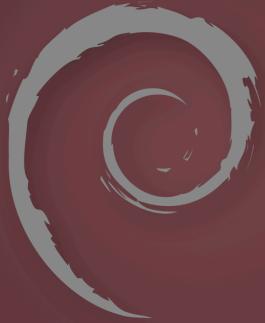


2024年全国大学生计算机系统能力大赛 操作系统设计赛(全国) OS功能挑战赛道

Linux的OTA升级系统

项目编号: Proj235 linux-upgrade-system • 队伍名称: 地铁行动2014



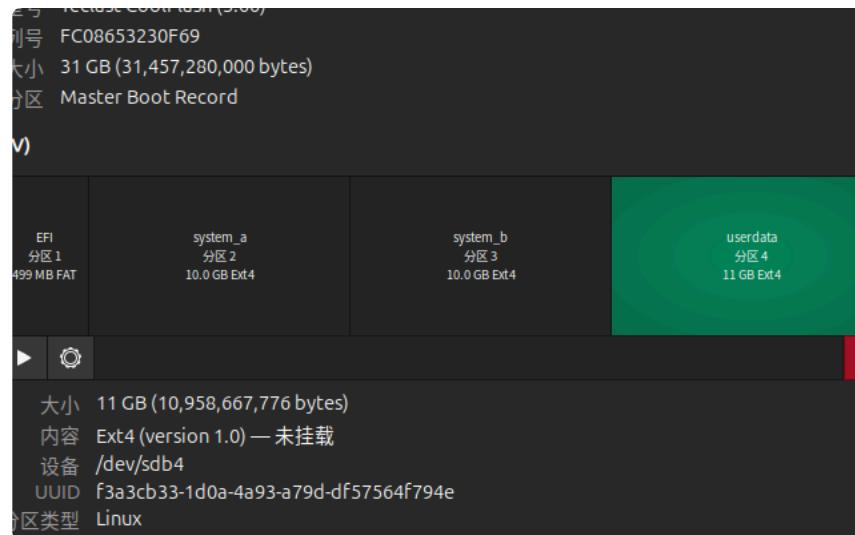
目录大纲

1. Linux的OTA升级系统
2. 目录大纲
3. 需求分析与实现方案
 1. 整体架构
 2. 技术实现
 1. 局域网升级
 2. 升级脚本
 3. 测试情况
 4. 项目收获
 5. 感谢指导

需求分析与实现方案

第一题 升级系统的升级功能实现 && 第二题 升级系统基础框架功能实现

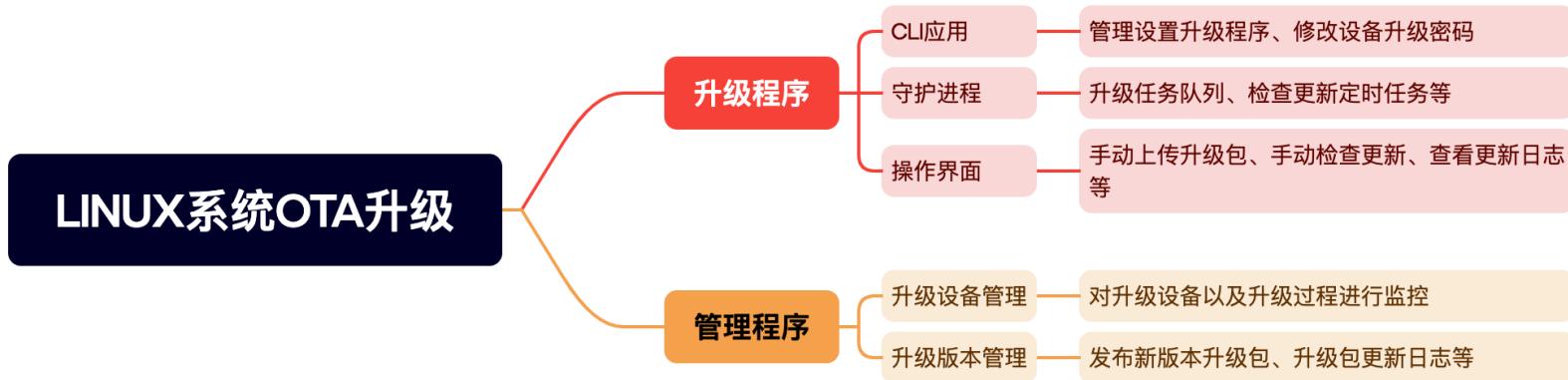
- 设立单独数据分区，将 home、opt、usr、var、www 等与设备配置相关的目录单独进行挂载点
 - 建立类似安卓的AB分区，升级过程中，通过修改 /etc/fstab 实现切换分区
 - 更改挂载点、使用 dd 刷写 (initrd,kernel,rootfs) 镜像
 - 安装开机自启应用，监控升级过程，检测重启次数，超出限制回滚回另一系统并标记
-
- 局域网升级功能
 - 版本发布管理平台
 - 设备升级管理平台



