**StarID: rc2282qr**

**SOURCE:**

//////////////////////////////////////////////////////////////////////////////////////////////////////////////// Makefile

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

all: interface.c DB.c

gcc -o interface interface.c

gcc -o DB DB.c

//////////////////////////////////////////////////////////////////////////////////////////////////////////////// accountData

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

1234567 102 08/11/21 4.00

1234567 101 08/14/21 14.00

3456787 9873 08/30/21 100.00

1234567 100 08/16/21 35.00

3456787 9874 09/30/21 4.00

12345 1010 09/01/21 34.00

1001001 905 08/14/21 9.00

1001001 903 08/30/21 11.00

12345 1001 09/14/2 16.00

12345 1111 08/24/21 2.00

1234 1112 08/31/21 44.00

1001001 902 09/25/21 19.00

//////////////////////////////////////////////////////////////////////////////////////////////////////////////// interface.c

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#define BUFFER\_SIZE 2048

void clearBuffer(char \* input);

void errcheck(int err);

int main()

{

int DBreturn = -1;

int err;

int Status=0;

int writeError, readError;

int toDB[2];

int toInterface[2];

char args1[50];

char args2[50];

char ReadBuffer[BUFFER\_SIZE+1]; // Input.

char WriteBuffer[BUFFER\_SIZE+1]; // Cmd.

int DBpid;

// Create Pipes.

pipe(toDB);

pipe(toInterface);

// fork child process.

DBpid = fork();

if(DBpid == -1){

printf("There was an error on fork. Error Number: %d\n", errno);

exit(-1);

}

else if (DBpid == 0){

// In child process

// Close unused pipe ends

close(toDB[1]);

close(toInterface[0]);

// Assign args

sprintf(args1,"%d", toDB[0]);

sprintf(args2,"%d", toInterface[1]);

//Execute DB Program

err = execl("./DB",args1,args2,NULL);

if(err == -1){

printf("It looks like there was an issue with starting DB \n");

printf("Error Number :%d", errno);

exit(-1);

}

// DB is initialized.

}else{ // in Parent process

//Close unused pipe ends

close(toDB[0]);

close(toInterface[1]);

// Pipes are established

// Expecting message from DB with a success.

printf("Reading from DB\n\n"); //debugging

readError = read(toInterface[0],ReadBuffer, 99);

printf("Response: %s", ReadBuffer);

clearBuffer(ReadBuffer);

errcheck(readError);

// Begin menu loop.

int Done = 0;

while(Done == 0 ){

printf("Please supply a command( 'account, id' | list | date,mm/dd/yy | exit)\n:");

// Recieve input from user, send to DB.

clearBuffer(WriteBuffer);

scanf("%s", WriteBuffer);

writeError = write(toDB[1], WriteBuffer, BUFFER\_SIZE);

errcheck(writeError);

// Read Response from DB.

readError = read(toInterface[0], ReadBuffer, BUFFER\_SIZE);

errcheck(readError);

printf("Response: %s\n", ReadBuffer);

// Check for exit return by DB.

if(atoi(ReadBuffer) == 1){

Done = 1;

}

clearBuffer(ReadBuffer);

}

// Wait for child process to finish.

err = waitpid(-1, &Status ,0);

errcheck(err);

printf("Goodbye!\n");

}// End of Parent Instructions.

return 0; //Exit

}

void clearBuffer(char \*input){

int i = 0;

for(i; i < BUFFER\_SIZE; i++){

\*(input+i) = '\0';

}

}

void errcheck(int err){

if(err == -1){

printf("It looks like there was an issue...");

printf("Error Number :%d\n", errno);

exit(-1);

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////// DB.c

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#define RL 1000 // Record Limit.

#define BUFFER\_SIZE 2048 // Maximum BufferSize

// Struct declaration for records.

typedef struct

{

long int id; // account number

int checkNumber; // check Number

char date[9]; // Date mm/dd/yy'\0'

float amount; // Dollar amount xx.xx

}record;

// Functions

void clearBuffer(char \*input);

void sort(record \*recs, int count);

void printRecords(record \*records, int count);

void errcheck(int err);

float getTotal(long int account, const record \*recs, int count);

float getDateTotal(char \* date, const record \* recs, int count );

// Expects 2 integers for program execution.

int main(int argc,char \*\*argv){

// Initializations for communication

char writeBuffer[2048];

char readBuffer[2048];

char Cmd[100];

int toDB[2];

int toInterface[2];

int writeError;

int readError;

// Initialize for file read.

