Python 简介

唐 欢
智能中心ASL小组
htang@ncic.ac.cn

2004-09-28

Content

- What’s In
  - 初识 Python
  - 快速入门
  - 操作系统接口
  - GUI设计
  - DCMS系统简介
- What’s NOT In
  - 调试
  - web程序设计
  - ……

初识 Python

Q1:What’s Python?

- Python is an ***interpreted, interactive, object-oriented*** programming language.

  - 1989年末，Guido van Rossum 为了突破ABC在软件开发能力的局限性，获取Amoeba机操作系统提供的系统调用。同时放弃发明Amoeba机专用语言的思路，选择通用语言。
  - 后来由荷兰CWI公司(发明ABC语言)继续开发，1991年公开发表。

Q2：Why Python？

- 软件品质
  - 可读性强（较其他脚本语言，Perl）
  - 重用性OOP
- 快速开发
  - 无需编译
  - 动态变量类型
  - 自动内存管理
- 可扩展性
  - 模块

Q2：Why Python？-2

- 可移植性
  - Unix(Solaris,Linux,FreeBSD,AIX,HP/UX,SunOS,IRIX)
  - Win 3.x/9x/NT/2000/ Windows CE
  - Macintoch
  - OS/2
  - DOS
  - PalmOS
  - Acorn/RISC OS
  - VMS/Open VMS
  - QNX
  - VxWorks
- 与其它语言的结合
  - 运行C/C++库，在C/C++程序中调用python
  - Java
  - communicate over COM, Corba, and .NET
  - SOAP and XML-RPC

Q3：What Can I Do with Python?

- Systems Programming
  - Portable command-line tools, testing systems
- GUIs
  - With APIs such as Tk, QT,MFC, Gnome, KDE
- Internet Scripting
  - CGI web sites, Java applets, XML, ASP, email tools
- Component Integration
  - C/C++ library front-ends

Q3：What Can I Do with Python?

- Database Programming
  - Persistent object stores, SQL database system interfaces
- Distributed programming
  - With client/server APIs like CORBA, COM
- Rapid Prototyping
  - Throwaway or deliverable prototypes
- Gaming, Images, AI, XML, and More

Q4:Who Uses Python Today?

- 500,000 and 1 million Python users around the world (Year2003)
- Internet services -Google and Yahoo!
- hardware testing -HP, Seagate, IBM
- movie animation -Industrial Light and Magic

More on http://www.pythonology.org/success

Q5：What Technical Strengths?

- Object-Oriented
- Free
- Portable
- Powerful
- Mixable
- Easy to Use & Learn

Q6: Difference to Language X?

- TCL--- Python’s support for “programming in the large”
- Perl --- cleaner syntax and simpler design
- Java ---simpler and easier scripting language
- C++ --- Python often serves different roles
- Visual Basic --- cross-platform & open source
- SmallTalk and Lisp --- simple, traditional syntax

Q7:What’s the Downside?

- Speed
  - 作为解释型语言，其运行速度一定无法与编译执行的语言相比。
  - 然而，由于其强大的扩展能力，可以将对运行时间敏感的组件使用其它语言(例如C)编写。然后将其与Python结合使用。

IDE & Editor

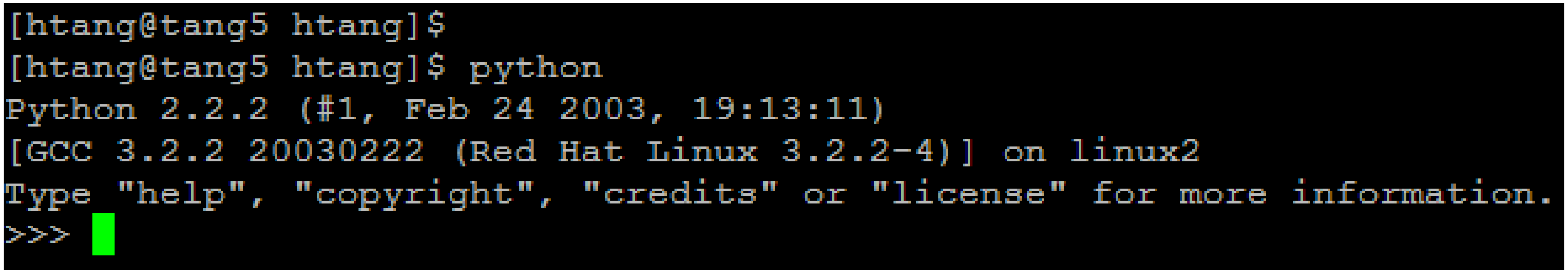
- VIM
- Emacs
- IDLE(Python+Tkinter)
- Eclipse
- Komodo
- BlackAdder (QT)
- PythonWin (Windows)
- Editplus (python.stx) /Source Insight (python.CLF)/ UltraEdit

More at http://www.python.org/moin/IntegratedDevelopmentEnvironments

快速入門

Starting and Stopping

- Unix



```
[htang@tang5 htang]$
[htang@tang5 htang]$ python
Python 2.2.2 (#1, Feb 24 2003, 19:13:11)
[GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Windows & Macintosh
  - Python 作为应用程序运行
- Termination
  - Ctrl+D

Hello World

- 命令行

```
>>>print 'Hello World!'
Hello World!
>>>
```

- 文件

```
#!/usr/bin/python
print 'Hello World!'            # Our first program
<EOF>
```

  - 运行文件：

```
% python hello.py
% ./hello.py
```

Variables and Expressions

- Expressions

```
3 +5
3 ** 2
'Tang' + 'huan'
```

- Variable assignment

```
a = 4<<3
b = a * 4.5
a = 'Tang huan'
```

  - 动态变量类型

Basic Type(Number)

- Number

```
a = 3                    # Integer
b = 3.1415926535         # Floating
c = 34343434334L         # Long int
d = 4 + 3j               # Complex
```

  - 长整数位数不受限制
  - 不区别 float 和 double
  - 直接支持复数类型

Basic Types (Strings)

- Strings

```
a = ’tang’                                    # Single quotes
b = ”huan”                                    # double quotes
c = ’Bob said ”Yes!” then go.’                # Mix
d = ”””This is uncle Wang.
   Uncle Wang likes making things.
   He makes many things.”””                   # Multiple lines
e = ’:-) ’ * 20                               # Repeat
```

- Python 不区分 char 与 String
- 不区分单引号与双引号
- 提供符号复制运算

Basic Type(List)

- List

```
a = [2 , 3 ,4]                  # A list of integer
b = [4, 3.1415, ’Hello’]        # A mixed list
c = []                          # An empty list
d = [2, [], [a, b]]             # A list containing a list
e = a + b                       # Join two lists
```

- List Manipulation

```
x = a[1]                        # x = 3
y = b[1:3]                      # y = [3.1415, ’Hello’]
z = d[2][0][1]                  # z = 3
a[1] = 9                        # Change an element
x = a[-1]                       # x = 4
```

  - Python没有数组(Array)
  - 列表长度及其元素都可变
  - 列表可以嵌套、合并
  - 下标可以为负数

Basic Type(Tuple)

- Tuple

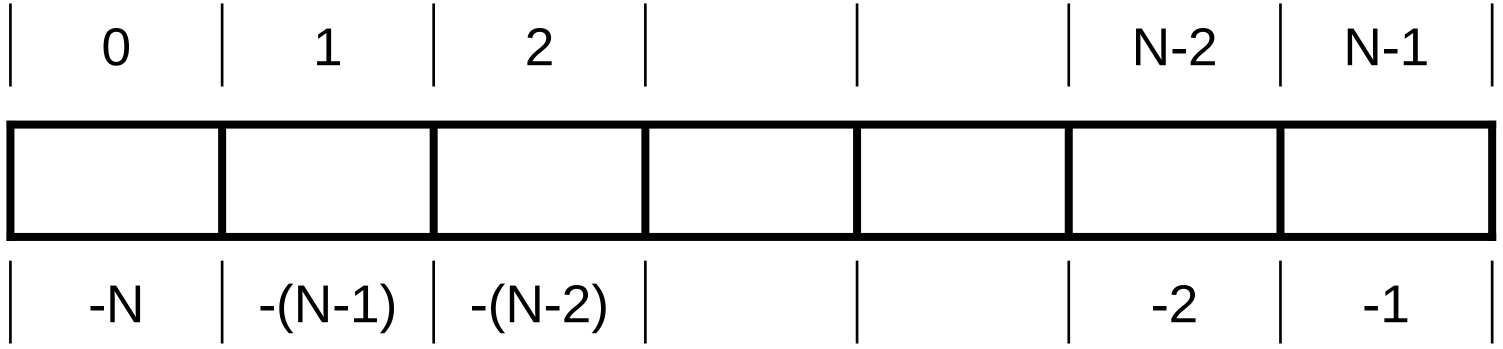
```
f = (2, 3, 4, 5)              # A tuple of integer
g = ()                        # An empty tuple
h = (2, [3, 4], f, (5,6))     # Minxed object
```

- Tuple Manipulation

