Proyecto de Sistemas Operativos

Objetivo del proyecto

Comprender las características de los distintos algoritmos de paginación y su impacto en el rendimiento de un sistema operativo doméstico.

Descripción del Proyecto

Implemente una simulación que contemple las siguientes características:

- Características de la computadora simulada
    - **Núcleos de procesamiento:** 1
    - **Instrucciones por segundo:** 1
    - **Tiempo de acceso a disco:** 5s
    - **Memoria RAM:** 400KB
    - **Disco Duro:** Indefinido (Asuma que nunca se acabará el espacio en disco duro)
    - **Tamaño de página:** 4KB
- Características de un proceso
    - A cada proceso se le asignará un PID único.
    - Cada proceso será modelado por una secuencia de instrucciones. Todas las instrucciones del proceso se ejecutarán en el MMU quien es responsable de resolverlas según corresponda. Las posibles instrucciones son:
        - **new (pid, size) : ptr** - Solicita nueva memoria de tamaño **size** en B y recibe de vuelta la dirección del puntero lógico (**ptr**) que se le asignó. Deberá guardar el **ptr** en su "tabla de símbolos"
        - **use(ptr) : void** - Utiliza un puntero ya definido en su tabla de símbolos
        - **delete(ptr)** - Elimina un puntero de su tabla de símbolos y libera la memoria asignada
        - **kill(pid)** - Libera toda la memoria asignada a este proceso. Una vez que corre la instrucción kill, no puede correr ninguna instrucción más de este proceso
- Memory Management Unit (MMU)
    - La MMU debe tener una memoria real de 100 páginas (400KB) y una memoria virtual en disco duro de tamaño indefinido.
    - La MMU debe tener un mapa de memoria que relacione cada (**ptr**) con una lista de páginas. Tome en cuenta que si un **ptr** tiene asignada más de 4KB de memoria, la MMU debió asignarle más de una página y no es requerido que las páginas se almacenen de forma secuencial. Por simplicidad de la simulación, no repita los valores de **ptr**, aún si un **ptr** ha sido borrado y estuviera "disponible".
    - Cada página debe tener un ID, una dirección física, una bandera que indique si está en memoria real o virtual, y cualquier otro dato que requiera el algoritmo de paginación respectivo.
    - Si la página está en memoria real la dirección física corresponde al número de segmento en memoria real en el que está asignada.
    - Si la página está en memoria virtual la dirección física no es relevante, puede utilizar lo que requiera para el correcto funcionamiento de su simulación.
    - Cuando un proceso hace la operación **new** la MMU debe crear las páginas respectivas en memoria real y retornarle al proceso el **ptr** resultante.

    - Cuando un proceso hace la operación **use** la MMU debe garantizar que las páginas que corresponden al **ptr** estén en memoria real.
    - Cuando un proceso hace la operación **delete** la MMU debe borrar todas las páginas asignadas a ese **ptr** y eliminar por completo el **ptr** de su mapa de memoria.
    - Cuando un proceso hace la operación **kill**, la MMU debe borrar todas páginas y **ptr** que le pertenecen a ese PID.
- Preparación
    - Al iniciar el programa, deberá mostrar una interfaz en la que el usuario introduzca los siguientes valores:
        - Semilla para random (de forma que se pueda repetir un escenario)
        - Algoritmo a simular (FIFO, SC, MRU, RND)
        - Archivo para simular (archivo con la secuencia de instrucciones de toda la sesión)
        - Número de procesos a simular P (10, 50, 100) (si el usuario no provee un archivo, el sistema deberá generar una cantidad P de procesos)
        - Cantidad de operaciones N (500, 1000, 5000) (si el usuario no provee un archivo, el sistema deberá generar una lista de operaciones para toda la sesión manteniendo las siguientes reglas)
            - Cada proceso puede tener una cantidad cualquiera de operaciones, siempre que la suma total de operaciones no supere N
            - La operación **new** se puede usar en cualquier momento, siempre que no sea después de una operación **kill**
            - La operación **use** sólo se puede usar si existe al menos un **ptr** en la tabla de símbolos
            - La operación **delete** sólo se puede usar si existe al menos un **ptr** en la tabla de símbolos
            - La operación **kill** se debe usar únicamente una vez, y debe ser la última operación del proceso
            - Pueden intercalarse libremente operaciones de distintos procesos, por ejemplo esta es una secuencia válida de operaciones:
                - new(1, 250) //*suponga que devuelve el ptr 1*
                - new(1, 50) //*suponga que devuelve el ptr 2*
                - new(2, 5320) //*suponga que devuelve el ptr 3*
                - new(3, 345) //*suponga que devuelve el ptr 4*
                - use(1)
                - use(3)
                - use(2)
                - use(1)
                - delete(1) //*a partir de este punto ya no se puede volver a usar el ptr 1*
                - kill(1) // *ya no se pueden volver a usar los punteros 1 y 2, a partir de este punto no puede hacerse new(1,x)*
                - kill(2)
                - kill(3)
            - Genere y permita descargar el archivo de operaciones antes de iniciar la simulación. El formato del archivo debe ser compatible con el que se muestra en el anexo 1.
- Simulación
    - Una vez calculada o importada la secuencia de instrucciones y seleccionado el algoritmo a simular (FIFO, SC, MRU, RND), su sistema deberá pasar a una nueva ventana en la cual se visualice la simulación.

