一、基础篇

1.1 JVM

1.1.1. Java内存模型，Java内存管理，Java堆和栈，垃圾回收

http://www.jcp.org/en/jsr/detail?id=133

http://ifeve.com/jmm-faq/

1.1.2. 了解JVM各种参数及调优

1.1.3. 学习使用Java工具

jps, jstack, jmap, jconsole, jinfo, jhat, javap, …

http://kenai.com/projects/btrace

http://www.crashub.org/

https://github.com/taobao/TProfiler

https://github.com/CSUG/HouseMD

http://wiki.cyclopsgroup.org/jmxterm

https://github.com/jlusdy/TBJMap

1.1.4. 学习Java诊断工具

http://www.eclipse.org/mat/

http://visualvm.java.net/oqlhelp.html

1.1.5. 自己编写各种outofmemory，stackoverflow程序

HeapOutOfMemory

Young OutOfMemory

MethodArea OutOfMemory

ConstantPool OutOfMemory

DirectMemory OutOfMemory

Stack OutOfMemory

Stack OverFlow

1.1.6. 使用工具尝试解决以下问题，并写下总结

当一个Java程序响应很慢时如何查找问题

当一个Java程序频繁FullGC时如何解决问题，如何查看垃圾回收日志

当一个Java应用发生OutOfMemory时该如何解决，年轻代、年老代、永久代解决办法不同，导致原因也不同

1.1.7. 参考资料

http://docs.oracle.com/javase/specs/jvms/se7/html/

http://www.cs.umd.edu/~pugh/java/memoryModel/

http://gee.cs.oswego.edu/dl/jmm/cookbook.html

1.2. Java基础知识

1.2.1. 阅读源代码

java.lang.String

java.lang.Integer

java.lang.Long

java.lang.Enum

java.math.BigDecimal

java.lang.ThreadLocal

java.lang.ClassLoader & java.net.URLClassLoader

java.util.ArrayList & java.util.LinkedList

java.util.HashMap & java.util.LinkedHashMap & java.util.TreeMap

java.util.HashSet & java.util.LinkedHashSet & java.util.TreeSet

1.2.2. 熟悉Java中各种变量类型

1.2.3. 熟悉Java String的使用，熟悉String的各种函数

1.2.4. 熟悉Java中各种关键字

1.2.5. 学会使用List，Map，Stack，Queue，Set

上述数据结构的遍历

上述数据结构的使用场景

Java实现对Array/List排序

java.uti.Arrays.sort()

java.util.Collections.sort()

Java实现对List去重

Java实现对List去重，并且需要保留数据原始的出现顺序

Java实现最近最少使用cache，用LinkedHashMap

1.2.6. Java IO&Java NIO，并学会使用

java.io.*

java.nio.*

nio和reactor设计模式

文件编码，字符集

1.2.7. Java反射与javassist

反射与工厂模式

java.lang.reflect.*

1.2.8. Java序列化

java.io. Serializable

什么是序列化，为什么序列化

序列化与单例模式

google序列化protobuf

1.2.9. 虚引用，弱引用，软引用

java.lang.ref.*

实验这些引用的回收

1.2.10. 熟悉Java系统属性

java.util.Properties

1.2.11. 熟悉Annotation用法

java.lang.annotation.*

1.2.12. JMS

javax.jms.*

1.2.13. JMX

java.lang.management.*

javax.management.*

1.2.14. 泛型和继承，泛型和擦除

1.2.15. 自动拆箱装箱与字节码

1.2.16. 实现Callback

1.2.17. java.lang.Void类使用

1.2.18. Java Agent，premain函数

java.lang.instrument

1.2.19. 单元测试

Junit，http://junit.org/

Jmockit，https://code.google.com/p/jmockit/

djUnit，http://works.dgic.co.jp/djunit/

1.2.20. Java实现通过正则表达式提取一段文本中的电子邮件，并将@替换为#输出

java.lang.util.regex.*

1.2.21. 学习使用常用的Java工具库

commons.lang, commons.*...

guava-libraries

netty

1.2.22. 什么是API&SPI

http://en.wikipedia.org/wiki/Application_programming_interface

http://en.wikipedia.org/wiki/Service_provider_interface

1.2.23. 参考资料

JDK src.zip 源代码

http://openjdk.java.net/

http://commons.apache.org/

https://code.google.com/p/guava-libraries/

http://netty.io/

http://stackoverflow.com/questions/2954372/difference-between-spi-and-api

http://stackoverflow.com/questions/11404230/how-to-implement-the-api-spi-pattern-in-java

