CS22510 Assignment One, 2012-2013
Runners and Riders – “Out and about”

Neal Snooke, Dave Price and Fred Labrosse

22 Feb 2013

1 Introduction

This is the first of two CS22510 assignments. It will count for 40% of the total marks for the module. You should spend roughly 40 hours of effort on this assignment. **Deadline: This assignment should be posted in the Departmental Coursework Postbox between 10:00am. and 4:00pm. on Friday, 22nd March 2013.**

Notes:

1. If there is good reason why you cannot hand in the assignment via the Departmental Coursework Postbox, you must make alternative arrangements with nns (or, exceptionally, with another member of staff) to hand in the assignment before the deadline.
2. If you hand in work at a time other than that specified you must ensure that the work is dated and a receipt obtained.
3. Work which is handed in after the deadline, without prior permission (which is rarely given), will usually receive no marks. If you do hand in your work after the deadline, please also submit a Late Assignment form explaining why the work was handed in late.
4. If you feel that you have a very good reason for handing in your work late, you must seek the permission of the year co-ordinator (Nigel Hardy <nwh@aber.ac.uk>) before the deadline for handing in the work. In almost all cases you will be told to hand in the work, in an incomplete state, by the deadline, and complete a Special Circumstances Form explaining the circumstances that made it impossible for you to complete the work, which must be accompanied by supporting evidence.
5. The assignment will not be returned to you. Therefore, if you want to keep a copy of it, you should make your own copy before submitting your assignment.
6. Because this is a coding assignment it cannot be submitted anonymously. Every piece of code submitted should show, in a header comment, the author of the piece of code.
7. Note: this is an “individual” assignment and must be completed as a one person effort by the student submitting the work. Your attention is drawn to section 6 of the Department’s Student Handbook, available at

   `http://www.aber.ac.uk/~dcswww/Dept/Teaching/Handbook/`

   If in doubt, make sure that you properly acknowledge material (including code that is reused or closely derived from others) in order to avoid any suspicion that you are trying to cheat and ask for advice if you are not sure. You should include a signed copy of the department’s standard “statement of originality” with your submission.

2 The Task

You are already familiar with the "Runners and Riders" assignment completed for the CS23710 module last semester. This assignment will extend that work to develop a more sophisticated system for the management of open countryside events. The same set of files and data are used for this assignment and you will need to refer to the CS23710 assignment and blackboard site for those details and data files.

In particular the improved system will be comprised of *three separate programs*. One of the programs must be written in the C programming language, one in C++, and one in Java. You may decide which language to use for which program. The following overview outlines the main functions of each program:

Event Creation This text based program will allow the data files that are required to define an event to be created and viewed.

Checkpoint Manager This program will allow the staff at each checkpoint to note the arrival of a competitor at each checkpoint. The intention is that this program will have a very simple graphical user interface to allow the checkpoint staff to easily monitor the arrival of competitors. Please note that it is easy to get carried away producing GUI's; the objective here is the simplest possible GUI that can achieve the requirements. Do not waste lots of time making it 'overly pretty'. Use the basic 'widgets' available in your selected toolkit/api.

Event Manager This text based program will allow the event organiser to query the system to obtain current information about competitors and to produce a summary list of results for all competitors. The functionality required is similar to the system produced for CS23710, however the new version also requires the creation of a 'log' file that lists each of the requests made by a race official.

The system will involve a number of files summarised as follows:

- **event file** – contains the event title, date and start time
- **nodes file** – each node represents a checkpoint or join point between tracks
- **tracks file** – each track is defined by: an identifying number; the start and end nodes; and the expected time a competitor will need to traverse the track.
- **courses file** – each course is defined by: a single letter identifier; the number of nodes; and a list of the node numbers that comprise a course.
- **entrants file** – each entrant is defined by: a competitor number; the course they are registered for; and their name
- **times file** – an entry is included for each competitor arriving at a checkpoint. Each entry contains information about the checkpoint number; the competitor number; time of arrival. Medical checkpoints also include time of departure. A code is used to indicate the type of checkpoint and whether the competitor was expected at that checkpoint.
- **log file** – this file is used to maintain a log of operations performed by the Checkpoint Manager and Event Manager.

With the exception of the log file, the exact format of the data for the files is contained in the CS23710 (2012-2013) assignment description and you *must* use exactly that format. The CS23710 Assignment description and an example of each of the files is available (with the exception of the log file) on the Blackboard 'Assignments' area for CS22510.

