

分类号 TP315

学校代码 10147

UDC 004

密 级 公开

硕 士 学 位 论 文

基于 ARM Linux 的码头集装箱堆场的应 用研究

Application Research of ARM Linux in Container
Terminal Yard

作者姓名 连照亮

指导教师 徐世国 教授

申请学位 工学硕士

学科专业 计算机应用技术

研究方向 嵌入式系统应用

辽宁工程技术大学

致 谢

本论文是在徐世国教授的指导下完成的，衷心的感谢我的导师徐世国老师，感谢老师在这两年半的时间里对我的指导和帮助。在我读研的期间，老师为我提供了良好的学习环境，在学习上和生活上给我很多指导和关怀。他严谨的治学态度、渊博的知识和广阔的学术视野，使我少走了很多弯路，同时也扩展了我的思维，使我掌握的知识更加全面。从老师身上，我学到了很多为人处事的道理，为我今后的学习和工作树立了榜样。在此，谨向我的导师致以最诚挚的敬意。

衷心的感谢答辩的各位老师，特别要感谢刘建辉教授、陈万志老师和关昕老师，他们给我论文上的细心指导，为我拓展专业知识，扩展视野奠定了坚实的基础。

衷心的感谢在攻读硕士期间出现在我身边的每一位朋友，感谢我的室友和朋友们，特别是李颖、李忠波、王楠、姚旭，在读研期间，不管遇到什么困难他们都能在我身边默默的支持我，鼓励我，是你们让我的生活更加丰富多彩。

衷心的感谢我的父亲母亲，是你们一直默默的鼓励我支持我关心我帮助我，在生活上和精神上给了我无穷的力量，让我战胜各种困难。是你让我顶住了压力，顺利完成学业。

摘 要

码头集装箱信息化管理是国际贸易和现代物流行业发展的必然趋势。随着全球经济与货物集装箱需求的飞速增长，码头集装箱作业环境也随之发生着变化。由于我国大部分集装箱堆场没有使用管理系统，使大量的集装箱堆积，造成了堆场空间的资源浪费。大部分作业都是通过人工抄录来进行集装箱的记录与管理，由于该方式受人工因素的影响，工作效率相对较低。针对目前集装箱堆场的这一管理现状，将嵌入式手持终端技术应用在码头物流的集装箱堆场管理当中，从而提升集装箱的周转速度，提高周转效率，进而提高集装箱堆场的管理水平和堆场空间的利用率。

本论文搭建了一个基于 ARM9 处理器的嵌入式 Linux 系统平台。主要分析了建立 linux 系统的软件与硬件的系统结构，研究了搭建嵌入式系统平台的过程，包括交叉编译环境的建立， Bootloader 引导程序的建立，内核的移植与根文件系统的建立等等；实现了嵌入式数据库 SQLite3 的移植。同时结合 Qt/Embedded 的特性和优点，分析了嵌入式 Linux 操作系统下 Qt/Embedded 图形用户界面的特点和结构。提出了用 Qt/Embedded 实现码头物流管理系统的图形界面这一设计思路，与此同时研究出了应用在嵌入式手持终端系统中的查询算法，并将其应用在嵌入式码头物流管理的手持设备终端中。

关键词：嵌入式 GUI；Qt/Embedded ； SQLite3 ； 集装箱堆场

Abstract

Container Terminal Information Management is the inevitable trend of international trade and modern logistics development. With the rapid growth of global economy and demand of cargo, container terminal operating environment changes continuously. Because most of the container yard in China doesn't use management system, a large number of container stacking make the waste of yard space resources. Most of operation management is the container record and management carried by artificial transcription. This way is affected by artificial factors, so the work efficiency is low. For the current management status of container yard, the paper use embedded handheld terminal technology in terminal logistics container yard management to enhance the efficiency and turnover rate of the container, to improve the management level of the container yard and efficiency of container yard space as much as possible.

This paper implements an embedded Linux system platform based on ARM9 processor. The article mainly analyzes software and hardware system architecture of the embedded Linux system, and the process of building the embedded system platform, including Kernel transplant, the establishment of root directory and so on. This paper implements transplant of embedded database SQLite3. Considering of the characteristics and advantages of Qt/Embedded, the paper analyzes the character and structure of Qt/Embedded GUI on Embedded Linux. This paper present the design of graphical interface of terminal logistics management system using Qt/Embedded and analyzes Search algorithms in application of embedded handheld terminal system, and uses the search algorithms in the handheld devices of embedded terminal logistics management.

Key Words: Embedded GUI; Qt/Embedded ; SQLite3; Container Yard

目 录

摘 要

Abstract

1	绪论	1
1.1	选题的研究目的及意义	1
1.2	国内外研究现状及发展趋势	2
1.3	论文研究的主要内容及安排	3
1.3.1	研究的主要内容	3
1.3.2	论文的主要工作及内容安排	4
2	嵌入式系统 GUI 开发平台的构建设计	6
2.1	嵌入式系统	6
2.1.1	嵌入式系统的特点	6
2.1.2	嵌入式系统的设计流程	7
2.2	嵌入式 GUI 系统分析	8
2.2.1	MiniGUI	8
2.2.2	OpenGUI	9
2.2.3	Qt/Embedded	9
2.2.4	嵌入式 GUI 性能分析	10
2.3	嵌入式系统的硬件平台设计	11
2.4	嵌入式系统的软件平台设计	13
2.4.1	交叉编译环境建立	13
2.4.2	建立引导加载程序	15
2.4.3	Linux 内核移植	17
2.4.4	建立根文件系统	21
2.5	嵌入式 GUI 的平台设计	23
2.5.1	Qt/Embedded 架构	23
2.5.2	信号与槽的机制原理	25

