



Instituto Tecnológico de Costa Rica  
Ingeniería en Computación  
IC6600 - Principios de Sistemas Operativos

## Proyecto 1

Paralelismo y Concurrencia

Laura Amador Salas  
lamador@estudiantec.cr  
2022091724

Tamara Nicole Rodríguez Luna  
nicolerodriguezluna@estudiantec.cr  
2021077818

Isaac Picado Ortega  
ipicado@estudiantec.cr  
2023086373

San José, Costa Rica  
Septiembre 2025

# Índice general

<b>1. Introducción</b>	<b>2</b>
1.1. Breve descripción del proyecto Gutenberg . . . . .	2
1.2. Breve historia y descripción del algoritmo de Huffman . . . . .	2
1.3. Explicación de qué es un algoritmo de compresión con pérdida y sin pérdida . . . . .	3
1.4. Explicación de la diferencia entre paralelismo y concurrencia . . . . .	4
<b>2. Desarrollo</b>	<b>5</b>
2.1. Explicación detallada del funcionamiento de la función fork() y la biblioteca pthread . . . . .	5
2.1.1. Función fork() . . . . .	5
2.1.2. Biblioteca Pthread . . . . .	6
2.2. Explicación de sus estrategias para parallelizar los programas en cada una de las versiones (fork y pthread) . . . . .	7
2.2.1. Versión fork() . . . . .	7
2.2.2. Versión pthread . . . . .	8
2.3. Descripción del proyecto con instrucciones de cómo compilarlo y correrlo en el entorno base del curso . . . . .	10
2.4. Discusión . . . . .	36
2.5. Conclusiones . . . . .	44
<b>Referencias</b>	<b>45</b>

# Capítulo 1

## Introducción

### 1.1. Breve descripción del proyecto Gutenberg

El Proyecto Gutenberg es una iniciativa que comenzó en 1971, cuando Michael Hart consiguió acceso a un mainframe de la Universidad de Illinois y decidió utilizar los recursos de cómputo para digitalizar y distribuir gratuitamente textos de dominio público. La filosofía detrás del proyecto se centra en lo que Hart denominó "tecnología replicadora", es decir, la capacidad de que cualquier obra ingresada en una computadora pueda ser copiada y compartida de manera indefinida, sin costos adicionales y en formatos accesibles para la mayor cantidad de personas posible (Hart, 1992).

Con el paso de los años, el Proyecto Gutenberg se consolidó como la primera gran biblioteca digital del mundo, albergando miles de libros electrónicos en formatos simples como *Plain Vanilla ASCII*, y posteriormente en otros formatos más avanzados. Su objetivo principal ha sido siempre el mismo que es poner a disposición de cualquier persona, sin barreras económicas ni legales, obras literarias y documentos de relevancia cultural. Actualmente, el proyecto continúa siendo una de las principales referencias de acceso libre al conocimiento, con colecciones que van desde literatura clásica hasta textos académicos (Hart, 1992).

### 1.2. Breve historia y descripción del algoritmo de Huffman

David Huffman siempre fue excelente en el ámbito académico. A los quince años se graduó de la secundaria, y entró a la universidad justo después de cumplir sus dieciséis. A los diecinueve años obtuvo su primer título de ingeniero y se convirtió en uno de los oficiales navales más jóvenes (Huffman, 2023). Unos años después Huffman fue aceptado al MIT, en donde uno de sus profesores les dió a elegir entre tomar el examen final tradicional o mejorar un algoritmo de

compresión de datos, a lo que David eligió la segunda opción (Huffman, 2023). A este algoritmo hoy en día se le conoce como el procedimiento de codificación de Huffman.

El algoritmo de Huffman sigue siendo utilizado como una manera sumamente efectiva de comprimir datos, logrando comprimir desde un 20 % y hasta un 90 % dependiendo del tipo de información. Este algoritmo usa una tabla de frecuencias que especifica qué tanto ocurre cada carácter, esto para optimizar la manera en la que se representan como cadenas de bits (Cormen, Leiserson, Rivest, y Stein, 2001).

Los datos necesarios para construir un algoritmo de Huffman son los caracteres del texto que se quiere comprimir y la frecuencia con la que aparecen. El proceso se basa en un for que se repite  $n-1$  veces ( $n$  siendo la cantidad de caracteres en el alfabeto) y que extrae el carácter con menor frecuencia y lo coloca como hijo del lado izquierdo de un nodo padre (representado con un 0 en la rama que los une). Luego se repite esto con el carácter con la segunda menor frecuencia y se coloca como hijo del lado derecho de ese mismo nodo padre (representado con un 1 en la rama que los une) (Cormen y cols., 2001). Básicamente, esto hace que los dos caracteres con menor frecuencia se combinen y sean hijos de un solo nodo. A este nodo nuevo se le asignará como frecuencia la suma de las frecuencias de sus dos hijos (Cormen y cols., 2001).

Cuando el proceso se repite, los dos hijos seleccionados en la repetición pasada no serán elegibles, sino que ahora el nodo nuevo, su padre, será elegible cuando se comparan las frecuencias menores. Siguiendo este algoritmo se crea un árbol binario completo, lo que significa que cada nodo padre tiene dos hijos, y vuelve el código Huffman óptimo para la compresión de archivos de caracteres (Cormen y cols., 2001). Finalmente, para la elegir la cadena de bits que representará cada carácter, simplemente debe seguirse el camino desde la raíz del árbol hasta la hoja que contiene el carácter. Dependiendo de las ramas por las que se pase, se generará una cadena de unos y ceros, la cual se utilizará para representar a ese carácter (Cormen y cols., 2001).

### **1.3. Explicación de qué es un algoritmo de compresión con pérdida y sin pérdida**

Los algoritmos de compresión buscan reducir la cantidad de bits necesarios para representar datos digitales, optimizando tanto su almacenamiento como su transmisión. En términos generales, se clasifican en dos categorías: los de compresión con pérdida (*lossy*) y los de compresión sin pérdida (*lossless*). Cada una de estas categorías tiene aplicaciones y ventajas específicas según el tipo de información y el nivel de fidelidad requerido (Kavitha, 2016; Singh, Kumar, Chouhan, y Shrivastava, 2016).

La compresión con pérdida sacrifica parte de la información original con el fin de obtener ratios de compresión mucho más elevados. Este enfoque se basa en descartar datos que el ojo o el oído humano apenas perciben, como

ocurre con ciertas variaciones de color o sonidos fuera del rango auditivo. Es ampliamente utilizada en imágenes (JPEG), audio (MP3) y video (MPEG), donde una ligera degradación en la calidad es aceptable frente al ahorro de espacio y la mejora en la velocidad de transmisión. A diferencia de los métodos sin pérdida, la compresión con pérdida no permite reconstruir de manera exacta el archivo original, pero logra un equilibrio entre calidad percibida y eficiencia (Singh y cols., 2016).

Por su parte, la compresión sin pérdida garantiza que al descomprimir la información se recupere exactamente el contenido original, sin alteraciones. Por este motivo, se utiliza en contextos donde la fidelidad de los datos es crítica, como en bases de datos, archivos de texto o imágenes médicas. Algunos ejemplos de técnicas sin pérdida son Huffman, LZW o la codificación aritmética, todas ellas se basan en la eliminación de redundancias presentes en los datos (Kavitha, 2016).

#### **1.4. Explicación de la diferencia entre paralelismo y concurrencia**

En primer lugar, la concurrencia es la ejecución de múltiples tareas en un periodo de tiempo superpuesto; es decir, que una tarea puede comenzar antes que se complete la anterior; no obstante, no se ejecutarán al mismo tiempo (Sari, 2023). Asimismo, la programación concurrente se define como una metodología implementada con el objetivo de dar resolución a los problemas por medio de la ejecución de diferentes labores de programación permitiendo al sistema continuar con sus actividades sin necesidad de iniciar o dar fin a otras labores, permitiendo así reducir el tiempo de respuesta del sistema, implementando solo una unidad de procesamiento. Esto implica la división de la tarea en múltiples partes, las cuales se procesan de forma simultánea, mas no al mismo tiempo (Bustos, 2024).

Por el contrario, el paralelismo significa la capacidad de ejecutar tareas independientes de un programa en el mismo instante, estas puede lograrse por medio de otro núcleo del procesador, otro procesador o una computadora diferente en un sistema distribuido (Sari, 2023). También, se conoce como paralelismo computacional o computación paralela a la técnica de programación donde se divide los inconvenientes de gran magnitud en apartados de menor tamaño que serán resueltos en paralelo, en otras palabras, a partir de la ejecución de múltiples instrucciones de forma simultánea haciendo uso de dos o más procesadores a cargo de resolver las tareas (Bustos, 2024).

# Capítulo 2

## Desarrollo

### 2.1. Explicación detallada del funcionamiento de la función fork() y la biblioteca pthread

#### 2.1.1. Función fork()

fork() es una llamada al sistema agregada por los diseñadores de Unix como un mecanismo para crear procesos. Este crea procesos nuevos idénticos al proceso padre con la excepción del valor de retorno del sistema (Baumann, Appavoo, Krieger, y Roscoe, 2019).

Así pues, para iniciar un nuevo proceso en los sistemas Unix, el shell ejecuta una llamada a sistema fork(). Luego, el programa seleccionado es cargado a la memoria por medio de una llamada al sistema exec() y el programa es ejecutado (Silberschatz, Galvin, y Gagne, 2018).

El comando fork() hace una copia completa del proceso en ejecución y la única forma de diferenciar entre el proceso hijo y el proceso padre es por medio del retorno del comando, puesto que si creamos un programa en C como el siguiente:

```
1 #include <stdio.h>
2
3 int main(void) {
4     int pid = fork();
5
6     if ( pid == 0 ) {
7         printf( "Esto se esta imprimiendo desde el proceso hijo\n" );
8     } else {
9         printf( "Esto se esta imprimiendo en el proceso padre:\n"
10                " - el identificador de proceso (pid) del hijo es %d\n",
11                pid );
12
13     return 0;
14 }
```

`pid` será 0 si es ejecutado desde proceso hijo, sino está en el proceso padre (Harder, 2012).

Sin embargo, algunos problemas conocidos con fork incluyen que no es thread-safe, es inefficiente, inescalable, introduce problemas de seguridad, ha perdido su simplicidad clásica al afectar las demás abstracciones del sistema con las que antes era ortogonal, es hostil a la implementación en modo usuario de la funcionalidad del sistema operativo al romper desde la entrada y salida en el búfer hasta las redes que eluden el núcleo y, lo más problemático es que al exigir que todas las capas de un sistema deban ser compatibles con el fork, fuerza a que las piezas del sistema operativo dejen de ser pequeñas e independientes (Baumann y cols., 2019).

### 2.1.2. Biblioteca Pthread

La biblioteca pthread (POSIX Threads) es un conjunto de funciones usadas tanto en C como en C++, las cuales incluyen operaciones para la creación, sincronización, eliminación y en general gestión de hilos. Para usarse se requiere de la inclusión del archivo header específico de Pthread. Cada función de la biblioteca generalmente usa una variable oculta para rastrear el estado del objeto o los recursos involucrados (Stewart y cols., 2013).

Pthread es usada normalmente en sistemas que utilizan arquitecturas de memoria compartida, específicamente para la implementación de programas paralelos. Al utilizar esta biblioteca para implementar *multithreading* (ejecución de múltiples hilos de forma paralela), permite que todos los hilos del programa tengan acceso a las variables globales, mientras que las variables locales sean privadas para el hilo que está ejecutando la función (Pacheco, 2011).

El orden de ejecución de líneas de código por los diferentes hilos, usualmente suele ser no determinista, o sea, pueden generar salidas diferentes aun cuando utilicen la misma entrada. Esto ocasiona que, a la hora de que varios hilos traten de acceder a un recurso compartido o global, ocurran condiciones de carrera. Controlar estas condiciones es una tarea sumamente importante cuando se desarrollan programas que utilizan memoria compartida; por suerte, Pthread también provee mecanismos como los mutex y las variables de condición para evitar conflictos de acceso en regiones críticas (Stewart y cols., 2013) (Pacheco, 2011).

El ciclo de vida de un hilo en Pthread tiene cuatro pasos: creación, sincronización, ejecución y terminación. Los hilos se crean con la función `pthread_create`, la cual requiere un puntero a un identificador de hilo de tipo `pthread_t`, un puntero a los atributos del hilo, un puntero a la función que ejecutará el hilo, y un puntero a un argumento para pasar datos a la función (Alessandrini, 2016). La sincronización de hilos se realiza utilizando las funcionalidades anteriormente mencionadas, como lo son los mutex (`pthread_mutex_t`) y las variables de condición (`pthread_cond_t`). Su trabajo es coordinar la ejecución de los hilos y prevenir errores en regiones de riesgo. Finalmente, la terminación se maneja a través de funciones como `pthread_exit` y `pthread_join`. Esta segunda permite que un hilo espere a que otro hilo termine (Alessandrini, 2016).

Profundizando en el proceso de sincronización de hilos, del cual las herramientas principales son los mutex y las variables de condición, se denotan las siguientes funcionalidades:

Los mutex se usan para acomodar y administrar el acceso por varios hilos a una sola región crítica. Se inicializan con `pthread_mutex_init`, se bloquean con `pthread_mutex_lock`, se desbloquean con `pthread_mutex_unlock`, y también tienen la función `pthread_mutex_trylock` la cual intenta bloquear un mutex, pero no hace nada si el mutex ya estaba previamente bloqueado (Alessandrini, 2016) (Domeika, 2013). Al mutex ser bloqueado, solamente permite que el hilo que lo solicitó pueda usar la región crítica encasillada dentro del mismo. Si mientras está bloqueado, otro hilo solicita el uso de esta región, este queda en espera hasta que el hilo anterior termine de ejecutar esa región y desbloquee el mutex.

Por otro lado, las variables de condición son objetos usados para pausar la ejecución de un hilo hasta que una condición específica ocurra, y siempre están asociadas a un mutex. Otro hilo en ejecución puede "despertar"<sup>el</sup> hilo en pausa con una señal de condición o un *broadcast*. Las funciones usadas para estas tres operaciones son `pthread_cond_wait`, `pthread_cond_signal` y `pthread_cond_broadcast` respectivamente (Domeika, 2013).

Finalmente, el uso de Pthread implica una gran cantidad de problemas de rendimiento si no se implementa de la manera correcta. Entender el costo de las llamadas de Pthread es importante a la hora de optimizar un programa, ya que la excesiva creación y destrucción de hilos puede llegar a ser muy costosa (Stewart y cols., 2013). No obstante, el uso de *thread pools* es una solución para este problema, ya que con ellas se crean todos los hilos juntos, y se reutilizan durante el tiempo de ejecución de la aplicación (Jeffers, Reinders, y Sodani, 2016). Si bien es cierto que Pthread tiene las soluciones para problemas ocasionados por *overheads* (que son de los más comunes en paralelización), estas soluciones son factibles solamente si se tiene el conocimiento de como estructurar el programa, tomando en cuenta las regiones críticas, el posicionamiento de los mutex, y el uso de los hilos.

## 2.2. Explicación de sus estrategias para paralelizar los programas en cada una de las versiones (`fork` y `pthread`)

### 2.2.1. Versión `fork()`

La estrategia de paralelización en la versión implementada con `fork()` está fundamentada en el modelo *fork-join*. El proceso padre identifica todos los archivos .txt en el directorio y los ordena para asegurar un procesamiento determinista. Por cada archivo, el padre crea un proceso hijo independiente mediante la llamada al sistema `fork()`, que se encarga de ejecutar todo el flujo de compresión: lectura, conteo de frecuencias, construcción del árbol de Huffman,

generación de la tabla de códigos y empaquetado en bits. Cada hijo escribe su resultado en un archivo temporal (`.part`), de modo que los procesos no comparten estado ni requieren mecanismos complejos de sincronización (Tammineni, 2022; Monroe, 2023).

Para gestionar la concurrencia, se definió un límite máximo de procesos activos, asociado por defecto al número de núcleos disponibles en la máquina. El proceso padre emplea `waitpid()` para esperar la finalización de alguno de los hijos antes de crear nuevos, evitando la sobrecarga del sistema. Este mecanismo implementa un control de flujo simple, similar al grado de paralelismo configurable en otros entornos de ejecución paralela, donde la cantidad de procesos concurrentes puede ajustarse según los recursos disponibles (Apache Maven, 2025). De esta forma, se garantiza que la ejecución pueda escalar sin comprometer la estabilidad del entorno.

Una vez que todos los procesos hijos han terminado, el proceso padre realiza la fase de *join*, ensamblando el archivo final comprimido (`archive.hfa`). Para ello, abre un archivo binario de salida, escribe un encabezado con el número total de archivos procesados y concatena cada pieza temporal en orden. El uso de búferes grandes para la copia de datos reduce el número de llamadas al sistema y mejora el rendimiento general. Este patrón se basa en principios de programación paralela en donde la independencia de las unidades de trabajo simplifica la integración final y reduce el riesgo de errores (Monroe, 2023).

