Andrew Michienzi

CIS 452

Project 1

**Design Documentation**

My program is able to sort 4 files assuming the files are named file\_(number).dat. The program operates with a struct called threadArgs struct. It’s named like this because I originally was using threads because I don’t know how to read directions.

struct threadArgs the following members.

* **int threadNumber** This is used to determine how large the tree is going to be and is then used to appropriately name the sorter which will then appropriately read from the correct file.
* **int open** lets the parent know if the child’s still is writing to the pipe.
* **struct Customer nextCustomer** is the customer that the parent child pull from the child process. It’s also used in the sorter to grab the next customer.
* **int fd[2]** is the file descriptor to link a parent to its child process.
* **int isMaster** is used in the writeToParent function used by the Master and Merger processes. It discerns between the master process and the merger processes.
* **int treeSizeArg** is used to determine the size of the tree that is created. When a merger’s treeSizeArg == 1, then it knows to create a sorter, else it will create another merger, decrementing the treeSizeArg.

struct Customer has the following members:

* transactionNum
* customerNum
* Amount

The program starts by determing the treeSizeArg of the master.

* 2^treeSizeArg = number of files to read.

This is calculated, then decremented and sent to the merger. This number will be the driving force behind the actual size of the tree.

The program starts with creating 2 merger processes from the master process. If the treeSizeArg == 1, 2 sorters will be created. Otherwise, the merger will spawn 2 more mergers and send down a decremented treeSizeArg. Creating the mergers and sorters also sets up the pipe and puts the file descriptor in its child’s args.

When a sorter process reads from a file, it finds “file\_(int).dat” based on its sorter number. It then creates a customer from each line and puts it into a customer array. It then takes that array and sorts it using bubble sort. After the array is sorted, it writes to the pipe between it and its parent one customer at a time. When the child is done writing, it closed the file descriptor and exits.

The Merger process creates two sorters and then reads from the 2 sorter children. It uses the open flag and the ready flag in its children’s args to decide which pipe to read from. The ready flag is flipped on when the pipe is read from and the open flag is flipped off once the child closes the pipe. Once it has read from both children, it will choose the customer with the smallest customer number between the 2 children. It chooses this customer and writes the customer to its parent, and then reads from that pipe again. If the pipe is closed by the child, it will read from the other pipe until that pipe is exhausted.

When one pipe closes, it will find the open pipe and read from it, closing both pipes when it realizes its done. When both children’s pipes are closed and the master has read its last file, it will close.

The Master process does the same thing as the merger, except for instead of writing to a parent, which it obviously cannot do, it prints the customer to the screen.

C source Code

#include <pthread.h>

#include <stdio.h>

#include <errno.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#define false 0

#define true 1

#define READ 0

#define WRITE 1

struct Customer

{

int transactionNum;

int customerNum;

double amount;

};

struct threadArgs

{

int threadNumber;

int ready;

int open;

struct Customer nextCustomer;

int fd[2];

int isMaster;

int treeSizeArg;

};

void\* merger(void\*);

void\* sorter(void\*);

void\* createMerger(struct threadArgs \*mergerArgs);

void\* createSorter(struct threadArgs \*sorterArgs);

void\* readDataFile(struct Customer \*customers, int sNumber);

void\* sortCustomers(struct Customer \*customers, int \* fileSize);

void\* printCustomers(struct Customer \*customers, int fileSize);

void\* printCustomer(struct Customer \*customer);

int countLines (int sNumber, int \* fileSize);

void writeToParent(struct threadArgs \* parentArgs, int \* fileSize);

void readFromChild(struct threadArgs \* childArgs);

int processInformation(struct threadArgs \* childArgs1, struct threadArgs \* childArgs2, struct threadArgs \* parentArgs);

void findTreeSizeArgs(int numOfFiles, int \* treeSizeArg);

int power(int base, int exp);

struct Customer\* getNextCustomer(struct threadArgs \* args1, struct threadArgs \* args2);

/\*

\* A file sorter with ability to sort 4 data files based on customer number from smallest customer number to largest. The sorter processes use bubble sort to sort the customer numbers, then send one customer up at a time to the merger. The merger has two children which it evaluates the 2 customers that are passed up to it and finds the smallest customer number, sending it to the master. The master again has 2 children that it finds the smallest customer number between the 2 and prints the smallest.

