Leon Zhang
llz2100000
CS 3377.006

Assignment 3

1. Consider the following scenario: You start vim to edit an existing file. You make many changes to the file and then realize that you deleted a critical section of the file early in your editing session. You want to get that section back but do not want to lose all the other changes you made. What would you do?
    a. The first step to do is to create a new buffer using the ':enew' command that and copy the contents of the existing buffer into it. Then, switch back to the original file and go through the undo history until you find the deleted critical section and copy it. Then, paste it into the created buffer which will now have the critical section along with the contents of the original file. Then, copy the contents and paste it back into the original buffer, getting the section back while not losing all the changes you made.

   Assume that your version of vim does not support multiple Undo commands. If you delete a line of text, then delete a second line, and then a third line, which commands would you use to recover the first two lines that you deleted?
    b. To recover the first two lines you deleted, you can use the '#P' commands to get them back. Vim stores deleted lines into registers that can be accessed using those commands, so the most recent being register 1. So in order to get back the first two, do '2P' and '3P' commands to get the lines back.

2. Write your own dup2 function that behaves the same way as the dup2 function described in Section 3.12, without calling the fcntl function. Be sure to handle errors correctly.
    a.

```c
#include <stdio.h>
#include <errno.h>
#include <fcntl.h>

int main(){
    int fd = open( Filename: "test.txt",  OpenFlag: O_RDONLY);
    int fd2 = 10;

    if (dup2( FileHandleSrc: fd,  FileHandleDst: fd2) != -1){
        printf( format: "Duplicated");
    }
    else{
        perror( ErrMsg: "dup2 failed");
    }

    return 0;
}
```

Leon Zhang
llz2100000
CS 3377.006

```c
int dup2(int fd, int fd2){
    if(fd < 0 || fd >= 256 || fd2 < 0 || fd2 >= 256){
        errno = EBADF;
        return -1;
    }
    if (lseek( FileHandle: fd,  Offset: 0,  Origin: SEEK_CUR) == -1 && errno == EBADF){
        return -1;
    }
    if (fd == fd2){
        return fd2;
    }
    close( FileHandle: fd2);

    int tempfd, dupfd = -1;
    tempfd = dup( FileHandle: fd);
    if(tempfd == -1) {
        return -1;
    }

    if(tempfd == fd2){
        return fd2;
    } else{
        dupfd = dup2( fd: tempfd, fd2);
        close( FileHandle: tempfd);
        return dupfd;
    }
}
```

```
C:\Users\leonz\Desktop\HW\UNIX\llz210000-HW3-Code\cmake-build-debug\llz210000_HW3_Code.exe
Duplicated
Process finished with exit code 0
```

3. If you open a file for read–write with the append flag, can you still read from anywhere in the file using lseek? Can you use lseek to replace existing data in the file? Write a program to verify this.
    a. Yes, you can still read from anywhere using lseek, but cannot replace existing data because of the O_APPEND.
    b.

```c
int main() {
    int fd = open( Filename: "test.txt",  OpenFlag: O_RDWR | O_APPEND);
    char buffer[100];
    int bytesRead;
    int offset = 10;

    lseek( FileHandle: fd, offset,  Origin: SEEK_CUR);
    printf( format: "Reading from lseek position %d: ", offset);
    while ((bytesRead = read( FileHandle: fd,  DstBuf: buffer,  MaxCharCount: sizeof(buffer))) > 0) {
        write( Filehandle: STDOUT_FILENO,  Buf: buffer,  MaxCharCount: bytesRead);
    }

    char *data = "this is text input from program\n";
    lseek( FileHandle: fd,  Offset: 10,  Origin: SEEK_CUR);
    write( Filehandle: fd,  Buf: data,  MaxCharCount: strlen( Str: data));
    close( FileHandle: fd);

}
```

Leon Zhang
llz2100000
CS 3377.006

Output:

```
Reading from lseek position 10: ontents of test.txt before any program modifications
...
this is text input from program


Process finished with exit code 0
```

Content of the file:

```
1   original contents of test.txt before any program modifications
2   ...
3   this is text input from program
4   this is text input from program
5   this is text input from program
6
```

You are still able to read from an offset using lseek. However, input from the program to the file appends to the end even when trying to write to an offset using lseek.

4. Modify the program in Figure 4.3 to use stat instead of lstat. What changes if one of the command-line arguments is a symbolic link?

