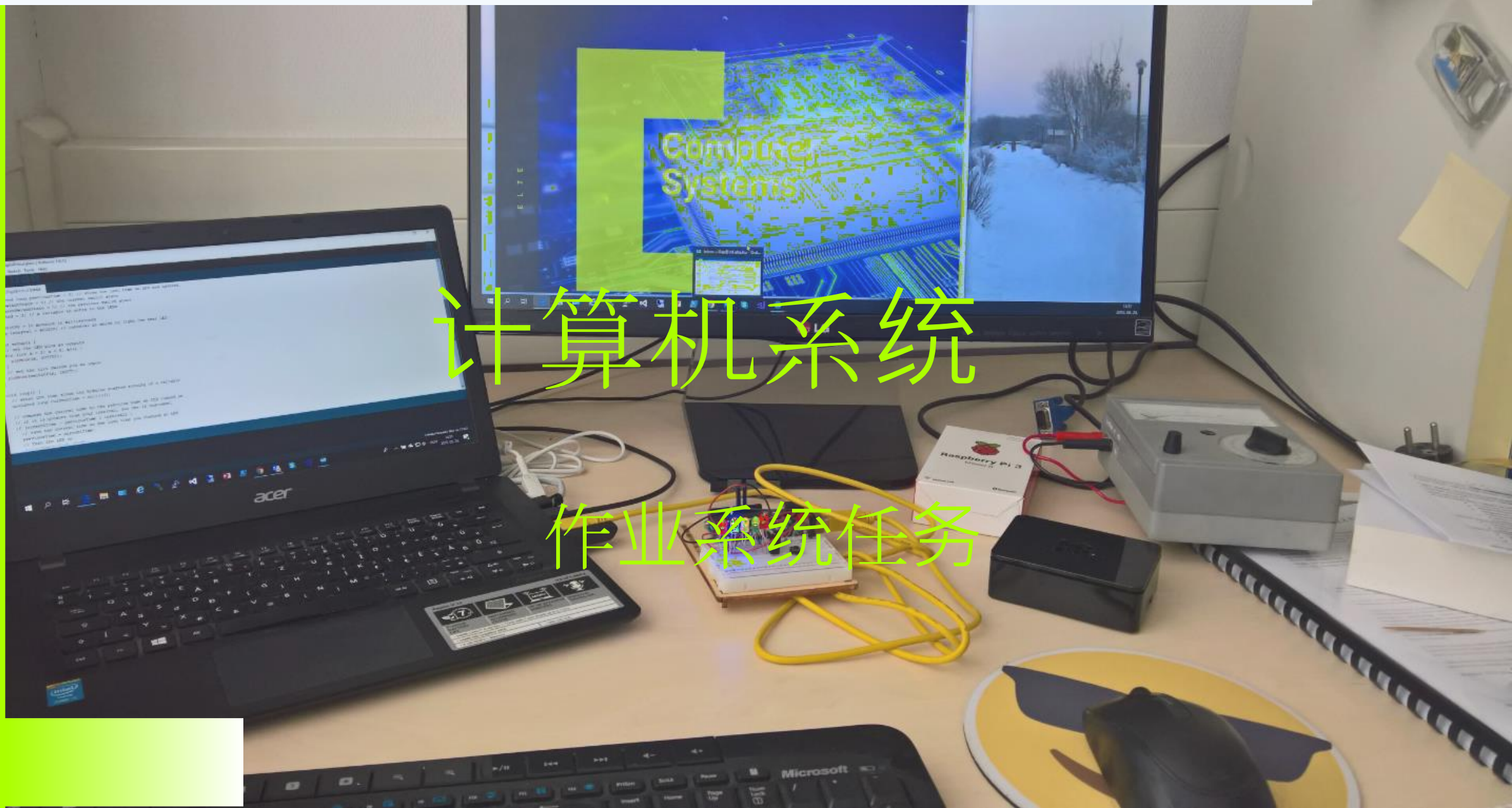


# 计算机系统 作业系统任务



# 评论

- 计算机、信息、数字的表示、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
- 变量、命令替换

- 算术、逻辑表达

# 今天会有什么？

- 控制结构
- 分支机构
- 循环
- 功能定义
- ...