Finalmente, la implementación mide los tiempos de compresión y descompresión usando un reloj, lo que permite evaluar el desempeño de manera precisa y comparar con las versiones serial y con hilos. Aunque la técnica no emplea bibliotecas de paralelismo de alto nivel como OpenMP, toma de ese ecosistema la idea de balancear el trabajo según los recursos de hardware disponibles, distribuyendo las tareas de manera eficiente y asegurando resultados equivalentes a los originales. La versión basada en `fork()` en general combina aislamiento de procesos, simplicidad de sincronización y capacidad de escalar según la carga de trabajo (Huseynzade, 2021).

### 2.2.2. Versión pthread

En la versión de Pthread, la estrategia utilizada para la paralelización se basó en dividir los trabajos a nivel de archivo. Cada archivo de texto se consideró una tarea independiente que se ejecutaría en paralelo. Para gestionar la distribución de tareas se implementó un *thread pool* con una cola de tareas, en el cual el *thread pool* crea un conjunto de hilos cuando se corre la aplicación, y cada uno de estos hilos ejecuta un bucle en el cual revisan la cola y esperan a que haya una tarea disponible (Jeffers y cols., 2016). El programa principal, añade a la cola las tareas, las cuales pueden ser la compresión o descompresión de un archivo. Con el uso de estos elementos se evita el costo de constantemente crear y destruir hilos por cada archivo, con el panorama de que se vuelva más eficiente en escenarios con grandes cantidades de archivos (Jeffers y cols., 2016).

La cola de tareas está protegida por un mutex y una variable de condición. Este mutex garantiza que solo un hilo a la vez pueda acceder a la cola para

insertar o extraer tareas, previniendo condiciones de carrera. La variable de condición se utiliza para poner en espera a los hilos cuando no hay trabajo y despertarlos cuando el programa principal añade nuevas tareas a la cola. Este esquema corresponde al patrón productor-consumidor en memoria compartida, donde el productor (hilo principal) alimenta la cola y los consumidores (hilos del pool) ejecutan las funciones asignadas (Jenkov, 2021).

En el archivo encargado de la compresión (`compress_dir.c`) se definió una estructura compartida que contiene un vector de resultados y un índice global. Cada hilo, al terminar de comprimir un archivo, debe guardar su resultado (nombre del archivo, texto original, árbol de Huffman, tabla de códigos y datos comprimidos). Para evitar que dos hilos escriban en la misma posición del vector, se protege esta operación con un mutex adicional. La sección crítica realmente es muy breve, solamente reserva un índice único y copia los punteros de los resultados. El trabajo costoso como la lectura de archivos, el conteo de frecuencias, la construcción del árbol de Huffman y compresión del texto se realiza fuera del mutex. La elección de este acomodo reduce la contención y aumenta la eficiencia, intentando que las secciones críticas sean lo más cortas posibles (Pacheco, 2011).

Una parte importante de la compresión en pthread fue el empaquetado de bits, implementado en `huffio.c`. El algoritmo de Huffman genera una cadena de caracteres ‘0’ y ‘1’, pero almacenarla así no reduce tanto el tamaño del archivo. Para disminuir el espacio que ocupa, se implementó una función que agrupa cada ocho caracteres en un byte, desplazando y combinando los bits. De esta forma se genera un arreglo de `uint8_t` que representa el texto comprimido de manera compacta. En la descompresión se aplica la operación inversa: se desempaquetan los bits y se reconstruye la cadena de ‘0’ y ‘1’, que luego se recorre con el árbol de Huffman para recuperar el texto original. Esta decisión aumenta el costo computacional, pero mejora la compacidad del archivo final (Cornell University, 2024).

Por último, en la parte de descompresión (`decompress_dir.c`) también se usó el thread pool. Primero, el programa principal indexa el archivo contenedor .hfa para identificar los metadatos de cada archivo comprimido: nombre, longitud original, árbol serializado y bloque de datos. Luego añade a la cola una tarea por archivo a descomprimir. Cada hilo reconstruye el árbol de Huffman de su entrada, lee y desempaquetá los bits correspondientes y finalmente escribe el archivo de texto. Como cada hilo trabaja en archivos distintos, no es necesaria más sincronización que la proporcionada por la cola del thread pool (Pacheco, 2011).

### **2.3. Descripción del proyecto con instrucciones de cómo compilarlo y correrlo en el entorno base del curso**

En Debian dele clic al botón en la esquina superior izquierda de la pantalla.

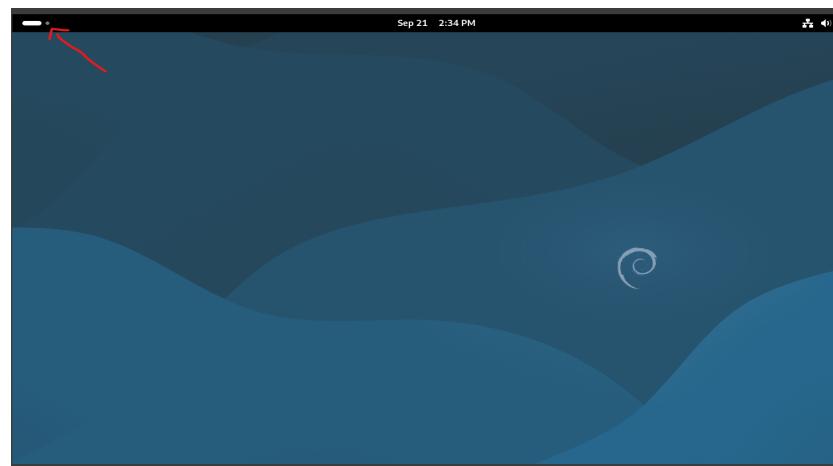


Figura 2.1: Menú de Debian.

Seleccione el ícono de Firefox para abrir el navegador.

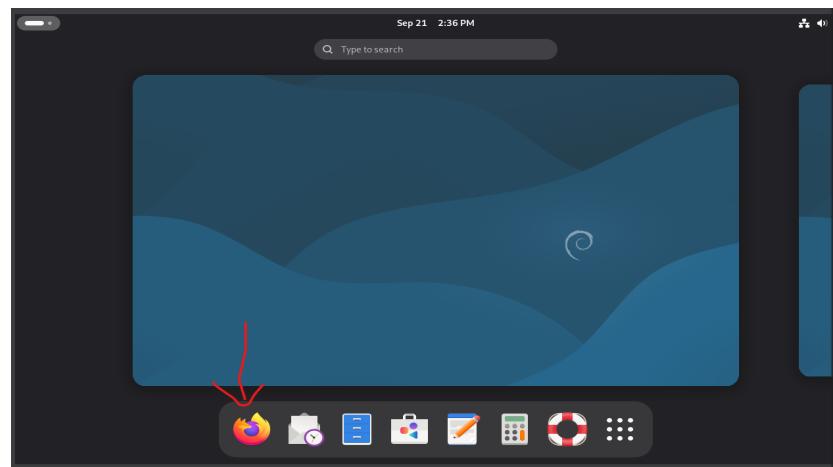


Figura 2.2: Ícono de Firefox en Debian.

Clic a la barra de búsqueda del navegador.

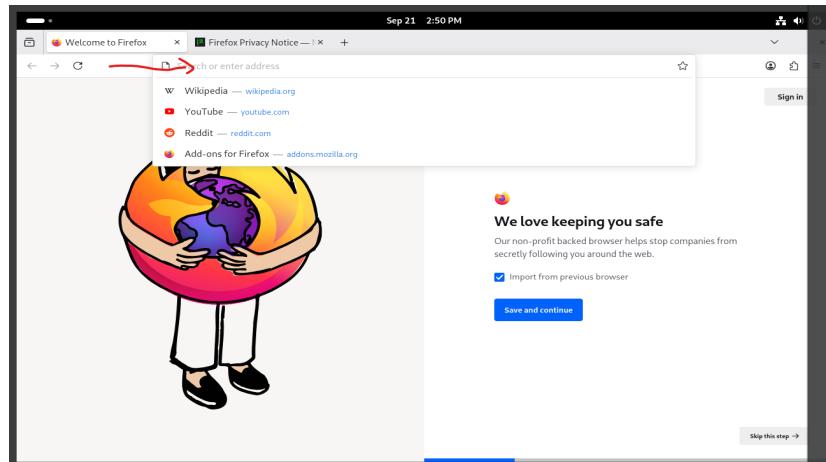


Figura 2.3: Barra de búsqueda en Firefox.

Ingresar la siguiente dirección <https://github.com/nicolerodriguezluna/OP-Proyecto1>

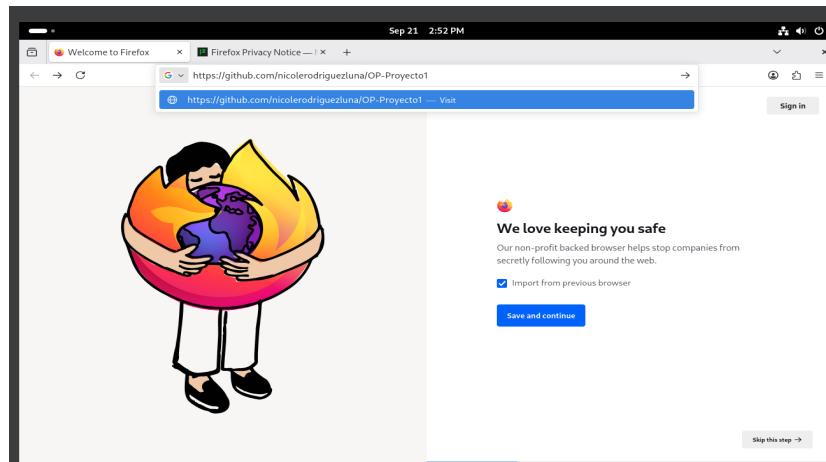


Figura 2.4: Ingresar dirección del repositorio en la barra de búsqueda.

Ingrese su usuario o correo asociado a su cuenta de GitHub en el espacio indicado.

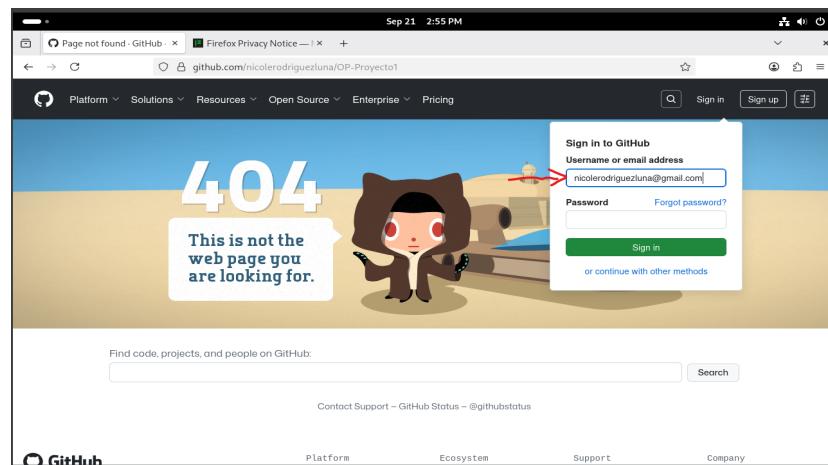


Figura 2.5: Ingresar usuario o correo asociado a GitHub.

Ingrese la contraseña de su cuenta de GitHub en el espacio indicado.

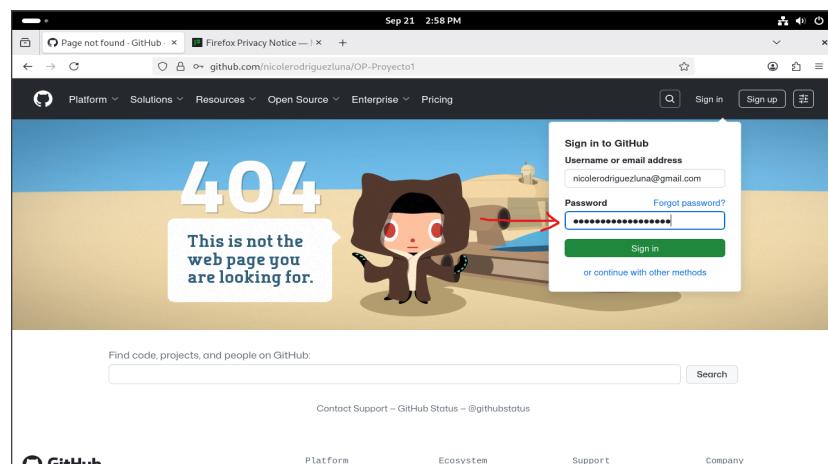


Figura 2.6: Ingresar contraseña de GitHub.

Clic al botón verde "Sign in"

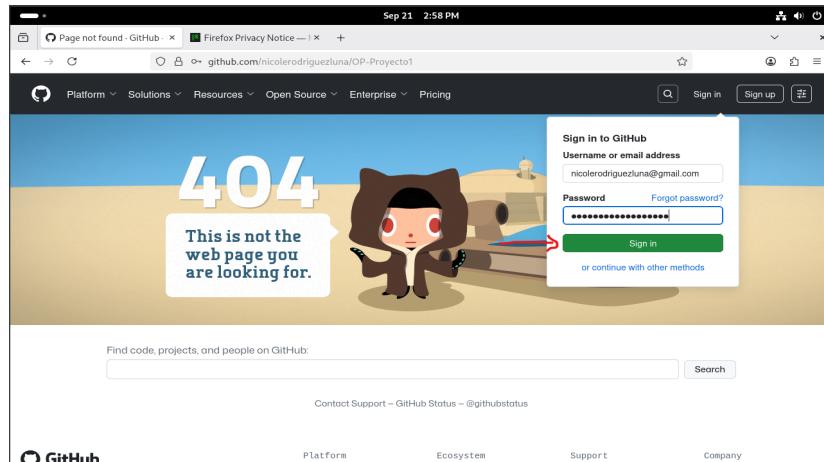


Figura 2.7: Botón "Sign in".

Si su cuenta está configurada con autenticación en 2 pasos necesitará ingresar a su correo, copiar el código recibido e ingresararlo en el espacio indicado

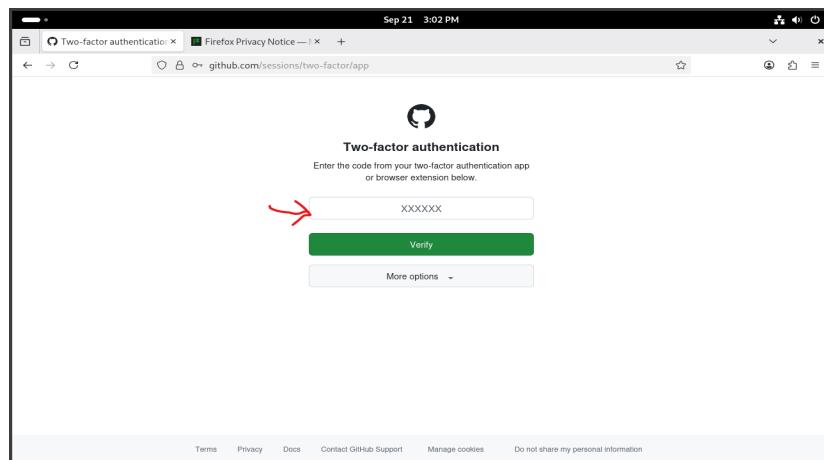


Figura 2.8: Ingresar autenticación en 2 pasos.

Después de eso, debería ser capaz de ver el repositorio así

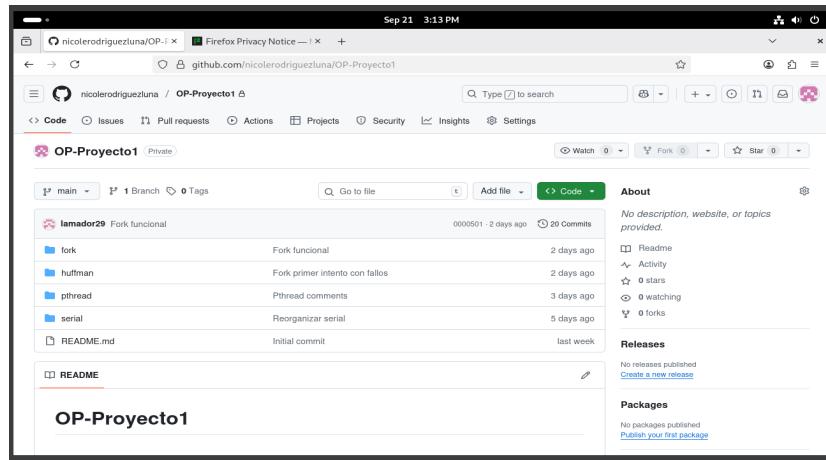


Figura 2.9: Vista general del repositorio.

Dale clic al botón verde “<>Code”

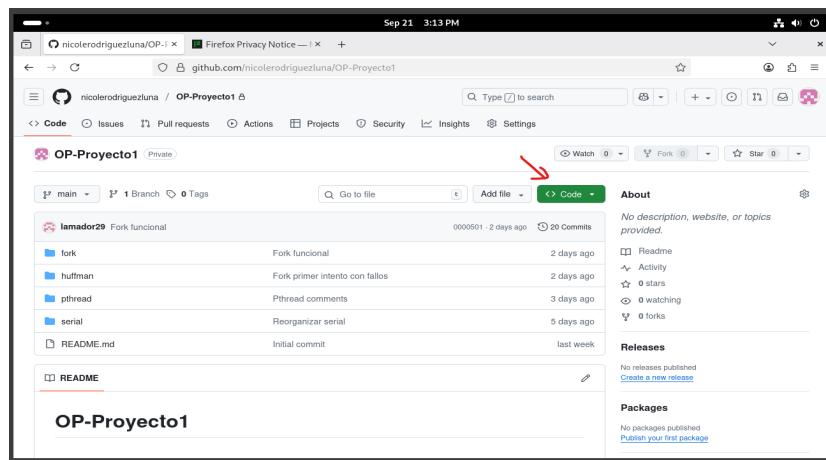


Figura 2.10: Botón Code.

