香农定理 :通道容量=带宽xlog2(1+SNR)奈奎斯特：比特率=2\*带宽\*log2 L,L为电平数量

三种网络交换技术：电路交换、报文交换、分组交换。

数据报交换（分组交换）：任何分组都当作单独的“小报文”处理，以报文交换方式单独处理分组;虚电路交换（分组交换）：通信双方在开始发送和接收分组之前，需要建立逻辑链路（虚电路）。所有分组都必须沿着事先建立的虚电路传输，需要虚呼叫建立和拆除

ISO-OSI体系：应用层，表示层，会话层。传输层，网络层，数据链路层，物理层

TCP/IP体系：应用层（HTTP 80，DNS 53，SNMP 161/162，TLNET 23，[FTP 20，POP3](ftp://FTP 20，POP3) 110，SMTP 25，），传输层（TCP，UDP），网络互联层，网络接口层（主机网络层）。

Socket chatSocket = new Socket(“127.0.0.1”, 5000);

InputStreamReader stream = new InputStreamReader (chatSocket.getInputStream());

BufferedReader reader = new BufferedReader(stream);

String message = reader.readLine();

PrintWriter writer = new PrintWriter(chatSocket.getOutputStream());

writer. println(“message to send”);

try{FileReader fr = new FileReader("input.txt");

BufferedReader input = new BufferedReader(fr);

FileWriter fw = new FileWriter("output.txt");

BufferedWriter output = new BufferedWriter(fw);

String s=null; int i=0;

while((s = input.readLine())!=null){ i++;

output.write(i + ": " + s);output.newLine();}

output.flush(); output.close(); input.close();

fw.close(); fr.close();}catch(IOException e){System.out.println(e);}

DataInputStream:readInt,readChars; DataOutputStream:writeInt,writeChars

字节输入流：new ByteArrayInputStream(output.toByteArray());)

字节输出流：new ByteArrayOutputStream();

CharArrayReader(output,toCharArray()),CharArrayWriter

线程对象：Thread t = new DigestThread(); t.start();

class DigestThread extends Thread {public void run() {}｝

Thread t = new DigestThread(new DigestRunnable());

设置优先级 t.setPriority(8)大的优先

class DigestRunnable implements Runnable{public void run() {}｝

synchronized (this) {...}同一个对象上同步的代码串行执行，否则可并行

public synchronized void writeEntry()定义是加锁

线程挂起t.join(long ms)throws InterruptedException

线程池:ExecutorService pool=Executors.newFixedThreadPool(n);

Runable task=new GzipRunnable(files[i]);pool.submit(task);提交任务

call方法有返回值:class Findmax implements Callable<Integer>{…

public Integer call(){… return max;}} Findmax task=new Findmax();

Future<Integer>future=pool.submit(task); value=future.get();

InetAddress类: 1.InetAddress address = InetAddress.getLocalHost();

System.out.println(address.getHostName());输出本机名

2.InetAddress address = InetAddress.getByName("www.szu.edu.cn");

System.out.println(address.getHostName());

InetAddress[] addresses = InetAddress.getAllByName(args[0]);返回数组

3.InetAddress address=InetAddress.getByName("141.146.8.66");

System.out.println(address.getHostName()); //访问DNS服务器

若IP地址不存在直接返回IP地址。异常UnknownHostException

getHostName返回域名;String getCanonicalHostName()查询DNS

String getHostAddress()返回ip Byte[] getAddress()

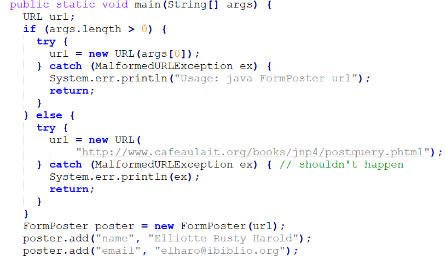
public void checkConnect(String hostname, int port)检查能否DNS解析

