ITCS343 Project

Creating a multi-thread program. [Last updated Mar 11, 2024]

Specification

In this project, we will develop a simple word-processing program that

1. read the text one word at a time,
2. spell-check words,
3. count characters, words, and misspelled words,
4. save the output to files.

These four will work synchronously in their thread(s). We will assume that the computer can only hold **B** input words. However, the threads may share buffers to facilitate data communication and the input words will be written to a file once the program processes them.

1. Reading Text

The program will read text from the standard input. However, there is a rate limit for reading. We will assume that the reading rate limit is **R** words per second. We will also assume that all words are separated by white space(s).

Tip: Direct a file into the program's standard input using “<” character (this is called “I/O Redirection”.) For example, consider the following program:

```
#include <stdio.h>

int main() {
    char word[300];
    while (1) {
        if(scanf("%s", word) != EOF) {
            printf("word: %s\n", word);
        } else {
            break;
        }
    }
    return 0;
}
```

nor.txt

```
kdkdkd aa ddd ccc
skdk
sldf
```

You can run the command (left) to get the output (right):

`./main < nor.txt`	`word: kdkdkd`
	`word: aa`
	`word: ddd`
	`word: ccc`
	`word: skdk`
	`word: sldf`

2. Spell Check

The program will check spelling one word at a time. If a word is misspelled, the program will tag the word as `[MS]____[/MS]`, such as "`[MS]cofee[/MS]`" (the actual word is "coffee" in case you do not know.)

We will assume that all of the words spelled correctly are in the following file: download. We will also assume that your program cannot hold all "correct" words in memory. So, your spell-check algorithm will have to read one reference word at a time (but you can check many input words in one iteration).

Again, a rate limit is applied here. We can only check **S** words per second. Additionally, we will assume that there are **N** spell-check threads, each with its own **S** limit.

3. Counting

The program will count the number of characters, words, and misspelled words. This counting does not involve any I/O, so there is no rate limit here.

4. Saving outputs

The program constantly saves output to files (you can think of these files as I/O to a screen display). There are two files here:

1. A text output file contains all input words after the spell check (i.e., including the [MS] tags.) The write mode should be "appended" to avoid re-writing all input words repeatedly.
2. A statistic file that outputs the number of characters, number of words, and number of misspelled words. The write mode should be "write" because the old statistics are not helpful.

We will assume that the program can only write `W` times per second. Writing the text output once and the statistics once contributed two times.

Experiments

You will run many experiments using different setting for the independent variables below:

- B
- R
- S
- N
- W

For each experiment, you should observe the dependent variables below:

- Run time
- Throughput rate