# 一个外壳的例子

- 下面的脚本是做什么的？
- 苹果有一个特殊的结构。
  - X=ldared&y=colorisred
  - 这种公式在哪里使用？
- 它从输入的登录名(login)和密码(pw)中切出！

读取苹果

```
login=`echo $apple|cut -f1 -d&|cut -f2 -d=`  
pw=`echo $apple|cut -f2 -d&|cut -f2 -d=`
```

```
cat <<till  
Content-Type: text/html
```

```
<html>  
<正文 bgcolor="#a1c1a1">
```

弊病

echo 登录名是：\$login，密码是。

\$pw！

```
cat <<till  
</body> </html>
```

直到



# 有条件的命令执行

- 如果成功，那么其他命令 fi
  - 成功&&其他命令
- 如果没有成功，那么其他命令就会出现。
  - 不成功||其他命令
  - 例子。

```
$ echo hello && echo kate  
hello
```

凯特

```
$ false || echo appletree # false:  
exit 1
```

小树

计算机系统



# 壳牌的多分支：案例

- 如果\$apple在  
idared)   echo 苹果是idared  
          ;;  
金色)    回声 苹果是金色的  
          ;;  
\*)       echo 这是一个未知类型的苹果  
          ;;  
esac

# 分支机构的例子一。

- 任务。读出一个数字，并决定它是正数、零数还是负数！

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

```
# 如果第一行是备注，它可以是shell的定义 读取x # 读取到变量x  
中  
如果
```

```
    [ $x -lt 0 ] 然  
后
```

```
    echo 变量X为负数，其值为：$x else  
    如果
```

```
        [$x -eq 0 ]  
        然后
```

```
            echo 变量x的值为零！否则
```

```
            echo 变量x是正数，值是：$x
```

```
斐济
```

斐济 计算机系统

# 分支机构实例二。

- 任何命令的执行都可以产生一个值，而这个值可以作为分支的逻辑测试值！

如果

```
who | grep john >/dev/null
```

然后

```
echo john is logged in
```

else

```
echo john is not logged in
```

fi

读取x #读取案例\$x

in

```
[dD]*)          日期      ;;
```

```
[wW]*)          谁        ;;
```

```
l*|L*)          ls -l     ;;
```

# 小l或大写字母L

\*)

echo  
bad  
choice;;  
esac

# 壳中的循环。为

- 在数据列表中为变量# 为循环的一般形式
  - 做
    - 说明
  - 完成
- for i in `who`# 最经常使用的形式
 

```
do
echo $i
done
```
- for i in apple pear peach# 经典使用 for cycle
 

```
做
echo $i
```

#一个接一个地把水果写出来 done

# 对于样本--seq

- `for i in 1 2 3 4 5; do echo $i; done` # 5次循环
- 如何创建一个N次循环？
  1. 或者我们必须写一个脚本，产生1...N个输出！"。
  2. 或者我们使用seq命令！
- `N=10; for i in `seq $N`;do echo $i; done` # 1-10个数字
  - `seq 2 6` # 2,3,4,5,6
  - 序号 2 2 6 # 2,4,6
- 更多信息: `man seq`

## 癌症的治疗方法

---



# 巴什 - 为

- `for x in $(date)`      # Bash命令的替换
  - 做
    - `cat $x`
  - 完成
- `for ((i=1;i<10;i++))`    # C风格的循环
  - 做
    - `echo $i`
  - 完成
- 我们建议使用经典的 (sh) 循环!