2.5.3	Qt/Embedded 的移植	27
2.5.4	嵌入式数据库 SQLite	29
2.5.5	SQLite3 交叉编译	31
2.6	本章小结	33
3	嵌入式码头物流管理手持终端的设计	34
3.1	码头物流管理界面程序的基本架构	34
3.2	码头物流管理手持终端界面设计	34
3.2.1	手持终端功能框架设计	35
3.2.2	手持终端功能模块界面设计	37
3.3	嵌入式数据库的设计	43
3.3.1	数据库的框架设计	43
3.3.2	嵌入式数据库 SQLite 函数分析	44
3.4	本章小结	46
4	集装箱堆场的空间分配及算法调度的研究	47
4.1	集装箱堆场作业模式	47
4.1.1	集装箱堆场	47
4.1.2	优化堆场的作业空间	48
4.2	集装箱堆场空间分配策略的研究	50
4.2.1	堆场箱位分配策略研究	50
4.2.2	堆场箱位任务优先级分配策略	51
4.3	嵌入式实时数据库查询优化算法研究	52
4.3.1	手持终端与嵌入式数据库的交互分析	53
4.3.2	嵌入式系统应用查询算法的问题分析	54
4.3.3	算法在嵌入式码头管理系统中的应用研究	54
4.4	本章小节	58
5	算法实验与分析	60
5.1	算法描述分析	60
5.2	实验环境	61

5.3 实验结果与性能分析	61
5.3.1 运行时间分析	61
5.3.2 生成计划的质量分析	62
5.3.3 算法模块的查询质量分析	62
5.4 本章小结	64
结论与展望	65
参 考 文 献	66
作 者 简 历	68
学位论文数据集	70

1 绪论

近年来由于电子信息化行业的飞速发展，嵌入式系统，已经广泛的应用于社会的各个领域当中，例如消费电子、制造业、工业控制、安防系统等等。与此同时，嵌入式系统也涵盖了多项应用技术，例如计算机软件和硬件的应用、微电子技术和电子信息技术等等。随着计算机技术的不断发展，嵌入式系统的应用将呈现系统复杂化、硬件集约化、应用多样化、软件平台化等特点。

码头集装箱堆场的发展是国际贸易和现代物流行业发展的产物。随着全球经济与货物集装箱需求所带来的飞速增长。中国港口和国际海运业务都得到了快速的发展，都给世界集装箱运输带来了新的增长趋势。与此同时，我国港口集装箱的吞吐量也增长迅速，码头集装箱作业环境也发生着不断的变化，给集装箱码头带来了前所未有的发展机遇。然而，由于我国集装箱堆场管理信息化的程度较低，大部分的集装箱堆场没有使用管理系统，大量的集装箱堆积造成了堆场空间的资源浪费。大部分情况都是通过人工抄录进行集装箱的记录和集装箱的管理。该方式受人工因素影响较大，工作效率较低。

针对目前集装箱堆场的这一管理现状，将嵌入式技术应用在码头物流的集装箱堆场管理当中。从而提升堆场集装箱的周转效率和周转速度，提高集装箱堆场的管理水平，尽可能的提高堆场空间的利用效率。

1.1 选题的研究目的及意义

由于嵌入式系统主要用在一些特定的专用设备上，通常这些特定专用设备的硬件资源（如处理器、存储器等）非常有限，而且对成本的要求敏感，有时对实时响应要求也很高。然而随着消费家电的智能化，嵌入式系统的发展就更显得重要了。这就给嵌入式系统技术带来了广阔的应用前景，例如 PDA、手机、可视电话、电子字典、汽车电子、家电控制系统、医疗仪器、航天航空设备等等都是典型的嵌入式系统^[1]。

由于嵌入式系统的广泛应用，用户对数据信息处理的要求也在不断的提高，同时希望对嵌入式产品中的数据进行更有效的管理，嵌入式数据库便是针对嵌入式系

统而出现的一个数据库工具。嵌入式数据库大多用在智能终端中，例如 PDA、车载设备等内存资源非常有限的嵌入式设备中。嵌入式系统与嵌入式数据库通常是集成在一起的应用，不用单独运行数据库引擎，由程序直接调用相应的 API 就可实现对数据的存取、查询等相关操作。嵌入式系统的开发环境决定了其数据如何在智能终端中使用嵌入式数据库，以及如何让嵌入式数据库在嵌入式设备中充分有效的使用有限的存储资源成为嵌入式系统研究的重要方向^[2]。

随着世界经济一体化和现代科技的迅速发展，物流业对全球经济各行业都产生了重要影响，而现代的物流业也是全球的范围内出现的与高科技相结合的新兴产业。码头集装箱运输是货物进行运送的一种新颖的运输方式。采用集装箱运输货物的方式具有装卸作业效率高、节约费用等特点。而随着码头集装箱港口货物的吞吐量的不断增加，港口集装箱作业效率的高低就直接关系到运输过程中的效率与生产的成本^[3]。而我国的大部分港口集装箱码头现有的作业方式还是完全依靠人工抄录集装箱存储的作业方式，这其中就存在着大量的问题，如审验集装箱的时间过长，效率低，容易造成港区集装箱码头的堵塞。而人工审验又容易出现漏洞及差错，由于作业工作量大，周转效率慢，大量的人工作业很难保证数据的准确性。如果港口拥堵的现象十分严重的话，这就会成为制约港口发展的主要因素。