1.3. Java并发编程

1.3.1. 阅读源代码，并学会使用

java.lang.Thread

java.lang.Runnable

java.util.concurrent.Callable

java.util.concurrent.locks.ReentrantLock

java.util.concurrent.locks.ReentrantReadWriteLock

java.util.concurrent.atomic.Atomic*

java.util.concurrent.Semaphore

java.util.concurrent.CountDownLatch

java.util.concurrent.CyclicBarrier

java.util.concurrent.ConcurrentHashMap

java.util.concurrent.Executors

1.3.2. 学习使用线程池，自己设计线程池需要注意什么

1.3.3. 锁

什么是锁，锁的种类有哪些，每种锁有什么特点，适用场景是什么

在并发编程中锁的意义是什么

1.3.4. synchronized的作用是什么，synchronized和lock

1.3.5. sleep和wait

1.3.6. wait和notify

1.3.7. 写一个死锁的程序

1.3.8. 什么是守护线程，守护线程和非守护线程的区别以及用法

1.3.9. volatile关键字的理解

C++ volatile关键字和Java volatile关键字

happens-before语义

编译器指令重排和CPU指令重排

http://en.wikipedia.org/wiki/Memory_ordering

http://en.wikipedia.org/wiki/Volatile_variable

http://preshing.com/20130702/the-happens-before-relation/

1.3.10. 以下代码是不是线程安全？为什么？如果为count加上volatile修饰是否能够做到线程安全？你觉得该怎么做是线程安全的？

```
publicclassSample{privatestaticint count =0;publicstaticvoid increment(){ count++;}}
```

1.3.11. 解释一下下面两段代码的差别

```
// 代码1publicclassSample{privatestaticint count =0;synchronizedpublicstaticvoid increment(){ count++;}}//
代码2publicclassSample{privatestaticAtomicInteger count =newAtomicInteger(0);publicstaticvoid increment(){
count.getAndIncrement();}}
```

1.3.12. 参考资料

http://book.douban.com/subject/10484692/

http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

二、 进阶篇

2.1. Java底层知识

2.1.1. 学习了解字节码、class文件格式

http://en.wikipedia.org/wiki/Java_class_file

http://en.wikipedia.org/wiki/Java_bytecode

http://en.wikipedia.org/wiki/Java_bytecode_instruction_listings

http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/

http://asm.ow2.org/

2.1.2. 写一个程序要求实现javap的功能（手工完成，不借助ASM等工具）

如Java源代码：

```
publicstaticvoid main(String[] args){int i =0; i +=1; i *=1;System.out.println(i);}
```

编译后读取class文件输出以下代码：

```
publicstaticvoid main(java.lang.String[]);Code:Stack=2,Locals=2,Args_size=10: iconst_0 1: istore_1 2: iinc
1,15: iload_1 6: iconst_1 7: imul 8: istore_1 9: getstatic #2; //Field
java/lang/System.out:Ljava/io/PrintStream;12: iload_1 13: invokevirtual #3; //Method
java/io/PrintStream.println:(I)V16:returnLineNumberTable: line 4:0 line 5:2 line 6:5 line 7:9 line 8:16
```

2.1.3. CPU缓存，L1，L2，L3和伪共享

http://duartes.org/gustavo/blog/post/intel-cpu-caches/

http://mechanical-sympathy.blogspot.com/2011/07/false-sharing.html

2.1.4. 什么是尾递归

2.1.5. 熟悉位运算

用位运算实现加、减、乘、除、取余

2.1.6. 参考资料

http://book.douban.com/subject/1138768/

http://book.douban.com/subject/6522893/

http://en.wikipedia.org/wiki/Java_class_file

http://en.wikipedia.org/wiki/Java_bytecode

http://en.wikipedia.org/wiki/Java_bytecode_instruction_listings

2.2. 设计模式

2.2.1. 实现AOP

CGLIB和InvocationHandler的区别

http://cglib.sourceforge.net/

动态代理模式

Javassist实现AOP

http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/

ASM实现AOP

http://asm.ow2.org/

2.2.2. 使用模板方法设计模式和策略设计模式实现IOC

2.2.3. 不用synchronized和lock，实现线程安全的单例模式

2.2.4. nio和reactor设计模式

2.2.5. 参考资料

http://asm.ow2.org/

http://cglib.sourceforge.net/

http://www.javassist.org/

2.3. 网络编程知识

2.3.1. Java RMI，Socket，HttpClient

2.3.2. 用Java写一个简单的静态文件的HTTP服务器

实现客户端缓存功能，支持返回304

实现可并发下载一个文件

使用线程池处理客户端请求

使用nio处理客户端请求

支持简单的rewrite规则

上述功能在实现的时候需要满足“开闭原则”

