Universidad Tecnológica Nacional
Facultad Regional Resistencia
Ingeniería en Sistemas de Información

Sistemas Operativos

Simulador de Asignación de Memoria y Planificación de Procesos

Equipo: Los Sims

Andrés, Aldo Omar.
Bravo Pérez, Agustín Nicolás.
Peralta Ruiz, Nadine.
Trespalacios Martinez, Carlos Daniel.
Vazquez, Máximo Ezequiel.

21 de Noviembre de 2023

Ciclo 2023, Segundo Cuatrimestre

Resistencia, Chaco, Argentina

Índice

Introducción

Desarrollamos este simulador de manera colaborativa y asincrónica, dividiendonos el trabajo en equipo, y cada integrante tuvo un rol a cumplir.

Figura 1: Logo del Grupo “Los Sims”

Aspectos Técnicos

Características del Enunciado

- El planificador a corto plazo usa la estrategia round robin con un quantum igual a 2.
- El esquema de memoria tiene una partición exclusiva para el sistema operativo y otras tres particiones fijas:
  - 250 kB para trabajos grandes,
  - 120 kB para trabajos medianos,
  - y 60 kB para trabajos pequeños.
- La política de asignación a dicha memoria es best-fit.
- El simulador acepta una carga de trabajo con hasta 10 procesos.
- El simulador soporta un grado de multiprogramación igual a 5 como máximo, pero la memoria principal solo permite 3 procesos a la vez, por lo que se necesita usar memoria virtual.

Decisiones de Diseño

- Al momento de activar un proceso suspendido, se lo activa siempre en la partición de mismo tamaño que tenía previamente al ser suspendido.

- Al momento de elegir una partición víctima para suspender, se elije siempre la partición de mismo tamaño que la partición que se está trayendo a memoria principal.
- La planificación round robin tiene en cuenta a los procesos listos y listos/suspendidos por igual. Si hubiera 3 procesos listos y 2 listos/suspendidos, el CPU se estaría compartiendo entre ellos 5, haciendo swap in y out para que los listos/suspendidos estén en memoria principal cuando les llegue su turno.
- Si un proceso desea ser admitido una vez alcanzado el grado de multiprogramación máximo (5), es rechazado pero se lo vuelve a intentar admitir en cada delta de tiempo hasta que otro proceso termine.
- Creamos un estado especial DENEGADO para los procesos que requieren una cantidad de memoria mayor al tamaño de la partición de memoria más grande, por lo que jamás podría ser admitido. Se lo envía al estado DENEGADO para siempre.

En particular el hecho de tener a los procesos listos y listos/suspendidos en la misma cola con la misma prioridad, asegura que un grado de multiprogramación igual a cinco sea posible, pero sacrifica el rendimiento del sistema dado que cuando el grado de multiprogramación es mayor a tres la cantidad de swap outs es muy dañina para la velocidad.

Esto pone en evidencia el compromiso entre la velocidad y la capacidad del sistema, siendo que sostener un grado de multiprogramación mayor a la cantidad de particiones, especialmente en un simulador que no tiene procesos que se bloqueen, se vuelve muy costoso.

Diseño

Al momento de realizar los primeros bocetos de cómo implementaríamos la consigna del simulador, elaboramos el siguiente diagrama de flujo, en el que se puede ver cómo la carga de trabajo es ingresada mediante un archivo de entrada, siendo los procesos ordenados por tiempo de arribo. Luego, a medida que se van admitiendo los procesos a memoria, se planifica su uso del CPU mediante un planificador Round Robin con un quantum igual a 2 hasta que se termine la carga completa de trabajo. Una vez finalizada la simulación, se imprime