```
x = f[1]                      # x = 3
y = f[1:3]                    # y = (3, 4)
z = h[1][1]                   # z = 4
x = f[-2]                     # x = 4
```

  - Tuple 称为“表列”，
  - 不同于列表(list)，表列(tuple)长度及元素不可修改。常用于参数传递，提高安全性。
  - 例外情况：h [1][0] = 9 成立

Sequences Manipulation



- Python将String、List、Tuple统称为Sequence(序列)对于序列的访问与其它语言有所差异：
  - 两种下标方式，使得对于序列的访问更加方便快捷
- 于是有：

  Seq[len(Seq) – 1] == Seq[–1]
  Seq[1 : len(Seq)–1] == Seq[1: –1]
  Seq[len(Seq)-5 : len(Seq)] == Seq[-5:]
  Seq[0 : len(Seq)-1] == Seq[: -1]

Funny

- 交换两个变量的值：

```
>>> a=1
>>> b=2
>>> (a,b)
(1, 2)
>>> (a,b)=(b,a)
>>> (a,b)
(2, 1)
>>>
>>> [a,b]=[b,a]
>>> [a,b]
[1, 2]
>>>
```

Basic Type(Dictionary)

- Dictionary

```
a = { }                                          # An empty one
b = {’x’ : 3,  ’y’ : 4}
c = { ’name’ : ’Cao Zheng’,   ’uid’ : 501，  ’home’ : ’/home/cz/’ }
```

- Dictionary Access

```
u = c[’uid’]                                     # get a element  u=501
c[’home’] = “/work/hpcog/”                       # set a element

if c.has_key(’shell’):
        d = c[’shell’]
else:
        d = None
d = c.get(’shell’,None)                          # Same thing
```

Conditionals

- if-else

```
# Compute Maximun (z) of a and b
if a < b:
    z = b
else:
    z = a
```

- pass

```
z = b
if a < b:
    pass    # Do nothing
else:
    z = a
```

Python 没有 ` ? : ` 操作符

Conditionals

- elif

```
if a == `+`:
        op = PLUS
elif a == `-`:
        op = MINUS
elif a == `*`:
        op = MULTIPLY
else:
        op = UNKNOWN
```

Python 没有 `switch`

Loop

- while

```
while a < b:
    a += 1                              # python has NO ‘a++’
```

- for

```
for i in [3, 4, 66, 128]:
    print i
for c in ”Hello World!”:
    print c
for i in range(3,10):
    print i
```

- for循环不同于其它语言
- range()内建函数产生连续序列

Function

- def

```
def remainder(a,b):
      q = a/b
      r = a – q*b
      return  r
y = remainder(44,3)                    # y=2
```

  - Returning multiple values

```
def divide(a,b):
      q = a/b
      r = a – q*b
      return  q,r
x , y = remainder(44,3)                # x=12,  y=2
```

Class

- class

```
class Account:
        def _ _init_ _ (self, initial):
                self.balance = initial
        def deposit(self, amt):
                self.balance = self.balance + amt
        def withdraw(self, amt):
                self.balance = self.balance – amt
         def getbalance(self):
                return self.balance
```

- Using

```
a = Account(10000.00)
a.depostit(123.45)
print a.getbalance()
```

Module

- File nmber.py

```
# number.py
def divide(a,b):
        q = a/b
        r = a – q*b
        return  q,r

def format(x,y):
        return ”The quotient is %d,
and the remiander is %d” %(x,y)
```

- File main.py

```
#!/usr/bin/python
import number
x , y = number.devide(44,3)
print number.format(x,y)
```

Exception

- try

```
try:
	f = open(”foo”)
except IOError:
	print ”Opening file ‘foo’ Error.”
```

- raise

```
def func(n):
	if n<0:
		raise ValueError, “Expected non-nagative number”
	else:
		 return n * func(n-1)
```

File

- open()

```
f = open (“foo”,”w”)
g = open (“bar”,”r+”)
```

- reading and writing data

```
f.write(“Hello world!”)
data = g.read()                    # Read all
line = g.readline()                # Read a single line
lines = g.readlines()              # Read data as a list of lines
```

- close()

```
f.close()
```

Python Library

- Python standard modules
  - String processing
  - Operating Sysytem interfaces
  - Networking
  - Threads
  - GUI
  - Database
  - Security ….
- And many third party modules
  - XML
  - Numeric Processing
  - Graphics ….

Structure of a moudle file

```
#/usr/bin/env python                    <1>启动语句行
""" This is a test module               <2>模块文档
    Author: Some body
"""
import sys                              <3>模块导入
import string
debug = 1                               <4>全局变量定义
class FooClass:                         <5>类定义
    pass
def test():                             <6>函数定义
    foo = FooClass()
    if debug:
        print 'ran test()'
if _ _ name_ _ == '_ _main_ _':           <7>程序"主体"
    test()
```

操作系统接口

OS moudle

- Wide variety
  - Basic system call
  - Operating environment
  - Processes
  - Timers
  - Signal handling
  - Error reporting
  - Users and passwords
- 并非适用于所有操作系统(Windows/Mac)

Process Environment

- **os.environ – A dirctionary**

```
user = os.environ[‘USER’]
os.environ[‘PATH’] = ”/bin:/usr/bin”
```

- **Current directory and umask**

```
os.chdir(path)          # Change current working directory
os.getcwd()             # Get current working directory
os.umask(mask)          # Change umask setting. Returns
                          previous umask
```

- **User and group identification**

```
os.getgid()             # Get group id
os.getuid()             # Get user id
os.setgid(gid)          # Set group id
os.setuid(uid)          # Set user id
```

Process Creation and Destruction

- **fork-exec-wait**

```
os.fork()                    # Create a child process.

os.execv(path,args)          # Execute a process
os.execve(path, args, env)
os.execvp(path, args)        # Execute process, use default path
os.execvpe(path,args, env)

os.wait([pid)]               # Wait for child process
os.waitpid(pid,options)      # Wait for change in state of child

os.system(command)           # Execute a system command

os._exit(n)                  # Exit immediately with status n.
```

Process Creation and Destruction

- Example

```
import os
pid = os.fork()                    # Create child
if pid == 0:
    # Child process
    os.execvp("ls", ["ls","-l"])
else:
    os.wait()                      # Wait for child
```

Pipes

- **os.popen()**

```
f = popen("ls -l", "r")
data = f.read()
f.close()
```

  - popen() returns a file-object.
- **The popen2 module**
  - Spawns processes and provides hooks to stdin, stdout, and stderr

```
popen2(cmd)      # Run cmd and return (stdout, stdin)
popen3(cmd)      # Run cmd and return (stdout, stdin, stderr)
```

Pipes

- Example (DCMS-Agent,RPM Module)

```
def rpm_install(params):
        params[0] = string.replace(params[0], ' ', '\ ')
        command = 'rcp ' + params[0] + ' /tmp/x.rpm'
        x = os.popen3(command)
        x2 = x[2].read()
        if len(x2) == 0:
                y = os.popen3('rpm -ivh /tmp/x.rpm')
                y2 = y[2].read()
                if len(y2) != 0:
                        output = [1, str(y2)]
                else:
                        output = [0, '']
        else:
                output = [1, 'Copying the rpm file failed!\n']
        return output
```

The commands Module

- **The easy way to capture the output of a subprocess**

```
import commands
data = commands.getoutput("ls -l")
```

- 其实现仅仅是对popen2()作了一层封装
- Only available on Unix.

Error Handling

- **System-related errors**
  - OSError – 系统错误
  - IOError – I/O错误
- **Example:**

```
import os, errno
…
try:
    os.execlp("foo")
except OSError,e:
    if e.errno == errno.ENOENT:
        print "Program not found. Sorry"
    elif e.errno == errno.ENOEXEC:
        print "Program not executable."
    else:
        # Some other kind of error
```

Signal Handling

- **The signal module**

