Assignment 2 Pthreads

Design Pattern:

Input

Types of Input

```
1. pargrep [-t] word [file]
2. cat file | pargrep word

The first type of input is for a file. The second one is for piped input.
Syntax
   • pargrep- the name of the executable
   • [-t]- the number of threads to be passed (Optional)
   • word- the word that is to be searched in the input or file
   • file- the name of the file(Not required for piped input)

Examples:
$ pargrep the frank.txt
$ pargrep -12 the frank.txt
$ cat frank.txt| pargrep the
```

Output

Output is the lines of the file/input containing the word that is passed as an argument.

Methods

1. main

This is the main method of the program. We get the number of arguments from the users, and the arguments, and segregate the program flow based on that.

2. process_file

Finding out size

This method is used for processing file inputs. In this process, we check the number of bytes of the file by opening it, and going to the end of the file, and checking the number of bytes. Then we close it, and check the offset to find the size of the file.

Number of threads

The number of threads to be used can be an input from the user. In case the user does not provide a number, the number of threads are used as per the following formula:

Number of threads=(size of file in bytes/90000)+1

After recording the times for several number of threads for 2 files of different sizes, it seemed this was giving the optimum result. Actually for one file, The optimum number of threads should be varied for the sizes of files, since if the file size is too small, and the number of threads are fixed, it would affect the

performance adversely, since we would probably not be needing that many threads. On the other hand, if the file is too large, then it makes sense to have more threads reading the file at the same time to optimize the performance. Recorded times for different files and thread numbers follows later.

