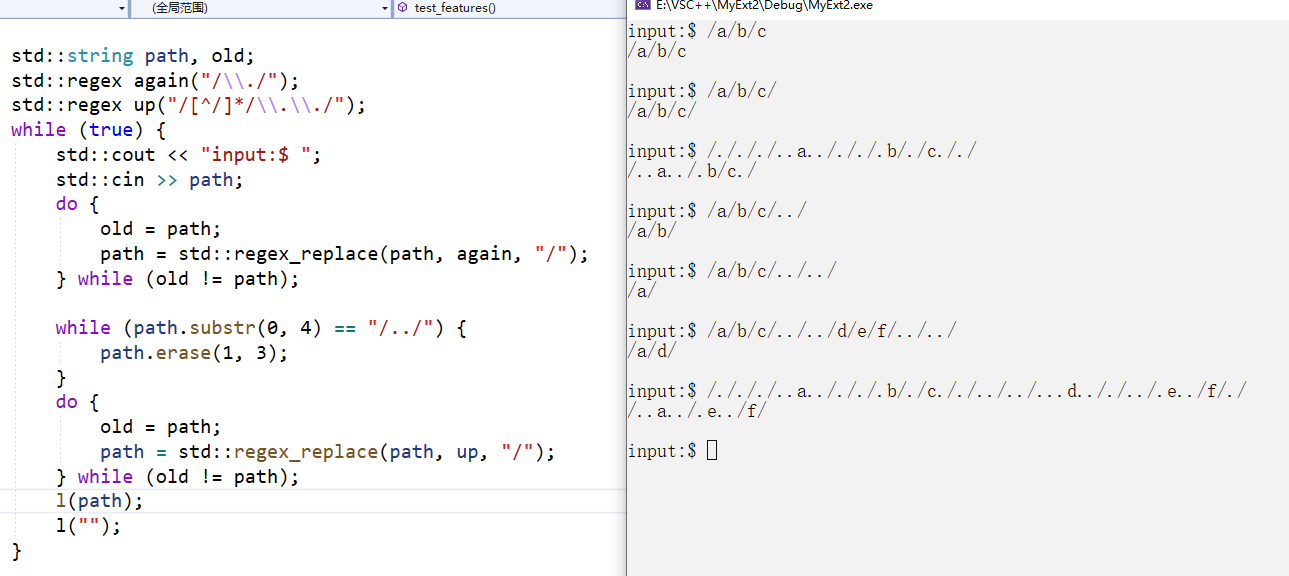
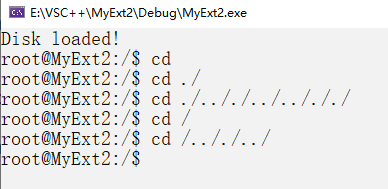


