



Tarea 2: Sistemas Operativos

Barreras y Memoria Virtual

Integrantes:

Eduardo Alfonso Mora Hernández
Lucas Daniel Morales Oyanedel
Vicente Ignacio Miranda Gómez
Ignacio Jesús Soto Miranda

Fecha: December 2, 2025



Contents

1	Introducción	2
2	Descripción de las Implementaciones	2
2.1	Barrera Reutilizable	2
2.2	Simulador de Memoria Virtual	3
3	Resultados del Simulador	4
3.1	Diferencias entre <code>trace1.txt</code> y <code>trace2.txt</code>	4
3.2	Explicación de columnas	5
3.3	Resultados: <code>trace1.txt</code>	5
3.4	Resultados: <code>trace2.txt</code>	6
4	Análisis de Resultados	7
4.1	Trace1	7
4.2	Trace2	7
4.3	Conclusiones del análisis	7
5	Conclusiones	8



1 Introducción

Este informe presenta la implementación, evaluación y análisis de dos componentes esenciales del estudio de sistemas operativos: una **barrera reutilizable para sincronización de hebras** y un **simulador de memoria virtual** basado en el algoritmo de reemplazo **Reloj**. Ambos fueron desarrollados en lenguaje C, utilizando la biblioteca `pthread` para la sincronización y estructuras dinámicas para la simulación de memoria virtual.

El documento combina explicación conceptual, detalles de diseño y análisis experimental basado en dos trazas de ejecución diferentes.

2 Descripción de las Implementaciones

2.1 Barrera Reutilizable

La barrera implementada permite que N hebras sincronizadas avancen en etapas sucesivas. Ninguna hebra puede iniciar la etapa siguiente hasta que todas hayan alcanzado el punto de sincronización.

El mecanismo implementado se basa en el patrón de diseño *monitor*, utilizando un mutex y una variable de condición.

Estructura de la barrera

La estructura principal almacena:

- **Mutex**: protege todas las variables compartidas.
- **Variable de condición**: bloquea hebras hasta completar la etapa.
- **Count**: número de hebras que han llegado a la barrera.
- **N**: total de hebras participantes.
- **Etapa**: contador que evita adelantamientos indebidos.

Funcionamiento de la operación de espera

El comportamiento del método es:

1. Las hebras adquieren el mutex.
2. Incrementan el contador de llegadas.
3. Si no son la última hebra, quedan bloqueadas mientras la etapa no cambie.
4. La última hebra en llegar:
 - reinicia el contador,
 - incrementa la etapa,
 - emite un **broadcast** para liberar a las demás.



Verificación del funcionamiento

Para comprobar la barrera se crean N hebras que avanzan en E etapas, introduciendo retardos aleatorios. Las salidas muestran claramente que:

- ninguna hebra avanza prematuramente,
- todas entran y salen de la barrera de manera sincronizada,
- la barrera es efectivamente reutilizable.

2.2 Simulador de Memoria Virtual

El simulador procesa una secuencia de direcciones virtuales, determina si cada acceso produce un **HIT** o un **FALLO**, y aplica el algoritmo de reemplazo Reloj. El programa permite variar tanto el **tamaño de página** como el **número de marcos físicos**.

Componentes internos del simulador

- **Tabla de páginas**: indica qué marco ocupa cada página.
- **Arreglo de marcos**: registra qué página reside en cada marco.
- **Bits de referencia**: usados para otorgar “segunda oportunidad”.
- **Puntero del reloj**: recorre los marcos cíclicamente.

Algoritmo de reemplazo Reloj

Cuando ocurre un fallo:

1. Si hay marcos libres, se asignan directamente.
2. Si no, se recorre secuencialmente:
 - Si el bit es 1, se limpia y avanza.
 - Si es 0, se reemplaza ese marco.

Este algoritmo es una variante eficiente de LRU.



3 Resultados del Simulador

Antes de presentar los resultados tabulados, es fundamental explicar las diferencias entre los archivos de traza usados y las columnas de las tablas.