# 壳中的循环。 循环：WHILE

- while指令  
做

#被执行

完成

- 而

echo -n 给我你的名字。

读取名称

! 做#

echo 你的名字是：\$name

done

#一般使用while， 如果最后

#指令为真， 循环种子将是

指令

#从键盘上读取永远不会给出一个错误的值！

来完成， 例如：ctrl+c

# 壳中的循环。 WHILE例子一。

- 如果读入的是文件末尾的符号， 结果将是假的
- 而

echo -n 给我你的名字。

read nev # 它将逐行读入文件do# 在读出的

文件末尾给出

back# 一个假值!

echo 你的名字是 : \$nev

done < names.txt

# 壳中的循环。 WHILE例子二。

- 如果参数是一个文件名，那么我们就把它的内容列出来！

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

```
while [ $# != 0 ] #还有其他参数吗？ do# yes
```

```
if test -f $1# $1 file?
```

```
然后
```

```
echo $1 content:
```

```
cat $1
```

```
else
```

```
echo $1不是一个文件!
```

```
斐济
```

```
shift #它将参数向左移动 echo 还有$#个
```

```
参数！ done
```

# 壳内循环。循环，直到

- 如果条件为真，则执行cyccle-seed，如果条件为假，则执行until!
- 直到  
    指示(s)做  
    已完成的循环播种指令(s)
- 几条指令可以后，如果最后一条是假的，循环种子指令就要被执行了!

# shell中的循环(loops)。 UNTIL例子

- 我们继续向文件中写名字，同时没有得到一个"不"!
- 在 [ \$answer = [Nn]o ] 的情况下，会是什么情况？
  - 没有什么，因为测试指令不 "喜欢" 正则表达式!

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

```
# 给出一个起始值 answer=yes
```

```
直到
```

```
[$answer = "no" ]
```

```
do
```

```
echo -n 给我你的名字。
```

```
阅读名称
```

```
echo $name >>names.txt
```

```
echo Continue?\
```

```
话
```

```
# 它创建了这个文
```

```
#如果它不存在的
```

```
# `是重要的
```

阅读答案  
完成

# 巴什：选择

- 编写菜单结构（仅在BASH中）。
- 选择I在苹果桃子李子
- 做
- i处理
- 完成
- 重要提示：BREAK

```
pandora.inf.elte.hu - PuTTY
illes@pandora:~$ select i in Continue Exit; do echo $i; if
> [ $i = "Exit" ]
> then
> break
> fi
> done
1) Continue
2) Exit
#? 1
Continue
#? 2
Exit
illes@pandora:~$
```



# 贝壳中的跳转指令

- 突破
  - 循环在break命令下终止，完成后继续执行程序。
- 继续
  - 循环种子的剩余部分被过度跳跃，循环的执行继续进行。
- 退出[n]。
  - 退出程序，返回值将是n!

# 壳I中的功能定义。

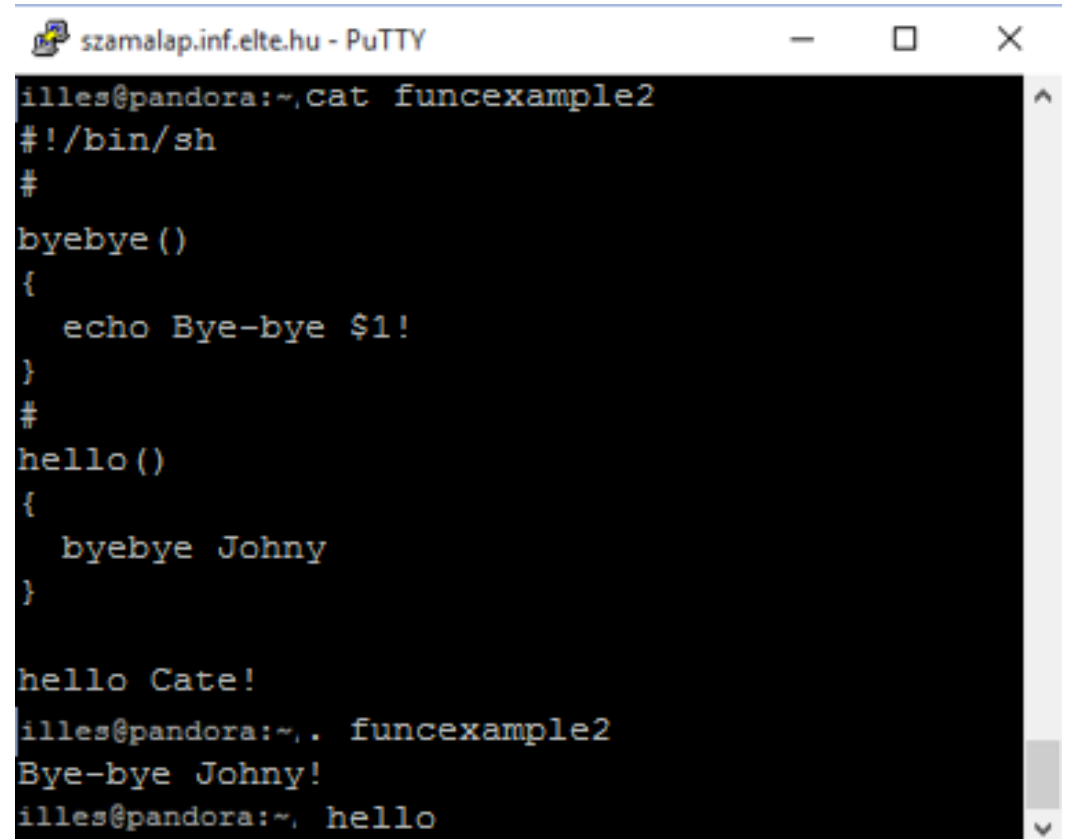
- C风格的函数，以类似的方式处理参数（如在shell脚本中）。
- 我们不能定义参数!
- 函数值可以通过返回指令来分配!

```
illes@panda:~$ cat funcexample
#!
/bin/sh #
byebye()
{
    回声：再见1美元!
```

```
byebye Johnny#Function call
#end of the script
illes@panda:~$ funcexample
Bye-bye Johnny!
```

