COMP 2401 -- Assignment #5

Due: Friday, December 6, 2013 at 12:00 (noon)

Goal

You will write a program in C, in the Ubuntu Linux environment, to simulate a two-player game of Twenty Questions, as a distributed system over the School of Computer Science network. One player (the *oracle*) chooses an object that the other player (the *guesser*) tries to guess by asking the oracle one question at a time. At every turn, the guesser asks a yes/no question, or tries to guess the object. The oracle responds by signalling one of three responses: yes, no, or win. If the guesser correctly guesses the object within 20 questions, s/he wins, otherwise the oracle wins. The winning player is given a choice of quitting the game, or playing another round. If s/he opts to play again, s/he becomes the oracle for the next round.

In this assignment, you will:

- implement a game between two users on different computers, communicating over TCP/IP sockets
- modify client-server code so that a single executable program can switch between client mode and server mode

Instructions

1. Process startup

You are writing code for a single executable program. Both guesser and oracle players will be running their own copy of the **same** program, but executing on different computers that are networked together.

Your program initiates a connection to another game process as follows:

- the user may specify an IP address on the command line or not
- if an IP address is specified, then the game process initiates a connection using TCP/IP sockets to the other game process running at the specified IP address
- if no IP address is specified, then the game process waits for another game process to initiate a connection

2. Game logic

The Twenty Questions game logic works as follows:

- setting up a round
  - the player on the game process that initiated the connection takes on the role of guesser
  - the other player becomes the oracle
- during the round
  - at each turn, the guesser is prompted for a yes/no question (questions should be numbered)
  - the question is sent to the oracle's game process
  - the oracle user enters a response using signals; do **not** prompt the oracle for input!
    - the SIGINT (Ctrl-C) signal indicates a *yes* answer to the question
    - the SIGTSTP (Ctrl-Z) signal indicates a *no* answer to the question
    - the SIGQUIT (Ctrl-\) signal is a *win* answer, meaning that the guesser has correctly guessed the object
  - both players see all the questions asked so far, along with the answer
  - if the guesser correctly guesses the object, s/he wins
  - if the number of questions reaches 20 without a correct guess, the oracle wins
- ending a round
  - when a player wins, s/he is prompted to either play another round or quit
  - the losing player does not get a choice and must play again if the winner chooses to do so
  - if the winner decides to play again, s/he becomes the oracle, and a new round starts up
  - if the winner decides to quit:
    - the winner's game process closes the connection, cleans up all its resources, and terminates
    - the loser's game process waits for a new connection to be initiated

Constraints

- **Design**:
  - you must separate your code into modular, reusable functions
  - **never** use global variables, unless otherwise instructed
- **Implementation**:
  - your program must perform all basic error checking and clean up resources upon termination
  - it must be thoroughly commented
- **Execution**
  - programs that do not compile, do not execute, or violate any constraint are subject to severe deductions

Submission

You will submit in *cuLearn*, before the due date and time, **one** tar file that includes all the following:

- all source and header files
- a Makefile
- a readme file, which must include:
  - a preamble (program author(s), purpose, list of source/header/data files)
  - exact compilation command(s)
  - launching and operating instructions

Grading

- **Marking breakdown:**

Component	Marks
process startup	40
game logic	60

- **Assignment grade:**
  - Your grade will be computed based on the completeness of your implementation, plus bonus marks, minus deductions.
- **Deductions:**
  - **100 marks** if:
    - any files are missing from your submission, or if they are corrupt or in the wrong format
  - **50 marks** if:
    - the Makefile is missing
    - the code does not compile using gcc in the Ubuntu Linux shell environment
    - code cannot be tested because it doesn't run
  - **25 marks** if:
    - your submission consists of anything other than exactly one tar file
    - your program is not broken down into multiple reusable, modular functions
    - your code is not correctly separated into header and source files
    - your program uses global variables (unless otherwise explicitly permitted)
    - the readme file is missing or incomplete
  - **10 marks** for missing comments or other bad style (non-standard indentation, improper identifier names, etc)
- **Bonus marks:**
  - Up to 5 extra marks are available for fun and creative additional features