**PYGTK-LinuxCNC笔记**

目录

[一、MINICNC 2](#_Toc477685974)

[1.三菱机床操作步骤 2](#_Toc477685975)

[2.机床数据 2](#_Toc477685976)

[3.G代码命令 3](#_Toc477685977)

[二、Glade 3](#_Toc477685978)

[三、Gtk 4](#_Toc477685979)

[1.重要函数 4](#_Toc477685980)

[2.部件颜色 5](#_Toc477685981)

[3.部件参数 7](#_Toc477685982)

[4.六个导入文件 7](#_Toc477685983)

[5.LED灯 8](#_Toc477685984)

[6.command指令 8](#_Toc477685985)

[7.图片设置 10](#_Toc477685986)

[8.GTK语句 11](#_Toc477685987)

[9.主题 11](#_Toc477685988)

[10.linux.stat参数 11](#_Toc477685989)

[四、Python 12](#_Toc477685990)

[1.语法 12](#_Toc477685991)

[2.os库 12](#_Toc477685992)

[3.python登录mysql数据库 13](#_Toc477685993)

[4.python读写文件 14](#_Toc477685994)

[5.python数组 16](#_Toc477685995)

[五、HAL 18](#_Toc477685996)

[1.文件结构 18](#_Toc477685997)

[六、更新 19](#_Toc477685998)

[11.27 19](#_Toc477685999)

一、MINICNC

1.三菱机床操作步骤

<1>启动机床，Home模式下，依次在X,Y,Z档点击“+”键归零，此时机床主轴位于机械零点。

<2>切换至MDI（Manual data input）模式，点击搜索，添加G代码文件。

<3>调整主轴到适当位置，开启主轴正转，将模式切换到Manual X1/X10/X100　确定G54原点。

<4>写入G54，（Z一般使用补正量），设置补正量。

<5>切换到自动模式，运行。

2.机床数据

▲机床X,Y轴：20mm一圈

Z轴：16mm一圈

3.G代码命令

▲G54~G59表示坐标系1~6

G90——绝对坐标

G91——相对坐标

G00——快速移动

G28——原点回归

二、Glade

1.对话框属性

公共－敏感

TRUE：对话框弹出后，其他面板无法操作

FALSE：对话框弹出后，不影响其他面板操作，但自身无法操作

2.常规－默认高度／宽度：需打钩才能生效

3.打开ｄｉａｌｏｇ框：

self.response = self.dialog.run()

4.Linuxcnc中，ini文件为主文件，其中包含有py文件，hal文件路径，各种参数等

python文件包含glade文件路径及库文件路径

5.安全常识，机床回零需先回Z轴

1. 调整部件大小，在格局处用请求高宽
2. Spin Button类滑动按钮需添加adj控件，用value change信号响应改变
3. Label底色改变需用evenbox

字体颜色和字体特性改变在label特性中

Button背景色改变直接在代码中，字体等需添加label再改变label中字体

1. Gremlin文件：/usr/share/pyshared/gremlin.py

修改字体，在该文件中搜索font

三、Gtk

1.重要函数

▲周期函数：gobject.timeout\_add(100,self.\_periodic)

每100ms执行一次\_periodic函数

▲获取部件label：

dir = widget.get\_label()[0]

变量dir为本部件标签的第一个字符

▲self.stat = linuxcnc.stat()

G代码给定进给：self.stat.settings[1]

G代码给定主轴转速：self.stat.settings[2]

▲if widget == self.widgets.btn\_set\_value\_x:

若本部件为btn\_set……

▲添加Dro部件后编辑glade会卡顿，可以先将Dro部件剪切至另一个窗口

▲部件敏感度设置

Self.widgets.tbtn\_on.set\_sensitive(True/False)

▲部件隐藏：self.widgets.label50.hide()

部件显示：self.widgets.label50.show()

▲发送MDI命令：

self.widgets.hal\_mdihistory.entry.set\_text(“g28”)

提交命令：

self.widgets.hal\_mdihistory.submit()

2.部件颜色

▲self.widgets.tbtn\_on.modify\_bg

(gtk.STATE\_ACTIVE,gtk.gdk.color\_parse(“#FFFF00”))