- Su sistema deberá simular dos algoritmos en simultáneo, el OPT y el algoritmo seleccionado por el usuario.
- Deberá correr ambas simulaciones de forma coordinada, es decir, cuando el OPT completa una operación, deberá completar una operación del otro algoritmo. Ambos algoritmos deberán terminar al mismo tiempo. Esto permitirá visualizar el estado de la memoria en ambos algoritmos.
- La simulación deberá llevar un "reloj". Cada operación que resulte en un hit de página (la página solicitada estaba en memoria real) suma 1s al reloj. Cada operación que resulte en un fallo de página (la página solicitada estaba en memoria virtual) suma 5s al reloj, este tiempo además deberá sumarse al tiempo de Thrashing.
- En el caso de páginas nuevas, considere un hit si existía un segmento disponible para la página en memoria real, y un fallo si fue necesario enviar una página a memoria virtual para hacer el espacio respectivo.
- Si la operación requiere acceder a más de una página, sume al reloj el tiempo respectivo por cada página involucrada. Es decir si requiere traer 3 páginas y las tres son hits, sume 3s al reloj. Si 2 son hit y una es fallo sume 7s al reloj.
- Visualize los siguientes elementos en simultáneo para el algoritmo óptimo y el seleccionado por el usuario:
  - El estado de la RAM
  - El estado de la MMU
  - Listar la cantidad de procesos corriendo
  - Mostrar el tiempo total de simulación hasta el momento
  - Mostrar la cantidad de RAM utilizada en KB y en %
  - Mostrar la cantidad de V-RAM utilizada en KB y en %RAM (por ejemplo, dado que hay 400KB de RAM, si hay 500 KB en V-RAM deberá mostrar 125%)
  - Mostrar la cantidad de tiempo que se ha gastado en fallos de páginación (Thrasing) y el porcentaje respecto al tiempo total de simulación. Si el Thrasing supera el 50% deberá marcarlo en rojo.
  - Mostrar la cantidad de memoria que se está desperdiciada por fragmentación interna. Considere sólo las páginas que están actualmente en el sistema, si una página se borra deberá remover esa cantidad del total de memoria desperdiciado.
  - Permita pausar y reanudar la simulación en cualquier momento.

Forma de Entrega

Podrá programar su proyecto en cualquier plataforma o lenguaje siempre y cuando corra en un entorno Linux, puede ser un sistema web si gusta. Considere que se revisará en un Fedora. Es indispensable que su proyecto contemple una visualización de la simulación, similar a la que se ilustra en el Anexo 2.

Deberá entregar el código fuente de su proyecto o un enlace con la simulación funcionando. Si su código fuente no compila o no corre, su asignación será anulada automáticamente. Deberá acompañar también su entrega con un informe que tenga las siguientes secciones:

- Portada
- Introducción (10%)
  - Sobre la gestión de memoria en los SO
  - Sobre qué es paginación y memoria virtual
- Desarrollo (60%, 15% cada uno)
  - Descripción del algoritmo óptimo y descripción de la implementación en el proyecto, ilustre con capturas de pantalla.
  - Descripción de los algoritmos FIFO y Second Chance, y descripción de la implementación en el proyecto, ilustre con capturas de pantalla.

