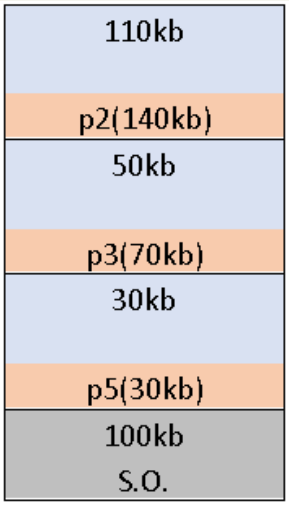
listos=[p0,p3]
ejecutando=[p0]
nuevos[p5,p6,
p7,p8]
lis/sus=[p1,p2,p4]
terminados=[]

t=2, mult=5



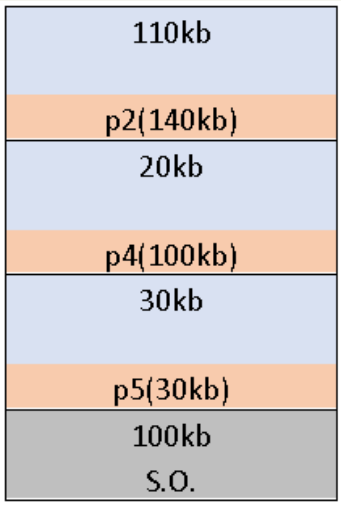
listos=[p3,p1,p5]
ejecutando=[p1]
nuevos=[p6,p7,p8]
lis/sus=[p2,p4]
terminados=[p0,]

t=4, mult=5



listos=[p2,p3,p5]
ejecutando=[p3]
nuevos=[p7,p8]
lis/sus=[p4,p6]
terminados=[p0,p1]

t=6, mult=5



listos=[p2,p4,p5]
ejecutando=[p2]
nuevos=[p8]
lis/sus=[p7,p6]
terminados=[p0, p1, p3]

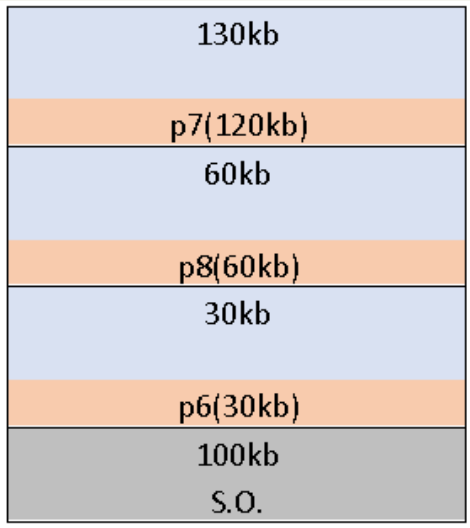
t=9, mult=5

listos=[p7,p4,p5]
ejecutando=[p5]
nuevos=[]
lis/sus=[p6,p8]
terminados=[p0, p1,p3,p2]

t=12, mult=4

listos=[p7,p4,p6]
ejecutando=[p4]
nuevos=[]
lis/sus=[p8]
terminados=[p0, p1,p3,p2,p5]

t=16, mult=3



listos=[p7,p8,p6]
ejecutando=[p7]
nuevos=[]
lis/sus=[]
terminados=[p0, p1,p3,p2,p5,p4]

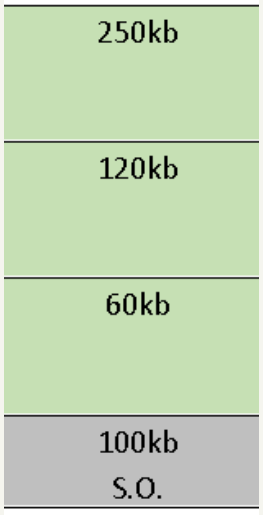
t=20, mult=2

listos=[p8,p6]
ejecutando=[p6]
nuevos=[]
lis/sus=[]
terminados=
[p0,p1,p3,p2,p5,p
4,p7]

t=25, mult=1

listos=[p8]
ejecutando=[p8]
nuevos=[]
lis/sus=[]
terminados=
[p0,p1,p3,p2,p5,p4,
p7,p6]

t=30, mult=0



listos=[]
ejecutando=[]
nuevos=[]
lis/sus=[]
terminados=
[p0,p1,p3,p2,p5,
p4,p7,p6,p8]

Los procesos usados en el ejemplo visto están implementados en los procesos por default, por lo que podemos verificar que el código funciona de la misma manera.

CONCLUSIÓN

Como conclusión nos gustaría agregar que este trabajo nos ayudó a comprender de una manera más tangible el funcionamiento de como la memoria trata procesos de un sistema operativo.

Tuvimos algunos problemas al ir implementando todas las funcionabilidades que necesitaba la memoria para funcionar de manera óptima, como el de tratar archivos externos al código del programa, los demás problemas no se nos presentaron con los requerimientos por separados, sino que fueron en si problemas, al ir implementando los requisitos de manera exponencial. Pese a esto no se nos presentaron inconvenientes con respecto al razonamiento funcional con respecto a la gestión de memoria y planificador de procesos de nuestro simulador de sistema operativo.