Project - 3
CSE 421/521 – Operating Systems
Due: December 10th @11:59pm, 2012

1. Preparation

Before beginning your work, please read the following carefully:

- Chapter 18 from Silberschatz
- Lecture notes 22 on Distributed Coordination

2. Programming Task: Implement a Distributed Event Coordination System

The objective of this project is to implement a distributed event coordination system (DEC) system in C/C++ on a UNIX-based platform. This system will consist of a DEC server which can interact and communicate with multiple DEC clients at the same time.

2.A DEC Server:

SYNOPSIS: dec_server [**−h**] [**-p** port-number] [**−l** *file*]

−h **:** Print a usage summary with all options and exit.

-p ***port-number*** **:** Listen on the given port. If not provided, **dec_server** will listen on port 9090.

−l ***file*** **:** Log all requests and responses to the given file. If not, print all to *stdout.*

DESCRIPTION: dec_server is the server component of the Distributed Event Coordination (DEC) System. It will listen on a certain port number for the incoming requests from DEC clients. There can be three different requests coming from clients:

i) Insert request **:** This request from any client will enter new **event ordering** information to the server using the **happened–before** relationship. In this relationship, events will be represented with capital letters (i.e., A, B, C, D… etc), and the happened before relationship will be represented with **"->" (*dash* *and* *right angle bracket*).** The event orderings will be separated with "white space" and will end with a semicolon.

Example for an insert request:

```
dec_client1$  insert A->B  B->C  C->D;
dec_client1$  response from server: INSERT DONE.
dec_client1$  █
```

This event insertion request is sent to the server from dec_client1, and it includes four events (A, B, C, D). Event A happened before event B; event B happened before event C; and event C happened before event D.

After receiving the insert request, the DEC server will insert this event ordering information to its **global event ordering graph.** If the insertion is successful, it will return to the client an **"INSERT DONE"** message.

```
dec_server$  request received from timberlake.cse.buffalo.edu:
             insert A->B  B->C  C->D;
dec_server$  INSERT DONE.
```

If the server detects any conflicting information during insert (i.e. it had B->A before and now it receives A->B, which is not possible), then the server will return a **"CONFLICT DETECTED. INSERT FAILED"** message. It will also show the conflicting events, i.e. **"A->B and B->A cannot be true at the same time!"**.

ii) Query request **:** This request from any client will ask the relative ordering between any two events. The queried events in this request will be separated with "white space" and will end with a semicolon.

Example for a query request:

```
dec_client1$  query A D;
dec_client1$  response from server: A happened before D.
dec_client1$  █
```

There can be multiple requests sent on the same line:

```
dec_client1$  insert E->F; query A E;
dec_client1$  response from server: A concurrent to E.
dec_client1$  █
```

If there is no information available about any particular event at the server, it should return an error message:

```
dec_client1$  query A G;
dec_client1$  response from server: Event not found: G.
dec_client1$  █
```

iii) Reset request **:** This request from any client will reset the **global event ordering graph** at the server. The reset request will not take any arguments and will be followed by a semicolon.

Example for a reset request:

```
dec_client1$  reset;
dec_client1$  response from server: RESET DONE.
dec_client1$  █
```

2.A DEC Client:

SYNOPSIS: dec_client [**−h**] [-s server-host] [-p port-number]

−h	**:** Print a usage summary with all options and exit.
-s ***server-host***	**:** Connect to the specified host (by hostname or by IP address). If not provided, connect to the *localhost.*
-p ***port-number***	**:** Connect to the server at the given port. If not provided, connect to 9090.

DESCRIPTION: dec_client is the client component of the Distributed Event Coordination (DEC) System. It will connect to the DEC server running on particular host and port number and will send the server one of the three requests mentioned above.

2.C Other Requirements:

i) There can be multiple DEC clients connected to the server at the same time.

ii) The insert requests from multiple clients will be inserted to the same **global event ordering graph** at the server until one of the clients send a "reset" request. At that point the graph will be reset, and the future insert requests will be inserted in a new graph.

iii) The client may send any number of requests in a single line separated by semicolons. All of these requests will be executed and responded by the server one by one in the same order.

```
dec_client1$  insert A->B B->C; insert C->D; query A D; reset;
dec_client1$  response from server: INSERT DONE.
dec_client1$  response from server: INSERT DONE.
dec_client1$  response from server: A happened before D.
dec_client1$  response from server: RESET DONE.
dec_client1$  █
```

For simplicity, in this project you can assume atomicity across multiple requests in a single line. Which means that while the server is executing multiple requests from a client (if sent on a single line – before ENTER/NEWLINE), requests from another client cannot interrupt/interleave this execution. But, if the requests are in different lines (meaning multiple ENTERs or NEWLINEs), requests from another client can still interrupt/interleave these requests.

iv) You cannot have cycles in the **global event ordering graph.** Be careful about that.

3. What to Submit?

- You need to prepare a tar package containing all source files of the project, and call it <yourlastname>.tar. This package should also include a Makefile and README file. The whole package should compile when the tester simply types make in the source code directory. Your README file should contain details and options on how to compile and run the server, if there are any.

  This package should be **emailed** to the TAs Ying Yang (yyang25@buffalo.edu) and Weida Zhong (weidazho@buffalo.edu) by **December 10th @11:59pm.**

- You also need to write a report explaining the design of your Distributed Event Coordination (DEC) server and client. The report should not exceed 3 pages. You should append your README file (Appendix 1) and your source code (Appendix 2) to this report.

  A hardcopy of this report should be submitted to Prof. Kosar's mailbox (338 Davis) on **December 11th, @4:00pm**