\*/

int main(int argc, char\* argv[])

{

//int status;

struct threadArgs childArgs1 = {1, 1, 1};

struct threadArgs childArgs2 = {2, 1, 1};

struct threadArgs masterArgs = {0, 1, 1};

masterArgs.isMaster = 1;

childArgs1.isMaster = 0;

childArgs1.isMaster = 0;

int treeSizeArg;

if(argc == 2)

{

int numOfFiles = atoi(argv[1]);

findTreeSizeArgs(numOfFiles, &treeSizeArg);

}

else

{

treeSizeArg = 1;

}

childArgs1.treeSizeArg = treeSizeArg;

childArgs2.treeSizeArg = treeSizeArg;

createMerger(&childArgs1);

createMerger(&childArgs2);

int fileSize;

fileSize = processInformation(&childArgs1, &childArgs2, &masterArgs);

printf("\n\nTotal Customers = %d\n", fileSize);

return(0);

}

/\*

\* This function creates a merger process and sends it to the merger function

\*/

void\* createMerger(struct threadArgs \*mergerArgs)

{

pid\_t pid;

if(pipe((\*mergerArgs).fd) < 0)

{

perror("plumbing error");

exit(1);

}

if((pid=fork())<0)

{

perror("fork failed");

exit(1);

}

if(pid == 0)

{

//child

close ((\*mergerArgs).fd[READ]);

merger(mergerArgs);

exit(0);

}

else

{

//parent

close((\*mergerArgs).fd[WRITE]);

}

return(0);

}

/\*

\* This function runs the merger processes and creates 2 sorters and then runs the processing information

\*/

void\* merger(void\* arg)

{

struct threadArgs \* parentArgs = (struct threadArgs \*)arg;

int \* mNumber = &parentArgs->threadNumber;

printf("Merger %d created\n", \*mNumber);

struct threadArgs childArgs1 = {0, 1, 1};

struct threadArgs childArgs2 = {0, 1, 2};

childArgs1.isMaster = 0;

childArgs2.isMaster = 0;

childArgs1.threadNumber = (((\*mNumber)\*2)-1);

childArgs2.threadNumber = ((\*mNumber)\*2);

if((\*parentArgs).treeSizeArg == 1)

{

createSorter(&childArgs1);

createSorter(&childArgs2);

}

else

{

childArgs1.treeSizeArg = (parentArgs->treeSizeArg - 1);

childArgs2.treeSizeArg = (parentArgs->treeSizeArg - 1);

createMerger(&childArgs1);

createMerger(&childArgs2);

}

processInformation(&childArgs1, &childArgs2, parentArgs);

return(0);

}

/\*

\* creates a sorter process

\*/

void\* createSorter(struct threadArgs \*sorterArgs)

{

pid\_t pid;

if(pipe((\*sorterArgs).fd) < 0)

{

perror("plumbing error");

exit(1);

}

if((pid=fork())<0)

{

perror("fork failed");

exit(1);

}

if(pid == 0)

{

//child

close ((\*sorterArgs).fd[READ]);

sorter(sorterArgs);

exit(0);

}

else

{

//parent

close((\*sorterArgs).fd[WRITE]);

}

return(0);

}

/\*

\*Counts the lines in the necessary file, puts the customers from the file in an array, sorts the customer and then writes one customer at a time to it's parent merger

\*/

void\* sorter(void\* arg)

{

int couldFindFile;

struct threadArgs \* mergerArgs = (struct threadArgs \*)arg;

