

**一、流式IO**

**1输入流(InputStream)**

基类方法：InputStream

int available() 返回可读取的字节数(非阻塞)

close() 关闭流并释放所有资源

int read() 读取一个字节 (抽象)

int read(byte[]) 读取字节到数组

long skip(long) 跳过指定字节

文件流：FileInputStream < InputStream

FileInputStream(File|FileDescriptor|String) 用文件/描述符/路径构建

FileChannel getChannel() 返回流的通道对象

FileDescriptor getFD() 返回文件描述符对象

数据流：ByteArrayInputStream < InputStream

ByteArrayInputStream(byte[]) 用字节数组构造流

mark(int) 标记读指针，以便reset后重读

reset() 将读指针复位到mark标记

管道流：PipedInputStream < InputStream

PipedInputStream(PipeOutputStream, int) 创建管道，int指定输入流缓冲大小

缓冲流：BufferdInputStream << InputStream

BufferedInputStream(InputStream, <int>) 构造缓冲流，int指定缓冲区大小

mark(int) 标记读指针，以便reset后重读

reset() 将读指针复位到mark标记

**2输出流(OutputStream)**

基类方法：OutputStream

close() 关闭流并释放所有资源

flush() 同步缓冲(针对有缓冲IO)

write(int b) 写入一个字节(抽象)

write(byte[]) 写入指定字节数组

文件流：FileOutputStream < OutputStream

FileOutputStream(File|FileDescriptor|String) 用文件/描述符/路径构建

FileOutputStream(File|String, boolean) 布尔指定是否以追加输出的方式

FileChannel getChannel() 返回流的通道对象NIO

FileDescriptor getFD() 返回文件描述符对象

数据流：ByteArrayOutputStream < ByteArrayOutputStream

ByteArrayOutputStream(<int>) 构造指定大小流(缺省32字节，可自增)

int size() 返回流的数据量(字节)

byte[] toByteArray() 返回流包含的字节数组

writeTo(OutputStream) 将流中数据写入另一个输出流

管道流：PipedOutputStream < OutputStream

PipedOutputStream(PipedInputStream) 创建管道

缓冲流：BufferedOutputStream << OutputStream

BufferedOutputStream(OutputStream, <int>) 构造缓冲流，int指定缓冲区大小

flush() 同步缓冲

**二、文件IO**

**1随机访问IO (RandomAccessFile)**

RandomAccessFile(File|String, (String)mode) 构造指定模式的随机访问

File|String：指定File对象或完整路径

mode：指定打开模式

·r 只读模式

·rw 读写模式，带有缓冲，文件不存在创建

·rws 读写模式，写入即同步

close() 关闭随机访问文件

FileChannel getChannel() 返回文件的通道对象NIO

FileDescriptor getFD() 返回文件描述符对象

long length() 返回文件的长度

int read() 读取一个字节

int read(byte[]) 读取字节到数组

seek(long) 移动文件读写指针

write(byte[]) 写入指定字节数组

**2文件通道IO (FileChannel)**

File...Stream、RandomAccessFile对象getChannel返回

信息

long position() 返回读写指针位置

long size() 返回文件的大小

操作

force(boolean) 同步缓冲到设备

position(long) 设置读写指针位置

read(ByteBuffer) 读取字节到缓冲区

write(ByteBuffer) 写入字节到缓冲区

**3文件锁(多进程同步)**

获取锁：FileChannel

FileLock lock() 获取排他锁(阻塞)

FileLock lock(position, size, shared) 获取局部锁(阻塞)

position long型，指定锁住内容的开始部分

size long型，指定锁住内容的长度

shared boolean型，指定共享锁(单写多读)

FileLock tryLock() 获取排他锁(非阻塞)

FileLock tryLock(position, size, shared) 获取局部锁(非阻塞)

锁管理：FileLock

FileChannel channel() 返回锁对应的通道

boolean isShared() 判断是否为共享锁

boolean overlaps(position, size) 判断是否局部锁住指定区域

release() 释放锁

**4文件内存影射(**MappedByteBuffer < ByteBuffer**)**

实现映射：FileChannel

MappedByteBuffer map(mode, position, size) 返回映射缓冲对象

position long型，映射文件内容的起始

size long型，映射文件内容的长度

mode 映射模式

·MapMode.READ\_ONLY：不能更改缓冲

·MapMode.READ\_WRITE：更改缓冲将同步到文件

·MapMode.PRIVATE：更改缓冲不同步到文件

**5缓冲区(ByteBuffer < Buffer)**

基类方法

字节缓冲：可开设虚拟机缓冲、系统缓冲

static ByteBuffer allocate(int) 分配一个虚拟机缓冲区

static ByteBuffer allocateDirect() 分配一个系统缓冲区

static wrap(byte[]) 以指定字节构造缓冲

byte[] array() 返回缓冲区中的数据

get(byte[]) 读取字节到数组

put(byte[]) 写入字节到缓冲