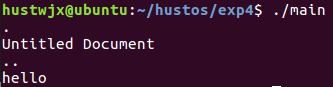
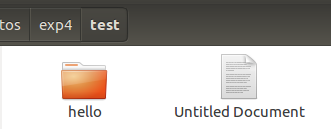
1、使用opendir、chdir、readdir来正确切换路径并读取路径下的文件及目录

完成(main)





另，经过研究发现，目录项既可以是上述，也可以是独自的文件，而且顺序之间没有必然联系

2、加入过滤目录项的流中，“.”项和“..”项的部分，经过调试，在这两项的文件名中，有大量的用处不明字符，它们与这两项中间用值为0的机器码隔开，因此书写新的函数，根据开头是否出现这两项加上空字符来进行过滤

完成(main1)

3、可以通过文件的stat结构区分普通文件与目录文件，并通过递归的方式获取目录文件的更多信息，同时，由于加入了返回父目录和关闭目录项的流的操作（否则会报错自己定义的opendirection fault），因此可以从容应对复杂情景(main2)

也就是现在能读取目录下的所有东西。

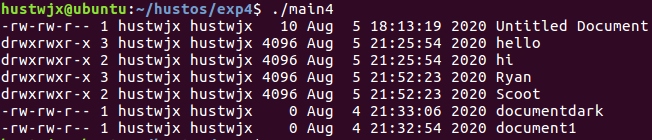
同时，为未来栈的实现准备了目录记忆的代码部分

4、输出文件的详细信息

<https://www.cnblogs.com/xuaijun/p/8036362.html>作为参考

现在能输出详细信息了，唯一缺点在于，不能自动让文件大小的数字自动左对齐（main3）

5、自动让文件大小的数字自动左对齐完成（main4）



最终效果如图

5、使用栈进行管理，像ls -lR一样输出

<https://www.cnblogs.com/operationhome/p/9177151.html>参考资料

等开学后看老师的具体要求

主动完成，现在是lslab.cpp文件，其中使用了c++的vector容器来进行栈管理

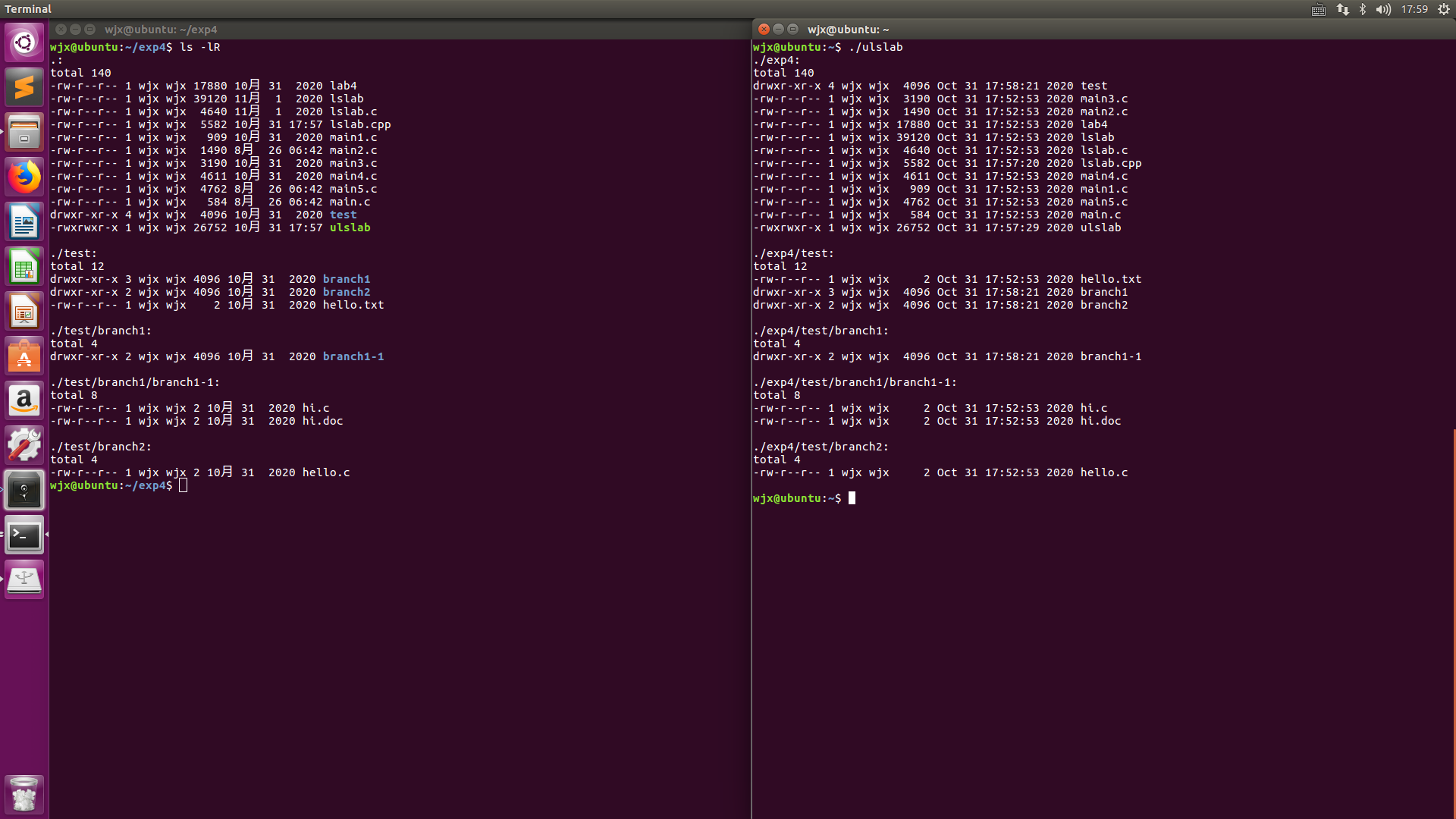
其次，对ls -lR中的total行进行了探究，详见

<https://blog.csdn.net/c1z2w3456789/article/details/79710550>

实际上是目录下所有文件所占的，以页为单位的空间的大小

参考这个思路，在print之前先遍历目录中的文件计算输出total行

但是发现manjaro的文件和内存管理比较高级，始终无法一致，因此在ubuntu下验证了自己的答案



如图可见，基本一致

在manjaro下又经过了一系列测试，发现挂载点需要选在/或者/home之下，自己使用了双系统，在其他磁盘分区中的文件运行时会挂载到/run目录下，该目录经过查阅资料可得知，是一种tmpfs，也就是虚拟文件系统，是使用时挂载到内存中，这也就解释了为什么每次开机后都会对一些双系统磁盘分区外的卷上的文件链接，出现文件不存在的现象，从而在这种文件系统中ls -lR得出来的数据也会和正常情况有不小的差距

因此，将原来的实验文件放到/home挂载点的子目录下进行测试，发现数据完美吻合

最后，通过getcwd()函数获取了当前的文件目录地址，经过适当改造之后，现在的实验四文件除了没有高亮、最后一行有个空行之外，和ls -lR完全吻合

终结：将显示的时间改成修改时间，和ls -lR一致，于是实验目标达成!