1. 文件系统
2. 概述

Path和Files类封装了在机器上处理文件系统所需的所有功能。

1. Path

Path表示的是一个目录名序列，其后可以跟着一个文件名，路径的第一部分可以是根路径，如“/”，以根路径起始的是绝对路径，否则就是相对路径。静态的Path.get方法接受多个字符串，用默认文件系统的路径分隔符连接起来，返回一个Path对象，如下所示：

Path absolute = Paths.get(“/home”, “harry”);

Path对象对应的并一定是一个存在的文件，仅仅是一个抽象的名字序列，在创建文件时，首先要创建一个路径。要解析路径，使用resolve()方法返回一个路径，对于p.resolve(q)，如果q是绝对路径则返回q；否则，将返回新的路径为p后面跟着q。

1. Files
2. 读写文件

Files类能够快速地读取文件的内容，如

读取所有字节：

byte[] bytes = Files.readAllBytes(path);

读取所有文本：

var content = Files.readString(path, charset);

对于读取中等长度的文本文件，可以使用Files类，对于大文件还是使用IO流和读入写出器。

2. 创建文件和目录

创建新目录：

Files.createDirectory(path);

创建空文件：

Files.createFile(path);

在创建时可以指定属性，如拥有者，权限等等。

3. 复制，移动，删除文件

复制文件：

Files.copy(fromPath, toPath);

移动文件：

Files.move(fromPath, toPath);

同样的，可以添加选项，如覆盖已有的目标路径StandardCopyOption.REPLACE\_EXISTING，复制所有的文件属性StandartCopyOption.COPY\_ATTRIBUTES。

4. 访问目录中的项

静态的Files.list方法返回一个可以读取目录中各个项的Stream<Path>对象，目录是被惰性读取的，这是十分高效的，由于读取目录涉及到需要关闭的系统资源，因此使用try块：

try(Stream<Path> entries = Files.list(pathToDirectory)){}

list方法不会进入子目录，要处理子目录，使用Files.walk方法。而对于更加细粒度的遍历目录，应该使用Files.newDirectoryStream对象，产生一个DirectoryStream，注意，这不是Stream的子接口，而是专门用于目录遍历的接口，是Iterable的子接口，可以在增强for循环中使用目录流，如下所示：

try(DirectoryStream<Path> entries = Files.newDirectoryStream(str)){}

如果想要访问某个目录的所有子孙成员，可以调用walkFileTree方法， 传入一个FileVisitor类型的对象，便捷类SimpleFileVisitor实现了FileVisitor接口，这个对象有以下的通知：

访问文件或目录时： FileVisitResult visitFile(…);

在目录被处理前： FileVisitResult preVisitDirectory(…);

在目录被处理后： FileVisitResult postVisitDirectory(…);

FileVisitResult对象指的是操作的返回结果，是继续访问下一个文件：FileVisitResult.CONTINUE，FileVisitResult.SKIP\_SUBTREE等等

walkFileTree方法通常用于延迟操作，如删除目录树时，需要在移除当前目录中的所有文件之后，才能够移除该目录。