//sleep(30);

int \* sNumber = &mergerArgs->threadNumber;

printf("Sorter %d created\n", \*sNumber);

int fileSize = 0;

printf("%d counting lines\n", \*sNumber);

couldFindFile = (int)countLines (\*sNumber, &fileSize);

if(!couldFindFile)

{

close((\*mergerArgs).fd[WRITE]);

return 0;

}

struct Customer \* customers = malloc(fileSize\*sizeof(struct Customer));

printf("%d reading file\n", \*sNumber);

readDataFile(customers, \*sNumber);

printf("%d sorting file\n", \*sNumber);

sortCustomers(customers, &fileSize);

//printCustomers(customers, fileSize);

int i;

printf("%d sending files\n", \*sNumber);

for(i = 0; i < fileSize; i++)

{

write((\*mergerArgs).fd[WRITE], (customers + i), sizeof(struct Customer));

printf("Sorter %d writing to parent\n", (\*mergerArgs).threadNumber);

}

close((\*mergerArgs).fd[WRITE]);

printf("Sorter %d closed\tFile Size = %d\n",\*sNumber, fileSize);

return(0);

}

/\*

\* counts the number of lines in the file

\*/

int countLines (int sNumber, int \* fileSize)

{

char \* line = NULL;

FILE \* fp;

size\_t len = 0;

ssize\_t read;

char \*file = malloc(12\*sizeof(char));

sprintf(file, "file\_%d.dat", sNumber);

fp = fopen(file, "r");

if(fp==NULL)

{

printf("Could not find file\_%d.dat\n", sNumber);

fflush(stdout);

return 0;

}

while((read = getline(&line, &len, fp)) != -1)

{

(\*fileSize)++;

}

fclose(fp);

if(line)

free(line);

return (1);

}

/\*

\* Reads information from file and then puts said information into the customer array

\*/

void\* readDataFile(struct Customer \*customers, int sNumber)

{

char \* line = NULL;

FILE \* fp;

size\_t len = 0;

ssize\_t read;

int linePtr;

char \*file = malloc(12\*sizeof(char));

sprintf(file, "file\_%d.dat", sNumber);

fp = fopen(file, "r");

if(fp==NULL)

{

printf("file failure\n");

fflush(stdout);

exit(EXIT\_FAILURE);

}

while((read = getline(&line, &len, fp)) != -1)

{

//Create a customer from a line in data

char \* delim;

delim = strtok(line, " ");

int column = 0;

while(delim != NULL)

{

if(column == 0)

{ //Trans Num

sscanf(delim, "%d", &((customers + linePtr)->transactionNum));

}

else if(column == 1)

{ //Cust Num

sscanf(delim, "%d", &(customers + linePtr)->customerNum);

}

else

{ //Amount

sscanf(delim, "%lf", &(customers + linePtr)->amount);

}

column++;

delim = strtok(NULL, " ");

}

linePtr++;

}

fclose(fp);

if(line)

free(line);

//sleep(10);

return (0);

}

/\*

\* Sorts customers in customer array using bubble sort

\*/

void\* sortCustomers(struct Customer \*customers, int \* fileSize)

{

struct Customer temp;

int i, j, swaps;

for(i=0; i<(\*fileSize)-2; i++)

{

swaps = 0;

for (j=0; j<(\*fileSize)-1; j++)

{

if((\*(customers+j)).customerNum > (\*(customers+j+1)).customerNum)

{

temp = \*(customers+j);

\*(customers + j) = \*(customers+j+1);

\*(customers + j + 1) = temp;

swaps++;

}

}

if(swaps == 0){

break;

}

}

return(0);

}

/\*

\* Prints a customer

\*/

void\* printCustomers(struct Customer \*customers, int fileSize)

{

int i;

for(i = 0; i < fileSize; i++)

{

printf("customerNum = %d\n",(\*(customers+i)).customerNum);

}

return(0);

}

/\*

\* Prints array of customers FOR TESTING

\*/

void\* printCustomer(struct Customer \* customer)

