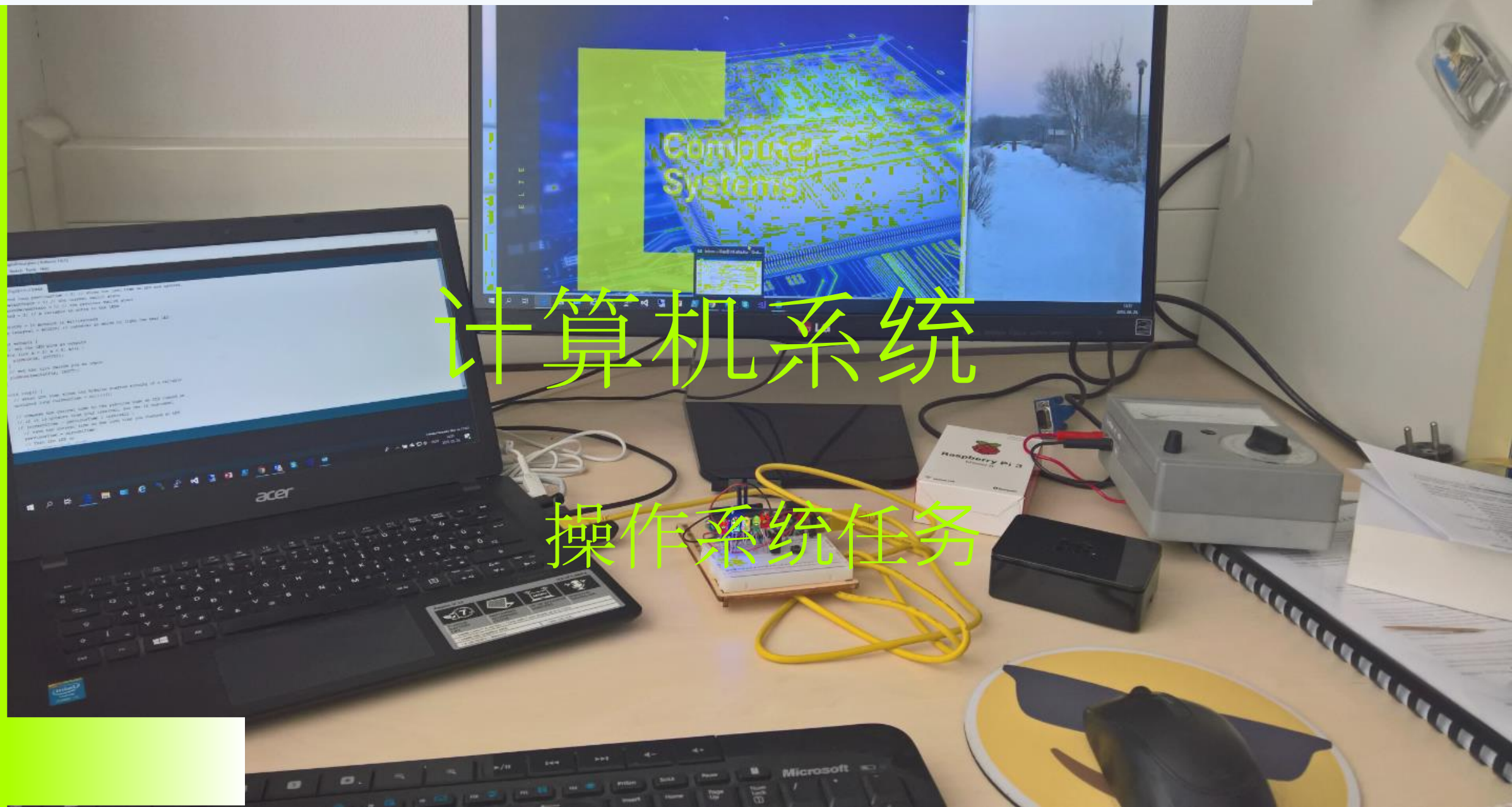


计算机系统 操作系统任务



评论

- 计算机、信号、信息、数字的实现、代码编写
- 架构、操作系统、客户-服务器角色
- 图形化、字符连接、文件系统
- 基本命令、前台和后台进程
- I/O重定向
- 过滤器
- 正则表达式

修改时间：2018年。10.01.

