UNIT-II

Threads

When two or more tasks in parallel it is called as thread. In networking it provides the means for two or more software applications running on separate computers to communicate with one another.

The fundamental problem is that while it is easy to write code that handles each incoming connection and each new task as a separate process this solution doesn’t scale.

* By the time a server is attempting to handle a thousand or more simultaneous connections. The connection becomes slower or crawls.

There are two solution to this problem:

1. The first is to reuse process rather than spawning new ones.

For e.g. when server starts up with fixed number of processes say 300 are spawned to handle request. Incoming request are placed in queue this continues for all the 300 process which leads to building up and tearing down process.

* These are now made possible to do the work of 1000 process.

1. The second solution to this problem is to use light weight threads to handle connection instead of heavy weights processes. Each process has its own block of memory.

* Threads are easier on resource because they share memory.
* By combining pool of reuseable threads your server can run nine times faster all on the same hardware and network connection.

Threading:

* No more than one server socket can listen to a particular port at one time
* Since a server may need to handle many connection at once, server program tend to be heavily multithreaded.
* Generally the server socket passes off the actual process of connections to a separate thread.

Adding threading to a server:

* It is better to make your server multithreaded.
* There should be a loop that continually accepts new connections directly the socket should be passed to thread object that handles connection.
* Multithreaded is a good thing but still it is not a perfect solution.

For e.g.,

While(true){

Try{

Socket s=ss.accept();

Thread echoserver tes=new threadechoserver(s);

tes.start();

}

Catch(IOException e){}

\*every pass through this loop a new thread gets created. Every time connection is finished the thread is disposed.

\*Spawning a new thread for each connection takes lot of time especially on a heavily loaded server.

\*too many simultaneous threads overload a virtual machine.

Call back digest user interface

Package javaapplication29;

Import java.io.\*;

Public class callbackDigestUserInterface{

Public static void receiveDigest(byte[] digest, String name){

StringBuffer result=new StringBuffer(name);

Result.append(“:”);

}

System.out.println(result);

}

Public static void main(String[] args){

For(int i=0;i<args.length;i++){

//calculate the digest

File f=new File(args[i]);

CallbackDigest cb = new CallbackDigest(f);

Thread t= new Thread(cb);

t.Start();

}

}

}

\*however unlike the main method in other variations of this program this one only starts the threads for the files names on the command line.

\* it does not attempt to actually read, print out or in any otherway work with the results of the calculation.

\*receive digest is not invoked by main method or any method.

\*receive digest method runs inside the digesting threads rather than inside the main thread of execution.

Program-1.

Running Threads:

How to get the name of running thread

Import java.io.\*;

Import java.security.\*;

Public class newclass1 extends thread {

Public void run(){

For(int i=0;i<10;i++){

printMsg();

}

}

Public void printMsg(){

Thread t=Thread.currentThread();

String name=t.getName();

System.out.println(“name=” +name);

}

Public static void main(String [] args){

NewClass1 tt=new NewClass1();

tt.Start();

for(int i=0;i<10;i++)

tt.printMsg();

}

}

}

Program-2.

Write a program to display the names of all running threads using the getNamemethod

Import java.io.\*;

Import java.security .\*;

Import java .untility;

Public class newclass2 extends Thread{

Public static void main(String[] args){

Newclass2 t1=new newclass2();

t1.SetName(“thread1”)l

t1.Start();

Thread Group currentGroup= Thread.Current Thread().getThreadGroup();

int noThreads=currentGroup.activeCount();

Thread[] firstThreads=CurrentGroup.activeCount();

Current Group.enumerate(firstThreads);

For(int i=0;i<noThreads;i++)

System.out.println(“thread.no:”+i+”=”+firstThread[i].getName());

}

}

Polling:

\*it is one simple idea which turns out to be not very satisfactory, it is for CPU to keep checking for incoming data over and over whenever it finds data it processes it this method is called polling.

Race condition:

How many thread the program spawns.

\*The relative speed of the CPU and the disk on the system where this is run

\*the algorithm the java virtual machine uses to allot time to different threads this is called race condition.

Callbacks:

\*rather than asking the main program the repeatedly ask each thread whether it is finished.

\* we let the thread to tell the main program when it is finished this is called callback because the thread calls its creator when it is done.

\*this way main program can go to sleep when waiting for the threads to finish.

Thread Pools:

\*create a pool of thread when server launches

\*store incoming connections in a que and have threads in the pool.

\*progressive remove connections from the queue and process them.

\*the main change you need to make to implement this is to call accept() in run() rather than in main method.

Synchronization:

It is a process of making two or more data storage devices or programs ( in the same or different computers) having exactly the same information at a given time.

e.g. the guy and girl had really great synchronization and very compatiable with each other, which made me really happy to see.

e.g. you should have great synchronization between everyone in your office so that you can work together efficiently all the time.

