Lab0 Linux笔记

Vim

- 这里注意一下 `C - s` 锁屏，`C - q` 退出锁屏

Linux命令

- ```
  echo
  -n 不输出换行符
  -e 解释反斜杠转义字符，如 \n
  ```

- `sed -i 's/$2/$3/g' $1` **在** `modify.sh` **中不会按预期工作**，因为 `sed` **不能直接解析 shell 变量**，需要用双引号 `"..."` 来正确解析变量。

  `-n` 只输出经过sed处理的内容，否则输出文件所有的内容

  `-i` 表示直接修改到文件中，否则只是展现修改之后长什么样

  `'s/str1/str2/'` 时，必须加最后一个 `/` ，之后可以加数字表示第几次出现或者使用 `g` 表示全部，最后加上 `i` 可以实现忽略大小写

  `-e` 多个连续指令依次执行

  `"/PATTERN/ a\\$var"` 可以往符合要求的位置放变量的值

  `&` 代表 `str1`，可以用作突出作用

  > 小技巧：对于sed替换单词，可以在 str1 和 str2 两侧加上空格
  >
  > ```
  > sed 's/ the / this /'
  > ```

  ```
  # 替换
  sed -i "s/$2/$3/g" "$1"
  # 删除 my.txt 文件的第二行到最后一行（$ 符号表示到最末尾）
  sed '2,$d' my.txt
  # 第一条是第四行后添加一个 str
  sed -e '4 a\str ' -e 's/str/aaa/' my.txt
  ```

- Linux awk命令

  用于对文本内容进行分析

  如 `39: int a;` 提取行数输出为 `39`，其中 `-F` 支持正则表达式，故可以实现按单词分割
  ```

多个输出之间用 `,` 连接，相当于Python，自动加入空格

```
grep "int a=1" file.txt | awk -F: '{print $1}' > output.txt
```

```
NR - 当前行标号
NF - 当前行分割的总列数
```

分支判断示例：

```
# example 1
awk '{
if($2 >=50 && $3>=50&&$4>=50)
    print $1,": Pass"
else
    print $1,": Fail"
}'
# example 2
awk '{
average = ($2 + $3 + $4)/3;
grade = "PASS";
if (average >= 80)
    grade = "A";
else if (average >= 60 && average < 80)
    grade = "B";
else if (average < 50)
    grade = "FAIL";

print $0, ":", grade;
}'
```

- `grep` 常用配套

  `grep -n PATTERN FILE` 显示行号

  `-i` 对大小写不敏感（不区分）

  `-w` 按整个单词匹配

  `-v` 找到没有出现 `PATTERN` 的行

  `-r` 在目录中递归查找

  `-n` 显示行号

  `-c` 仅统计行号

  加入 `FILE` 字段后可以将其输出到文件路径中

- `find` 命令 `find dir/ -name lab0_x > lab0_exam.c`

如果没有目录，默认为当前目录

默认的文件名是全匹配，可以用 `*` 来任意匹配

- `basename` 指令可以获得输入路径的最后一部分

  `basename $(pwd)` 获得当前目录名

- `mkdir -p` 递归创建目录

  也可用于创建目录存在时不想要报错时使用

- `mv` 可以用于移动目录（直接移动目录内所有的内容）
- `cp` 配套 `cp -r SOURCE TARGET` 可以递归复制目录及其子目录内容
- `rm` 配套 `rm -r` 递归删除目录及其内容

  另外 `rm -f` 强制删除

- `cut` 命令，对于每一行进行数据处理

  相当于 `split()` 函数，`cut -c 3` 按字符分割，`-d '.' -f 2` 为按指定分隔符分割；`-f` 指定输出的列

  输出一定位置或范围的字符 `cut -c 3,5-7`

  `cut -f 1-3` `cut -f 13-` 默认按csv文件格式进行每行的分割

```
echo '0000 192.168.1.100 192.168.100.1' |cut -d ' ' -f 2
192.168.1.100
echo '0000 192.168.1.100 192.168.100.1' |cut -d ' ' -f 2 |cut
-d '.' -f 4
100
echo '0000 192.168.1.100 192.168.100.1' |cut -d ' ' -f 2 |cut
-d '.' -f 4|cut -c 1
1
```

CMake

- `make -C` 用于 **切换到指定目录** 并在该目录执行 `make` 命令

```
make -C <目录> <目标>
```

  e.g. `make -C code clean`

- 默认 gcc 的 include 路径为默认路径

  如果需要在当前目录下进行查找，则需要加上 `-I[INCLUDE PATH]` 参数

```
gcc -Iinclude code/fibo.c main.c -o fibo
```

- Makefile中如果使用变量，获取值使用 `()`

```
INCP=-I../include
all: fibo.c main.c
    gcc $(INCP) -c fibo.c -o fibo.o
    gcc $(INCP) -c main.c -o main.o
```

Shell

- 每个 `.sh` 文件后最好加上一个 `exit 0`，让这个程序的返回值为 `1`（不然可能会让系统以为这个**程序报错**了）
- 注意，shell中返回值为 `0` 表示真，`1` 表示假
- 在有双引号的情况下，`$` 会起到引用的作用，所以一定要想清楚是否需要空格隔开，如`sed -i '$ a\$i'`
- 由于shell自身功能的而特点，**赋值语句**不加入空格，否则解析为 `<command> <args...>`
- 命令传参

```
$n 第n个参数
$0 命令名
$# 传参个数
$* 一个字符串显示传递的全部参数
$? 获取前一个命令的返回值
```

- 对于字符串条件判断关键字 `-s, -z, -n`

```
-s [file] # 文件存在且文件不为空，真；反之，假
-z # 字符串长度为 0 ，真；反之，假
-n # 与 -z 相反
```

- 获取字符串长度 `${#str}$`
- 截取字符串的方法（类似于 `substring()`）：`${s:a:b}`，字符串 `s` 从第 `a` 位开始的 `b` 个（从 0 开始）
- **括号用法**
    1. `$(...)`：隔离并执行命令，将输出替换至原处
    2. `(())`：用于算术表达式（亦可用于赋值）

```
a=1
b=2
c=$((a + b))
((a = 10))
((b++))
```

3. `[] or [[]]`：判断条件，`[[]]` 还可以使用逻辑运算

   由于 `[` 其实是一个可执行文件，所以需要加入**空格隔开**

```
# [] or [[]]
[ $a -ne 2 ]
# [[]]
[[ $a -gt 1 && $a -lt 10 ]]
# 对于多个判断，可以使用如下更加方便
[[ $x -eq $y ]] || [[ $x -eq $z ]] || [[ $y -eq $z ]]
```

- 字符串比较一般不使用 `-eq` 而是直接使用 `==`
- 数组使用

```
arr=(ele1 ele2 ele3) # 定义数组
echo ${#arr[*]} # 数组长度
echo ${arr[0]} # 数组第一个元素
echo ${arr[@]} # 数组所有元素
echo ${arr[*]} # 数组所有元素
```