c语言下的工厂模式ipmi源码分析 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/448/2021_2022_c_E8_AF_AD_E8_A8_80_E4_B8_8B_c97_448758.htm IPMItool的架构==========源码目录如下 ---contrib //用于建立web管理页面的shell脚本 ---control //包含一些安装、配置信息 ---debian //包含changelog等信息 ---doc //man的帮助信息 ---include | \---ipmitool //头文件定义 ---lib //对IPMI规范的对应实现，如ipmi_session.c处理session \---src //此目录下是ipmitool的三个主程序 \---plugins // ipmi_intf.c interface一些通用功能的实现 ---bmc // ipmitool与bmc kernel driver之间的接口 ---imb // Intel IMB Interface ---lan // IPMI v1.5 LAN Interface ---lanplus // IPMI v2.0 RMCP LAN Interface ---lipmi // Solaris x86 LIPMI interface \---open // Linux OpenIPMI Interface [default]C语言下的工厂模式==========从上述的目录结构不难看出，IPMItool设计上的一个重要特色就是将不同的interface看作plugin。从而使系统具有清晰的结构和良好的扩充性。IPMI规范中定义的实体，如session,fru,sdr,chassis,sensor等等，都在lib中做对应的实现。这部分是与具体interface相分离的。interface的通用接口在include\ipmitool\ipmi_intf.h中定义；interface的通用功能实现，在\src\plugins\ipmi_intf.c中。值得注意的是，interface是IPMI规范中定义的概念，普通意义上的接口本文中均使用中文“ 接口 ”。这种将通用接口与具体实现相分离的方式无疑就是一种简单工厂模式了。实践====那么interface具体是怎么实现为plugin的呢？可以从一个具体的例子看一下。include\ipmitool\ipmi_intf.h中用ipmi_intf定义了

了 struct ipmi_intf { ... struct ipmi_session * session. struct ipmi_oem_handle * oem. uint32_t my_addr. uint32_t target_addr. int (*setup)(struct ipmi_intf * intf). int (*open)(struct ipmi_intf * intf). void (*close)(struct ipmi_intf * intf). ... }.与OO语言类似，struct内部定义了数据和方法。不同的是，方法采用的是函数指针的方式。因为没有this指针，所以函数的形参就是指向自身struct的指针。如setup。而在具体实现中，如src\plugins\lan\lan.c中给出了具体的函数实现。如ipmi_lan_setup。在ipmi_lan_setup中，即可使用形参定义的intf指针实现对ipmi_intf结构中相应数据的操作。struct ipmi_intf ipmi_lan_intf = { name: "lan", desc: "IPMI v1.5 LAN Interface", setup: ipmi_lan_setup, open: ipmi_lan_open, close: ipmi_lan_close, sendrecv: ipmi_lan_send_cmd, sendrsp: ipmi_lan_send_rsp, keepalive: ipmi_lan_keepalive, target_addr: IPMI_BMC_SLAVE_ADDR,}.