针对目前集装箱堆场的管理现状，将嵌入式手持终端技术应用在码头物流的集装箱堆场管理当中。提出嵌入式码头集装箱手持终端这一观点，主要是提高集装箱码头堆场的运作效率，提高港口集装箱的调度效率，提升集装箱堆场的管理水平，减少人为地失误，尽可能的提高堆场空间的利用效率。为集装箱码头堆场的自动化提供了有力的条件。

1.2 国内外研究现状及发展趋势

在欧美的一些发达国家，手持终端设备的应用最早是在超市和物流行业兴起的。在超市中，嵌入式手持设备主要是盘点一些出入仓库货物的工作；在物流行业中，物流的工作人员使用嵌入式手持设备，将收发货物的一些基本信息输入到手持设备中。在早期的应用中，嵌入式手持设备主要是用于记录数据的。随着嵌入式手持终端设备的发展，如 PDA、各种手持条码数据采集器等手持终端的出现，除了物

流、盘点外，在抄表、卫生医疗和餐饮等行业都开始使用手持设备来记录数据，用以替代原始的手工数据记录作业以提高作业效率。就嵌入式手持设备的整体研究来看，欧美发达国家嵌入式技术起步较早，因此其产品较为成熟，功能较为完善，交互性也较好^[3]。

随着科技信息技术的发展以通讯技术、网络技术、控制技术等为代表的全球化信息网络技术的兴起，使我们可以应用先进的科技信息技术，将这些技术应用在码头集装箱的管理上，为实现集装箱港口进行快速、高效地生产提供了可能。

目前，信息技术已经在港口航运企业得到了应用。例如，我国沿海一些较为先进的大型集装箱港口都使用了无线电通讯技术来代替高频对讲机，使港口集装箱的物流信息和堆场的作业信息传递更为快速、准确。除此之外，一些先进的嵌入式技术，如条码技术、射频技术和全球定位系统等技术也应用在集装箱港口中，为集装箱堆场物流管理的高效运作提供了保障^[4]。

1.3 论文研究的主要内容及安排

1.3.1 研究的主要内容

本课题的出发点是以嵌入式系统开发平台为基础，设计出带有人机交互界面的码头物流管理的手持终端，以更方便的应用于码头管理这一领域。同时针对码头物流管理的特点，针对集装箱堆场的集装箱出入库区的优化作出研究。并提出一种适用于手持终端的嵌入式实时数据库查询算法的研究，并将其应用于嵌入式码头管理终端中。

本课题的主要研究内容包括以下几个方面：

(1)构建 ARM-Linux 开发环境。包括，Linux 内核的配置与编译系统的研究，系统开发环境的建立等等。

(2)移植 QT/Embedded，移植 SQLite 数据库。为该平台上的应用程序开发提供图形用户界面的开发环境。

(3)建立 QT/Embedded 和 Qtopia 的开发环境以及运行环境；数据库文件的建立；SQLite 与 QT/Embedded 的连接；采用 designer 界面设计与源代码编写相结合的方式

编写应用程序实现码头物流管理的上述基本功能。将应用程序发布到 Qtopia 平台上。

(4)对码头集装箱堆场的作业空间及针对应用在嵌入式手持终端物流管理系统的算法进行研究分析。

(5)对提出的优化算法进行实验分析。

1.3.2 论文的主要工作及内容安排

论文的结构安排如下：

第一章 绪论

介绍了选题研究的目的和意义，阐述了嵌入式系统在国内外的研究现状及其发展趋势，分析了集装箱港口作业效率低下的问题，以及以后集装箱码头管理作业的发展趋势。

第二章 嵌入式系统 GUI 开发平台的构建设计

首先分析嵌入式系统的特点及嵌入式系统的设计流程。研究了目前嵌入式 Linux 下比较流行的几种嵌入式 GUI 系统，对嵌入式 GUI 的优缺点进行了比较。在搭建硬件平台同时也分析了 Linux 操作系统平台的构建过程和构建方法。首先介绍了 Linux 环境下如何建立交叉编译环境，接着在嵌入式软件平台的基础上分析研究了 BootLoader 的移植、Linux 内核配置和移植、驱动程序的加载、根文件系统的制作等。并对嵌入式图形开发工具 QT 以及 Qt/Embedded 的架构进行了深入分析，针对其特性分析了信号与槽的机制原理，研究了 Qt/Embedded 的移植以及开发平台。与此同时对嵌入式数据库进行了分析，研究了 SQLite3 的移植、安装过程。为下一章节的系统构建打好基础。

第三章 嵌入式码头物流管理手持终端的设计

设计了码头物流管理界面程序的基本框架，详解论述并分析了码头物流管理终端界面的设计，分析了各个功能模块。并对各功能模块的界面进行设计。设计了嵌入式数据库的框架，并对 SQLite 的各功能接口函数进行分析研究。

第四章 集装箱堆场的空间分配策略及算法调度的研究

对堆场箱位分配策略的研究，以及堆场任务优先级分配策略的研究。重点研究了应用在嵌入式实时数据库查询优化算法，分析了适合应用在嵌入式手持终端中的算法，设计算法使其应用在嵌入式码头管理系统中。

