Lab 3 (Collaborative)

Start Assignment

Due Tuesday by 11:59pm **Points** 15 **Submitting** a file upload

University of Washington Bothell

CSS 503: Systems Programming

Lab 3: Files

Purpose

The purpose of this lab is to help you get familiarized with Linux and C/C++ Standard I/O. You will implement a program that reads a given file using Linux `read()` system call as well as C/C++ `fgetc()`/`fread()` functions.

System Calls

You will use the following Unix and C/C++ Standard I/O functions. Check the specification of each function using **man**.

open: is a Unix system call that opens a given file.

read: is a Unix system call that reads a file into a given buffer.

close: is a Unix system call that closes a given file.

fopen: is a C/C++ standard I/O function that opens a given file.

fgetc: is a C/C++ standard I/O function that reads one byte from a file.

fread: is a C/C++ standard I/O function that reads a file into a given data structure.

fclose: is a C/C++ standard I/O function that closes a give file.

Statement of Work

The code posted at the end is a template for measuring time elapsed to read a given file with Linux `read()` as well as that elapsed to read the same file with C/C++ Standard's `fread()`. The program reads a file name and the number of bytes to read per `read()` or `fread()`.

The command line argument specifies how many bytes should be read with each read() call. Regardless of what the value of this argument is, the program should always read the entire file. **Note:** When the 2nd argument (i.e., #bytes to read) is 1, you must use `fgetc()`.

For this lab, complete the **main()** function so that the program runs and compares the two mechanisms for reading from a file.

What to Turn in

As this is a lab you are required to work on this with one other student in the class. Make sure you turn in only one submission for both students but make sure both names are on the assignment.

- The completed lab3.cpp
- Your execution output
- You can test against the file hamlet.txt on canvas.

Example Execution Output

The following shows execution outputs when reading a file by 512, 1024, 2048, and 4096 bytes (times may vary depending on the machine you are using but the time measurements will be proportional to what you see below)

```
[css503@uw1-320-18 lab3]$ ./lab3 hamlet.txt 512
 Unix read's elapsed time        = 969
 Standard fread's elapsed time    = 230
 [css503@uw1-320-18 lab3]$ ./lab3 hamlet.txt 1024
 Unix read's elapsed time        = 528
 Standard fread's elapsed time    = 219
 [css503@uw1-320-18 lab3]$ ./lab3 hamlet.txt 2048
 Unix read's elapsed time        = 358
 Standard fread's elapsed time    = 219
 [css503@uw1-320-18 lab3]$ ./lab3 hamlet.txt 4096
 Unix read's elapsed time        = 245
 Standard fread's elapsed time    = 222
 [css503@uw1-320-18 lab3]$
```

Lab3 Code:

```
#include <fcntl.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/uio.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <iostream>
```

```
using namespace std;
struct timeval start, end2; //maintain starting and finishing wall time.
void startTimer()
{
   gettimeofday(&start, NULL);
}

void stopTimer(const char *str)
{
   gettimeofday(&end2, NULL);
      cout << str << "'s elapsed time\t= "
      << ((end2.tv_sec - start.tv_sec) * 1000000 + (end2.tv_usec - start.tv_usec))
      << endl;
}

int main(int argc, char *argv[])
{
   if (argc != 3)
   {
      cerr << "usage: lab3 filename bytes" << endl;
      return -1;
   }
   int bytes = atoi(argv[2]);
   if (bytes < 1)
   {
      cerr << "usage: lab3 filename bytes" << endl;
      cerr << "where bytes > 0" << endl;
      return -1;
   }
   char *filename = argv[1];
   char *buf = new char[bytes];

   // linux i/o
```

```
   int fd = open(filename, O_RDONLY);
   if (fd == -1)
   {
      cerr << filename << " not found" << endl;
      return -1;
   }

   startTimer();
   while (read(fd, buf, bytes) > 0);
   stopTimer("Unix read");
   close(fd);
// standard i/o
// write the same functionality as in linux i/o
// but use fopen(), fgetc(), fread(), and fclose( )
// use fgetc() if bytes == 1
return 0;
}
```

Lab 3 Rubric		
Criteria	**Ratings**	**Pts**
Follows Coding Guidelines		5 pts
Compiles		3 pts
Runs		2 pts
Correct Output		5 pts
		Total Points: 15