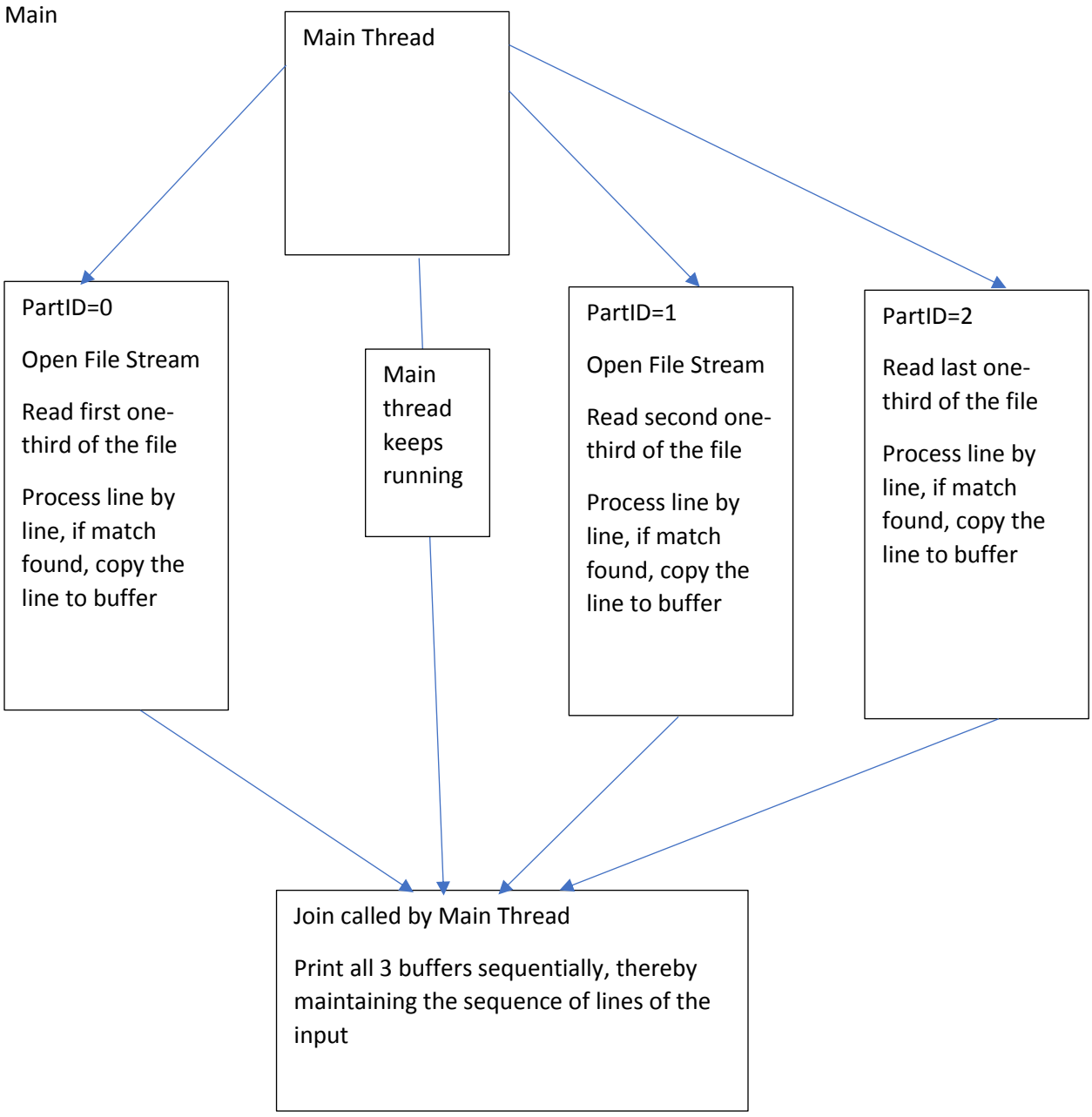
Generation of threads

After determining the number of threads, we generate the threads one by one. We pass the filename, number of threads, number of bytes to be processed by the thread, and the sequence number of that thread (known in the program as partid). We also pass a buffer for the thread to store, so we can print out the result from that.

3. processbytes

So now the program spawns n number of threads, and inside each thread, we open the file, read the file , process the lines according to the input word.

Below is a diagram illustrating the same(for 3 threads):



Recorded running times

Please note, the recorded running times vary widely for different times, and therefore one might not get the same times when running the program. The times recorded below are recorded within a span of 5 minutes, to try and minimize the variance of resources availabe.

File-Pride and Prejudice by Jane Austen

https://www.gutenberg.org/ebooks/1342

File Size=724725 bytes

# of threads	5	6	7	8	9	10	11	12	13	14	Standard GREP
Run 1(ms)	14	14	11	12	12	11	12	12	12	10	26
Run 2(ms)	14	12	12	11	11	12	11	10	12	12	25
Run 3(ms)	13	12	10	11	12	11	12	12	12	13	25
Run 4(ms)	13	14	11	11	11	12	13	11	13	11	25
Run 5(ms)	14	13	10	12	13	10	11	12	11	11	25
Average(ms)	13.6	13	10.8	11.4	11.8	11.2	11.8	11.4	12	11.4	25.2

File-The Count of Monte Cristo, Illustrated by Alexandre Dumas

https://www.gutenberg.org/ebooks/1184

File Size: 2783701 bytes

# of threads	25	26	27	28	29	30	31	32	33	34	Standard GREP
Run 1(ms)	535	339	345	368	614	454	503	548	592	657	765
Run 2(ms)	720	571	574	651	509	360	418	474	526	574	994
Run 3(ms)	504	430	417	565	390	650	406	396	444	503	862
Run 4(ms)	329	645	658	434	719	548	279	321	384	421	1120
Run 5(ms)	547	494	517	324	556	452	610	658	322	351	998
Average(ms)	527	496	502	468	558	493	443	479	454	501	948

Formula for number of threads

We take the second book to get the formula to determine the number of threads, since there are less external factors affecting the run time, since the size of this file is relatively large.

So for number of threads, we take:

Absolute value of (2783701/ 90000)+1=31.

Handling Piped Input

For inputs where we have filenames as command line arguments, we can easily apply pthreads to increase the performance of the program. This can be done because we know the size of the file, we can go to any point of the file at any point of time when the file stream is open. However in case of Piped input, which C treats as Standard input(STDIN), we don't have this luxury. Stdin is a stream of data which is passed as input, and there is absolutely no way to get the size of the data or the middle points where the data can be divided. So, pthread is NOT implanted for piped inputs. In case of piped input, the program simply scans them line by line, and if the given word is present in the line, it prints the word.

Changing Bashrc

The assignment asks that the below command be executable

```
pargrep [-t] word [file]
```

If the program needs to run without the dot slash symbol, the following line needs to be added in the ~/.bashrc file.

alias pargrep='./pargrep'