第五章 算法实验与分析

通过举例和实验进一步证明了结合禁忌搜索算法和贪婪算法提出改进连接顺序优化算法的可行性。举例用具体的数值来说明嵌入式查询算法 TGI 与贪婪算法和禁忌搜索算法的不同，直观的说明了 TGI 算法的优点。与禁忌搜索算法和贪婪算法的模拟实验表明，本算法在系统的运行时间、生成计划的质量改进比和查询的效率都有一定的优势。通过算法的实验与分析表明，在嵌入式实时数据库查询系统中 TGI 算法更适于应用的环境，也更适于系统的性能。

第六章 结论与展望

总结了论文在研究设计阶段所作的工作。指出存在的不足，并对下一步工作进行展望。

2 嵌入式系统 GUI 开发平台的构建设计

2.1 嵌入式系统

随着嵌入式科技的不断发展，嵌入式系统已经成为当今最为热门的领域之一，它快速的发展势头引起了广泛的关注。

嵌入式系统就是以嵌入式计算机为核心，面向用户、面向产品、面向应用，软硬件可裁减的，它们适用于那些对功能、可靠性、成本、体积、功耗等综合性能有严格要求的专用计算机系统。它主要由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及嵌入式应用软件等部分组成。它具有“嵌入性”、“专用性”与“计算机系统”的三个基本要素。从组成上看，嵌入式系统可分为嵌入式硬件系统与嵌入式软件系统两大部分^[5]，如图 2.1 所示。

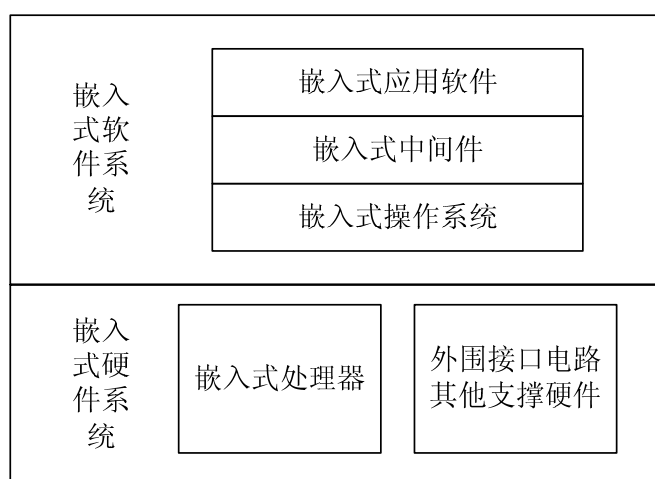


图 2.1 嵌入式系统的组成

Fig.2.1 Composition of embedded system

2.1.1 嵌入式系统的特点

嵌入式系统的特点如下^[6]：

(1) 面向特定应用的特点。嵌入式系统与其他通用型系统的最大区别就是在于嵌入式系统为一些特定用户人群设计的系统，因此它具有低功耗、体积小、集成度高等特点，并且可以满足不同应用领域的特定需求。

(2) 嵌入式系统的软硬件都需要高效的设计，量体裁衣、去除冗余，这就要求在硅片大小的面积上实现更高的性能。

(3) 嵌入式系统是将先进的计算机应用技术、半导体技术和电子技术与各行业具体应用在一起相结合的产物。

(4) 嵌入式开发的软件要求代码高质量性、高可靠性。这是因为嵌入式设备所处的环境往往是无人看守或是环境条件恶劣的情况下。

(5) 嵌入式系统本身没有二次开发能力，即设计完成后用户不能对其中的程序功能进行修改，必须需要有一套开发工具和环境才能对其进行再次开发。

2.1.2 嵌入式系统的设计流程

嵌入式系统设计过程与一般的工程设计方法没有太大的差别，都要有需求分析、系统设计、系统实现以及测试等流程。一般为了更好的加快嵌入式系统的开发，通常可以采用软硬件并行设计的方法^[5]，如图 2.2 所示。