Luego, clic al botón "Download ZIP"

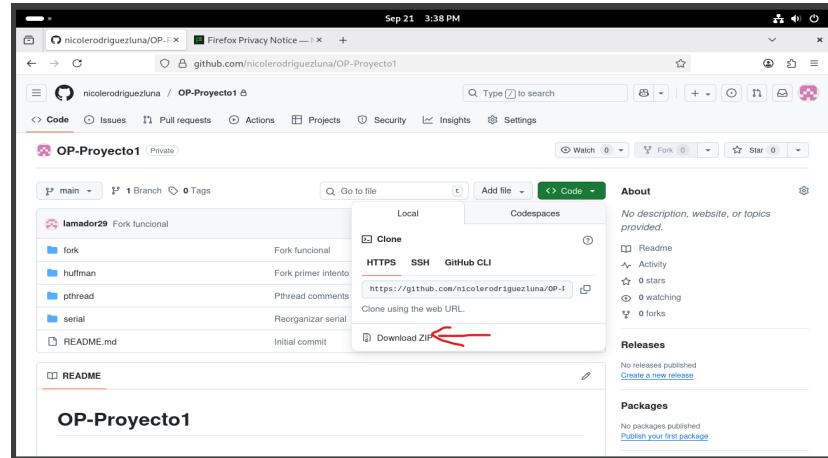


Figura 2.11: Botón "Download ZIP".

Se da clic al botón de carpeta

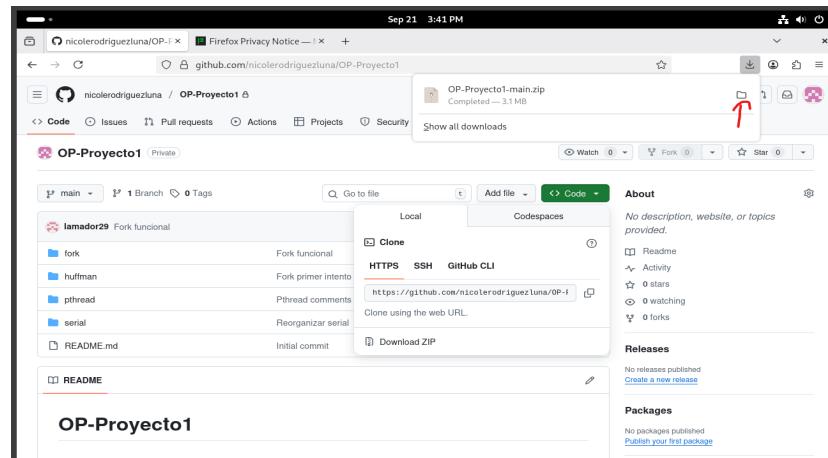


Figura 2.12: Botón de carpeta.

Se va a abrir una ventana y se le deba dar clic derecho al archivo

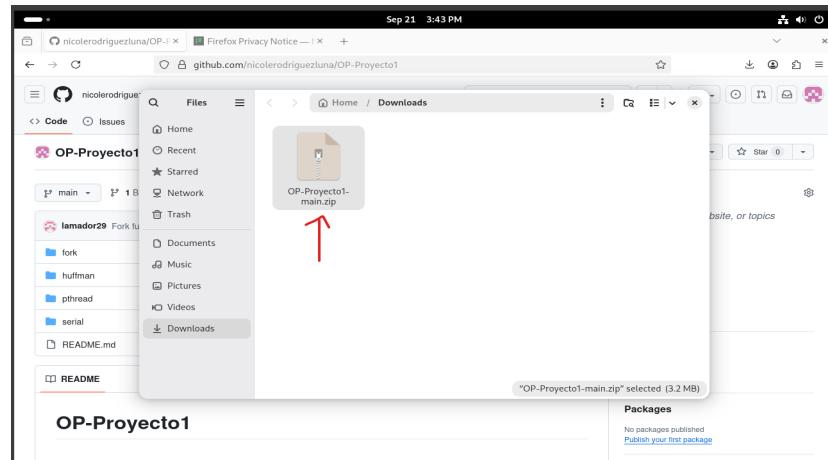


Figura 2.13: Clic derecho al archivo.

Después se da clic al botón “Extract“

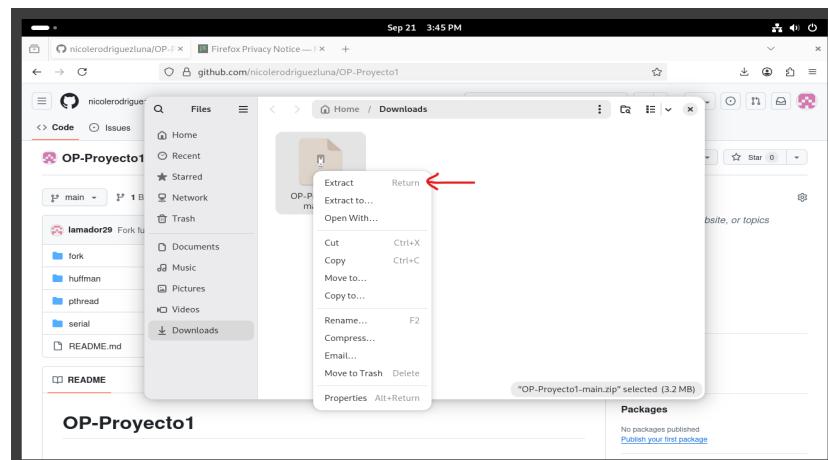


Figura 2.14: Botón de “Extract“.

Doble click a la carpeta que se creó

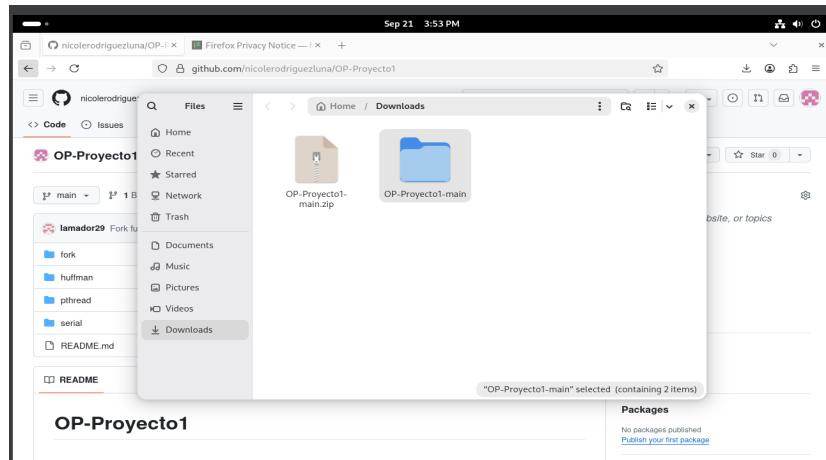


Figura 2.15: Doble clic a la carpeta creada.

Dentro de la carpeta del proyecto se le debe dar clic derecho a una parte en blanco de la ventana y darle clic a "Open in Terminal"

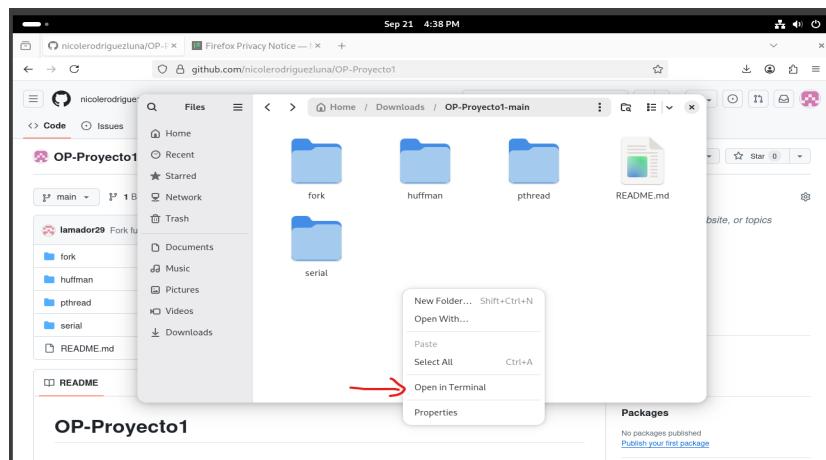


Figura 2.16: Abrir carpeta en Terminal.

Ahora, en la ventana de terminal primero hay que escribir el comando "su" y presionar la tecla Enter

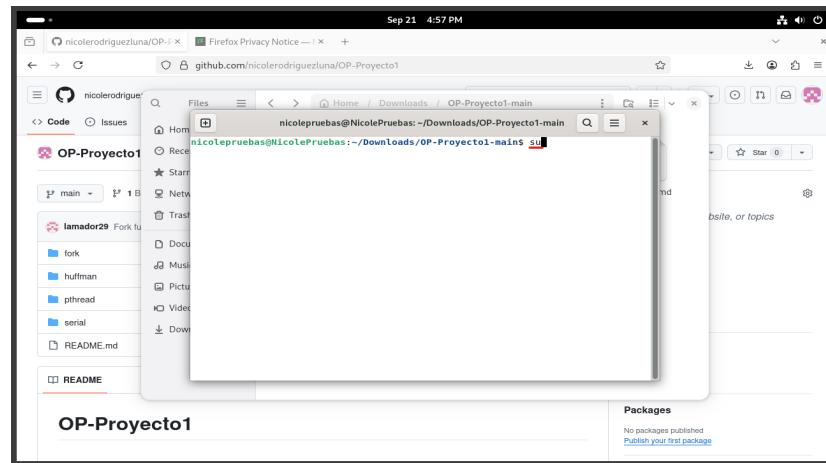


Figura 2.17: Usar comando "su".

Se escribe la contraseña de root y se presiona la tecla Enter

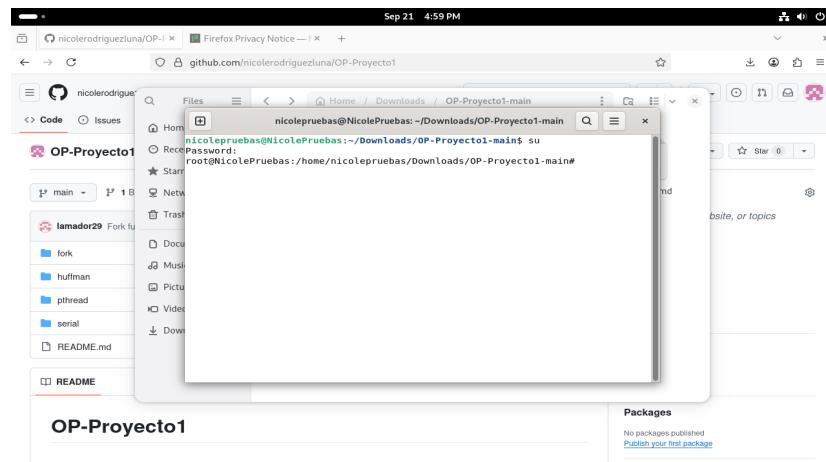
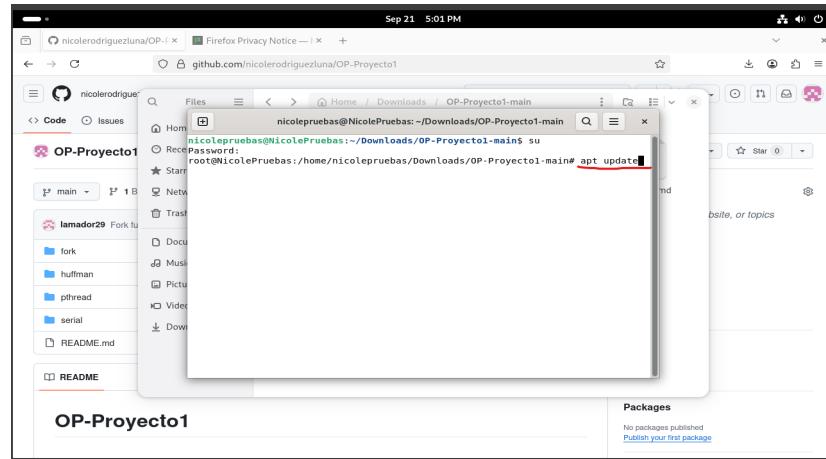


Figura 2.18: Escribir contraseña de root.

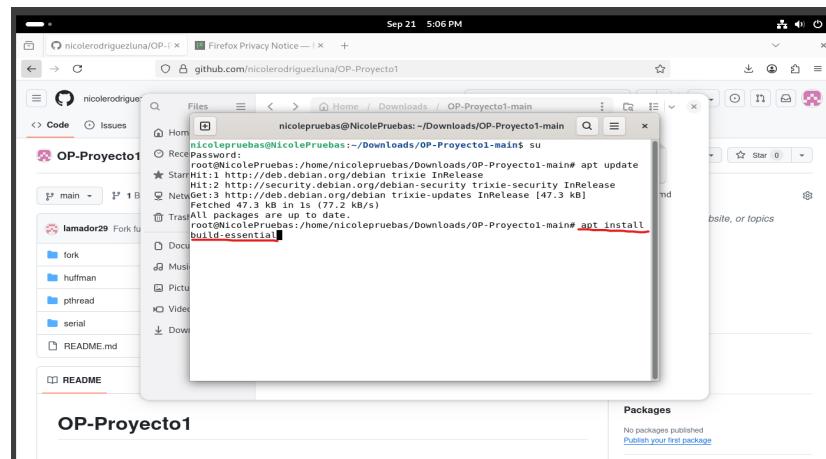
Se escribe el comando ‘‘apt update’’ y se presiona la tecla Enter. En caso de cualquier mensaje solicitando por espacio solo se acepta



A screenshot of a terminal window titled 'nicolerodriguezluna/OP-1'. The window shows a command-line interface where the user is typing 'apt update'. The terminal is running as root ('root@NicolePruebas:~'). The background shows a GitHub repository named 'OP-Proyecto1'.

Figura 2.19: Escribir comando ‘‘apt update’’.

Se escribe el comando ‘‘apt install build-essential’’ y se presiona la tecla Enter



A screenshot of a terminal window titled 'nicolerodriguezluna/OP-1'. The window shows a command-line interface where the user is typing 'apt install build-essential'. The terminal is running as root ('root@NicolePruebas:~'). The background shows a GitHub repository named 'OP-Proyecto1'. The output of the command shows that all packages are up to date.

Figura 2.20: Escribir comando ‘‘apt install build-essential’’.

Se acepta escribiendo "y" y presionando la tecla Enter

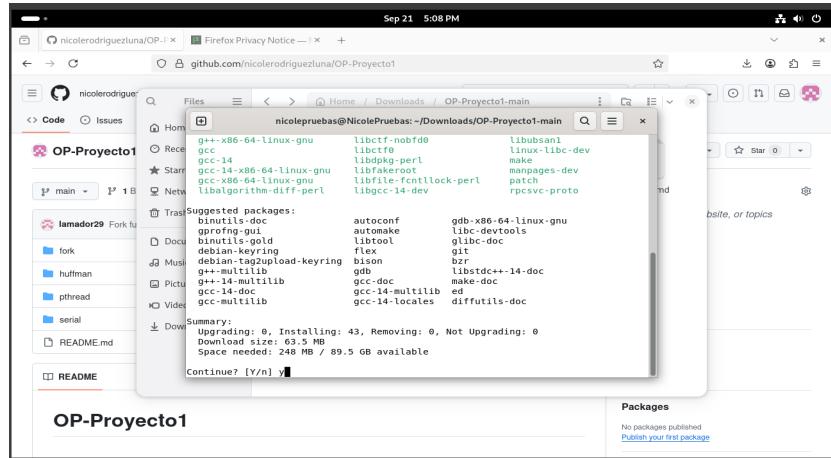


Figura 2.21: Aceptar escribiendo "y".

