

《Linux 系统基础》实验报告

第一次实验：

姓名： 陈翔宇

学号： 231220088

2023 级 计算机学院 院/系

邮箱： 231220088@smail.nju.edu.cn

时间：2024 年 07 月 24 日

一、实验内容

描述实验的实现方法，包括实验涉及到的命令、程序和关键代码等
回答实验中提到的问题

1. 在/tmp 下新建一个名为 test 的目录。
2. 用命令 man 查看命令 touch 的使用手册。
3. 用命令 touch 在 test 目录中新建一个名为 test 的文件。
4. 用命令 echo 将以下内容一行一行地写入 test 文件。
`#!/bin/sh curl --head --silent https://www.nju.edu.cn`
5. 尝试执行这个文件，即将该脚本的路径 (./test) 输入到您的 shell 中并回车。如果程序无法执行，请使用 ls 命令来获取信息并给出其不能执行的原因。
6. 查看命令 chmod 的手册，使用命令 chmod 改变 test 文件的权限，使 ./test 能够成功执行，不要使用 sh test 来执行该程序。
7. 请问你的 shell 是如何知道这个文件需要使用 sh 来解析的。请通过网络搜索 “unix shebang”来了解更多信息。
8. 请使用 | 和 >，将 test 文件输出的最后 5 行内容写入自己主目录下的 last-5-lines.txt 文件中。

二、实验结果

请说明本次实验实现了哪些功能或得到了什么执行结果，并给出主要功能实现或执行结果的截图。

1. 在/tmp下新建一个名为 test 的目录。

```
$ ls -l | grep test
drwxr-xr-x  - rongzi 24 7月 15:25 test
```

2. 用命令 man 查看命令 touch 的使用手册。



The screenshot shows the man page for the 'touch' command. The title is 'touch - 改变文件时间戳'. The description states: 'touch [选项]... 文件列表 ... 将每个文件的访问时间和修改时间更新为当前时间。除非提供了 -c 或 -h 参数，否则程序会在文件不存在时创建一个空文件。' The options listed are: -a (仅更改文件访问时间), -c, --no-create (不要创建任何文件), -d, --date=字符串 (使用指定字符串所表示的时间而非当前时间), -f (被忽略的选项), -h, --no-dereference (影响所有符号链接而非被引用的文件), -m (仅更改文件修改时间), -r, --reference=文件 (使用指定文件的时间代替当前时间), -t STAMP (使用 [[CC]YY]MMDDhhmm[.ss] 代替当前时间), and --time=关键字 (改变文件的指定时间: 若关键字是 access, atime 或 use, 则等同于 -a; 若关键字是 modify 或 mt).

3. 用命令 `touch` 在 `test` 目录中新建一个名为 `test` 的文件。

```
[rongzi test]# touch test && tree
.
└── test

1 directory, 1 file
```

4. 用命令 `echo` 将以下内容一行一行地写入 `test` 文件。

```
#!/bin/sh
```

```
curl --head --silent https://www.nju.edu.cn
```

```
[rongzi test]# echo '#!/bin/sh' >> test
[rongzi test]# echo 'curl --head --silent https://www.nju.edu.cn' >> test
[rongzi test]# cat test
#!/bin/sh
curl --head --silent https://www.nju.edu.cn
[rongzi test]#
```

5. 尝试执行这个文件，即将该脚本的路径 (`./test`) 输入到您的 `shell` 中并回车。

如果程序无法执行，请使用 `ls` 命令来获取信息并给出其不能执行的原因。

```
[rongzi test]# ./test
bash: ./test: 权限不够
[rongzi test]#
```

