**Linux桌面测试系统**

**\_开发向导手册**

**文档管理信息表**

|  |  |
| --- | --- |
| **主题** |  |
| **版本** |  |
| **内容** |  |
| **关键字** |  |
| **参考文档** |  |
| **创建时间** |  |
| **创建人** |  |
| **最新发布日期** |  |

**文档变更纪录**

|  |  |  |  |
| --- | --- | --- | --- |
| **更改人** | **日期** | **版本** | **更改内容** |
| 卢兆涵 | 2015/4/16 | V0.01 | 创建文件 |
| 卢兆涵 | 2015/4/23 | V0.02 | 修改log接口; 添加CaseObject获取用户密码方法 |
|  |  |  |  |
|  |  |  |  |

**目 录**

## 基本设计描述

基于LDTP（Linux Desktop Test Project）开发的Linux桌面系统设计自动化测试系统，该系统可安装在当前的Linux系统中，用户手动设置配置文件后，自动完成测试。

## 系统总体逻辑结构

### 系统模块划分

系统划分为两个部分：测试公用库和测试用例模块。

#### 2.1.1 测试公用库

包含开发测试脚本所需的通用功能和一些示例代码，其中：

* 通用功能

测试用例模板（caseobject.py）：提供一个包含了测试用例常用接口的测试用例模板类CaseObject，新用例脚本可通过两种方法：继承该类，或创建新实例后调用的方法引用其中的功能。

模块模板（module.py）：提供一个包含了模块用例常用接口的模块模板类ModuleObject，新模块可通过两种方法：继承该类，或创建新实例后调用的方法引用其中的功能。

日志（logcase.py）：提供测试用例所需的日志信息打印接口。

屏幕截图（screenshot.py）：提供测试用例所需全屏截图的接口。

* 示例代码

包含一些在开发过程中常用的示例代码，如多进程开发等。

#### 2.1.2 测试用例模块

测试用例模块是按照测试需求，将所有的测试用例划分为多个多层的模块，包含功能测试模块、性能测试模块、用户体验模块等。各模块按照目录顺序依次调用，直至最底层的测试用例文件。

### 工程目录结构

#### 2.2.1 模块目录结构

将所有的测试用例分组，呈现树状结构，如图所示。根节点为入口，树上的每个节点代表的是一个分组，叶子节点为具体的测试用例。

DesktopAutotest

Public Function

Case Template

Module Template

Screenshot

Log

Functional Test

Printer

Printer Case1

Printer Case2

...

Scanner

Video Card

Harddisk

Memory

...

User Experience Test

Update Manager

Software Manager

...

Property Test

每个模块和测试用例节点都包含两种文件：

* 配置文件（XML文件）

包含测试所需的配置。常见配置如：操作系统分类，执行次数，环境变量等参数。

* 可执行文件

一般采用脚本文件（python或shell），用于分析输入，根据用户配置执行输入文件。

与模块相关的文件放在该模块下，例如，Function Test对应的目录名为“function”，该目录下的function.py和function.xml分别是功能测试模块的可执行文件和配置文件。

#### 3.2.2 测试用例目录结构

##### 3.2.2.1 测试用例运行文件存放

与测试用例运行相关的所有文件（包含脚本文件和配置文件）都放在测试用例文件夹下。

例如，显卡对应的目录名为“video”，该目录下存在名为“videocase1”的目录（用例名）。其中，videocase1.py和videocase1.xml分别是显卡模块测试用例1对应的可执行文件和配置文件。video目录下存在的python脚本文件仅有video.py。

##### 3.2.2.2 测试用例输入输出文件存放

对于每个测试用例，如有与该用例相关的文件，例如需要读取的图片或压缩包，或该用例运行过程中产生的截屏文件，都放在该用例所在模块下，一个与该用例同名的文件夹中。

例如，测试用例case1在module1下，则测试用例1的截屏文件会放在module1/case1/screenshot中。

通用的目录结构

* resource：该测试用例运行所需的基础文件，例如运行所需的shell脚本文件，读取的图片、工具压缩包等。
* screenshot：该测试用例的截屏文件。
* output：输出文件。