问题:路径显示不正确



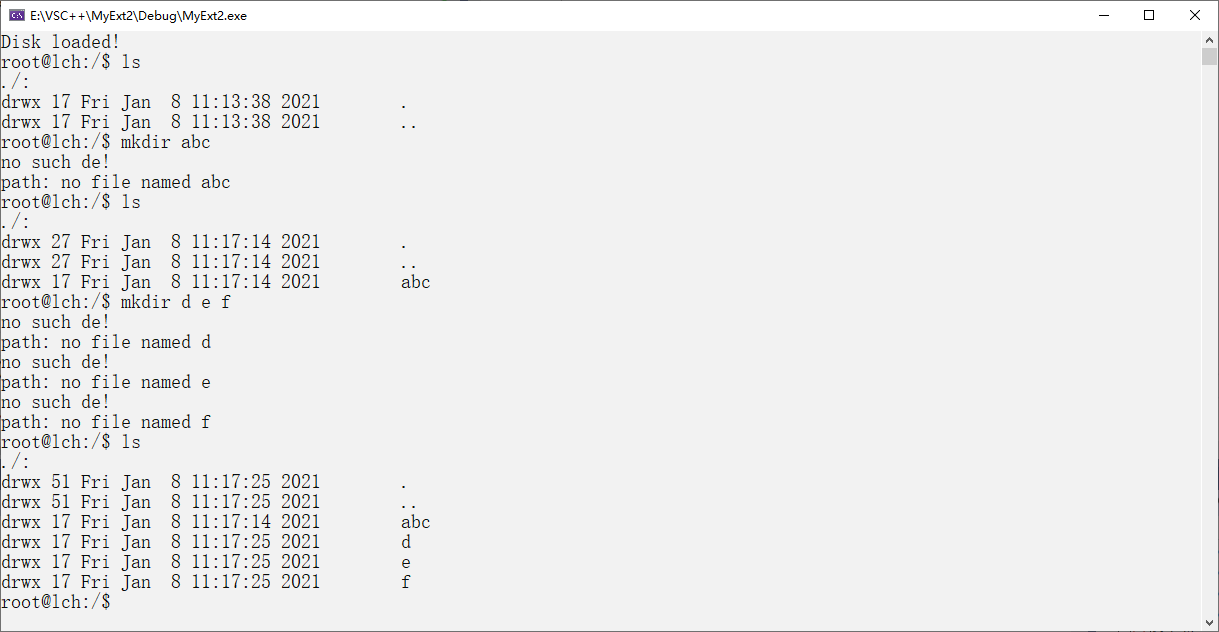
关于路径显示的bug修复: 使用正则表达式

效果: 

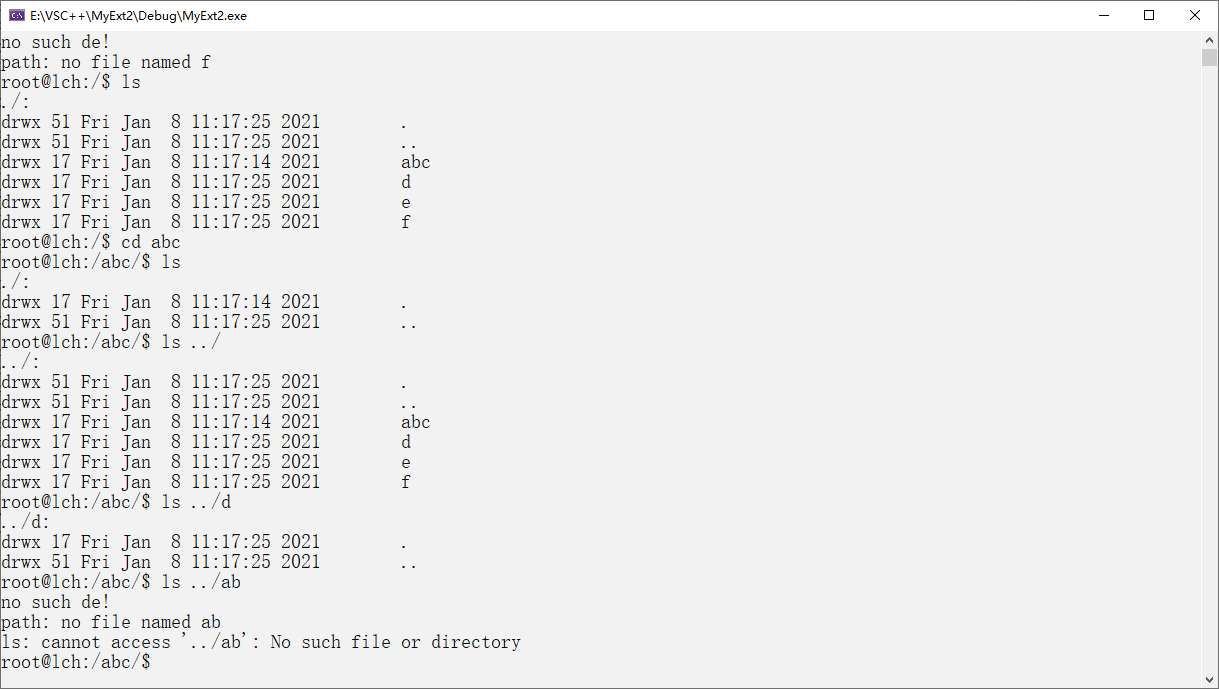
问题: mkdir无效, ls不显示新建的文件夹

解决:路径解析函数 目录项查找函数 均出现问题,已debug; 还有一个原因是添加新的目录项成功, 但是目录文件的长度没有更改, 导致下次读取时还是只能读取到之前的. ..两个目录项(共17字节, 后来添加了文件长度修改的代码后, 能够正常ls显示出来)