页码:2

今天会有什么？

- 变量和它们的用法
- 变量替换
- 命令替换
- 初级运算（算术、字符串）。
- 逻辑表达法
- ...

修改时间：2018年。10.01.

页码:3

UNIX外壳中的变量。

- 你能够创建一个字符串变量。
 - 有环境变量或shell变量。
- 变量名称的第一个字符是一个字母，后面的字符可以是字母、数字或下划线。
- 环境变量在环境和从环境中启动的每个命令中都是可见的。(全局)
 - **env** - 程序在定义的环境中运行。
 - 在没有参数的情况下，它写出的是变量!
 - 在不同的系统中，写入的变量可能不同
 - **PATH** (**.profile**)，分隔符**:**, **PS1**， **LOGNAME**等。

修改时间：2018年。10.01.

页码:4

UNIX shell中的变量II。


- 定义一个变量：`=` 符号
 - 例如：`团队=RealMadrid`；`数字=5`。
 - `sentence="苹果在树下"`。
 - `special_sentence='这是5美元和3%/10%！'`
- 变量的内容始终是一个字符串!数字不是"数字" - 它们也是字符!
- `set`命令：它写出了所有的变量（环境也是）。
- `unset`命令：删除一个变量
 - `unset team # set result:_=apple`



变量的内容和范围

- 例如：`tallyho="Tally-ho Real Madrid !"`
- 内容：`$tallyho`
 - `echo $tallyho # Tally-ho Real Madrid!`
 - `echo Say: $tallyho #substitution`
- 输出`tallyho`。`tallyho`变量成为一个环境变量!
 - 取消设置也会删除它!
- 环境变量可以被每个人看到!
 - 一个 "正常 "的变量只能通过实际的shell来使用。

变量替换

- `team=RealMadrid; echo $teams` # 结果会是什么？
- `echo $team is the best!; # Naturally` 
- `echo ${team}`是最好的!
- `unset team; x=${team-FCBarcelona}`。# 如果团队变量不存在，x值将是FCBarcelona
 - 如果它不存在，那么它就是一种非正式化。
 - `x=${team=FCBarcelona}`。# 团队也有变化!
 - `echo $x`是最好的!# ???
- `y=${team?Hello}`# 如果没有定义team，Hello被写出来了，shell退出了，y没有得到新的值!
- `y=${team+FCBarcelona}`。# 如果定义了球队，y=FCBarcelona

修改时间：2018年。10.01.

页码:7

命令替换

- ‘命令’该命令将被执行，它将被替换成它的输出。
 - 在Bash shell中：`$(command)`
 - `who_am_i=`whoami` # $(whoami)`
 - `a=`date` ; b='date' #`
 - `echo $a# ? ? ? ?`
 - `echo $b # ? ? ? ?`
 - 结果将是一个词：`date`。
 - 评价 `$b#man` 评价
 - 与命令相同，应该是`$b!`

我的第一个shell脚本

- 文件的名称：第一

```
illes@panda:~$ vi first
illes@panda:~$ chmod +x first
illes@panda:~$ cat first
echo 这是我的第一个shell脚本!
```

I
n
f
o
r
m
a
t
i
k
a

- 一个小的修改...

```
illes@panda:~$ ./first
这是我的第一个shell脚本！
illes@panda:~$ 首先
这是我的第一个shell脚本！
illes@panda:~$
illes@panda:~$ cat first
# ! /bin/sh
```

```
echo 这是我的第一个shell脚本!
```


修改时间：2018年。10.01.