```
signal.signal(signalnum, handler)   # Set a signal handler
signal.alarm(time)                  # Schedules a SIGALRM signal
signal.pause()                      # Go to sleep until signal
signal.getsignal(signalnum)         # Get signal handler
```

- **Supported signals (platform specific)**

SIGABRT	SIGFPE	SIGKILL	SIGSEGV	SIGTTOU	SIGALRM	SIGHUP
SIGPIPE	SIGSTOP	SIGURG	SIGBUS	SIGILL	SIGPOLL	SIGTERM
SIGUSR1	SIGCHLD	SIGINT	SIGPROF	SIGTRAP	SIGUSR2	SIGCLD
SIGIO	SIGPWR	SIGTSTP	SIGVTALRM	SIGCONT	SIGIOT	
SIGQUIT	SIGTTIN	SIGWINCH	SIGXCPU	SIGXFSZ		

Time

- **The time module**

```
time.clock()           # Current CPU time in seconds
time.time()            # Current time (GMT) in seconds since epoch
time.localtime(secs) # Convert time to local time.
time.gmtime(secs)    # Convert time to GMT (returns a tuple)
time.asctime(tuple)  # Creates a string representing the time
time.ctime(secs)       # Create a string representing local time
time.mktime(tuple)   # Convert time tuple to seconds
time.sleep(secs)      # Go to sleep for awhile
```

- Example

```
import time
t = time.time()
# Returns (year,month,day,hour,minute,second,weekday,day,dst)
tp = time.localtime(t)
# Produces a string like ’Mon Jul 12 14:45:23 1999’
print time.localtime(tp)
```

User and Group

- **The pwd module**
  - Provides access to the Unix password database

```
pwd.getpwuid(uid)          # Returns passwd entry for uid
pwd.getpwname(login)       # Returns passwd entry for login
pwd.getpwall()             # Get all entries
```

- **The grp module**
  - Provides access to Unix group database

```
grp.getgrgid(gid)          # Return group entry for gid
grp.getgrnam(gname)        # Return group entry for gname
grp.getgrall()             # Get all entries
```

- Expmple

```
>>>import pwd
>>> pwd.getpwuid(517)
('htang', 'x', 517, 517, '', '/work/htang', '/bin/bash')
```

GUI设计

Python, Qt and PyQt

- **Python**
  - 并没有像VB等语言一样拥有自己的GUI接口。不过其仍然拥有众多的图形库可以使用：
    - wxPython
    - Tkinter
    - PyGTK
    - PyFLTK
    - FoxPy
    - PyQt

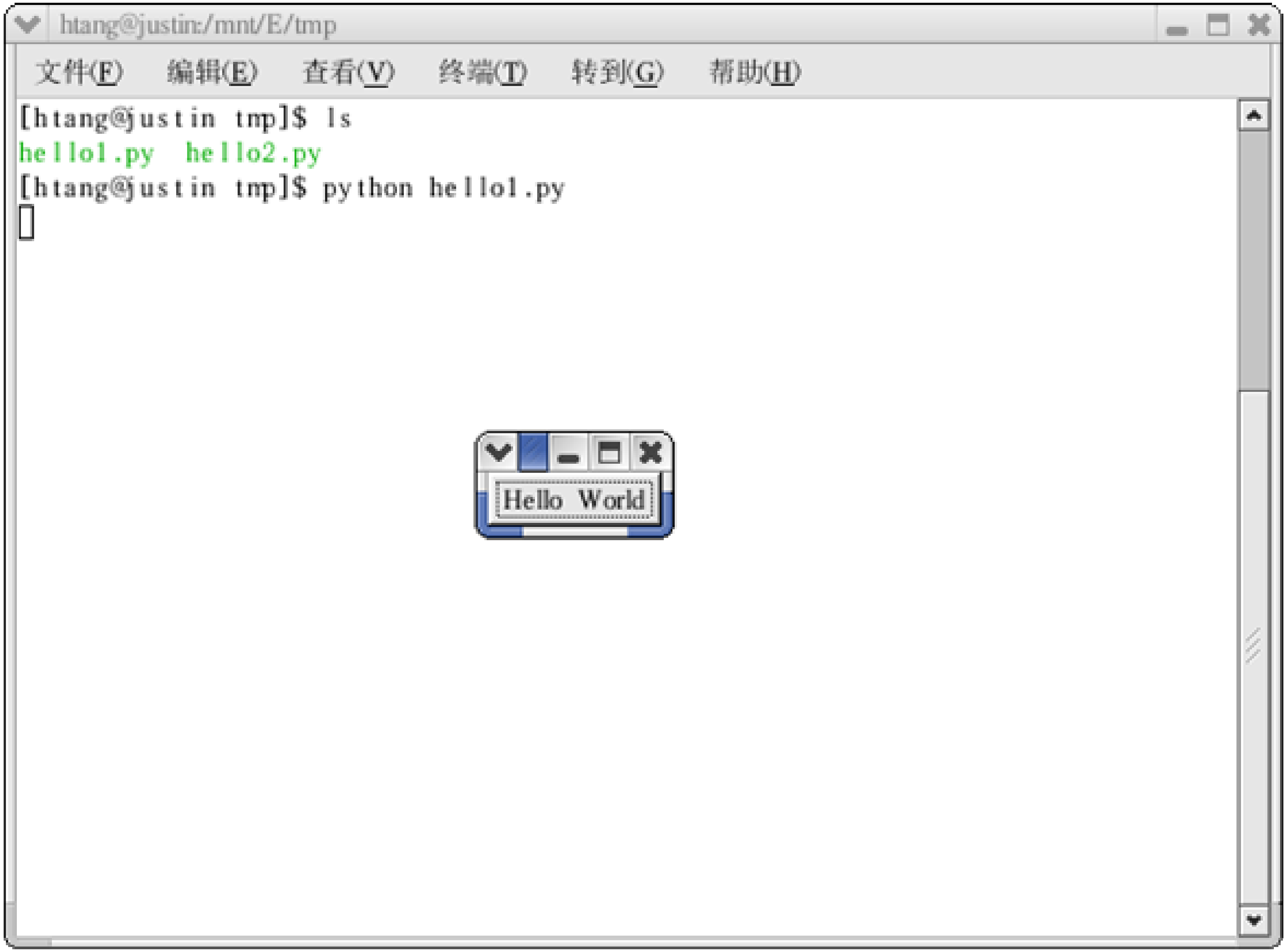
Python, Qt and PyQt

- **QT**
  - Qt是一个多平台的C++图形用户界面应用程序框架。Qt是完全面向对象的，很容易扩展，是流行的Linux桌面环境KDE 的基础。
  - 支持多种平台：
    - MS/Windows — 95、98、NT 4.0、ME、和2000
    - Unix/X11 — Linux、Sun Solaris、HP-UX、Compaq Tru64 UNIX、IBM AIX、SGI IRIX和其它很多X11平台
    - Macintosh — Mac OS X
    - Embedded — 有帧缓冲(framebuffer)支持的Linux平台。
  - 多种版本
    - Qt企业版和Qt专业版 — 提供给商业软件开发
    - Qt自由版 —开放源码软件 提供的Unix/X11版本
    - Qt/嵌入式自由版 — 为了开发自由软件提供的嵌入式版本

Python, Qt and PyQt

- PyQt
  - Qt面向C++语言
  - PyQt是Qt工具包与Python语言的一组绑定
  - 绑定的实现是将PyQt作为Python的Module：
    - qt
    - qtcanvas
    - qtgl
    - qtnetwork
    - qtsql
    - qttable
    - qtui
    - qtxml
  - 共实现了300个类超过5750个函数

Hello World -1



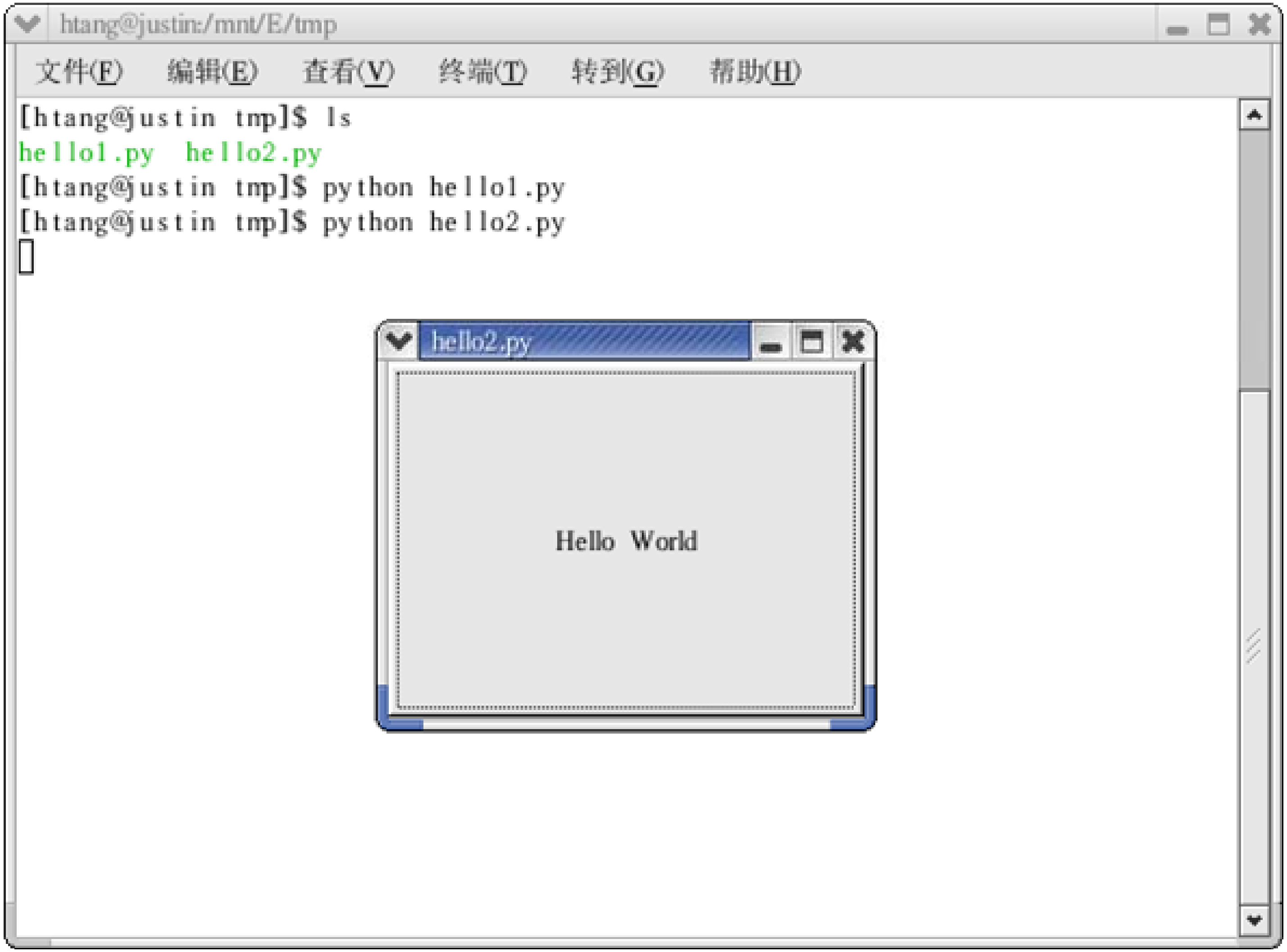
```
[htang@justin tmp]$ ls
hello1.py  hello2.py
[htang@justin tmp]$ python hello1.py
```

Hello1.py

