Rockchip Linux web开发基础

文件标识：RK-FB-YF-358

发布版本：V1.3.3

日期：2021-05-21

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档提供Linux应用开发基础说明。

产品版本

芯片名称	内核版本
RV1126, RV1109	Linux 4.19
RK1808, RK1806	Linux 4.4

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	Fenrir Lin	2020-04-28	初始版本
V1.1.0	Fenrir Lin	2020-06-04	增加ispserver和onvif_server部分
V1.1.1	CWW	2020-06-29	更新RK_OEM编译打包命令
V1.2.0	Allen Chen	2020-08-24	更新ipcweb-ng部分
V1.3.0	Fenrir Lin	2020-09-16	增加dbserver对外接口，更新onvif_server开发环境
V1.3.1	Fenrir Lin	2020-10-15	更新文件路径
V1.3.2	Ruby Zhang	2021-03-15	完善产品版本信息
V1.3.3	Fenrir Lin	2021-05-21	删除onvif_server部分

目录

1. ipcweb-ng

1.1 开发基础

web前端，采用Angular 8框架。

开发语言： Typescript，JavaScript，HTML5，SCSS

参考文档：

Angular官方入门教程
TypeScript中文网
w3school

代码路径： app/ipcweb-ng

编译命令：

```
# 初次使用需安装开发环境见3.2开发环境
# 在app/ipcweb-ng目录下
ng build --prod
# 将编译生成在app/ipcweb-ng/dist目录下的文件，都移动到app/ipcweb-backend/www路径下，如此操作后在编译ipcweb-backend的同时将会进行www的复制
# 在SDK根目录下
make ipcweb-backend-rebuild # 重新编译后端文件，并根据配置复制www到对应路径
make rk_oem-dirclean && make rk_oem target-finalize # 重新编译oem
./mkfirmware.sh # 打包oem.img,再进行烧写
```

常见编译冲突：

```
# 错误提示如下，为类型定义冲突
ERROR in ./src/app/shared/player/download.worker.ts (./node_modules/worker-plugin/dist/loader.js?{"name":"0"}!./src/app/shared/player/download.worker.ts)
Module build failed (from ./node_modules/worker-plugin/dist/loader.js):
# 解决修改node_modules\@types\emscripten\index.d.ts文件
# 冲突语句
```

```
declare function addFunction(func: () => any, signature?: string): number;
# 修改如下
declare function addFunction(func: Function, signature?: string): number;

# 错误提示如下，为别名冲突
Type alias 'PluginConfig' circularly reference
Type alias 'ProtractorPlugin' circularly reference
# 解决修改node_modules\protractor\built\index.d.ts文件
# 冲突语句
import { PluginConfig, ProtractorPlugin } from './plugins';
export declare type PluginConfig = PluginConfig;
export declare type ProtractorPlugin = ProtractorPlugin;
# 修改如下
import { PluginConfig as PluginCfg, ProtractorPlugin as ProtractorPlu} from
'./plugins';
export declare type PluginConfig = PluginCfg;
export declare type ProtractorPlugin = ProtractorPlu;
```

1.2 开发环境

```
# Ubuntu
sudo apt update
sudo apt install nodejs
sudo apt install npm
sudo npm install -g n # 安装 n 模块
sudo n stable # 用 n 模块升级
npm npm --version # 确认 npm 版本
sudo npm install -g @angular/cli # 安装 Angular 命令行工具
# 在app/ipcweb-ng目录下
sudo npm install # 安装 Angular 以及相关依赖库

# Windows
# https://nodejs.org/en/download/官网内下载对应Nodejs并安装
# 检查 npm node是否安装成功，若失败至官网下载最新版本安装
npm --version
node --version
npm install -g @angular/cli # 安装 Angular 命令行工具
# 在ipcweb-ng 目录下
npm install # 安装 Angular 以及相关依赖库
```

1.3 在线调试

启动webpack开发服务

```
ng serve
```

成功的话，可见以下log

```
** Angular Live Development Server is listening on 0.0.0.0:4200, open your
browser on http://localhost:4200/ **
```

随后使用chrome浏览器访问 http://localhost:4200/，即可在线调试。

也可使用`ng build --prod`命令编译，将生成在dist目录下的文件，推送到板端，替换oem/www或usr/www(根据产品类型而定)下的文件。如果浏览器访问页面未更新，需要清理浏览器图片和文件的缓存。

1.4 代码框架

```
src/
├── app
│ ├── about # 关于页面，项目说明文字
│ ├── app.component.html # 应用主入口
│ ├── app.component.scss # scss 样式文件
│ ├── app.component.spec.ts # 测试 spec 文件
│ ├── app.component.ts # app 组件
│ ├── app.module.ts # app 模块
│ ├── app-routing.module.ts # 主路由
│ ├── auth # 认证模块，包括登录页面，用户认证
│ ├── config # 配置模块，包含所有配置子组件
│ ├── config.service.spec.ts # 配置模块测试 spec 文件
│ ├── config.service.ts # 配置模块服务，用于与设备通信以及模块间通信
│ ├── footer # footer 模块，版权声明
│ ├── header # header 模块，导航路由，用户登录/登出
│ ├── preview # 预览模块，主页面码流播放器
│ ├── download # 录像/截图下载模块，录像/截图的查询以及下载
│ ├── face # 人脸模块，人脸总模块，包含参数以及人员管理功能
│ ├── face-manage # 人员管理模块，人脸识别注册以及人脸识别记录管理功能
│ ├── face-para # 人脸参数配置模块
│ ├── shared # 共享资源
│ │ ├── func-service # 通用服务以及函数
│ │ ├── player # player 模块，播放器功能模块
│ │ └── validators # Angular输入验证函数
│ └── tip # 提示框管理模块
├── assets
│ ├── css # 样式
│ ├── i18n # 多国语言翻译
│ ├── images # 图标
│ └── json # 调试用 json 数据库文件
├── environments # angular 发布环境配置
│ ├── environment.prod.ts
│ └── environment.ts
├── favicon.ico # 图标
├── index.html # 项目入口
├── main.ts # 项目入口
├── polyfills.ts
├── styles.scss # 项目总的样式配置文件
└── test.ts
```

