LABWORK 4: Word Count

Student's name: Nguyen Thai Duong

Student's ID: BI12-117

I. Choice of Mapreduce implementation

The MapReduce implementation in my C code was selected for its simplicity and effectiveness in performing word count on a text file. Here are the main reasons for opting for this implementation:

- Simplicity: The implementation is straightforward and easy to grasp, making it suitable for educational purposes and initial exploration of map-reduce concepts.
- Clear Separation of Tasks: The code neatly separates the mapping and reducing phases, adhering closely to the fundamental map-reduce paradigm.
- Minimalistic Approach: It relies on basic C programming constructs without external dependencies, ensuring portability and lightweight operation.
- Educational Value: This implementation serves as a practical illustration of how map-reduce principles can be applied to solve a simple problem, aiding in learning and understanding core concepts.

- **Workflow**

a. Data Structures

The code defines a WordCount structure containing a word (of maximum length MAX_WORD_LENGTH) and its corresponding count.

```
typedef struct {
    char word[MAX_WORD_LENGTH];
    int count;
} WordCount;
```

b. Map Function

- Parses each line of the input file.
- Tokenizes the line into words using strtok.

```c
void map(const char *line, WordCount **wordCounts, int *numWords) {
    char *token;
    char copyLine[1000];
    strcpy(copyLine, line);
    token = strtok(copyLine, " ");
```

- Converts each word to lowercase and removes non-alphabetic characters.
- Counts the occurrences of each word and updates the wordCounts array accordingly.
- Dynamically allocates memory for new words as needed using realloc.

```c
while (token != NULL) {

    int i, j = 0;
    char word[MAX_WORD_LENGTH];
    for (i = 0; token[i]; i++) {
        if (isalpha(token[i])) {
            word[j++] = tolower(token[i]);
        }
    }
    word[j] = '\0';
    if (strlen(word) > 0) {

        int found = 0;
        for (i = 0; i < *numWords; i++) {
            if (strcmp(word, (*wordCounts)[i].word) == 0) {
                found = 1;
                (*wordCounts)[i].count++;
                break;
            }
        }

        if (!found) {
            *wordCounts = (WordCount *)realloc(*wordCounts, (*numWords + 1) * sizeof(WordCount));
            strcpy((*wordCounts)[*numWords].word, word);
            (*wordCounts)[*numWords].count = 1;
            (*numWords)++;
        }
    }
    token = strtok(NULL, " ");
}
```

c. Reduce Function

- Definition of the reduce function.
- Implementation of sorting the wordCounts array based on word counts.

```
void reduce(WordCount *wordCounts, int numWords) {
    for (int i = 0; i < numWords - 1; i++) {
        for (int j = i + 1; j < numWords; j++) {
            if (wordCounts[i].count < wordCounts[j].count) {
                WordCount temp = wordCounts[i];
                wordCounts[i] = wordCounts[j];
                wordCounts[j] = temp;
            }
        }
    }
}
```

d. Figure

Here's is the figure that illustrated the workflow of Mapper and Reducer:

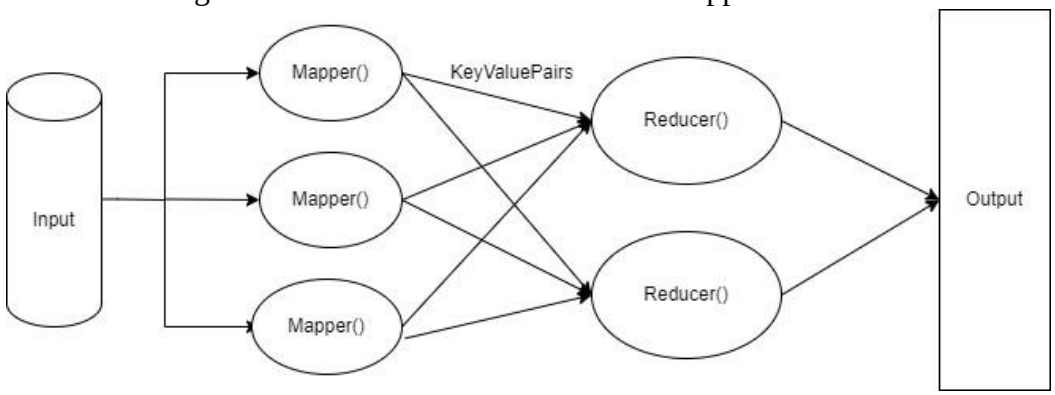
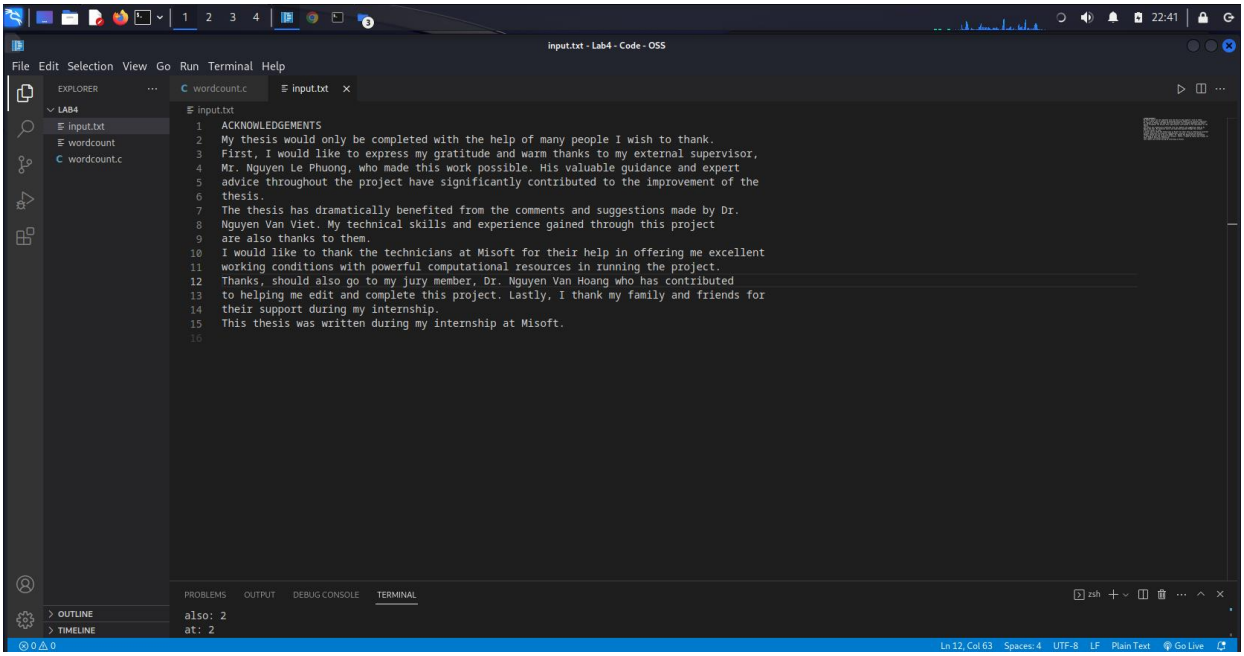