页码:9

在外壳中的操作

- 只有一个操作：字符串连接
 - 在shell中，每个变量都是一个字符串！
- `x=red; echo summer $x apple!# ?`
 - 算术表达式不能够直接被评估。
 - `x=apple; y=$x+tree; echo $y # apple+tree`
 - `a=5; b=$a+1; echo $b #5+1`
- 字符串连接是唯一被支持的操作。

算术运算

- **let**指令，包含在**bash**中，在旧的**sh**中没有包含。
 - `a=2; let b=a+1; echo $b` # result 3
- **表达式指令**
 - `expr $a op $be` .g.: `expr 3* 4 !`
 - 操作: `<, <=, >, >=, !=, =, +, -, *, /, % (mod)`
 - 建议：由于兼容性问题，使用**expr!**
- **bc命令（过滤器）**
 - 它等待一个类似C语言的输入字符串
 - 循环，三角函数，函数的定义是现有的
 - 例如: `echo 2*16 | bc` #32



公元前的例子

- 复杂的例子。
 - 功能定义
 - 平方根和三角函数的使用!
- 文件名：bcexample
- 执行：chmod +x bcexample
 - ./bcexample

```
# ! /usr/bin/bc -l
```

```
# -l选项表示使用数学库（用于使用s） 定义fact (x) {  
    如果 (x <= 1) 返回 (1) 。  
    返回 (fact(x-1) * x) 。  
}
```

```
print " 5 阶乘 =:";
```

```
print fact(5);
```

```
print " !"\n";
```

```
print "25的平方根是。 "; print  
sqrt(25);
```

```
print "\n";
```

```
print "sinus PI/2的值 : "。
```


```
print s(3.1415/2);
```

```
print "\n";
```

```
退出
```



命令参数

- 例子。最好的球队是皇家马德里!`#`如果这是个命令. 
 - 该命令的名称是什么？
- `1, 2, 3, ...` `#` 参数变量
 - `echo $1# team`
- `0#` 命令的名称（最佳）。
- `###` 参数的数量
- `$*#` 每个参数，双引号没有效果！"。
- `"$@"` `#` 参数分开，例如：`param`



参数示例

- 称其为：准桃树'苹果树'

呼应参数的使用

In echo "我们正在给一个复合参数
f 太'苹果树'"#每一个都是单独的
o echo '\$*用法'
r 对于\$*中的i
m 做
a echo \$i #桃子、树、苹果、树 #
t 各自在一个新行中。
i 完成
k
a
E
L
T
E

```
# 一切都在一起
echo "$*"在循环中的用法'
for i in "$*"
do
echo $i# 结果在一行中!
      # 参数在""之间
完成
echo "$@"用法'
# 下面提到的这句话是
#的简短版本：for i in $@
为i
做
echo $i
完成
```

修改时间：2018年。10.01.

页码:14

执行参数的例子

illes@panda:~\$ param grass tree 'apple tree' 参数的用法

我们也在使用一个复合参数（'苹果树'）。

\$*用途草

木苹果树

"\$*"在循环草木苹果树

上的用法 "\$@"的用法

草树 苹果

树

illes@panda:~\$

修改时间：2018年。
10.01.

更多的命令参数

- 审查。
 - 0美元--命令名称
 - 1,...9美元的参数
- 只有9个参数可以？
- 不，我们可以使用{}符号！
- 它是可以的，但大量的参数是相当少的。

修改时间：2018年。10.01.

页码:16

大量参数的使用

- `${10}`

- 第十个参数，没有括号，它是10美元，它将是一个串联（0字符串联到1美元）！
- `${100} # 100.参数...`

- 移位指令

- 它将参数向左移动了一个。
 - 1美元将被扔掉，其他的参数步入左边的一个：`2->$1,...,$10->$9`（如果有\$11，则\$10获得其价值）。
- 在周期的帮助下进行典型处理。

修改时间：2018年。10.01.

页码:17

其他有用的变量

- \$\$ - 正在运行的程序的PID号码!
- \$!- 最后执行的后台进程的PID号!
- \$-----贝壳选项.??????

程序的终止和结果

- 一个shell脚本在完成最后一条指令的执行后，就结束了它的工作，回到了调用者那里，进入了shell!
- 有什么结果吗？
 - 是的，有!
- 每个指示都有一个结果!
 - 这个结果显示了程序的成功或失败!
- 如果一个程序能够有规律地执行它的任务，那么它就能成功

地运行!

- 每个项目的情况都不一样!

修改时间：2018年。10.01.

页码:19

命令的结果

- 这种命令的结果在每一种情况下都会出现!
- 命令的结果在\$?变量中!
- echo hello! ; echo \$?# 0将是echo命令的结果!
- 如果结果是0, 那么命令就成功地运行下来了。
- 如果结果成功。

```
illes@panda:~$ cp frog peter
cp: cannot stat 'frog': No such file or directory
illes@panda:~$ echo $?
1
illes@panda:~$
```

修改时间：2018年。10.01.