You may assume that all of the machines running the programs have access to a central file store where the files are kept. For this assignment it is enough to run the programs on the same computer.

The following sections provide the detailed requirements for each of the programs.

2.1 The Event Creation Program

The event creation program must use a textual command line ('terminal window') interface that allows creation of the **events file**, **entrants file**, and **courses file** associated with each event. The user will interact with the program by selecting from numbered menu items, and typing in information in response to prompts. The **nodes file** and **tracks file** should *not* be created, you will be given examples of these to use (they rarely change). The functions required are to create a new event; add entrants (competitors) to an event; create the courses for an event, allowing only the nodes from the node file.

2.2 The Checkpoint Manager Program

This program should have a *simple* GUI to make the program easier for the checkpoint personnel to use. The names of the files required should be supplied as command line parameters when the program is executed, to simplify the GUI. If no parameters are supplied, the program should print a message on the command line explaining the required command line parameters. The checkpoint manager program will allow the user to select the type of checkpoint (Time or Medical). As each competitor arrives the name should be selected from the list of registered entrants. It must be possible to enter the arrival time for either type of checkpoint, and the departure time for Medical checkpoints only. For Time checkpoints it may help the user to set the arrival time to the current time as a default (optional). For Medical checkpoints it may make sense to use the current time as the departure time (assume there is a helper that notes the arrival times, so they can be typed in). For Medical checkpoints there will be a way of indicating that the competitor was excluded.

The user should press a 'submit' button to add an entry to the **times file**. This will include the indicator to signify that the competitor was/was not expected at the checkpoint, but the system must determine this (not the user). Notice that file locking is required for the **times file**, since several instances of the checkpoint manager program will normally be running concurrently.

2.3 The Event Manager Program

This program provides several functions to access information about competitors' progress, and also compiles the results list for an event. The event manager program has essentially the same functionality as the 'extended' program produced for the CS23710 assignment, with the *addition* of the following functionality:

- Every time the Event Manager carries out an activity, a line is appended to the **log file** recording the date and time of the activity, and what the activity was.

The function to supply the times competitors reached checkpoints is not required in the Event Manager Program, since the Checkpoint Manager program can be used to do this. The Event Manager program must use a textual command line interface.

The format for the log file is not predefined and you should create a suitable format. An example solution to the Event Manager program (as produced for the CS23710 Assignment) is available which you may use as a starting point, although it is likely to be preferable to use your own C module assignment as the basis. The detailed descriptions of each function is contained in the CS23710 (2012-2013) assignment description.

3 Hints

This section contains some hints on how you might design and implement your system. You do not have to follow any of these hints, as long as your system satisfies the requirements set out above.

3.1 File Locking

File locking is required to avoid problems that could be caused by contention over simultaneous access to the same file. In particular, there is the potential for file corruption due to concurrent changes being made by multiple instances of the Checkpoint Manager program accessing the 'Times file'. Depending on the design of your system, you may have to lock other files. File locking is platform dependent. Microsoft Windows supports mandatory file locking and Unix-based platforms generally support cooperative file locking, however this is sufficient for the application. Java supports file locking in its java.nio.channels.FileLock Class which is mapped to (and therefore is dependent on) the underlying operating system file locking capabilities. Hence cooperative locking is provided by Java on most *nix based platforms. When using C or C++, most Unix, Linux and MacOSX based systems support Posix-based cooperative file locking using the fcntl() function specified in fcntl.h. The Microsoft Windows operating system supports mandatory locking using the WinAPI LockFile functions on file handles. This requires use of the Microsoft specific WinAPI libraries and therefore such code will only work on windows platforms. Alternatively the Posix-based fcntl() function could be used if the Cygwin DLL is installed on an MS Windows machine (this also provides the Gnu g++ series of compilers which may be used with NetBeans for example). Example code illustrating simple use of the Java and Posix-based file locking mechanisms are provided on Blackboard together with the URL for Cygwin and the Microsoft WinAPI documentation.

3.2 System Construction

You need to create the system in such a way that the various parts can be tested progressively. Remember that although some of your programs create or append to the data files used by other programs, you have examples of the data files from the CS23710 assignment. This allows some degree of flexibility for testing the individual programs without the need to manually create test data files. In addition you already have much of the code for the event manager program written in C. You may decide to update this code, or port to another language, dependent upon your choice for writing the other programs. Concurrent access to the 'times' file and 'log' file may cause problems and should be avoided by using file locking mechanisms, this is however a separate issue to the main functionality and you could test the main system functionality (by manually avoiding any concurrent access) before adding and testing the file locking.

