# Linux Bash & EM

**Shared by Group 4**

展示人：许芸阁（Bash）、贺泽群（EM）

资料提供：贺泽群（Bash introduction & EM）、许芸阁（Bash introduction补充 & Bash homework）

讨论与交流：苏厚祯、范泽玥、张天婧

# 目录

Contents

PART 01

## 背景介绍

Bash introduction

PART 02

## 必做题+干货分享

Bash homework

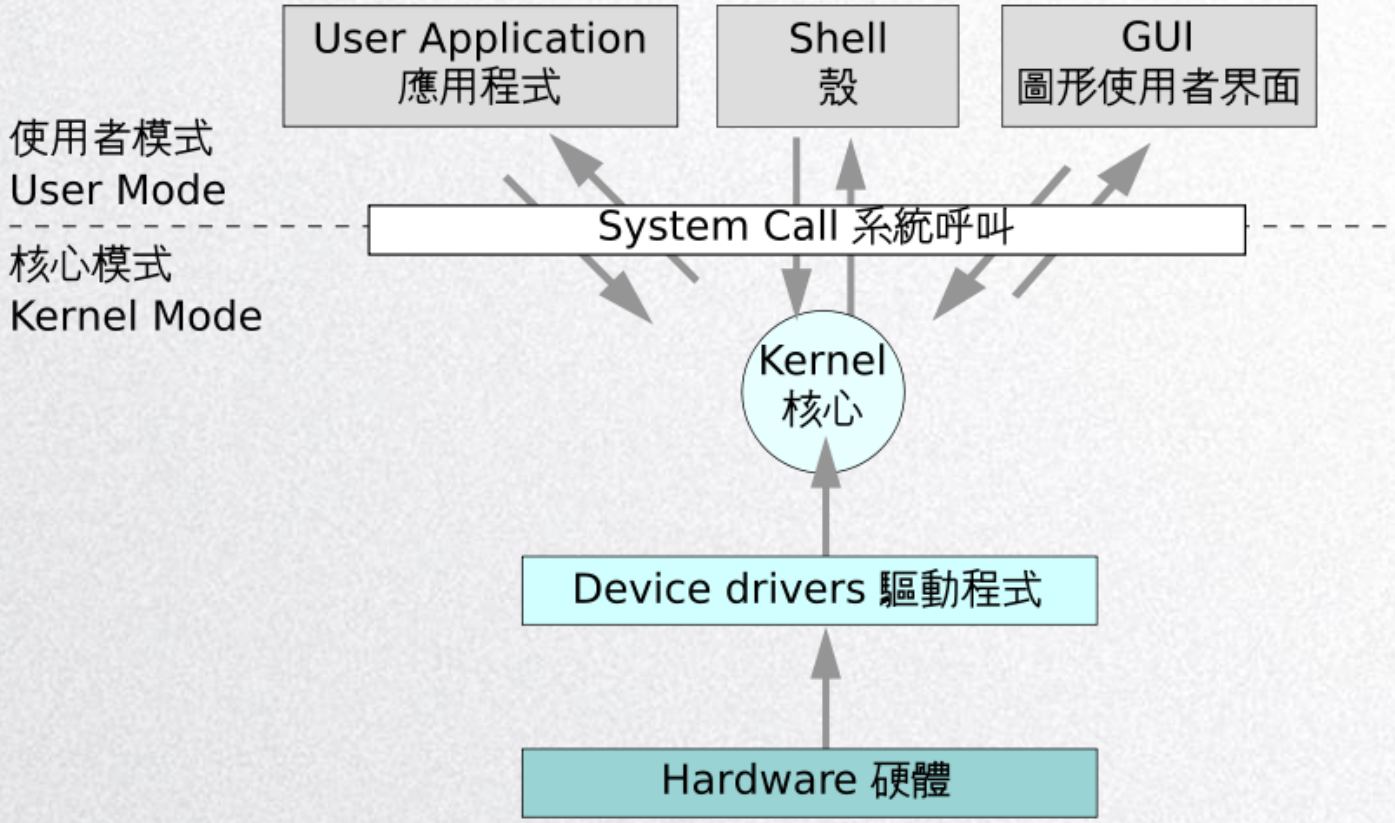
PART 03

## 练习题（EM）

About EM

# Linux

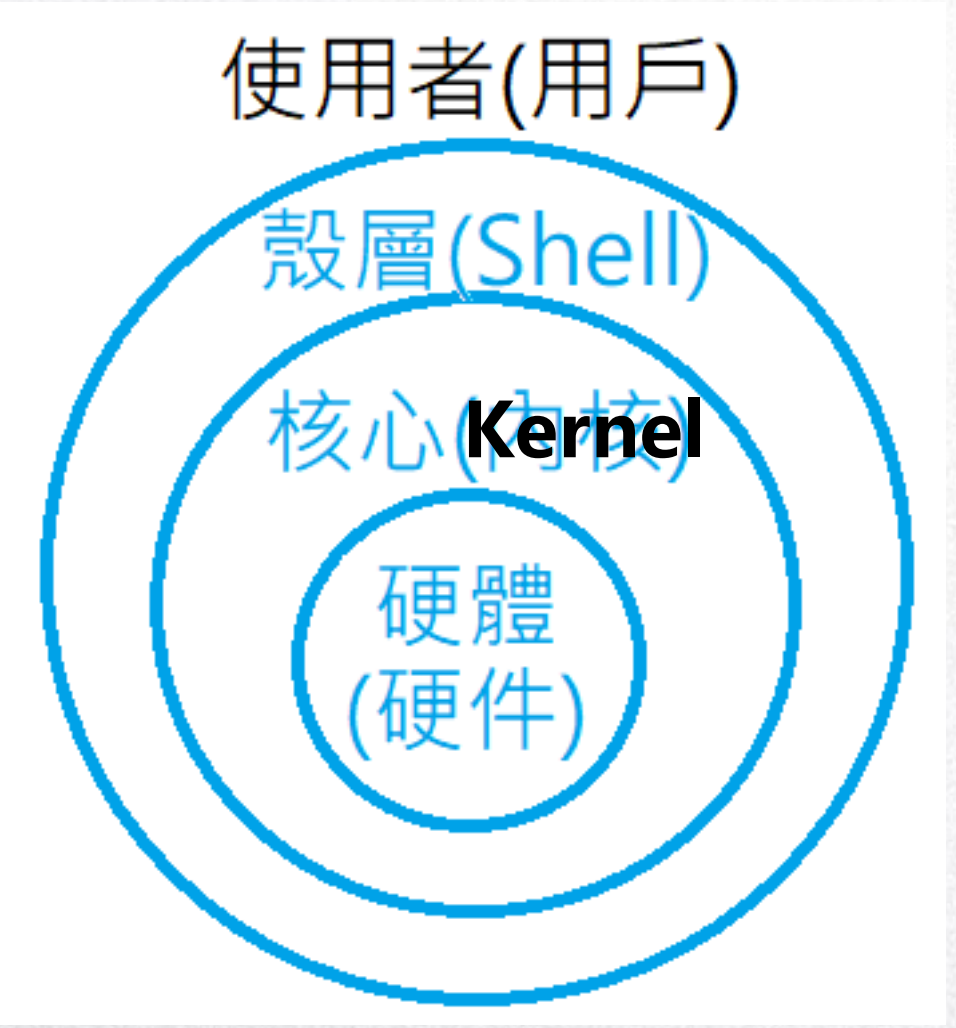
- 与Windows, iOS, and Mac OS一样，Linux是一个**操作系统**。操作系统是一组主管并控制计算机操作、运用和运行硬件、软件资源和提供公共服务来组织用户交互的相互关联的系统软件程序。简单来说， 操作系统是软件与硬件之间交流的桥梁。
- GNU是什么？和Linux是什么关系？



# Shell



- Shell是用户与Unix操作系统交互的界面。用户通过**直接输入命令**来执行各种各样的任务。Shell包围在kernel之外，用户通过shell将输入的指令与kernel沟通，从而使得kernel可以准确地控制硬件来进行工作。



# Bash



- Linux的shell根据发展者的不同，有许多版本，比如Bourne Shell 、C Shell、K Shell等。Bash即Bourne Again SHell，是Bourne Shell（sh）的增强版本，在1987年由布莱恩·福克斯编写。它是GNU操作系统中重要的软件工具之一，目前也是GNU操作系统的基本shell。
- 特性：

1. 命令记忆能力（history 上下键）
2. 命令和文件补全功能（[Tab]键）
3. 命令别名设置功能（alias）
4. 程序脚本（**bash script**）
5. 通配符（wildcard * ? ）