```
#
# hello1.py
#
import sys                                    #  两种Import方式
from qt import *
app=QApplication(sys.argv)                                  <1>
button=QPushButton("Hello World", None)                     <2>
app.setMainWidget(button)                                   <3>
button.show()
app.exec_loop()                                             <4>
```

1. 生成应用程序对象，处理分发鼠标键盘事件
2. Widget是PyQt中所有可视化类的父类
3. 通知应用程序对象Button将作为其唯一Widget
4. 开始应用程序对象的事件循环

Hello World -2



```
[htang@justin tmp]$ ls
hello1.py  hello2.py
[htang@justin tmp]$ python hello1.py
[htang@justin tmp]$ python hello2.py
```

Hello2.py --Better Structure

```
import sys
from qt import *

class HelloButton(QPushButton):
    def __init__(self, *args):
        apply(QPushButton.__init__, (self,) + args)
        self.setText("Hello World")

class HelloWindow(QMainWindow):                                   <1>
    def __init__(self, *args):
        apply(QMainWindow.__init__, (self,) + args)
        self.button=HelloButton(self)
        self.setCentralWidget(self.button)                        <2>

def main(args):                                                   <3>
    app=QApplication(args)
    win=HelloWindow()
    win.show()                                                    <4↓>
    app.connect(app, SIGNAL("lastWindowClosed()"), app,SLOT("quit()")
    app.exec_loop()