6. 查看命令 `chmod` 的手册，使用命令 `chmod` 改变 `test` 文件的权限，使 `./test` 能够成功执行，不要使用 `sh test` 来执行该程序。

```
chmod
授予所有者 [u] 执行 [x] 文件的权限：
chmod u+x 文件

授予所有者 [u] 读 [r] 和写 [w] 文件或目录的权限：
chmod u+rw 文件或目录

移除用户组 [g] 的文件执行 [x] 权限：
chmod g-x 文件

授予所有用户 [a] 读 [r] 以及执行 [x] 文件的权限：
chmod a+rx 文件

授予其他用户 [o] （不在所有者用户组）和用户组 [g] 同样的权限：
chmod o=g 文件

移除其他用户 [o] 的所有权限：
chmod o= 文件

递归授予用户组 [g] 和其他用户 [o] 目录下所有文件和子目录的写 [w] 权限：
chmod -R g+w,o+w 目录

递归授予所有用户 [a] 目录下文件的读 [r] 权限和子目录的执行 [X] 权限：
chmod -R a+rX 目录
```

```
[rongzi test]# chmod u+x ./test
[rongzi test]# ./test
HTTP/1.1 200 Connection established

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2024 08:07:45 GMT
Content-Type: text/html
Content-Length: 279998
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Referer-Policy: no-referer-when-downgrade
X-Download-Options: noopen
X-Permitted-Cross-Domain-Policies: master-only
Last-Modified: Wed, 24 Jul 2024 07:25:50 GMT
Accept-Ranges: bytes
Vary: User-Agent,Accept-Encoding
Cache-Control: private, max-age=600
Expires: Wed, 24 Jul 2024 08:18:26 GMT
ETag: "445be-61df9300de880-gzip"
Content-Language: zh-CN

[rongzi test]#
```

7. 请问你的 shell 是如何知道这个文件需要使用 sh 来解析的。请通过网络搜索

“unix shebang”来了解更多信息。

shell 的命令执行步骤

1. 命令扩展(本次 lab 不涉及)

2. 命令执行

+ 如果命令 a 是一个 shell 函数，执行函数并退出，否则下一步

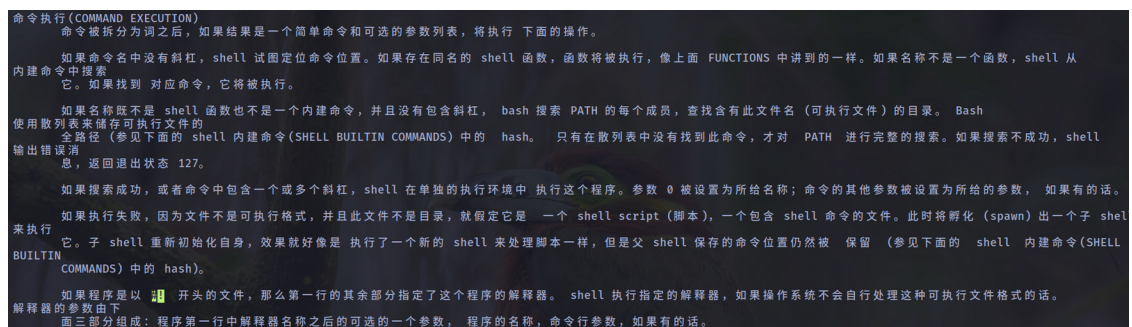
+ 如果命令是一个 shell 内置命令，执行命令并推出，否则下一步

+ 搜索 PATH 中的目录如果有对应的程序，运行程序，否则下一步

+ 如果操作系统不能处理这种可执行文件的格式，并且程序是一个#!开头的非目录文件，那么第一行的其余部分指定了这个程序的解释器。

解释器的参数由下面三部分组成：程序第一行中解释器名称之后的可选的一个参数， 程序的名称，命令行参数，如果有的话。

bash 的 man 手册中关于#!的部分



命令执行 (COMMAND EXECUTION)
命令被拆分为词之后，如果结果是一个简单命令和可选的参数列表，将执行 下面的操作。

如果命令中没有斜杠，shell 试图定位命令位置。如果存在同名的 shell 函数，函数将被执行，像上面 FUNCTIONS 中讲到的一样。如果名称不是一个函数，shell 从内建命令中搜索它。如果找到 对应命令，它将被执行。

