Reporte técnico: Proyecto final de Sistemas Operativos y Laboratorio

1. Información del Proyecto

- **Título del Proyecto:** compresor de archivos paralelo
- **Curso/Materia:** Sistemas Operativos
- **Integrantes:**
Juan Sebastian Mazo Loayza (juans.loaiza@udea.edu.co)
Sulay Gisela Martínez Barreto (sulay.martinez@udea.edu.co)
- **Fecha de Entrega:** 10 Julio 2025

2. Introducción

2.1. Objetivo del Proyecto

El objetivo principal es implementar un compresor de archivos paralelo que utilice múltiples hilos de ejecución para dividir archivos grandes en bloques más pequeños, comprimirlos de manera simultánea y crear un formato de archivo comprimido personalizado (.pz). El proyecto busca demostrar la implementación práctica de conceptos de sistemas operativos como threading, sincronización, operaciones de I/O paralelas y gestión de memoria.

2.2. Motivación y Justificación

La compresión de archivos es una operación computacionalmente intensiva que tradicionalmente se ejecuta de forma secuencial. En sistemas modernos con múltiples núcleos de CPU, esta aproximación no aprovecha completamente los recursos disponibles. ParZip aborda este problema implementando un algoritmo de compresión paralela que puede reducir significativamente el tiempo de procesamiento de archivos grandes.

La relevancia en sistemas operativos radica en la implementación de:

- Gestión de hilos (pthread)
- Sincronización con mutex
- Operaciones de I/O concurrentes
- Gestión eficiente de memoria
- Interacción con el sistema de archivos

2.3. Alcance del Proyecto

Funcionalidades Implementadas:

- Compresión paralela de archivos utilizando algoritmo zlib
- División automática en bloques configurables
- Formato personalizado .pz con header y metadatos
- Descompresión paralela con reconstrucción del archivo original
- Configuración automática basada en CPUs disponibles
- Niveles de compresión configurables (0-9)
- Interfaz de línea de comandos completa

Limitaciones:

- Formato propietario .pz (no compatible con compresores estándar)
- Requiere que el archivo completo quepa en disco durante el proceso
- No incluye verificación de integridad mediante checksums

3. Marco Teórico / Conceptos Fundamentales

3.1. Programación Paralela con Hilos (pthread)

ParZip utiliza la biblioteca pthread para crear múltiples hilos de ejecución que procesan bloques de datos de forma simultánea. Cada hilo es responsable de comprimir o descomprimir un bloque específico del archivo, permitiendo que el procesamiento se distribuya entre múltiples núcleos de CPU.

3.2. Algoritmo de Compresión zlib

La biblioteca zlib implementa el algoritmo de compresión DEFLATE, que combina LZ77 (para eliminar redundancias) con codificación Huffman (para compresión estadística). ParZip utiliza las funciones `compress2()` y `uncompress()` de zlib para el procesamiento de bloques individuales.

3.3. Sincronización con Mutex

Para evitar condiciones de carrera durante la escritura al archivo de salida, ParZip implementa un mutex (`pthread_mutex_t`) que serializa las operaciones de escritura, asegurando que solo un hilo pueda escribir al archivo en un momento dado.

3.4. Formato de Archivo Personalizado

El formato .pz incluye:

- **Header:** Metadatos del archivo (tamaño original, número de bloques, nivel de compresión)
- **Tabla de Bloques:** Información sobre cada bloque (tamaño original, comprimido, offset)
- **Datos Comprimidos:** Bloques de datos comprimidos almacenados secuencialmente

4. Diseño e Implementación

4.1. Diseño de la Solución

Decisiones de Diseño Clave:

1. **División por Bloques:** Permite paralelización eficiente y gestión de memoria controlada
2. **Pool de Hilos Dinámico:** Reutiliza hilos para procesar múltiples bloques
3. **Escritura Sincronizada:** Evita condiciones de carrera usando mutex
4. **Formato Personalizado:** Incluye metadatos necesarios para reconstrucción

4.2. Tecnologías y Herramientas

- **Lenguaje:** C (estándar C99)
- **Biblioteca de Compresión:** zlib
- **Sistema de Hilos:** pthread (POSIX threads)
- **Compilador:** GCC
- **Sistema Operativo:** Linux/Unix
- **Herramientas de Desarrollo:** Make, GDB (depuración)

