//创建文件，如果文件存在则更新时间；如果不存在，创建一个空文件

//创建空文件的方式为：

// final OutputStream out = openOutputStream(file);

// IOUtils.closeQuietly(out);

FileUtils.touch(new File("D://test/test4"));

//文件内容的对比

FileUtils.contentEquals(new File("D://test/test1"),new File("D://test/test2"));

FileUtils.contentEqualsIgnoreEOL(new File("D://test/test1"),new File("D://test/test2"),null);//忽略换行符，第三个参数是字符集

//根据URL获取文件

FileUtils.toFile(new URL("file://D://test/test1"));

FileUtils.toFiles(null);

FileUtils.toURLs(new File[]{new File("D://test/test1")});

//文件拷贝

FileUtils.copyFileToDirectory(new File("/test1"),new File("/dir"),true);//第三个参数是否更新时间

FileUtils.copyFile(new File("/source"),new File("/target"),true);//TODO 阅读doCopy源码

//目录拷贝

File srcDir = new File("/source");

File destDir = new File("/target");

FileUtils.copyDirectoryToDirectory(new File("/source"),new File("/target"));

FileUtils.copyDirectory(new File("/source"),new File("/target"));

FileUtils.copyDirectory(srcDir, destDir, DirectoryFileFilter.DIRECTORY);//仅仅拷贝目录

IOFileFilter txtSuffixFilter = FileFilterUtils.suffixFileFilter(".txt");// 创建.txt过滤器

IOFileFilter txtFiles = FileFilterUtils.andFileFilter(FileFileFilter.FILE, txtSuffixFilter);

// 创建包含目录或者txt文件的过滤器

FileFilter filter = FileFilterUtils.orFileFilter(DirectoryFileFilter.DIRECTORY, txtFiles);

// Copy using the filter

FileUtils.copyDirectory(srcDir, destDir, filter);//TODO 阅读doCopyDirectory源码

//文件拷贝

FileUtils.copyInputStreamToFile(new FileInputStream("/test"),new File("/test"));

FileUtils.copyURLToFile(new URL("file:/test"),new File("/test"));

//删除文件

FileUtils.deleteDirectory(new File("/test"));//删除目录下所有的内容

FileUtils.deleteQuietly(new File("/test"));//如果是目录，会级联删除；不会抛出异常

//判断文件是否存在

FileUtils.directoryContains(new File("/dir"),new File("/file"));

//清除目录中的内容,不会删除该目录；

//先verifiedListFiles检查目录，检查目录是否为目录、是否存在，然后调用listFiles，如果返回null，则抛出异常

//遍历目录中的文件，如果是目录则递归删除；如果是文件则强制删除，删除失败（文件不存在或无法删除）都会抛出异常

FileUtils.cleanDirectory(new File("/dir"));//TODO verifiedListFiles <------2016/11/24

//等待一个文件xx秒，知道文件创建后才返回。每max(100,remainning)循环检查一次

while(FileUtils.waitFor(new File("/dir"),60)){}

//读取目标文件，内部调用IOUtils.toString(inputstream,"utf-8")

String str = FileUtils.readFileToString(new File("/dir"),"utf-8");

//内部调用IOUtils.toByteArray(in)

byte[] bytes = FileUtils.readFileToByteArray(new File("/dir"));

//内部调用IOUtils.readLines(in, Charsets.toCharset(encoding));

List<String> strs = FileUtils.readLines(new File("/dir"),"utf-8");

//内部调用IOUtils.lineIterator(in, encoding)

FileUtils.lineIterator(new File("/dir"),"utf-8");

//四个参数分别为：目标文件，写入的字符串，字符集，是否追加

FileUtils.writeStringToFile(new File("/target"),"string","utf-8",true);

//write可以接受charsequence类型的数据，string,stringbuilder和stringbuffer都是实现了charsequence接口

FileUtils.write(new File("/target"),"target char sequence","utf-8",true);

FileUtils.writeByteArrayToFile(new File("/target"),"bytes".getBytes());//(file,字符数组)

FileUtils.writeByteArrayToFile(new File("/target"),"bytes".getBytes(),true);//(file,字符数组，是否追加)

FileUtils.writeByteArrayToFile(new File("/target"),"bytes".getBytes(),0,10);//(file,字符数组，起始位置，结束位置)

FileUtils.writeByteArrayToFile(new File("/target"),"bytes".getBytes(),0,10,true);//(file,字符数组，起始位置，结束位置，是否追加)

//writeLines多了一个lineEnding参数

FileUtils.writeLines(new File("/target"),"utf-8", FileUtils.readLines(new File("/target"),"utf-8"));

//强制删除

FileUtils.forceDelete(new File("/target"));

//在JVM

FileUtils.forceDeleteOnExit(new File("/target"));

//强制创建文件目录，如果文件存在，会抛出异常

FileUtils.forceMkdir(new File("/target"));

//强制创建父级目录

FileUtils.forceMkdirParent(new File("/xxxx/target"));

//如果是文件，直接读取文件大小；如果是目录，级联计算文件下的所有文件大小

FileUtils.sizeOf(new File("/target"));//返回Long

FileUtils.sizeOfAsBigInteger(new File("/target"));//返回BigInteger

FileUtils.sizeOfDirectory(new File("/target"));

FileUtils.sizeOfDirectoryAsBigInteger(new File("/target"));

//对比文件新旧

FileUtils.isFileNewer(new File("/target"),new File("/file"));

FileUtils.isFileOlder(new File("/target"), new Date());

FileUtils.checksum(new File("/target"),new CRC32());

FileUtils.checksumCRC32(new File("/target"));

FileUtils.moveDirectory(new File("/target"),new File("/file"));

FileUtils.moveDirectoryToDirectory(new File("/target"),new File("/file"),true);

FileUtils.moveFile(new File("/target"),new File("/file"));

FileUtils.moveFileToDirectory(new File("/target"),new File("/dir"),true);

FileUtils.moveToDirectory(new File("/target"),new File("/dir"),true);

FileUtils.isSymlink(new File("/target"));