if __name__=="__main__":
    main(sys.argv)
```

Hello2.py

1. QMainWindow： 绝大多数PyQt程序需要创建一个主窗口和至少一个可见的Widget
2. setCentralWidget() 将button和mainwindow连接起来，同时Mainwindow将自动管理button位置
3. 定义函数main()，这里仅仅为了寻求与C语言“看起来”一致，可以是任意的名字。
4. 一个应用程序有可能打开多个window，在仍有window存在的情况下不应该使得程序终止。因此，只有当lastWindowClosed() 事件发生时，才可以调用quit()方法结束应用程序。

   使用“signal-slot”机制联系事件与其对应的处理方法，是Qt一个重要特色。

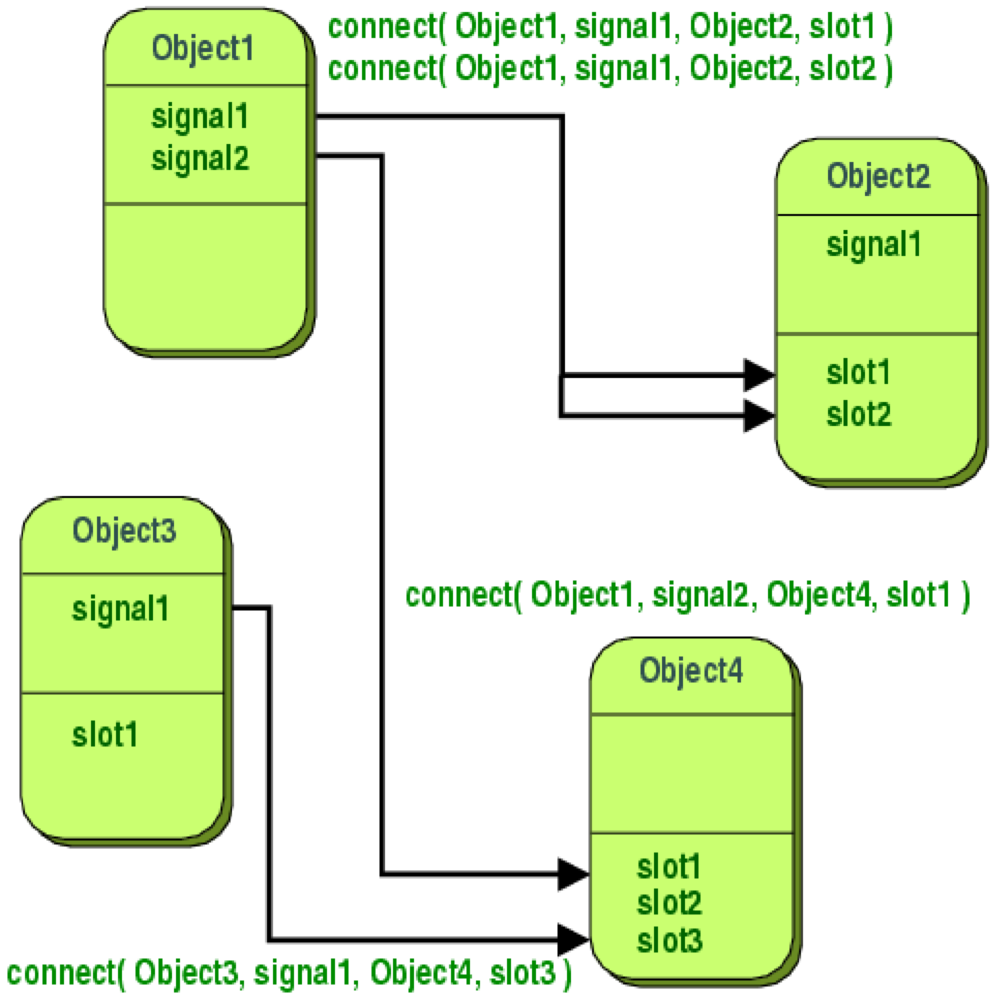
Signals and Slots

- 在图形用户界面编程中，我们经常希望一个窗口部件的一个变化被通知给另一个窗口部件。更一般地，我们希望任何一类的对象可以和其它对象进行通讯。
- 例如：
  - 如果我们正在解析一个XML文件，当我们遇到一个新的标签时，我们也许希望通知列表视图我们正在用来表达XML文件的结构。

Signals and Slots

- 其他程序包使用 Callbacks 通讯方式
  - 回调是指一个函数的指针，如果你希望一个处理函数通知你一些事件，你可以把另一个函数（回调）的指针传递给处理函数。处理函数在适当的时候调用回调。
- 主要缺点
  - 不是类型安全的。我们从来都不能确定处理函数使用了正确的参数来调用回调。
  - 回调和处理函数是非常强有力地联系在一起的，因为处理函数必须知道要调用哪个回调。

Signals and Slots



connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
Object1
signal1
signal2
Object2
signal1
slot1
slot2
connect(Object1, signal2, Object4, slot1)
Object3
signal1
slot1
Object4
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

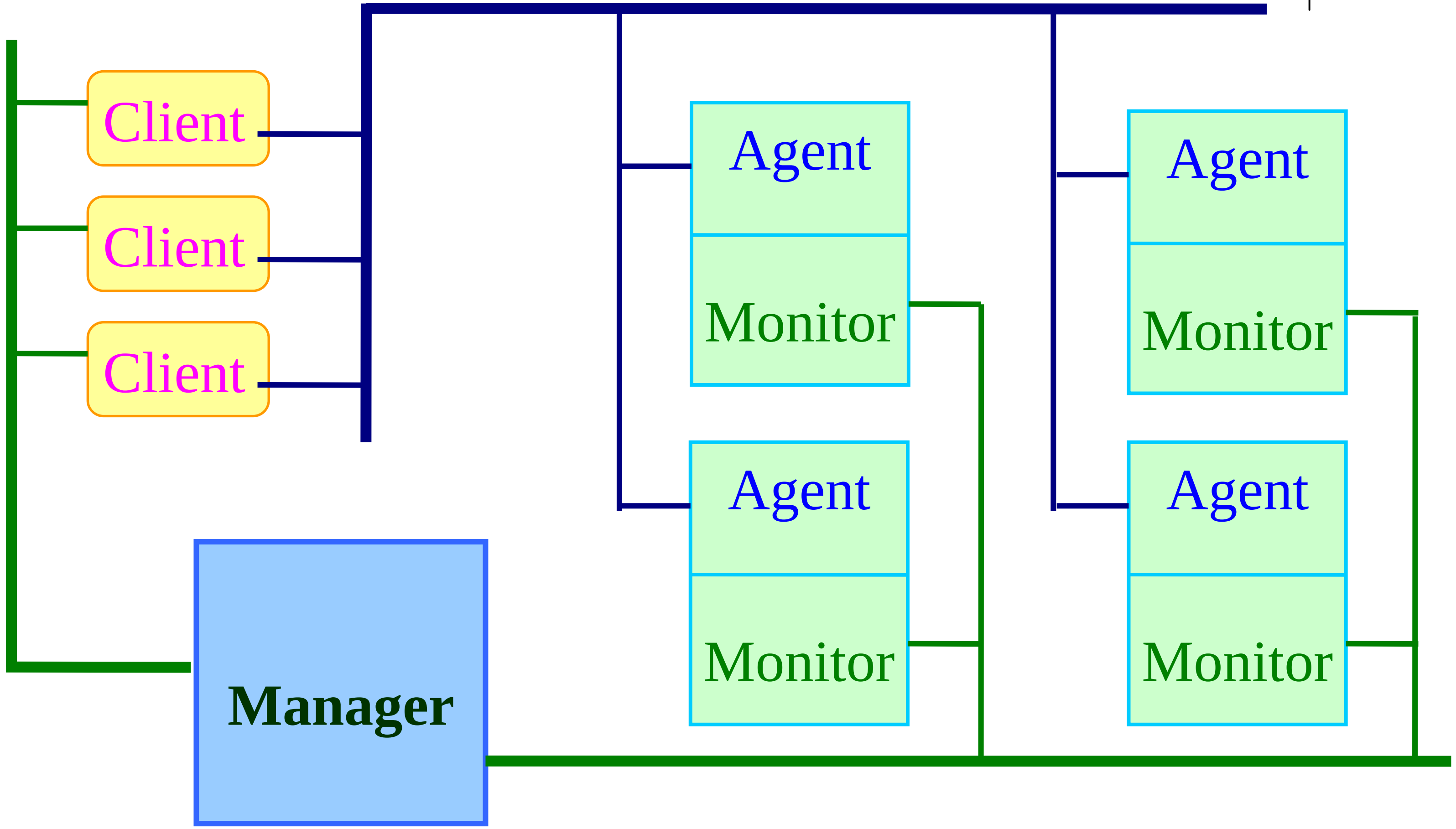
- Qt中使用 signal/slot 机制
  - 当一个特定事件发生的时候，一个***Signal***被发射。
  - ***slot*** 就是一个可以被调用处理特定信号的函数。
- 特性：
  - 对象并不知道是否有函数处理其发射的Signal
  - 一个Slot也仅仅是一个正常的成员函数，本身也不知道其是否连接Signal