页码:20

退出命令

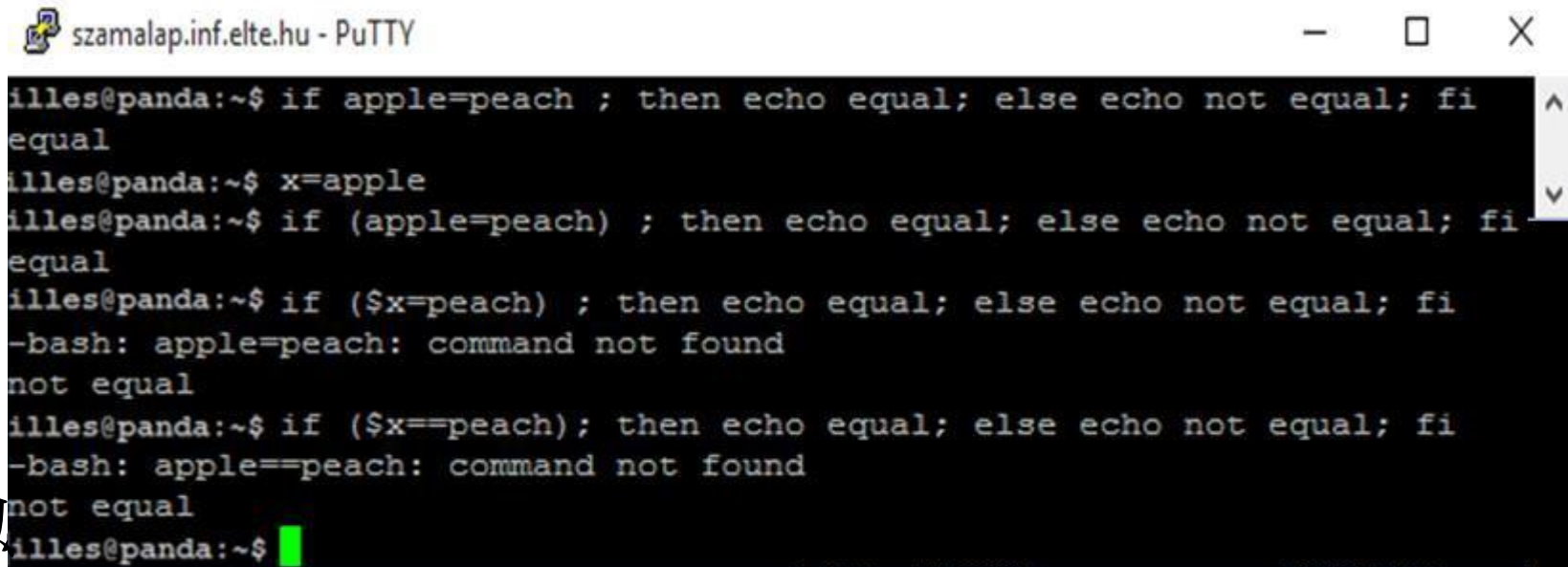
- 使用**exit**命令，我们可以从一个**shell**程序中退出来。
- **exit value#** 其中**value**是0-255之间的整数。
 - 如果**exit**的参数是0，那么就意味着程序执行成功了。
 - 这种成功可以被解释为逻辑上的真实。
 - 如果参数是一个正数，则执行失败！
 - 它可以被解释为逻辑上的错误！
- 它类似于**C**编程语言，但其表示方法是相反的

修改时间：2018年。10.01.