▲self.widgets.btn\_axischoose.modify\_bg

(gtk.STATE\_NARMAL,gtk.gdk.Color(0,65535,0))

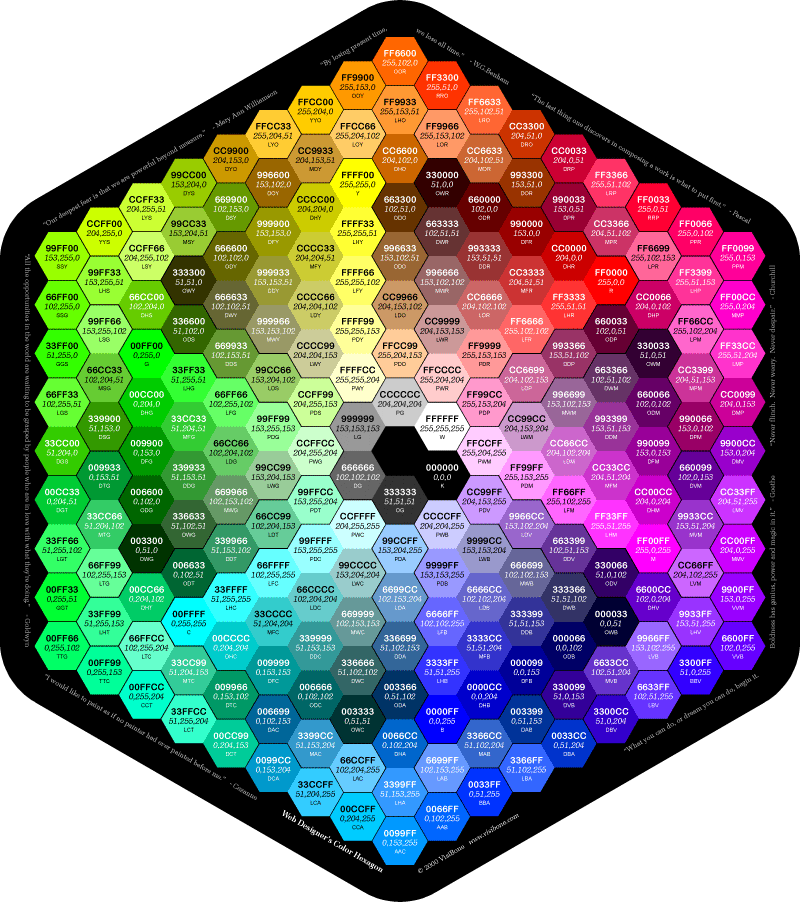
▲模式

modify\_

fg/text/bg/base

gtk.STATE\_

NORMAL/INSENSITIVE/PRELIGHT/ACTIVE/SELECTED



颜色代码图

3.部件参数

self.widgets.spindle\_speed\_set.set\_text(str(300))

set\_text属性只能传递字符串

self.widgets.lbl\_time.set\_label(strftime(“%H:%M:%S”)+“/n”+strftime(“%Y.%m.%d”))

set\_label属性只能传递字符串

self.widgets.rbtn\_spindlespeed100.set\_active(True)

设置该部件为激活状态

4.六个导入文件

位置:（/usr/lib/pymodules/python2.7/gmoccapy/）

widgets :处理窗口部件

player :处理声音

notification :处理错误

preference :处理参数

getiniinfo :阅读与检查ini文件

dialogs :获取所有对话框代码

eg：from gmoccapy import widgets

gmoccapy目录中还有一个\_init\_.py空文件

5.LED灯

self.widgets.led1.set\_property(“is on”,bool(True))

常规中：

Blink when off :True 指is on是off时闪烁

False 指is on是on时闪烁

Blinkrate控制是否闪烁

6.command指令

self.command = linuxcnc.command()

self.command.mode(linuxcnc.MODE\_MANUAL)

进入手动模式

self.command.mode(linuxcnc.MODE\_AUTO)

进入自动模式

self.command.mode(linuxcnc.MODE\_MDI)

进入命令模式

self.command.wait\_complete()

等待上一步完成

self.command.home(-1)