Se escribe "exit" y se presiona la tecla Enter

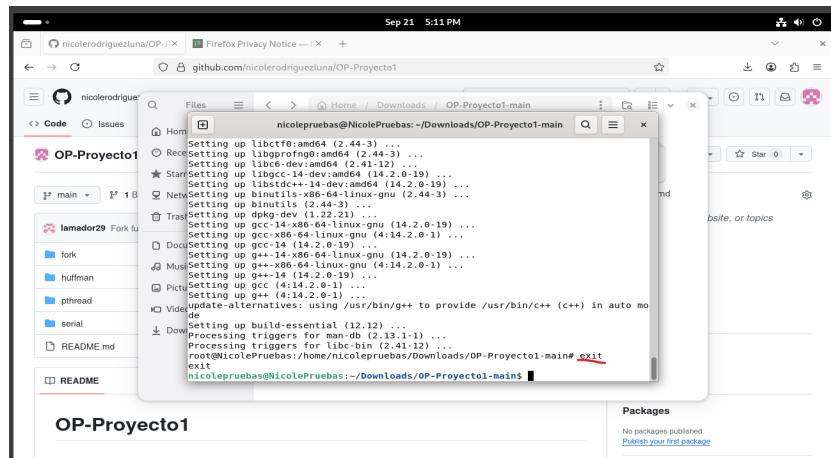
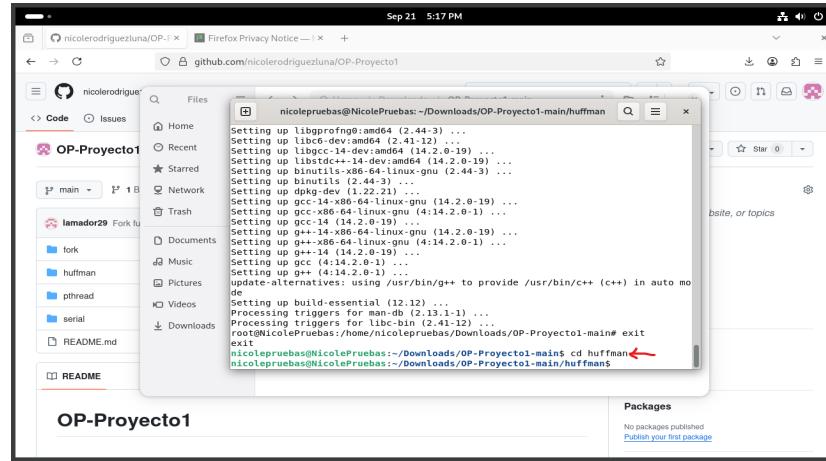


Figura 2.22: Salir del Root

Aquí termina la instalación del programa y todas sus dependencias, pero no cierre ninguna de las ventanas anteriores

Ahora, dependiendo de lo que desee probar habrá que hacer una u otra compilación

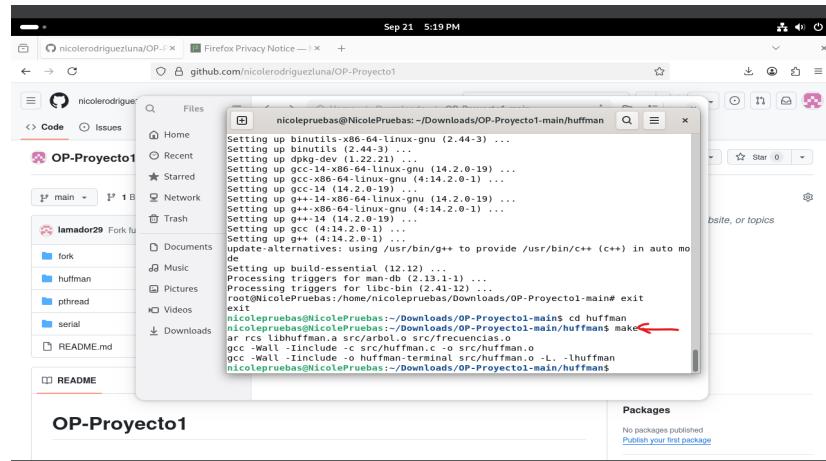
Primero, para probar el algoritmo de Huffman únicamente con el texto que esté en terminal primero debe usar el comando “cd huffman” y presionar la tecla Enter



```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$ cd huffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$
```

Figura 2.23: Navegar al Algoritmo de Huffman

Luego, para compilar debe usar el comando “make” y presionar la tecla Enter



```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$ make
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$
```

Figura 2.24: Compilación de Algoritmo de Huffman

Por último, para ejecutar debe usar el comando “./huffman-terminal” y presionar la tecla Enter

The screenshot shows a Firefox browser window with two tabs open: "nicolerodriguezluna/OP-Projecto1" and "github.com/nicolerodriguezluna/OP-Projecto1". The main content area displays a terminal session on a Linux system. The user has run several commands to build a C++ program named "huffman". The terminal output is as follows:

```
nicolepruebas@NicolePruebas:~/Downloads/OP-Projecto1-main$ g++ -Wall -Wextra -c huffman.c -o huffman.o
nicolepruebas@NicolePruebas:~/Downloads/OP-Projecto1-main$ cd huffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Projecto1-main/huffman$ make
gcc -Wall -Iinclude -c src/huffman.c -o src/huffman.o
gcc -Wall -Iinclude -o huffman-terminal src/huffman.o -L./huffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Projecto1-main/huffman$ ./huffman-terminal
Por favor, introduce el texto que deseas analizar:
```

The GitHub interface shows the repository structure on the left, including branches like "main" and "Branches", and files like "README.md", "fork", "huffman", "phread", "serial", and "VIDEOS". The terminal window has a title bar "nicolepruebas@NicolePruebas" and a status bar at the bottom right indicating "huffman" selected (containing 5 items). The GitHub sidebar on the right includes options to "Star" the repository.

Figura 2.25: Ejecución de Algoritmo de Huffman

Ahora, puede escribir el texto que desee y presionar la tecla Enter para que sea procesado

nicolerodriguezluna/OP-Proyecto1 · GitHub Privacy Notice — +

nicolerodriguezluna@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman

```
Setting up gcc-14-x86_64-linux-gnu (14.2.0-19) ...
Setting up g++-14-x86_64-linux-gnu (4:14.2.0-1) ...
Setting up g++-14 (14.2.0-19) ...
Setting up gcc (4:14.2.0-1) ...
Setting up g++ (4:14.2.0-1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.12) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for libc-bin (2.41.0-1) ...
nicolerodriguezluna@NicolePruebas:~/Downloads/OP-Proyecto1-main# exit
xterm[1]: killing...
nicolerodriguezluna@NicolePruebas:~/Downloads/OP-Proyecto1-main$ make
gcc -Wall -Iinclude -o huffman-terminal src/huffman.o -L.. -lhuffman
nicolerodriguezluna@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./huffman-terminal
Por favor, introduce el texto que deseas analizar:
En la mañana, un agil pingüino Albert, fénix del cielo, voló con Angel, una niña
que vivía en la cima de la montaña. Algunos días más tarde, Angel decidió visitar a sus
Amigos; se unieron a ellos Órián, Áñees y Edipo. Al final del día salió la luna,
y Albert se despidió de sus compañeros de viaje.
```

Figura 2.26: Escritura de Texto a Comprimir

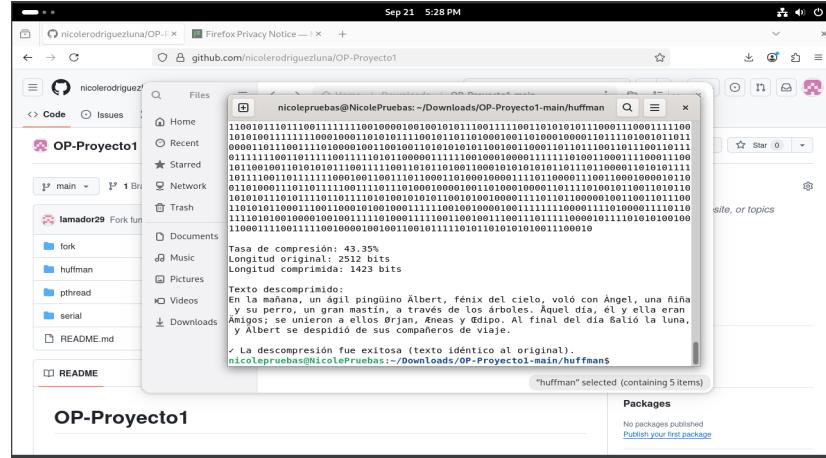


Figura 2.27: Texto Procesado

Por otro lado, para compilar la versión serial hay que ir a la carpeta serial, para devolverse a la carpeta principal puede usar el comando “cd ..”

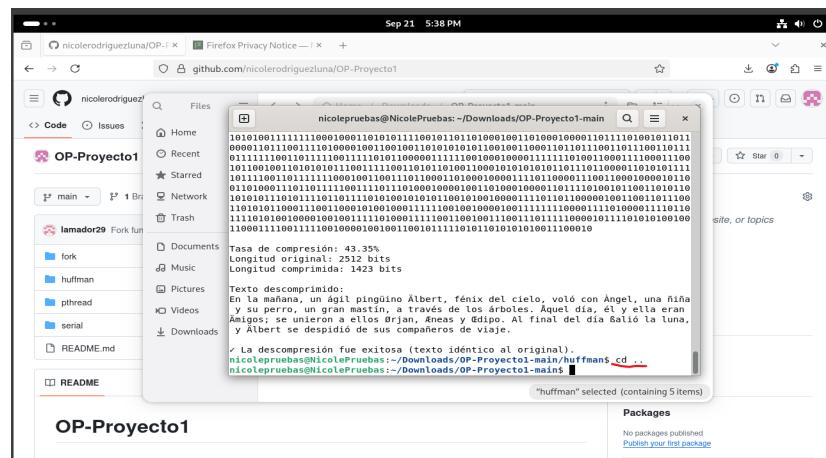


Figura 2.28: Retornar a la Carpeta Principal

Luego, debe usar el comando "cd serial"

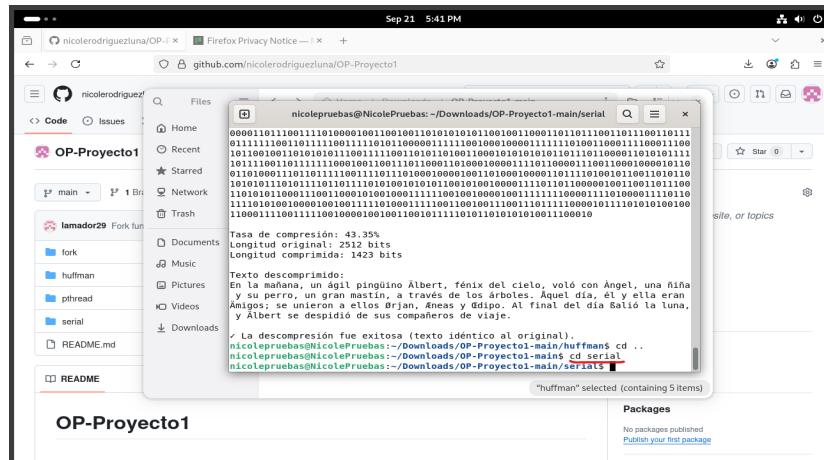


Figura 2.29: Navegar a Versión Fork

Después, para compilar debe usar el comando “make” y presionar la tecla Enter

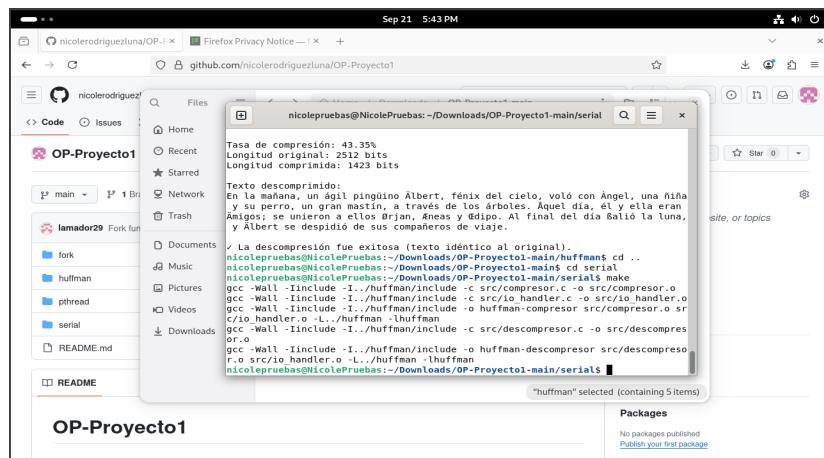


Figura 2.30: Compilación de Versión Serial

Ahora, volviendo al explorador de archivos, primero dele doble clic a la carpeta "serial"

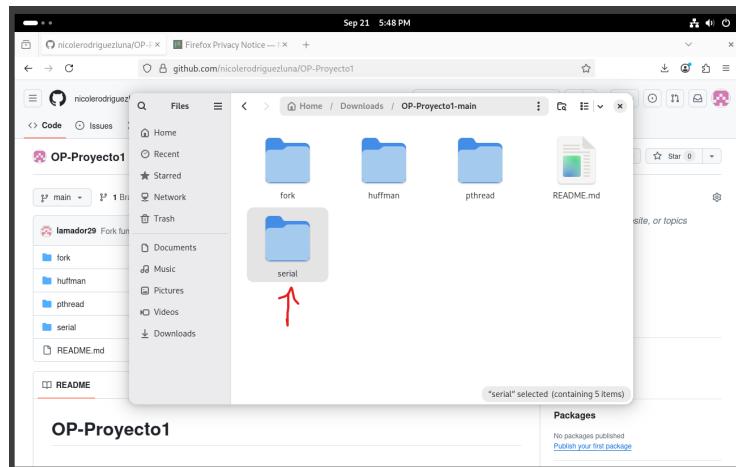


Figura 2.31: Entrar a la Carpeta Serial

Clic derecho a alguna sección en blanco de la ventana y clic a "New Folder"

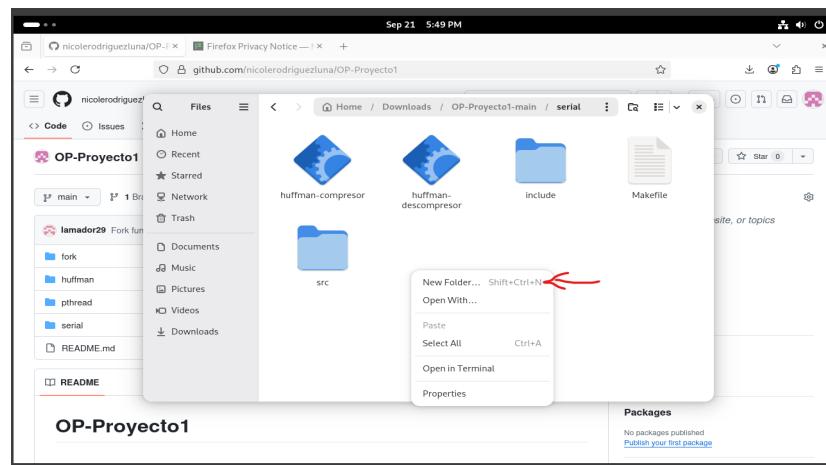


Figura 2.32: Creación de Carpeta

Escriba el nombre que desee para la carpeta a comprimir y clic a "Create"

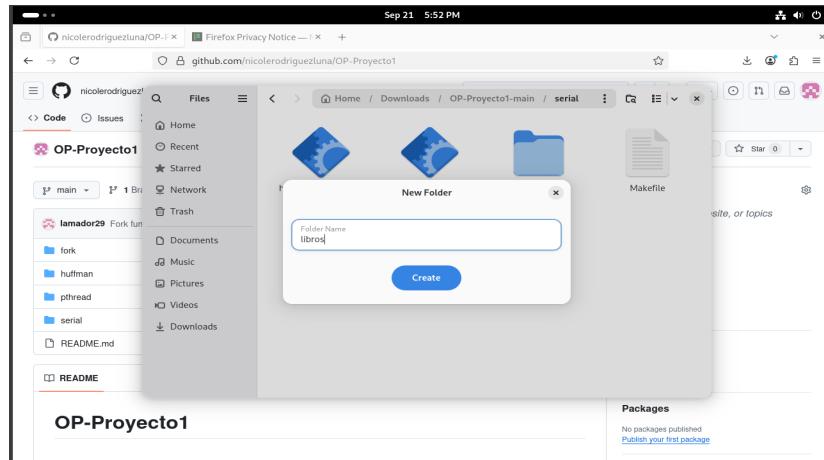


Figura 2.33: Nombrar Carpeta

Ya con la carpeta creada cree todos los archivos de texto que deseé procesar, para el ejemplo voy a traer 2 libros de <https://www.gutenberg.org/browse/scores/top#books-last30Links> entonces ingreso la dirección en una nueva pestaña de Firefox y presiono Enter

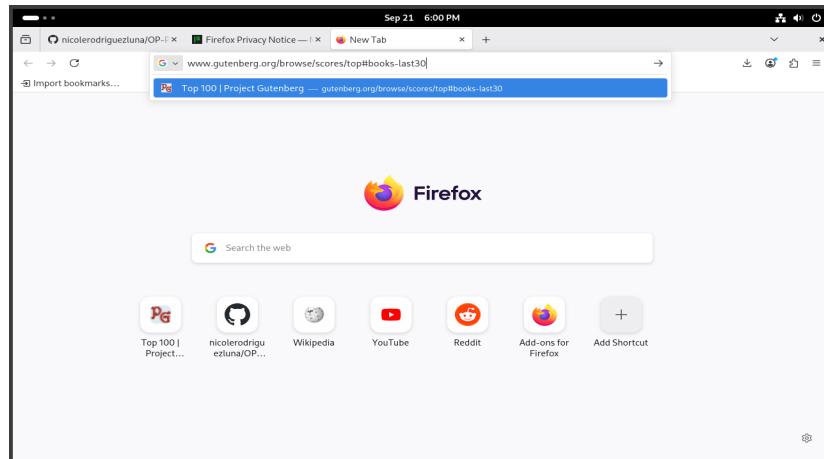


Figura 2.34: Ingreso de Dirección Web

## Selecciono Frankenstein

Sep 21 6:08 PM

nicolerodriguezluna/OP-1 Firefox Privacy Notice — Top 100 | Project Gutenberg

Project Gutenberg

About Frequently Downloaded Main Categories Reading Lists Search Options

Top 100 eBooks last 30 days

Rank	Title	Author	Downloads
1.	Frankenstein; Or, The Modern Prometheus	by Mary Wollstonecraft Shelley	(129228)
2.	Moby Dick; Or, The Whale	by Herman Melville	(124050)
3.	Romeo and Juliet	by William Shakespeare	(82917)
4.	Pride and Prejudice	by Jane Austen	(74206)
5.	Alice's Adventures in Wonderland	by Lewis Carroll	(64751)
6.	A Room with a View	by E. M. Forster	(57060)
7.	Beowulf: An Anglo-Saxon Epic Poem		(56799)
8.	The Complete Works of William Shakespeare	by William Shakespeare	(56479)
9.	Middlemarch	by George Eliot	(55562)
10.	Little Women; Or, Meg, Jo, Beth, and Amy	by Louisa May Alcott	(53903)
11.	Dracula	by Bram Stoker	(49595)
12.	The Strange Case of Dr. Jekyll and Mr. Hyde	by Robert Louis Stevenson	(49191)
13.	How to Observe Morals and Manners	by Harriet Martineau	(46844)
14.	The Blue Castle	by L. M. Montgomery	(45695)
15.	The Enchanted April	by Elizabeth Von Arnim	(45153)
16.	Clarissa	by Eliza Haywood	(40924)
17.	The Adventures of Ferdinand Count Fathom — Complete	by T. Smollett	(40500)
18.	Leviathan	by Thomas Hobbes	(40473)
19.	The Expedition of Humphry Clinker	by T. Smollett	(40329)
20.	Twenty years after	by Alexandre Dumas and Augusta Maquet	(40191)
21.	History of Tom Jones, a Foundling	by Henry Fielding	(39946)
22.	The Adventures of Roderick Random	by T. Smollett	(39424)

Figura 2.35: Selección de Libro

## Selecciono Plain Text UTF-8

Sep 21 6:10 PM

nicolerodriguezluna/OP-1 Firefox Privacy Notice — Frankenstein; Or, The Mo...