FILE \*fd;

// assign pipe values from start;

sscanf(argv[0],"%d",&toDB[0]);

sscanf(argv[1],"%d",&toInterface[1]);

// Open the accountData file.

fd = fopen("accountData","r");

//Initialize array to hold records.

record records[RL+1];

int i =0;

int numRecords;

int err;

// read records from file.

for(i=0; !feof(fd) && i<RL; i++){

err = fscanf(fd,"%ld %d %s %f \n",&records[i].id, &records[i].checkNumber, records[i].date, &records[i].amount);

if(err ==-1){

printf("Error in processing file... Error:%d \n",errno);

exit(-1);

}

printf("[%d]:%ld %d %s %.2f \n",i,records[i].id, records[i].checkNumber, records[i].date, records[i].amount);

numRecords++;

}

// write records to buffer

sort(records, numRecords);

clearBuffer(writeBuffer);

// records have been read in, sort them for further displays

sprintf(writeBuffer,"All records added successfully.\n - DB Initialized - \n");

writeError = write(toInterface[1],writeBuffer,99);

errcheck(writeError);

// ready to recieve input from interface.

// Create call structure for different calls from interface.

int Done = 0;

while(Done == 0){

readError = read(toDB[0],readBuffer, BUFFER\_SIZE);

clearBuffer(writeBuffer);

errcheck(readError);

if(strncmp(readBuffer,"\0",1) == 0){wait(10);}

else if(strncmp(readBuffer,"account",7)== 0 ){

long int account;

float amount;

char accountch[12];

int i;

int j = 0;

// extract account number.

for (i = 8; i < 20; i++){

accountch[j++] = readBuffer[i];

}

sscanf(accountch,"%ld",&account);

clearBuffer(readBuffer);

// Total up the amount for the given account number.

amount = getTotal(account, records, numRecords);

// Write account amount to the buffer in the formatted string.

clearBuffer(writeBuffer);

sprintf(writeBuffer,"total for %ld = %.2f.\n",account,amount);

writeError = write(toInterface[1],writeBuffer,99);

errcheck(writeError);

}

else if (strncmp(readBuffer,"list",4)== 0 ){

clearBuffer(readBuffer);

int done = 0;

if(numRecords ==0){done = 1;}

int i=0;

int offset = 0;

// loop through records each to buffer.

clearBuffer(writeBuffer);

for(i = 0; (i < numRecords) && (done != 1); i++){

offset += sprintf(writeBuffer+offset,"[%d]:%ld %d %s %.2f\n",i,records[i].id ,records[i].checkNumber

, records[i].date, records[i].amount);

}

// Write entire buffer.

writeError = write(toInterface[1],writeBuffer,BUFFER\_SIZE);

errcheck(writeError);

}

else if(strncmp(readBuffer, "date",4) == 0){

float amount;

char date[12];

int i;

int j = 0;

// Extract date from the buffer.

for (i = 5; i < 17; i++){

date[j++] = readBuffer[i];

}

clearBuffer(readBuffer);

// get total for the given date.

amount = getDateTotal(date, records, numRecords);

// write total to buffer.

clearBuffer(writeBuffer);

sprintf(writeBuffer,"total for %s = %.2f.\n",date,amount);

writeError = write(toInterface[1],writeBuffer,99);

errcheck(writeError);

}

else if(strncmp(readBuffer, "exit",4) == 0){

// Command from interface is exit.