  - 一个Signal可以连接到多个Slot，还可以连接到其它的信号。

DCMS系统简介

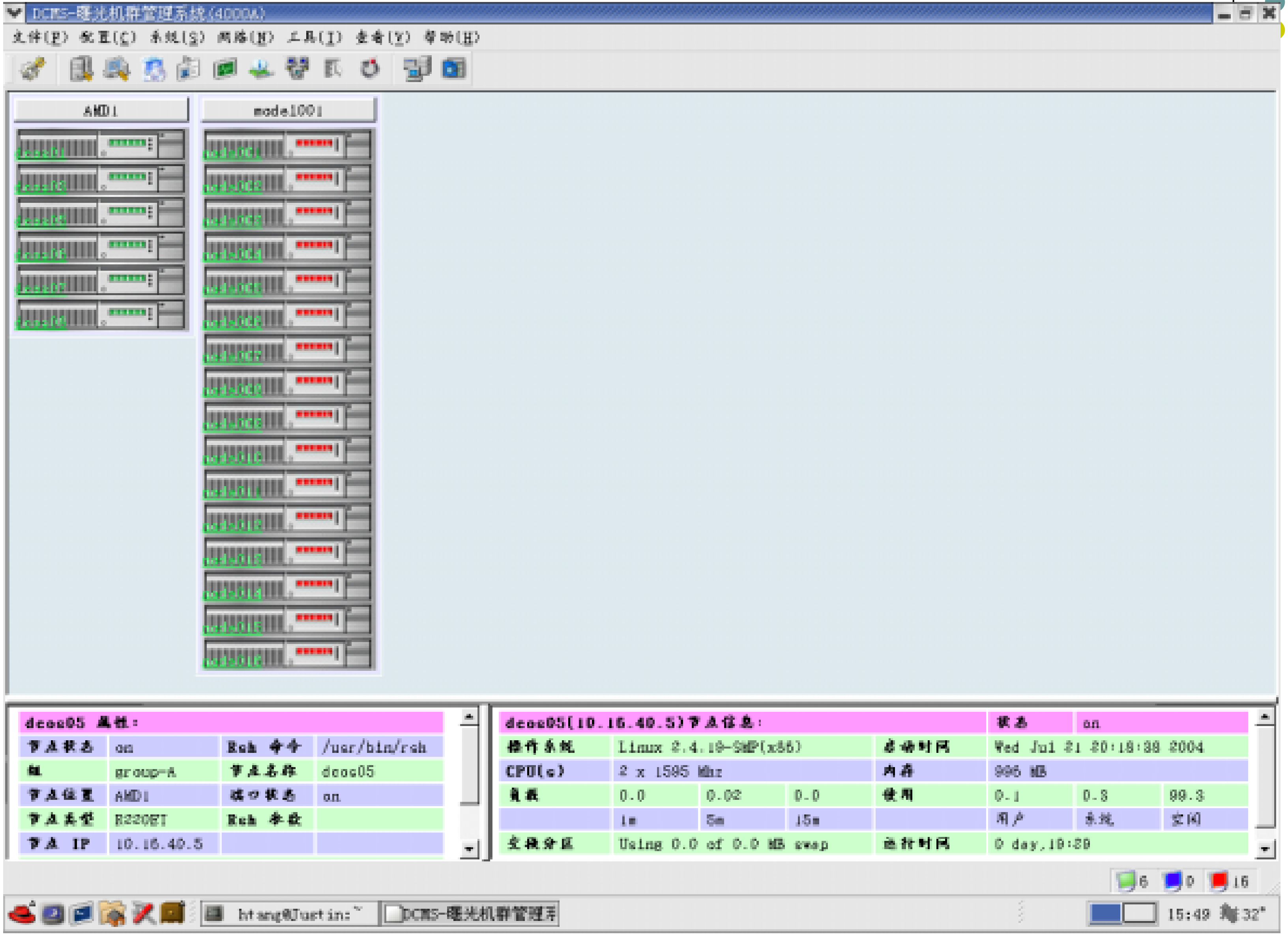
DCMS

- Dawning Cluster Management System
  - 是一款对曙光集群系统进行管理的工具软件。通过图形化界面对集群进行统一监控和配置管理，其主要功能包括：
    - 状态监控预警(Cpu、内存、交换分区负载，进程监控)
    - RPM安装包管理
    - 用户/组管理
    - 进程管理
    - 网络管理 (IP、DNS、Host 配置)
    - 结点关机管理

All done by Python!



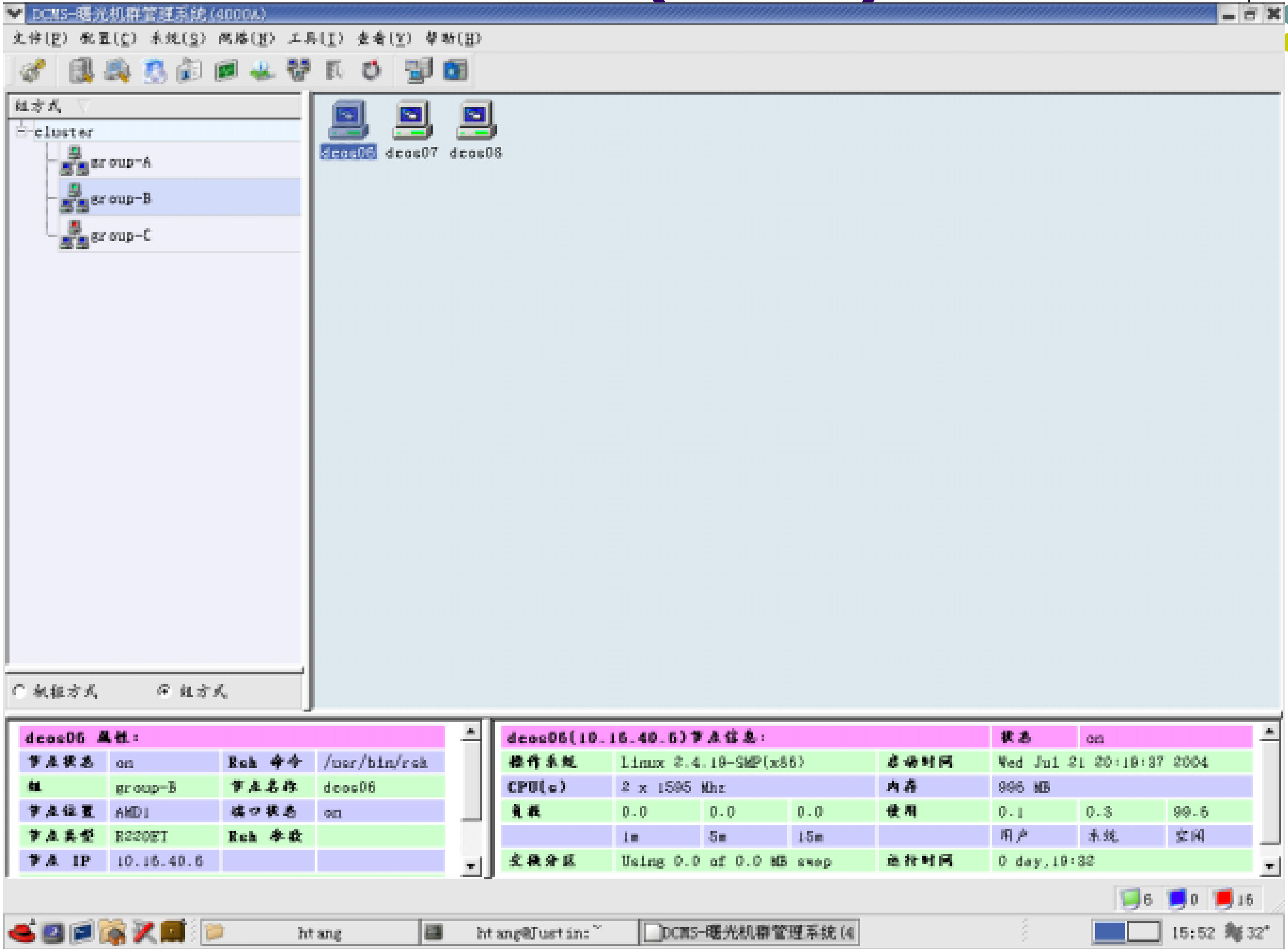
Screenshot—01(main)


DCMS-曙光机群管理系统(4000A)
文件(F) 配置(C) 系统(S) 网络(N) 工具(T) 查看(V) 帮助(H)
AMD1
node1001
6
0
16
15:49

dcos05 属性:			
节点状态	on	Rsh 命令	/usr/bin/rsh
组	group-A	节点名称	dcos05
节点位置	AMD1	端口状态	on
节点类型	R220ET	Rsh 参数	
节点 IP	10.16.40.5		

dcos05(10.16.40.5)节点信息:				状态	on	
操作系统	Linux 2.4.19-SMP(x86)			启动时间	Wed Jul 21 20:18:38 2004	
CPU(s)	2 x 1595 Mhz			内存	996 MB	
负载	0.0	0.02	0.0	使用	0.1	0.8	99.3
	1m	5m	15m		用户	系统	空闲
交换分区	Using 0.0 of 0.0 MB swap			运行时间	0 day,19:29	

Screenshot—02(main)

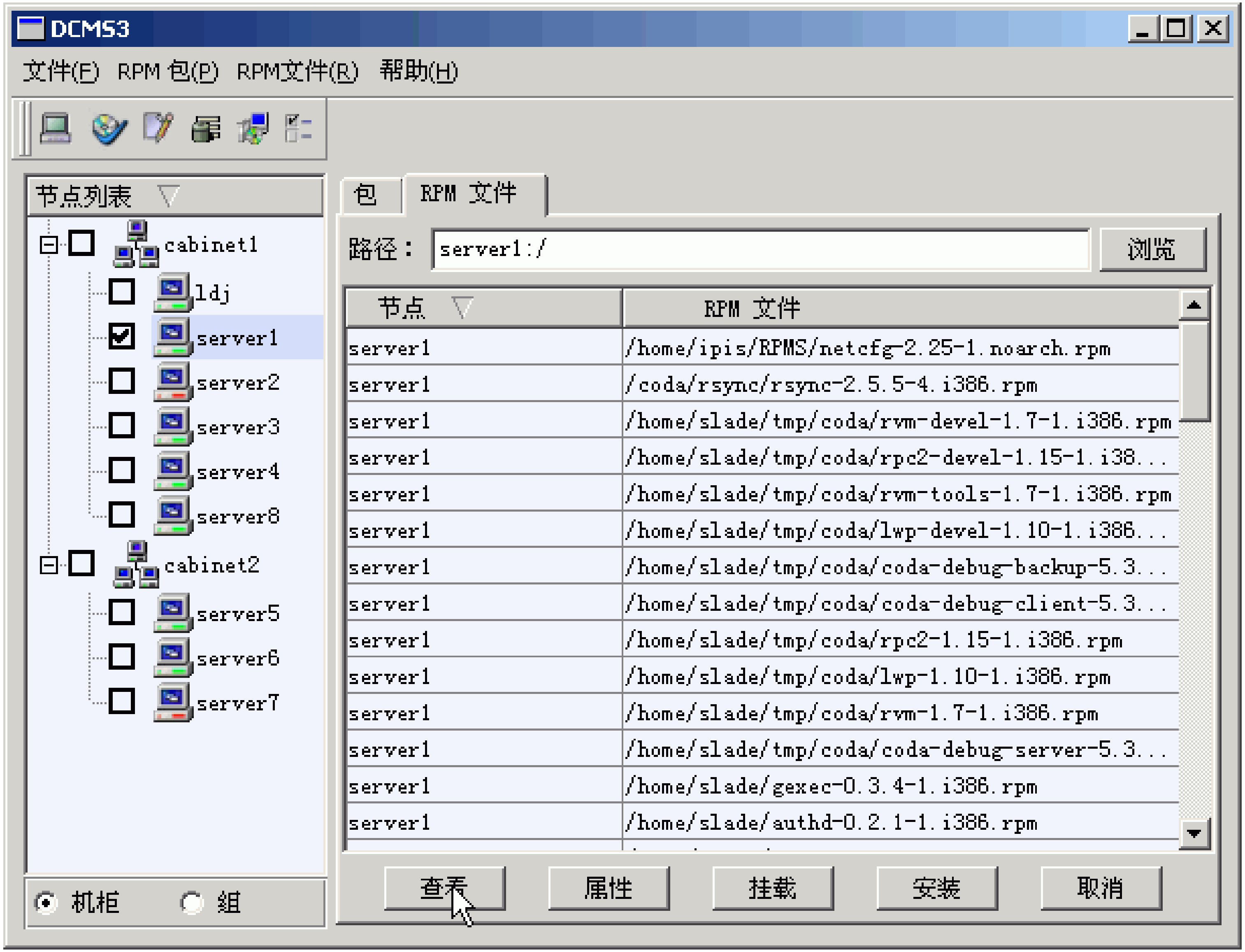

DCMS-曙光机群管理系统(4000A)
文件(F) 配置(C) 系统(S) 网络(N) 工具(T) 查看(V) 帮助(H)
组方式
cluster
group-A
group-B
group-C
dcos06
dcos07
dcos08
机柜方式
组方式
15:52

dcos06 属性:			
节点状态	on	Rsh 命令	/usr/bin/rsh
组	group-B	节点名称	dcos06
节点位置	AMD1	端口状态	on
节点类型	R220ST	Rsh 参数	
节点 IP	10.16.40.6		

dcos06(10.16.40.6)节点信息:				状态	on		
操作系统	Linux 2.4.18-SMP(x86)			启动时间	Wed Jul 21 20:18:37 2004		
CPU(s)	2 x 1595 Mhz			内存	996 MB		