Project Gutenberg

About Frequently Downloaded Main Categories Reading Lists Search Options

Read or download for free

How to read	Size	Download
Read now!	457 kB	
EPUB3 (E-readers incl. Send-to-Kindle)	465 kB	
EPUB (older E-readers)	465 kB	
EPUB (no images, older E-readers)	350 kB	
Kindle	667 kB	
older Kindles	639 kB	
Plain Text UTF-8	438 kB	
Download HTML (.zip)	789 kB	

Similar Books

Figura 2.36: Selección de Formato de Libro

Clic derecho y clic a "Save Page As..."

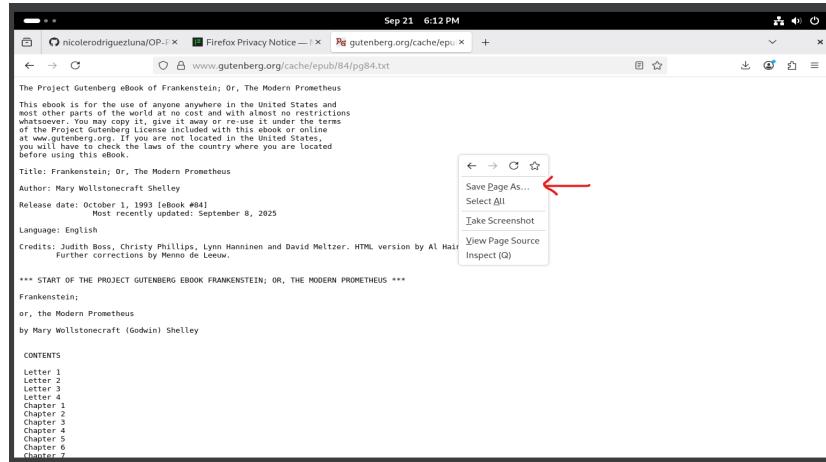


Figura 2.37: Descarga de Libro

Selecciono la carpeta que creé antes y clic a "Save"

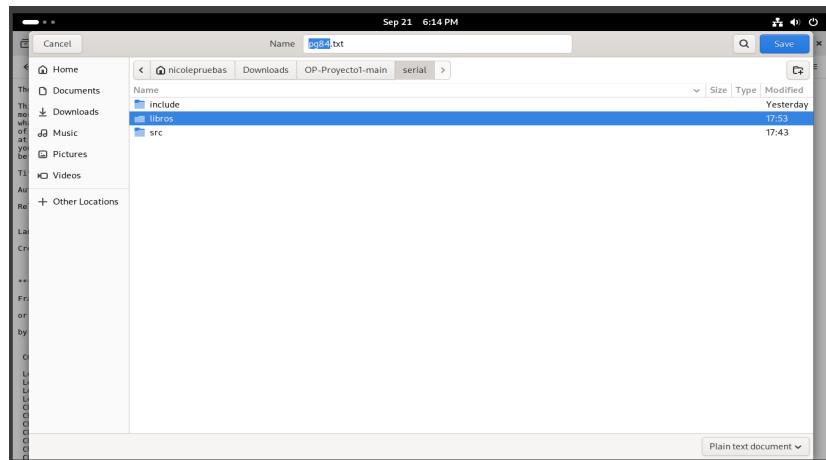


Figura 2.38: Selección de Carpeta y Guardado de Libro

Se repite el proceso las veces necesarias para conseguir todos los libros como textos en la carpeta.

Devuelta en la terminal, para comprimir se usa la estructura ‘‘./huffman-compresor directorio\_entrada archivo\_salida.huf’’, como se puede apreciar en la siguiente imagen.



```

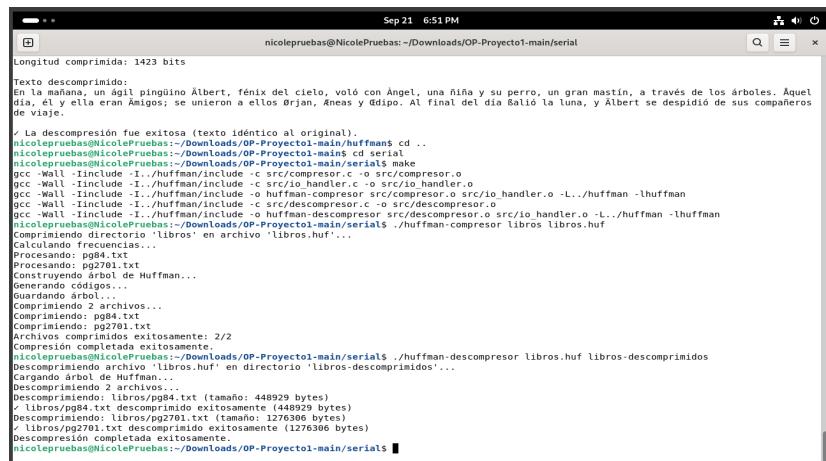
Sep 21 6:48 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compresor libro1 libro1.huf
Tasa de compresión: 43.3%
Longitud original: 2512 bits
Longitud comprimida: 1423 bits
Texto descomprimido:
En la mañana, un ágil pingüino Albert, fénix del cielo, voló con Ángel, una niña y su perro, un gran mastín, a través de los árboles. Aquel día, él y ella eran Amigos; se unieron a ellos Órjan, Áneas y Dípido. Al final del día salió la luna, y Albert se despidió de sus compañeros de viaje.

✓ La descompresión fue exitosa (texto idéntico al original).
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ serial$ make
gcc -Wall -Iinclude -L./huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -L./huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -L./huffman/include -o huffman-compresor src/compressor.o src/io_handler.o -L..../huffman -lhuffman
gcc -Wall -Iinclude -L./huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -L./huffman/include -o huffman-descomprimor src/descompressor.o src/io_handler.o -L..../huffman -lhuffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compresor libros libros.huf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árboles...
Comprimiendo 2 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ 

```

Figura 2.39: Compresión de Versión Serial

Por último, para descomprimir se usa la estructura ‘‘./huffman-descomprimor archivo\_entrada.huf directorio\_salida’’, como se puede apreciar en la siguiente imagen.



```

Sep 21 6:51 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprimor libros.huf libros-descomprimidos
Longitud comprimida: 1423 bits
Texto descomprimido:
En la mañana, un ágil pingüino Albert, fénix del cielo, voló con Ángel, una niña y su perro, un gran mastín, a través de los árboles. Aquel día, él y ella eran Amigos; se unieron a ellos Órjan, Áneas y Dípido. Al final del día salió la luna, y Albert se despidió de sus compañeros de viaje.

✓ La descompresión fue exitosa (texto idéntico al original).
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ serial$ make
gcc -Wall -Iinclude -L./huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -L./huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -L./huffman/include -o huffman-compresor src/compressor.o src/io_handler.o -L..../huffman -lhuffman
gcc -Wall -Iinclude -L./huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -L./huffman/include -o huffman-descomprimor src/descompressor.o src/io_handler.o -L..../huffman -lhuffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprimor src/io_handler.o src/descompressor.o src/compressor.o -L..../huffman -lhuffman
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árboles...
Comprimiendo 2 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprimor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Caso 1: Árbol de Huffman.
Descomprimiendo 2 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 449929 bytes)
Libros/pg84.txt descomprimido exitosamente (449929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
Libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descompresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ 

```

Figura 2.40: Descompresión de Versión Serial

Todo el proceso de compresión y descompresión se puede validar que sucedió en la carpeta serial.

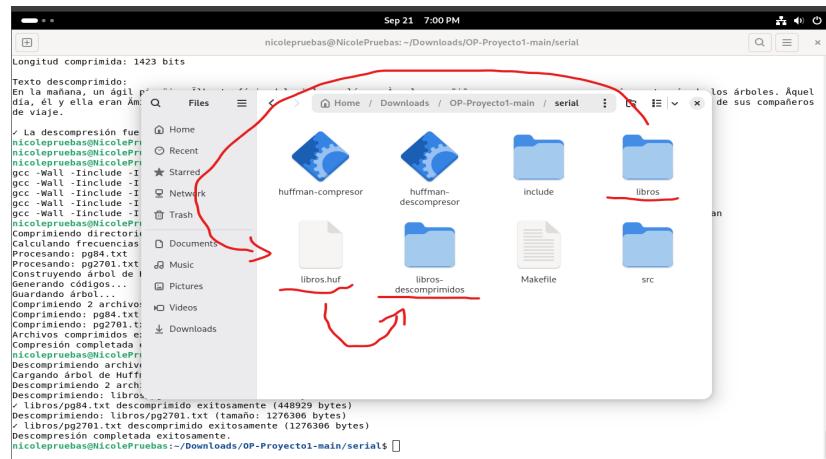


Figura 2.41: Validación de los Resultados de Versión Serial

Por otro lado, para compilar la versión fork() hay que ir a la carpeta fork, para devolverse a la carpeta principal puede usar el comando “cd ..”

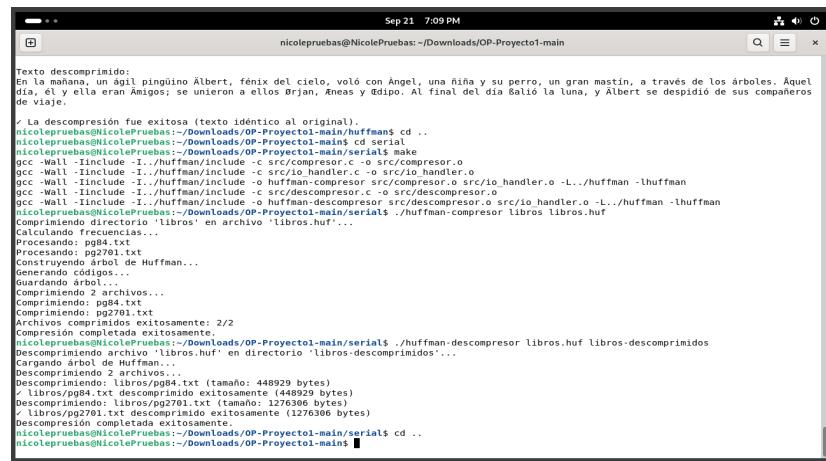


Figura 2.42: Retornar a la Carpeta Principal

Luego, debe usar el comando “cd fork”

```

Sep 21 7:10 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork

Texto descomprimido:
En la mañana, un ágil pingüino Álbert, fénix del cielo, voló con Ángel, una hína y su perro, un gran mastín, a través de los árboles. Aquel día, él y ella eran Amigos; se unieron a ellos Órjan, Áneas y Edipo. Al final del día salió la luna, y Albert se despidió de sus compañeros de viaje.

✓ La descompresión fue exitosa (texto idéntico al original).
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd serial
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ make
gcc -Wall -Iinclude -L./huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -L./huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -L./huffman/include -c src/compressor/src/compressor.o src/io_handler.o -L./huffman -lhuffman
gcc -Wall -Iinclude -L./huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -L./huffman/include -c src/descompressor/src/descompressor.o src/io_handler.o -L./huffman -lhuffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./huffman-compresor libros libros.huf
Comprimiendo directorio 'libros' en archivo libros.huf...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando Árboles...
Guardando Árboles...
Comprimiendo 2 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./huffman-descompresor libros.huf libros-descomprimidos
Descomprimiendo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo 2 archivos...
Descomprimiendo: pg84.txt (tamaño: 440929 bytes)
libros/pg84.txt descomprimido exitosamente (440929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descomprimiendo: libros-descomprimidos
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd fork
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ fork$ 

```

Figura 2.43: Navegar a Versión Fork

Después, para compilar debe usar el comando “make” y presionar la tecla Enter

```

Sep 21 7:19 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork

Texto descomprimido:
En la mañana, un ágil pingüino Álbert, fénix del cielo, voló con Ángel, una hína y su perro, un gran mastín, a través de los árboles. Aquel día, él y ella eran Amigos; se unieron a ellos Órjan, Áneas y Edipo. Al final del día salió la luna, y Albert se despidió de sus compañeros de viaje.

✓ La descompresión fue exitosa (texto idéntico al original).
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/huffman$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd serial
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ make
gcc -Wall -Iinclude -L./huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -L./huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -L./huffman/include -c src/compressor/src/compressor.o src/io_handler.o -L./huffman -lhuffman
gcc -Wall -Iinclude -L./huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -L./huffman/include -c src/descompressor/src/descompressor.o src/io_handler.o -L./huffman -lhuffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./huffman-compresor libros libros.huf
Comprimiendo directorio 'libros' en archivo libros.huf...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando Árboles...
Guardando Árboles...
Comprimiendo 2 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./huffman-descompresor libros.huf libros-descomprimidos
Descomprimiendo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo 2 archivos...
Descomprimiendo: pg84.txt (tamaño: 440929 bytes)
libros/pg84.txt descomprimido exitosamente (440929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descomprimiendo: libros-descomprimidos
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd fork
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ make
make: Nothing to be done for 'all'.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ forks$ 

```

Figura 2.44: Compilación de Versión Fork

Para comprimir debe usar el comando ‘‘./bin/huff\_compress\_fork’’ directorio’’ y presionar la tecla Enter

```

Sep 21 7:31 PM
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/fork

/ La descompresión fue exitosa (texto idéntico al original).
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/fork$ cd ..
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ cd serial
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./serial
gcc -Wall -Iinclude -I..huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -I..huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -I..huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -I..huffman/include -c src/io_handler.c -o src/io_handler.o
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ make
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./huffman-compressor libroshuf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árbol...
Comprimiendo 2 archivos...
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./huffman-descomprimor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 448929 bytes)
libros/pg84.txt descomprimido exitosamente (448929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descompresión completada exitosamente.
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ cd ..
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./fork
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ make
make: Nothing to be done for 'all'.
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_fork libros
[OK] Escrribi libros/archive.hfa con 2 archivos
Tiempo de escritura: 37 ms
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$
```

Figura 2.45: Compresión de Versión Fork

Y para descomprimir debe usar el comando ‘‘./bin/huff\_decompress\_fork’’ directorio’’ y presionar la tecla Enter

```

Sep 21 7:32 PM
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/fork

nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./serial
gcc -Wall -Iinclude -I..huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -I..huffman/include -c src/io_handler.c -o src/io_handler.o
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./descompressor
gcc -Wall -Iinclude -I..huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -I..huffman/include -c src/io_handler.c -o src/io_handler.o
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./huffman-compressor libroshuf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árbol...
Comprimiendo 2 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./huffman-descomprimor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo 2 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 448929 bytes)
libros/pg84.txt descomprimido exitosamente (448929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descompresión completada exitosamente.
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ cd ..
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./fork
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ make
make: Nothing to be done for 'all'.
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_fork libros
[OK] Escrribi libros/archive.hfa con 2 archivos
Tiempo de escritura: 37 ms
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_fork libros
[OK] Restauración libros (archivos: 2)
Tiempo de descompr: 37 ms
nicoolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main$
```

