



Undergraduate Lab Report

Course Title: Experiment of Computer Organization

Course No: 60080014

Student Name: _____

Student No: _____

School: International School

Department: _____

Major: Computer Science & Technology

Instructor: SUN Heng

Academic Year: 202~202, Semester : 1st [☒] 2nd [☐]

Academic Affairs Office of Jinan University

Date (dd/mm/yyyy) _____

Computer Organization Lab List

Student Name:_____ Student No:_____

ID	Lab Name	Type
1	Number Storage Lab	Individual
2	Manipulating Bits	Individual
3	Simulating Y86-64 Program	Individual
4	Performance Lab	Team
5	A Simple Real-life Control System	Team
6	System I/O	Individual

Undergraduate Lab Report of Jinan University

Course Title Experiment of Computer Organization Evaluation

Lab Name System I/O Instructor SUN Heng

Lab Address _____

Student Name _____ Student No _____

College International School

Department _____ Major CST

Date _____ / _____ / _____ Afternoon

1. Introduction

You have implemented Input/Output interfaces between mechanical and computational worlds in Lab 5. Further, this is a system robust I/O lab from the section 10.5 in textbook. The purpose of this lab is to become more familiar with the Unix-I/O.

2. Lab Instructions or Steps

You are provided with the C programs. These codes are available from the lab materials. Related concepts can be found in the file *07.ppt* of our theoretical course.

Activities to do-

a) Modify the *cpfile* program so that it uses the Rio unbuffered functions *Rio_readn* and *Rio_writen* to copy standard input to standard output, 10 bytes at a time. Use **\$ make** to compile, and then use **\$./main.out** to run. Remember to use **\$ make** again after modifying *cpfile.c*;

b) Modify the original *cpfile* program so that it takes an optional command line argument *infile*, such as: *\$./main.out infile.txt*; If *infile* is given, then copy *infile* to standard output; otherwise, copy standard input (*STDIN_FILENO*) to standard output as before. The twist is that your program must use the original copy loop (lines 13–15) for both cases. You are only allowed to insert code, and you are not allowed to change any of the existing code.

3. Lab Device or Environment

Ubuntu 16.04 (64-bit) with AMD Ryzen 9 5900HS CPU @ 3.30GHz and 4GB memory on virtual machine (Oracle VM VirtualBox)

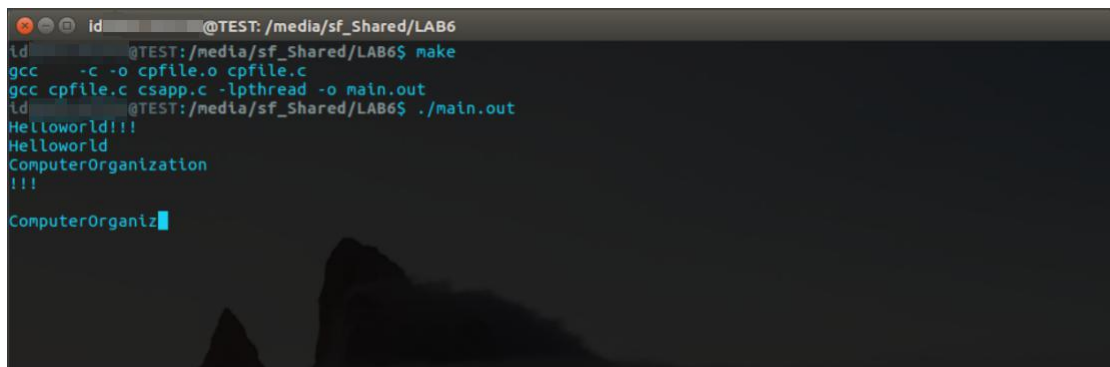
4. Results and Analysis

Activity a):

Analysis:

Modify the *cpfile* program so that it uses the Rio unbuffered functions *Rio_readn* and *Rio_writen* to copy standard input to standard output, 10 bytes at a time. So, just need to change the *Rio_readlineb* function in the while loop to *Rio_readn* and change the third argument to 10.

Result:



```
id @TEST: /media/sf_Shared/LAB6
id @TEST: /media/sf_Shared/LAB6$ make
gcc -c -o cpfile.o cpfile.c
gcc cpfile.c csapp.c -lpthread -o main.out
id @TEST: /media/sf_Shared/LAB6$ ./main.out
Helloworld!!!
Helloworld
ComputerOrganization
!!!
ComputerOrganiz
```

Activity b):

Analysis:

Modify the original *cpfile* program so that it takes an optional command line argument *infile*, such as: *\$./main.out infile.txt*; If *infile* is given, then copy *infile* to standard output; otherwise, copy standard input (*STDIN_FILENO*) to standard output as before. Therefore, using the I/O redirection knowledge from Chapter 10 and the way the *main* function accepts parameters, when the number of parameters accepted during the execution of *main.out* is 2 (where the first is the path of the program file), we open the second parameter (the file name) with the *open* function to obtain a new file identifier. The standard input identifier is then redirected to the new file identifier as input to the program by using *dup2* function.

Result:

```
id @TEST: /media/sf_Shared/LAB6
id @TEST:/media/sf_Shared/LAB6$ make
gcc -c -o cpfile.o cpfile.c
gcc cpfile.o csapp.c -lpthread -o main.out
id @TEST:/media/sf_Shared/LAB6$ ./main.out
Hello, Computer Organization LAB!
Hello, Computer Organization LAB!
^C
id @TEST:/media/sf_Shared/LAB6$ ./main.out infile.txt
I/O redirection is successful!
id @TEST:/media/sf_Shared/LAB6$
```

5. Appendix (Program Code)

Activity a):

```
1.  /* ALL of rio function implementations are obtained from csapp.c
   */
2.  /* Please note the usage of wrapper function */
3.
4.  /* $begin cpfile */
5.  #include "csapp.h"
6.
7.  int main(int argc, char **argv)
8.  {
9.      int n;
10.     char buf[MAXLINE];
11.
12.     while((n = Rio_readn(STDIN_FILENO, buf, 10)) != 0)
13.         Rio_writen(STDOUT_FILENO, buf, n);
14.     exit(0);
15. }
16. /* $end cpfile */
```

Activity b):

```
1.  /* ALL of rio function implementations are obtained from csapp.c
   */
2.  /* Please note the usage of wrapper function */
3.
4.  /* $begin cpfile */
5.  #include "csapp.h"
6.
7.  int main(int argc, char **argv)
8.  {
9.      int n;
```

```
10.     rio_t rio;
11.     char buf[MAXLINE];
12.
13.     int file_number;
14.
15.     if(argc == 2){
16.         file_number = open(argv[1], O_RDONLY);
17.         Dup2(file_number, STDIN_FILENO);
18.     }
19.
20.     Rio_readinitb(&rio, STDIN_FILENO);
21.     while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0)
22.         Rio_writen(STDOUT_FILENO, buf, n);
23.     exit(0);
24. }
25. /* $end cpfile */
```