Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
ISIS 1311 – Tecnología e Infraestructura de Cómputo

Caso 2. Memoria Virtual

1. Objetivo

- Entender la importancia de contar con una administración "apropiada" de la memoria: considerando infraestructura de soporte, número de procesos en ejecución, demanda del recurso por proceso y decisiones para adicionar y/o remover marcos de página asignados.
- Implementar un simulador en Java que imite el comportamiento de un sistema de memoria virtual que usa paginación por demanda.

MUY IMPORTANTE: El entregable de este caso debe ser 100% de su autoría. No está permitido usar ayudas no autorizadas (incluyendo chatbots o tecnologías similares). El incumplimiento de lo anterior resultará en una calificación de cero (0.0) para el caso 2.

Además, tenga en cuenta que los casos están diseñados para apoyar su proceso de aprendizaje e identificar y resolver sus dudas sobre los temas estudiados antes del parcial.

2. Descripción del problema

En un sistema con memoria virtual, los procesos generan accesos a direcciones virtuales (dv) que deben ser traducidas a direcciones físicas. Este proceso es manejado automáticamente por el sistema operativo mediante elementos como:

- **Tablas de páginas** por proceso, que mapean páginas virtuales a marcos físicos.
- **Memoria física**, que tiene una capacidad limitada de marcos.
- **Política de reemplazo**, que permite seleccionar qué página sacar de la memoria física en caso de que esté llena y se necesite espacio.

Su tarea en este caso es desarrollar un simulador, **en Java**, que modele el comportamiento de estos componentes durante la ejecución de un número definido de procesos. No use librerías externas, solamente librerías Java estándar.

3. El simulador

El simulador debe tener dos opciones de ejecución. A continuación, se describe cada opción:

Opción 1: Generación de las referencias.

Para simplificar el problema del manejo de memoria, y concentrarnos en entender los retos asociados, solamente estudiaremos las direcciones virtuales que serían generadas por procesos que suman matrices.

La siguiente figura presenta el código para sumar matrices.

```java
public void sumarMatrices(int pnf, int pnc) {
    int filas = pnf;
    int columnas = pnc;

    for (int i=0; i<filas;i++) {
        for (int j=0; j<columnas; j++) {
            matriz3[i][j]=matriz1[i][j] + matriz2[i][j];
        }
    }
}
```

Tenga en cuenta que cada proceso simulará la suma de dos matrices del mismo tamaño, pero los tamaños de matriz manejados por diferentes procesos sí pueden ser diferentes. Por ejemplo, si simulamos dos procesos: el primer proceso puede ejecutar `sumaMatrices` con matrices de 10x10, mientras el segundo proceso puede ejecutar `sumaMatrices` con matrices de 20x20.

La opción 1 recibe un archivo de configuración con los siguientes parámetros: TP: tamaño de página, NPROC: número de procesos y TAMS: tamaños de las matrices. Y genera n archivos de salida, donde n corresponde al número de procesos. Cada archivo_i de salida tiene las direcciones dv que el proceso_i generaría al ejecutar el método `sumaMatrices`.

Supondremos que las matrices se almacenan por filas (esto se conoce como row-major order) y en orden: primero la matriz1, luego la matriz2 y finalmente la matriz3. Tenga en cuenta que, dependiendo del tamaño de las matrices y las páginas, podemos tener varias matrices en una misma página o parte de una matriz en una página y parte en otra. Tomaremos 4 bytes como el tamaño de un entero.

Cada archivo generado debe incluir: TP (tamaño de página), NF (num filas), NC (num columnas), NR: #referencias y NP: #páginas virtuales del proceso. Después de estos datos, debe estar la lista de direcciones dv generadas por el proceso. En el anexo encontrará un ejemplo ilustrativo.

El nombre de cada archivo de salida debe seguir el patrón proc<i>.txt, donde <i> corresponde al número del proceso.

Opción 2: Simulación de la ejecución.

Esta opción recibe como parámetros:

- El número de procesos. Observe que este número debe ser consistente con el número de archivos de salida generados por la opción 1.
- El número de marcos totales en la memoria RAM del sistema que simulamos. Para simplificar el problema, siempre usaremos un número de marcos que sea múltiplo del número de procesos.

Esta opción simula la ejecución de los procesos, para esto debe:

- Inicio:
  - Crear los procesos de acuerdo con el parámetro número de procesos.
  - Cargar las direcciones dv de cada proceso (a partir de los archivos generados en la opción anterior).
  - Asignar los marcos.

- Simulación:
    - Procesar cada dv de cada proceso, turnando los procesos. En cada turno, un proceso solo puede procesar una dv.
    - Tenga en cuenta que, si un proceso tiene menos direcciones dv que otro entonces su procesamiento terminará primero.

Al terminar la simulación de todos los procesos que recibe como parámetro, la opción 2 presenta las siguientes estadísticas por proceso:

- Total de referencias generadas por el proceso.
- Total de fallos de página.
- Total de accesos a SWAP.
- Tasa de fallos (fallos / referencias).
- Tasa de éxito (hits / referencias).

Tenga en cuenta que la opción 2 debe:

A. Manejar múltiples procesos.

- Cada proceso debe estar representado por una instancia de una clase Proceso y tener su propia **tabla de páginas**. La tabla debe indicar si una determinada página virtual del proceso está cargada en memoria y en qué marco físico.
- Cada proceso tiene una **lista de direcciones dv** que intentará acceder secuencialmente. La lista de direcciones de cada proceso estará definida por el contenido del archivo correspondiente generado por la opción 1.