Leon Zhang
llz2100000
CS 3377.006

a.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

void err_ret(const char *msg) {
    perror( ErrMsg: msg);
}

int main(int argc, char *argv[]) {
    int i;
    struct stat buf;
    char *ptr;

    for (i = 1; i < argc; i++) {
        printf( format: "%s: ", argv[i]);
        if (stat( Filename: argv[i], Stat: &buf) < 0) {
            err_ret( msg: "stat error");
            continue;
        }
        if (S_ISREG(buf.st_mode))
            ptr = "regular";
        else if (S_ISDIR(buf.st_mode))
            ptr = "directory";
        else if (S_ISCHR(buf.st_mode))
            ptr = "character special";
        else if (S_ISBLK(buf.st_mode))
            ptr = "block special";
        else if (S_ISFIFO(buf.st_mode))
            ptr = "fifo";
        else if (S_ISLNK(buf.st_mode))
            ptr = "symbolic link";
        else if (S_ISSOCK(buf.st_mode))
            ptr = "socket";
        else
            ptr = "** unknown mode **";

        printf( format: "%s\n", ptr);
    }

    exit( Code: 0);
}
```

Leon Zhang
llz2100000
CS 3377.006

Output:

```
C:\Users\leonz\Desktop\HW\UNIX\llz210000-HW3-Code\figure43.exe ../figure43test/dev/log ../figure43test/etc/logLink
../figure43test/dev/log: regular
../figure43test/etc/logLink: regular
```

b. Here is proof that logLink is a symbolic link:

```
leonzh@Leon-Laptop:/mnt/c/Users/leonz/Desktop/HW/UNIX/figure43test$ ls -l etc
total 0
lrwxrwxrwx 1 leonzh leonzh 7 Mar 24 21:02 logLink -> dev/log
```

c. Changing lstat to stat will make symbolic links that link to a file show 'regular' compared to its lstat output of symbolic link.

5. Create a program named prime.

a.

```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <ctype.h>
#include <math.h>

bool onlyDigits(const char* str){
    for (int i = 0; str[i] != '\0'; i++){
        if(!isdigit( c: str[i])){
            return false;
        }
    }
    return true;
}

bool is_prime(int num){
    if (num <= 1) return false;
    if (num <= 3) return true;
    if (num % 2 == 0 || num % 3 == 0) return false;
    for (int i = 5; i < sqrt( x: num); i += 6){
        if (num % i == 0 || num % (i + 2) == 0)
            return false;
    }
    return true;
}

int makeBigInteger(char *num){
    int sum;
    for(int i = 0; i < strlen( str: num); i++){
        int intVal = num[i] - '0';
        sum += intVal * pow( x: 10,  y: strlen( str: num) - 1 - i);
    }
    return sum;
}
```

```c
int main(){
    char input[256];

    while (fgets( buf: input,  maxCount: sizeof(input),  file: stdin)){
        input[strcspn( str: input,  control: "\n")] = 0;
        if (strcmp(input, "exit") == 0){
            break;
        }
        if (strcmp(input, "0") == 0 || strcmp(input, "1") == 0){
            printf( format: "%s neither\n", input);
        } else if (!onlyDigits( str: input)){
            printf( format: "%s malformed\n", input);
        } else{
            int num = makeBigInteger( num: input);
            if (is_prime(num)){
                printf( format: "%s probably prime\n", input);
            } else{
                printf( format: "%s composite\n", input);
            }
        }
    }