URL u=new(url);url为String。 ctach(MalformedURLexception ex)

URL u = new URL("http", "www.eff.org", "/blueribbon.html#intro");

从URL获取数据try{InputStream in=u.openStream();int c;

while ((c = in.read()) != -1)System.out.write(c);

in.close();}catch (IOException ex) {System.err.println(ex);}

或者try {URLConnection uc = u.openConnection();

InputStream in = uc.getInputStream();// read from the connection... } catch (IOException ex) {}} catch (MalformedURLException ex) {}

下载一个对象URL u = new URL(args[0]);Object o = u.getContent();

System.out.println("I got a " + o.getClass().getName());

很难预测获得哪种对象（InputStream/ImageProducer/AudioClip），需使用instanceof操作符进行检查。if(o instanceof URLImageSource){

读取URL信息getFile(), getHost(), getPort(),getProtocol(), getRef(), getQuery(),getPath(), getUserInfo(), getAuthority()

http://www.oreilly.com/index.html#p1 与……#q2 sameFile()两个指向相同的资源;equals()是两个是不等的;局限sameFile()不考虑片段标识符。

虽URL可视作一种URI的抽象，但Java的URL类并不是URI的子类

URI的语法模式包括：data, file, ftp, http, mailto, magnet, telnet, urn

形式://authority/path?query只检查URI语法，不检查协议

URLEncoder.encode()不区分特殊字符和需要编码的字符

Cookie是客户端技术，程序把每个用户的数据以cookie的形式写给用户各自的浏览器。当用户使用浏览器再去访问服务器中的web资源时，就会带着各自的数据去。这样，web资源处理的就是用户各自的数据了。2xx请求成功200 OK3xx 重定位或重定向4xx 客户端错误400 Bad Request401 Unauthorized403 Forbidden404 Not Found5xx 服务器错误500 Internal Server Error;GET方法用于获取URL所标识的一个资源的表示。POST：将资源的一个表示上传到已知URL的服务器，但没有指定服务器如何处理这个新提供的资源。启用cookie:编程时用URL类连接HTTP服务器，希望存储和获取HTTP服务器发送的cookie ,必须在程序中启用cookie。CookieManager manager = new CookieManager(); 阻塞第三方cookie\\ manager.setCookiePolicy (CookiePolicy.ACCEPT\_ORIGINAL\_SERVER);CookieHandler.setDefault(manager

下载Web页面：InputStream in = null; try {URL u = new URL(args[0]);

URLConnection uc = u.openConnection(); in = uc.getInputStream();

in = new BufferedInputStream(in); Reader r = new InputStreamReader(in);

int c; while ((c = r.read()) != -1) {System.out.print((char) c); }

头字段信息获取getContentType() getContentLength() getContentEncoding() getDate() getExpiration() getLastModified()

保存二进制文件：URLConnection uc=u.openConnection(); String contentType=uc.getContenType(); int contentLength=uc.getContentLength();

if(contentType.startsWith(“text/”)||contentLength==-1)throw new IOException();}try(InputStream raw=uc.getInputStream()){InputStream in =new BufferedInputStream(raw); byte[]data=new byte[contentLength];

int offset=0; while(offset<contenLength){int bytesRead= in.read (data,offset,data.length-offset); if(bytesRead==-1)break; offset+=byteRead;}

String filename=u.getFile(); filename=filename.substring(filename.lastIndexOf(‘/’)+1);try(FileOutputstream fout=new FileOutpuStream(filename)) fout.write(data); fout.flush();}

Expires：指示可以缓存这个资源的表示，直到指定的时间为止。

Cache-control:细粒度控制max-age=[seconds]从现在直到缓存项过期之前的秒数;s-maxage=[seconds]:共享缓存过期之前的秒数; no-store: 无论怎样都不缓存; Last-modified: 资源最后一次修改的时间; Etag: 资源改变时这个资源的唯一标识符当本地缓存的副本有一个不同的Etag时,才会执行get命令