### 系统运行主要流程

开始

读入脚本

解析脚本

设置环境变量

设置循环次数

为下一节点传入参数

循环结束？

结束

## 测试用例开发

### 用例名

对于每个用例，需要提供一个系统中的唯一命名（建议命名中添加测试用例编号以便区分，例如“3DGraphicProcess\_1\_3\_2\_10”，1\_3\_2\_10表示对应于测试用例编号1.3.2.10）。

该用例的配置文件（.xml），测试用例脚本文件（.py）和测试用例数据来源文件夹（该文件夹在测试用例所在的模块下）都以该名称命名，便于后续配置。

例如，case1对应的配置文件为case1.xml，脚本文件case1.py，数据来源文件夹为case1

### 用例配置文件

对应于每个测试用例的配置文件，用于存储测试用例的配置信息、数据来源等。

测试用例的XML文件包含一些默认的通用配置标签和通用属性，这些标签的名称在所有用例中相同。其他的可以按照XML格式自定义。

#### 3.2.1 配置文件类型

xml文件

#### 3.2.2 通用设置

##### 3.2.2.1 通用标签名

* 配置主标签：<config>
* 测试用例名：<casename>，例如“2D图像处理能力”，“视频处理能力”等。该名称将作为测试用例的备注名称提供给用户，便于用户识别和查找该测试用例。
* 用例脚本配置参数标签：<common>
* 用例脚本编号：<casenum>，例如<casenum>1.3.2.1</casenum>
* 操作系统配置：<system>
* 用例循环次数标签：<count>
* 用例循环时间标签：<exectime>
* 测试用例数据主标签：<data>

##### 3.2.2.2 通用属性名

* 标签类型属性：type

该属性用于在解析XML文件时取值。

目前包含的类型：

int 整型，标识标签中的内容是一个整数

intarray 整型数组，标识标签中的内容是一个整数数组，内容以英文半角逗号“,”分隔开。

[示例] <order>1,3,5,7</order> 表示整型数组{1,3,5,7}

string 字符串，标识标签中的内容是一个字符串

time 时间，标识标签中的内容是一个时间串，现在可加入的是时（h）、分（M）、秒（S），例如，10h30M4S表示10小时30分钟4秒，5M表示5分钟。如用通用结构getXMLNodeValue（请参见接口定义）获取的值是以秒为单位的一个整数。例如，5M返回值是300

* 标签中文名属性：tagname

该属性注明该标签的名称，用于后续统一解析时获取各标签的内容。

### 用例需求分析

请与QA沟通该测试用例的使用习惯，如预设文件内容，测试效果呈现等。

## 通用接口

### 测试用例模板类

#### 4.1.1 文件

caseobject.py

#### 4.1.2 描述

该模块包含一个公用类CaseObject，提供测试用例以及测试模块的通用函数。

#### 4.1.3 集成和部署

caseobject使用python语言编写的模块，使用时导入模块中的类。

[示例]

方法一：单独运行某个测试脚本时

import sys #python模块sys

sys.path.append(‘/home/Lenovo/test’) #’ /home/Lenovo/test’是caseobject.py所在的文件夹目录

方法二：集成运行系统环境时

g\_publicPath = os.environ[‘AUTOTEST\_PUBLIC\_LIB’] # AUTOTEST\_PUBLIC\_LIB是运行系统主文件目录desktop.py时导入的

sys.path.append(g\_publicPath)

#### 4.1.4 接口定义

* 获取文档元素

**getXMLDocument()**

<函数描述>

取出XML文档元素，相当于整个XML文件的根节点。

<输入参数>

空

<输出值>

正确：返回一个不为空的XML文档节点；

错误：失败时输出错误信息，返回None。

* 获取节点列表

**getXMLNodeList(parentNode, childNodeName)**

<函数描述>

根据XML文件的结构，取出对应于某个节点的子节点列表。例如，在<data>中，可能有多个文件链接标签<path>，返回的是一个包含多个path标签的列表。

<输入参数>

parentNode-父节点，是一个XML node类型的变量

childNodeName-子节点名，是一个字符串

<输出值>

正确：返回取出的子节点列表

错误：返回空列表[]