{

printf("transactionNum = %d, customerNum = %d, amount = %lf\n", (\*customer).transactionNum, (\*customer).customerNum, (\*customer).amount);

return(0);

}

/\*

\* Takes in 2 threadArgs. Returns the customer in the args with the smallest customer Num. The ready allows for reading of the customer that was used in the function processInformation

\*/

struct Customer \* getNextCustomer(struct threadArgs \* args1, struct threadArgs \* args2)

{

if((\*args1).nextCustomer.customerNum < (\*args2).nextCustomer.customerNum)

{

//args1

(\*args1).ready = 1;

(\*args2).ready = 0;

return &args1->nextCustomer;

}

else

{

//args2

(\*args2).ready = 1;

(\*args1).ready = 0;

return &args2->nextCustomer;

}

return(0);

}

/\*

\* Processes information from child args. It will read while children are writing and send the appropriate customer up to the parent arguments. Master and Merger processes use this function

\*/

int processInformation(struct threadArgs \* childArgs1, struct threadArgs \* childArgs2, struct threadArgs \* parentArgs)

{

int fileSize = 0;

do

{

//sleep(1);

readFromChild(childArgs1);

readFromChild(childArgs2);

struct Customer \* cust = getNextCustomer(childArgs1, childArgs2);

parentArgs->nextCustomer = \*cust;

if(childArgs1->open || childArgs2->open)

writeToParent(parentArgs, &fileSize);

} while((\*childArgs1).open && (\*childArgs2).open);

while((\*childArgs1).open && !(\*childArgs2).open)

{

(\*childArgs1).ready = 1;

readFromChild(childArgs1);

parentArgs->nextCustomer = (\*childArgs1).nextCustomer;

if(childArgs1->open)

writeToParent(parentArgs, &fileSize);

}

close((\*childArgs1).fd[READ]);

while((\*childArgs2).open && !(\*childArgs1).open)

{

(\*childArgs2).ready = 1;

readFromChild(childArgs2);

parentArgs->nextCustomer = (\*childArgs2).nextCustomer;

if(childArgs2->open)

writeToParent(parentArgs, &fileSize);

}

close((\*childArgs2).fd[READ]);

return fileSize;

}

/\*

\* This function reads from the pipe of a child process. Used by Master and Merger processes

\*/

void readFromChild(struct threadArgs \* childArgs)

{

if((\*childArgs).ready)

(\*childArgs).open = read((\*childArgs).fd[READ], &childArgs->nextCustomer, sizeof(struct Customer));

}

/\*

\* This process writes a customer to the parent process. Used by Merger and Master processes

\*/

void writeToParent(struct threadArgs \* parentArgs, int \* fileSize)

{

if((\*parentArgs).isMaster)

{

(\*fileSize)++;

printCustomer(&(\*parentArgs).nextCustomer);

}

else

{

printf("Merger %d writing to parent\n", (\*parentArgs).threadNumber);

write((\*parentArgs).fd[WRITE], &(\*parentArgs).nextCustomer, sizeof(struct Customer));

(\*fileSize)++;

}

}

/\*

\* This function takes the amount of files needed to sort and determines treeSizeArg, which is stored in the Customer struct and is used to determine how many mergers will be needed.

\*/

void findTreeSizeArgs(int numOfFiles, int \* treeSizeArg)

{

(\*treeSizeArg) = 1;

int p;

p = power(2, 1);

while(numOfFiles > p)

{

(\*treeSizeArg)++;

p = power(2, (\*treeSizeArg));

}

(\*treeSizeArg)--;

}

/\*

\* Got from stack overflow

\* stackoverflow.com/questions/213042/how-do-you-do-exponentation-in-c

\* user: ephemient

\* Probably could have figured this out on my own, but the internet

\*/

int power(int base, int exp)

{

if(exp == 0)

return 1;

else if(exp % 2)

return base \* power(base, exp - 1);

else

{

int temp = power(base, exp/2);

return temp \* temp;