// send 1 down the pipe to terminate parent loop.

clearBuffer(writeBuffer);

sprintf(writeBuffer,"%d",1);

writeError = write(toInterface[1],writeBuffer,99);

errcheck(writeError);

Done = 1;

}

else{

// Unknown command, clear buffer and try again to recieve input from the user.

clearBuffer(writeBuffer);

sprintf(writeBuffer,"DB did not understand the command... \n");

writeError = write(toInterface[1],writeBuffer,99);

errcheck(writeError);

}

clearBuffer(readBuffer);

}

return 0;

}

////////////////////////////////////////////////////////////////////////////////

// Function list

////////////////////////////////////////////////////////////////////////////////

void clearBuffer(char \*input){

int i = 0;

for(i; i < BUFFER\_SIZE; i++){

\*(input+i) = '\0';

}

}

void sort( record \*recs, int count){

// Simple sort takes in pointer to array of record.

int i = 0;

int j = 0;

int position;

for ( i = 0 ; i < (count -1) ; i++ )

{

position = i;

for ( j = i + 1 ; j < count ; j++ )

{

if (recs[position].id > recs[j].id ) // If Id is greater, swap.

{

position = j;

}

if ( recs[position].id == recs[j].id ){ // sort by date, id's are equal

int k = 0;

char y[2],d[2],m[2];

char jy[2],jd[2],jm[2];

int x=0,w=0,z=0;

// First, extract the year, day, and month from the date.

for (k = 0; k < 8;k++){

if(k == 0||k ==1){// Month

m[x]= recs[position].date[k];

jm[x]= recs[j].date[k];

x++;

}

if(k == 3 ||k == 4){// Day

d[w]= recs[position].date[k];

jd[w]= recs[j].date[k];

w++;

}

if(k == 6 ||k == 7){// Year

y[z]= recs[position].date[k];

jy[z]= recs[j].date[k];

z++;

}

}

if(strncmp(y,jy,2)>0){// Compare if y > jy, will evaluate true.

position = j;

break;

}

else if (strncmp(y,jy,2)==0){// years equal, continue.

if(strncmp(m,jm,2)>0){

position = j;

break;

}

else if(strncmp(m,jm,2)==0){// months equal, continue.

if(strncmp(d,jd,2)>0){

position = j;

}

}

}

}

}

// swap actual struct array elements.

if ( position != i )

{

record temp;

temp = recs[i];

recs[i] = recs[position];

recs[position] = temp;

}

}

}

void printRecords(record \*records, int count){

int i = 0;

for(i; i < count; i++){

printf("%ld %d %s %.2f \n",records[i].id, records[i].checkNumber, records[i].date, records[i].amount);

}

}

// Wrote function to minimize the space by this check.

void errcheck(int err){

if(err == -1){

printf("It looks like there was an issue...");

printf("Error Number :%d\n", errno);

exit(-1);

}

}

// Takes and account number and accumulates the total for the give date.

float getTotal(long int account, const record \* recs, int count ){

float total = 0;

int i;

for(i=0; i< count;i++){

if(recs[i].id == account){

total+=recs[i].amount;

}

}

return total;

}

// Takes in a date: mm/dd/yy, and accumulates the total for the given date.

float getDateTotal(char \* date, const record \* recs, int count ){

float total = 0;

int i;

for(i=0; i< count;i++){

if(strncmp(recs[i].date,date,8)==0){

total+=recs[i].amount;

}

}

return total;

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

**OUTPUT:**

[rc2282qr@csci4 ~]$ cc -o interface interface.c

[rc2282qr@csci4 ~]$ make interface

make: `interface' is up to date.

[rc2282qr@csci4 ~]$ cc -o DB DB.c

[rc2282qr@csci4 ~]$ ./interface

Reading from DB

[0]:1234567 102 08/11/21 4.00

[1]:1234567 101 08/14/21 14.00

[2]:3456787 9873 08/30/21 100.00

[3]:1234567 100 08/16/21 35.00

[4]:3456787 9874 09/30/21 4.00

[5]:12345 1010 09/01/21 34.00

[6]:1001001 905 08/14/21 9.00

[7]:1001001 903 08/30/21 11.00

[8]:12345 1001 09/14/2 16.00

[9]:12345 1111 08/24/21 2.00

[10]:1234 1112 08/31/21 44.00

[11]:1001001 902 09/25/21 19.00

Response: Please supply a command( 'account, id' | list | date,mm/dd/yy | exit)

:account,1001001