# 壳中的功能定义二。

- 一个函数可以调用另一个函数!
  - hello函数不处理它的参数!
- 脚本名称调用。
  - 如何从命令行中达到功能。
  - `unset -f functionname`
  - # 删减功能



```
szamalap.inf.elte.hu - PuTTY
illes@pandora:~,cat funcexample2
#!/bin/sh
#
byebye()
{
    echo Bye-bye $1!
}
#
hello()
{
    byebye Johny
}

hello Cate!
illes@pandora:~,. funcexample2
Bye-bye Johny!
illes@pandora:~, hello
```

# 命令执行选项

- `Sh funcexample #` 没有执行权限就会被执行!
- `Sh -v funcexample`
  - `-v`它在执行前写出命令。整个函数也是如此!
- `Sh -x funcexample`
  - 在`-x`选项下, 它只写出测试的逻辑表达式。(在`-v`选项下, 它写出了整个分支。)
- `Sh -n funcexample`
  - 语法检查

# 更多BASH

- 除了标准shell的可能性外，我们还提到了BASH的实现！  
例如，对于循环
- 我们并不想强调BASH！的每一种可能性。
  - 例如，数组的使用就是这种可能性。
- 完整的BASH参考资料同样可以从这里获得。  
<https://www.gnu.org/software/bash/manual/bash.html>

# 脚本实例一。

- 下面的脚本是做什么的？

```
为i  
做  
case $i  
在  
[A-Z]*)F="$F$i";。  
[a-z]*) f="$f $i"; 。  
[0-9]*) break; 。  
esac  
done  
echo $F  
echo $f
```

# 脚本示例二。 让我们做一个包!

- 例子。

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

```
echo '# 包装制作'
```

```
echo '# 再次把它打碎 : sh ./filename # 我们一个接一个地得到参数 for i  
做
```

```
echo "echo $i 1>&2"
```

```
echo "cat >$i <<'$i end'"
```

```
# 当 "是主程序时, '$i结束'被评估为cat $i
```

```
echo "$i end" #这里的输入结束了  
完成
```

# 包装的使用

- 包装应用程序将在下面写出结果  
标准输出a!

**\$打包forexample排序#结果在屏幕上显示**

**\$ ....**

**\$ pack forexample sorting > package #result into file**

**\$ sh ./package# unpacking**

**# 如果我们没有X权限，我们确实要这样开始#**



# 示范

- 包装脚本使用的现场演示

