University of Toronto
Department of Computer Science
CSC 228 - Summer 1994

Assignment 1: Managing Mutual Funds

This assignment will give you some practice with separate compilation, file I/O, top-down design, and the master file update algorithm.

You are to write a program, make up test data for all the necessary files, and write a report. The report is intended to convince your tutor that your program is correct. Read pages 1-2 of the *course guide* for more information.

The problem

Sinking interest rates have encouraged large numbers of new investors to enter the mutual funds market. As a CSC 228 student, you have been recruited by a large bank to write a computer program which will manage their files of mutual funds investors and transactions. Although your program will eventually be used to process transactions on-line, the bank manager would like you to develop it so that the transactions are handled by batch processing.

Developing your program

This assignment will proceed in three stages. In the first stage, you will develop a basic version of the program which includes most of the functions required. In the second stage, you will add more features and make the program more useful. Finally, in the third stage, you will modify your program so that it can accommodate the file format currently being used by the bank.

You are required to use separate compilation, so you should give careful consideration as to how you will break up your program into modules.

Part I - the basic version
Due: in tutorial, June 2

Your first task is to write a program which reads entries from a *transactions* file and maintains information about investors and their investments. You may assume that this information is only needed while your program is running, so it does not have to be saved in a file. The transactions you will encounter include:

- **newaccount**: Create a new account for the named investor.
- **purchase**: Purchase a quantity of the named fund.
- **sell**: Sell a quantity of the named fund.
- **transfer**: Transfer a quantity from one fund to another.
- **status**: Display the status of an investor's investments.

All transactions, including the **newaccount** transaction, must indicate the investor account number. In other words, your program is not responsible for assigning account numbers. The transaction file will be in sorted order by this number. All purchases, sales, and transfers will be specified in dollars. In order to handle a transfer of $100.00 from the *international fund* to the *bond fund* for investor account 48723852, your program must calculate the number of units of each fund which correspond to the $100.00 transfer amount.

Any transaction which is in error must be logged in an *error* file. Erroneous transactions include, but are not limited to:

- unknown transaction
- unknown investor
- attempt to sell more than currently held value of fund

For this first part of the assignment, your program will only run for one day, and we will assume that the mutual fund values remain constant during this time. At the end of the day, a report must be generated listing the dollar values of every investor's investments.

What to submit

For this first part of the assignment, you must only submit a sample *transactions* file, as well as the output it produces when processed by your program, including the *error* file. The results you provide should be concise, easy to read, and demonstrate that your program is working correctly. Although your report is not due until the third part of this assignment, you will find life much easier if it is already prepared in rough form, and ready to be submitted with part I.

University of Toronto
Department of Computer Science
CSC 228 - Summer 1994

Assignment 1: Managing Mutual Funds

Part II – extending your program
Due: in tutorial, June 9

Now we improve things. Your program will be run once a week for one month and must maintain the status of all investor information in a master *investor* file between each run. Mutual funds values may be updated many times during this period. To reflect this, each transaction must include the unit values[1] of any fund(s) being purchased, sold, or transferred. In the case of a transfer, the transaction entry must include the unit values of both funds.

A different transaction file will be specified each week your program is run. Every successfully processed transaction must be stored in a file-based linked list, one list per investor. At the end of the month, each investor is sent a report containing a list of his or her transactions, along with an account summary, including the number of units held of each mutual fund, the closing unit values[2] and the total value of all investments.

What to submit

Just like in the first part of this assignment, you must only submit a sample *transactions* file, as well as the output it produces when processed by your program, including the *error* file. The results you provide should be concise, easy to read, and demonstrate that your program is working correctly. Your report should be almost complete by this point, as you will only need to make a few minor changes for part III.

[1] The unit value of a mutual fund is the dollar value of one unit of that fund.
[2] The closing unit value is the dollar value of one unit at the end of the month.

University of Toronto
Department of Computer Science
CSC 228 - Summer 1994

Assignment 1: Managing Mutual Funds

Part III – file format compatability
Due: in tutorial, June 16

Finally, we come to reality. The transactions file no longer includes the unit values of each fund. Instead, these values are supplied in a separate *funds* file, organized as follows:

date	time	fund name	fund value

In order to ensure that all transactions are based on the current values of mutual funds, each entry in the transactions file will include a date and time field. Your program must maintain the correct funds values appropriate to each transaction as it is being processed. To locate the correct funds values, you could perform a sequential search of the *funds* file for every transaction, but there are certainly more efficient methods. For instance, you might consider using a slightly modified binary search.

You will find a sample *transactions* file and *funds* file in the ~/jer/228/as1 directory. You will also see a README file which describes the relevant file formats. You are to modify your program so that it uses these file formats. The names of the input files may be hardcoded in your program, or you may request these from the user.

What to submit

You must submit your executable program using the *submit* command on the CDF system. Do not submit your source files, only a single, final executable. See `man submit` for more information.

Prior to the start of the quiz in the tutorial hour, your printed report is due. Your report should concentrate on the final version of your program, but must indicate what changes you had to make from the first to the second to the third version. You must include your program code as well as sample test files of input and output data, including *error* files, as appendices.

You must also provide the output, including the *error* file, obtained by running your revised program on the files in the ~jer/228/as1 directory.