}

}

Output

$./a.out 8

Merger 1 created

Merger 1 created

Sorter 1 created

1 counting lines

1 reading file

1 sorting file

1 sending files

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 writing to parent

Sorter 1 closed File Size = 22

Merger 1 created

Merger 2 created

Sorter 4 created

4 counting lines

4 reading file

4 sorting file

4 sending files

Sorter 4 writing to parent

Sorter 4 writing to parent

Sorter 4 writing to parent

Sorter 4 writing to parent

Sorter 4 writing to parent

Sorter 4 writing to parent

Sorter 4 closed File Size = 6

Merger 2 created

Merger 3 created

Sorter 5 created

5 counting lines

5 reading file

5 sorting file

5 sending files

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 writing to parent

Sorter 5 closed File Size = 15

Merger 1 created

Merger 2 created

Sorter 3 created

3 counting lines

3 reading file

3 sorting file

3 sending files

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 writing to parent

Sorter 3 closed File Size = 25

Merger 1 created

Merger 2 created

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 created

Merger 3 created

Sorter 6 created

6 counting lines

6 reading file

6 sorting file

6 sending files

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 writing to parent

Sorter 6 closed File Size = 40

Merger 2 created

Merger 4 created

Sorter 7 created

7 counting lines

7 reading file

7 sorting file

7 sending files

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 writing to parent

Sorter 7 closed File Size = 15

Merger 2 created

Merger 3 created

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 3 writing to parent

Merger 1 created

Merger 1 created

Sorter 2 created

2 counting lines

2 reading file

2 sorting file

2 sending files

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 writing to parent

Sorter 2 closed File Size = 35

Merger 1 created

Merger 1 created

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 2 created

Merger 4 created

Sorter 8 created

8 counting lines

8 reading file

8 sorting file

8 sending files

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 writing to parent

Sorter 8 closed File Size = 42

Merger 2 created

Merger 4 created

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 4 writing to parent

Merger 1 created

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 1 writing to parent

Merger 2 created

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

Merger 2 writing to parent

transactionNum = 19, customerNum = 1002, amount = 56.470000

transactionNum = 15, customerNum = 1006, amount = 143.280000

transactionNum = 1, customerNum = 1007, amount = 64.890000

transactionNum = 17, customerNum = 1016, amount = 83.200000

transactionNum = 25, customerNum = 1021, amount = 117.700000

transactionNum = 10, customerNum = 1021, amount = 54.370000

transactionNum = 6, customerNum = 1026, amount = 31.730000

transactionNum = 14, customerNum = 1030, amount = 17.950000

transactionNum = 7, customerNum = 1031, amount = 21.480000

transactionNum = 7, customerNum = 1031, amount = 90.950000

transactionNum = 13, customerNum = 1034, amount = 69.140000

transactionNum = 2, customerNum = 1038, amount = 98.700000

transactionNum = 17, customerNum = 1041, amount = 24.270000

transactionNum = 33, customerNum = 1046, amount = 7.800000

transactionNum = 14, customerNum = 1047, amount = 106.170000

transactionNum = 15, customerNum = 1050, amount = 26.430000

transactionNum = 5, customerNum = 1051, amount = 103.830000

transactionNum = 11, customerNum = 1055, amount = 114.720000

transactionNum = 19, customerNum = 1057, amount = 131.430000

