```
/*
Group B
Zac Bland
A20199624
zac.bland@okstate.edu
10/10/22

My part was doing the menu loop, the pipes between the process and server, and
user
option 1 and 3.

I used a while loop in the client, server and process to accomplish this.
I created an array in process. When the child processes were closing, the
child process would send all its info through the pipe to the parent process.
Then, the parent process would send the process list to the server, and the
server would ask the client which record they would like to chose. The client
would then reply and the server would then ask the process for the list from that
specific record. Once recieved from the process, the server would send the list
to the client and the client would print it out on the screen.

User option 3 would just print the list of book sizes for each respective
process.
They would all keep track of their respective sizes and be sent from the parent
process to the server. Then, the server would send this list to the client.
*/

//CLIENT.C ------------------------------------------------------------------

//This was the menu loop for the client to recieve and send the
//Info to the server.
char process_buff[4000];
while(1){
    //Write user options
    bzero(buffer, sizeof(buffer));
    input[0] = '\0';
    printf("Please select option:\n");
    printf("1. Display the records\n");
    printf("2. Save the records\n");
    printf("3. Display the summary\n");
    printf("4. Exit\n");
    printf(">> ");
    scanf("%s", input);
    printf("\n");

    //Send input to server
```

```
    send(clientSock, input, strlen(input), 0);

    //break if option 4
    if(atoi(input) == 4)
        break;

    //Wait for server response
    memset(process_buff, 0 , sizeof(process_buff));
    bzero(process_buff, sizeof(process_buff));
    recv(clientSock, process_buff, 4000, 0);
    printf("%s\n", process_buff);

    //Go into option 1 if chosen
    if(atoi(input) == 1){
        //input choice
        char record[100];
        printf("Please select a record: \n");
        printf(">> ");
        scanf(" %[^\n]%*c", record);

        //send choice to server
        send(clientSock, record, strlen(record), 0);

        //recieve summary and display it
        char arr[200][200];
        if( recv(clientSock, arr, sizeof(sizeof(char) * 200) * 200, 0) < 0){
            return 1;
        }

        for(int i= 0; i < 200; i++){
            if(strcmp(arr[i], "") != 0)
                printf("%s\n",arr[i]);
        }

        printf("\n");

    }

}

printf("Exiting Server. Goodbye!\n");



//SERVER.C ------------------------------------------------------------------
```

```
//Create Pipes
int fin[2];
int fout[2];
if(pipe(fin) < 0 || pipe(fout) < 0){
    perror("pipe");
    exit(1);
}

//Close unneccesary pipe ends
close(fin[1]);
close(fout[0]);

//wait for processes to finish
char wait_buff[10];
read(fin[0], wait_buff, 10);
send(clientSock, "Ready", 10, 0);

char process_buff[4000];
//Menu Loop
while(1){
    //recieve choice from user
    bzero(buffer, sizeof(buffer));
    memset(buffer, 0 , sizeof(buffer));
    recv(clientSock, buffer, 1024, 0);
    int option = atoi(buffer);

    //write choice to process
    size_t length = strlen( buffer );
    write( fout[1], buffer, length );

    //read response from process
    bzero(process_buff, sizeof(process_buff));
    read(fin[0], process_buff, 4000);

    //send result from process to client
    send(clientSock, process_buff, sizeof(process_buff), 0);

    if(option == 1){
        //wait for process list
        bzero(buffer, sizeof(buffer));
        recv(clientSock, buffer, 1024, 0);

        //write client choice to process
        length = strlen(buffer);
```

```
        write(fout[1], buffer, length);

        //read string array from process
        char arr[200][200];
        if(read(fin[0], arr, sizeof(sizeof(char) * 200) * 200) < 0){
            return 1;
        }

        //write string array to client
        if(write(clientSock, arr, sizeof(sizeof(char) * 200) * 200) < 0){
            return 3;
        }

    }
    else if(option > 3 && option < 1){
        break;
    }

}



//PROCESS.C ----------------------------------------------------------------

//PIPE INTERCOMMUNICATION
close(fin[1]);
close(fout[0]);
close(fd[1]);

//send message to server when ready
write(fout[1], "ready", 10);

char buffer[20];
ssize_t count;
char str[4000];
while(1){
    //READ FROM SERVER
    bzero(buffer, sizeof(buffer));
    do{
        count = read(fin[0], buffer, sizeof(buffer)-1);
    }while(count <= 0);

    buffer[count] = '\0';

    //Do option based on choice from client given by server
```

```
    memset(str,0,strlen(str));
    int option = atoi(buffer);


    if(option == 1){
        // Do option one

        //Get list of processes
        for(int i = 0; i < processes; i++){
            char process_str[100];
            sprintf(process_str, "%s\n", values[i]);
            strcat(str, process_str);
        }
        //send process list through pipe
        size_t length = strlen(str);
        write(fout[1], str, length);

        //read choice from client
        read(fin[0], buffer, sizeof(buffer));

        //find index of process
        int choice;
        for(int i = 0; i< processes; i++){
            if(strcmp(values[i], buffer) == 0){
                choice = i;
                break;
            }
        }

        //copy process results into arr
        char arr[200][200];
        for(int i = 0; i < 200; i++){
            strcpy(arr[i], process_array[choice][i]);
        }

        //send array to server
        if(write(fout[1], arr, sizeof(sizeof(char) * 200) * 200) < 0){
            return 1;
        }

    }
    else if(option == 2){
        sprintf(str, "OPTION 2");
        size_t length = strlen(str);
        write(fout[1], str, length);
```

```
    }
    else if(option == 3){
        for(int i = 0; i < processes; i++){
            char process_str[100];
            sprintf(process_str, "%s : Total books = %d\n", values[i], sizes[i]);
            strcat(str, process_str);
        }
        size_t length = strlen(str);
        write(fout[1], str, length);
    }
    else{
        break;
    }


}
```