UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

501251-1 SISTEMAS OPERATIVOS

Tarea 2

Integrantes:

- Felipe Adolfo Mendoza Ojeda

Introducción:

Esta tarea tiene dos partes, una relacionada con las primitivas de sincronización y otra con el manejo de memoria virtual. Cada parte tiene sus propios objetivos, descripción y actividades a realizar.

Una **barrera reutilizable** implementada como monitor, para sincronizar múltiples hebras en distintos puntos de encuentro.

Un **simulador secuencial de memoria virtual**, con traducción de direcciones y algoritmo de reemplazo de páginas **Reloj**, capaz de medir la tasa de fallos de página bajo distintas configuraciones de memoria.

Parte I. Sincronización con Barrera reutilizable

En la Parte I se implementó una barrera reutilizable como un monitor en C, definida en barrier_t con las variables count, N, etapa y las primitivas pthread_mutex_t y pthread_cond_t.

En barrier_init deja la barrera lista inicializando estos campos y las primitivas de sincronización; barrier_wait realiza la sincronización capturando la etapa actual, aumentando el contador y, si la hebra es la última, avanzando la etapa, reseteando el contador y haciendo pthread_cond_broadcast, mientras que las demás hebras esperan en un while con pthread_cond_wait hasta que la etapa cambie por último barrier_destroy libera los recursos destruyendo el mutex y la variable de condición, permitiendo reutilizar la misma barrera en múltiples etapas.

En el programa de verificación se usan dos parámetros: N hebras y E etapas. Cada hebra ejecuta el mismo ciclo imprime un mensaje “esperando etapa e”, llama a barrier_wait para sincronizarse con las demás hebras y, una vez que la barrera la libera, imprime “paso de barrera etapa e”.

Parte II – Simulador de Memoria Virtual

En la Parte II se usa una tabla de páginas implementada como un arreglo de entradas, donde cada entrada guarda al menos si la página es válida (valid), el número de marco físico asociado (frame), el número de página virtual (vpn) y un bit de uso (use_bit) que emplea el algoritmo Reloj para decidir reemplazos.

Cada dirección virtual DV leída de la trace1 o trace2 se calcula el desplazamiento (offset) y el número de página virtual (npv) usando el tamaño de página (PAGE_SIZE), aplicando offset = DV & (PAGE_SIZE - 1) y npv = DV >> b, donde b = log2(PAGE_SIZE).

Con npv se consulta la tabla de páginas y si la entrada es válida hay un HIT, se obtiene el marco físico, se marca su use_bit = 1 y se construye la dirección física como DF = (frame << b) | offset. Si no hay entrada válida, se produce un FALLO de página: si existe un marco libre se asigna directamente a esa página; en caso contrario se aplica el algoritmo Reloj, avanzando el puntero circular sobre los marcos, limpiando use_bit cuando está en 1 y eligiendo como víctima el primer marco cuyo use_bit está en 0. Una vez seleccionado el marco, se actualiza la tabla de páginas, se pone use_bit = 1 para la página cargada.

Experimentos y resultados

Se realizaron experimentos variando con 8, 16 y 32 número de marcos y 8 bytes (trace1.txt) y 4096 bytes (trace2.txt):

Traza	N marcos	Page size	Referencias	Fallos	Tasa de fallos
trace1	8	8	8192	8073	0.985474
trace1	16	8	8192	7943	0.969604
trace1	32	8	8192	7713	0.941528
trace2	8	4096	8192	7649	0.933716
trace2	16	4096	8192	7138	0.871338
trace2	32	4096	8192	6142	0.749756

en la trace1 tiene tasas de fallos muy altas incluso con 32 marcos de 98% a 94%, lo que sugiere baja localidad o un conjunto de trabajo mucho más grande que la memoria física disponible y en la trace2 mejora más al aumentar los marcos de 93% a 75%, indicando algo más de localidad de referencia: con más marcos, el algoritmo logra retener mejor las páginas activas. En ambas trazas, al aumentar el número de marcos (de 8 → 32) la tasa de fallos disminuye, como se espera con el algoritmo Reloj.

Conclusión

En conjunto, el trabajo permitió integrar y aplicar conceptos fundamentales de sincronización y gestión de memoria vistos en el curso de Sistemas Operativos. En la Parte I, la barrera reutilizable implementada como monitor demostró sincronizar correctamente las hebras, asegurando que ninguna avance de etapa hasta que todas lleguen al punto de encuentro. En la Parte II, el simulador de memoria virtual con algoritmo Reloj mostró cómo se traducen direcciones y cómo el número de marcos y la localidad de referencia influyen en la tasa de fallos de página. En conjunto, ambas partes evidencian la importancia de la sincronización y la gestión de memoria en el funcionamiento de un sistema operativo.