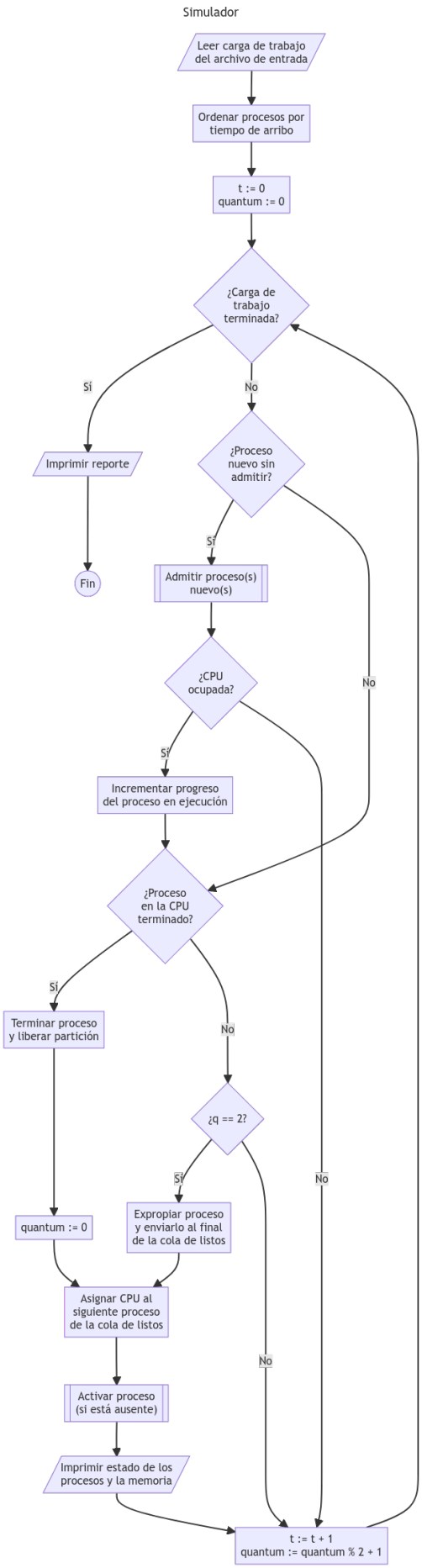

Figura 2: Diagrama del simulador.

Explayamos el proceso de asignar procesos nuevos y activar procesos en otros dos diagramas:

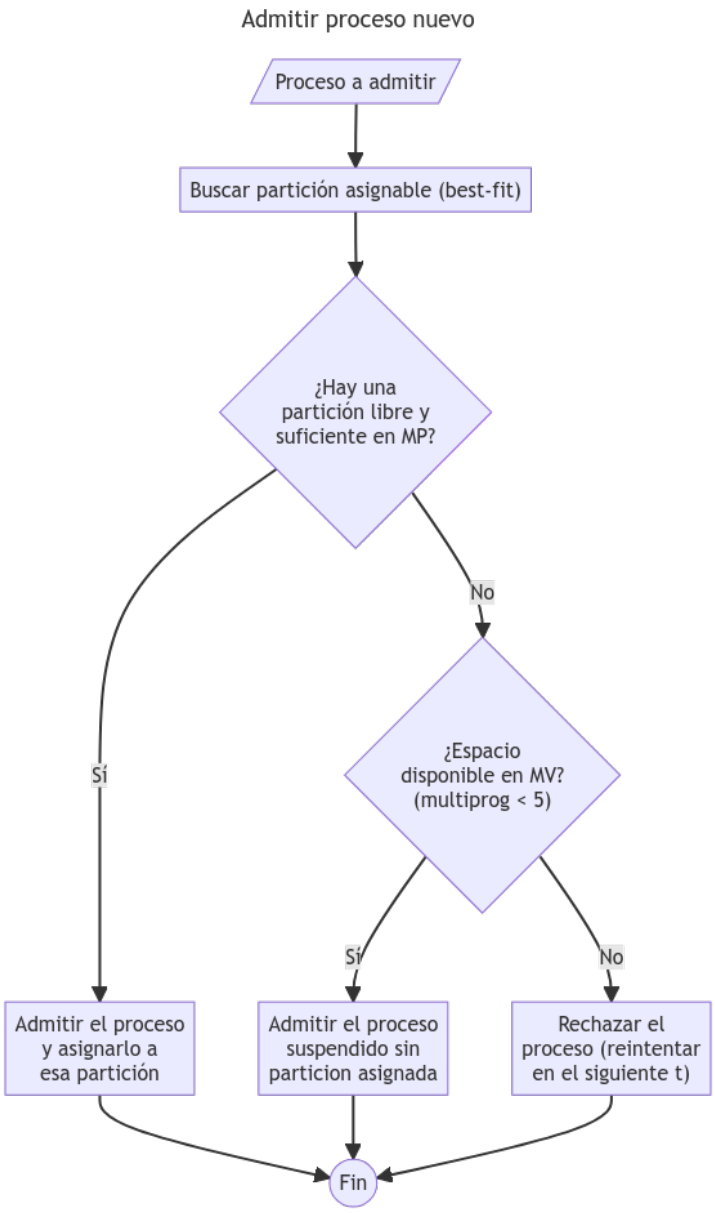

Figura 3: Diagrama de la admisión de un proceso nuevo.

Para poder admitir un proceso, debe haber una partición libre y con suficiente capacidad para alojarlo, caso contrario, si no se tiene un grado de multiprogramación superior a 5 el proceso será suspendido sin partición asignada. Si el grado de multiprogramación es mayor a 5 entonces se rechaza el proceso, y luego se volverá a intentar admitirlo.