所有轴homeall

self.command.jog(linuxcnc.JOG\_CONTINUOUS,选轴,方向\*速度)

命令：轴持续移动

self.command.jog(linuxcnc.JOG\_INCREMENT,选轴,方向\*速度,步长)

命令：轴以步长移动

self.command.jog(linuxcnc.JOG\_STOP,选轴)

命令：轴停止移动

self.command.state(linuxcnc.STATE\_ESTOP\_RESET)

进入重置状态

self.command.state(linuxcnc.STATE\_ESTOP)

进入急停状态

self.command.state(linuxcnc.STATE\_ON)

运行

self.command.state(linuxcnc.STATE\_OFF)

停止

self.command.feedrate(self.feed\_override)

进给率

self.command.spindle(0)

主轴停止

self.command.spindle(1,rpm)

主轴正转，转速为rpm

self.command.spindle(-1,rpm)

主轴反转，转速为rpm

self.command.spindleoverride(self.spindle\_override)

主轴转率

self.command.auto(linuxcnc.AUTO\_RUN,self.start\_line)

再搜索：自动从self.start\_line处开始运行G代码

self.command.mdi(system)

运行“system”mdi命令

7.图片设置

方法1：

在glade中添加image组件

修改名称：

修改文件：

（路径为相对于glade的路径名）

然后在python文件中调用该组件：

Self.widgets.tbtn\_on.set\_image(self.widgets.img\_zoom\_in4)

方法2：

直接从路径中提取文件

self.widgets.image1.set\_from\_file(“/usr/share/……”)

8.GTK语句

▲若该组件已激活：

if widget.get\_active():

▲if逻辑：

if ( flg1<0 ) or ( flg2<0 ) or ( flg3<0 ):

▲批量修改部件状态

定义部件组：

widgetlist = [“tbtn\_search”,”tbtn\_research”…]

添加：widgetlist.append(“tbtn\_search”)

删减：widgetlist.remove(“tbtn\_search”)

批量改变：

for i in widgetlist:

self.widgets[i].set\_active/sensitive(False)

▲x = [4,6,2,1,7,9]

y = sorted(x) #由小到大排序

x.sort() #排序，仍在x中

9.主题

主题相关文件在：/usr/share/themes/

10.linux.stat参数

网址：

<http://blog.csdn.net/linuxarmsummary/article/details/46636125>

四、Python

1.语法

▲python行首不能有空格

▲ubuntu下，朋友文件先转化为.pyc文件（字节码文件）再转化为机器语言

▲python中字符串拼接直接用“+”号，但是效率较低

可用join方法

▲list列表用append（）拼接

▲exec方法，执行字符串

eg：exec（‘self.mode\_choose = %d’%(self.mode\_choose)）

2.os库

os.path.abspath(path) #返回绝对路径

os.path.basename(path) #返回文件名

os.path.commonprefix(list) #返回list(多个路径)中，所有path共有的最长的路径。

os.path.dirname(path) #返回文件路径

os.path.exists(path)  #路径存在则返回True,路径损坏返回False

os.path.lexists  #路径存在则返回True,路径损坏也返回True

os.path.expanduser(path)  #把path中包含的"~"和"~user"转换成用户目录

os.path.expandvars(path)  #根据环境变量的值替换path中包含的”$name”和”${name}”

os.path.getatime(path)  #返回最后一次进入此path的时间。

os.path.getmtime(path)  #返回在此path下最后一次修改的时间。

os.path.getctime(path)  #返回path的大小

os.path.getsize(path)  #返回文件大小，如果文件不存在就返回错误

os.path.isabs(path)  #判断是否为绝对路径

os.path.isfile(path)  #判断路径是否为文件

os.path.isdir(path)  #判断路径是否为目录

os.path.islink(path)  #判断路径是否为链接

os.path.ismount(path)  #判断路径是否为挂载点（）

os.path.join(path1[, path2[, ...]])  #把目录和文件名合成一个路径

os.path.normcase(path)  #转换path的大小写和斜杠

os.path.normpath(path)  #规范path字符串形式

os.path.realpath(path)  #返回path的真实路径

os.path.relpath(path[, start])  #从start开始计算相对路径

os.path.samefile(path1, path2)  #判断目录或文件是否相同

os.path.sameopenfile(fp1, fp2)  #判断fp1和fp2是否指向同一文件

os.path.samestat(stat1, stat2)  #判断stat tuple stat1和stat2是否指向同一个文件

os.path.split(path)  #把路径分割成dirname和basename，返回一个元组

os.path.splitdrive(p00000000000000000000000000000000000000000000000000000000000000ath)   #一般用在windows下，返回驱动器名和路径组成的元组