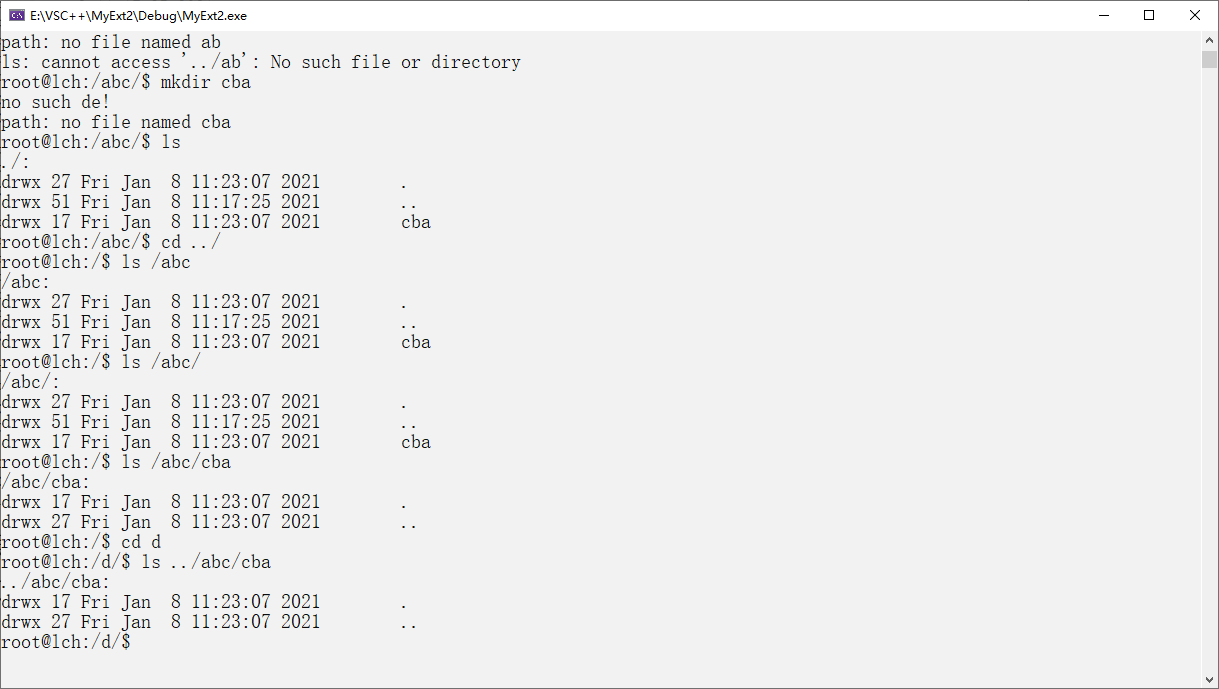
效果:



ls与mkdir准确无误, 其中mkdir支持批量创建(在使用mkdir新建文件夹时, 需要检测是否已存在此文件夹, 所以复用了路径解析函数, 在目前debug的条件下, 它会在找不到文件时输出两句警示, 这是正常现象, 可以无视)