transactionNum = 9, customerNum = 1057, amount = 92.870000

transactionNum = 22, customerNum = 1067, amount = 131.740000

transactionNum = 12, customerNum = 1079, amount = 59.800000

transactionNum = 26, customerNum = 1081, amount = 15.270000

transactionNum = 4, customerNum = 1085, amount = 41.560000

transactionNum = 3, customerNum = 1087, amount = 16.710000

transactionNum = 24, customerNum = 1095, amount = 6.370000

transactionNum = 1, customerNum = 1095, amount = 132.410000

transactionNum = 4, customerNum = 1098, amount = 105.960000

transactionNum = 20, customerNum = 1099, amount = 92.430000

transactionNum = 22, customerNum = 1103, amount = 100.630000

transactionNum = 7, customerNum = 1107, amount = 108.980000

transactionNum = 2, customerNum = 1111, amount = 59.360000

transactionNum = 11, customerNum = 1115, amount = 119.610000

transactionNum = 7, customerNum = 1119, amount = 130.300000

transactionNum = 12, customerNum = 1119, amount = 108.370000

transactionNum = 11, customerNum = 1124, amount = 60.660000

transactionNum = 8, customerNum = 1129, amount = 13.870000

transactionNum = 6, customerNum = 1135, amount = 140.300000

transactionNum = 15, customerNum = 1023, amount = 38.330000

transactionNum = 14, customerNum = 1137, amount = 74.590000

transactionNum = 28, customerNum = 1139, amount = 83.660000

transactionNum = 16, customerNum = 1147, amount = 71.800000

transactionNum = 18, customerNum = 1149, amount = 22.270000

transactionNum = 13, customerNum = 1153, amount = 47.430000

transactionNum = 31, customerNum = 1157, amount = 62.870000

transactionNum = 17, customerNum = 1158, amount = 119.830000

transactionNum = 33, customerNum = 1162, amount = 122.720000

transactionNum = 5, customerNum = 1166, amount = 138.220000

transactionNum = 1, customerNum = 1166, amount = 121.580000

transactionNum = 30, customerNum = 1166, amount = 91.280000

transactionNum = 13, customerNum = 1172, amount = 17.620000

transactionNum = 21, customerNum = 1176, amount = 92.700000

transactionNum = 41, customerNum = 1177, amount = 117.900000

transactionNum = 8, customerNum = 1180, amount = 83.700000

transactionNum = 22, customerNum = 1183, amount = 143.370000

transactionNum = 5, customerNum = 1183, amount = 24.600000

transactionNum = 14, customerNum = 1187, amount = 147.980000

transactionNum = 20, customerNum = 1188, amount = 149.730000

transactionNum = 15, customerNum = 1193, amount = 71.460000

transactionNum = 5, customerNum = 1196, amount = 9.390000

transactionNum = 10, customerNum = 1196, amount = 119.410000

transactionNum = 4, customerNum = 1198, amount = 110.120000

transactionNum = 6, customerNum = 1205, amount = 106.310000

transactionNum = 11, customerNum = 1219, amount = 37.740000

transactionNum = 16, customerNum = 1224, amount = 110.980000

transactionNum = 9, customerNum = 1230, amount = 103.180000

transactionNum = 7, customerNum = 1243, amount = 77.110000

transactionNum = 24, customerNum = 1243, amount = 60.300000

transactionNum = 3, customerNum = 1243, amount = 94.460000

transactionNum = 13, customerNum = 1246, amount = 147.000000

transactionNum = 18, customerNum = 1248, amount = 16.750000

transactionNum = 2, customerNum = 1251, amount = 43.890000

transactionNum = 3, customerNum = 1260, amount = 65.320000

transactionNum = 9, customerNum = 1261, amount = 132.300000

transactionNum = 27, customerNum = 1262, amount = 87.980000

transactionNum = 29, customerNum = 1262, amount = 50.650000

transactionNum = 34, customerNum = 1267, amount = 108.400000

transactionNum = 28, customerNum = 1268, amount = 148.700000

transactionNum = 38, customerNum = 1272, amount = 72.380000

transactionNum = 21, customerNum = 1272, amount = 102.300000