负载	0.0	0.0	0.0	使用	0.1	0.3	99.6
	1m	5m	15m		用户	系统	空闲
交换分区	Using 0.0 of 0.0 MB swap			运行时间	0 day,19:32		

🟩 6 🟦 0 🟥 16

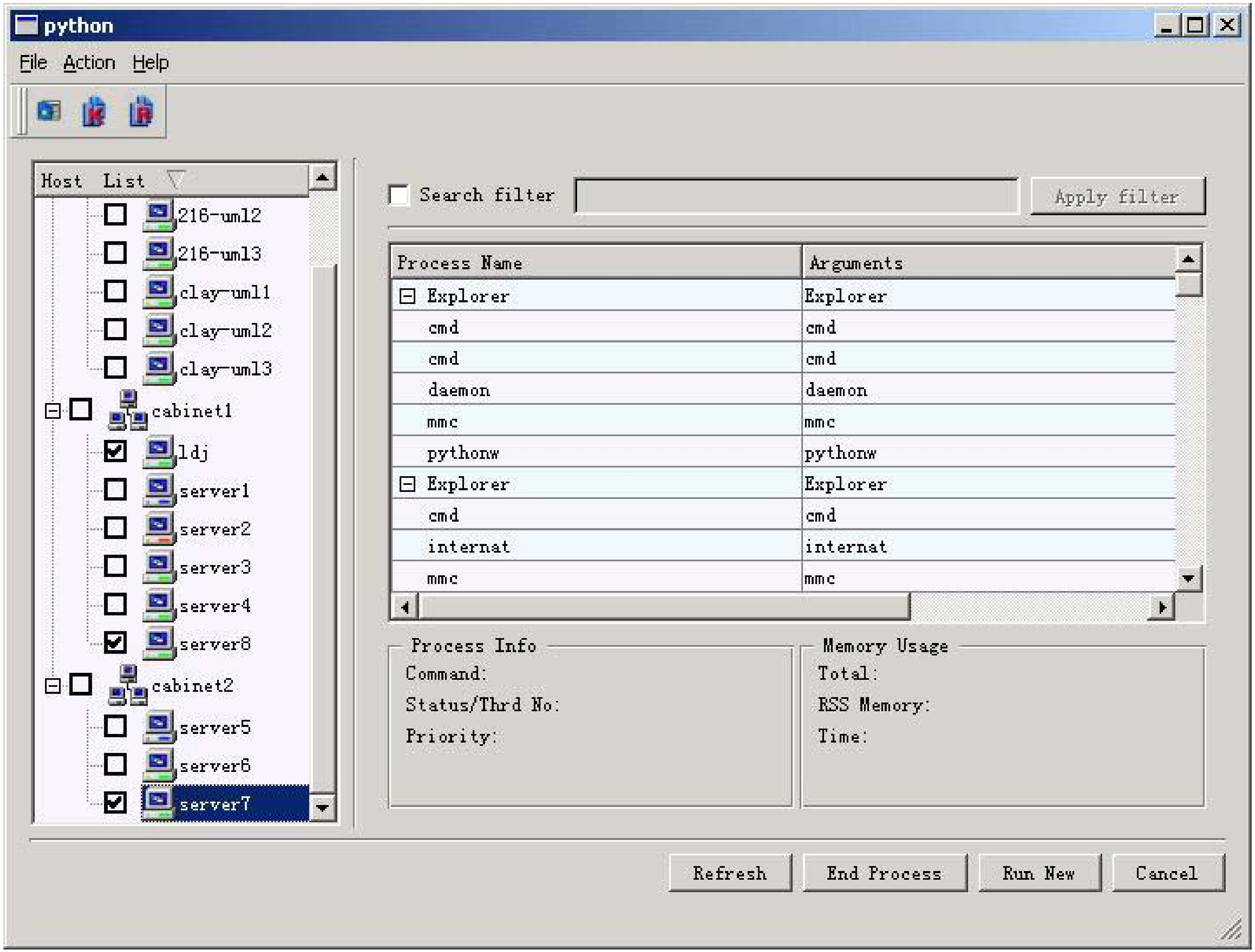
Screenshot—03(RPM)



路径：server1:/

节点	RPM 文件
server1	/home/ipis/RPMS/netcfg-2.25-1.noarch.rpm
server1	/coda/rsync/rsync-2.5.5-4.i386.rpm
server1	/home/slade/tmp/coda/rvm-devel-1.7-1.i386.rpm
server1	/home/slade/tmp/coda/rpc2-devel-1.15-1.i38...
server1	/home/slade/tmp/coda/rvm-tools-1.7-1.i386.rpm
server1	/home/slade/tmp/coda/lwp-devel-1.10-1.i386...
server1	/home/slade/tmp/coda/coda-debug-backup-5.3...
server1	/home/slade/tmp/coda/coda-debug-client-5.3...
server1	/home/slade/tmp/coda/rpc2-1.15-1.i386.rpm
server1	/home/slade/tmp/coda/lwp-1.10-1.i386.rpm
server1	/home/slade/tmp/coda/rvm-1.7-1.i386.rpm
server1	/home/slade/tmp/coda/coda-debug-server-5.3...
server1	/home/slade/gexec-0.3.4-1.i386.rpm
server1	/home/slade/authd-0.2.1-1.i386.rpm

Screenshot—04(Process)



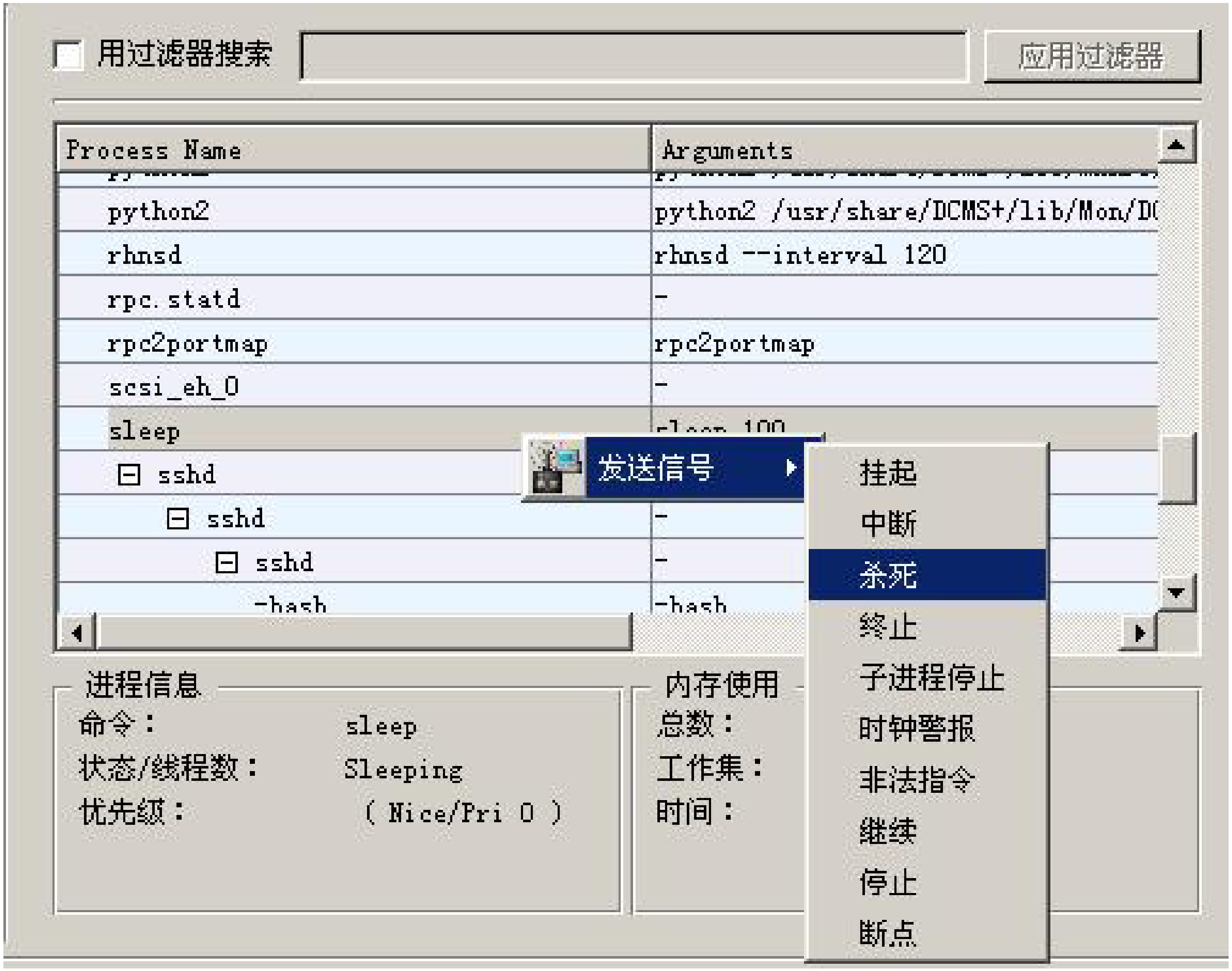
- 216-uml2 [ ]
- 216-uml3 [ ]
- clay-uml1 [ ]
- clay-uml2 [ ]
- clay-uml3 [ ]
- cabinet1 [ ]
  - ldj [✓]
  - server1 [ ]
  - server2 [ ]
  - server3 [ ]
  - server4 [ ]
  - server8 [✓]
- cabinet2 [ ]
  - server5 [ ]
  - server6 [ ]
  - server7 [✓]

Process Name	Arguments
⊟ Explorer	Explorer