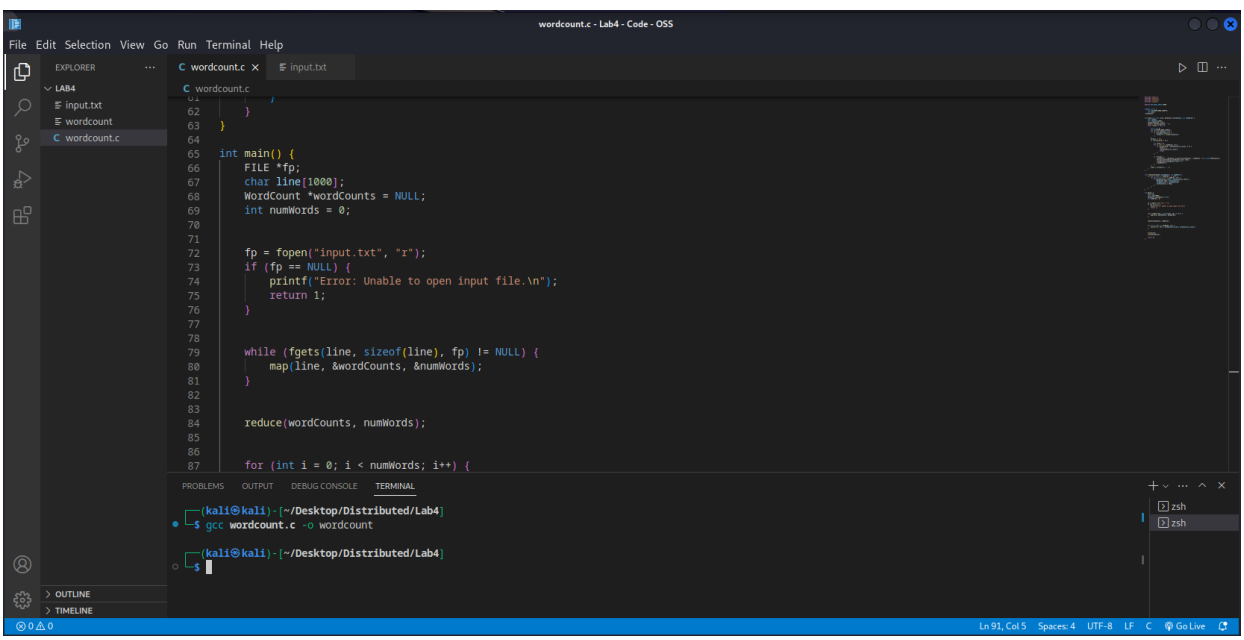
Figure 1. Workflow of MapReduce

II. Implementation

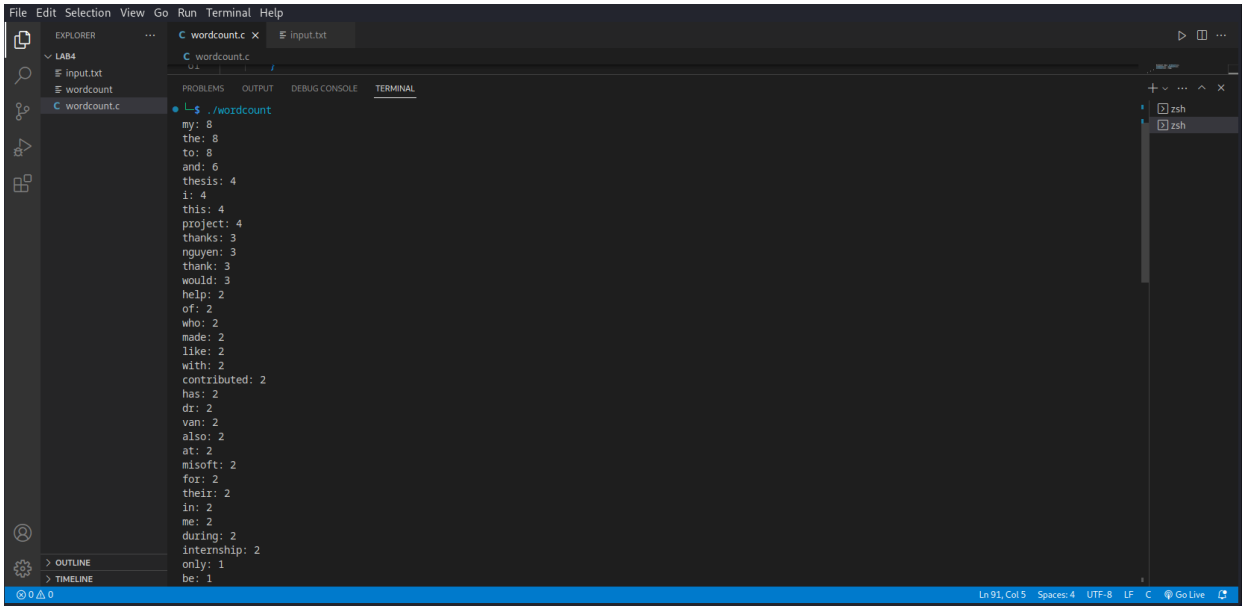
This is the test file, which is "input.txt"

Now we run the wordcount.c with input is the "input.txt"


input.txt - Lab4 - Code - OSS
File Edit Selection View Go Run Terminal Help
EXPLORER
LAB4
input.txt
wordcount
wordcount.c
ACKNOWLEDGEMENTS
My thesis would only be completed with the help of many people I wish to thank.
First, I would like to express my gratitude and warm thanks to my external supervisor,
Mr. Nguyen Le Phuong, who made this work possible. His valuable guidance and expert
advice throughout the project have significantly contributed to the improvement of the
thesis.
The thesis has dramatically benefited from the comments and suggestions made by Dr.
Nguyen Van Viet. My technical skills and experience gained through this project
are also thanks to them.
I would like to thank the technicians at Misoft for their help in offering me excellent
working conditions with powerful computational resources in running the project.
Thanks, should also go to my jury member, Dr. Nguyen Van Hoang who has contributed
to helping me edit and complete this project. Lastly, I thank my family and friends for
their support during my internship.
This thesis was written during my internship at Misoft.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
also: 2
at: 2


wordcount.c - Lab4 - Code - OSS
File Edit Selection View Go Run Terminal Help
EXPLORER
LAB4
input.txt
wordcount
wordcount.c
}
}
int main() {
FILE *fp;
char line[1000];
WordCount *wordCounts = NULL;
int numWords = 0;
fp = fopen("input.txt", "r");
if (fp == NULL) {
printf("Error: Unable to open input file.\n");
return 1;
}
while (fgets(line, sizeof(line), fp) != NULL) {
map(line, &wordCounts, &numWords);
}
reduce(wordCounts, numWords);
for (int i = 0; i < numWords; i++) {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(kali㉿kali)-[~/Desktop/Distributed/Lab4]
$ gcc wordcount.c -o wordcount
(kali㉿kali)-[~/Desktop/Distributed/Lab4]
$

=> The result:


File Edit Selection View Go Run Terminal Help
EXPLORER
LAB4
input.txt
wordcount
wordcount.c
wordcount.c
input.txt
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
└─$ ./wordcount
my: 8
the: 8
to: 8
and: 6
thesis: 4
i: 4
this: 4
project: 4
thanks: 3
nguyen: 3
thank: 3
would: 3
help: 2
of: 2
who: 2
made: 2
like: 2
with: 2
contributed: 2
has: 2
dr: 2
van: 2
also: 2
at: 2
misoft: 2
for: 2
their: 2
in: 2
me: 2
during: 2
internship: 2
only: 1
be: 1
zsh
zsh
OUTLINE
TIMELINE
Ln 91, Col 5 Spaces: 4 UTF-8 LF C Go Live