4.3. Detalles de Implementación

Estructuras de Datos Principales:

```
// Header del archivo .pz
typedef struct {
    uint64_t magic;         // Número mágico: 0x504152574F52 ("PARZIP")
    uint32_t num_blocks;    // Número de bloques
    uint32_t block_size;    // Tamaño de cada bloque
```

```
    uint32_t compression_level; // Nivel de compresión (0-9)
    uint64_t original_size;   // Tamaño original del archivo
} parzip_header_t;


// Información de cada bloque
typedef struct {
    uint32_t block_id;       // ID del bloque
    uint32_t original_size;   // Tamaño original
    uint32_t compressed_size; // Tamaño comprimido
    uint64_t offset;         // Offset en el archivo
} block_info_t;
```

Algoritmo de Compresión:

1. **Inicialización:** Leer archivo, calcular número de bloques, crear header
2. **Distribución:** Asignar bloques a hilos disponibles
3. **Compresión Paralela:** Cada hilo procesa su bloque asignado
4. **Sincronización:** Escribir bloques comprimidos de forma thread-safe
5. **Finalización:** Actualizar metadatos y cerrar archivos

Gestión de Memoria:

- Asignación dinámica de buffers por hilo
- Liberación automática mediante mecanismos de limpieza (`cleanup`)
- Validación de asignaciones para prevenir fugas de memoria (`memory leaks`)

5. Pruebas y Evaluación

5.1. Metodología de Pruebas

Describe cómo probaste tu proyecto. ¿Qué enfoque de pruebas utilizaste? ¿Creaste casos de prueba específicos?

5.2. Casos de Prueba y Resultados

Presente los casos de prueba más importantes que ejecutó y los resultados obtenidos. Puede usar para ello una tabla como la siguiente:

ID Caso de prueba	Descripción del caso de prueba	Resultado esperado	Resultado obtenido	Éxito/Fallo
CP-001	Comprimir archivo de texto pequeño (100 KB)	Se crea un archivo .pz comprimido, sin errores.	Archivo .pz creado	Éxito
CP-002	Descomprimir archivo previamente comprimido	Se obtiene un archivo idéntico al original	Archivo recuperado correctamente	Éxito
CP-003	Comprimir archivo con nivel de compresión 9	Archivo comprimido significativamente menor que el original	Compresión con zlib nivel 9, el archivo se comprimió en más de un 90%	Éxito
CP-004	Ejecutar con 1 solo hilo -t 1	Compresión/descompresión correcta, sin errores de concurrencia	Compresión realizada con 1 hilo, tamaño final: 32 bytes, sin errores	Éxito
CP-005	Ejecutar con hilos máximos (-t 32 dependiendo de la máquina)	Compresión correcta sin errores de concurrencia, incluso con hilos en exceso	Compresión con 32 hilos, 1 bloque procesado correctamente sin errores	Éxito
CP-006	Comprimir archivo que no existe	El programa debe detectar el error y mostrar el mensaje claro	Error: El archivo de entrada 'noexiste.txt' no existe	Éxito
CP-007	Descomprimir archivo corrupto o inválido	Error de formato .pz	Error: No se pudo	Éxito

		, termina con código de error.	leer el header del archivo	
CP-008	Cancelar sobreescritura de archivo existente	Se muestra mensaje de advertencia y, al responder “n”, la operación se cancela	`" El archivo ya existe... ¿Sobrescribir? (s/N): n" → Operación cancelada.`	Éxito
CP-009	Verificar integridad bit a bit con cmp entre archivo original y recuperado	cmp o diff no deben reportar ninguna diferencia	`cmp archivo.txt verificacion_out.txt ejecutado sin salida ⇒ archivos iguales`	Éxito
CP-010	Comprimir con tamaño de bloque personalizado (-b 32768)	Compresión realizada correctamente con bloques de 32 KB	Bloque comprimido correctamente (1 bloque de 32 KB)	Éxito

5.3. Evaluación del Rendimiento (si aplica)

Hardware: Rizen 5 7600X (6 cores, 12 threads), 32GB RAM

Archivos de prueba:

- Texto: 100MB (archivo .txt repetitivo)
- Video: 500MB (archivo .mp4)
- Binario: 1GB (archivo .bin aleatorio)