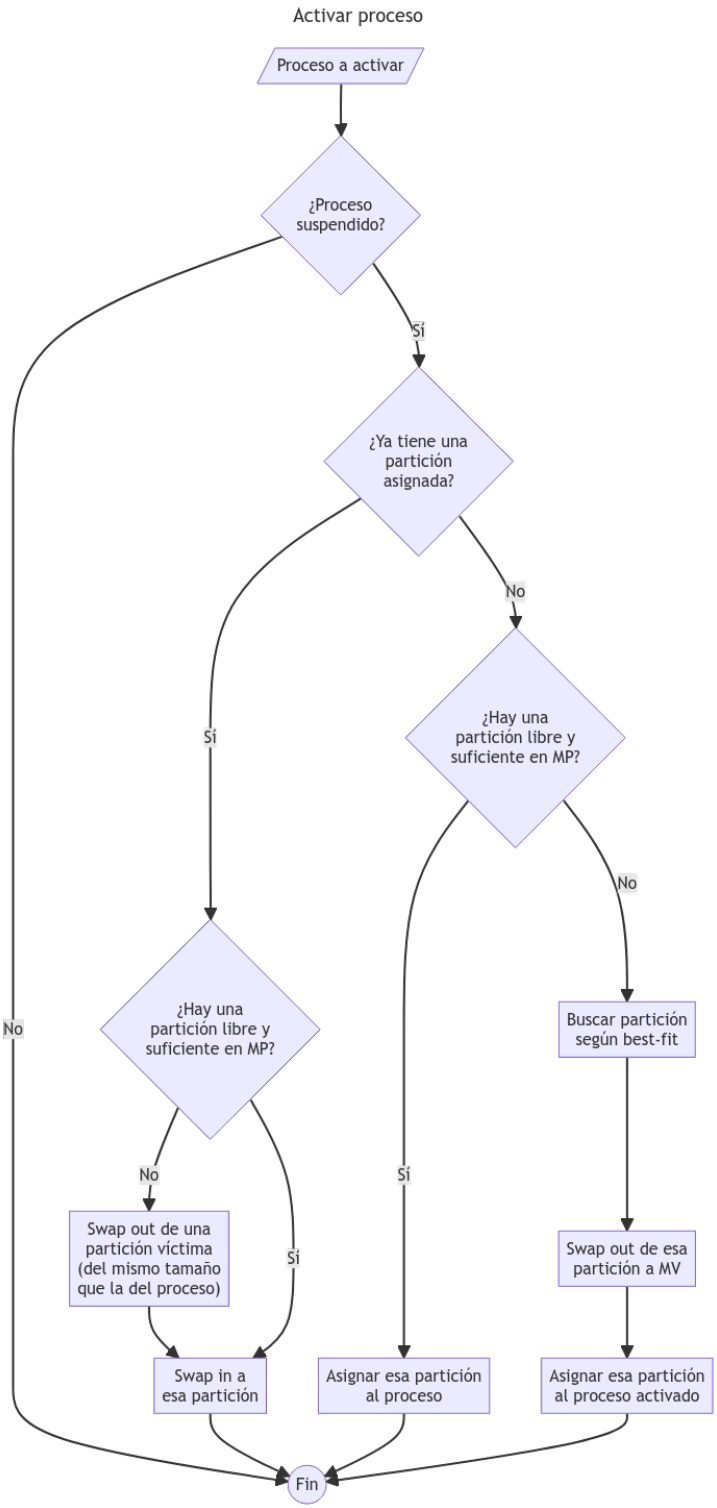

Figura 4: Diagrama de la activación de un proceso.

Un proceso debe ser activado cuando es asignado la CPU. Si está en memoria principal no es necesario hacer nada, pero si este se encuentra en la memoria virtual se debe primero verificar que exista una partición a la que pueda asignarse dicho proceso. De ser así, se hace un swap in a la misma. Sino, se debe hacer un swap out de alguna partición víctima antes de realizar el swap in. Si además ese proceso no tenía una partición asignada, entonces se le asigna la partición encontrad.

Implementación

Organización del Equipo

Utilizamos un tablero kanban de la plataforma de Trello para organizarnos de una mejor manera en la división de responsabilidades y de avances del trabajo. Esto nos permitía ver en todo momento el estado en el que se encontraba el progreso de nuestro trabajo.