3.1 Diferencias entre trace1.txt y trace2.txt

Los dos archivos de traza difieren sustancialmente en el patrón de acceso a memoria.

Resumen conceptual

- **trace1.txt:** alta localidad espacial; los accesos tienden a agruparse.
- **trace2.txt:** accesos dispersos y menos predecibles; baja localidad.

Comparación resumida

Característica	trace1.txt	trace2.txt
Localidad espacial	Alta	Baja
Localidad temporal	Media-alta	Baja
Sensibilidad al tamaño de página	Muy alta	Moderada
Sensibilidad al número de marcos	Baja	Alta
Efecto típico	Casi sin fallos con páginas grandes	Fallos persistentes



3.2 Explicación de columnas

- **Tamaño de página:** tamaño de cada página virtual en bytes.
- **Marcos:** memoria física disponible.
- **Referencias:** accesos totales procesados.
- **Fallos:** número de faltas de página.
- **Tasa:** porcentaje de fallos respecto al total.

3.3 Resultados: trace1.txt

Tamaño de página	Marcos	Referencias	Fallos	Tasa
8	4	8192	8129	99.23%
8	8	8192	8073	98.55%
8	16	8192	7943	96.96%
8	32	8192	7713	94.15%
2048	4	8192	2	0.02%
2048	8	8192	2	0.02%
2048	16	8192	2	0.02%
2048	32	8192	2	0.02%
4096	4	8192	1	0.01%
4096	8	8192	1	0.01%
4096	16	8192	1	0.01%
4096	32	8192	1	0.01%
8192	4	8192	1	0.01%
8192	8	8192	1	0.01%
8192	16	8192	1	0.01%
8192	32	8192	1	0.01%

Table 1: Resultados del simulador con `trace1.txt`.



3.4 Resultados: trace2.txt

Tamaño de página	Marcos	Referencias	Fallos	Tasa
8	4	8192	8192	100.00%
8	8	8192	8191	99.99%
8	16	8192	8190	99.98%
8	32	8192	8189	99.96%
2048	4	8192	8053	98.30%
2048	8	8192	7925	96.74%
2048	16	8192	7665	93.57%
2048	32	8192	7145	87.22%
4096	4	8192	7917	96.64%
4096	8	8192	7649	93.37%
4096	16	8192	7138	87.13%
4096	32	8192	6142	74.98%
8192	4	8192	7675	93.69%
8192	8	8192	7150	87.28%
8192	16	8192	6155	75.13%
8192	32	8192	4147	50.62%

Table 2: Resultados del simulador con `trace2.txt`.



4 Análisis de Resultados

4.1 Trace1

`trace1.txt` muestra alta localidad espacial:

- tasas altísimas de fallos con páginas pequeñas,
- tasas casi nulas con páginas grandes,
- poca dependencia del número de marcos.

4.2 Trace2

`trace2.txt` produce presión persistente sobre el sistema:

- con páginas pequeñas casi todos los accesos fallan,
- incluso con páginas grandes, los fallos siguen siendo numerosos,
- los 32 marcos reducen la tasa, pero no eliminan los fallos.

4.3 Conclusiones del análisis

- El tamaño de página impacta más profundamente que el número de marcos.
- `trace1.txt` evidencia fuerte localidad espacial.
- `trace2.txt` es más errático y adverso para la memoria.
- El algoritmo Reloj se comporta como un reemplazo eficiente y estable.



5 Conclusiones

La implementación de la barrera reutilizable permitió validar conceptos fundamentales de sincronización, demostrando un funcionamiento correcto, robusto y libre de condiciones de carrera. Por otro lado, el simulador de memoria virtual evidenció cómo el comportamiento de las trazas y el tamaño de página afectan drásticamente el rendimiento de la memoria bajo demanda.

Los resultados demuestran la relevancia de los patrones de acceso en el rendimiento del sistema y permiten visualizar claramente el aporte práctico del algoritmo de reemplazo Reloj.