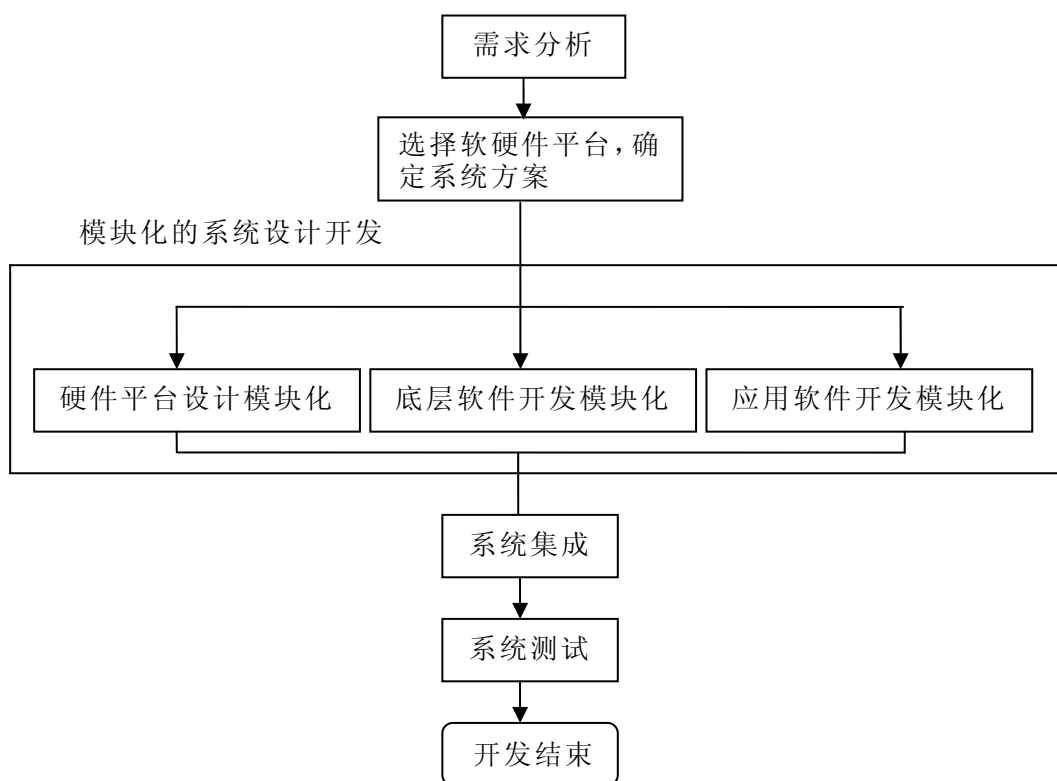


图 2.2 软硬件并行设计的嵌入式系统设计流程

Fig. 2.2 The design flow of embedded system with concurrent design of hardware and software

嵌入式系统项目的需求分析主要是根据应用中产生的需求来确定要解决的问题及需要达到的目标，并且将应用需求转变为嵌入式系统的系统要求，确定嵌入式系统在性能、存储容量和所需要外设等设计条件。完成需求分析后，然后的工作就是根据需求分析得出的设计结果以及限制条件来选择系统的软硬件平台，确定整个系统的方案。

2.2 嵌入式 GUI 系统分析

嵌入式 GUI 为嵌入式系统提供了应用于特殊场合的人机交互接口。嵌入式 GUI 要求简单、直观、占用资源小且反应快速等特点，以适应系统硬件资源非常有限的条件。另外，由于对嵌入式系统硬件本身的特殊性，嵌入式 GUI 应该具备可移植性与可裁减性，以适应不同系统的硬件条件和使用需求。综上所述，嵌入式 GUI 具备以下特点^[7]：

- (1) 体积小；
- (2) 运行时耗用系统资源小；
- (3) 上层接口与硬件无关，高度可移植性；
- (4) 高可靠性；
- (5) 在某些应用场合应具备实时性；

2.2.1 MiniGUI

MiniGUI 是由北京飞漫软件技术有限公司创办的开源 Linux 图形用户界面支持系统，经过近几年的发展，MiniGUI 已经发展成为比较成熟的、性能优良的、功能丰富的跨操作系统的嵌入式图形界面支持系统。“小”是 MiniGUI 的特色，它目前已经广泛应用于通讯、医疗、工控、电子、机顶盒、多媒体等领域。目前，MiniGUI 的最新版本为 MiniGUI 3.0，MiniGUI 对中文的支持最好。它支持 GB2312 与 BIG5 字元集，其他字元集也可以轻松的加入。

由于，MiniGUI 开发的主要目标是为基于 Linux 的实时嵌入式系统提供一个轻量级的图形用户界面支持系统。所以，MiniGUI 为应用程序定义了一组轻量级的窗口和图形设备接口。利用 MiniGUI 提供的这些接口，每个应用程序可以建立多个主

窗口，然后在这些主窗口中创建按钮、编辑框等控件。MiniGUI 还为用户提供了丰富的图形化功能，帮助用户显示各种格式的位图并且可以在窗口中绘制比较复杂的图形^[8]。

2.2.2 OpenGUI

OpenGUI 采用 LGPL 条款发布。OpenGUI 比较适合基于 x86 平台的实时系统，它的跨平台的可移植性较差，目前发展较慢。

由于，OpenGUI 在 Linux 系统上已经存在一段时间了。最初的 OpenGUI 只支持 256 色的线性显存模式，而现在支持其它的显示模式，并且支持多种操作系统平台，比如 MS-DOS、QNX 和 Linux 等，不过目前只能支持 x86 硬件平台。OpenGUI 共分为三层：最低层是由汇编语言编写的快速图形引擎；中间层是提供了图形绘制的 API 接口函数，包括线条、矩形和圆弧等，并且兼容 Borland 的 BGI API；第三层是用 C++ 语言编写的，提供了完整的 GUI 对象库。另外 OpenGUI 还提供二维绘图原语、消息驱动的 API 及 BMP 文件格式支持等，使用较为方便。OpenGUI 同样支持鼠标和键盘事件，在 Linux 上基于 Frambuffer 或 SVGALib 实现绘图。由于它基于汇编实现的内核并且利用 MMX 指令进行了优化，所以其运行速度非常快。正是由于其内核用汇编实现，使可移植性受限^[9]。

2.2.3 Qt/Embedded

Qt/Embedded^[10] 是 Trolltech 公司开发的面向嵌入式系统的 Qt 版本。Qt/Embedded 和 Qt/X11 类似，它的类库采用 C++ 封装，包含丰富的控件资源，有良好的可移植性。许多基于 Qt/X11 的程序可以非常方便地移植到 Qt/Embedded 上。Qt/Embedded 和 X11 版本不同的地方在于，Qt/Embedded 采用 FrameBuffer(帧缓冲)作为底层图形接口(如图 2.3 所示)。使用 Qt/X11 下的开发工具 Qt Designer 可以直接开发基于 Qt/Embedded 的用户操作接口界面。从 4.0 版本之后，Qt/Embedded 称为 Qtopia-Core。基于 Qt/Embedded 的特点，越来越多的第三方软件公司也开始采用 Qt/Embedded 开发嵌入式 Linux 下的应用软件^[6]。