    return 0;
}
```

b. Output:

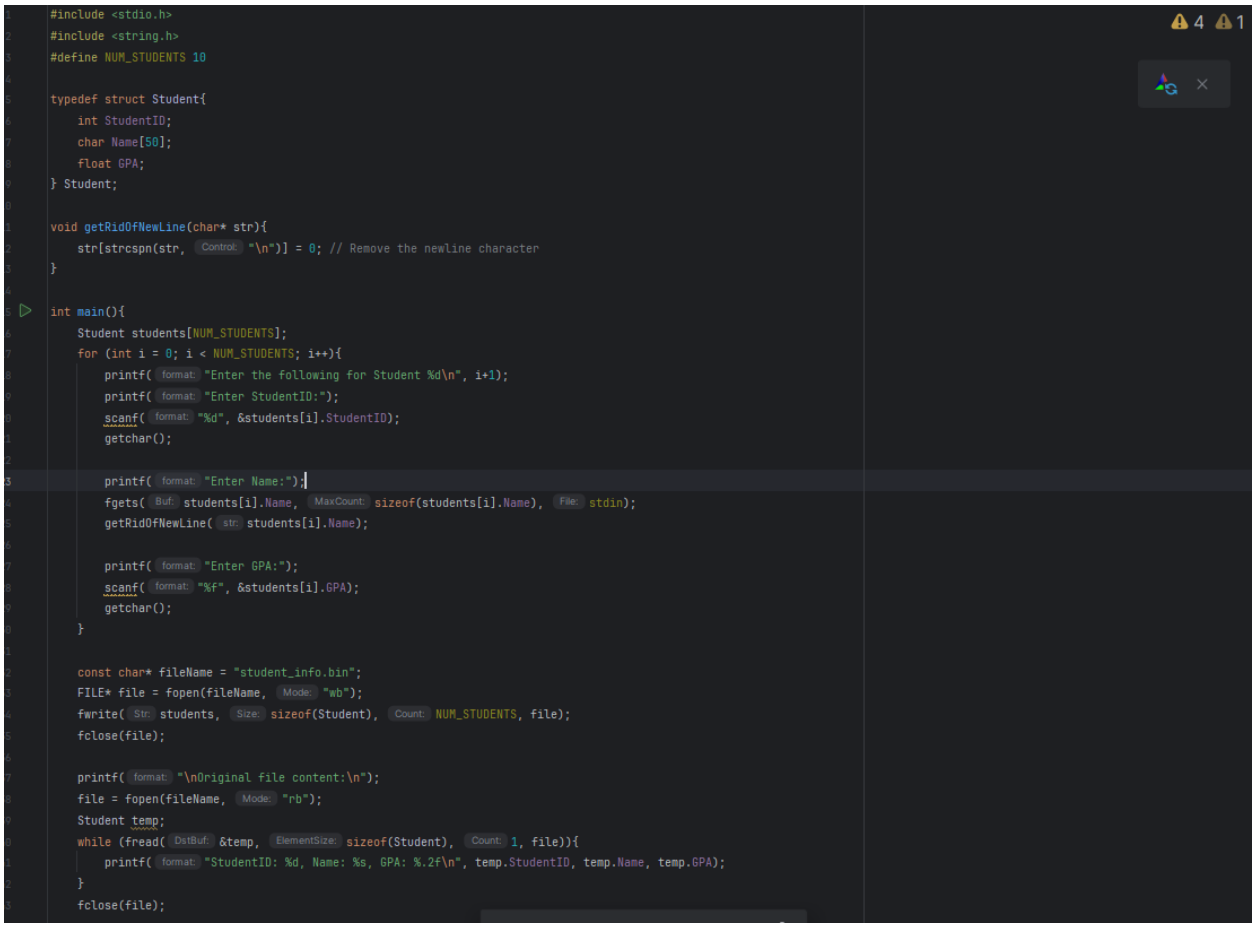
```
C:\Users\leonz\Desktop\HW\UNIX\llz210000-HW3-Code\prime.exe
1
1 neither
0
0 neither
2
2 probably prime
3
3 probably prime
16
16 composite
32890
32890 composite
709
709 probably prime
7883
7883 probably prime
exit

Process finished with exit code 0
```

6. Write a program to get information of 10 students (StudentID, Name, and GPA) from standard input and store them in a binary file. The program should display the content of the file on the screen and copy the file in another file in a reverse order (last record of the original file should be the first record in the second file and so on). The program should also display the content of the original file from 4th record to the end of the file on the screen.

Leon Zhang
llz2100000
CS 3377.006

a.



```c
#include <stdio.h>
#include <string.h>
#define NUM_STUDENTS 10

typedef struct Student{
    int StudentID;
    char Name[50];
    float GPA;
} Student;

void getRidOfNewLine(char* str){
    str[strcspn(str, Control: "\n")] = 0; // Remove the newline character
}

int main(){
    Student students[NUM_STUDENTS];
    for (int i = 0; i < NUM_STUDENTS; i++){
        printf( format: "Enter the following for Student %d\n", i+1);
        printf( format: "Enter StudentID:");
        scanf( format: "%d", &students[i].StudentID);
        getchar();

        printf( format: "Enter Name:");
        fgets( Buf: students[i].Name, MaxCount: sizeof(students[i].Name), File: stdin);
        getRidOfNewLine( str: students[i].Name);

        printf( format: "Enter GPA:");
        scanf( format: "%f", &students[i].GPA);
        getchar();
    }

    const char* fileName = "student_info.bin";
    FILE* file = fopen(fileName, Mode: "wb");
    fwrite( Str: students, Size: sizeof(Student), Count: NUM_STUDENTS, file);
    fclose(file);

    printf( format: "\nOriginal file content:\n");
    file = fopen(fileName, Mode: "rb");
    Student temp;
    while (fread( DstBuf: &temp, ElementSize: sizeof(Student), Count: 1, file)){
        printf( format: "StudentID: %d, Name: %s, GPA: %.2f\n", temp.StudentID, temp.Name, temp.GPA);
    }
    fclose(file);
```

Leon Zhang
llz2100000
CS 3377.006