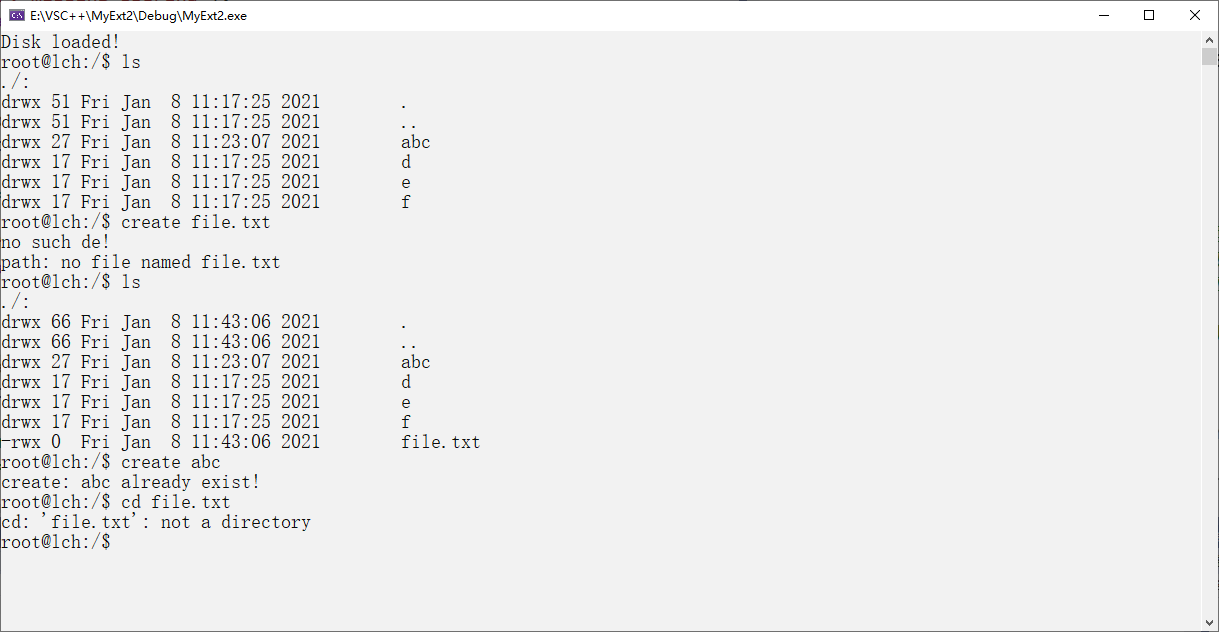


cd无误, ls在非根目录下同样无误; ls允许输入路径, 列出其它目录下的内容, 这有赖于一个统一的路径处理函数; 通过ls一个不存在的路径, 可以看到路径处理函数也具有错误检测能力.



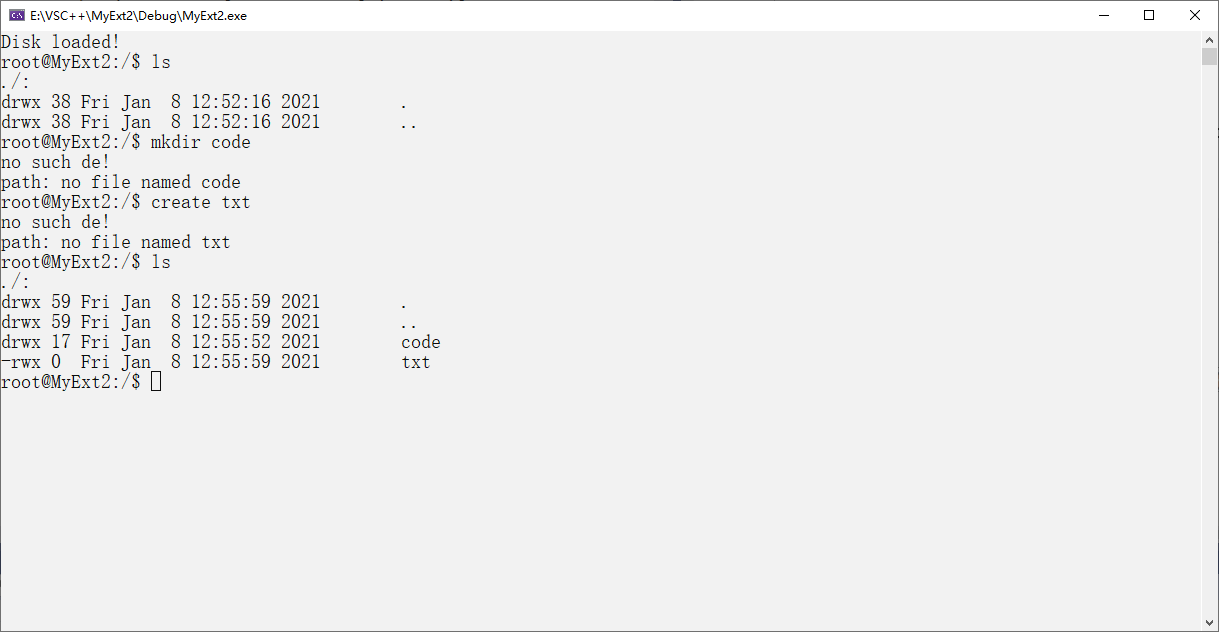
在上图前,根目录/ 下有4个创建的文件夹:/abc /d /e /f . 现在在/abc下创建新文件夹/abc cba, 由ls命令可知创建成功. 此时用cd 分别回退到根目录下以及/d目录中, 通过相对路径和绝对路径来ls二级目录/abc/cba, 结果均成功, 证明此文件系统是树形文件系统, 且具备相应的处理能力.

create展示:

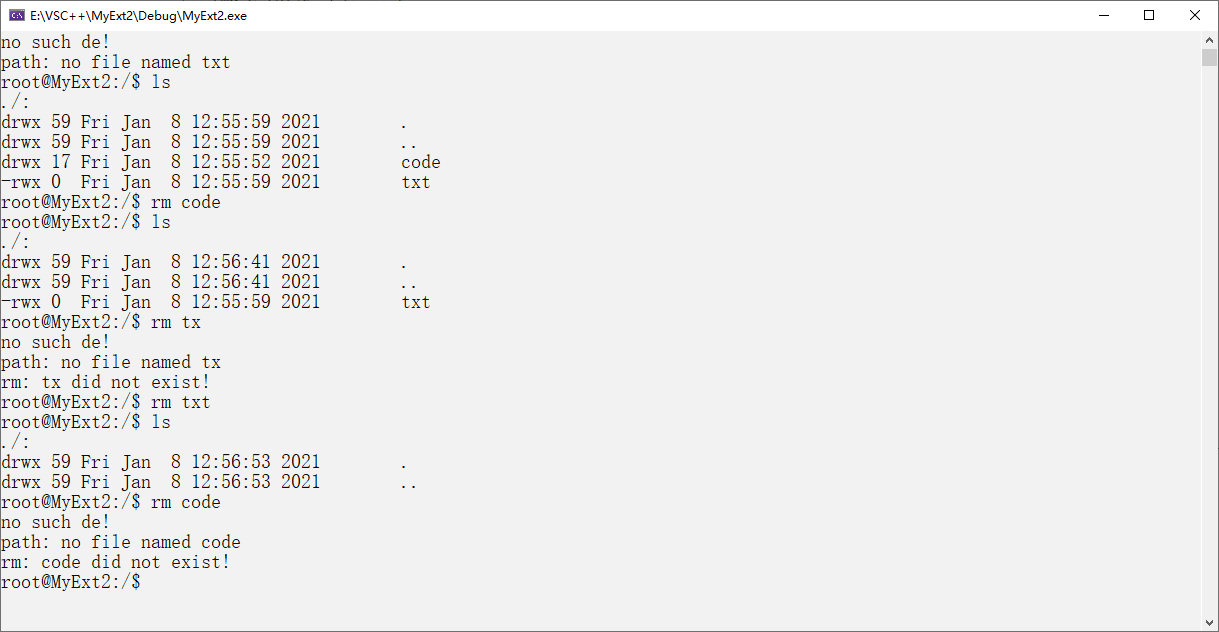


新建文件file.txt, 前后的ls可证明成功, 且新建时不会新建同名文件, 且新建的是文件, cd能对跳转到非目录文件有正确的反馈(跳转失败且正确报错)

rm展示:



新建一个文件夹/code/ , 以及一个文件txt.