transactionNum = 9, customerNum = 1273, amount = 85.760000

transactionNum = 2, customerNum = 1277, amount = 123.650000

transactionNum = 8, customerNum = 1279, amount = 10.420000

transactionNum = 18, customerNum = 1279, amount = 6.100000

transactionNum = 1, customerNum = 1283, amount = 104.950000

transactionNum = 4, customerNum = 1292, amount = 135.420000

transactionNum = 8, customerNum = 1302, amount = 93.510000

transactionNum = 27, customerNum = 1303, amount = 139.530000

transactionNum = 10, customerNum = 1304, amount = 147.440000

transactionNum = 22, customerNum = 1306, amount = 9.000000

transactionNum = 7, customerNum = 1306, amount = 82.400000

transactionNum = 35, customerNum = 1308, amount = 68.180000

transactionNum = 14, customerNum = 1310, amount = 29.150000

transactionNum = 4, customerNum = 1315, amount = 77.550000

transactionNum = 3, customerNum = 1319, amount = 49.900000

transactionNum = 6, customerNum = 1319, amount = 94.770000

transactionNum = 24, customerNum = 1324, amount = 109.980000

transactionNum = 8, customerNum = 1333, amount = 7.520000

transactionNum = 5, customerNum = 1337, amount = 73.340000

transactionNum = 28, customerNum = 1350, amount = 110.150000

transactionNum = 9, customerNum = 1359, amount = 62.140000

transactionNum = 32, customerNum = 1361, amount = 136.610000

transactionNum = 17, customerNum = 1364, amount = 133.750000

transactionNum = 16, customerNum = 1371, amount = 50.500000

transactionNum = 11, customerNum = 1379, amount = 31.520000

transactionNum = 30, customerNum = 1380, amount = 83.200000

transactionNum = 21, customerNum = 1381, amount = 31.260000

transactionNum = 10, customerNum = 1395, amount = 92.950000

transactionNum = 35, customerNum = 1399, amount = 124.220000

transactionNum = 26, customerNum = 1403, amount = 86.880000

transactionNum = 10, customerNum = 1405, amount = 78.330000

transactionNum = 25, customerNum = 1405, amount = 117.830000

transactionNum = 6, customerNum = 1407, amount = 85.990000

transactionNum = 16, customerNum = 1407, amount = 145.830000

transactionNum = 16, customerNum = 1409, amount = 71.930000

transactionNum = 3, customerNum = 1410, amount = 92.440000

transactionNum = 2, customerNum = 1411, amount = 15.370000

transactionNum = 19, customerNum = 1412, amount = 78.580000

transactionNum = 34, customerNum = 1413, amount = 57.820000

transactionNum = 12, customerNum = 1424, amount = 63.840000

transactionNum = 23, customerNum = 1426, amount = 13.000000

transactionNum = 23, customerNum = 1431, amount = 127.800000

transactionNum = 2, customerNum = 1433, amount = 8.160000

transactionNum = 8, customerNum = 1434, amount = 86.860000

transactionNum = 12, customerNum = 1434, amount = 80.270000

transactionNum = 6, customerNum = 1438, amount = 76.340000

transactionNum = 6, customerNum = 1438, amount = 76.340000

transactionNum = 25, customerNum = 1443, amount = 23.120000

transactionNum = 4, customerNum = 1450, amount = 5.130000

transactionNum = 15, customerNum = 1454, amount = 53.130000

transactionNum = 2, customerNum = 1457, amount = 133.920000

transactionNum = 36, customerNum = 1459, amount = 14.480000

transactionNum = 12, customerNum = 1470, amount = 102.760000

transactionNum = 9, customerNum = 1475, amount = 129.700000

transactionNum = 1, customerNum = 1478, amount = 94.860000

transactionNum = 18, customerNum = 1479, amount = 118.940000

transactionNum = 19, customerNum = 1479, amount = 100.310000

transactionNum = 19, customerNum = 1479, amount = 100.310000

transactionNum = 19, customerNum = 1479, amount = 100.310000

transactionNum = 19, customerNum = 1479, amount = 100.310000

