SOUTHWEST JIAOTONG UNIVERSITY • UNIVERSITY OF LEEDS
西南交通大学－利兹学院
SWJTU-LEEDS JOINT SCHOOL

SWJTU-Leeds Joint School

Coursework Submission – Cover Sheet

Please complete ALL information

Leeds Student ID Number: 201199581
SWJTU Student ID Number: 2017110248
Student Name: Kang Liu
Module Code & Name: XJCO2211 Operating Systems
Title of Coursework Item: Coursework Report
For the Attention of: Dr Jie Xu

Deadline Time:	Deadline Date: Monday, 21th October
Student Signature: Kang Liu	**For office use:** *date stamp here*

DECLARATION of Academic Integrity

I am aware that the University defines plagiarism as presenting someone else's work, in whole or in part, as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance.

I promise that in the attached submission I have not presented anyone else's work, in whole or in part, as my own and I have not colluded with others in the preparation of this work. Where I have taken advantage of the work of others, I have given full acknowledgement. I have not resubmitted my own work or part thereof without specific written permission to do so from the University staff concerned when any of this work has been or is being submitted for marks or credits even if in a different module or for a different qualification or completed prior to entry to the University. I have read and understood the University's published rules on plagiarism and also any more detailed rules specified at School or module level. I know that if I commit plagiarism I can be expelled from the University and that it is my responsibility to be aware of the University's regulations on plagiarism and their importance.

I re-confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes.

I confirm that I have declared all mitigating circumstances that may be relevant to the assessment of this piece of work and that I wish to have taken into account. I am aware of the University's policy on mitigation and the School's procedures for the submission of statements and evidence of mitigation. I am aware of the penalties imposed for the late submission of coursework.

Coursework Report

XJCO2211 Operating Systems

Liu Kang

sc17kl@leeds.ac.uk

Operating Systems Coursework Report

1. Introduction

This coursework required to build a simple shell in C in Linux. A shell is a user interface for access to an operating systems services. The shell had implemented 4 common built-in commands in Linux, *grep* command, *ex* command to execute outer program and an additional built-in command. This report also discussed the differences of shell in other operating system and reflection through this coursework in the end.

2. *Built-in* Commands

There were 4 built-in commands which were required to be implemented. They were:

a. *info*: Print simple information of this shell.

b. *exit*: Close the shell.

c. *pwd*: Print the current working directory of the shell to the standard output.

d. *cd PATH*: Change the current working directory of the shell to the directory given by the parameter PATH.

To achieve those features, the shell first read its own working directory and stored it to a string named *CWdir* via *readlink("/proc/self/exe",CWdir,255)*. Then the shell started a loop to receive user's commands, parse those commands and execute them.

Those implemented built-in commands were showed below:

```
[myShell@llkser]$$ info
XJCO2211 Simplified Shell by LLKSER.
[myShell@llkser]$$ pwd
/root/OScw
[myShell@llkser]$$ cd /root
Current working direction changed!
[myShell@llkser]$$ pwd
/root
[myShell@llkser]$$ exit
Bye!
```

3. String-matcher

The shell had implemented a *grep* command: *grep -c pattern file*, which output the times of the file that satisfies the pattern match. A simple method was to compare every character in pattern string to every position in text file. Suppose the length of pattern string was n while text file was m, the method achieved an algorithm of O(n*m).

In this shell, KMP algorithm which was modified from finite automaton had achieved O(n+m), which was far better than traditional methods. Instead of computing transition function δ(q,a) in finite automaton, KMP algorithm considered the offset of pattern string and compute function π to show if any of the offset of pattern string would be possible to match itself.

4. Executing Outer Program

The shell had implemented 3 *ex* commands to execute outer program. They were:

a. *ex PATH ARGS*: Execute the program specified in the PATH parameter and pass it the remaining arguments.

b. *ex ProgA* | *ex ProgB*: Redirect the standard output of ProgA to the standard input of ProgB.

c. *ex ProgA* > *a.txt*: Redirect the standard output of ProgA to the file a.txt.

To execute an outer program in Linux, the shell should first fork a new process. In C program, statement: *pid_t pid=fork();* can fork a child process from father process. And argument *pid=0* meant the following statement was executed in child process. The statement: *wait(NULL);* kept father process waiting for the end of child process.

Then the shell executed outer program in child process by *exec* functions. The function: *execl(char* address, char* args[0], char* args[1]…)* received the address of a program and its command line arguments. The *args[0]* was the name of the program. The *args[1]* and following arguments were passed to the program. What's more, to redirect the standard output of a program to a given file, the function *dup2()* was used. The function *dup2()* changed output to the given file stream instead of standard stream. So the program would print its output in the given file.

5. Additional Feature

The shell had implemented an additional built-in command *ls* to list all the files under the current working directory. This feature was aimed to help users to access the file information of current working directory and check for their files easily.

The feature was implemented by functions in library <dirent.h>:

*DIR *dirp;*
*struct dirent *dp;*

```
dirp = opendir(CWdir);
while ((dp = readdir(dirp)) != NULL)
    printf("%s\n", dp->d_name );
closedir(dirp);
```

6. Differences in Other Operating System

The shell was built in C in Linux. So, it had used a lot of libraries and functions like <linux/limits.h> which were only available in Linux.

For example, function fork() created a new process in Linux while function CreateProcess() did the same job in Windows. What's more, function readlink() was used to access current working directory of the program in Linux while function getCwd() should be called in Windows. So, if this shell was built in other operating system, the libraries and functions it used should be available in its working system.

7. Reflection

In this coursework, I had built a simple shell in C in Linux, through which I had learnt a lot of useful commands and functions in Linux and understood more about how a operating system was built and worked. More specifically, I learnt how to create new process and how to build a pipeline between two programs to communicate information. What's more, in the string-matcher, I had used a more efficient algorithm KMP to achieve the feature.