cmd	cmd
cmd	cmd
daemon	daemon
mmc	mmc
pythonw	pythonw
⊟ Explorer	Explorer
cmd	cmd
internat	internat
mmc	mmc

Process Info
Command:
Status/Thrd No:
Priority:

Memory Usage
Total:
RSS Memory:
Time:

Screenshot—05(Process)



用过滤器搜索 应用过滤器

Process Name	Arguments
python2	python2 /usr/share/DCMS+/lib/Mon/DC
rhnsd	rhnsd --interval 120
rpc.statd	-
rpc2portmap	rpc2portmap
scsi_eh_0	-
sleep	sleep 100
⊟ sshd	
⊟ sshd	-
⊟ sshd	-
-bash	-bash

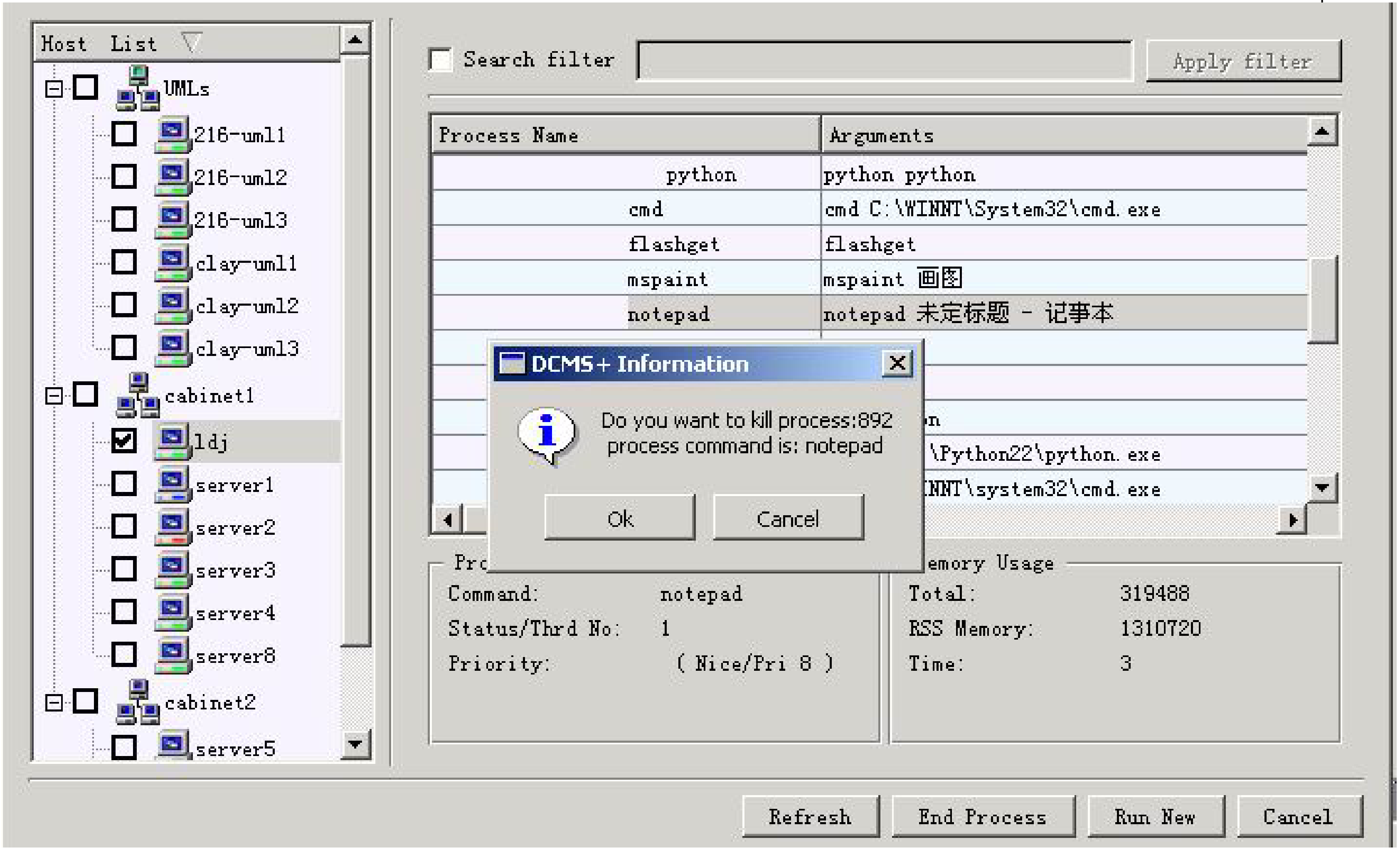
进程信息

命令：	sleep
状态/线程数：	Sleeping
优先级：	(Nice/Pri 0)

内存使用

总数：
工作集：
时间：

Screenshot—06(Process)



Process Name	Arguments
python	python python
cmd	cmd C:\WINNT\System32\cmd.exe
flashget	flashget
mspaint	mspaint 画图
notepad	notepad 未定标题 - 记事本
	n
	\Python22\python.exe
	INNT\system32\cmd.exe

DCMS+ Information

Do you want to kill process:892
process command is: notepad

Ok Cancel

Pro

Command: notepad
Status/Thrd No: 1
Priority: (Nice/Pri 8)

emory Usage

Total: 319488
RSS Memory: 1310720
Time: 3

Links

- Python
  - *http://www.python.org/*
- Python Books
  - file:// 10.20.10.179/ebook/-python-/
  - *GUI Programming with Python: QT Edition*
- Qt
  - *www.trolltech.com*
  - *Qt Reference*
- Others
  - *XML-RPC for Python*

Qt Vedio

- Overview
- Signals and Slots
- Database