Project2 A Pipe-Based Word Count Tool Report

gzl0034 Gaoxiang Li

1.Separate Compilation Design(Small file only)

Divide the project into three parts.

(1)word_count function: a function only use to count the words from a string(not a file)

(2)load_file function: a function to load the file and check error of the file type and read from the file then write to a string(buffer)

(3)pwordcount (main function)

A.check error of input and use load_file function to write into buffer

B.create 2 pipes

C.create parent and child process

D.send message to child process from parent process and wait child process to finish

D.read message and do word_count function send the result of word_count to parent process in child process

F.parent process return the result

2.Implement detail

(1)load_file function

```
char *load_file(const char *filename){
        FILE *fp;
         static char res[100];
        fp=fopen(filename,"r");
        if(fp==NULL){
                printf("Read file failed!.\n");
                exit(1);
        }else{
                printf("Read file success!\n");
                fgets(res,100,fp);
                return res;
                fclose(fp);
        }
};
```

Input parameter: file name as const char

Open the file and return the txt in file as a char

(2) word_count function

```
int word_count(const char word[]){
        static int count=0; //result
        const char *pos =word;//postion to show the state
        int inword=0;
        do switch(*pos){ // use do-while  loop at least run once
                case '\0':
        case ' ': case '\t': case '\n': case '\r':
            if (inword) { inword = 0; count++; }
            break;
        default: inword = 1;
        }while(*pos++); //continue to next position

        return count;
}
```

(3) pwordcount (main function)

Detail in source code pwordcount.c

3.Sample Input and Output

(1) input: ./pwordcount

Output:

```
[root@localhost ~]# ./pwordcount
Invalid input please check your input.
 Usage: ./pwordcount <file_name>
[root@localhost ~]#
```

(2) input: ./pwordcount test1 test2

Output:

```
[root@localhost ~]# ./pwordcount test1 test2
Invalid input please check your input.
 Usage: ./pwordcount <file_name>
```

(3) Input: ./pwordcount test

Output :

```
[root@localhost ~]# ./pwordcount test
Begin to load file.
Read file success!
Load file into buffer.
Read file success!
Try to create pipe.
Try to fork a child process
Parent process begin to write msg to pipe
Child process begin to read msg from pipe
Child begin to count words.
Child process begin to send result to parent by pipe.
Child process finish
parent process get result from pipe.
The result is 20.
process finish.
[root@localhost ~]#
```

(4) Input: ./pwordcount wordtest

Output:

```
[root@localhost ~]# ./pwordcount wordtest
Begin to load file.
Read file success!
Load file into buffer.
Read file success!
Try to create pipe.
Try to fork a child process
Parent process begin to write msg to pipe
Child process begin to read msg from pipe
Child begin to count words.
Child process begin to send result to parent by pipe.
Child process finish
parent process get result from pipe.
The result is 4.
process finish.
[root@localhost ~]#
```

Reference： word_count function in stackoverflow(use the idea only)