1. 系统架构

本视频播放器基于 Python3.12 编写，使用：

- **Tkinter 库作为 GUI 界面库**
- **Vlc 库处理多媒体任务**
- **Os 库处理文件路径**
- **Datetime 转换时间**
- **Cv2 读取视频时长**
- **PIL 处理图片**

整个系统主要由以下几个部分组成：

- **主窗口**：包含所有组件，如按钮、进度条和播放列表等。
- **VLC 实例**：负责管理媒体播放器对象，控制视频的播放、暂停、停止等功能。
- **显示控件**：提供对播放器的基本操作。
- **右键菜单**：提供对播放器的基本操作，包含显示控件的大部分功能以及一些其他功能。

2. 方法详解

```
    def __init__(self, root):
        初始化播放器的各项参数，包括：主窗口的创建，组件 ICON 和背景图片的读取，组件的创建与初始化，事件菜单的初始化，播放参数的初始化

@staticmethod
    def get_duration(video_path):
    """文件时长获取"""
        使用 ffprobe 获取并返回视频时长

@staticmethod
    def get_unique_filename(output_path):
    """生成唯一文件名"""
        静态方法，调用 os 的方法分离文件名，然后根据路径是否存在相同文件名生成并返回唯一文件名

    def update_total_time(self, video_path):
    """获取视频时长"""
        调用 get_duration_from_cv2 获取时长并转换为(时:分:秒)

    def capture_frame(self, output_path):
    """保存视频截图"""
        调用 get_unique_filename 获取唯一文件名并返回给 vlc 方法 video_take_snapshot()保存截图

    def switch_background(self):
    """背景图片切换"""
        根据 self.bg 的状态 self.canvas.remove()或 self.canvas.grid()

    def exit_app(self):
    """右键菜单退出"""
        弹出消息框确认，确认后释放资源

    def toggle_fullscreen(self):
    """切换全屏模式"""
        根据 self.not_fullscreen 的状态调用 tk 库 attributes()

    def set_playback_rate(self, rate):
    """设置播放速率"""
        根据传参 rate 调用 vlc 方法 set_rate()，并禁用或恢复相关的按钮
```

```
def play_modes_switch(self):
"""播放模式的按键循环"""
    1234 四种模式，按 12341234 的顺序循环，并让 ICON 保持更新

def play_modes_set(self, mode):
"""播放模式选择"""
    根据传参 mode 改变播放参数 self.play_mode，并保持按键状态与 ICON 的更新

def play_modes(self):
"""播放模式切换"""
    根据 self.play_mode 对播放模式进行匹配

def on_mousewheel(self, event):
"""滚轮音量调整"""
    根据滚轮事件判定方向，对音量进行加减，减为 0 时判断，保证 ICON 的更新

def update_volume_icon(self):
"""音量图标切换"""
    根据判断 self.volume_level 是否为零，更新 ICON

def switch_volume(self):
"""一键静音与恢复"""
    根据判断 self.volume_level 是否为零，记忆原音量并切换

def set_volume(self, volume):
"""音量大小调整"""
    通过 vlc 的 audio_set_volume()方法改变音量，并改变滑块位置与 ICON 显示

def show_popup_menu1(self, event):
"""显示右键菜单"""
    右击窗口内展开右键菜单

def show_popup_menu_del(self, event):
"""显示双击菜单"""
    双击列表展开双击菜单

def show_rate_menu(self, event):
"""显示播放速率菜单"""
    读取鼠标事件位置调用 tk 的 tk_popup()并释放

def resize_image(self, event=None):
"""背景大小自适应"""
```

```
        通过 PIL 的 resize()改变背景大小

    def load_videos(self):
    """选择视频加载"""
        加载选中的视频并判断重复，播放选中的第一个视频

    def play_next_video(self):
    """播放下一视频"""
        根据 self.current_index 是否越界播放下一视频或停止播放

    def stop_video(self):
    """视频停止播放"""
        部分播放参数恢复初始化

    def load_and_play_video(self, video_path):
    """播放加载视频"""
        根据操作系统选择方法，部分部分参数初始化，使用 play()开始播放视频

    def get_current_index(self):
    """获取当前编号"""
        获取当前播放视频在列表中的编号

    def double_event(self, event):
    """鼠标双击事件"""
        用于双击切换全屏/窗口

    def toggle_play_pause(self):
    """暂停播放切换"""
        判断 self.is_playing 进行暂停播放切换

    def remove_for_background(self):
    """欣赏背景图片"""
        隐藏所有组件

    def toggle_to_switch(self):
    """组件显示切换"""
        根据 self.forget 隐藏不同组件

    def show_for_items(self):
    """组件显示"""
        包含所有组件的位置参数

    def update_time_display(self, num):
    """时间显示更新"""
```

```
        根据播放状态更新时间显示并判定是否播放完毕

    def update_name_display(self):
    """视频标题获取"""
        获取视频标题并显示在播放器上方

    def on_playlist_select(self, event):
    """播放列表选择"""
        根据鼠标事件选择视频进行播放
    def play_selected_video(self):
    """播放选中视频"""
        通过鼠标事件定位选择的视频并播放

    def delete_selected_video(self, *args):
    """删除选中视频"""
        通过鼠标事件定位选择的视频并删除

    def bind_progress_bar_events(self):
    """绑定进度条事件"""
        加载视频后绑定进度条鼠标事件

    def unbind_progress_bar_events(self):
    """解绑进度条事件"""
        停止视频后绑定进度条鼠标事件

    def on_leave(self, event):
    """预览窗隐藏"""
        鼠标离开进度条隐藏预览窗

    def update_progress_bar_position(self, event):
    """更新进度条"""
        根据进度条的鼠标事件更新进度条位置并显示预览窗

    def show_preview(self, event):
    """显示预览窗"""
        提供预览窗显示的延迟(500 毫秒)，加载预览窗内容，之后根据鼠标事件定位预览
    窗并显示

    def hide_preview(self):
    """隐藏预览窗"""
        停止预览窗的视频并隐藏(关闭)预览窗

    def create_preview_window(self, event):
    """创建预览窗"""
```

```
        创建一个新的窗口独立于主窗口，设置透明度、大小、显示内容，其上显示的预览
    画面用加载的新视频并暂停充当

    def capture_frames(self, interval=24):
    """读取预览画面"""
        加载视频时根据总帧加载对应位置的 24 张图作为预览画面

    def update_item(self):
    """更新按钮状态"""
        根据播放状态更新按钮状态

if __name__ == "__main__":
    root = tk.Tk()
    icon_image = PhotoImage(file='icon2.png') #主窗口的 ICON
    root.wm_iconphoto(False,icon_image)
    app = VideoPlayerApp(root)
    root.mainloop()
```