整体架构

升级程序总共分为 **ota-updater** 升级程序 与 **ota-manage** 管理平台 俩个部分。

- **ota-updater** 由 多个 客户端设备部署
- **ota-manage** 由 一个 管理服务端设备运行



技术实现

1. ota-updater 升级程序客户端

- 基于 **Golang + Gin** 开发的多模块应用
- 使用 BadgerDB 用于作为 KV 存储引擎
 - 数据加密（存储 webui 登录密码）
 - 存储日志
 - 存储队列（取代 redis）
- 模块：cmd/sys-updater cli 应用
 - 软链接到 /bin/sys-updater
 - sys-update switch_ab/update/check-update/reset-pwd args...
 - 对一些操作进行封装，用于手动操作或查看调试信息

```
var db *badger.DB

// InitBadgerDB KV Storage https://github.com/dgraph-io/badger
func InitBadgerDB() {
    opts := badger.DefaultOptions(flag.GetFlags().StoragePath)
    opts.IndexCacheSize = 100 << 20
    conn, err := badger.Open(opts)
    if err != nil {
        panic(err)
    }
    db = conn
}

func CloseBadgeDB() {
    err := getDB().Close()
    if err != nil {
        fmt.Println(err)
    }
}

func getDB() *badger.DB {
    if db == nil {
        InitBadgerDB()
    }
    return db
}
```

- 模块：cmd/daemon 守护进程
 - 维护更新队列、检查更新任务 cron/job
 - 提供 WEBUI 供用户操作，由 WEBUI 可套壳窗口应用等
 - 提供应用程序 HTTP API 接口

```
func RegisterRouter(router *gin.Engine) {  
    // frontend: client ui  
    router.Static("/_nuxt", flag.GetFlags().NuxtOutput+"/_nuxt")  
    router.StaticFile("/favicon.ico", flag.GetFlags().NuxtOutput+" /favicon.ico")  
    router.StaticFile("/", flag.GetFlags().NuxtOutput+"/index.html")  
  
    // backend: api  
    router.POST("/api/auth", services.LoginHandler)  
    router.GET("/api/info", services.InfoHandler) // 系统信息  
    router.GET("/api/check-status", services.CheckStatusHandler) // 检测更新状态  
    router.POST("/api/update", services.UpdateHandler) // 提交更新任务到最新版本  
    router.POST("/api/check-version", services.CheckVersionHandler) // 检测更新  
    router.POST("/api/manual-update", services.ManualUpdateHandler) // 手动更新  
}
```

- client-ui 升级程序客户端界面 基于 Nuxt(Vue3) + NuxtUI 构建升级操作界面，与 ota-updater 交互实现系统升级、升级设置、日志查看等操作。

```
async function onSubmit() {
  try {
    const res: { error: number, data: { token: string } } = await $fetch('/api/auth', {
      method: 'POST', body: { account: state.username, password: state.password }
    })
    if (res.error === 0) {
      session.value = res.data.token
      auth.value = true
      toast.add({ title: '登录成功!' })
      useRouter().push({ path: '/' })
    } else {
      error.value = '密码错误，请重试'
    }
  } catch {
    toast.add({ title: '无法连接守护进程，请重试!', color: 'red' })
  }
  loading.value = false
}
```