布莱恩·福克斯

# 抓虫小游戏



```
#!/bin/bash

path = "./bash_homework"
name='ls $path'

for val in name
do
  if [ -f $val ];then
    echo "$val" > filename.txt
  elif [ -d $val ];then
    echo "$val" > dirname.txt
  fi
done

exit 0
```

- 写出一个 bash 脚本，可以使它**自动读取一个文件夹**（例如bash_homework/）的内容，
- 将该文件夹下文件的名字输出到 filename.txt,
- 子文件夹的名字输出到 dirname.txt 。

找出图中代码的 **5** 处bug

# 抓虫小游戏



```
#!/bin/bash

path = "./bash_homework"
name='ls $path'

for val in name
do
  if [ -f $val ];then
    echo "$val" > filename.txt
  elif [ -d $val ];then
    echo "$val" > dirname.txt
  fi
done

exit 0
```

- 给变量赋值时，=两边不能有空格
- `命令`或者$(命令)表示先执行该命令，命令输出结果替换变量赋值结果
- 调用变量的值时要在变量名前加$
- 注意文件路径
- 重定向符>表示覆盖，>>表示追加

找出图中代码的 5 处bug

# 抓虫小游戏



```
#!/bin/bash

path = "./bash_homework"
name='ls $path'

for val in name
do
  if [ -f $val ];then
    echo "$val" > filename.txt
  elif [ -d $val ];then
    echo "$val" > dirname.txt
  fi
done

exit 0
```

错误代码

```
#!/bin/bash

path="./bash_homework"
name=`ls $path`

for val in $name
do
  if [ -f "$path/$val" ];then
    echo "$val" >> filename.txt
  elif [ -d "$path/$val" ];then
    echo "$val" >> dirname.txt
  fi
done

exit 0
```

改正后

# 干货分享 – 1. 引号内字符串



- 单引号中不对$等特殊符号做解释执行
- 双引号中会对$等特殊符号做解释执行

```
#!/bin/bash

FRUIT_PRICE="22"
echo "The price of fruit is $FRUIT_PRICE\$"
echo 'The price of fruit is $FRUIT_PRICE\$'

exit 0
```

```
The price of fruit is 22$
The price of fruit is $FRUIT_PRICE\$
```

```
#!/bin/bash

ha="nya"

echo "Ke$ha is a country."
echo 'Ke$ha is a singer.'

exit 0
```

```
Kenya is a country.
Ke$ha is a singer.
```

# 干货分享 – 2. 分号



- 单行语句一般用分号区分代码块
- 若该代码写作多行，则用换行符来区分代码块，无需分号


```
#!/bin/bash

# 设置起始值为0
val=0

while true
do
if [ "$val" -eq "5" ];then
# 如果val=5, 则跳出循环
break;
else
# 输出数值
echo "val=$val"
# 将数值加1
((val++))
fi
done

exit 0
```

```
if 条件1
then 命令1
elif 条件2
then 命令2
else 命令3
fi
```

# 干货分享 – 3. ${变量名}



- 避免变量名被执行时出现歧义

```
#!/bin/bash

echo "Input a word:"
read WORD  # bash内置命令read: 从标准输入读入一行，赋值给变量
echo "My word is relax"
echo "I will merge your word with mine -- $WORD_relax"
echo "I will merge your word with mine -- ${WORD}_relax"

exit 0
```

```
Input a word:
happy
My word is relax
I will merge your word with mine --
I will merge your word with mine -- happy_relax
```

```
#!/bin/bash

array=(23 61 127 98 45)

echo $array[1]
echo ${array[1]}

echo ${array[@]}
echo ${#array[@]}
```

```
23[1]
61
23 61 127 98 45
5
```

# 干货分享 – 4. [ ]条件判断语句

Part 2 必 做 题

```
[ Expression ]
[[ Expression ]]

# 判断文件/Users/xyg/abc.txt是否存在
[ -f /Users/xyg/abc.txt ]

# 判断变量val是否等于字符串"helloworld"（字符串比较表达式）
[ "$val" = "helloworld" ]
[[ "$val" == "helloworld" ]]

# 判断20是否小于100(数值比较运算表达式)
[ 20 -lt 100 ]
[ 20 \< 100 ]
[[ 20 -lt 100 ]]
[[ 20 < 100 ]]

# 判断文件abc.txt是否可读和可写(逻辑运算符)
[ -r abc.txt -a -w abc.txt ]
[ -r abc.txt ] && [ -w abc.txt ]
[[ -r abc.txt && -w abc.txt ]]
```

| | [ ] | [[ ]] |
|---|---|---|
| 字符串比较 | = | == |
| | != | != |
| | … | … |
| 数值比较 | -lt或\< | -lt或< |
| | -gt或\> | -gt或> |
| | -le | -le |
| | -ge | -ge |
| | -eq或= | -eq或== |
| | -ne或!= | -ne或!= |
| 逻辑运算符 | 逻辑与 -a | 逻辑与 && |
| | 逻辑或 -o | 逻辑或 \|\| |
| | 逻辑非 ! | 逻辑非 ! |