os.path.splitext(path)  #分割路径，返回路径名和文件扩展名的元组

os.path.splitunc(path)  #把路径分割为加载点与文件

os.path.walk(path, visit, arg)  #遍历path，进入每个目录都调用visit函数，visit函数必须有3个参数(arg, dirname, names)，dirname表示当前目录的目录名，names代表当前目录下的所有文件名，args则为walk的第三个参数

os.path.supports\_unicode\_filenames  #设置是否支持unicode路径名

3.python登录mysql数据库

**登录mysql数据库:**

conn = MySQLdb.Connect(host = '127.0.0.1', port = 3306, user = 'root', passwd = '14301430', db = 'imooc', charset = 'utf8')

**将conn传入集中:**

app = gui1002(conn)

**传入初始化函数，并定义为self全局变量：**

def \_\_init\_\_(self, conn):

self.conn = conn

**使用cursor工具：**

cursor = self.conn.cursor()

**执行命令语句：**

sql = "update account set money=money+%s where acctid=%s" % (money, acctid)

cursor.execute(sql)

**提交事件：**

self.conn.commit()

**断开连接：**

cursor.close()

conn.close()

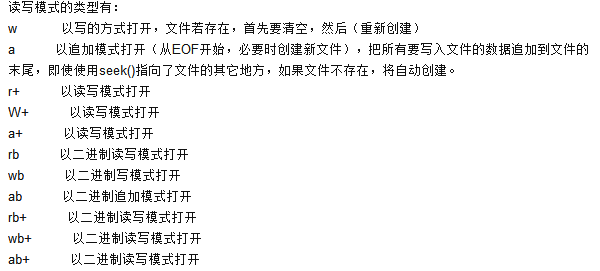
4.python读写文件

▲python打开文件方式：

file\_object = open(‘file\_path’,’选项’)

默认为读文本模式

选项：



▲每次操作完文件必须关闭!

file\_object.close()

▲read方法：

file\_object.read(x)

读取文件x长度，从标记开始

file\_object.readline(x)

读取文件一行内x长度，从标记开始

file\_object.readlines()[x][y]

将文件每行拆分，读取文件第x行，第y个字节

▲write方法：

file\_object.write(str)

将str写入文件中，不会自动换行

file\_object.write(seq)

将seq整体写入文件中，不会自动换行

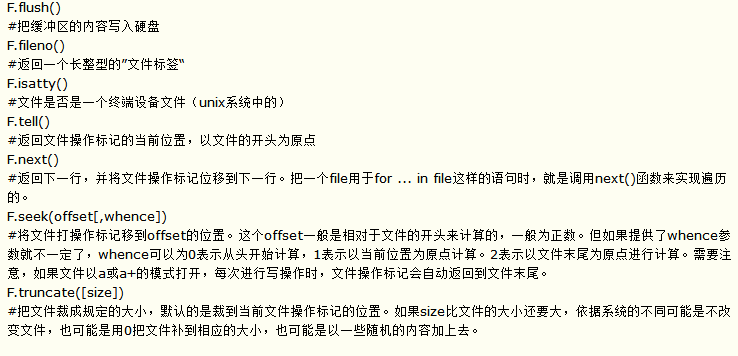
▲seek方法：

file\_object.seek(offset,whence)