页码:21

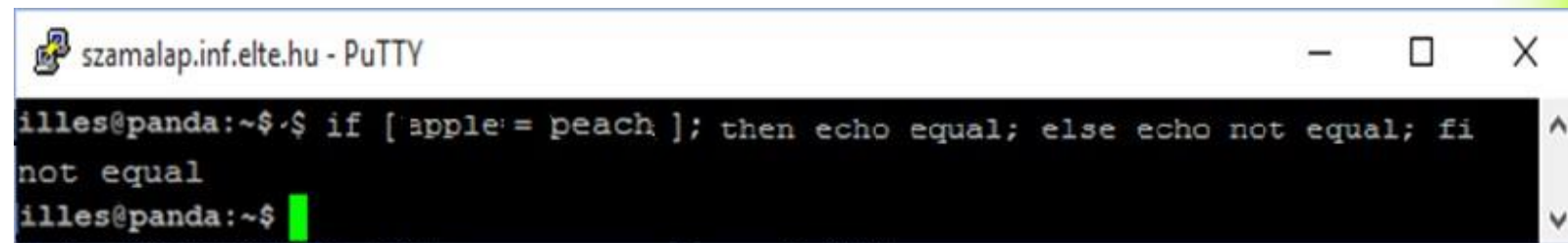
逻辑表达法

- 让我们看看以下情况。
- 本身就没有逻辑性的表达!
- 有一个任务它也可以被解释为一个逻辑性的任务!(真)
- 只有[]的逻辑表达是好的!



szamalap.inf.elte.hu - PuTTY

```
illes@panda:~$ if apple=peach ; then echo equal; else echo not equal; fi
equal
illes@panda:~$ x=apple
illes@panda:~$ if (apple=peach) ; then echo equal; else echo not equal; fi
equal
illes@panda:~$ if ($x=peach) ; then echo equal; else echo not equal; fi
-bash: apple=peach: command not found
not equal
illes@panda:~$ if ($x==peach); then echo equal; else echo not equal; fi
-bash: apple==peach: command not found
not equal
illes@panda:~$
```



szamalap.inf.elte.hu - PuTTY

```
illes@panda:~$ if [ apple=peach ]; then echo equal; else echo not equal; fi
not equal
illes@panda:~$
```

修改时间：2018年。10.01.

页码:22

作为指令结果的逻辑值

- 正如我们所看到的，没有任何逻辑性的操作！
 - 每个命令的结果可以解释为一个逻辑值，也!!!!。
- **测试**指令有帮助!这条指令可以用[]符号代替!
- 测试操作数1 操作数2 # 空格很重要!
 - 或。[operand1 operator operand2] # 这里的空格也很重要，在[!

测试--算术运算

- 测试, 或[...]逻辑检查
- 0-真, 1-假, 回声\$?
- true - 真正的命令, 退出0
- False - 错误的命令, 退出1
 - -lt,-gt,-le,-ge,-eq,-ne 数学考试
 - [\$x -lt 5]
 - -f 文件, -d dir 文件或目录存在
 - -o, or, -a and operator, ! Negation

- `-r,-w,-x` 测试文件权限的读取、写入、执行

修改时间：2018年。10.01.

页码:24

测试--字符串， 文件检查

- 对比字符串。

- =, !=相等、不相等的字符串比较（感叹号和等号，如C语言中的或C++）

- test \$a = \$b # true, if \$a and \$b are equal strings

- test \$a != \$b # true, if \$a and \$b are not equal strings

- test -z \$X # true, 如果\$X字符串的长度=0， 它就是一个空字符串。

- test -n \$X # true, if the length of \$X string is not 0, not an empty

- 完整的参考资料：男子测试

修改时间：2018年。10.01.

页码:25

测试样本 - I.

- 这个样本显示了最经常使用的功能!
- 简单的逻辑表达式。

```
$ x=4
```

```
$ [ $x -lt 6 ] # 测试 $x -lt 6
```

```
$ echo
```

```
$?0 (真)
```

```
$ test "alma" = "körte" # [ "alma" = "körte" ]
```

```
$ # 文字方程的结果
```

```
$ echo $?
```

```
1美元 (假) 。
```

```
$test -z "alma" # 检查空字符串
```

```
$echo
```

```
$?1
```

修改时间：2018年。
10.01.