Para simular la ejecución de múltiples procesos, el simulador debe usar una política de turnos. Por ejemplo, si simulamos 2 procesos, el simulador deberá simular el procesamiento de las direcciones dv así: proc0-dv0, proc1-dv0, proc0-dv1, proc1-dv1, etc. repitiendo sucesivamente hasta terminar.

Para esto necesitará:

- Distribuir de forma equitativa el número de marcos entre los procesos simulados. Para simplificar el manejo, siempre correremos la opción 2 con un número de marcos que sea múltiplo del número de procesos y al empezar se asignará el mismo número de marcos a cada proceso. Tenga en cuenta que tomamos esta decisión para simplificar el problema, pero en la práctica el sistema operativo calcula de forma dinámica el número de marcos que debe asignar.
- Reasignar los marcos de un proceso que termina. La política de reasignación se basará en el número de fallos generados por cada proceso en el momento de la reasignación. Es decir, si un proceso termina y tenía 2 marcos asignados, el simulador buscará el proceso en ejecución con mayor número de fallo de páginas y le asignará los 2 marcos. Si el proceso que termina es el último en ejecución entonces no es necesario reasignar los marcos.
- Utilizar una cola de procesos para simular que los procesos se van turnando para acceder a la memoria. Si después de procesar una dv un proceso aún tiene direcciones por acceder, se vuelve a insertar al final de la cola de procesos; si no, se retira.
- Si un proceso genera una falla de página el simulador lo demorará un turno. Es decir, resolverá la falla, pero no marcará la dv como procesada, en vez de eso vuelve a insertar el proceso al

final de la cola de procesos y la misma dv será procesada en el siguiente turno, en el cual ya no habrá falla de página.

B. Convertir direcciones dv a páginas en memoria RAM.

- Tenga en cuenta que el tamaño de página está en el archivo de entrada.
- Para este caso las páginas y los marcos tienen el mismo tamaño, y todos los procesos tienen el mismo tamaño de página.
- Necesitará considerar la tabla de páginas para traducir una dv a la correspondiente dirección real en RAM.
- Implemente LRU (Least Recently Used) como política de reemplazo. Tanebaum explica este algoritmo en su libro Sistemas Operativos Modernos, capítulo 3 – sección “Simulación de LRU en software” (disponible en versión electrónica en la biblioteca).
- Para manejar el envejecimiento asociado con el algoritmo LRU use el tipo de dato *long.*

C. Para calcular las estadísticas.

- Cada acceso a memoria puede resultar en:
  - Un acierto (la página ya está cargada).
  - Un fallo de página sin operación de reemplazo (la página no está cargada y hay que resolver el fallo). Esto cuenta como un acceso a SWAP.
  - Un fallo de página con operación de reemplazo (si hay un fallo de página y todos los marcos asignados ya están llenos será necesario reemplazar un marco). Esto cuenta como dos accesos a SWAP.

D. Informe

Escriba un informe que incluya:

- Nombre completo y código de los dos integrantes del grupo.
- Descripción del algoritmo usado para generar las referencias de página (opción uno).
- Descripción de las estructuras de datos usadas para simular el comportamiento del sistema de paginación y cómo usa dichas estructuras (cuándo se actualizan, con base en qué y en qué consiste la actualización).
- Corra su programa con matrices de 100x100 variando el tamaño de página. Cree una gráfica que ilustre el comportamiento del número de fallas de página con respecto al tamaño de página.
- Además de los escenarios definidos, considere y documente en su informe otras 4 configuraciones. Por ejemplo, variando el tamaño de las matrices y el número de marcos, para entender cómo afecta la memoria virtual el desempeño del programa.
- Escriba su interpretación de los resultados: ¿corresponden a los resultados que esperaba? Explique su respuesta.
- ¿Cómo se afecta el tiempo de respuesta de un proceso por el número de accesos a SWAP?
- ¿Sumar matrices representa un problema de localidad alta, media o baja? Explique su respuesta en un párrafo de máximo 10 líneas.

4. Condiciones de Entrega

- En un archivo .zip entregue el código fuente del programa y un informe con el nombre de los integrantes. El nombre del archivo debe ser: caso2_login1_login2.zip
- El trabajo se debe realizar en grupos de 2 estudiantes. No debe haber consultas entre grupos. Recuerde que debe conformar un grupo diferente al del caso 1 y debe trabajar en grupo.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- En el parcial se incluirá una pregunta sobre el caso.
- El proyecto debe ser entregado por Bloque Neón por uno solo de los integrantes del grupo. **Al comienzo del documento PDF, deben estar los nombres y códigos de estudiante de los integrantes del grupo.** Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente, sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- Tenga en cuenta las reglas establecidas en el programa del curso sobre la obligatoriedad del trabajo en grupo. Si no cumple con las reglas habrá una penalización de 0,5/5.
- **Se debe entregar por Bloque Neón a más tardar el 23 de septiembre a las 11:59 p.m.**
- **La fecha de entrega no será movida.**
- **Política de entrega tarde.** Para las entregas tarde, se aplicará la siguiente política: por cada 30 minutos o fracción de retraso, con respecto a la hora de entrega establecida en Bloque Neón, habrá una penalización de 0,5/5.
- En Bloque Neón, **es necesario inscribirse** en alguno de los grupos disponibles para poder acceder a la actividad de entrega del caso. Para ello, debe ingresar a la sección Grupos → Ver grupos disponibles → Caso 2. Es muy importante asegurarse de que ambos integrantes estén registrados en el mismo grupo; de lo contrario, quien no se inscriba no podrá recibir la calificación.