一、实验思考

Think1.1

(1) ls –l: 列出文件的详细信息。

-rwxr-xr-x 1 14061039 14061039 844 Mar 27 11:59 Makefile

第1字段: 文件属性字段，共十个字符组成

即-rwxr-xr-x，字符’-’表示该文件是一个普通文件；

字母“d”表示该文件是一个目录，字母"d"，是dirtectory(目录)的缩写；

字母“l”表示该文件是一个链接文件。字母"l"是link(链接)的缩写，类似于windows下的快捷方式；

第1字符的后面9个字符表示该文件或目录的权限位。

r表是读 (Read) 、w表示写 (Write) 、x表示执行 (eXecute)

其中前四个表示文件拥有者的权限，中间两个个表示文件所属组拥有的权限，最后一个表示其他用户拥有的权限。

第2字段：文件硬链接数

如果是一个目录，则第2字段表示该目录所含子目录的个数。

第3字段：文件（目录）拥有者

第4字段：文件（目录）拥有者所在的组

第5字段: 文件所占用的空间(以字节为单位)

第6字段：文件（目录）最近访问（修改）时间

第7字段：文件名

(2) mv test1.c test2.c

将test1.c里的内容移到test2.c里并覆盖test2.c，并删除test1.c。

(3)cp test1.c test2.c

将test1.c里的内容复制到test2.c里并覆盖test2.c。

(4) cd ..

回到上一级目录。

Think1.2

grep

grep [-acinv] [--color=auto] '搜寻字符串' filename

搜寻字符串可以是正则表达式

选项与参数：

-a ：将 binary 文件以 text 文件的方式搜寻数据

-c ：计算找到 '搜寻字符串' 的次数

-i ：忽略大小写的不同，所以大小写视为相同

-n ：顺便输出行号

-v ：反向选择，亦即显示出没有 '搜寻字符串' 内容的那一行！

--color=auto ：可以将找到的关键词部分加上颜色的显示

查找项目中所有的宏：grep –r ‘#define’ \*

查找指定函数名：grep –r ‘函数名’ \*

grep ‘energywise’ \* #在当前目录搜索带'energywise'行的文件

grep -r ‘energywise’ \* #在当前目录及其子目录下搜索'energywise'行的文件

Think1.3

gcc -Wall：打开gcc的所有警告

它可以使gcc产生尽可能多的警告。

gcc -Werror: 要求gcc将所有的警告当成错误进行处理

GCC给出的警告信息虽然从严格意义上说不能算作错误，但却很可能成为错误的栖身之所，

所以，在编译程序时带上-Werror选项，那么GCC会在所有产生警告的地方停止编译，迫使

程序员对自己的代码进行修改。

Think2.1

(1) 由于小明没有git add到暂存区，所以使用git checkout -- printf.c即可恢复。

(2) 由于小红只是删除，没有commit，所以使用git reset HEAD printf.c，之后再使用git checkout

-- printf.c即可。

(3) 使用git rm -- cached Tucao.txt即可解除跟踪。

Think2.2

(1) 正确。

参考资料：<https://git-scm.com/docs/git-clone>

Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using git branch -r), and creates and checks out an initial branch that is forked from the cloned repository’s currently active branch.

(2) 正确。

git log

参考资料：<https://git-scm.com/docs/git-commit>

<https://git-scm.com/docs/git-log>

Stores the current contents of the index in a new commit along with a log message from the user describing the changes.

Shows the commit logs.

cd 14061039-lab/.git/logs/refs/heads我们可以看到每个分支的提交信息，可以看出提交信息是保存在本地的。

git status

参考资料：<https://git-scm.com/docs/git-status>

Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by Git.

可以看出其只访问本地工作区和暂存区。

git checkout

参考资料：<https://git-scm.com/docs/git-checkout>

Updates files in the working tree to match the version in the index or the specified tree. If no paths are given, git checkout will also update HEAD to set the specified branch as the current branch.

由此可以看出，此命令只是更新本地工作区文件，或者重置HEAD指针。

git commit

参考资料：<https://git-scm.com/docs/git-commit>

由git三棵树可知，git commit命令只是将暂存区文件更新到版本库中，版本库也是本地的。

(3) 不正确。

由(1)可知，clone是克隆所有分支。

(4) 正确。

参考资料：<https://git-scm.com/book/zh/ch3-5.html>

Git 的 clone 命令会为你自动将其命名为 origin，拉取它的所有数据，创建一个指向它的 master 分支的指针，并且在本地将其命名为 origin/master。 Git 也会给你一个与 origin 的 master 分支在指向同一个地方的本地 master 分支，这样你就有工作的基础。

二、难点及流程图

填写start.S文件

设置栈指针

跳转到main函数

填写scse0\_3.lds文件，使内核加载到正确位置

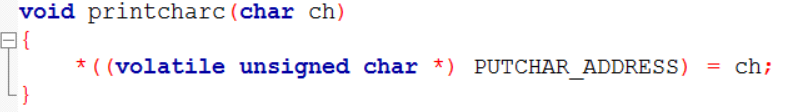
修改include.mk生成vmlinux内核

熟悉git命令

能够使用git操作提交作业

补全print.c中的lp\_Print()函数，实现字符的输出

难点：



这个在console.c文件中，不明白这个函数中唯一一句话的意义。

在补全lp\_Print()函数中，遇到很大问题。

在学习git命令时也比较难以掌握。

三、心得体会

总的来说实验一在指导书的帮助下还是不算太难的，基本在两周就做完了，除了补全lp\_Print()函数时感到一脸懵逼之外，别的东西感觉还可以，在由之前的完全不了解操作系统，到清楚它是如何启动内核，这其中也经历了不少的曲折。在遇到困难也查了很多资料，也问了问班里的大神才得以解决。在这次实验中，我明白了链接文件的作用，也清楚了如何设置各个段的位置，同时也学会了如何使用make命令，虽然仅仅是肤浅的认识，但对于之前没有接触过的我来说，也是受益匪浅。同时，我从一个对Linux完全不了解的新手转变成一个可以基本运用Linux的基础命令的人，这一点对我来说也是非常重要的。除此之外，在git命令的学习上也令我受益颇多，尤其是git时光机让我十分感兴趣，也感受到了它的神奇。