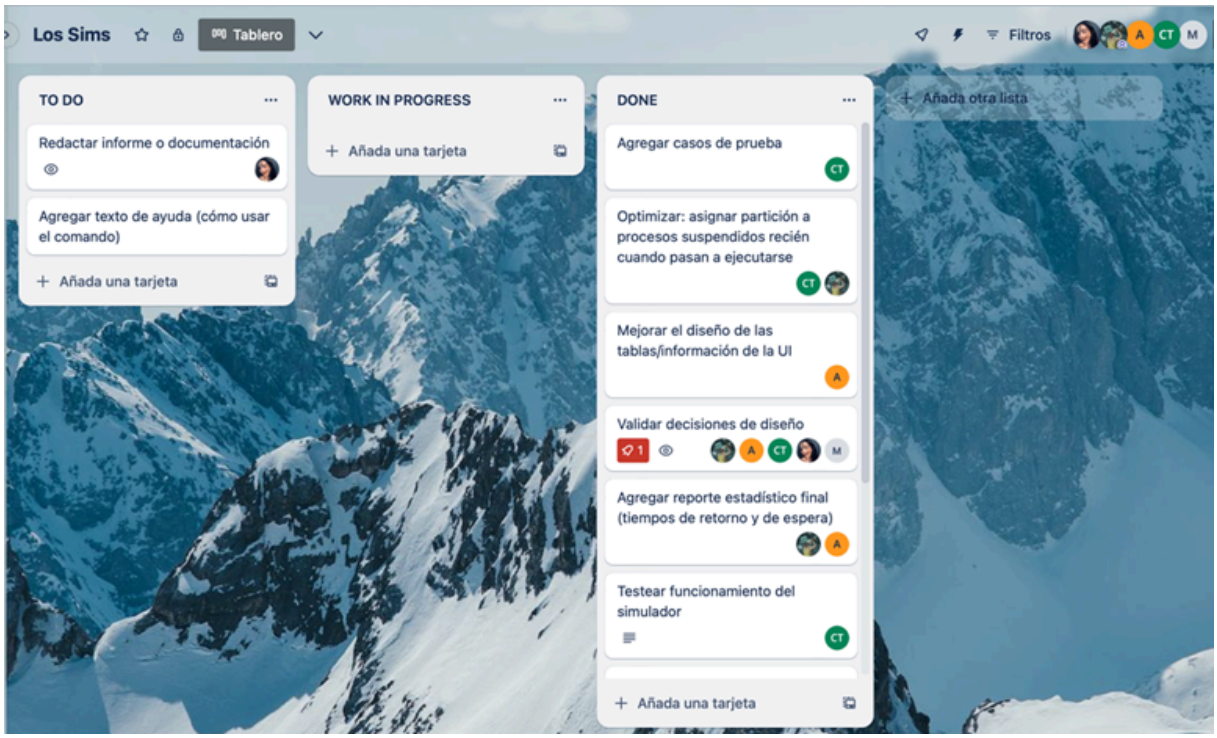

Figura 5: Tablero kanban online en Trello del equipo.

Control de Versiones

Para facilitar el desarrollo colaborativo del trabajo usamos a GitHub como repositorio remoto, para tener un mejor detalle de los cambios que se fueron realizando en cada momento, con acceso a versiones anteriores en caso de hubiésemos cometido algún error.

```
* 02aebd2 - (HEAD -> main, origin/main, origin/HEAD) actualizados diagramas en base a la optimización de procesos suspendidos
Agustin Bravo Pérez>
* f1d657e - workload fuera de carpeta de procesos eliminado (27 hours ago) <Carlos Daniel>
* 764bd5d - Carpeta para la carga de trabajos (27 hours ago) <Carlos Daniel>
* 132691d - Se admiten procesos a suspendidos sin asignarles memoria antes de ser necesario (2 days ago) <Carlos Daniel>
* a95f0cd - corregidas comillas anidadas (5 days ago) <Aldo>
* 7ac7d12 - tabla de memoria ahora muestra valores en bytes (en lugar de bits) (6 days ago) <Agustin Bravo Pérez>
* 4722a76 - agregada barra de progreso a la tabla de la carga de trabajo (6 days ago) <Agustin Bravo Pérez>
* ddfe37d - agregados colores a la información por pantalla (7 days ago) <Agustin Bravo Pérez>
* 103f731 - Decisión de crear el estado `DENEGADO` documentada (8 days ago) <Agustin Bravo Pérez>
*   7cb3dea - Merge pull request #1 from agustinbravop/estadistico (8 days ago) <Agustin Bravo Pérez>
|\
| * fc5e1d9 - (estadistico) mejorada la información y formato del reporte estadístico final (8 days ago) <Agustin Bravo Pérez>
| * 0f5a207 - Calculo estadistico (2 weeks ago) <Aldo>
|/
* 4d4cc0b - agregado logo del equipo al README.md (5 weeks ago) <Agustín Bravo Pérez>
* 9071c5c - aclaración de aspectos técnicos y decisiones de diseño (5 weeks ago) <Agustín Bravo Pérez>
* 40a2df1 - main.py modularizado en un archivo distinto para cada clase (6 weeks ago) <Agustín Bravo Pérez>
*   f408ea8 - Merge branch 'main' of https://github.com/agustinbravop/utn-sims (6 weeks ago) <Agustín Bravo Pérez>
|\
| * 6147614 - Update main.py (6 weeks ago) <peraltanadine>
* | 01829d1 - agregada memoria virtual (y simulador funcionando bien) (6 weeks ago) <Agustín Bravo Pérez>
* | 712deeb - eliminado simulador.py (integrado a main.py) (6 weeks ago) <Agustín Bravo Pérez>
* | 77f8d1f - renombrado simulador2.0 a simulador.py (6 weeks ago) <Agustín Bravo Pérez>
|/
* 72a1114 - implementacion_de_la_carga_de_trabajo (6 weeks ago) <Carlos Daniel>
* b8e118b - implementacion_de_la_carga_de_trabajo (6 weeks ago) <Carlos Daniel>
* 77f75ea - Update simulador2.0 (6 weeks ago) <Maximo-Vazquez>
*   2d3b676 - Merge branch 'main' of https://github.com/agustinbravop/utn-sims (6 weeks ago) <Agustín Bravo Pérez>
|\
| * 1d3ed61 - Create simulador2.0 (7 weeks ago) <Maximo-Vazquez>
* | 5e92487 - diagrama de flujo inicial (6 weeks ago) <Agustín Bravo Pérez>
|/
* 4fcaea9 - leer la carga de trabajo desde un archivo csv (7 weeks ago) <Agustín Bravo Pérez>
* e52b690 - Update README.md (8 weeks ago) <Agustín Bravo Pérez>
* 211d015 - Initial commit (8 weeks ago) <Agustín Bravo Pérez>
```