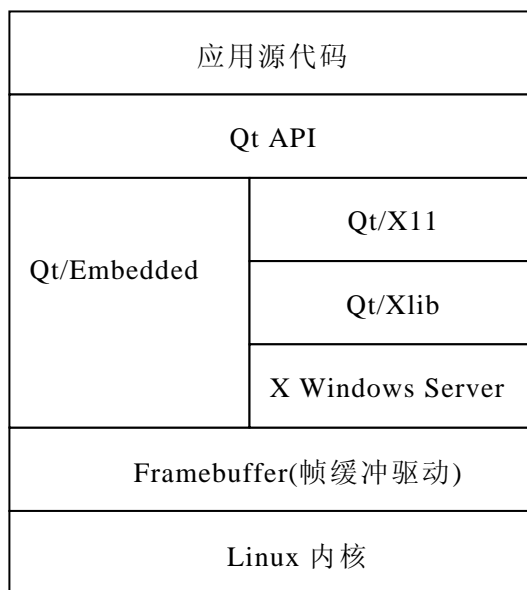


图 2.3 Qt/Embedded 与 Qt/X11 的 Linux 版本的比较

Fig. 2.3 Comparison of Qt/Embedded and Qt/X11 in Linux version

2.2.4 嵌入式 GUI 性能分析

综上所述对嵌入式 GUI 各方面的性能分析，表 2.1 列出了几种嵌入式 GUI 的性能比较^[11]，但是在具体选择使用那种嵌入式 GUI 还要根据操作的硬件平台和根据具体的实际情况进行选择。

表 2.1 嵌入式 GUI 的性能分析

Tab. 2.1 Performance analysis of embedded GUI

项目	MiniGUI	OpenGL	Qt/Embedded
API	Win32 风格	X、Win32 风格	Qt (C++)
API 是否完备	是	Win32 不完善	是
函数库大小	500KB	600KB	1.5MB
可移植性	很好	很好	较好
授权	GPL/商业许可证	MPL/LGPL	QPL/GPL/商业许可证
多进程支持	好	X 支持好，Win32 不支持	好
健壮性/稳定性	好	很差	差
多语种支持	多字符集	一般	Unicode，效率低

可定制性和可配置性	好	一般	差
消耗系统资源	小	较大	最大
效率	好	较差	差
支持的操作系统	Linux/uClinux、 uC/OS、VxWorks	Linux	Linux
支持的硬件平台	X86、ARM、 MIPS、PowerPC	X86、ARM、MIPS	X86、ARM
主要应用区域	中国	美国	欧美、韩国

2.3 嵌入式系统的硬件平台设计

本系统采用 Mini2440 开发平台，它是以 Samsung S3C2440 为微处理器，并采用专业稳定的 CPU 内核电源芯片和复位芯片来保证系统运行时的稳定性。并配备了实用的外设资源，如外围设备：串口；USB Host、USB Slave；16 位 TFT LCD；触摸屏；DM9000 网卡；UDA1341 音频解码器；128MB Nand Flash；64MB SDRAM。它具有高性能、低功耗的优点。Mini2440 开发平台如图 2.4 所示：