- 由 ota-updater 启动 HTTP 服务器，由根目录输出 client-ui 前端构建的静态文件
- client-ui 通过 fetch 向后端 API 发起请求



- `/login` 登陆页面 基于浏览器
SessionStorage 和 Token 的鉴权-机制

```
import { useSessionStorage } from '#imports'

const auth = ref(false)

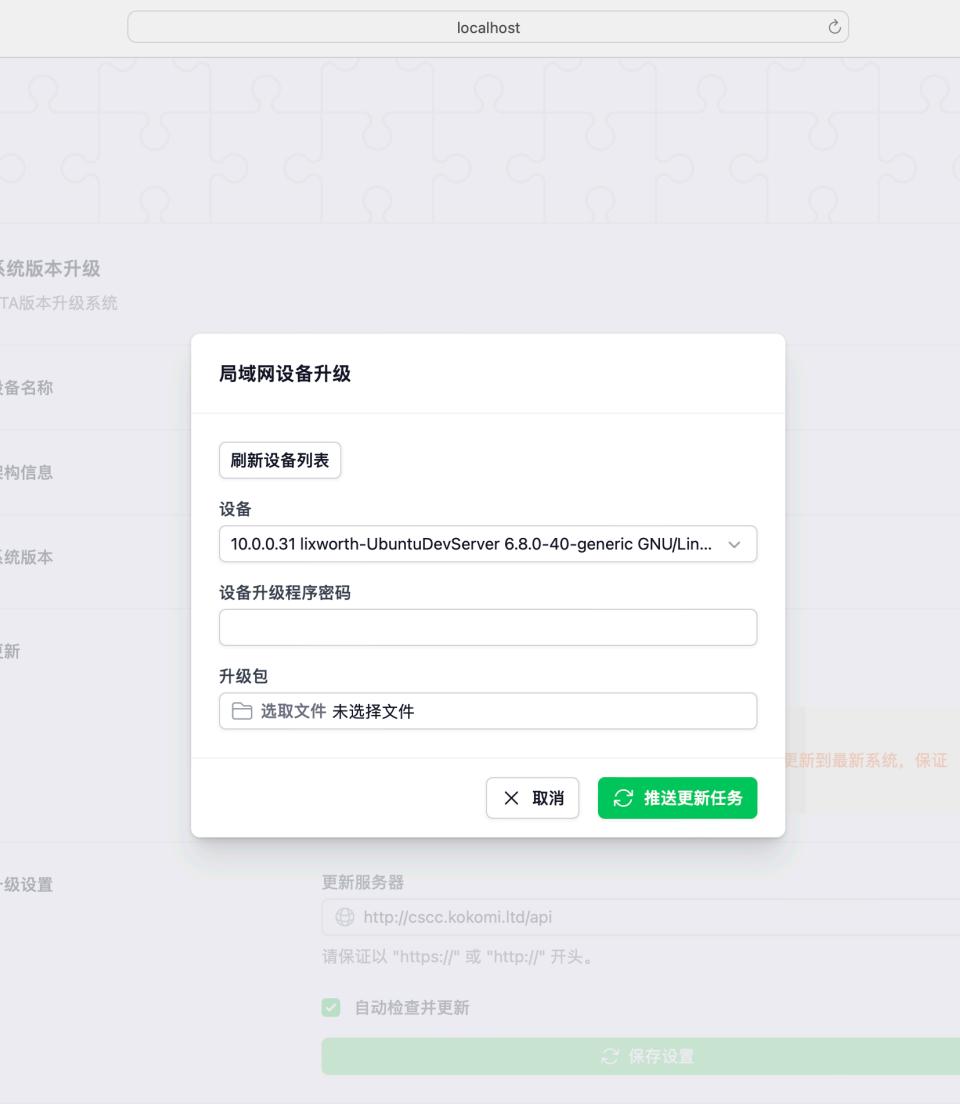
export function useAuthSession() {
  return useSessionStorage('AuthToken', 'null')
}

export function useAuth() {
  if (useAuthSession().value === 'null') {
    auth.value = false
  }
  else {
    auth.value = true
  }
  return auth
}

export function useLogout() {
  useAuthSession().value = 'null'
  auth.value = false
}
```

