COMP 2401 -- Assignment #5

Due: Monday, December 8, 2014 at 9:00 PM
Collaboration: You may work in groups of 2 students

Goal

You will modify a given program so that a simulated shell secretly captures all the commands that the user enters and sends them to a logger process located on the network, possibly on another computer. You will also implement the networking and logging portions of the logger process. You will begin with the skeleton code found here.

Learning Objectives

- practice understanding and modifying existing code
- program with TCP/IP sockets for network communications
- implement signal handlers for resource cleanup

Instructions

1. Shell process

Read and understand the **`cuShell`** program found in the skeleton code. Make sure that you know exactly what it does and how it does it. Don't forget to check the readme file!

Modify **`cuShell`** as follows:

- add the networking code to the **`initShell`** function; it must:
  - create a TCP/IP socket to communicate with the **`logger`** process
  - connect to the **`logger`** process; the IP address and port number from the config file must be used
  - return the socket as an output parameter
- every time the user enters a command on the command line, it must be sent to the **`logger`** process on that socket

Notes:

- the **`cuShell`** process behaves as a TCP/IP client in this program
- do **not** make any changes to the existing code provided

2. Logger process

Read and understand the **`logger`** program found in the posted code.

Modify **`logger`** so that it does the following:

- install a signal handler for SIGINT so that a cleanup function is called
- create a TCP/IP socket to receive a connection request from a **`cuShell`** process
- bind the socket to the port specified in the config file, and listen on that socket
- accept an incoming connection request
- store every command received from the **`cuShell`** process into the log file, along with a timestamp
  - each timestamp must use the following format: YYYY-MM-DD-HH:MM:SS
  - your code must accept commands until "exit" is received
- implement a cleanup function that closes sockets and files

Notes:

- the **`logger`** process behaves as a TCP/IP server in this program
- do **not** make any changes to the existing code provided

Constraints

- you may use **one** global variable for the log file; you may not use any other global variables
- your program must be thoroughly commented

Submission

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

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

Grading

- **Marking breakdown:** The grading scheme is posted in cuLearn.

- **Deductions:**
  - Up to 50 marks for any of the following:
    - the code does not compile using gcc in the VM provided for the course
    - unauthorized changes have been made to the code provided for you, where applicable
    - code cannot be tested because it doesn't run
  - Up to 20 marks for any of the following:
    - the Makefile or the readme file is missing or incomplete
  - Up to 10 marks for missing comments or other bad style (indentation, identifier names, etc.)

- **Bonus marks:**
  - Up to 5 extra marks are available for fun and creative additional features