* 获取节点

**getXMLNode(parentNode, childNodeName, childNodeIndex)**

<函数描述>

根据XML文件的结构，取出对应于某个节点的子节点

<输入参数>

parentNode-父节点，是一个XML node类型的变量

childNodeName-子节点名，是一个字符串

childNodeIndex-子节点编号，取的是第几个子节点

<输出值>

返回取出的子节点，成功返回一个不为空的节点。

* 获取节点值

**getXMLNodeValue(node, childNodeIndex)**

<函数描述>

根据XML文件的结构，取出对应于某个节点编号为childeNodeIndex的子节点，并根据子节点的类型属性“type”返回对应类型的值。

属性type定义如下：

int：整型

intarray：整型数组

string：字符串

time：时间，定义为一个格式化字符串，包含时（h），分（M）和秒（S）。例如，“10h30M4S”表示10小时30分钟4秒

[示例]含有属性的标签定义，此时，tag为自定义标签，type和tagname是tag标签的属性，type是值的类型，tagname是该tag的名称

<tag type=”int” tagname=”临时标签”></tag>

<输入参数>

node-需要取值的节点

childNodeIndex-子节点编号，取的是第几个子节点

注意：由XML文件特决定，标签为tag的节点中，子节点0是文本Text。因此，如需取值，childNodeIndex设为0

<输出值>

成功时返回取出的节点取值，失败时抛出错误。

* 获取操作系统类型

**getOSName()**

<函数描述>

返回用于标识当前操作系统的字符串，可用于判断当前用例或模块运行所在的操作系统。

<输入参数>

无

<输出值>

返回操作系统名称。当前包含的操作系统对应如下：

|  |  |
| --- | --- |
| **操作系统名称** | **返回字符串** |
| Ubuntu | ubuntu |
| 普华 | isoft |
| 方德 | cos |
| 中标麒麟 | kylin |

* 获取当前用户的密码

**getPasswd ()**

<函数描述>

在配置好desktop.xml的用户信息(user标签)后。根据当前操作系统的用户，返回对应的密码。

[示例]在系统主文件夹下desktop.xml文件的common标签内添加用户信息。例如，

…

<common>

<user> <!--一个用户的信息- ->

<username type=”string” tagname=”用户名”>Lenovo</username>

<passwd type=”string” tagname=”密码”>123456</passwd>

</user>

</common>

…

<输入参数>

无

<输出值>

返回操作系统名称。当前包含的操作系统对应如下：

* 按照时间循环运行用例

**runInExecTime (execFunction, timeout)**

<函数描述>

按时间循环执行函数，到时退出。

[示例] 用例将需要执行的函数写在一个函数中，例如，

def func():

func1()

func2()

调用函数并设定超时值，

runInExecTime(func, 3)

则函数序列func()循环执行，3秒后停止。

<输入参数>

execFunction-需要按时间循环执行的函数

timeout-需要定时退出的时间，不能为0

<输出值>

无

* 按照次数循环运行用例

**runInCount(execFunction, count)**

<函数描述>

按给出的次数循环执行函数。

[示例] 用例将需要执行的函数写在一个函数中，例如，

def func():

func1()

func2()

调用函数并设定次数

runInCount (func, 3)

则函数序列func()循环执行，3次后停止。

<输入参数>

execFunction-需要循环执行的函数

count-需要循环的次数

<输出值>

无

* 禁用/使能鼠标和键盘

**enableMouseKeyboard(enableFlag)**

<函数描述>

根据用户输入，禁用或使能当前机器的键盘、鼠标，防止误操作。

[示例] 禁用鼠标、键盘

enableMouseKeyboard(False)

<输入参数>

enableFlag-布尔值True/False

<输出值>

操作正确，返回True，否则返回False

### 日志打印

#### 4.2.1 文件

logcase.py

#### 4.2.2 描述

该模块包含一个公用类Logcase，提供测试用例以及测试模块的消息打印。消息分为三个等级：

* 错误（ERROR）
* 警告（WARNING）
* 信息（INFO）：运行过程中运行到的步骤，运行提示等信息。

#### 4.2.3 集成和部署

logcase.py是使用python语言编写的模块，使用时导入模块中的类。

[示例]