2.3.3. 了解nginx和apache服务器的特性并搭建一个对应的服务器

http://nginx.org/

http://httpd.apache.org/

2.3.4. 用Java实现FTP、SMTP协议

2.3.5. 什么是CDN？如果实现？DNS起到什么作用？

搭建一个DNS服务器

搭建一个 Squid 或 Apache Traffic Server 服务器

http://www.squid-cache.org/

http://trafficserver.apache.org/

http://en.wikipedia.org/wiki/Domain_Name_System

2.3.6. 参考资料

http://www.ietf.org/rfc/rfc2616.txt

http://tools.ietf.org/rfc/rfc5321.txt

http://en.wikipedia.org/wiki/Open/closed_principle

2.4. 框架知识

spring，spring mvc，阅读主要源码

ibatis，阅读主要源码

用spring和ibatis搭建java server

2.5. 应用服务器知识

熟悉使用jboss，https://www.jboss.org/overview/

熟悉使用tomcat，http://tomcat.apache.org/

熟悉使用jetty，http://www.eclipse.org/jetty/

三、 高级篇

3.1. 编译原理知识

3.1.1. 用Java实现以下表达式解析并返回结果（语法和Oracle中的select sysdate-1 from dual类似）

```
sysdate sysdate -1 sysdate -1/24 sysdate -1/(12*2)
```

3.1.2. 实现对一个List通过DSL筛选

```
QList<Map<String,Object>> mapList =newQList<Map<String,Object>>; mapList.add({"name":"hatter test"});
mapList.add({"id":-1,"name":"hatter test"}); mapList.add({"id":0,"name":"hatter test"});
mapList.add({"id":1,"name":"test test"}); mapList.add({"id":2,"name":"hatter test"});
mapList.add({"id":3,"name":"test hatter"}); mapList.query("id is not null and id > 0 and name like
'%hatter%'");
```

要求返回列表中匹配的对象，即最后两个对象；

3.1.3. 用Java实现以下程序（语法和变量作用域处理都和JavaScript类似）：

代码：

```
var a =1;var b =2;var c =function(){var a =3; println(a); println(b);};c();println(a);println(b);
```

输出：

```
3212
```

3.1.4. 参考资料

http://en.wikipedia.org/wiki/Abstract_syntax_tree

https://javacc.java.net/

http://www.antlr.org/

3.2. 操作系统知识

Ubuntu

Centos

使用linux，熟悉shell脚本

3.3. 数据存储知识

3.3.1. 关系型数据库

MySQL

如何看执行计划

如何搭建MySQL主备

binlog是什么

Derby，H2，PostgreSQL

SQLite

3.3.2. NoSQL

Cache

Redis

Memcached

Leveldb

Bigtable

HBase

Cassandra

Mongodb

图数据库

neo4j

3.3.3. 参考资料

http://db-engines.com/en/ranking

http://redis.io/

https://code.google.com/p/leveldb/

http://hbase.apache.org/

http://cassandra.apache.org/

http://www.mongodb.org/

http://www.neo4j.org/

3.4. 大数据知识

3.4.1. Zookeeper，在linux上部署zk

3.4.2. Solr，Lucene，ElasticSearch

在linux上部署solr，solrcloud，，新增、删除、查询索引

3.4.3. Storm，流式计算，了解Spark，S4

在linux上部署storm，用zookeeper做协调，运行storm hello world，local和remote模式运行调试storm topology。

3.4.4. Hadoop，离线计算

Hdfs：部署NameNode，SecondaryNameNode，DataNode，上传文件、打开文件、更改文件、删除文件

MapReduce：部署JobTracker，TaskTracker，编写mr job

Hive：部署hive，书写hive sql，得到结果

Presto：类hive，不过比hive快，非常值得学习

3.4.5. 分布式日志收集flume，kafka，logstash

3.4.6. 数据挖掘，mahout

3.4.7. 参考资料

http://zookeeper.apache.org/

https://lucene.apache.org/solr/

https://github.com/nathanmarz/storm/wiki

http://hadoop.apache.org/

http://prestodb.io/

http://flume.apache.org/，http://logstash.net/，http://kafka.apache.org/

http://mahout.apache.org/

3.5. 网络安全知识

3.5.1. 什么是DES、AES

3.5.2. 什么是RSA、DSA

3.5.3. 什么是MD5，SHA1

3.5.4. 什么是SSL、TLS，为什么HTTPS相对比较安全

3.5.5. 什么是中间人攻击、如果避免中间人攻击

3.5.6. 什么是DOS、DDOS、CC攻击

3.5.7. 什么是CSRF攻击

3.5.8. 什么是CSS攻击

3.5.9. 什么是SQL注入攻击

3.5.10. 什么是Hash碰撞拒绝服务攻击

3.5.11. 了解并学习下面几种增强安全的技术

http://www.openauthentication.org/

HOTP http://www.ietf.org/rfc/rfc4226.txt

TOTP http://tools.ietf.org/rfc/rfc6238.txt

OCRA http://tools.ietf.org/rfc/rfc6287.txt

http://en.wikipedia.org/wiki/Salt_(cryptography)