Figura 6: Historial de commits del repositorio.

Estructura del Código

Modelamos el simulador en cuatro clases de la siguiente manera:

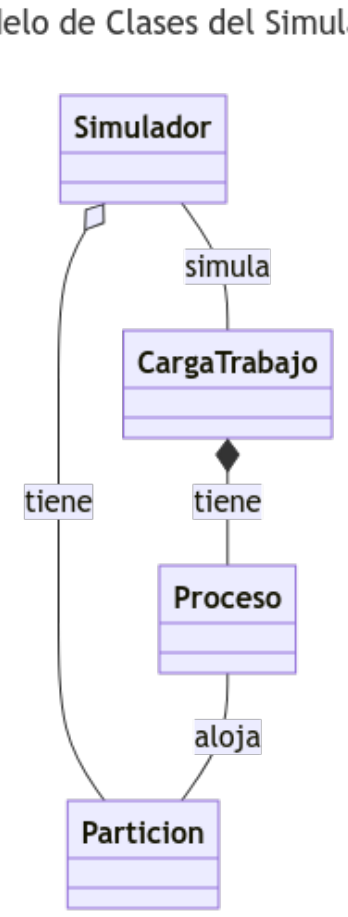

Figura 7: Figura 7. Diagrama de clases.

A grandes rasgos, la `CargaTrabajo` es un conjunto de `Procesos` tomados del archivo de entrada, que serán simulados por la clase `Simulador`, la cual representa su memoria de particiones fijas con un arreglo de `Particiones`, en la cual aloja los procesos para ser ejecutados.

La clase Simulador es la encargada de interconectar el resto de las clases, de implementar la planificación de la CPU y de la administración de la memoria.

Conclusiones

Finalizada la experimentación del desarrollo del simulador, enlazamos los conceptos de teoría en relación a la administración de memoria y planificación de la CPU, con una aplicación tangible de las mismas, gracias a haber diseñado la administración las particiones de la memoria, la asignación de procesos, y el manejo de la cola de planificación.

Una oportunidad de optimización que encontramos fue asignar la partición de memoria a los procesos suspendidos recién cuando vayan a ejecutarse. Originalmente el simulador les asignaba memoria ni bien eran admitidos, lo cual era innecesario. Este cambio reduce la cantidad de memoria virtualizada, pero solo ayuda a disminuir la cantidad de swap outs (y por ende mejorar la velocidad) cuando el haber esperado a asignar la partición efectivamente implica asignar una partición de un tamaño que cumpla el algoritmo best-fit.

Algo que nos benefició mucho al momento de la realización de este trabajo, fue el uso de Trello porque de esta manera pudimos llevar un control real del estado en el que nos encontrábamos. Esto también nos ayudó en el trabajo en equipo, porque de esta manera cada integrante del equipo sabia que tarea debía realizar para poder avanzar de manera organizada. Es por esta razón que seguiremos utilizando esta herramienta para futuros trabajos.