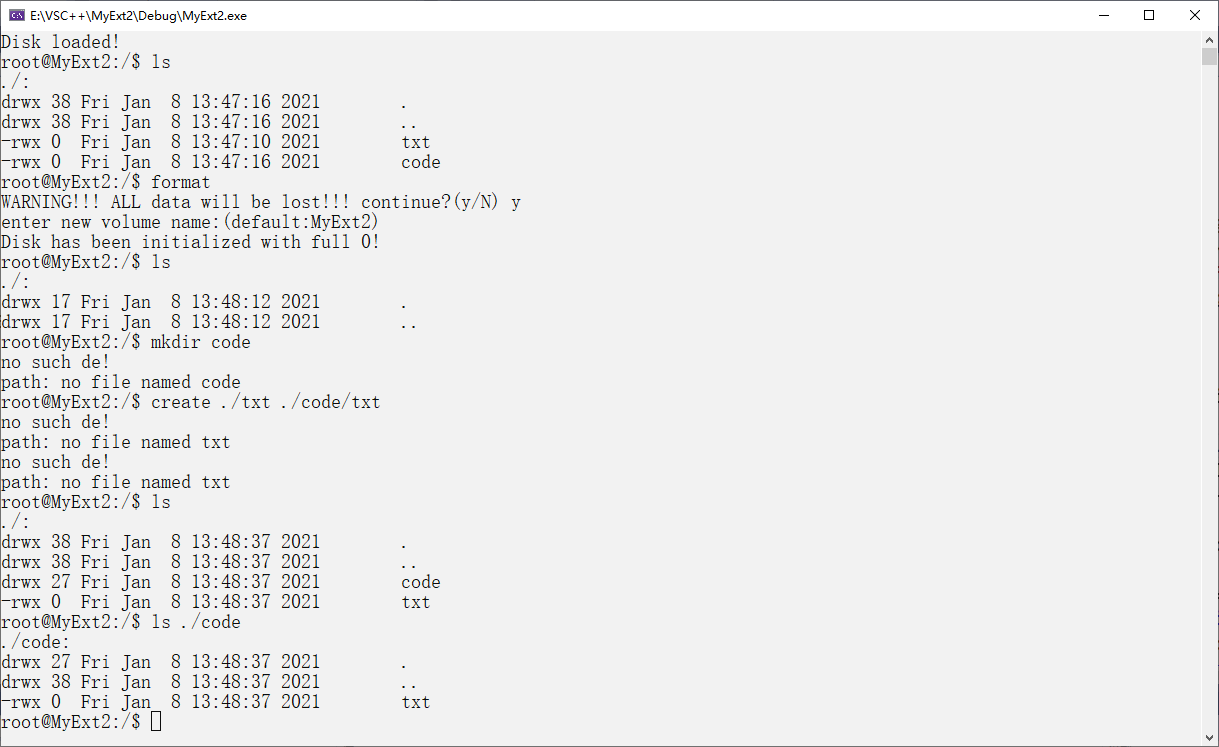


对code/和txt分别进行删除, ls可得删除成功.

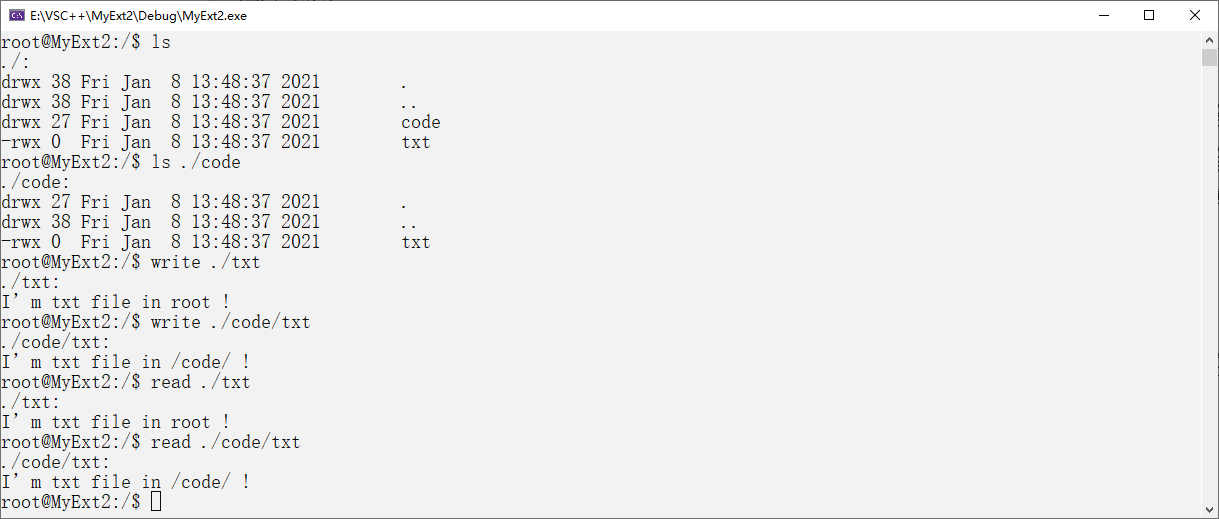
编写测试rm时也遇到很多问题:目录文件的next函数不能正确判断是否到达文件终点, 在死循环里出不来; 删除目录项的函数没有将缓冲区的脏标志dirty置为true, 导致修改完的缓冲区没有被写入磁盘, 进而删除不生效, 等等。以上问题均借助vs的断点、监视等功能完成debug。

read和write展示：

在本次展示中，将在根目录/下新建文件txt，并写入“I’m txt file in root !”，然后再在文件夹/code/下新建文件txt，并写入“I’m txt file in /code/ !”，随后变幻不同位置进行读取，验证效果。