Comparación: gzip, 7zip, ParZip

Archivo	Método	Tiempo (s)	Speedup	Ratio Compresión
100MB texto	gzip	8.5s	1.0x	95.2%
100MB texto	ParZip (1 hilo)	9.1s	0.93x	94.8%
100MB texto	ParZip (4 hilos)	2.8s	3.0x	94.8%
100MB texto	ParZip (8 hilos)	1.9s	4.5x	94.8%
500MB video	gzip	45.2s	1.0x	12.3%
500MB video	ParZip (8 hilos)	12.1s	3.7x	11.8%

- Memoria base: ~50MB por proceso
- Por hilo: tamaño_bloque × 3 (buffers entrada/salida/compresión)
- `Fórmula: Memoria ≈ 50MB + (num_hilos × tamaño_bloque × 3)`

Casos donde ParZip sobresale:

- Archivos grandes (>100MB): 3-5x más rápido
- Sistemas multi-core: Aprovecha todos los núcleos
- Archivos con buena compresibilidad: Máximo beneficio

Limitaciones:

- Archivos pequeños (<10MB): Overhead de sincronización
- Archivos ya comprimidos: Poco beneficio del paralelismo
- Sistemas con poca RAM: Limitado por memoria disponible

6. Conclusiones

- **Cumplimiento de objetivos**:
  Se logró implementar exitosamente un compresor de archivos paralelo que aprovecha múltiples núcleos de CPU para realizar operaciones de compresión y descompresión en bloques. El sistema desarrollado cumple con las funcionalidades propuestas, incluyendo compresión con zlib, manejo de hilos con `pthread`, y un formato de archivo propio `.pz`.

- **Aplicación de conceptos de sistemas operativos**:
  El proyecto permitió aplicar y consolidar conocimientos clave de sistemas operativos, como la programación con hilos, sincronización mediante mutex, operaciones de entrada/salida concurrentes, y gestión eficiente de memoria. Estas habilidades son fundamentales para el desarrollo de software de alto rendimiento en entornos multitarea.

- **Resultados funcionales y confiables**:
  Las pruebas realizadas evidencian que el programa funciona correctamente en escenarios típicos y de borde. Casos como la compresión con diferentes niveles, manejo de errores (archivos inexistentes o corruptos), y validación de integridad bit a bit fueron superados exitosamente, lo cual demuestra la robustez del sistema.

- **Limitaciones identificadas**:
  Aunque el formato `.pz` diseñado es funcional, no es compatible con otros compresores estándar. Además, el sistema requiere espacio suficiente en disco para mantener el archivo completo durante el proceso, y actualmente no incluye verificación de integridad con checksums, lo cual puede ser una mejora futura.

- **Aprendizaje significativo**:
  El desarrollo de ParZip representó una experiencia enriquecedora en términos de diseño, implementación y depuración de aplicaciones concurrentes. Se adquirió una mejor comprensión de cómo los sistemas operativos gestionan la concurrencia, el acceso al sistema de archivos y el uso de bibliotecas de compresión a bajo nivel.

- **Evaluación del éxito del proyecto**:
  El proyecto fue exitoso en términos de funcionamiento, rendimiento y aprendizaje. Se implementó una solución que, aunque académica, tiene potencial de uso práctico, demostrando el poder de la programación paralela para resolver problemas reales de procesamiento de datos.

7. Trabajo Futuro (Opcional)

Sugiera posibles extensiones o mejoras que se podrían realizar en el proyecto en el futuro.

8. Referencias

Stevens, W. Richard & Rago, Stephen A. (2013). *Advanced Programming in the UNIX Environment, 3rd Edition*. Addison-Wesley Professional.

Butenhof, David R. (1997). *Programming with POSIX Threads*. Addison-Wesley Professional.

Tanenbaum, Andrew S. & Bos, Herbert (2014). *Modern Operating Systems, 4th Edition*. Pearson.

zlib Documentation. (2023). *zlib 1.2.11 Manual*. Retrieved from https://www.zlib.net/manual.html

POSIX Threads Programming. (2023). *pthread Tutorial*. LLNL Computing. Retrieved from https://computing.llnl.gov/tutorials/pthreads/

Deutsch, P. & Gailly, J-L. (1996). *RFC 1951 - DEFLATE Compressed Data Format Specification*. Internet Engineering Task Force.

GNU C Library Documentation. (2023). *File System Interface*. Retrieved from https://www.gnu.org/software/libc/manual/

9. Anexos

- Repositorio: https://github.com/juanselm/finalSO.git