Cache-control和Expires同时出现Cache-control和Expires同时出现

ResponseCache的put()方法返回一个CacheRequest类型 的对象。CacheRequest类表示在 ResponseCache 中存储资源的通道。CacheRequest类的实例包装了一个OutputStream对象，HTTP客户端程序可以调用该对象来将资源数据存储到缓存中。

true为读写服务器,false只能读if(!uc.getDoInput()){uc.setDoInput(true); }

if(!uc.getDoOutput()){uc.setDoOutput(true); }连接用于写数据到服务器

禁用缓存,确保获取文档最新版本uc.setUseCatches(false);

将cookie增加到连接uc.setRequestProperty("Cookie","username=elharo; password=ACD0X9F23JJJn6G; session=100678945");

URL u = new URL(“http://www.somehost.com/cgi-bin/acgi”);

//打开连接，准备post URLConnection uc = u.openConnection();

uc.setDoOutput(true);OutputStream raw = uc.getOutputStream();

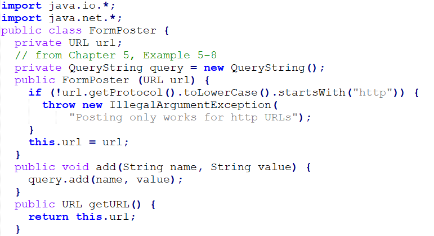
OutputStream buffered = new BufferedOutputStream(raw);

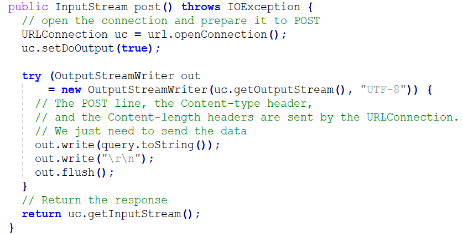
OutputStreamWriter out = new OutputStreamWriter(buffedred, “8859-1”);

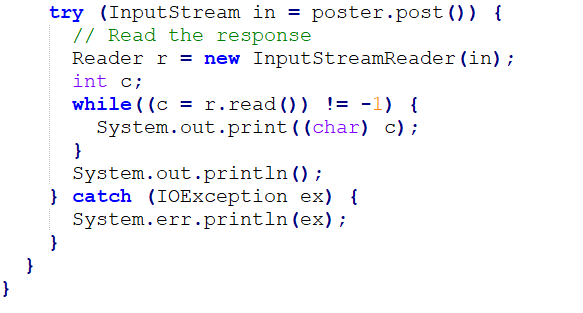
out.write(“first=Julie&middle=&last=Harting&work=String+Quartet\r\n”);

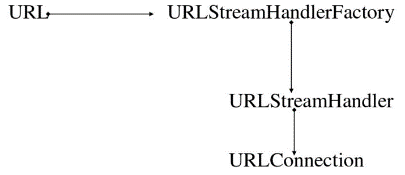
out.flush();out.close(); catch(IOExceptionex){}

post实例

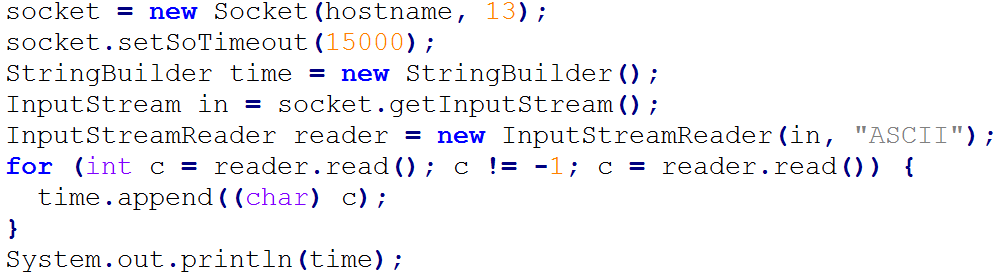








URL类的openConnection方法是通过调用URLStreamHandler类的同名方法创建的;URL类的openStream方法是通过调用URLConnection类的getInputStream方法获得的;URL类的getContent方法是调用URLConnection类的同名方法（该方法又调用了ContentHandler类的同名方法获得）