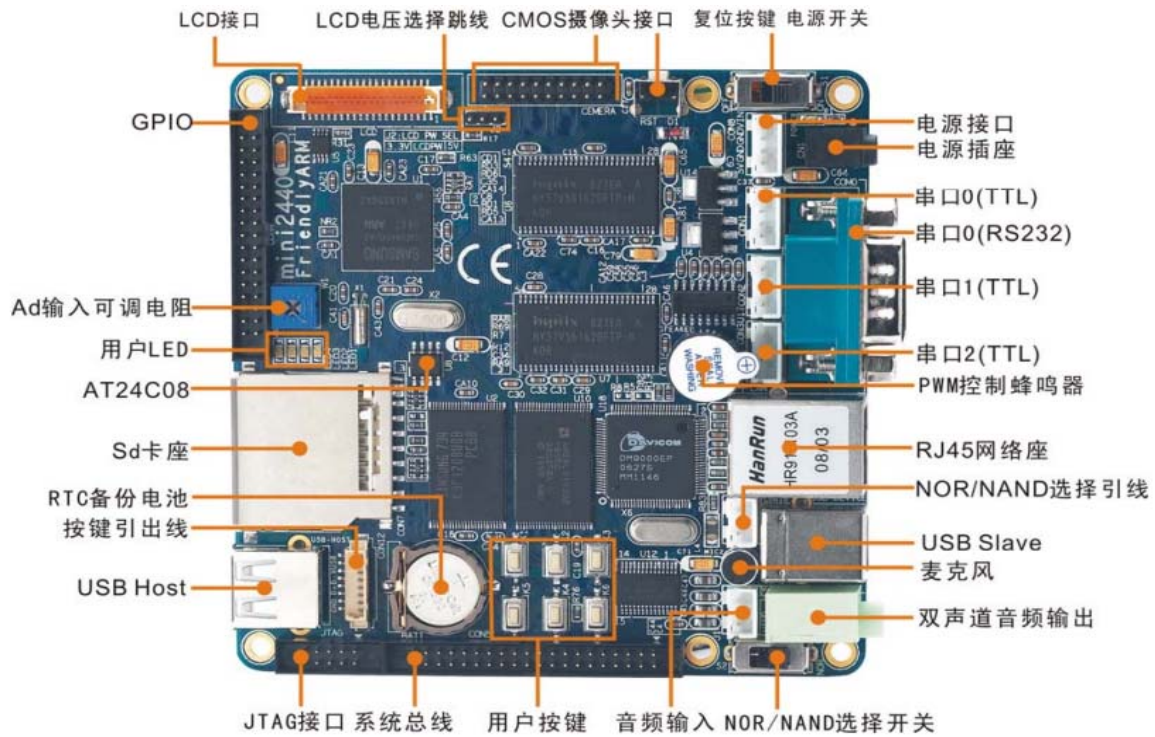


图 2.4 Mini2440 开发平台实物图

Fig. 2.4 Mini2440 development platform physical diagram

S3C2440A 的杰出的特点是其核心处理器（CPU），是一个由 Advanced RISC Machines 有限公司设计的 16/32 位 ARM920T 的 RISC 处理器。ARM920T 实现了 MMU, AMBA BUS 和 Harvard 高速缓冲体系结构。这一结构具有独立的 16KB 指令 Cache 和 16KB 数据 Cache。每个都是由具有 8 字长的行组成。通过提供一套完整的通用系统外设，S3C2440A 减少整体系统成本和无需配置额外的组件。S3C2440A 芯片的内部逻辑结构和 Mini2440 系统结构框图^[13]如下图所示：

