Bhuvana Kanakam
SE21UCSE035
CS4101 Lab 04
09.20.2024

Distributed System Configuration and Shared Memory Operations

Question 1

Problem: Configure following options on server socket and tests them: SO_KEEPALIVE, SO_LINGER, SO_SNDBUF, SO_RCVBUF, TCP_NODELAY. It is a socket option that automatically sends keep-alive packets on an idle TCP connection to check if the other end is still responsive. This option helps detect and close inactive connections more reliably.

Server Code

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class MyServerSocket {
    public static void main(String[] args) {
        int port = 8080;
        try {
            ServerSocket serverSocket = new ServerSocket(port);

            System.out.println("Server is listening on port " + port);

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Accepted connection from " +
                    clientSocket.getInetAddress());

                clientSocket.setKeepAlive(true);
                clientSocket.setSoLinger(true, 30);
                clientSocket.setSendBufferSize(8192);
                clientSocket.setReceiveBufferSize(8192);
                clientSocket.setTcpNoDelay(true);

                clientSocket.close();
            }
        } catch (SocketException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output

Client Code

```
poseidon@okbe distributed systems % javac MyServerSocket.java
poseidon@okbe distributed systems % java MyServerSocket
Server is listening on port 8080
Accepted connection from /127.0.0.1
```

```java
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

public class ClientSocket {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 8080;

        try {
            Socket socket = new Socket(hostname, port);
            System.out.println("Connected to the server");

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter your name: ");
            String name = scanner.nextLine();

            System.out.print("Enter your roll number: ");
            String rollNumber = scanner.nextLine();

            String message = "Name: " + name + ", Roll Number: " +
                rollNumber;
            OutputStream output = socket.getOutputStream();
            output.write(message.getBytes());

            socket.close();
            System.out.println("Client socket closed");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Output

```
poseidon@okbe distributed systems % javac ClientSocket.java
poseidon@okbe distributed systems % java ClientSocket
Connected to the server
Enter your name: Bhuvana Teja Kanakam
Enter your roll number: se21ucse035
Client socket closed
poseidon@okbe distributed systems %
```

Question 2 : Incrementing a Counter in Shared Memory

SharedMemoryCounter.java Code

```
import java.util.Scanner;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class SharedCounter {
    private int counter = 0;
    private final Lock lock = new ReentrantLock();

    public void increment() {
        lock.lock();
        try {
            counter++;
            System.out.println("Counter: " + counter);
        } finally {
            lock.unlock();
        }
    }

    public int getCounter() {
        lock.lock();
        try {
            return counter;
        } finally {
            lock.unlock();
        }
    }
}

class IncrementingThread extends Thread {
    private final SharedCounter sharedCounter;

    public IncrementingThread(SharedCounter sharedCounter) {
        this.sharedCounter = sharedCounter;
    }

    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            sharedCounter.increment();
            try {
                Thread.sleep(100); // Sleep for demonstration
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

public class SharedMemoryCounter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();
```

```
        System.out.print("Enter your roll number: ");
        String rollNumber = scanner.nextLine();

        System.out.println("Name: " + name);
        System.out.println("Roll Number: " + rollNumber);

        SharedCounter sharedCounter = new SharedCounter();

        Thread thread1 = new IncrementingThread(sharedCounter);
        Thread thread2 = new IncrementingThread(sharedCounter);
        Thread thread3 = new IncrementingThread(sharedCounter);

        thread1.start();
        thread2.start();
        thread3.start();

        try {
            thread1.join();
            thread2.join();
            thread3.join();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        System.out.println("Final Counter Value: " + sharedCounter.getCounter());
    }
}
```

Output

```
poseidon@okbe distributed systems % javac SharedMemoryCounter.java
poseidon@okbe distributed systems % java SharedMemoryCounter
Enter your name: Bhuvana Teja Kanakam
Enter your roll number: se21ucse035
Name: Bhuvana Teja Kanakam
Roll Number: se21ucse035
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
Counter: 6
Counter: 7
Counter: 8
Counter: 9
Counter: 10
Counter: 11
Counter: 12
Counter: 13
Counter: 14
Counter: 15
Counter: 16
Counter: 17
Counter: 18
Counter: 19
Counter: 20
Counter: 21
Counter: 22
Counter: 23
Counter: 24
Counter: 25
Counter: 26
Counter: 27
Counter: 28
Counter: 29
Counter: 30
Final Counter Value: 30
poseidon@okbe distributed systems %
```