University of Toronto

CSCB28F—File Structures, Fall 1997

Assignment 1: Sequential Files

Due: Wednesday October 15; electronic submission at 10:00 am, source listings at 11:10 am.
Partners: not allowed for this assignment.
Worth: 15%.

Introduction

The purpose of this assignment is to teach you cosequential processing. It will help everyone get comfortable with file access in C. To learn about this general technique, you should read your lecture notes on the topic and Sections 7.1 and 7.2 of the textbook.

You are to write a program that maintains a set of flights where the users will make requests of the following types:

- add a new flight;
- delete a flight;
- change information of a flight; and
- list all information of a certain flight.

The information is maintained in a master file ordered by the field *Identifier* that uniquely determines each flight. There are several transaction files that introduce modifications to the master file. Your task is to process all transactions contained in the different transaction files updating the master file. You can read the master file *only once*. Consequently, you must process all transaction files *at the same time*. In addition, changes in a record of the master file must be recorded only once, after processing all transcations that deal with a certain flight.
We will assume that there are at most 10 transaction files. We will also assume that the records on each transaction file are ordered by the field *Identifier*. Of course, this is not a realistic assumption but is sufficient for the purpose of this assignment.

File Descriptions

The files contain one record per flight. The records are of fixed length, with fixed-length fields, and they are sorted according to *Identifier*.

The records in transaction files contain the same information as in the master file with the exception that there is one more field *Type*. The action you should take depends on the contents of this field:

- Type=1: insert a new record into the master file;
- Type=2: delete an existing record from the master file;
- Type=3: change the information of an existing record in the master file; and
- Type=4: list the information of the flight *Identifier* in the master file.

Field#	Field Name	Type	Length in bytes	Notes	File
1	Identifier	numeric	long	unique key	master/trans
2	FlightNumber	numeric	int		master/trans
3	DateFlight	alphanumeric	char[8]	format = yy/mm/dd	master/trans
4	NumberCities	numeric	int		master/trans
5	ListCities	array of string	char[5][15]		master/trans
6	NumberSeats	numeric	int		master/trans
7	NumberCrew	numeric	int		master/trans
8	CrewNames	array of string	char[5][10]		master/trans
9	Type	alphanumeric	char		trans

Your program

First, you must create the master file and the transaction files. Call these files: **master**, **t1**, **t2**, ..., **t10**.

Second, your program should process the transaction files sequentially and in batch mode (no further input from the user while it executes). Call your executable program **flight**. We will use the following command to run your program:

```
flight master t1 t2...
```

Do not read the entire transaction files or the master file all at once into main memory. In real situations, these files may be too large to fit in main memory.

There may be errors in the transaction files and your program should do some error checking. Do not apply the transactions with errors to the master file. Instead, just output the transaction and a short message to an error file. There are two kind of errors: (a) errors that can be detected by looking at the transaction record (bad syntax), and (b) errors that depend on the present content of the master file (for instance, a change in a nonexisting flight).

How to hand in your assignment

First, you must submit your files. This involves four programs: `flight.c` (the program that process the transactions and updates the master file), `tests.c` (the program that tests your work), `master.c` (the program that creates the master file), `trans.c` (the program that creates the transaction files), an the error output `error.log`.

The marker must be able to compile your program using **gcc**, and run it successfully on an appropriate test file.

Use the submit command to submit copies of your source codes:

```
submit -N a1 cscb28f flight.c tests.c master.c trans.c error.log
```

At the beginning of the lecture of October 15, you must hand in source listing for your C programs (Unix). You do not need to include a report but marks will be given for good style and modularity. The program should be properly documented with features such as comments for each variable and function, variables with sensible names, and so on. In addition, your `tests.c` program should contain comments indicating what each test is checking. Also include a description of the transaction error checking done by your program.