```
    const char* reversedFileName = "student_info_reversed.bin";
    FILE* original = fopen(fileName, Mode: "rb");
    FILE* copy = fopen( Filename: reversedFileName, Mode: "wb");
    fseek( File: original, Offset: 0, Origin: SEEK_END);
    long fileSize = ftell( File: original);
    for (long i = fileSize - sizeof(Student); i >= 0; i -= sizeof(Student)){
        Student temp;
        fseek( File: original, Offset: i, Origin: SEEK_SET);
        fread( DstBuf: &temp, ElementSize: sizeof(Student), Count: 1, File: original);
        fwrite( Str: &temp, Size: sizeof(Student), Count: 1, File: copy);
    }
    fclose( File: copy);
    fclose( File: original);

    printf( format: "\nReversed file content:\n");
    file = fopen( Filename: reversedFileName, Mode: "rb");
    while (fread( DstBuf: &temp, ElementSize: sizeof(Student), Count: 1, file)){
        printf( format: "StudentID: %d, Name: %s, GPA: %.2f\n", temp.StudentID, temp.Name, temp.GPA);
    }
    fclose(file);

    printf( format: "\nOriginal file content from 4th record:\n");
    file = fopen(fileName, Mode: "rb");
    fseek(file, Offset: (3) * sizeof(Student), Origin: SEEK_SET);
    while (fread( DstBuf: &temp, ElementSize: sizeof(Student), Count: 1, file)){
        printf( format: "StudentID: %d, Name: %s, GPA: %.2f\n", temp.StudentID, temp.Name, temp.GPA);
    }
    fclose(file);

    return 0;
}
```

Leon Zhang
llz2100000
CS 3377.006

b. Output:

```
Enter the following for Student 1
Enter StudentID:12345
Enter Name:Leon Zhang
Enter GPA:4.0
Enter the following for Student 2
Enter StudentID:21345
Enter Name:Noel Gnahz
Enter GPA:2.0
Enter the following for Student 3
Enter StudentID:34567
Enter Name:Bill Bob
Enter GPA:3.5
Enter the following for Student 4
Enter StudentID:98765
Enter Name:Tom Green
Enter GPA:4.0
Enter the following for Student 5
Enter StudentID:67822
Enter Name:Sally Susan
Enter GPA:3.8
Enter the following for Student 6
Enter StudentID:78914
Enter Name:Jill Jane
Enter GPA:2.3
Enter the following for Student 7
Enter StudentID:52873
Enter Name:Chris Chris
Enter GPA:1.9
Enter the following for Student 8
Enter StudentID:17239
Enter Name:Amy Angela
Enter GPA:4.0
Enter the following for Student 9
Enter StudentID:28793
Enter Name:Fred
Enter GPA:2.8
Enter the following for Student 10
Enter StudentID:16239
Enter Name:Molly Mary
2.4
Enter GPA:
```

Leon Zhang
llz2100000
CS 3377.006

```
Original file content:
StudentID: 12345, Name: Leon Zhang, GPA: 4.00
StudentID: 21345, Name: Noel Gnahz, GPA: 2.00
StudentID: 34567, Name: Bill Bob, GPA: 3.50
StudentID: 98765, Name: Tom Green, GPA: 4.00
StudentID: 67822, Name: Sally Susan, GPA: 3.80
StudentID: 78914, Name: Jill Jane, GPA: 2.30
StudentID: 52873, Name: Chris Chris, GPA: 1.90
StudentID: 17239, Name: Amy Angela, GPA: 4.00
StudentID: 28793, Name: Fred , GPA: 2.80
StudentID: 16239, Name: Molly Mary, GPA: 2.40

Reversed file content:
StudentID: 16239, Name: Molly Mary, GPA: 2.40
StudentID: 28793, Name: Fred , GPA: 2.80
StudentID: 17239, Name: Amy Angela, GPA: 4.00
StudentID: 52873, Name: Chris Chris, GPA: 1.90
StudentID: 78914, Name: Jill Jane, GPA: 2.30
StudentID: 67822, Name: Sally Susan, GPA: 3.80
StudentID: 98765, Name: Tom Green, GPA: 4.00
StudentID: 34567, Name: Bill Bob, GPA: 3.50
StudentID: 21345, Name: Noel Gnahz, GPA: 2.00
StudentID: 12345, Name: Leon Zhang, GPA: 4.00

Original file content from 4th record:
StudentID: 98765, Name: Tom Green, GPA: 4.00
StudentID: 67822, Name: Sally Susan, GPA: 3.80
StudentID: 78914, Name: Jill Jane, GPA: 2.30
StudentID: 52873, Name: Chris Chris, GPA: 1.90
StudentID: 17239, Name: Amy Angela, GPA: 4.00
StudentID: 28793, Name: Fred , GPA: 2.80
StudentID: 16239, Name: Molly Mary, GPA: 2.40
```