## 1 Introduction

We 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 and Steps

### 2.1 Lab Instructions

We 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.

### 2.2 Steps

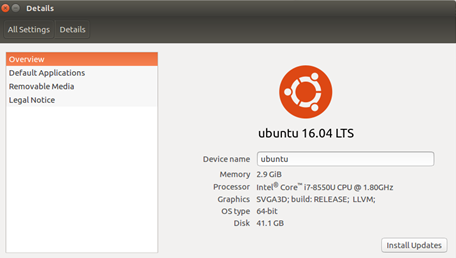
1. Modify the *cpfile* program so that it uses the Rio unbuffered functions *Rio\_readn* and *Rio\_writen* to copy standard input to standard output, 10 bytes at a time. Use *$ make* to compile, and then use *$ ./main.out* to run. Remember to use *& make* again after modifying *cpfile.c;*
2. Modify the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

### 3.1 Device

版本 Windows 11 家庭版  
版本号 21H2  
安装日期 ‎2021/‎11/11  
操作系统版本 22000.318

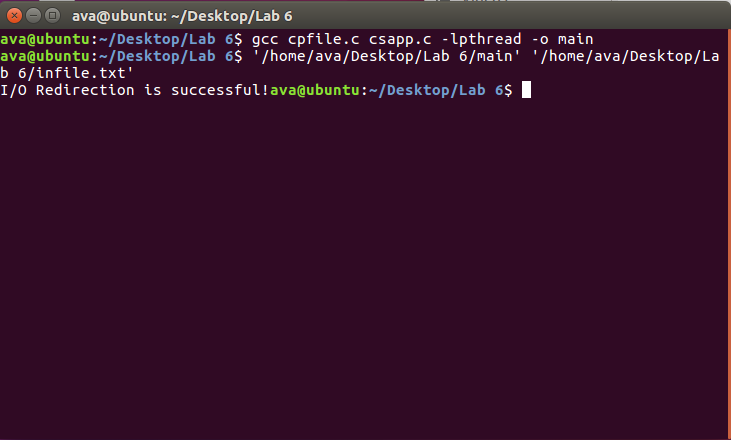
### 3.2 Environment



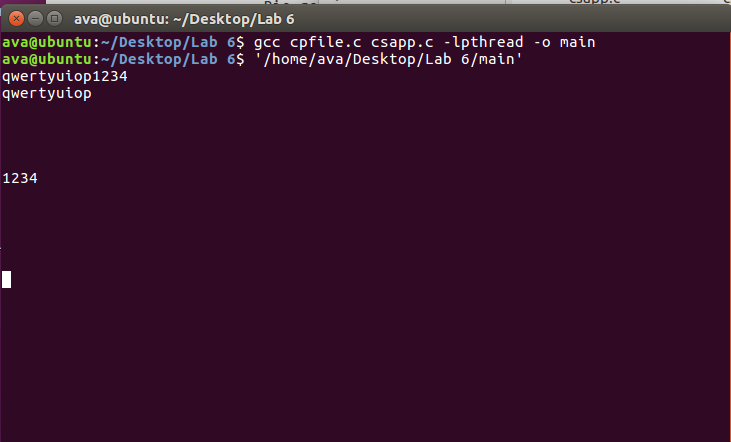
## 4 Results and Analysis

### 4.1Results

The cpfile program takes an optional coommand line argument, if the infile is given, we copy infile to standard output.



Otherwise, we copy standard input to standard output 10 bytes at a time.



### 4.2 Analysis

We use argc to judge whether it have the optional command line argument. If argc equals to 1, it means we only put this program in terminal without any argument, so we just use Rio\_readn and Rio\_writen to copy standard input to standard output 10 bytes at a time. That is step 1.

If argc equals to 2, it means we input the optional command line argument, So we use Open function to open it. If there is no error, we will get an int value greater than or equal to 0. Then we use the dup2 function to make the file pointer point to the file we open instead of standard input.

Then we read the content of the file and print it to the standard output.

## 5 Appendix

/\* All of rio function implementations are obtained from csapp.c \*/
  
   
/\* Please note the usage of wrapper function \*/
  
   
   
   
/\* $begin cpfile \*/
  
   
#include "csapp.h"
  
   
   
   
int main(int argc, char \*\*argv) {
  
   
 int n;
  
   
 rio\_t rio;
  
   
 char buf[MAXLINE];
  
   
 int fd1 = STDIN\_FILENO, fd2;
  
   
 if (argc == 2){
  
   
 fd1 = Open(argv[1], O\_RDONLY, 0);
  
   
 if (fd1 < 0){
  
   
 Rio\_readinitb(&rio, STDIN\_FILENO);
  
   
 while((n = Rio\_readlineb(&rio, buf, MAXLINE)) != 0){
  
   
 Rio\_writen(STDOUT\_FILENO, buf, n);
  
   
 }
  
   
 }
  
   
   
   
 else{
  
   
 fd2 = dup2(fd1,STDIN\_FILENO);
  
   
 Rio\_readinitb(&rio, fd2);
  
   
 while((n = rio\_readlineb(&rio, buf, MAXLINE)) != 0){
  
   
 Rio\_writen(STDOUT\_FILENO, buf, n);
  
   
 }
  
   
 Close(fd1);
  
   
 Close(fd2);
  
   
 }
  
   
   
   
 }
  
   
   
   
 if (argc == 1){
  
   
 Rio\_readinitb(&rio, STDIN\_FILENO);
  
   
 while((n = Rio\_readn(STDIN\_FILENO, buf, 10)) != 0){
  
   
 Rio\_writen(STDOUT\_FILENO, buf, 10);
  
   
 }
  
   
 }
  
   
   
   
 else{
  
   
 exit(-1);
  
   
 }
  
   
   
   
 exit(0);
  
   
   
   
}
  
   
/\* $end cpfile \*/