Informe Proyecto Chunks

José Jesús Díaz Moreno y María Moreno

22 de Mayo de 2023

Nuestro proyecto consiste en crear un algoritmo en C++ que, contenga lo siguiente:

1.- Crear un vector y dividirlo usando Chunks en C++.

2.- Leer el vector desde un archivo de entrada.

3.- Implementar un generador de archivo de entrada para escribir en el archivo. Nosotros estamos utilizando un generador de archivo aleatorio.

4.- El manejo de la memoria utilizando la función mmap, para casos pequeños, medianos y grandes de información.

5.- Se quiere leer un archivo de imagen en formato PPM, que lo mapea en memoria usando mmap.

Antes de comenzar el algoritmo, hablaremos sobre las librerías que se utilizan:

- **fstream (file stream)**, esta librería se utiliza cuando deseemos alternativamente leer o escribir del mismo fichero en el mismo programa.
- **string**, esta librería contiene un conjunto de funciones para manipular cadenas: copiar, cambiar caracteres, comparar cadenas, etc.
- **random**, esta librería nos permite generar algunos números aleatorio.
- **fcntl**, con esta librería se manipula el descriptor de ficheros. Es una función versátil y se usa para modificar archivos de muchas maneras, como abrir, leer y escribir, etc.
- **unistd**, esta librería define constantes y tipos simbólicos misceláneos y declara funciones misceláneas.
- **sys/stat**, la llamada al sistema fstat envía información de estado sobre el archivo asociado con un descriptor de archivo abierto.
- **iostream**, esta librería declara los objetos que controlan la lectura y escritura en los flujos estándar. Esta inclusión suele ser el único encabezado que necesita incluir para realizar la entrada y salida de un programa de C++ (cin, cout, cerr y clog, etc).
- **limits**, esta librería contiene parámetros de entorno, información sobre limitaciones y rangos para tipos enteros.
- **sys/mman**, esta librería permite declaraciones de gestión de memoria.
- **Ctime**, es la librería que controla el manejo del tiempo.

Para un mejor entendimiento del proyecto, se realizaron tres códigos:

PRIMER CODIGO:

Este código en C++, tiene como objetivo tomar un vector de tamaño determinado por el usuario, dividirlo en subvectores de tamaño 10 y luego imprimir estos subvectores. A continuación, se presenta un resumen de las partes clave del código:

1. Se incluyen las bibliotecas necesarias $\langle iostream \rangle$ *y* $\langle vector \rangle$.

2. Se define una función llamada print_Vector, que recibe un vector y lo imprime elemento por elemento. Si el elemento es menor que 10, se agrega un espacio adicional para mantener la alineación en la salida.

3. En la función main, se solicita al usuario que ingrese el tamaño del vector (max_Vector).

4. Si max_Vector, es mayor que 0, se crea un vector llamado myVector de tamaño max_Vector, y se llena con números enteros desde 0 hasta max_Vector (- 1). Por ejemplo: si el usuario ingresa 25 como tamaño del vector, aparecerá un vector desde 0 hasta 24, en forma ordenada.

5. Se calcula el número total de subvectores de tamaño 10, y se crea un array de vectores llamado chunks para almacenar estos subvectores.

6. Se utiliza un bucle for, para dividir myVector en subvectores de tamaño 10 y almacenarlos en chunks. Es posible que, el último subvector tenga menos de 10 elementos, se ajusta su tamaño.

7. Finalmente, se imprime cada subvector almacenado en chunks utilizando la función print_Vector.

El código utiliza varias técnicas y funciones de la biblioteca estándar de C++, como: vector, resize, copy y next para realizar estas operaciones.

Se realizó una prueba ingresando n=100, y se generaron 10 subvectores de 10 elementos cada uno.

SEGUNDO CODIGO:

El siguiente código en C++, realiza las siguientes acciones:

1. Incluye las bibliotecas necesarias para leer y escribir archivos, generar números aleatorios y trabajar con vectores y memoria mapeada en archivos.

2. Define una función print_Vector, que imprime el contenido de un vector.

3. Define una función archivo_entrada_mmap, que realiza el mapeo de un archivo en memoria utilizando mmap.

4. Dentro de la función **main**:

• Se declara una variable data de tipo puntero a carácter, una variable filename de tipo string y un vector de enteros llamado myVector.

• Se pide al usuario ingresar un número que indica el tamaño del vector del archivo.

• Se crea un archivo llamado "entrada.txt", y se llena con números aleatorios generados utilizando la clase random_device y la función uniform_int_distribution.

• Se lee el contenido del archivo "entrada.txt" en el vector myVector.

• Se divide el vector en subvectores de tamaño “n”. Se realiza el manejo de la memoria utilizando la función mmap, para casos pequeños, medianos y grandes de información. Estos casos son:

1- **CASOS PEQUEÑOS**:

Si el tamaño del vector es menor a 500.000, n = 100.000; los vectores generados en forma aleatoria se ordenarán en subvectores de 10 elementos cada uno, aunque el último subvector es posible que sea menor. Obteniéndose los siguientes mensajes:

- El número ingresado es el tamaño del vector dividido en 10 elementos por subvector.

- El archivo se ha mapeado en memoria de forma exitosa.

- Tiempo de mapeo del archivo en memoria: 0.091234 segundos.

2- **CASOS MEDIANOS**:

Si el tamaño del vector es mayor a 500.000, n = 1.000.000; los vectores generados en forma aleatoria se ordenarán en 4 subvectores. Obteniéndose los siguientes mensajes:

- El número ingresado es el tamaño del vector dividido en 250.000 elementos por subvector.

- El archivo se ha mapeado en memoria de forma exitosa.

- Tiempo de mapeo del archivo en memoria: 0.810535 segundos.

3- **CASOS GRANDES**:

Si el tamaño del vector es mayor a 500.000, n = 10.000.000; los vectores generados en forma aleatoria se ordenarán en 4 subvectores. Obteniéndose los siguientes mensajes:

- El número ingresado es el tamaño del vector dividido en 2.500.000 elementos por subvector.

- El archivo se ha mapeado en memoria de forma exitosa.

- Tiempo de mapeo del archivo en memoria: 8.03681 segundos.

• Se crea un archivo de salida llamado ”salida.txt” y se escriben los elementos de los subvectores en él.

El código completo realiza la tarea de leer un archivo de entrada con números aleatorios, dividirlo en subvectores y escribir esos subvectores en un archivo de salida.

TERCER CODIGO:

Este algoritmo en C++, es un programa que lee una imagen PPM utilizando la función mmap para mapear el archivo en memoria. El programa realiza los siguientes pasos:

1. Verifica si se proporcionó la ruta del archivo de imagen PPM como argumento. Si no se proporcionó, muestra un mensaje de error y termina el programa.

2. Llama a la función archivo_entrada_mmap pasando la ruta del archivo y algunas variables para almacenar información sobre la imagen. Esta función realiza lo siguiente:

- Abre el archivo en modo de solo lectura, usando la función (open). Si no se puede abrir el archivo, muestra un mensaje de error y termina el programa.

- Obtiene información del archivo usando la función fstat. Si falla, muestra un mensaje de error y termina el programa.

- Mapea el archivo en memoria usando la función mmap. Si falla, muestra un mensaje de error y termina el programa.

- Extrae el número mágico, el ancho y el alto de la imagen PPM del archivo mapeado en memoria.

3. Imprime un mensaje de éxito si la imagen se cargó correctamente.

4. Al finalizar, desmapea la memoria y cierra el archivo utilizando las funciones munmap y close(fd).