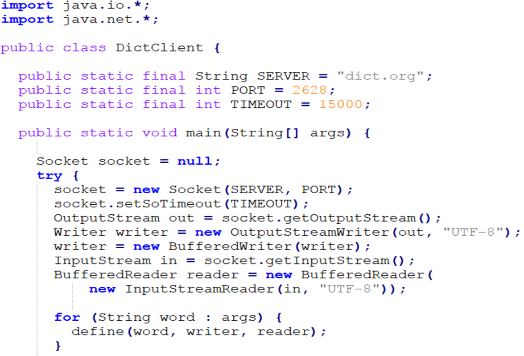
//从时间服务

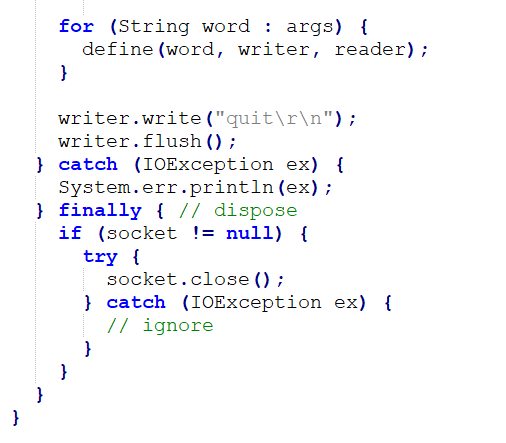
器获取数据

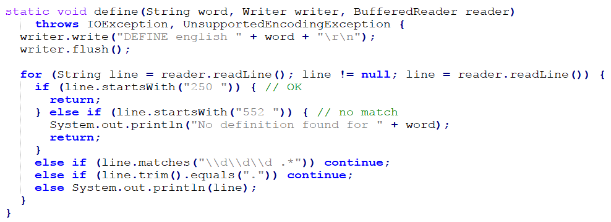
并打印

用parseDate(time.toString());生成Date对象;异常ParseException

Java内的Date类表示的1970年1月1日 00:00:00.000 之后的后毫秒数

英文拉丁文单词转换实例：





try (Socket connection = new Socket("www.oreilly.com", 80)) {Writer out = new OutputStreamWriter(connection.getOutputStream(), "8859\_1");

out.write("GET / HTTP 1.0\r\n\r\n");out.flush();

connection.shutdownOutput();read the response...

} catch (IOException ex) {}//半关闭Socket//以下构造后再连接

public void connect(SocketAddress endpoint, int timeout) throws IOException

获取Socket信息InetAddress getInetAddress();int getPort();InetAddress getLocalAddress();int getLocalPort()

setSoTimeout(int milliseconds)；int getSoTimeout();setReceiveBufferSize(int size); getReceiveBufferSize();getReceiveBufferSize();int getSendBufferSize()

SeverSocket实现try(ServerSocket server=new ServerSocket(PORT)){

while(true){try (Socket connection=server.accept()){

Writer out=new OutputStreamWriter(connection.getOutputStream());

Date now=new Date();out.write(now.toString()+”\r\n”);out.flush();

connection.close();}catch (IOException ex){ }}}catch(IOExecption ex){}

线程池Callable<Void>task=new DaytimeTask(connection);pool.submit(task);

private static class DaytimeTask implements Callable<Void>{private Socket connection; DaytimeTask(Socket connection){this.connection=connection;}

public Void call(){try{实现与上述一致}}}

检查ss是否打开ss.isBound && !ss.isClosed()

日志: logger.log(Level.SEVERE, "unexpected error " + ex.getMessage(), ex);