        - Descripción de los algoritmos LRU y MRU, y descripción de la implementación en el proyecto, ilustre con capturas de pantalla.
        - Descripción del algoritmo Random, y descripción de la implementación en el proyecto, ilustre con capturas de pantalla.
    - Enlace al código e instrucciones de compilación y ejecución
    - Conclusiones (30%)
        - Al menos 3 conclusiones concretas
    - Referencias bibliográficas en el formato de la plantilla provista
        - Si omite alguna cita o referencia bibliográfica, o si comete errores en el formato de las referencias, su entrega será anulada automáticamente.

Respecto a su sistema, se evaluará de la siguiente manera:

- Implementación del Algoritmo Óptimo (10%)
    - Debe ser posible visualizar el algoritmo para evaluarlo
- Implementación de los Algoritmos (60%, 15% cada uno)
    - Debe ser posible visualizar los algoritmos para evaluarlos
- Diseño de la interfaz y visualización (20%)
    - Que la interfaz represente toda la información solicitada
    - Que la interfaz se pueda operar de forma intuitiva
- Repetibilidad del escenario (10%)
    - Que sea posible repetir las condiciones de una simulación utilizando la misma semilla

Se diseñará un archivo de pruebas con un listado de procesos que sigue el formato requerido en este documento. Si su simulación no corre con el archivo de pruebas no se evaluará. Note que el formato es de un archivo separado por comas, tal como se ilustra en el anexo 1.

Puede desarrollar este proyecto en grupo de máximo 4 personas.

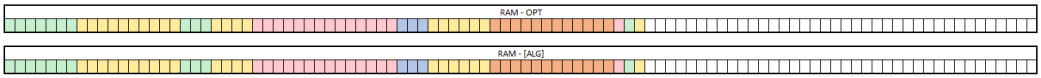
Anexos

1. Formato del archivo de operaciones:

Este ejemplo utiliza dos procesos, note que el comando use asume que el número de puntero se asigna por el orden de aparición. Por ejemplo la línea 1 produce el puntero 1, la línea 3 produce el puntero 2, la línea 6 produce el puntero 3, y así consecutivamente.

```
new(1,500)
use(1)
new(1,1000)
use(1)
use(2)
new(2,500)
use(3)
use(1)
new(2,50)
use(4)
delete(1)
use(2)
use(3)
delete(2)
kill(1)
kill(2)
```

2. Ejemplo de la Visualización



MMU - OPT

PAGE ID	PID	LOADED	L-ADDR	M-ADDR	D-ADDR	LOADED-T	MARK
1	1	X	1	1		0s	
2	1	X	2	2		1s	
3	1		3		143		
4	1	X	4	3		5s	
5	1		5		100		
6	2	X	6	8		3s	
7	2	X	7	9		4s	
8	2	X	8	10		7s	
9	2	X	9	11		12s	
10	2	X	10	12		13s	
11	2		11		231		
12	2	X	12	21		14s	
13	2		13		278		
14	2		14		245		
15	2	X	15	22		151s	
16	3	X	16	25		152s	
17	3	X	17	26		153s	
18	3		18		97		
19	3		19		12		
...	...	...	...	...	...	...	...

Processes	Sim-Time
5	250s

RAM KB	RAM %	V-RAM KB	V-RAM %
244	61	40	4%

PAGES		Thrashing		Fragmentación
LOADED	UNLOADED	150s	60%	256KB
61	85			

MMU - [ALG]

PAGE ID	PID	LOADED	L-ADDR	M-ADDR	D-ADDR	LOADED-T	MARK
1	1	X	1	1		0s	
2	1	X	2	2		1s	
3	1		3		143		
4	1	X	4	3		5s	
5	1		5		100		
6	2	X	6	8		3s	
7	2	X	7	9		4s	
8	2	X	8	10		7s	
9	2	X	9	11		12s	
10	2	X	10	12		13s	
11	2		11		231		
12	2	X	12	21		14s	
13	2		13		278		
14	2		14		245		
15	2	X	15	22		151s	
16	3	X	16	25		152s	
17	3	X	17	26		153s	
18	3		18		97		
19	3		19		12		
...	...	...	...	...	...	...	...

Processes	Sim-Time
5	250s

RAM KB	RAM %	V-RAM KB	V-RAM %
244	61	40	4%

PAGES		Thrashing		Fragmentación
LOADED	UNLOADED	150s	60%	256KB
61	85			