如果名称既不是 shell 函数也不是一个内建命令，并且没有包含斜杠，bash 搜索 PATH 的每个成员，查找含有此文件名（可执行文件）的目录。Bash 使用散列表来缓存可执行文件的全路径（参见下面的 shell 内建命令 (SHELL BUILTIN COMMANDS) 中的 hash。只有在散列表中没有找到此命令，才对 PATH 进行完整的搜索。如果搜索不成功，shell 输出错误消息，返回退出状态 127。

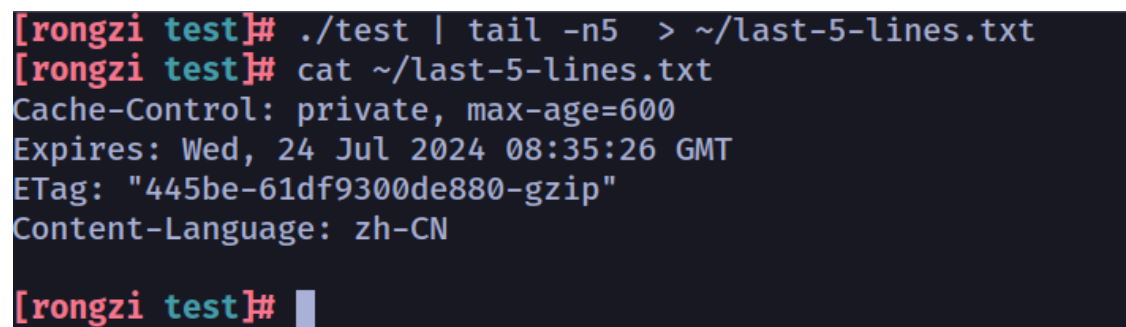
如果搜索成功，或者命令中包含一个或多个斜杠，shell 在单独的执行环境中 执行这个程序。参数 0 被设置为所给名称；命令的其他参数被设置为所给的参数， 如果有的话。

如果执行失败，因为文件不是可执行格式，并且此文件不是目录，就假定它是 一个 shell script (脚本)，一个包含 shell 命令的文件。此时将孵化 (spawn) 出一个子 shell 来执行它。子 shell 重新初始化自身，效果就好像是 执行了一个新的 shell 来处理脚本一样，但是父 shell 保存的命令位置仍然被 保留（参见下面的 shell 内建命令 (SHELL BUILTIN COMMANDS) 中的 hash)。

如果程序是以 #! 开头的文件，那么第一行的其余部分指定了这个程序的解释器。shell 执行指定的解释器，如果操作系统不会自行处理这种可执行文件格式的话。

解释器的参数由下面三部分组成：程序第一行中解释器名称之后的可选的一个参数， 程序的名称，命令行参数，如果有的话。

8. 请使用 | 和 >，将 test 文件输出的最后 5 行内容写入自己主目录下的 last-5-lines.txt 文件中。



```
[rongzi test]# ./test | tail -n5 > ~/last-5-lines.txt
[rongzi test]# cat ~/last-5-lines.txt
Cache-Control: private, max-age=600
Expires: Wed, 24 Jul 2024 08:35:26 GMT
ETag: "445be-61df9300de880-gzip"
Content-Language: zh-CN

[rongzi test]#
```

三、实验中遇到的问题及解决方案

没有解决的问题也可以写在这里。

四、实验的启示/意见和建议

在互联网中总是能看到 Unix 哲学这几个字眼，但是网上没有什么准确的定义，我实在好奇到底什么是 Unix 哲学？它和编程本身是什么关系，能帮助编程吗？我们该怎么学习这种哲学？

用时：1.5h

附：本次实验你总共用了多长时间？包括学习时间、编写代码时间和测试时间。

（仅做统计用，时间长短不影响本次实验的成绩。）