在上图中，由于之前文件系统已经拥有文件，故进行格式化；格式化后新建code以及位于两个不同文件夹下的txt文件。

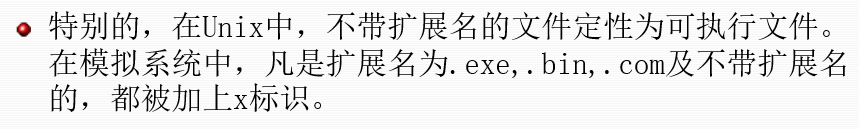


在上图中，按计划向两个文件写入不同字符，并读取。其结果与预期一致。

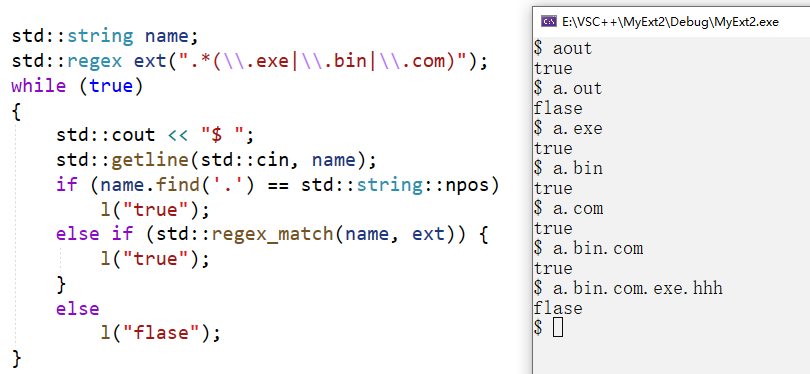
与此同时，它也证明了这是一个树形文件系统：它允许两个不同文件夹下的文件同名，且内容也可不同（是两个独立的文件。）

rwx权限管理:

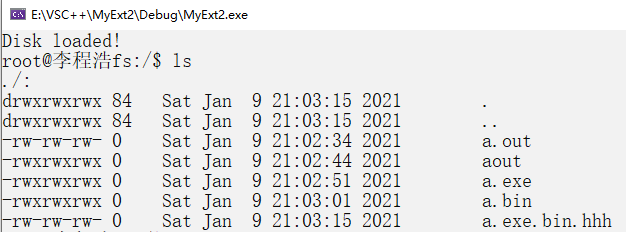
首先, ppt上有这样一条要求:



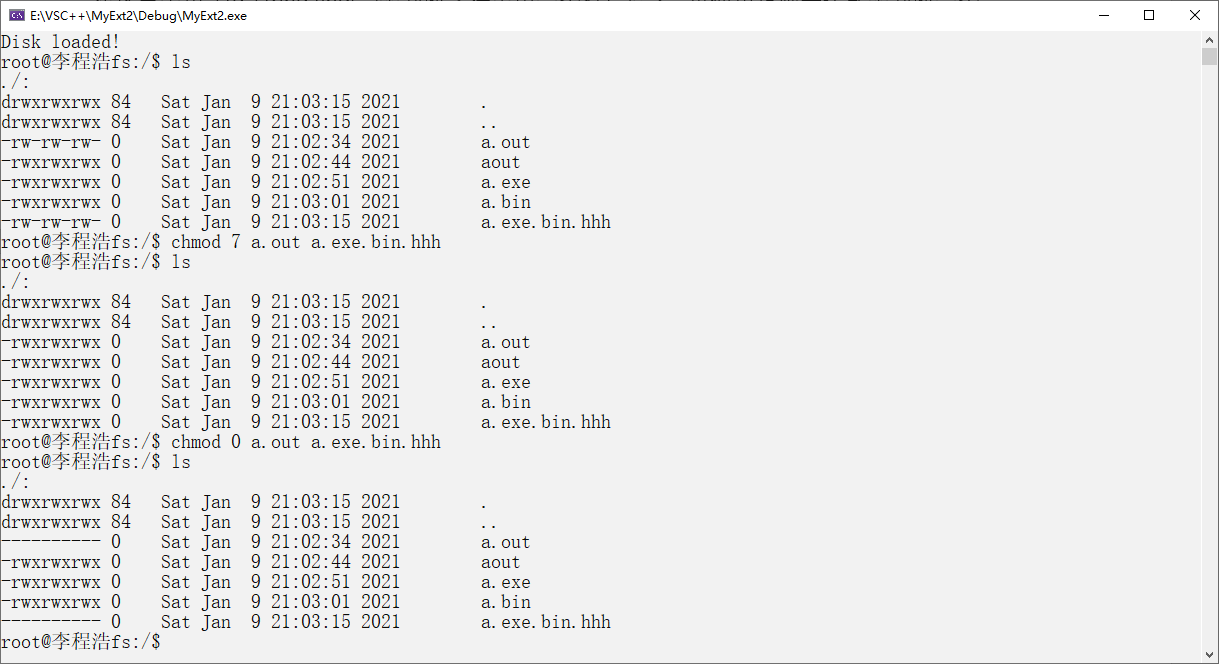
在这里采用std::string.find(‘.’)来判断字符串中是否有后缀名, 再使用正则表达式来判断是否是由以上三种后缀名结尾, 其测试效果如下图:



测试成功, 将其添加进系统内, 此时创建文件时自动生成的文件权限如下图:

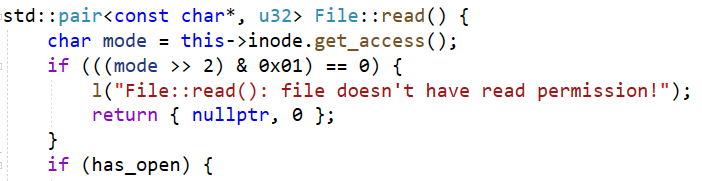


随后加入chmod命令, 它先接受一个数字, 其3位二进制位分别代表将要修改的文件的r(读取)w(写入)x(执行)权限, 如7就代表rwx=111,即此文件可读可写可执行;而6则代表rwx=110,即此文件可读可写, 但不可执行.随后的若干个参数, 是要修改权限的若干个文件的路径, 以此达到批量修改的目的, 方便使用.使用情况如下图:

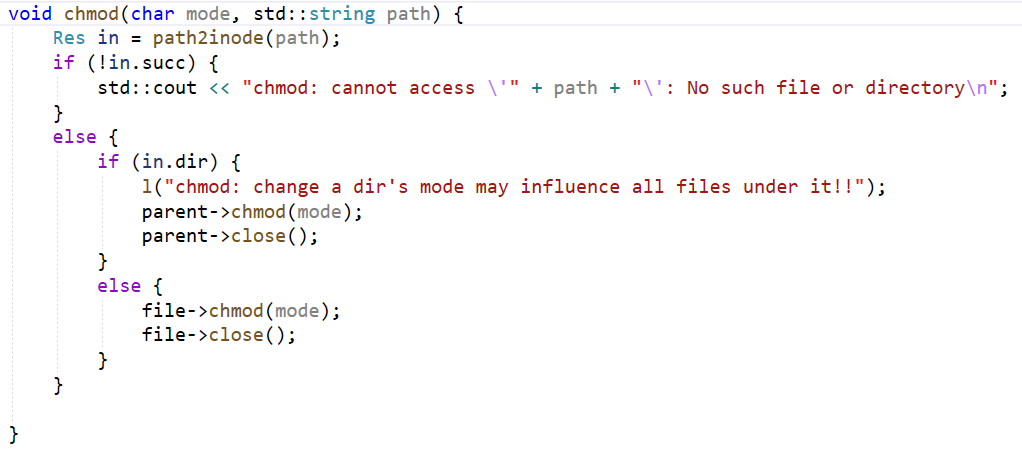


在上图中, 先用chmod 7，将没有执行权限a.out、a.exe.bin.hhh两个文件赋予所有权限，然后用ls证实；再用chmod 0，将其所有权限全部剥夺。测试结果是准确无误的。

之前在进行文件读写时一直没有加入权限对其的控制，现在加入。由于设计架构时采用了多层次的面向对象结构，此时修改起来非常轻松：只需要在File类中的read write等几个方法中加入几行判断语句即可，Dir类由于继承自File类，无需进行任何修改。如下图：



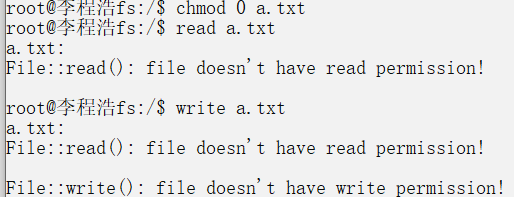
需要注意的是目录文件如果没有写权限，则无法向其添加子文件；如果没有读权限，则不能访问到它之下的全部子文件和子文件夹，因此目录文件的默认权限是rwx=111，并且在修改其权限时也会发出警告，如下图：



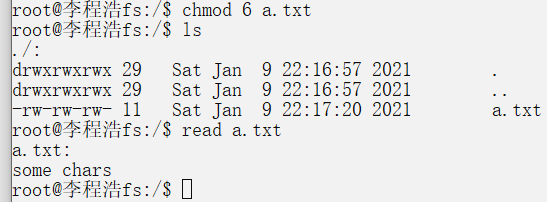
对权限管理进行综合展示：



如上图，创建一个文件a.txt，根据其后缀名.txt判断其不是可执行文件，故创建时默认为rwx=110权限。随后正常读写其内容。



如上图，将其所有权限置0，则read 与write命令都失效。



上图设置其读写权限，则a.txt又可以正常读写。综上，权限管理已完成既定目标。