1. Problem: A Distributed Insurance Information System

Consider an insurance company that offers three kinds of policies: life, auto and home. A policy number identifies each insurance policy, of whatever kind. For all three types of insurance, it is necessary to have the policyholder's name (a character string of maximum length 20), address (a character string of maximum length 40), starting date, expiration date, amount of the insurance (a real number), and the monthly premium payment (a real number).

For auto and home insurance policies, a deductible amount is needed. For a life insurance policy, the insured's birth date (month, day and year) and beneficiary (a person's name, of the same type as that of the policyholder's name) are needed. For an auto insurance policy, a license number (a character string of length 7), state (a character string of length 2), car model (a character string of maximum length 8), and year are required. For a homeowner's policy, an indication of the age of the house and the presence of any security devices are required.

You are asked to design and implement an information system for this insurance company to store the customers' insurance policies information. You will write both client and server programs of this system in C or C++ using UNIX socket and Internet TCP protocol for inter-process communication between the client and server.

The service interface (i.e. operations) that the server supports is as follows:

- buyPolicy([in] int policyType, [out] success)

A client invokes this operation to buy an insurance policy; the first argument specifies the type of the policy the client desires. This operation returns a go-ahead or denial value in "success". A value 0 means "go-ahead". A client should then supply his/her personal information including name, address, starting date and expiration date as well as insurance type specific information (e.g., birth date if life insurance policy
is wanted.) A value of 1 means the request is denied.


  A client invokes this operation to supply his/her personal information to the server. This operation returns two things: a structure containing the approved policy information including a policy number and a code of success (of value 1) or failure (of value 0) indicating whether the operation has been successfully executed or failed (i.e. request is rejected.)

  - For a life policy, the client submits a) insured's name, b) insured's address and c) nominal starting date, d) nominal expiration dates and e) beneficiary's name.
  - For an auto policy, the client submits a) insured's name, b) insured's address and c) nominal starting date, d) nominal expiration dates and e) a license number f) state (a character string of length, g) car model and h) year of purchase.
  - For a homeowner's policy, the client submits a) insured's name, b) insured's address and c) nominal starting date, d) nominal expiration dates and e) age of the house and f) an indicator of the presence of any security devices.

  The server returns a) policy number, b) name, c) address, d) policy starting date, e) policy expiration date, f) amount of the insurance, and g) monthly premium payment. For auto and home insurance policies, a deductible amount is also returned from the server.


  A client gives the policy number and a structure that describes the new customer information. If the return value of the "success" argument is 1, this means the modification has been successfully done. The "policyInfoStruct" argument contains the full description of the updated policy information. Otherwise, with value 0, the contents of the "policyInfoStruct" argument should be ignored.


  A client gives the policy number to obtain the corresponding policy information.
1.1. Requirements

The client program may send out any number of requests while waiting for replies from the server. In other words, both your client and server should be capable of handling multiple outstanding requests and replies and their sequencing.

- You should code your client and server program in such a way that either one can run on any UNIX machine without code changes. Namely, do not hardcode machine domain name or IP address in your program; the information should be obtained from the system at the run time.
- You are free to choose a port at that the server will be listening. The client program gets the host name and the listening port of the server from the command line arguments. The format of the command is `<client-program-name, server-host-name, server-port-number>`.
- The requests are initiated by end users at the client side. Namely, your client program should prompt users to enter requests with proper description about how to use and what information is needed.

The goal of this assignment is to allow students to familiarize basic Internet socket programming and hands-on experience. Students are encouraged to design and implement a user-friendly graphic user interface.

2. Submission

You should submit a printout of the program source codes with proper documentation and comments (Note: documentation is mandatory.), a floppy diskette of both the source and executable files sealed in an A4 size yellow envelope at the class of November 31, 2001. Note that you will also need to make an appointment with the Teaching Assistant for the demo of your program.

WARNING: Students who copy other's program get zero point from this assignment and their names will be made known to the department.