Figura 2.46: Descompresión de Versión Fork

Por otro lado, para compilar la versión pthread hay que ir a la carpeta pthread, para devolverse a la carpeta principal puede usar el comando "cd .."

```

Sep 21 7:39 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main
gcc -Wall -Iinclude -I..../huffman/include -c src/compressor.c -o src/compressor.o
gcc -Wall -Iinclude -I..../huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -I..../huffman/include -o huffman-compressor src/compressor.o src/io_handler.o -L..../huffman -lhuffman
gcc -Wall -Iinclude -I..../huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -I..../huffman/include -c src/io_handler.c -o src/io_handler.o -L..../huffman -lhuffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compressor libros libros.huf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árbol...
Guardando árbol...
Comprimiendo 2 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprimor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo 2 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 449929 bytes)
/ libros/pg84.txt descomprimido exitosamente (449929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
/ libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descompresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd fork
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ make
make: Nothing to be done for 'all'.
[OK] Escribi libros/archive.hfa con 2 archivos
Tiempo de compresión: 7749 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_fork libros
[OK] Restauración libros (archivos: 2)
[OK] Restauración libros (archivos: 2)
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ 
```

Figura 2.47: Retornar a la Carpeta Principal

Luego, debe usar el comando "cd pthread"

```

Sep 21 7:40 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread
gcc -Wall -Iinclude -I..../huffman/include -c src/io_handler.c -o src/io_handler.o
gcc -Wall -Iinclude -I..../huffman/include -o huffman-compressor src/compressor.o src/io_handler.o -L..../huffman -lhuffman
gcc -Wall -Iinclude -I..../huffman/include -c src/descompressor.c -o src/descompressor.o
gcc -Wall -Iinclude -I..../huffman/include -c src/io_handler.c -o src/io_handler.o -L..../huffman -lhuffman
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compressor libros libros.huf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árbol...
Guardando árbol...
Comprimiendo archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Archivos comprimidos exitosamente: 2/2
Compresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprimor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando Árbol de Huffman...
Descomprimiendo 2 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 449929 bytes)
/ libros/pg84.txt descomprimido exitosamente (449929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 1276306 bytes)
/ libros/pg2701.txt descomprimido exitosamente (1276306 bytes)
Descompresión completada exitosamente.
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd fork
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ make
make: Nothing to be done for 'all'.
[OK] Escribi libros/archive.hfa con 2 archivos
Tiempo de compresión: 7749 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_fork libros
[OK] Restauración libros (archivos: 2)
Tiempo de descompresión: 37 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ 
```

Figura 2.48: Navegar a Versión Pthread

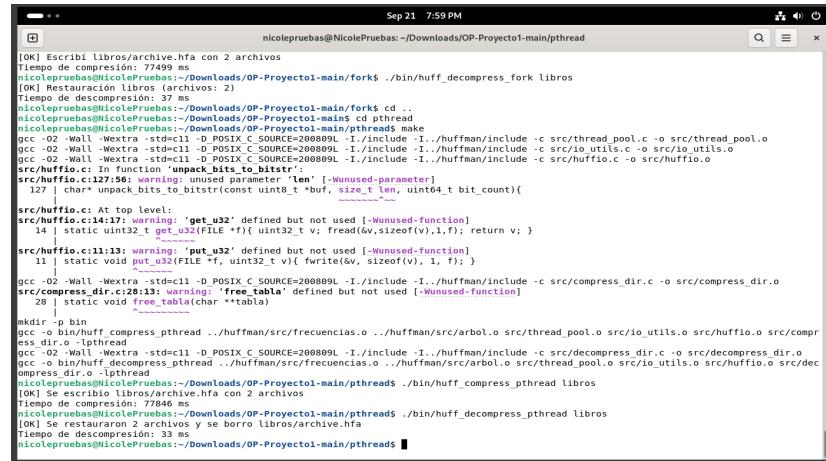
Después, para compilar debe usar el comando “make” y presionar la tecla Enter

Figura 2.49: Compilación de Versión Pthread

Para comprimir debe usar el comando ‘‘./bin/huff\_compress\_pthread’’ dentro del directorio y presionar la tecla Enter

Figura 2.50: Compresión de Versión Pthread

Y para descomprimir debe usar el comando ‘‘./bin/huff\_decompress\_pthread’’ en el directorio ‘‘y presionar la tecla Enter’’



```

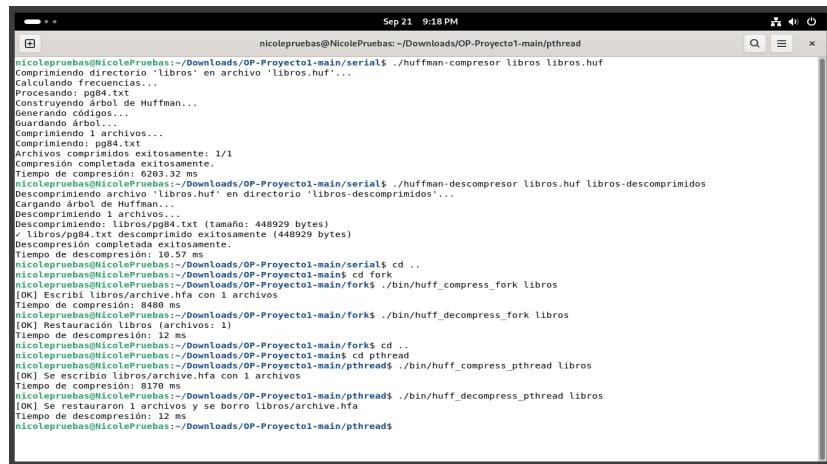
Sep 21 7:59 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthread
[0K] Escribi libros/archive.hfa con 2 archivos
Tiempo de compresión: 77499 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ ./bin/huff_decompress_fork libros
[0K] Restauración de libros (archivos: 2)
Tiempo de descompresión: 33 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthread$ make
gcc -O2 -Wall -Wextra -std=c11 -D POSIX_C_SOURCE=200809L -I./include -I.../huffman/include -c src/thread_pool.c -o src/thread_pool.o
gcc -O2 -Wall -Wextra -std=c11 -D POSIX_C_SOURCE=200809L -I./include -I.../huffman/include -c src/io_utils.c -o src/io_utils.o
gcc -O2 -Wall -Wextra -std=c11 -D POSIX_C_SOURCE=200809L -I./include -I.../huffman/include -c src/huffio.c -o src/huffio.o
src/huffio.c: In function 'decompress'
src/huffio.c:127:56: warning: unused parameter 'len' [-Wunused-parameter]
  127 |     char unpack_bits_to_bitstr(const uint8_t *buf, size_t len, uint64_t bit_count){
      |             ^
src/huffio.c: At top level:
src/huffio.c:14:17: warning: 'get_u32' defined but not used [-Wunused-function]
  14 | static uint32_t get_u32(FILE *f){ uint32_t v; fread(&v,sizeof(v),1,f); return v; }
src/huffio.c:11:13: warning: 'put_u32' defined but not used [-Wunused-function]
  11 | static void put_u32(FILE *f, uint32_t v){ fwrite(&v, sizeof(v), 1, f); }
gcc -O2 -Wall -Wextra -std=c11 -D POSIX_C_SOURCE=200809L -I./include -I.../huffman/include -c src/compress_dir.c -o src/compress_dir.o
src/compress_dir.c:28:13: warning: 'free_table' defined but not used [-Wunused-function]
  28 | static void free_table(char **tabla)
      |             ^
mkdir -p bin
gcc -o bin/huff_compress_pthread ..//huffman/src/frecuencias.o ..//huffman/src/arbol.o src/thread_pool.o src/io_utils.o src/huffio.o src/compr
ess_dir.o -lpthread
gcc -O2 -Wall -Wextra -std=c11 -D POSIX_C_SOURCE=200809L -I./include -I.../huffman/include -c src/decompress_dir.c -o src/decompress_dir.o
gcc -O2 -Wall -Wextra -std=c11 -D POSIX_C_SOURCE=200809L -I./include -I.../huffman/include ..//huffman/src/frecuencias.o ..//huffman/src/arbol.o src/thread_pool.o src/io_utils.o src/huffio.o src/dec
ompress_dir.o -lpthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthreads$ ./bin/huff_compress_pthread libros
[0K] Escribi libros/archive.hfa con 2 archivos
Tiempo de compresión: 77846 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthreads$ ./bin/huff_decompress_pthread libros
[0K] Se restauró libros y se borra libros/archive.hfa
Tiempo de descompresión: 33 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthreads$
```

Figura 2.51: Descompresión de Versión Pthread

## 2.4. Discusión

Se hará la comparación de los diferentes algoritmos al comprimir y descomprimir 1, 5, 10, 25, 50 y 98 libros.

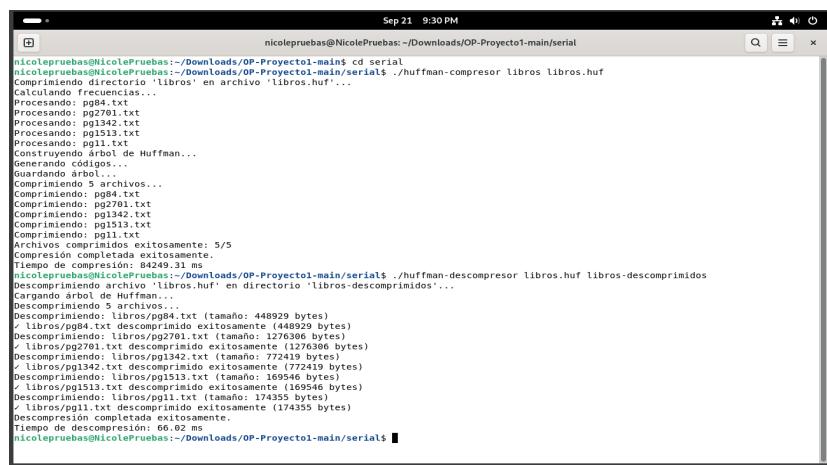
### Compresión y Descompresión de 1 libro



```
Sep 21 9:18 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compresor libros libros.huf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árbol...
Comprimiendo 1 archivos...
Comprimiendo: pg84.txt
Archivos comprimidos exitosamente: 1/1
Tiempo de compresión: 6263.32 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprisor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo 1 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 448929 bytes)
✓ libros/pg84.txt descomprimido exitosamente (448929 bytes)
Descompresión completada exitosamente.
Tiempo de descompresión: 10.57 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd fork
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ ./bin/huff_compress_fork libros
[OK] Escribi libros/archive.hfa con 1 archivos
Tiempo de compresión: 8170 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ ./bin/huff_decompress_fork libros
[OK] Restauración libros (archivos: 1)
Tiempo de descompresión: 12 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ ./bin/huff_compress_pthread libros
[OK] Se escribió libro archive.hfa con 1 archivos
Tiempo de compresión: 8170 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ ./bin/huff_decompress_pthread libros
[OK] Se restauraron 1 archivos y se borrar libros/archive.hfa
Tiempo de descompresión: 12 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$
```

Figura 2.52: Compresión y Descompresión de 5 Libros. Parte 1.

### Compresión y Descompresión de 5 libros



```
Sep 21 9:30 PM
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compresor libros libros.huf
Comprimiendo directorio 'libros' en archivo 'libros.huf'...
Calculando frecuencias...
Procesando: pg84.txt
Procesando: pg2701.txt
Procesando: pg1342.txt
Procesando: pg1513.txt
Procesando: pg11.txt
Construyendo árbol de Huffman...
Generando códigos...
Guardando árbol...
Comprimiendo 5 archivos...
Comprimiendo: pg84.txt
Comprimiendo: pg2701.txt
Comprimiendo: pg1342.txt
Comprimiendo: pg1513.txt
Comprimiendo: pg11.txt
Archivos comprimidos exitosamente: 5/5
Compresión completada exitosamente.
Tiempo de compresión: 84249.31 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descomprisor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Descomprimiendo 5 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 448929 bytes)
✓ libros/pg84.txt descomprimido exitosamente (448929 bytes)
Descomprimiendo: libros/pg2701.txt (tamaño: 127600 bytes)
✓ libros/pg2701.txt descomprimido exitosamente (127600 bytes)
Descomprimiendo: libros/pg1342.txt (tamaño: 137600 bytes)
✓ libros/pg1342.txt descomprimido exitosamente (137600 bytes)
Descomprimiendo: libros/pg1513.txt (tamaño: 772419 bytes)
✓ libros/pg1513.txt descomprimido exitosamente (772419 bytes)
Descomprimiendo: libros/pg11.txt (tamaño: 169546 bytes)
✓ libros/pg11.txt descomprimido exitosamente (169546 bytes)
Descomprimiendo: libros/pg1342.txt (tamaño: 174355 bytes)
✓ libros/pg1342.txt descomprimido exitosamente (174355 bytes)
Descompresión completada exitosamente.
Tiempo de descompresión: 66.02 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$
```

Figura 2.53: Compresión y Descompresión de 5 Libros. Parte 1.

```

Sep 21 9:39PM nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthread
Compriiendo: pg1342.txt
Comprimiendo: pg1311.txt
Comprimiendo: pg11.txt
Archivos comprimidos exitosamente: 5/5
Compresión completada exitosamente.
Tiempo de compresión: 430349.31 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descompresor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffmann...
Descomprimiendo 5 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 448929 bytes)
✓ libros/pg84.txt descomprimido exitosamente (448929 bytes)
Descomprimiendo: libros/pg781.txt (tamaño: 1276306 bytes)
✓ libros/pg781.txt descomprimido exitosamente (1276306 bytes)
Descomprimiendo: libros/pg1342.txt (tamaño: 772419 bytes)
✓ libros/pg1342.txt descomprimido exitosamente (772419 bytes)
Descomprimiendo: libros/pg1513.txt (tamaño: 169546 bytes)
✓ libros/pg1513.txt descomprimido exitosamente (169546 bytes)
Descomprimiendo: libros/pg11.txt (tamaño: 174355 bytes)
✓ libros/pg11.txt descomprimido exitosamente (174355 bytes)
Descompresión completada exitosamente.
Tiempo de descompresión: 66.02 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_fork libros
[OK] Escribi libros/archive.hfa con 5 archivos
Tiempo de compresión: 929.03 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_fork libros
[OK] Restauración libros (archivos: 5)
Tiempo de descompresión: 36 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_pthread libros
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_pthread libros
[OK] Se escribió libros/archive.hfa con 5 archivos
Tiempo de compresión: 36 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_pthread libros
[OK] Se restauraron 5 archivos y se borrar libros/archive.hfa
Tiempo de descompresión: 38 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ pthreads

```

Figura 2.54: Compresión y Descompresión de 5 Libros. Parte 2.

### Compresión y Descompresión de 10 libros

```

Sep 21 11:09PM nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial
Compriiendo: pg2691.txt
Comprimiendo: pg2691.txt
Comprimiendo: pg7108.txt
Comprimiendo: pg16328.txt
Comprimiendo: pg1342.txt
Comprimiendo: pg1513.txt
Comprimiendo: pg145.txt
Comprimiendo: pg100.txt
Comprimiendo: pg11.txt
Archivos comprimidos exitosamente: 10/10
Compresión completada exitosamente.
Tiempo de compresión: 430347.17 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descompresor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffmann...
Descomprimiendo 10 archivos...
Descomprimiendo: libros/pg84.txt (tamaño: 448929 bytes)
✓ libros/pg84.txt descomprimido exitosamente (448929 bytes)
Descomprimiendo: libros/pg2641.txt (tamaño: 415864 bytes)
✓ libros/pg2641.txt descomprimido exitosamente (415864 bytes)
Descomprimiendo: libros/pg781.txt (tamaño: 1276306 bytes)
✓ libros/pg781.txt descomprimido exitosamente (1276306 bytes)
Descomprimiendo: libros/pg37106.txt (tamaño: 113473 bytes)
✓ libros/pg37106.txt descomprimido exitosamente (113473 bytes)
Descomprimiendo: libros/pg16328.txt (tamaño: 308093 bytes)
✓ libros/pg16328.txt descomprimido exitosamente (308093 bytes)
Descomprimiendo: libros/pg1342.txt (tamaño: 772419 bytes)
✓ libros/pg1342.txt descomprimido exitosamente (772419 bytes)
Descomprimiendo: libros/pg1513.txt (tamaño: 169546 bytes)
✓ libros/pg1513.txt descomprimido exitosamente (169546 bytes)
Descomprimiendo: libros/pg145.txt (tamaño: 1065732 bytes)
✓ libros/pg145.txt descomprimido exitosamente (1065732 bytes)
Descomprimiendo: libros/pg100.txt (tamaño: 5638525 bytes)
✓ libros/pg100.txt descomprimido exitosamente (5638525 bytes)
Descomprimiendo: libros/pg11.txt (tamaño: 174355 bytes)
✓ libros/pg11.txt descomprimido exitosamente (174355 bytes)
Descompresión completada exitosamente.
Tiempo de descompresión: 272.82 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ 

```

Figura 2.55: Compresión y Descompresión de 10 Libros. Parte 1.

```
Sep 22 1:52 AM nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthread
```

Descomprimiendo: libros/pg84.txt (tamaño: 448829 bytes)  
✓ libros/pg84.txt descomprimido exitosamente (448829 bytes)

Descomprimiendo: libros/pg2641.txt (tamaño: 415864 bytes)  
✓ libros/pg2641.txt descomprimido exitosamente (415864 bytes)

Descomprimiendo: libros/pg2701.txt (tamaño: 1276396 bytes)  
✓ libros/pg2701.txt descomprimido exitosamente (1276396 bytes)

Descomprimiendo: libros/pg3186.txt (tamaño: 1113473 bytes)  
✓ libros/pg3186.txt descomprimido exitosamente (1113473 bytes)

Descomprimiendo: libros/pg16328.txt (tamaño: 380503 bytes)  
✓ libros/pg16328.txt descomprimido exitosamente (380503 bytes)

Descomprimiendo: libros/pg16328.txt (tamaño: 772419 bytes)  
✓ libros/pg16328.txt descomprimido exitosamente (772419 bytes)

Descomprimiendo: libros/pg1342.txt (tamaño: 109546 bytes)  
✓ libros/pg1342.txt descomprimido exitosamente (109546 bytes)

Descomprimiendo: libros/pg1513.txt (tamaño: 1865732 bytes)  
✓ libros/pg1513.txt descomprimido exitosamente (1865732 bytes)

Descomprimiendo: libros/pg108.txt (tamaño: 563825 bytes)  
✓ libros/pg108.txt descomprimido exitosamente (563825 bytes)

Descomprimiendo: libros/pg11.txt (tamaño: 174355 bytes)  
✓ libros/pg11.txt descomprimido exitosamente (174355 bytes)

Tiempo de descompresión: 272.82 ms

```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serials cd ..  
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serials cd fork  
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/forks ./bin/huff_compress_fork libros  
[OK] Escribi libros/archive.hfa con 10 archivos
```

Tiempo de compresión: 4838376 ms

```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/forks ./bin/huff_decompress_fork libros  
[OK] Restauración libros (archivos: 10)
```

Tiempo de descompresión: 148 ms

```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/forks cd ..  
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serials cd pthread  
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread) ./bin/huff_compress_pthread libros  
[OK] Se escribió libro libros/archive.hfa con 10 archivos
```

Tiempo de compresión: 490592 ms

```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread) ./bin/huff_decompress_pthread libros  
[OK] Se restauraron 10 archivos y se borro libros/archive.hfa
```

Tiempo de descompresión: 16 ms

```
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread) [ ]
```

Figura 2.56: Compresión y Descompresión de 10 Libros. Parte 2.