■ / 主页面 查看当前系统/版本信息、版本检查更新、手动更新、更新设置等操作





局域网升级

- nmap 扫描局域网
 - 发现 sys-updater 守护进程中的 API 服务
 - 获取到对应设备的设备信息与版本
- 任意局域网内的客户端发起 局域网设备升级操作
 - 上传升级包
 - 输入所更新局域网设备的 sys-updater 密码
 - P2P 发送升级包

在一些特殊情况下，无法访问外网或者说更新服务器的时候，便可使用这种方法，更新局域网的设备群。

升级脚本

- check_img.sh 校验升级镜像文件
- write_image_by_dd.sh 用 dd 写入镜像
- ab_switch.sh 通过修改 /etc/fstab 实现AB分区的切换
- check-osfullinfo.sh 检测系统信息
- check-install-sysenv.sh 检测系统依赖
- check-sysupdate.sh 检测系统更新
- sysupdate.sh 系统更新

I ❤️ #!/bin/bash

测试情况

操作系统	测试架构	启动方式	测试结果
Ubuntu/Debian	x86	grub	✓
openKylin	x86	grub	✓
ArchLinux	x86	grub	✓
openEuler	x86	grub	✓
Deepin	risc-v	grub/uboot	✓
AOSC	x86	grub	✓

- ota-manage 管理程序服务端 基于 Hyperf + ArcoDesignPro 构建的 前后端分离 版本发布与设备管理平台。

```
version: '3'
services:
  ota-frontend:
    container_name: ota-frontend
    image: ota-frontend
    restart: always
    build:
      context: ./ota-frontend
    volumes:
      - ./ota-frontend:/opt/www/ota-frontend
    networks:
      - ota_manage_network
    ports:
      - 80:8080
    environment:
      - API_PROXY=ota-backend:9501
  mysql:
    image: mysql:8.0.26
    environment:
      MYSQL_ROOT_PASSWORD: root123
      MYSQL_DATABASE: ota_manage
    restart: always
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
```

```
redis:
  image: redis
  restart: always
  ports:
    - "63790:6379"
  networks:
    - ota_manage_network
  volumes:
    - redis_data:/data
  command: redis-server --appendonly yes --appendfsync e
ota-backend:
  container_name: ota-backend
  image: ota-backend
  build:
    context: ./ota-backend
  restart: always
  volumes:
    - ./ota-backend:/opt/www/ota-backend
  networks:
    - ota_manage_network
  ports:
    - 9501:9501
  environment:
    - APP_ENV=prod
    - SCAN_CACHEABLE=false
```

项目收获

开发过程中所遇问题

- Q: 传统开发项目中，需外挂 redis、持久化db 等数据库用来存储信息与维护队列，但是系统级应用需轻量
- A: 采用 BadgerDB/SQLite 类型的“自给自足的、无服务器的、零配置的”数据库软件

收获心得

- Openwrt 中 `sysupgrade` 通过执行 shell 脚本以及配合 luci webui 界面进行升级的结构
- Android OTA 更新中的 AB 分区方案

由于时间精力原因，并不能完全完善所有设想。开发后期，也了解到了其他小组使用 `ostree`（类似于 git 版本控制）的升级方案，确实在实现 OTA 升级问题上拥有更多优势，未来有时间可能会深入了解一下。

本次项目制作经历仍然受益匪浅，感谢各位导师！

感谢指导

@lixworth 2024年8月

Powered by  Sliderv