Name: Yuxiang Chen

SID: 904224400

Subject: Advanced Operating Systems

09 February 2022

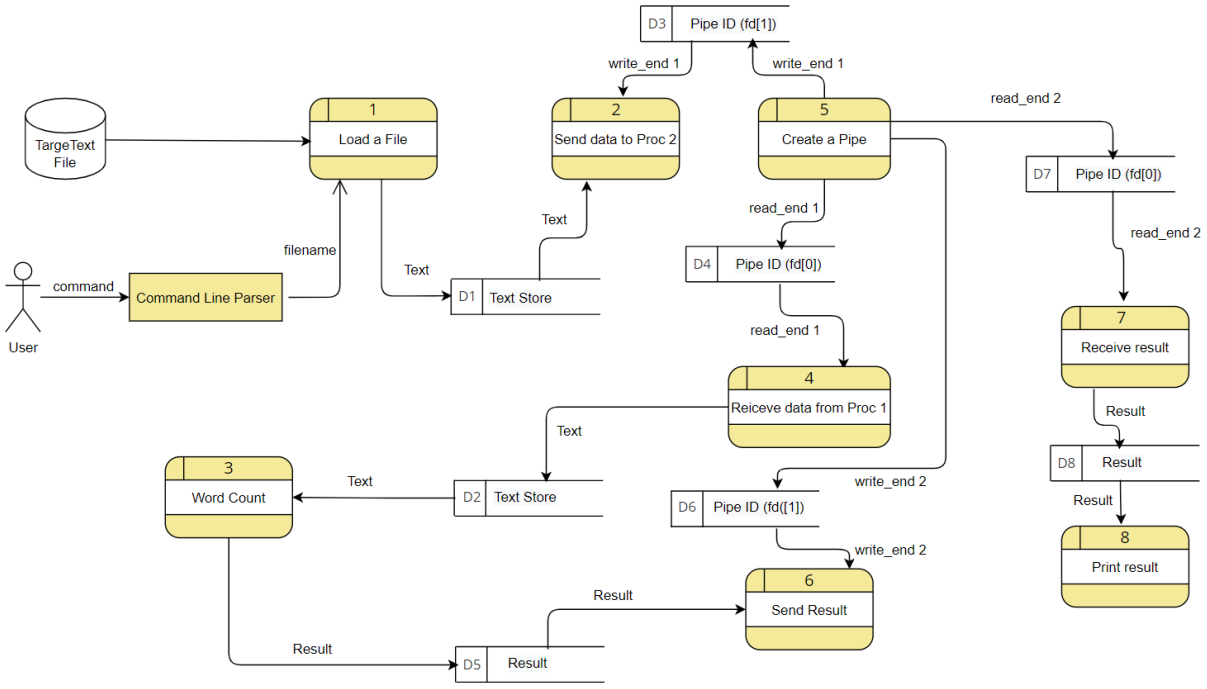
COMP7500 Project 2
pWordCount: A Pipe-based WordCount Tool

TASK OVERVIEW

This project aims at providing you with an opportunity to design and implement a C program, where two processes are cooperating through Unix Pipes. Pipes is one of the first interprocess communication mechanisms or IPCs in early Unix systems. Unix pipes offer simple yet efficient ways of communicating among collaborating processes. In this project, we focus on ordinary pipes that enable two processes to communicate in a producer-consumer fashion. In other words, a producer process writes to one end of a pipe (i.e., write-end) and a consumer process reads from the other end (i.e., read-end).

DESIGN

Data Flow Diagram



The architecture of the program

(1) User–input handling;

(2) File-input handling;

(3) Two Unix pipes;

(4)Two Cooperating Processes;

EXECUTION

Makefile:

```
CC = gcc
CFLAGS = -g -lm

pWordCount: pWordCount.c
        $(CC) $(CFLAGS) pWordCount.c -o  pWordCount

clean:
        rm pWordCount
makefile (END)
```

Usage:

1. *Make*

2. */pWordCount <file_name>*

After execution:

3. *make clean*

RESULT & TEST

Two text files were used to test the functionality.

1. Error handling for input file name validation.

    a. Case 1: Not filename

```
[root@localhost project2]# ./pWordCount
Please enter a file name.
Usage: ./pWordCount <file name>
```

Result: Successful!

    b. Case 2: File cannot be opened

```
[root@localhost project2]# ./pWordCount xxx
Error: File xxx cannot be opened
```

Result: Successful!

    c. Case 3: Pipe failed

```
// Create the pipe
for(i = 0; i < 2; i++){
  if(pipe(fd[i]) == -1)
  {
    fprintf(stderr, "Error: Pipe failed");
    return EXIT_FAILURE;
  }
}
```

    d. Case 4: Fork Failed

```
// Fork a child process
pid = fork();
if (pid < 0)
{
  fprintf(stderr, "Error: Fork failed");
  return EXIT_FAILURE;
}
```

2. “input.txt” is a simple text file with. “This is …”

Inputs:

```
This is the second project of COMP7500 class.
input.txt (END)
```

Outputs:

```
[root@localhost project2]# ./pWordCount input.txt
Beginning of pipe....
Process 1 is reading file "input.txt" now ...
Process 1 starts sending data to Process 2 ...
Process 2 finishes receiveing data from Process 1 ...
Process 2 is counting words now ...
Process 2 is sending the result back to Process 1 ...
Process 1 Total words: 8.
```

Result: Successful! Expected result will be 8.

3. “text.txt” is a more complicate file, with much more ‘\n’ new line commend. The aim is going to check whether this program can handle Escape Character. And multi line inputs.

Inputs:

```
Hello
this is trying to
count how many
line and times

Hello world!
end.
text.txt (END)
```

Outputs:

```
[root@localhost project2]# ./pWordCount text.txt
Beginning of pipe....
Process 1 is reading file "text.txt" now ...
Process 1 starts sending data to Process 2 ...
Process 2 finishes receiveing data from Process 1 ...
Process 2 is counting words now ...
Process 2 is sending the result back to Process 1 ...
Process 1 Total words: 14.
```

Result: Successful! Expected result will be 14.

CONCLUSION

I will successful compelete Assignment Project 2 pWordCount. During this assignment,

For technique I used. I have more understanding for gdb and gcc, I tried use vi tool for small text file edit. Emacs for writing program. And I used Makefile to compiler file.

For operating system knowledge, in this project, I use pipe() for creating two pipes to communicate between processes, I used fork() for parent process to creating child process. I handed the file input and read file from user input problem.