transactionNum = 13, customerNum = 1503, amount = 20.500000

transactionNum = 13, customerNum = 1510, amount = 56.400000

transactionNum = 42, customerNum = 1512, amount = 129.530000

transactionNum = 10, customerNum = 1517, amount = 17.240000

transactionNum = 21, customerNum = 1520, amount = 36.860000

transactionNum = 13, customerNum = 1522, amount = 150.860000

transactionNum = 5, customerNum = 1528, amount = 25.100000

transactionNum = 7, customerNum = 1531, amount = 135.770000

transactionNum = 10, customerNum = 1533, amount = 46.520000

transactionNum = 4, customerNum = 1536, amount = 53.120000

transactionNum = 5, customerNum = 1541, amount = 58.570000

transactionNum = 40, customerNum = 1553, amount = 45.930000

transactionNum = 36, customerNum = 1571, amount = 56.410000

transactionNum = 1, customerNum = 1571, amount = 75.670000

transactionNum = 32, customerNum = 1572, amount = 104.640000

transactionNum = 18, customerNum = 1575, amount = 35.280000

transactionNum = 20, customerNum = 1593, amount = 116.720000

transactionNum = 34, customerNum = 1597, amount = 45.980000

transactionNum = 20, customerNum = 1603, amount = 127.560000

transactionNum = 2, customerNum = 1608, amount = 141.670000

transactionNum = 4, customerNum = 1615, amount = 138.990000

transactionNum = 27, customerNum = 1637, amount = 58.920000

transactionNum = 40, customerNum = 1638, amount = 74.580000

transactionNum = 8, customerNum = 1647, amount = 19.130000

transactionNum = 29, customerNum = 1647, amount = 96.840000

transactionNum = 39, customerNum = 1675, amount = 139.800000

transactionNum = 23, customerNum = 1677, amount = 79.930000

transactionNum = 21, customerNum = 1678, amount = 119.310000

transactionNum = 35, customerNum = 1710, amount = 72.670000

transactionNum = 37, customerNum = 1710, amount = 119.180000

transactionNum = 1, customerNum = 1712, amount = 32.290000

transactionNum = 23, customerNum = 1727, amount = 138.110000

transactionNum = 5, customerNum = 1731, amount = 70.840000

transactionNum = 29, customerNum = 1751, amount = 70.610000

transactionNum = 1, customerNum = 1762, amount = 144.660000

transactionNum = 17, customerNum = 1767, amount = 46.680000

transactionNum = 12, customerNum = 1768, amount = 144.940000

transactionNum = 12, customerNum = 1770, amount = 101.770000

transactionNum = 33, customerNum = 1810, amount = 119.970000

transactionNum = 3, customerNum = 1813, amount = 108.280000

transactionNum = 22, customerNum = 1828, amount = 39.830000

transactionNum = 32, customerNum = 1847, amount = 89.930000

transactionNum = 3, customerNum = 1852, amount = 128.500000

transactionNum = 15, customerNum = 1852, amount = 91.800000

transactionNum = 6, customerNum = 1858, amount = 31.460000

transactionNum = 15, customerNum = 1864, amount = 117.930000

transactionNum = 31, customerNum = 1872, amount = 147.930000

transactionNum = 14, customerNum = 1874, amount = 98.190000

transactionNum = 11, customerNum = 1899, amount = 86.330000

transactionNum = 11, customerNum = 1899, amount = 86.330000

transactionNum = 3, customerNum = 1914, amount = 72.530000

transactionNum = 3, customerNum = 1914, amount = 72.530000

transactionNum = 25, customerNum = 1916, amount = 63.280000

transactionNum = 37, customerNum = 1937, amount = 39.700000

transactionNum = 6, customerNum = 1958, amount = 79.780000

transactionNum = 24, customerNum = 1977, amount = 139.900000

transactionNum = 31, customerNum = 1986, amount = 126.710000

transactionNum = 30, customerNum = 1988, amount = 116.690000

transactionNum = 30, customerNum = 1988, amount = 116.690000

transactionNum = 26, customerNum = 1990, amount = 108.870000

Total Customers = 200