3.3 Previous knowledge

This module builds on several prerequisite modules. You have learned the C programming language in CS23710 last semester alongside data structures and algorithms in CS21120. Prior to that Java and AWT/Swing was covered as part of CS12420, which in turn built upon the earlier java introduction courses. You may well need to refer to your notes and assignments for those modules to help complete this assignment.

4 Assessment Criteria

This assignment will be assessed using the Assessment Criteria for Development. See the Student Handbook, Appendix AA, via the URL:

`http://www.aber.ac.uk/compsci/Dept/Teaching/Handbook/AppendixAA.pdf`

The usual requirements for coding projects apply, namely that programs should be well commented with comments that add real value and do not just, in essence, duplicate code. Programs should have good layout and must use meaningful names for variables, classes and other identifiers.

5 The Material You Must Submit

You must hand in a document and a machine readable version of your system. The document must contain twelve sections/subsections (as listed below) but it need not be a single document in the sense of a single Word or PDF document; it can be a set of twelve sections/subsections that are generated independently. However, the twelve sections/subsections **must** be clearly identified.

The twelve sections/subsections of the document must be as follows:

1. (a) A printed listing of the code for the Event Creation program.

   (b) A printout showing the results of attempting to compile your Event Creation program with any errors or warnings that are produced. If you use an IDE such as NetBeans to develop your code, screen images of the compiler output is acceptable.
   (c) A printout of the terminal session generated by your Event Creation program when running your system. Screen images are acceptable. You should run the system and show an event being created from scratch for a small number of competitors and a short course.
   (d) A printout of the files that were created by this execution of the Event Creation program. Again, screen images are acceptable.

2. (a) A printed listing of the code for the Checkpoint Manager program.
   (b) A printout showing the results of attempting to compile your Checkpoint Manager program with any errors or warnings that are produced. If you use an IDE such as NetBeans to develop your code, screen images of the compiler output is acceptable.
   (c) Screenshots showing your Checkpoint Manager program being used to register a competitor arrival at a checkpoint. Show at least one example of use for a Time checkpoint and at least one example of a Medical checkpoint. To demonstrate that file locking works you may need to create a special 'test' scenario/or version of the code.

3. (a) A printout showing the results of attempting to compile your Event Manager program with any errors or warnings that are produced. If you use an IDE such as NetBeans to develop your code, screen images of the compiler output is acceptable.
   (b) A printout of the output generated by your Event Manager program when running your system. You should show at least one attempt for each of the types of query available from the Event Manager. Again, screen images are acceptable.
   (c) A printout of the results list produced at the end of a competition. You will need to use your Checkpoint Manager to ensure that a number of competitors have completed the race.
   (d) A printout of the log file produced by the system after running your simulation.
   Note you do not need to provide a printed listing of the code for the Event Manager program (since it will be based on previously submitted work, or the example provided), however it *must* be included in the machine readable version of your submission.

4. A short description of your three programs. This section must be brief and should be at most two sides of A4 in length. Include diagrams if you wish to help explain your programs.

As well as the document described above, you must submit a machine readable version of your *entire system* (including at least all source code and data files) on either a CD, DVD, or memory stick. The digital media that you submit will not be returned to you. Please keep a copy of your your entire submission, in case of failed media.

Notes:

1. You must put all the material you hand in in a single folder and it must be possible to read and write on all sheets that you hand in without having to remove them from the folder. In particular, you must not hand in loose sheets in a bag, or put everything in a single plastic pocket of a folder. The folder does not need to be a commercially made folder, indeed, we are happy to accept sheets neatly stapled together as a leaflet.
2. The CD, DVD, or pen drive must be attached to your written material in a sensible manner so that on the one hand it will not get lost, but on the other hand, it is easy for me to remove and use. Also, the CD, DVD, or pen drive must be labelled with your name, in case it becomes separate from the main body of work.
3. It is your document that will be marked. The machine readable version of your system will be used if it is necessary to check anything claimed in the documentation. Your printed code listings must be easy to read. One thing that makes code difficult to read is if the lines are so long that they wrap. It is acceptable for you to submit your code listings in landscape format to avoid lines wrapping.

Neal Snooke,
22 Feb 2013.