# 干货分享 – 5. 数值运算



- $((算术运算表达式))
- 只支持整数运算+ - * / ** %
- 表达式中，变量不加$

```
#!/bin/bash

val1=$((3*(5+2)))
val2=$((val1**2/5%7))

echo "val1=$val1"
echo "val2=$val2"
```

```
val1=21
val2=4
```

- 使用命令行交互式工具bc，可以支持浮点运算
- -l表示标准数学库

```
#!/bin/bash

val3=`echo "scale=3; 1/3" | bc -l`
val4=`echo "scale=7; a(4)" | bc -l`

echo "val3=$val3"
echo "val4=$val4"

exit 0
```

```
val3=.333
val4=1.3258176
```

# 干货分享 – 6. Tips on Vim



- 按“i”进入编辑模式，按esc退出编辑模式、进入命令模式
- 在命令模式中：
  1. HJKL分别代表左下上右
  2. 按u表示回退
  3. 在某行连按两次d表示删除该行
  4. 按住2同时按两次d表示删除该行和后一行
  5. :1,$ d表示清空
  6. ……

# About EM

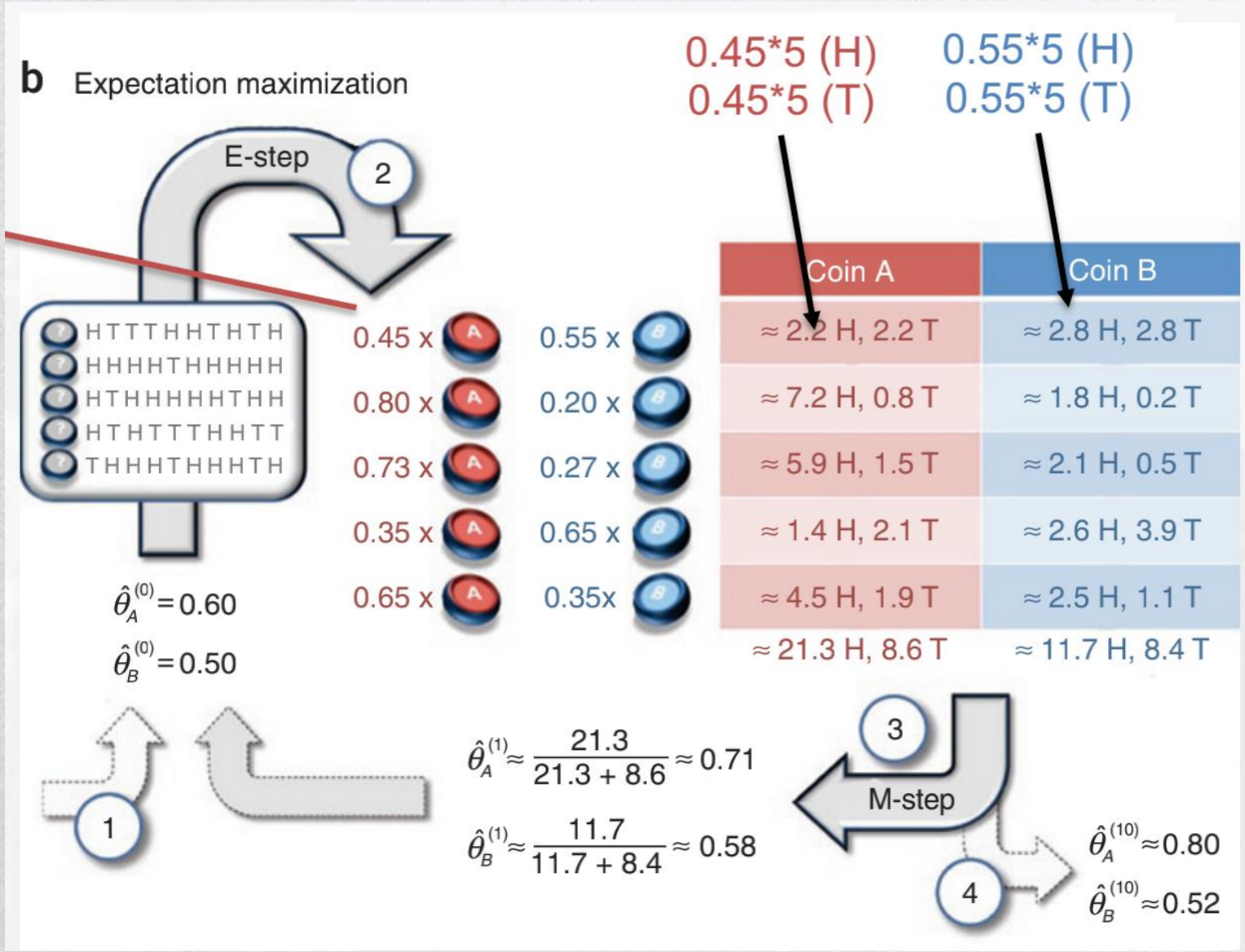
- 最大期望算法（Expectation-maximization algorithm，又译期望最大化算法）在统计中被用于寻找，**依赖于不可观察的隐性变量的概率模型**中，参数的**最大似然估计**。