## Compresión y Descompresión de 25 libros

```
Sept 22 12:16 PM  
nicolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/pthread  
  
Procesando: pg11.txt  
Generando árbol de Huffmann...  
Guardando árbol...  
Comprimiendo 25 archivos...  
Comprimiendo: pg1200.txt  
Comprimiendo: pg1260.txt  
Comprimiendo: pg67999.txt  
Comprimiendo: pg33944.txt  
Comprimiendo: pg11000.txt  
Comprimiendo: pg844.txt  
Comprimiendo: pg1259.txt  
Comprimiendo: pg16389.txt  
Comprimiendo: pg11009.txt  
Comprimiendo: pg2641.txt  
Comprimiendo: pg2701.txt  
Comprimiendo: pg3700.txt  
Comprimiendo: pg17109.txt  
Comprimiendo: pg16328.txt  
Comprimiendo: pg1342.txt  
Comprimiendo: pg11001.txt  
Comprimiendo: pg945.txt  
Comprimiendo: pg943.txt  
Comprimiendo: pg1513.txt  
Comprimiendo: pg11002.txt  
Comprimiendo: pg9485.txt  
Comprimiendo: pg1900.txt  
Comprimiendo: pg6761.txt  
Comprimiendo: pg11111.txt  
Comprimiendo: pg11.txt  
Archivos comprimidos exitosamente: 25/25  
Compresión completada al 100%  
[Process completed] 4785949.47 ms  
nicolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serials ./huffman-descomprisor libros.huf libros-descomprimidos  
Descomprimiendo archivo: 'libros.huf' en directorio 'libros-descomprimidos'...  
Cargando árbol de Huffmann...  
Descomprimiendo archivos...  
Descomprimiendo: libros/pg3207.txt (tamaño: 1254981 bytes)  
/ libros/pg3207.txt descomprimido exitosamente (1254981 bytes)  
Descomprimiendo: libros/pg1260.txt (tamaño: 1084843 bytes)
```

Figura 2.57: Compresión y Descompresión de 25 Libros. Parte 1.

```

Sep 22 12:23 PM nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/pthread
[+] Descomprimiendo: libros/pg1161.txt (tamaño: 89456 bytes)
[+] Descomprimiendo: libros/pg180.txt descomprimido exitosamente (88120 bytes)
Descomprimiendo: libros/pg345.txt (tamaño: 89038 bytes)
Descomprimiendo: libros/pg433.txt descomprimido exitosamente (890394 bytes)
Descomprimiendo: libros/pg433.txt (tamaño: 103520 bytes)
[+] Descomprimiendo: libros/pg433.txt descomprimido exitosamente (890394 bytes)
Descomprimiendo: libros/pg113.txt (tamaño: 169546 bytes)
[+] Descomprimiendo: libros/pg113.txt descomprimido exitosamente (169546 bytes)
Descomprimiendo: libros/pg113.txt (tamaño: 186512 bytes)
[+] Descomprimiendo: libros/pg113.txt descomprimido exitosamente (186512 bytes)
Descomprimiendo: libros/pg4985.txt (tamaño: 119354 bytes)
libros/pg4985.txt descomprimido exitosamente (119354 bytes)
Descomprimiendo: libros/pg100.txt (tamaño: 1563825 bytes)
[+] Descomprimiendo: libros/pg100.txt descomprimido exitosamente (1563825 bytes)
Descomprimiendo: libros/pg761.txt (tamaño: 994557 bytes)
[+] Descomprimiendo: libros/pg761.txt descomprimido exitosamente (994557 bytes)
Descomprimiendo: libros/pg761.txt (tamaño: 2034104 bytes)
[+] Descomprimiendo: libros/pg111.txt descomprimido exitosamente (2034104 bytes)
Descomprimiendo: libros/pg111.txt (tamaño: 174355 bytes)
[+] Descomprimiendo: libros/pg111.txt descomprimido exitosamente (174355 bytes)
Descompreision completada exitosamente
Tiempo de descompresion: 556.64 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd fork
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ ./bin/huff_compress_fork libros
[OK] Escribi libros/archive.hfa con 25 archivos
Tiempo de compresion: 52800 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ ./bin/huff_decompress_fork libros
[OK] Restauracion libros (archivos: 25)
Tiempo de descompresion: 281 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ ./bin/huff_compress_pthread libros
[OK] Se establecio archivo archive.hfa con 25 archivos
Tiempo de compresion: 52800 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ ./bin/huff_decompress_pthread libros
[OK] Se restauraron 25 archivos y se borro libros/archive.hfa
Tiempo de descompresion: 211 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$
```

Figura 2.58: Compresión y Descompresión de 25 Libros. Parte 2.

### Compresión y Descompresión de 50 libros

```

Sep 22 5:59 PM nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial
[+] [OK] Escribi libros/archive.hfa con 50 archivos
Tiempo de compresion: 610887 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ ./bin/huff_decompress_fork libros
[OK] Restauracion libros (archivos: 50)
Tiempo de descompresion: 256 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/fork$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd pthread
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ ./bin/huff_compress_pthread libros
[OK] Se establecio archivo archive.hfa con 50 archivos
Tiempo de compresion: 5921647 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ ./bin/huff_decompress_pthread libros
[OK] Se restauraron 50 archivos y se borro libros/archive.hfa
Tiempo de descompresion: 289 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main(pthread$ cd ..
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ cd serial
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-compresor libros.libros.huf
Calculando frecuencias...
Procesando: pg3297.txt
Procesando: pg3297.txt
Procesando: pg56517.txt
Procesando: pg56517.txt
Procesando: pg67979.txt
Procesando: pg67979.txt
Procesando: pg2591.txt
Procesando: pg2591.txt
Procesando: pg52621.txt
Procesando: pg52621.txt
Procesando: pg7370.txt
Procesando: pg7370.txt
Procesando: pg84.txt
Procesando: pg84.txt
Procesando: pg98.txt
Procesando: pg98.txt
Procesando: pg16389.txt
Procesando: pg16389.txt
Procesando: pg1080.txt
Procesando: pg1080.txt
Procesando: pg174.txt
Procesando: pg174.txt
Procesando: pg2641.txt
Procesando: pg2641.txt
Procesando: pg2781.txt
Procesando: pg2781.txt
Procesando: pg37108.txt
Procesando: pg37108.txt
Procesando: pg64317.txt
Procesando: pg64317.txt
```

Figura 2.59: Compresión y Descompresión de 50 Libros. Parte 1.

```

Sep 22 6:05 PM nicolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serial
Comprimiendo: pg3554.txt
Comprimiendo: pg5280.txt
Comprimiendo: pg26184.txt
Comprimiendo: pg408.txt
Comprimiendo: pg1952.txt
Comprimiendo: pg1998.txt
Comprimiendo: pg844.txt
Comprimiendo: pg76.txt
Comprimiendo: pg11497.txt
Comprimiendo: pg1080.txt
Comprimiendo: pg0130.txt
Comprimiendo: pg1497.txt
Comprimiendo: pg1260.txt
Comprimiendo: pg25344.txt
Comprimiendo: pg2148.txt
Comprimiendo: pg1998.txt
Comprimiendo: pg0593.txt
Comprimiendo: pg2542.txt
Comprimiendo: pg11.txt
Archivos comprimidos exitosamente: 50/50
Compresión completa en 0:00:00.20 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ ./huffman-descompresor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Característica de los archivos comprimidos:
Descomprimiendo 50 archivos...
Descomprimiendo: libros/pg3207.txt (tamaño: 1254981 bytes)
✓ libros/pg3207.txt descomprimido exitosamente (1254981 bytes)
Descomprimiendo: libros/pg1260.txt (tamaño: 1089481 bytes)
✓ libros/pg1260.txt descomprimido exitosamente (1089481 bytes)
Descomprimiendo: libros/pg050517.txt (tamaño: 704748 bytes)
✓ libros/pg050517.txt descomprimido exitosamente (704748 bytes)
Descomprimiendo: libros/pg050517.txt (tamaño: 526481 bytes)
✓ libros/pg050517.txt descomprimido exitosamente (526481 bytes)
Descomprimiendo: libros/pg05979.txt (tamaño: 428245 bytes)
✓ libros/pg05979.txt descomprimido exitosamente (428245 bytes)
Descomprimiendo: libros/pg33944.txt (tamaño: 417851 bytes)
Descomprimiendo: libros/pg33944.txt (tamaño: 389525 bytes)
✓ libros/pg33944.txt descomprimido exitosamente (389525 bytes)
Descomprimiendo: libros/pg2591.txt (tamaño: 560207 bytes)
✓ libros/pg2591.txt descomprimido exitosamente (560207 bytes)

```

Figura 2.60: Compresión y Descompresión de 50 Libros. Parte 2.

```

Sep 22 6:05 PM nicolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serial
Descomprimiendo: libros/pg408.txt (tamaño: 428788 bytes)
✓ libros/pg408.txt descomprimido exitosamente (428788 bytes)
Descomprimiendo: libros/pg1952.txt (tamaño: 526481 bytes)
✓ libros/pg1952.txt descomprimido exitosamente (526481 bytes)
Descomprimiendo: libros/pg145.txt (tamaño: 1865732 bytes)
✓ libros/pg145.txt descomprimido exitosamente (1865732 bytes)
Descomprimiendo: libros/pg05044.txt (tamaño: 141801 bytes)
✓ libros/pg05044.txt descomprimido exitosamente (141801 bytes)
Descomprimiendo: libros/pg76.txt (tamaño: 622594 bytes)
✓ libros/pg76.txt descomprimido exitosamente (622594 bytes)
Descomprimiendo: libros/pg1998.txt (tamaño: 622594 bytes)
✓ libros/pg1998.txt descomprimido exitosamente (622594 bytes)
Descomprimiendo: libros/pg0108.txt (tamaño: 5638525 bytes)
✓ libros/pg1080.txt descomprimido exitosamente (5638525 bytes)
Descomprimiendo: libros/pg0593.txt (tamaño: 1161405 bytes)
✓ libros/pg0593.txt descomprimido exitosamente (1161405 bytes)
Descomprimiendo: libros/pg1497.txt (tamaño: 1244214 bytes)
✓ libros/pg1497.txt descomprimido exitosamente (1244214 bytes)
Descomprimiendo: libros/pg3296.txt (tamaño: 632216 bytes)
✓ libros/pg3296.txt descomprimido exitosamente (632216 bytes)
Descomprimiendo: libros/pg25344.txt (tamaño: 526365 bytes)
✓ libros/pg25344.txt descomprimido exitosamente (526365 bytes)
Descomprimiendo: libros/pg2148.txt (tamaño: 638516 bytes)
✓ libros/pg2148.txt descomprimido exitosamente (638516 bytes)
Descomprimiendo: libros/pg0761.txt (tamaño: 99471 bytes)
✓ libros/pg0761.txt descomprimido exitosamente (99471 bytes)
Descomprimiendo: libros/pg1998.txt (tamaño: 682965 bytes)
✓ libros/pg1998.txt descomprimido exitosamente (682965 bytes)
Descomprimiendo: libros/pg0593.txt (tamaño: 2034104 bytes)
✓ libros/pg0593.txt descomprimido exitosamente (2034104 bytes)
Descomprimiendo: libros/pg2542.txt (tamaño: 169083 bytes)
✓ libros/pg2542.txt descomprimido exitosamente (169083 bytes)
Descomprimiendo: libros/pg1232.txt (tamaño: 308864 bytes)
✓ libros/pg1232.txt descomprimido exitosamente (308864 bytes)
Descomprimiendo: libros/pg11.txt (tamaño: 174355 bytes)
✓ libros/pg11.txt descomprimido exitosamente (174355 bytes)
Tiempo de descompresión: 945.13 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main/serial$ 

```

Figura 2.61: Compresión y Descompresión de 50 Libros. Parte 3.

Compresión y Descompresión de 98 libros

```
Sep 29 09:23AM

nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_fork libros
[OK] Escribi libros/archive.hfa con 98 archivos
Tiempo de compresión: 944512 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_fork libros
[OK] Restauración libros (archivos: 98)
Tiempo de descompresión: 712 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_compress_pthread libros
[OK] Se escribió libros/archive.hfa con 98 archivos
Tiempo de compresión: 445 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_pthread libros
[OK] Se restauraron 98 archivos y se borro libros/archive.hfa
Tiempo de descompresión: 445 ms
nicolepruebas@NicolePruebas:~/Downloads/OP-Proyecto1-main$ ./bin/huff_decompress_pthread libros
[OK] Restauración libros (archivos: 98)
Calculando frecuencias...
Procesando: pg3267.txt
Procesando: pg1224.txt
Procesando: pg1225.txt
Procesando: pg1226.txt
Procesando: pg36.txt
Procesando: pg12.txt
Procesando: pg1230.txt
Procesando: pg36517.txt
Procesando: pg67579.txt
Procesando: pg2291.txt
Procesando: pg2291.txt
Procesando: pg41445.txt
Procesando: pg1251.txt
Procesando: pg1252.txt
Procesando: pg4391.txt
Procesando: pg55.txt
Procesando: pg2680.txt
Procesando: pg73790.txt
Procesando: pg73791.txt
Procesando: pg73792.txt
Procesando: pg1727.txt
```

Figura 2.62: Compresión y Descompresión de 98 Libros. Parte 1.

```
Sep 23 9:13 AM nicolepruebass@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serial

Comprimiendo: pg1138.txt
Comprimiendo: pg2542.txt
Comprimiendo: pg41.txt
Comprimiendo: pg1184.txt
Comprimiendo: pg1185.txt
Comprimiendo: pg34991.txt
Comprimiendo: pg4363.txt
Comprimiendo: pg1186.txt
Comprimiendo: pg2852.txt
Comprimiendo: pg11.txt
Comprimiendo: pg1187.txt
Archivos comprimidos exitosamente: 98/98
Compresión completada exitosamente.
Tiempo de compresión: 10355328.37 ms
nicolepruebass@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serials ./huffman-descompresor libros.huf libros-descomprimidos
Descomprimiendo archivo 'libros.huf' en directorio 'libros-descomprimidos'...
Cargando árbol de Huffman...
Arbol de Huffman cargado.
Descomprimiendo: libros/pg3207.txt (tamaño: 1254981 bytes)
  /libros/pg3207.txt descomprimido exitosamente (1254981 bytes)
Descomprimiendo: libros/pg1228.txt (tamaño: 978604 bytes)
  /libros/pg1228.txt descomprimido exitosamente (978604 bytes)
Descomprimiendo: libros/pg37683.txt (tamaño: 2988053 bytes)
  /libros/pg37683.txt descomprimido exitosamente (2988053 bytes)
Descomprimiendo: libros/pg1210.txt (tamaño: 196575 bytes)
  /libros/pg1210.txt descomprimido exitosamente (196575 bytes)
Descomprimiendo: libros/pg1260.txt (tamaño: 1884843 bytes)
  /libros/pg1260.txt descomprimido exitosamente (1884843 bytes)
Descomprimiendo: libros/pg58517.txt (tamaño: 704749 bytes)
  /libros/pg58517.txt descomprimido exitosamente (704749 bytes)
Descomprimiendo: libros/pg67979.txt (tamaño: 428245 bytes)
  /libros/pg67979.txt descomprimido exitosamente (428245 bytes)
Descomprimiendo: libros/pg33944.txt (tamaño: 417855 bytes)
  /libros/pg33944.txt descomprimido exitosamente (417855 bytes)
Descomprimiendo: libros/pg2591.txt (tamaño: 560287 bytes)
  /libros/pg2591.txt descomprimido exitosamente (560287 bytes)
Descomprimiendo: libros/pg1445.txt (tamaño: 438153 bytes)
  /libros/pg1445.txt descomprimido exitosamente (438153 bytes)
```