详细模块位于 src/app/config

```
$ tree -L 2 src/app/config
├── config-audio # 音频配置
├── config.component.html # config组件主页面
├── config.component.scss # config组件样式
├── config.component.spec.ts
```

```
├── config.component.ts # config组件
├── config-event # 事件配置
├── config-image # ISP/OSD图像配置
├── config-intel # Intelligent智能分析配置
├── config.module.ts # 配置模块
├── config-network # 网络配置
├── config-routing.module.ts # 配置模块子路由
├── config-storage # 存储配置
├── config-system # 系统配置
├── config-video # 视频编码配置
├── MenuGroup.ts # 菜单数据类
├── NetworkInterface.ts # 网络接口数据类
├── peripherals # 外设拓展模块
└── shared # 一些共享子模块，可复用，方便后面主模块调整
    ├── abnormal
    ├── advanced-encoder
    ├── alarm-input
    ├── alarm-output
    ├── cloud
    ├── ddns
    ├── email
    ├── encoder-param
    ├── ftp
    ├── gate-config
    ├── hard-disk-management
    ├── info
    ├── intrusion-detection
    ├── intrusion-region
    ├── isp
    ├── motion-arming
    ├── motion-detect
    ├── motion-linkage
    ├── motion-region
    ├── ntp
    ├── osd
    ├── overlay-snap
    ├── picture-mask
    ├── port
    ├── pppoe
    ├── privacy-mask
    ├── protocol
    ├── region-crop
    ├── roi
    ├── screen-config
    ├── screenshot
    ├── smtp
    ├── tcpip
    ├── time-table
    ├── upgrade
    ├── upnp
    ├── user-manage
    └── wifi
```

2. ipcweb-backend

2.1 开发基础

web后端，采用ngingx+fastcgi，调试可以使用curl、postman或者直接与web前端联调。

开发语言： C++

参考文档：

HTPP协议知识
RESTful API 规范
Nginx + CGI/FastCGI + C/Cpp
POSTMAN

代码路径： app/ipcweb-backend

编译命令：

```
#在SDK根目录下
make ipcweb-backend-dirclean && make ipcweb-backend
make rk_oem-dirclean && make rk_oem target-finalize #重新编译oem
./mkfirmware.sh #打包oem.img，再进行烧写
```

配置文件：

nginx配置文件位于buildroot/board/rockchip/rv1126_rv1109/fs-overlay/etc/nginx/nginx.conf，部分摘要如下：

```
location /cgi-bin/ {
    gzip off;
    # 网页根目录
    root /oem/www;
    fastcgi_pass unix:/run/fcgiwrap.sock;
    fastcgi_index entry.cgi;
    fastcgi_param DOCUMENT_ROOT /oem/www/cgi-bin;
    # CGI 应用唯一入口
    fastcgi_param SCRIPT_NAME /entry.cgi;
    include fastcgi_params;

    # 解决PATH_INFO变量问题
    set $path_info "";
    set $real_script_name $fastcgi_script_name;
    if ($fastcgi_script_name ~ "^(.+?\.cgi)(/.+)$") {
        set $real_script_name $1;
        set $path_info $2;
    }
    fastcgi_param PATH_INFO $path_info;
    fastcgi_param SCRIPT_FILENAME $document_root$real_script_name;
    fastcgi_param SCRIPT_NAME $real_script_name;
}
```

2.2 编译环境

可以在SDK根目录下使用make ipcweb-backend编译，也可以使用以下命令编译。

```
mkdir build && cd build
【可选】该项目使用Google Test作为测试框架。初始化googletest子模块以使用它。
  git submodule init
  git submodule update

cmake .. -DCMAKE_TOOLCHAIN_FILE=
<path_of_sdk_root>/buildroot/output/rockchip_puma/host/share/buildroot/toolchain
.cmake

make
```

2.3 调试环境

1. 将编译出的entry.cgi文件推送到设备端的/oem/www/cgi-bin/路径下，确保entry.cgi文件的权限和用户组如下：

```
-rwxr-xr-x 1 www-data www-data 235832 Apr 26 20:51 entry.cgi
```

2. 确保设备端nginx服务已经启动，可使用ps命令查看。

```
538 root     12772 S    nginx: master process /usr/sbin/nginx
539 www-data 13076 S    nginx: worker process
```

3. 使用ifconfig -a命令获取设备端的IP地址。
4. 使用curl命令进行调试，示例如下：

```
$ curl -X GET http://172.16.21.217/cgi-bin/entry.cgi/network/lan
{"ipv4":
{"sV4Address":"172.16.21.217","sV4Gateway":"172.16.21.1","sV4Method":"dhcp","sV4N
etmask":"255.255.255.0"},"link":
{"sAddress":"84:c2:e4:1b:66:d8","sDNS1":"10.10.10.188","sDNS2":"58.22.96.66","sIn
terface":"eth0","sNicType":"10MD"}}
```

5. 由于CGI不能使用标准输出流，所以log保存在以下路径。

```
$ cat /var/log/messages
# 调试log输出到syslog
$ cat /var/log/nginx/error.log
# 网页服务器错误log
$ cat /var/log/nginx/access.log
# 网页服务器访问log
```