癌症的治疗方法

测试Példa - II

○

- 复式逻辑检查
- 基金会, 基金会的活动情况

```
$ x=4
```

```
$ y=fradi
```

```
$ [ $x -eq 4 -a $y != "vasas" ]
```

```
$ # 逻辑与关系 (-a) 。
```

```
$ echo
```

```
$?0 (真)
```

```
$ [ ! $x -eq 4 ]
```

```
$ echo
```

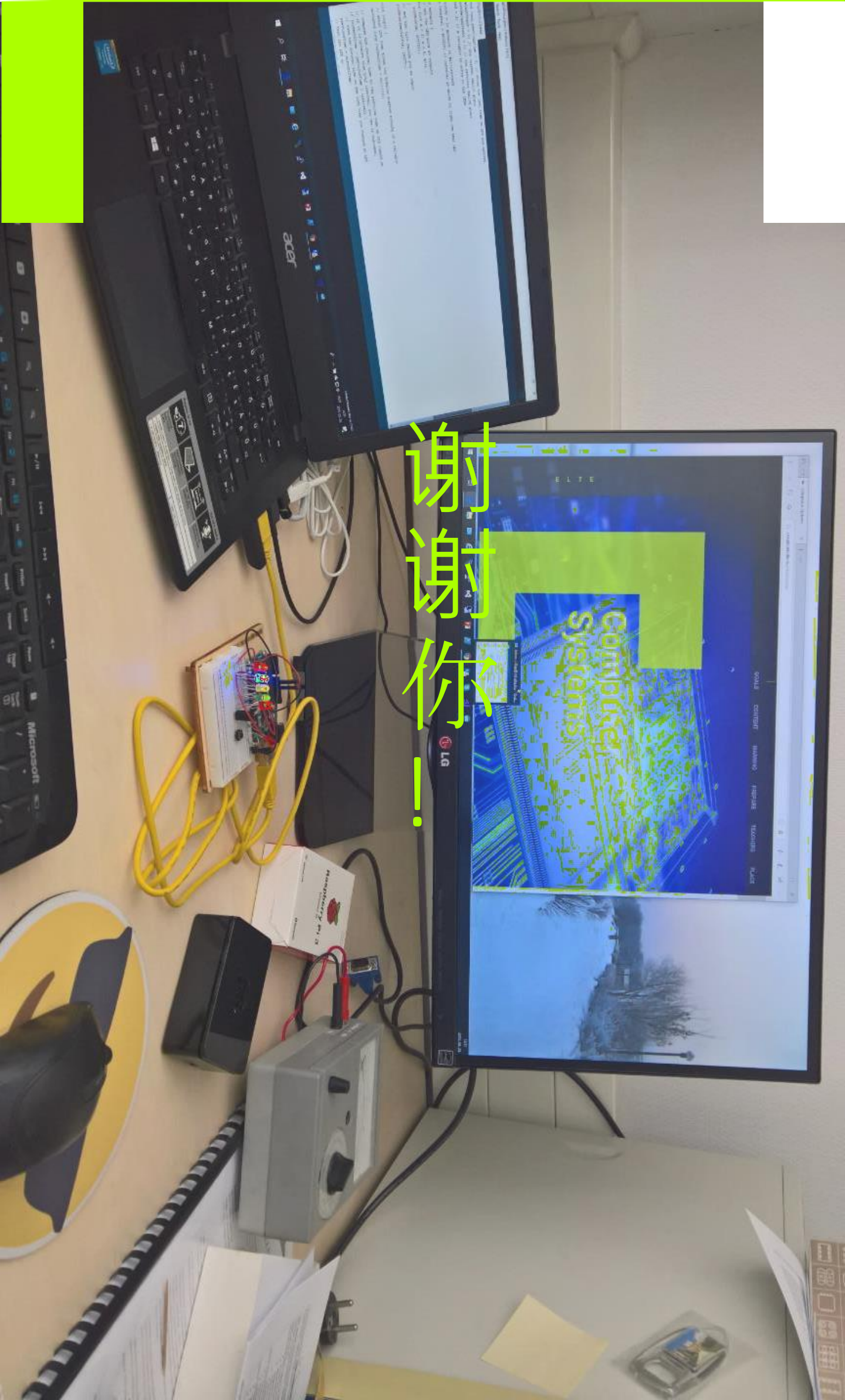
```
$?1 (假)
```

```
$ test -f fradi # true, if file fradi is exist
```

```
$ [ -d alma ] # true, if directory alma is exist
```

修改时间：2018年。
10.01.

癌症的治疗方法



谢谢你！