- 最大期望算法经常用在机器学习和计算机视觉的数据聚类（Data Clustering）领域。

- 最大期望算法经过两个步骤交替进行计算，**第一步是计算期望（E）**，利用对隐藏变量的现有估计值，计算其最大似然估计值；**第二步是最大化（M）**，最大化在E步上求得的最大似然值来计算参数的值。M步上找到的参数估计值被用于下一个E步计算中，这个过程不断交替进行。

# 算法实现思路






- 编写函数EM，并循环运行，结束循环的条件是两次结果差的绝对值小于$10^{-3}$

- EM函数：
  - 参数：P_a_i , P_b_i
  - 循环：
  - 对于每一条序列分别计算A、B概率并计算H、T数量，存入总数
  - 计算：
    分别计算P_a_i+1 , P_b_i+1

# 脚本参考（Python）



```python
#!/usr/bin/env python

results = [
    "HTTTHHTHTH",
    "HHHHTHHHHH",
    "HTHHHHHTHH",
    "HTHTTTHHTT",
    "THHHTHHHTH"
]

def EM(hA, hB, results):
    nhA, nhB = 0, 0
    ntA, ntB = 0, 0

    for res in results:
        pA, pB = 1, 1
        for s in res:
            if s == 'H':
                pA, pB = pA * hA, pB * hB
            else:
                pA, pB = pA * (1 - hA), pB * (1 - hB)

        pA = pA / (pA + pB)
        pB = 1 - pA

        nH, nT = res.count('H'), res.count('T')
        nhA, nhB = nhA + pA * nH, nhB + pB * nH
        ntA, ntB = ntA + pA * nT, ntB + pB * nT

    return nhA / (nhA + ntA), nhB / (nhB + ntB)

if __name__ == '__main__':
    eps = 1e-6
    hA, hB = 0.6, 0.5

    while True:
        h1, h2 = EM(hA, hB, results)
        if abs(h1 - hA) < eps and abs(h2 - hB) < eps:
            break
        hA, hB = h1, h2
    print(hA, hB)
```

输出结果

```
0.7967887593831099 0.5195839356752803
```

# Resources & Reference

- https://www.cnblogs.com/skywang12345/archive/2013/05/30/3106570.html#a6 Bash入门
- https://www.cnblogs.com/tony1314/p/8315666.html [ ]与[[ ]]的区别
- https://blog.csdn.net/universe_hao/article/details/54962630 bash环境变量
- https://zh.wikipedia.org/wiki/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F 操作系统
- https://www.zhihu.com/question/319783573 GNU是什么？和Linux是什么关系？
- https://zh.wikipedia.org/wiki/%E6%AE%BC%E5%B1%A4 Shell
- https://zh.wikipedia.org/wiki/Bash Bash
- https://blog.csdn.net/hulifangjiayou/article/details/47910927 bash shell的优点
- https://zh.wikipedia.org/wiki/%E6%9C%80%E5%A4%A7%E6%9C%9F%E6%9C%9B%E7%AE%97%E6%B3%95 EM最大期望算法

THANK YOU FOR YOUR PARTICIPATION.

# 欢迎提问交流 ~