方法一：单独运行某个测试脚本时

import sys #python模块sys

sys.path.append(‘/home/Lenovo/test’) #’ /home/Lenovo/test’是caseobject.py所在的文件夹目录

方法二：集成运行系统环境时

g\_publicPath = os.environ[‘AUTOTEST\_PUBLIC\_LIB’] # AUTOTEST\_PUBLIC\_LIB是运行系统主文件目录desktop.py时导入的

sys.path.append(g\_publicPath)

#### 4.2.4 日志文件位置

./desktop/log目录下：

* 错误（ERROR）：error.log
* 警告（WARNING）：warn.log
* 信息（INFO）：info.log

#### 4.2.5 接口定义

* 打印错误信息

**elog(tag,msg)**

<函数描述>

打印包含标签tag和信息msg的错误信息。

<输入参数>

tag：字符串类型，一般为测试用例的名称，用于标注是哪个用例发起的错误

msg：用户自定义希望出现在log中的信息，一般包含在用例中执行到的步骤，发出该错误信息的时机等。

<输出值>

输出错误信息字符串到error.log文件中。

* 打印警告信息

**wlog(tag,msg)**

<函数描述>

打印包含标签tag和信息msg的警告信息。

<输入参数>

tag：字符串类型，一般为测试用例的名称，用于标注是哪个用例发起的

msg：用户自定义希望出现在log中的信息，一般包含在用例中执行到的步骤，发出该信息的时机等。

<输出值>

输出警告信息字符串到warn.log文件中。

* 打印提示信息

**ilog(tag,msg)**

<函数描述>

打印包含标签tag和信息msg的调试信息。

<输入参数>

tag：字符串类型，一般为测试用例的名称，用于标注是哪个用例发起的信息

msg：用户自定义希望出现在log中的信息，一般包含在用例中执行到的步骤，发出该信息的时机等。

<输出值>

输出错误信息字符串到info.log文件中。

* 获取节点列表

**getXMLNodeList(parentNode, childNodeName)**

<函数描述>

根据XML文件的结构，取出对应于某个节点的子节点列表。例如，在<data>中，可能有多个文件链接标签<path>，返回的是一个包含多个path标签的列表。

<输入参数>

parentNode-父节点，是一个XML node类型的变量

childNodeName-子节点名，是一个字符串

<输出值>

正确：返回取出的子节点列表

错误：返回空列表[]

### 截屏

#### 4.3.1 文件

screenshot.py

#### 4.3.2 描述

该模块包含一个公用类Screenshot，提供测试用例以及测试模块截屏的函数。

#### 4.3.3 集成和部署

screenshot使用python语言编写的模块，使用时导入模块中的类。

[示例]

方法一：单独运行某个测试脚本时

import sys #python模块sys

sys.path.append(‘/home/Lenovo/test’) #’ /home/Lenovo/test’是caseobject.py所在的文件夹目录

方法二：集成运行系统环境时

g\_publicPath = os.environ[‘AUTOTEST\_PUBLIC\_LIB’] # AUTOTEST\_PUBLIC\_LIB是运行系统主文件目录desktop.py时导入的

sys.path.append(g\_publicPath)

#### 4.3.4 接口定义

* 截屏函数

**scrprint(caseName, fileName, casePath)**

<函数描述>

取当前系统截屏，加入时间戳后存储到用例对应的文件夹中

<输入参数>

caseName-用例名

fileName-截屏图片文件的名称

casePath-用例脚本文件所在的路径

<输出值>

将全局截屏图片存储到文件夹casePath/caseName/screenshot中。例如，对于位于module1的case1，存储的图片在module1/case1/screenshot文件夹下

### 环境变量

提供两个系统使用的环境变量。

AUTOTEST\_\_DIR： 自动测试系统根目录，例如，“/home/Lenovo/desktop”

AUTOTEST\_PUBLIC\_LIB：自动测试系统开发所需的公共库所在目录，例如，“/home/Lenovo/desktop/public”

## 测试用例运行

在对应的模块XML文件中添加case所需的脚本和数据文件。例如，video.xml中，picture用例的配置。

<data>

<case>

<script type=”string” typename=”用例脚本”>picture.py</script>

<caseConfig type=”string” typename=”用例配置”>picture.py</caseConfig>

</case>

<!- - case2, case3 - - !>

</data>