**项目名称：基于Orange Pi Zero2的嵌入式系统资源监控器**

技术栈：C语言、Linux系统编程、Orange Pi Zero2（ARM64）、交叉编译、Proc文件系统

一、 项目背景与核心目标

本项目是项目驱动学习理念下的首次完整实践，旨在打通从宿主机开发到目标板部署的完整嵌入式开发流程。核心目标是开发一个在Orange Pi Zero2上稳定运行的守护程序，实时监控并汇报系统关键资源状态（内存、CPU负载），为后续复杂的音视频应用奠定底层监控基础。

二、 系统架构与技术选型

| 模块 | 技术选型 | 理由 |
| --- | --- | --- |
| 开发环境 | Windows宿主机 + ARM64交叉编译工具链 | 拒绝模拟器，建立真实的工业级开发流程，确保程序在目标架构高效运行。 |
| 硬件平台 | Orange Pi Zero2 (全志H616, 1GB RAM) | 选择主流且性价比高的嵌入式平台，贴合中小企业实际项目环境。 |
| 系统接口 | Linux Proc文件系统 (/proc/meminfo, /proc/loadavg) | 采用Linux内核标准接口获取系统信息，代码可移植性强，不依赖特定硬件驱动。 |
| 项目结构 | 多文件C语言项目，头文件与实现分离 | 建立良好的代码架构，摒弃“面条代码”，便于功能扩展与维护。 |

1. 详细实现过程与关键技术点

开始：9.30

环境搭建（1.5天）- 核心能力：工具链配置与硬件调试

交叉编译环境：在Windows宿主机上成功配置aarch64-none-linux-gnu-gcc工具链。

系统部署与调试：将Debian Server系统烧录至TF卡，并通过USB-TTL串口完成首次登录与调试，解决了引脚定义、波特率不匹配等典型硬件问题。

网络打通：通过配置Windows网络共享，为开发板分配静态IP，建立了SCP/SSH高效部署通道，极大提升了后续开发调试效率。

核心模块开发（0.5天）- 核心能力：Linux系统编程

内存监控模块 (memory\_monitor.c/h)：

通过文件操作接口 (fopen, fgets) 读取并解析 /proc/meminfo。

提取 MemTotal, MemFree, MemAvailable 关键字段。

实现使用率百分比计算，提供数据给显示模块。

系统负载模块 (load\_monitor.c/h)：

解析 /proc/loadavg，获取系统1、5、15分钟平均负载。

主程序 (main.c)：

集成各监控模块，实现定时轮询与信息打印。

构建与部署

使用交叉编译命令：aarch64-none-linux-gnu-gcc -o monitor main.c memory\_monitor.c load\_monitor.c -static

通过SCP命令将生成的可执行文件部署至开发板。

四、 遇到的问题与解决方案（体现批判性思维与执行力）

| 问题 | 分析与解决过程 | 体现的能力 |
| --- | --- | --- |
| 串口无输出 | 1. 排查线序：纠正TX-RX交叉连接原则。 2. 排查引脚：发现错误的13pin定义，最终定位至26pin GPIO的UART1。 3. 排查波特率：从乱码锁定波特率不匹配，最终确认 1500000 为正确值。 | 系统性硬件调试能力 |
| 网络不可用 | 1. 放弃DHCP自动获取（因环境限制）。 2. 自主探索并成功配置Windows网络共享，手动设置静态IP，打通网络。 | 环境适应与自主解决问题能力 |
| 终端显示异常 | 长命令粘贴错乱，通过调整MobaXterm终端宽度解决，理解了终端行列设置对显示的影响。 | 底层工具原理的理解 |

五、 项目成果与输出

功能成果：一个在Orange Pi Zero2上稳定运行的命令行系统监控程序，可实时显示内存使用情况与系统平均负载。

流程成果：成功建立了 “Windows编辑 & 交叉编译 -> SCP部署 -> ARM64目标板运行” 的标准化嵌入式开发工作流。

代码成果：具备良好架构的C语言项目，为后续功能扩展打下坚实基础。

六、 待优化方向与未来规划（呼应你的核心诉求）

功能深化：

实现 get\_cpu\_usage() 函数，通过解析 /proc/stat 计算实时CPU利用率。

增加进程监控功能。

架构升级：

引入多线程，将数据采集、数据处理与显示分离。

将数据通过Socket发送至PC端，实现简单的C/S架构可视化。

向音视频演进：

将本项目作为系统底座，集成即将开发的音视频流模块，实现对编码、推流等进程的资源监控，形成完整解决方案。

**要求提升：**

1.把它改造成一个守护进程，要求系统启动时自动运行，崩溃时自动重启。

2.实现日志轮替，防止日志文件占满整个TF卡。

3.加入网络通信，让数据能在局域网内被查询。

4.给你现有的监控器项目，增加两个“工业级”功能：

4.1全面的错误处理：所有系统调用（fopen, fgets等）都要检查返回值，并打印清晰的错误日志。

4.2做成一个“守护进程”：让它可以一直在后台运行，就像nginx那样。

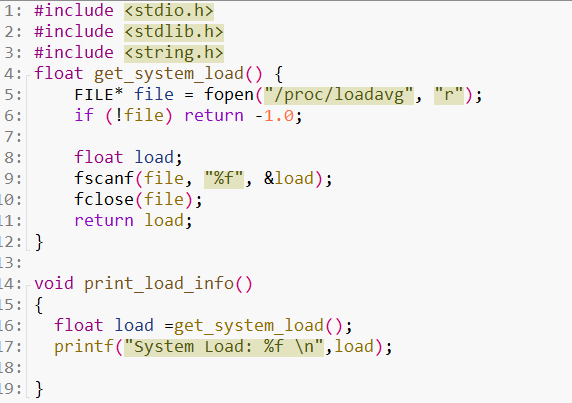
实现这两个功能时，你会遇到无数问题（比如怎么让进程后台化，怎么不让它退出）。

每遇到一个报错，就把它当成一次面试提问。去查、去学、去解决。

当你搞定这两个功能，你就掌握了：错误处理、进程控制、信号、日志系统。这些就是C语言在Linux系统编程里的核心。

**功能：**

一．实现命令cat /proc/loadavg 读取第一个浮点数并打印相关信息，也就是系统过去1分钟平均负载，即平均有 0.32 个进程在运行队列中。

代码实现：

1. 实现命令cat /proc/Meminfo | grep Memavailable,读取Memavailable行并打印相关信息，即打印可用内存

代码实现：

1. 实现命令 cat/proc/stat | grep cpu ，读取该行，通过把cpu分配的时间片数量计算cpu使用率，也就是把10个无符号整数加起来得到总时间，idle和iowait加起来得到空闲时间，两次记录求差再把时间相除得到空闲率，1减去它就是cpu使用率。

代码实现