Level.SEVERE (highest value);Level.WARNING; Level.INFO;Level.CONFIG

Level.FINE; Level.FINER; Level.FINEST 日志级别

logger.info(new Date() + " " + connection.getRemoteSocketAddress());

构造函数ServerSocket(int port, int backlog, InetAddress bindAddr)

InetAddress local = InetAddress.getByName("192.168.210.122");

ServerSocket httpd = new ServerSocket(5776, 10, local);

还可以这样创建ServerSocket ss = new ServerSocket();

SocketAddress http = new InetSocketAddress(80);ss.bind(http);

获取服务器Socket信息InetAddress getInetAddress()

int getLocalPort();SocketAddress getLocalSocketAddress()

对称加密:加密方和解密方使用相同的密钥;非对称加密:加密方和解密方使用不同的密钥;组合:非对称加密传输对称加密的密钥, 用对称加密传输数据

创建安全Socket对象SocketFactory factory = SSLSocketFactory.getDefault(); Socket socket = (SSLSocket)factory.createSocket("login.ibiblio.org", 7000);

String[] getSupportedCipherSuites()指出给定socket上可用的算法组合

String[] getEnabledCipherSuites()指出给定socket上哪些密码组可以使用

void setEnabledCipherSuites(String[] suites)指定使用哪些密码组

ServerSocketFactory ssf = SSLServerSocketFactory.getDefault()；  
ServerSocket ss = ssf. createServerSocket(443)

一个非阻塞IO:

SocketAddress rama = new InetSocketAddress("rama.poly.edu", 19);

SocketChannel client = SocketChannel.open(rama);

ByteBuffer buffer = ByteBuffer.allocate(74);

int bytesRead = client.read(buffer); WritableByteChannel output = Channels.newChannel(System.out);buffer.flip(); output.write(buffer);

NIO客户端:client.configureBlocking(false);ServerSocketChannel serverChannel=ServerSocketChannel.open(); serverChannel.bind(new InetSocketAddress(19));serverChannel.configureBlocking(false);SocketChannel clientChannel =serverChannel.accept();clientChannel.configureBlocking(false);

clientChannel.write(buffer);

ByteBuffer, CharBuffer, ShortBuffer, IntBuffer, LongBuffer, FloatBuffer, DoubleBuffer; XxxBuffer buffer=XxxBuffer.allocate(int capacity) //创建方法

get()与put() 方法获取和放入数据buffer.put('H'); buffer.get();重新读写使用flip()limit在位置,位置为0;mark()指定到位置;reset();rewind()limit不变,位置0。

查看主机前1024个端口哪些在运行TCP服务器：String host = "localhost";for(int i=1;i<=1024;i++){Socket socket = new Socket();

socket.connect(new InetSocketAddress(host,i),200);System.out.println(i);}

断点续传：existingFileSize = file.length();try {URL url = new URL(fileURL);

connection = (HttpURLConnection) url.openConnection();// 设置Range头，用于断点续传connection.setRequestProperty("Range", "bytes=" + existingFileSize + "-");int responseCode = connection.getResponseCode();

if (responseCode == HttpURLConnection.HTTP\_PARTIAL) { // 206

inputStream = new BufferedInputStream(connection.getInputStream());

raf = new RandomAccessFile(file, "rw");raf.seek(existingFileSize);byte[] buffer = new byte[1024];int bytesRead;

while ((bytesRead = inputStream.read(buffer, 0, buffer.length)) != -1) {

raf.write(buffer, 0, bytesRead);}

} else if (responseCode == HttpURLConnection.HTTP\_OK) { // 200

inputStream = new BufferedInputStream(connection.getInputStream());

raf = new RandomAccessFile(file, "rw"); byte[] buffer = new byte[1024]; int bytesRead;

while ((bytesRead = inputStream.read(buffer, 0, buffer.length)) != -1) {

raf.write(buffer, 0, bytesRead)；}