Synchronized methods:

To make a method synchronized, simply add the synchronized keyword to its declaration,

Public class Synchronizedcounter{

Private int c=0;

Public Synchronized void increment(){

C++;

}

Public Synchronized void decrement(){

c--;

}

Public Synchronized int value(){

Return c;

}

}

Alternatives to Synchronization:

1. The first is useage local variable for the method these local variable do not have synchronization problem.
2. Everytime a method is entered the virtual machine creates completely new set of local variable for the method.

Synchronization statements:

* Another way to create Synchronization code is with Synchronized statement.
* Unlike Synchronized methods, Synchronized statements must specify the object that provides the intrinsic lock:

Public void addName(String Name){

Synchronized(this){

lastName=name;

namecount++;

nameList.add(name);

}

\*In this example the addName method needs to synchronize changes to lastName and namecount but also needs to avoid Synchronizing invocations of other object method.

\*without synchronized statements there would have too be separate Unsynchronised method for the sole purpose of invoking nameList.Add.

\*it is useful for improving concurrency with fine grained synchronization.

For e.g.

Mr.A has two instances c1 and c2 that are never used together all updates of these field are synchronized but there is no reason to prevent an update of c1 from being interleaved with an update of c2 and doing so reduces concurrency by creating unnecessary blocking.

* Instead of using synchronized methods or otherwise using lock associated with this we create two objects solely to provide locks.

Public class MrA{

Private long c1=0;

Private long c2=0;

Private object lock1=new object();

Private object lock2=new object();

Private void inc1(){

C1++;

}

}

Public void inc2(){  
Synchronized(lock2){

C2++;

}

}

}

Deadlock:

A situation where two or more threads are blocked for ever waiting for each other.

e.g.

\*the most important technique for preventing deadlock is to avoid unnecessary synchronization.

\*the best you can do is carefully consider whether the deadlock is likely to be a problem and design a code around it.

For e.g.,

If class A and class B need exclusive access to object x first and y second. If neither request Y unless it already possess x deadlock is not a problem.

Yielding:

It is a way to give up control this is done by invoking the static thread yield method.

Public static void yield()

* Before yielding a thread should make sure that it or its associated runnable object is in a consistent state that can be use by other objects.
* Making a thread yield is quite simple if the threads run() method simply consists of an infinite loop just put a call to Thread.yield() at the end of the lop.

Public void run(){

While(true){

//do threads work

Thread.yield();

}

}

Inet4Address and Inet6Address:

IPAddresses:

* IPAddressing is Logical Addressing
* IPAddressing works on Network Layer

IPAddressing have two version scheme

1.IPV4: 32 Bits

2.IPV6: 128 Bits

\*currently used version is IPV4 i.e., Four octet: 8Bit.8Bit.8Bit.8Bit

MacAddress:

* Media access controller.
* MAC Addressing is called Physical Addressing
* MAC address works in ATA-LinkLayer
* 48-Bit – 24bit(vendor code)+24bit(unique value)

Inet4Address: This is sub class of InetAddress class and used to represent 32 bits IPV4 Addresses e.g. 192.168.50.3

Inet6Address: it is 128 bit IPV6 address.

To see the basic useage of Inet4Address in Java create a class called simple InetAddress example.

Import java.net.Inet4Address;

Import java.net.UnknownHostException;

Public class javaapp12

Public static void main(string[] args){

String url=”google.com”;

Try{

Inet4Address address=(Inet4Address)Inet4Address.getByName(url);

System.out.println(“IP of”+url+”is +address.getHostAddress());

Catch(UnknownException e){

e.PrintStackTrace();

}

}

}

In the above example firstly I get the InetAddress of <http://www.google.com/> by using the parameter. That getByName() method is inherited from InetAddress.

\*After getting the InetAddress, use the getHostAddress() method to print the IP address of Google.com.

URL:

URL class: the URL class contains many methods for accessing the various parts of the URL being represented some of the methods in the URL class includes:

1. Public String getPath()-> Returns the path of the URL
2. Public String getQuery()-> Returns the query part of the URL
3. Public String getAuthority()->Returns the authority of the URL
4. Public int getPort()->Returns the port of the URL
5. Public int getDefaultPort()->Returns the default port for the protocol of the URL
6. Public String getProtocol()->Returns the protocol of the URL
7. Public String getHost()-> Returns the Host of the URL
8. Public String getFile()-> Returns the filename of the URL
9. Public String getRef()->Returns the reference part of the URL
10. Public URL connection openConnection()throws IOException-> opens a connection to the URL allowing a client to communicate with the resource.

