Design Document

Data Communications: Linux Chat Server

Jonathan Chu, A00881533, 4O
Eric Tsang, A00841554, 4O

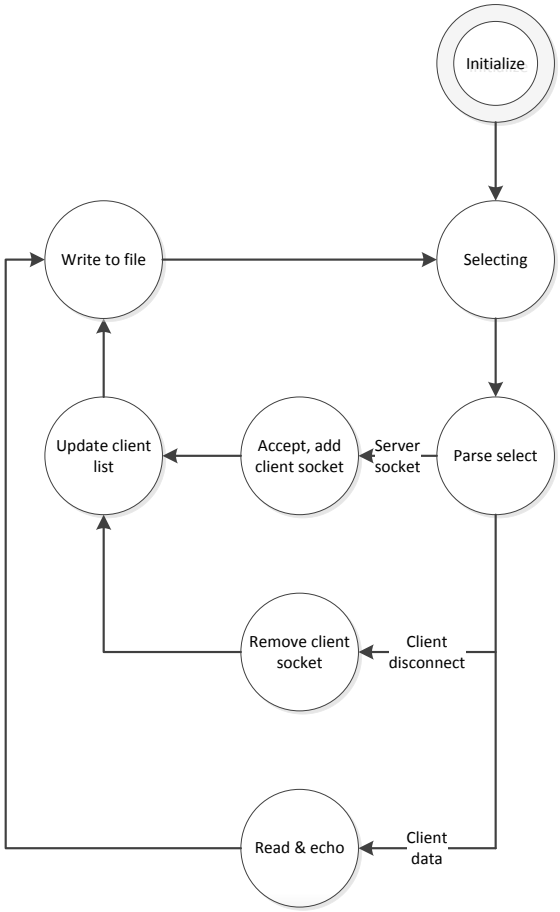
Table of Contents

January 10, 2015

State Transition Diagrams

The state transition diagrams in this section describe the states of the server and client applications.

Server

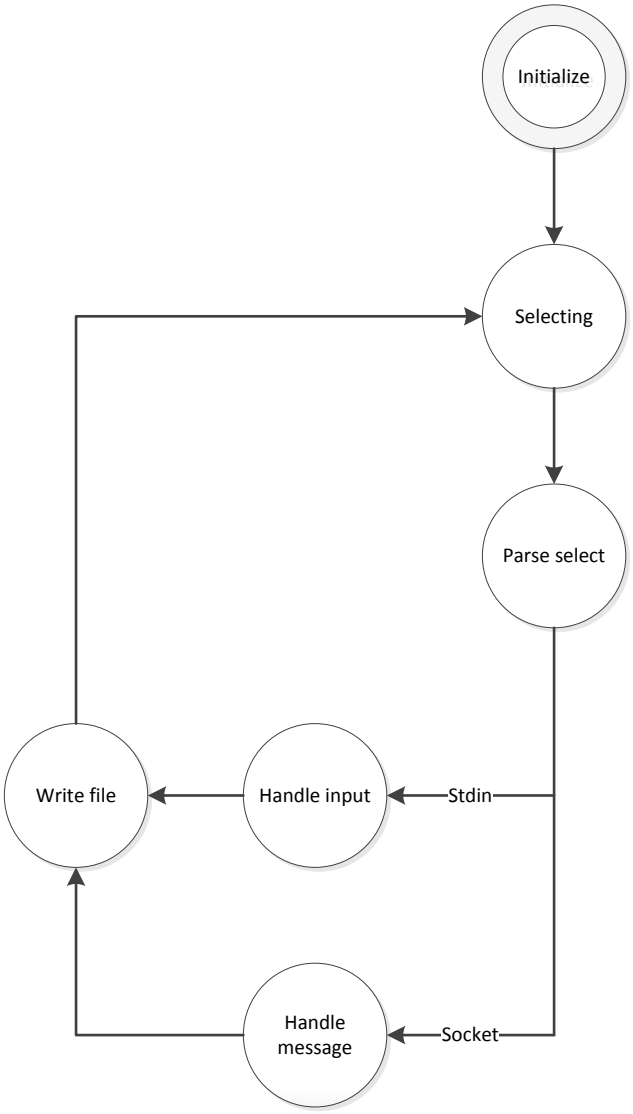


The diagram above illustrates the states of the server application:

- **Initialize**; creates the server socket, and sets up a set of sockets for select to monitor.
- **Selecting**; call select to monitor our set of sockets.
- **Parse select**; determine which socket caused select to return.
- **Accept, add client socket**; accept a connection from the server socket, and add them to our socket set so select can monitor it as well.
- **Remove client socket**; remove the client from our socket list, so select will no longer monitor it.
- **Read & echo**; read data from the client socket, then write the data read from the client socket to all other sockets except the one that sent it.
- **Update client list**; update our list of clients that are being displayed on std out from the select's socket set.
- **Write to file**; write out output to a file for bonus marks.

The server application can be signaled at anytime, causing it to perform a cleanup routine, and terminate.

Client



The above diagram illustrates the states of the client application:

- **Initialize**; opens a socket that is connected to the server.
- **Selecting**; select monitors stdin, and the socket.
- **Parse select**; select has returned, determine which file descriptor made select return.
- **Handle input**; reads data from stdin, then transmits the data read from stdin to the server through the socket.
- **handle message**; reads data from the socket, then displays the received data on the terminal.
- **Write file**; record inputs and outputs to the file.

The client application can be signaled at anytime, causing it to perform a cleanup routine, and terminate.

Pseudocode

The pseudo code is a programming-language-agnostic code-like description of what happens in each of the states in the server and client applications.

Server

This section contains pseudo code for the server application.

Initialize

Sets up the server application, and acquires resources.

```
1  parse command line input to get port number, and file name
2  create server socket on the specified port
3  open file for appending, or create it if it doesn't exist
```

Cleanup

Releases system resources, and terminates the application.

```
1  close all sockets in the socket set
2  close the file
```

Selecting

Waiting for an event to occur.

```
1  issue a select call on the socket set
```

Parse select

Determine which event has occurred.

```
1  loop through socket set, and determine which socket needs
   attention
```

Accept, add client socket

Handles a connection request from a client to the server.

```
1  accept a new connection from the server socket
2  add the socket to the socket set
3  add client meta data to connected clients list
```

Remove client socket

Handles a disconnection of a client from the server.

```
1  remove the socket from the socket set
2  remove the corresponding client information from connected
   clients list
```

Read & echo

Handles incoming data from a socket.

```
1  read from, data from the client socket
2  transmit the read data to all sockets except the one that the
   data was read from
```

Update client list

Updates the display to redisplay the connected clients.

```
1  redisplay the list of connected clients
```

Write to file

Records history to a file.

```
1  record the chat, disconnect, and connect history to the file
```

Client

This section contains pseudo code for the client application.

Initialize

Sets up the client application, and acquires the needed resources.

```
1  parse command line input to get file name, remote address and
   port, and display name
2  connect to the server
3  send server our display name or nothing if none
```

Cleanup

Releases system resources.

```
1  disconnect from the server, and release any resources
```

Selecting

Wait for an event to occur instead of looping, and pinning a core.

```
1  call select with a file descriptor set composed of stdin, and
   the socket
```

Parse select

Select has returned, determine which event it was.

```
1  select has returned, determine which file descriptor caused
   select to return
```

Handle input

Handle an input event from the standard input stream.

```
1  read data from stdin, and then send the data from std in to
   the server
```

Handle message

Handle a message from the socket.

```
1  read data from the socket, and display it
```

Write file

Record chat history to the user specified file.

```
1  record the chat history to the user specified file
```