Informe tarea 2 Sistemas Operativos.

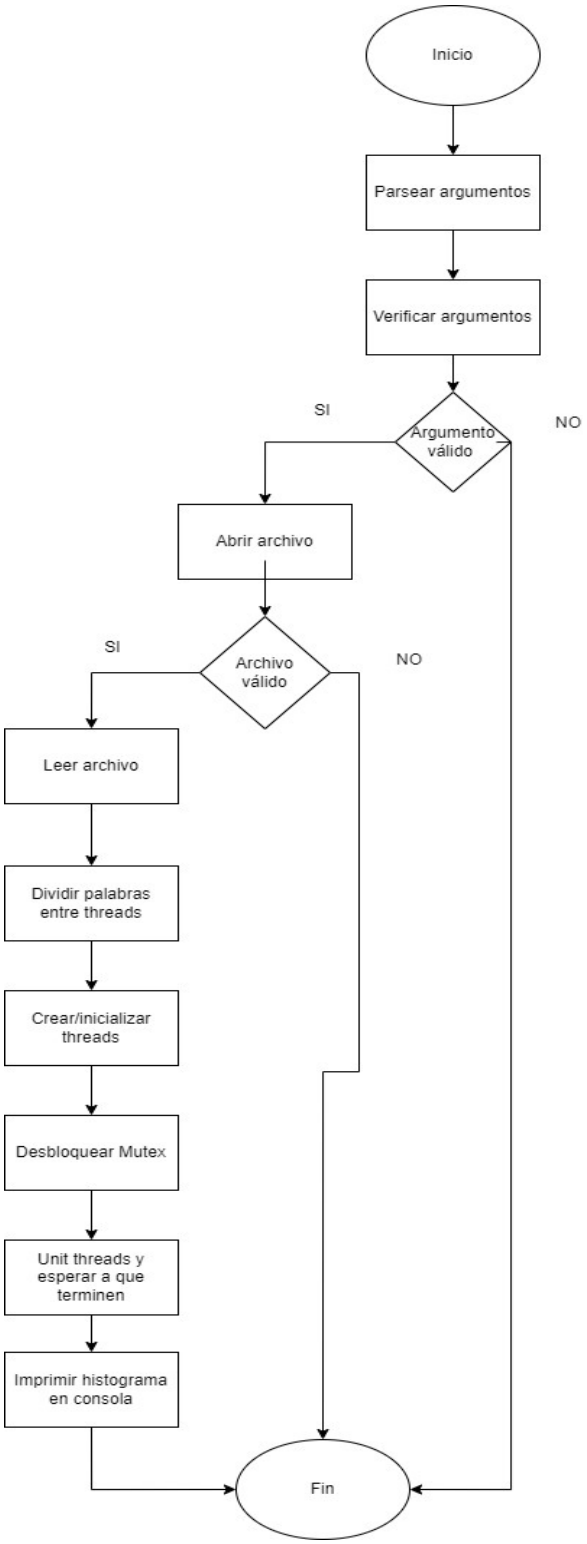
Bastian Freire Flores bastian.freire@alumnos.uv.cl

Stephano Almendra Chaparro stephano.almendra@alumnos.uv.cl

1. Introduccion

El presente informe detalla el trabajo realizado en la Tarea 2, denominada "Histograma de palabras multi-hilo". El objetivo principal de esta tarea fue la implementación de un programa en C++17 que pudiera generar un histograma de palabras a partir de un archivo de texto dado, haciendo uso de múltiples hilos de ejecución (threads). El programa resultante, denominado "histograma_mt", se desarrolló basándose en la solución secuencial proporcionada como punto de partida. La tarea requería que el resultado obtenido con la solución multi-hilo fuese idéntico al obtenido con la solución secuencial, sin ninguna diferencia.

2. Diagrama de flujo de la implementación:



3. Diseño multi-thread

A continuación, se presenta el diseño utilizado para la resolución lo requerido:

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <mutex>
#include <thread>
#include <vector>
#include <unistd.h>

std::mutex mtx;   // Mutex para controlar el acceso al mapa de histograma

void countWords(const std::vector<std::string>& words, std::map<std::string, int>& histogram) {
    for (const auto& word : words) {
        std::lock_guard<std::mutex> lock(mtx);
        histogram[word]++;
    }
}

int main(int argc, char *argv[]) {
    int threads = 1;
    std::string filename;

    int opt;
    while ((opt = getopt(argc, argv, "t:f:h")) != -1) {
        switch (opt) {
        case 't':
            threads = std::stoi(optarg);
            break;
        case 'f':
            filename = optarg;
            break;
        case 'h':
            std::cout << "Modo de uso: " << argv[0]
                      << " -t N -f FILENAME [-h]" << std::endl;
            return 0;
        default:
```

```
            std::cerr << "Uso incorrecto. Usa -h para ayuda." << std::endl;
            return 1;
        }
    }

    std::ifstream file(filename);   // Abre el archivo
    if (!file) {
        std::cerr << "No se pudo abrir el archivo " << filename << std::endl;
        return 1;
    }

    std::string word;
    std::vector<std::string> words;
while (file >> word) {
        words.push_back(word);
    }

    std::map<std::string, int> histogram;
    std::vector<std::thread> threadList;
    int wordsPerThread = words.size() / threads;

    for (int i = 0; i < threads; ++i) {
        int start = i * wordsPerThread;
        int end = (i == threads - 1) ? words.size() : (i + 1) * wordsPerThread;
        std::vector<std::string> subWords(words.begin() + start, words.begin() + end);

        threadList.push_back(std::thread(countWords, subWords, std::ref(histogram)));
    }

    // Espera a que todos los threads terminen
    for (auto& th : threadList) {
        th.join();
    }

    // Imprime el histograma
    for (const auto& pair : histogram) {
        std::cout << pair.first << ": " << pair.second << std::endl;
    }

    return 0;
}
```

4. **Resultados y conclusiones**

   En esta sección, se presentan los resultados obtenidos tras la implementación del programa "histograma_mt" y su ejecución en diferentes escenarios. El objetivo principal de esta tarea era verificar que la solución multi-hilo generara un histograma de palabras idéntico al producido por la versión secuencial, asegurando que no hubiera diferencias en los conteos de palabras. A continuación, se detallan los resultados relevantes:

   Comparación de la Solución Multi-hilo con la Solución Secuencial:
   Para evaluar la solución multi-hilo y su consistencia con la solución secuencial, se realizaron varias ejecuciones utilizando diferentes archivos de texto y la misma entrada. Los resultados revelaron que ambas versiones del programa produjeron histogramas de palabras idénticos. No se encontraron diferencias en los conteos de palabras ni en la distribución del histograma.

   Uso del Programa con Diversos Archivos de Texto:
   El programa "histograma_mt" se ejecutó con varios archivos de texto de diferentes tamaños y contenidos. Se comprobó que el rendimiento del programa multi-hilo era efectivo y eficiente en la generación de histogramas de palabras, independientemente del tamaño del archivo de entrada.