Figura 2.63: Compresión y Descompresión de 98 Libros. Parte 2.

```

Sep 23 09:14 AM
nicolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serial
Descomprimiendo: libros/pg8800.txt (tamaño: 656776 bytes)
✓ libros/pg1000.txt descomprimido exitosamente (656776 bytes)
Descomprimiendo: libros/pg1001.txt (tamaño: 208000 bytes)
✓ libros/pg100.txt descomprimido exitosamente (288813 bytes)
Descomprimiendo: libros/pg28054.txt (tamaño: 2042003 bytes)
✓ libros/pg28054.txt descomprimido exitosamente (2042003 bytes)
Descomprimiendo: libros/pg2148.txt (tamaño: 438516 bytes)
✓ libros/pg2148.txt descomprimido exitosamente (438516 bytes)
Descomprimiendo: libros/pg6761.txt (tamaño: 994557 bytes)
✓ libros/pg6761.txt descomprimido exitosamente (994557 bytes)
Descomprimiendo: libros/pg998.txt (tamaño: 682965 bytes)
✓ libros/pg998.txt descomprimido exitosamente (682965 bytes)
Descomprimiendo: libros/pg6931.txt (tamaño: 983951 bytes)
✓ libros/pg6931.txt descomprimido exitosamente (983951 bytes)
Descomprimiendo: libros/pg135.txt (tamaño: 3369259 bytes)
✓ libros/pg135.txt descomprimido exitosamente (3369259 bytes)
Descomprimiendo: libros/pg34901.txt (tamaño: 4099018 bytes)
✓ libros/pg34901.txt descomprimido exitosamente (4099018 bytes)
Descomprimiendo: libros/pg2542.txt (tamaño: 169983 bytes)
✓ libros/pg2542.txt descomprimido exitosamente (169983 bytes)
Descomprimiendo: libros/pg41.txt (tamaño: 99938 bytes)
✓ libros/pg41.txt descomprimido exitosamente (99938 bytes)
Descomprimiendo: libros/pg184.txt (tamaño: 2787166 bytes)
✓ libros/pg184.txt descomprimido exitosamente (2787166 bytes)
Descomprimiendo: libros/pg1480.txt (tamaño: 1058246 bytes)
✓ libros/pg1480.txt descomprimido exitosamente (1058246 bytes)
Descomprimiendo: libros/pg34901.txt (tamaño: 331200 bytes)
✓ libros/pg34901.txt descomprimido exitosamente (331200 bytes)
Descomprimiendo: libros/pg491.txt (tamaño: 1058246 bytes)
✓ libros/pg491.txt descomprimido exitosamente (1058246 bytes)
Descomprimiendo: libros/pg493.txt (tamaño: 99938 bytes)
✓ libros/pg493.txt descomprimido exitosamente (99938 bytes)
Descomprimiendo: libros/pg730.txt (tamaño: 99938 bytes)
✓ libros/pg730.txt descomprimido exitosamente (99938 bytes)
Descompresión completada exitosamente.
Tiempo total de ejecución: 1659.66 ms
nicolepruebas@NicolePruebas: ~/Downloads/OP-Proyecto1-main/serial$ 

```

Figura 2.64: Compresión y Descompresión de 98 Libros. Parte 3.

Cabe aclarar que la versión en texto plano UTF-8 de los Top 100 libros de los últimos 30 días del proyecto Gutenberg no estaba disponible para 2 libros, por lo que la prueba máxima son 98 libros.

Libros	1	5	10	25	50	98
<b>Serial ms</b>	6203	84249	4303478	4789749	6683896	10355328
<b>Fork ms</b>	8480	79260	4838376	5055062	6180812	9455512
<b>Pthread ms</b>	8170	80959	4356952	5294854	5921647	9192930

Cuadro 2.1: Resultados del rendimiento de compresión en milisegundos.

Libros	1	5	10	25	50	98
<b>Serial ms</b>	10	66	272	556	945	1659
<b>Fork ms</b>	12	36	148	282	256	712
<b>Pthread ms</b>	12	38	167	217	289	445

Cuadro 2.2: Resultados del rendimiento de descompresión en milisegundos.

Al analizar las tablas de resultados 2.1 y Tabla 2.2, se puede apreciar que al procesar 1 solo archivo el serial fue más eficiente tanto en compresión como descompresión, pero de 50 archivos en adelante siempre queda como el más ineficiente y el tiempo de corrida entre fork y pthread están tan cercanas que en algunos casos fork queda como mejor opción y en otros resulta pthread. Aún así, el algoritmo de serial obtuvo resultados muy cercanos a los de fork y pthread, teniendo un resultado de compresión de 98 archivos de 10355328 y siendo superado por ellos apenas por 9455512 ms y 9192930 ms respectivamente (Figura 2.65 y 2.66).

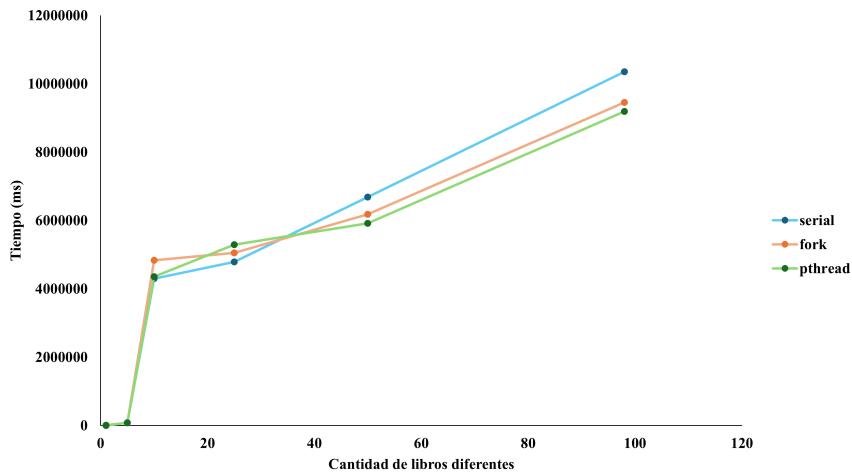


Figura 2.65: Tiempos de compresión para diferentes cantidades de libros.

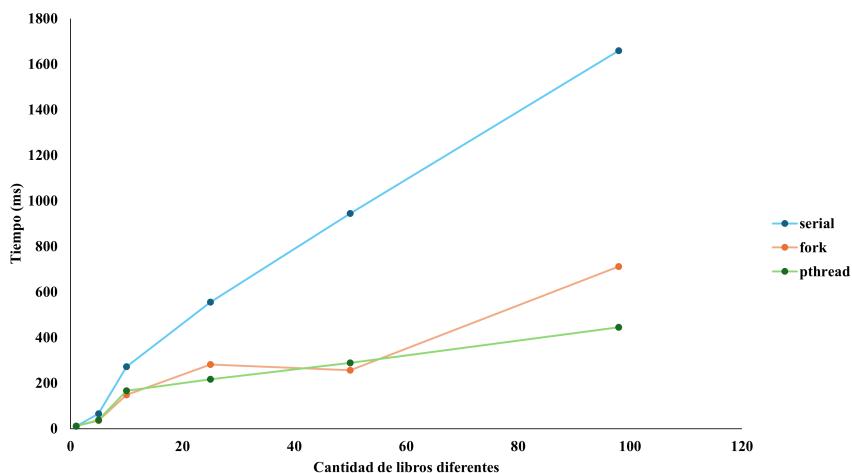


Figura 2.66: Tiempos de descompresión para diferentes cantidades de libros.

Los resultados reflejan que con pocos archivos de texto la compresión es bastante variada, lo que es acertado tomando en cuenta que soluciones implementadas con fork y pthread, o en general parallelizadas, suelen ser demasiado complicadas y pocas veces valen la pena cuando se trata de un número pequeño de tareas. Por otro lado, cuando el número de archivos de texto a comprimir empieza a aumentar, se puede notar que los resultados se normalizan, y el programa serial comienza a tomar más tiempo que los paralelizados (Figura 2.65 y 2.66).

En cuanto a los resultados de la compresión y descompresión, se observa en general que la descompresión es un proceso mucho más rápido. En el caso de la compresión el uso de fork y pthread mostró mejoras solo a partir de un número mayor de archivos y en la descompresión el beneficio de la parallelización fue más evidente incluso desde los 5 libros (Figura 2.65 y 2.66). Esto se debe a que la descompresión implica operaciones que van siguiendo patrones más estables para poder obtener de nuevo el archivo. En cambio, la compresión requiere generar estructuras de datos más complejas, como árboles de Huffman que varían para cada archivo, lo que genera mayor variabilidad en los tiempos (Pu, 2006).

## 2.5. Conclusiones

Aunque la parallelización de tareas puede ser útil para algunos casos, no siempre merece la pena, puesto que el manejo y coordinación de hilos y procesos puede complicar el algoritmo sin dar un gran beneficio.

La implementación de la cola de tareas, el empaquetado de bits, y la creación de un árbol de código Huffman por archivo de texto, fueron factores que afectaron de manera negativa el tiempo de compresión y descompresión de los programas hechos con fork y pthread. En su momento se realizaron para que el archivo comprimido tuviera realmente un tamaño menor a los originales, y para que los hilos ejecutaran tareas específicas y bien denotadas (usando los datos guardados en el vector compartido). Sin embargo, a la hora de revisar los resultados, estas decisiones definitivamente tuvieron un efecto negativo en la eficiencia de los programas, y el tiempo en el que tardan en realizar la compresión y descompresión total.

Las pruebas realizadas muestran que la parallelización depende no solo de la cantidad de archivos, si no también del tipo de tarea que requiere realizarse. En el caso de la compresión y la descompresión se exemplifica esto porque una conlleva procesos más complejos en su funcionamiento. Por eso, aunque la programación paralela puede ser muy eficiente se debe tomar en cuenta el contexto en el que se va a aplicar para verificar si de verdad vale la pena utilizarla.

# Referencias

- Alessandrini, V. (2016). Chapter 3 - creating and running threads. En V. Alessandrini (Ed.), *Shared memory application programming* (p. 29-81). Boston: Morgan Kaufmann. Descargado de <https://www.sciencedirect.com/science/article/pii/B9780128037614000034> doi: <https://doi.org/10.1016/B978-0-12-803761-4.00003-4>
- Apache Maven. (2025). *Fork options and parallel execution (surefire plugin)*. Documentación oficial. Descargado de <https://maven.apache.org/surefire/maven-surefire-plugin/examples/fork-options-and-parallel-execution.html>
- Baumann, A., Appavoo, J., Krieger, O., y Roscoe, T. (2019). A fork() in the road. *Workshop on Hot Topics in Operating Systems (HotOS '19)*. Descargado de <https://www.microsoft.com/en-us/research/uploads/prod/2019/04/fork-hotos19.pdf> doi: 10.1145/3317550.3321435
- Bustos, J. L. (2024, 25 de Abril). *Diferencias entre paralelismo y concurrencia computacional*. Publicación de un blog. Descargado de <https://keepcoding.io/blog/paralelismo-y-concurrencia-computacional/>
- Cormen, T., Leiserson, C., Rivest, R., y Stein, C. (2001). *Introduction to algorithms* (2nd ed.). MIT Press. Descargado de <https://estigia.fi-b.unam.mx/maestria/cormen.pdf>
- Cornell University. (2024). *Bit packing*. <https://www.cs.cornell.edu/courses/cs3410/2024fa/notes/bitpack.html>. (CS 3410: Computer System Organization and Programming, Lecture Notes. Accedido el 23 de septiembre de 2025)
- Domeika, M. (2013). Chapter 8 - communication and synchronization libraries. En B. Moyer (Ed.), *Real world multicore embedded systems* (p. 269-288). Oxford: Newnes. Descargado de <https://www.sciencedirect.com/science/article/pii/B9780124160187000080> doi: <https://doi.org/10.1016/B978-0-12-416018-7.00008-0>
- Harder, D. W. (2012). *fork() and exec()*. Departamento de Ingeniería Eléctrica e Informática, Universidad de Waterloo. Descargado de [https://ece.uwaterloo.ca/~dwharder/icsrts/Tutorials/fork\\_exec/](https://ece.uwaterloo.ca/~dwharder/icsrts/Tutorials/fork_exec/)
- Hart, M. (1992). *The history and philosophy of project gutenberg*. Proyecto Gutenberg. Descargado de [https://www.gutenberg.org/about/background/history\\_and\\_philosophy.html](https://www.gutenberg.org/about/background/history_and_philosophy.html)

- Huffman, K. (2023). *My uncle's bio*. Publicación de un blog. Descargado de <https://www.huffmancode.com/my-uncle/david-bio>
- Huseynzade, S. (2021). *Parallel programming with openmp in c*. Medium. Descargado de <https://medium.com/@localdate/parallel-programming-with-openmp-in-c-b3273fc52ded>
- Jeffers, J., Reinders, J., y Sodani, A. (2016). Chapter 8 - tasks and threads. En J. Jeffers, J. Reinders, y A. Sodani (Eds.), *Intel xeon phi processor high performance programming (second edition)* (Second Edition ed., p. 155-172). Boston: Morgan Kaufmann. Descargado de <https://www.sciencedirect.com/science/article/pii/B9780128091944000089> doi: <https://doi.org/10.1016/B978-0-12-809194-4.00008-9>
- Jenkov, J. (2021). *Java concurrency - producer consumer*. <https://jenkov.com/tutorials/java-concurrency/producer-consumer.html>. (Accedido: 2025-09-23)
- Kavitha, P. (2016). A survey on lossless and lossy data compression methods. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 7(3), 110–114. Descargado de <https://www.ijcset.com/docs/IJCSET16-07-03-049.pdf>
- Monroe, W. (2023). *Parallel programming using the fork-join model in salesforce*. Artículo técnico. Descargado de <https://www.westmonroe.com/insights/parallel-programming-using-the-fork-join-model-in-salesforce>
- Pacheco, P. S. (2011). Chapter 4 - shared-memory programming with pthreads. En P. S. Pacheco (Ed.), *An introduction to parallel programming* (p. 151-207). Boston: Morgan Kaufmann. Descargado de <https://www.sciencedirect.com/science/article/pii/B978012374260500004X> doi: <https://doi.org/10.1016/B978-0-12-374260-5.00004-X>
- Pu, I. M. (2006). Chapter 7 - dictionary-based compression. En I. M. Pu (Ed.), *Fundamental data compression* (p. 117-144). Oxford: Butterworth-Heinemann. Descargado de <https://www.sciencedirect.com/science/article/pii/B9780750663106500106> doi: <https://doi.org/10.1016/B978-075066310-6/50010-6>
- Sari, S. (2023, 8 de Junio). *Concurrency vs parallelism*. Publicación de un blog. Descargado de <https://www.baeldung.com/cs/concurrency-vs-parallelism>
- Silberschatz, A., Galvin, P. B., y Gagne, G. (2018). *Operating system concepts* (10th ed.). John Wiley & Sons. Descargado de <https://os.ecci.ucr.ac.cr/slides/Abraham-Silberschatz-Operating-System-Concepts-10th-2018.pdf>
- Singh, M., Kumar, S., Chouhan, S. S., y Shrivastava, M. (2016). Various image compression techniques: Lossy and lossless. *International Journal of Computer Applications*, 142(6), 23–29. Descargado de [https://www.researchgate.net/publication/303319054\\_Various\\_Image\\_Compression\\_Techniques\\_Lossy\\_and\\_Lossless](https://www.researchgate.net/publication/303319054_Various_Image_Compression_Techniques_Lossy_and_Lossless) doi: 10.5120/ijca2016909829

- Stewart, D., Domeika, M., Hissam, S. A., Hovsmith, S., Ivers, J., Dickson, R., ... Oshana, R. (2013). Chapter 17 - multicore software development for embedded systems: This chapter draws on material from the multicore programming practices guide (mpp) from the multicore association. En R. Oshana y M. Kraeling (Eds.), *Software engineering for embedded systems* (p. 563-612). Oxford: Newnes. Descargado de <https://www.sciencedirect.com/science/article/pii/B9780124159174000177> doi: <https://doi.org/10.1016/B978-0-12-415917-4.00017-7>
- Tammineni, N. (2022). *How to create new process of current program in c using fork()*. DEV Community. Descargado de <https://dev.to/namantam1/how-to-create-new-process-of-current-program-in-c-using-fork-24ho>