offset为读写指针位置

whence可选

默认为0，表示绝对的文件定位；1表示相对于当前位置；2表示相对于文件的末尾

▲

5.python数组

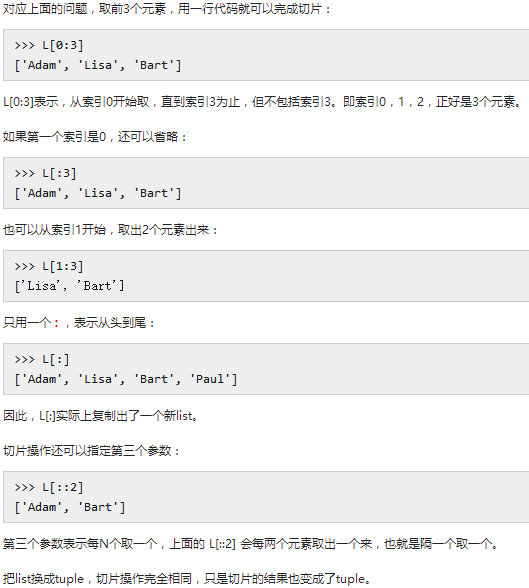
▲python数组分为三种类型：

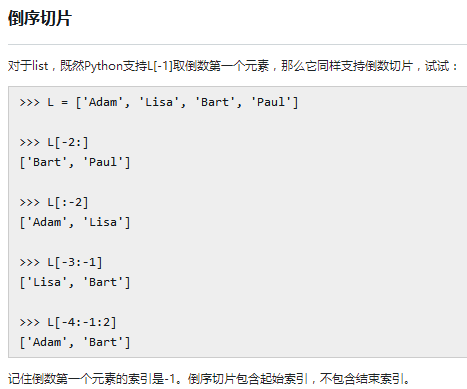
（1）list普通的链表，初始化后可以通过特定方法动态增加元素。  
定义方式：arr = [元素]

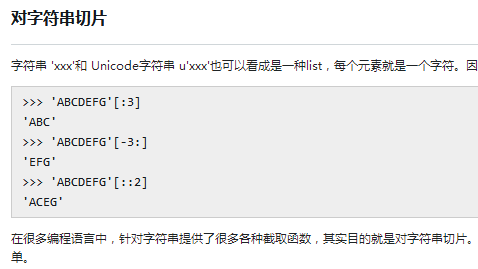
（2）Tuple 固定的数组，一旦定义后，其元素个数是不能再改变的。  
定义方式：arr = (元素)

（3） Dictionary 词典类型， 即是Hash数组。  
定义方式：arr = {元素k:v}

▲数组切片：







五、HAL

1.文件结构

CNC访问configs下GUI2目录中的GUI2.ini文件

ini文件包含路径：

python文件（/usr/bin/GUI2.py）

hal文件（/home/家目录/linuxcnc/configs/GUI2/）

python文件：

导入库文件

6个函数文件（/usr/lib/pymodules/python2.7/GUI2）

GUI目录（/usr/share/GUI2/）：GUI2.glade、images、DigitSave

六、CNC原理

1.刀具切换

(1)当进行刀具切换执行指令：TX M6

(2)主轴先移动至刀具切换位置：TOOL\_CHANGE\_POSITION

(3)系统判断当前刀具号是否为X

如果为X则继续执行指令

如果不为X则说明需要切换刀具,执行(4)步骤

(4)HAL虚拟引脚iocontrol.0.tool-change向halcomp:GUI2新建的GUI2.toolchange-change引脚传递True值

(5)GUI2.toolchange-change从False变成True发生 value\_changed,触发执行函数on\_tool\_change（）

(6) on\_tool\_change（）函数：

获取self.halcomp['toolchange-change']

获取self.halcomp['toolchange-number']

当self.halcomp['toolchange-change']==True

若刀具号self.halcomp['toolchange-number']不为0

弹出选择框显示将要切换的刀具信息

(7)当点击确认

self.halcomp["toolchange-changed"] = True

on\_tool\_change（）函数执行完毕

随后

self.halcomp['toolchange-change'] = False

value\_changed执行on\_tool\_change（）函数

self.halcomp["toolchange-changed"] = False

(8)当点击取消，中止指令，刀具号仍为原刀具

self.halcomp['toolchange-change'] = False

self.halcomp["toolchange-changed"] = True

弹窗通知

self.halcomp["toolchange-changed"] = False