How to Encode URL content in Java?

\*it is a utility class for HTML form encoding

\*This class contains static methods for converting a string to the application form url encoded MIME format

\*This class is useful to Encode query string values.

e.g.

Import java.io.UnsupportedEncodingException;

Import java.net.URLEcoder;

Public class Encoder{

Public static void main(String a[]){

Try

{

System.out.println(URLEncoder.encode(“String with spaces”,”UTF-8”));

System.out.println(URLEncoder.encode(“Special chars:&%”,”UTF-8”));

}

Catch(UnsupportedEncodingException ex) {

Ex.PrintStactkTrace();

}

}

}

Output:

String+with+spaces

Special+chars%3A+%26%25\*

How to Decode URL content in Java?

URLDecoder is a utility class for HTML for decoding. This class contains static methods for decoding a string from the application

e.g.

import java.io.UnsupportedEncodingException;

import java.net.URLDecoder;

public class decoder{

public static void main(String a[]){

try{

System.out.println(URLDecoder.decode(“Special+Chars%3A+%26%25\*+”,”UTF-8”));

}

Catch(UnsupportedEncodingException ex){

Ex.printStacktrace();

}

}

}

Output:

&%\*

Proxy: many system access the web and sometimesother non-HTTP parts of the Internet through Proxy server.

\*The proxy server makes request to the remote server from local client

\*the proxy server makes the requestto the remote and forwards the result back to the local client..

Proxyclass:

\*specifically this allows to choose different proxy server for different remote host.

The three kinds of proxies are

\*Proxy.Type.Direct

\*Proxy.Type.HTTP

\*Proxy.Type.Socks

SocketAddress address = new InetSocketAddress(“proxy.example.com”,80);

Proxy proxy=new proxy(proxy.Type.HTTP,address);

Proxy selector class:

Proxy selector class inspects the various systems properties and the URL’s to connect different hosts.

Communicating with server side program:

* The URL class makes it easy for java applets and application to communicate with server side program such as CGI, Servlets, PHP, and others that uses the Get()
* Server side program that uses the post method requires the URL connection class

<form name=”useinfo”method=”get” action=”info.html”>

<p>please give us your information so that we can send you spam </p>

<p>Name: <input type =”text” name=”name”/></p>

<p>E-mail:<input type=”text” name=”email”/></p>

<p>sex:<select name=”sex”>

<option>Male</option>

<option>Female</option>

</select></p>

<p><input name=”send” type =”submit” value=”send”/></p>

</form>

Accessing password protected sites:

\*java’s URL class can access sites that use HTTP authentication although you need to tell what username and password to use.

\*java does not provide support for sites that use non standard cookie-based authentication.

\*you can provide this support yourself using the URL connection class to r ead and write the HTTP headers where cookies are set and returned.

Authentication class:

* Authentication is an abstract class you must subclass it
* Different subclasses may retrieve the information in different ways
* Public abstract class authenticator extend object
* To make the URL class to use the subclass
* Install it as a default authenticator by passing it to the static authenticator. Set default method:

Public static void SetDefault(Authenticator a)

For if you have written an Authenticator subclass named DialogAuthen you will install it as

Authenticator.SetDefault(new DialogAuthen());

This is done once from this point when the URL class needs a username and password it will ask the Dialog Authen using the static Authenticator.requestPasswordAuthentication() method

Public static password Authentication request PasswordAuthentication(InetAddress address,int Port,String protocol,String Prompt,String scheme)throws SecurityException.

The port argument is the port on that host

-protocol argument is the application layer protocol by which the site is being accessed

-the HTTP server provides the prompt

-the scheme is authentication scheme being used

Protected Password Authentication getPasswordAuthentication()

->if you don’t want to authenticate this request return Null and Java will tell the server it does not know how to Authenticate the connection.

->if you submit incorrect user name or password java will call getPasswordAuthentication() again to provide the right data.

You can get more details about the request by invoking any of these methods: inherited from AuthenticatorSuperClass.

Protected final InetAddress getRequestingSite()

Protected final int getRequestingPort()

Protected final String getRequestingProtocol()

Protected final String getRequestingScheme()

Protected final String getRequestingHost()

The Password AuthenticationClass:

* Is very simple final class support two read only properties:username and password
* The username is string password in char array so that the password can be erased

Public PasswordAuthentication(String username, char[] Password)

Each is accessed via getter method:

Public String getUserName()

Public String char[]getPassword()

The JPasswordField Class:

* JPasswordField component is a useful tool for asking users for their password
* Public class JpasswordField extends JTextField
* jPasswordField stores the password as a char array so that when you are done with the password you can overwrite it with zeros
* it provides the getPassword method to return this

public char[] getPassword()