3.5.12. 用openssl签一个证书部署到apache或nginx

3.5.13. 参考资料

http://en.wikipedia.org/wiki/Cryptographic_hash_function

http://en.wikipedia.org/wiki/Block_cipher

http://en.wikipedia.org/wiki/Public-key_cryptography

http://en.wikipedia.org/wiki/Transport_Layer_Security

http://www.openssl.org/

https://code.google.com/p/google-authenticator/

四、 扩展篇

4.1. 相关知识

4.1.1. 云计算，分布式，高可用，可扩展

4.1.2. 虚拟化

https://linuxcontainers.org/

http://www.linux-kvm.org/page/Main_Page

http://www.xenproject.org/

https://www.docker.io/

4.1.3. 监控

http://www.nagios.org/

http://ganglia.info/

4.1.4. 负载均衡

http://www.linuxvirtualserver.org/

4.1.5. 学习使用git

https://github.com/

https://git.oschina.net/

4.1.6. 学习使用maven

http://maven.apache.org/

4.1.7. 学习使用gradle

http://www.gradle.org/

4.1.8. 学习一个小语种语言

Groovy

Scala

LISP, Common LISP, Schema, Clojure

R

Julia

Lua

Ruby

4.1.9. 尝试了解编码的本质

了解以下概念

ASCII, ISO-8859-1

GB2312, GBK, GB18030

Unicode, UTF-8

不使用 String.getBytes() 等其他工具类/函数完成下面功能

```
publicstaticvoid main(String[] args)throwsIOException{String str ="Hello, 我们是中国人。";byte[] utf8Bytes = toUTF8Bytes(str);FileOutputStream fos =newFileOutputStream("f.txt"); fos.write(utf8Bytes); fos.close();}publicstaticbyte[] toUTF8Bytes(String str){returnnull;// TODO}
```

想一下上面的程序能不能写一个转GBK的?

写个程序自动判断一个文件是哪种编码

4.1.10. 尝试了解时间的本质

时区 & 冬令时、夏令时

http://en.wikipedia.org/wiki/Time_zone

ftp://ftp.iana.org/tz/data/asia

http://zh.wikipedia.org/wiki/%E4%B8%AD%E5%9C%8B%E6%99%82%E5%8D%80

闰年

http://en.wikipedia.org/wiki/Leap_year

闰秒

ftp://ftp.iana.org/tz/data/leapseconds

System.currentTimeMillis() 返回的时间是什么

4.1.11. 参考资料

http://git-scm.com/

http://en.wikipedia.org/wiki/UTF-8

http://www.iana.org/time-zones

4.2. 扩展学习

4.2.1. JavaScript知识

4.2.1.1. 什么是prototype

修改代码，使程序输出“1 3 5”：

http://jsfiddle.net/Ts7Fk/

4.2.1.2. 什么是闭包

看一下这段代码，并解释一下为什么按Button1时没有alert出“This is button: 1”，

如何修改：

http://jsfiddle.net/FDPj3/1/

4.2.1.3. 了解并学习一个JS框架

jQuery

ExtJS

ArgularJS

4.2.1.4. 写一个Greasemonkey插件

http://en.wikipedia.org/wiki/Greasemonkey

4.2.1.5. 学习node.js

http://nodejs.org/

4.2.2. 学习html5

ArgularJS，https://docs.angularjs.org/api

4.2.3. 参考资料

http://www.ecmascript.org/

http://jsfiddle.net/

http://jsbin.com/

http://runjs.cn/

http://userscripts.org/

五、 推荐书籍

《深入Java虚拟机》

《深入理解Java虚拟机》

《Effective Java》

《七周七语言》

《七周七数据》

《Hadoop技术内幕》

《Hbase In Action》

《Mahout In Action》

《这就是搜索引擎》

《Solr In Action》

《深入分析Java Web技术内幕》

《大型网站技术架构》

《高性能MySQL》

《算法导论》

《计算机程序设计艺术》

《代码大全》

《JavaScript权威指南》