CSCI 4401/5401
Principles of Operating Systems
Fall Semester 2021
Assignment 3 (10/13)

Due Monday, November 7th @ 11:59pm
Reading: Tanenbaum Chapter 3

Submission Guidelines:
1. This assignment is worth 100 points for all students.

2. All answers in the form of text (or short commands) and images should be added to a single PDF file named LastName_FirstName.pdf.

3. Source code should be in separate files (not the main PDF file). Reference these files in your main PDF file.

4. Put all files in a directory named LastName_FirstName, compress it (zip or gz), and upload the compressed file to Moodle. Do not include the data files you will download; they will be too large.

5. It is your responsibility to make sure all files are readable and submitted on time.

Submission:
- Part A.1 requires you to submit a Java source code file and a screenshot of your output.

- Part A.2 requires you to submit a Java source code file and a screenshot of your output.

- Part B.3 requires you to submit a Java source code file and a screenshot of your output.

- Part B.4 requires you to submit a Java source code file and a screenshot of your output.

- Part C requires you to answer two short answer questions about your results.

Introduction

You will be working with 14 different datasets (each file has a different dataset). All 14 files are included with the assignment on Moodle, and below are links to *most* of them if you want to know more about that data. Download all 14 compressed files from Moodle and unzip them into a single directory. DO NOT include the dataset file in your submission; it will be too large.

UFO Reports
Wine Reviews
SMS Spam Collection
Wikipedia Movie Plots
NYC Restaurant Inspection
Fake and Real News
IMDB Movies
Hotel Reviews
Traffic Violations
American Time Use
Resumes
Car Ads

Part A. Find the Most frequent Word in a Single File (26 points)

1. Single-Threaded

Implement a single-threaded Java program that finds the most frequent word of at least 5 characters in a single file for any of the 14 datasets (your choice). Make sure to remove case sensitivity, and I recommend using a regular expression to identify words. Your code should ignore (i.e., not included in results) whitespace and punctuation.

For example, a file, hello.txt, containing "a penny saved is a penny earned" should return:

hello.txt: penny

Submit: 1) Your source code in a separate file, and 2) include in your PDF a screenshot of the output printed to the console.

2. Multithreaded

Create a new program that modifies your program from Part A.1 to be multithreaded. Each thread should find the most frequent (of at least 5 characters) word for a single file. Since you are working with a single file, you will be creating one thread (besides the main() thread). Again, your program should only print (to the console) the file name and the most frequent word.

Submit: 1) Your source code in a separate file, and 2) include in your PDF a screenshot of the output printed to the console.

Part B. Find the Most frequent Word in Many Files (66 points)

3. Single-Threaded

Create a new program that builds on Part A.1. The code should now:

1) iterate through all 14 dataset files in a directory
2) perform a single-threaded most frequent word find for all files
3) time your function and print the total time to the console
4) print (to the console) the file names and the most frequent word for each file

Submit: 1) Your source code in a separate file, and 2) include in your PDF a screenshot of the runtime and output printed to the console.

4. Multithreaded

Create a new program that does the same modifications you just did but now for Part B.1.

1) iterate through all files in a directory
2) perform a multi-threaded most frequent word find for all files
3) time your function and print the total time to the console
4) print (to the console) the file names and the most frequent word for each file

If you are using a Map to store the file names and their most frequent word, you can synchronize a Map using:

Map<String, String> fileWords = Collections.synchronizedMap(new HashMap<String, String>());

This will also require you to import the Collections package:

import java.util.Collections;

Submit: 1) Your source code in a separate file, and 2) include in your PDF a screenshot of the runtime and output printed to the console.

Part C. Observations (8 points)

5. Was there a difference in your runtimes between Part B.3 and Part B.4? Why or why not?

6. Was there a difference in your output between Part B.3 and Part B.4? Why or why not?