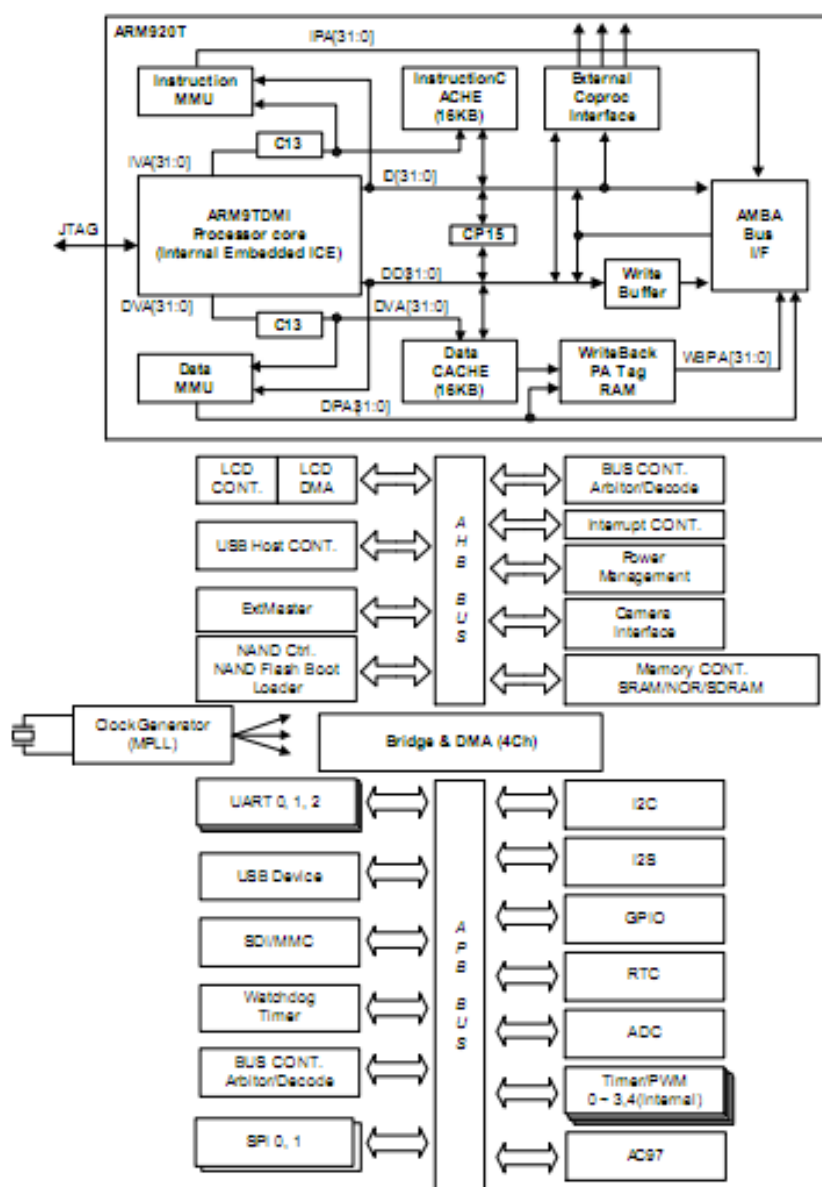


图 2.5 S3C2440A 芯片内部结构示意图

Fig 2.5 Internal structure diagram of S3C2440A chip

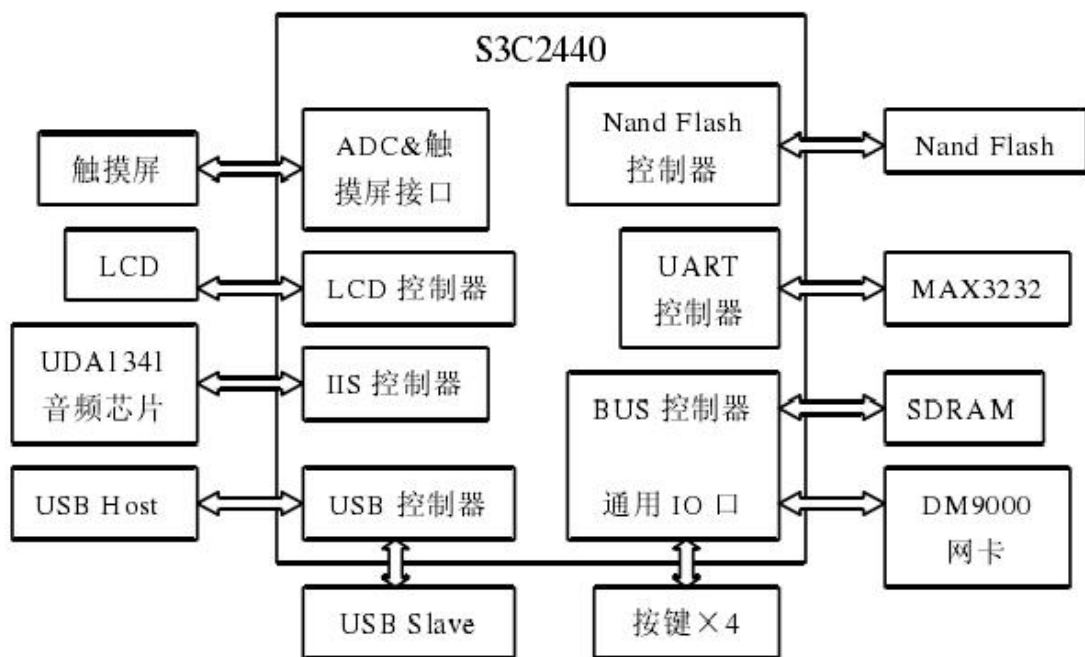


图 2.6 Mini2440 系统结构框图

Fig. 2.6 Mini2440 system structure diagram

2.4 嵌入式系统的软件平台设计

2.4.1 交叉编译环境建立

嵌入式系统软件采用交叉编译的方式，在宿主机上进行交叉编译生成可执行文件，然后放在目标机上运行，所以宿主机和目标机要进行文件系统的共享。当宿主机和目标机都使用 Linux 操作系统时，可以配置 NFS 服务来实现信息的共享。

NFS(Network File System)服务可以将远程文件系统载入到本地文件系统下，远程的硬盘、目录和光驱都可以变成本地主机目录树下的一个子目录。载入后于处理自己的文件系统一样使用即可。这样处理不仅方便，而且也节省了重复保存文件的空间、传输文件的时间和网络带宽^[11]。如何配置 NFS 服务主要步骤为如下所示：

(1) 关闭防火墙。

选择“系统设置”选择“安全级别配置”，将“安全级别”设置为无防火墙。

(2) 设置目标主机的 IP 地址。

目标主机的 IP 地址只要和 NFS 服务器的 IP 地址在同一个网段即可。

（3）配置 NFS 服务器。

选择“系统设置”选择“服务器设置”选择“NFS 服务”，进行 NFS 服务器配置。单击“增加”按钮，添加 NFS 共享。在“基本”选项卡中填写共享的目录和主机 IP 地址，主机的基本权限选择读/写。

（4）设置 NFS 服务器的 IP 地址。

通过命令 `ifconfig eth0` 来设置 NFS 服务器的 IP 地址。

（5）启动 NFS 服务。

```
#/etc/init.d/nfs start
```

（6）挂载 NFS。

通过 `mount` 命令挂载 NFS，挂载以后，进入 `mnt` 目录就可以看到宿主机目录下的所有文件了。

这里一般都是把进行交叉编译的主机叫做宿主机，也就是普通的通用计算机，而把程序实际的运行环境称为目标机，也就是嵌入式系统环境。如图 2.7 所示：

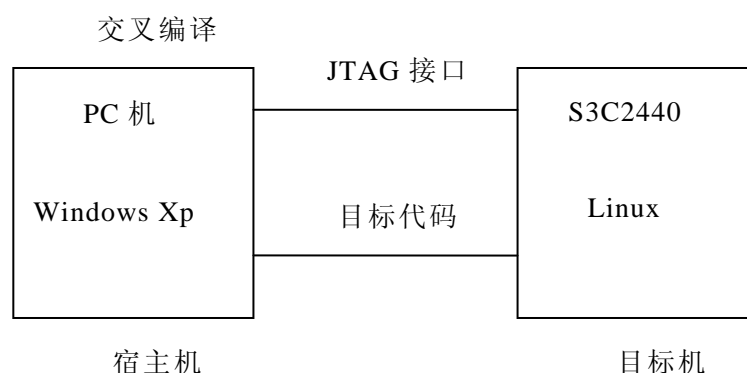


图 2.7 嵌入式交叉编译环境的硬件配置

Fig. 2.7 Hardware configuration of embedded cross-compiler environment

嵌入式交叉编译环境的软件配置包括 RedHat Linux 9.0、Vmware、软件开发工具包和交叉编译工具包。嵌入式软件最后是要运行在 ARM-Linux 平台上的，但是在开发过程中，首先在 Linux 平台上进行初步的调试，Linux 平台的搭建主要借助 Vmware 构建虚拟的 RedHat Linux 9.0，在其上安装交叉编译工具包及其开发所需的工具包。程序初步调试成功以后，利用交叉编译工具进行交叉编译，然后挂载到开发板上运行。

在 Redhat 9.0 里面建立一个能编译 arm-linux 内核及驱动、应用程序等开发环境的步骤如下。arm-linux-gcc-4.3.2.tgz 复制到目录下\tmp，然后进入到该目录，执行解压命令：

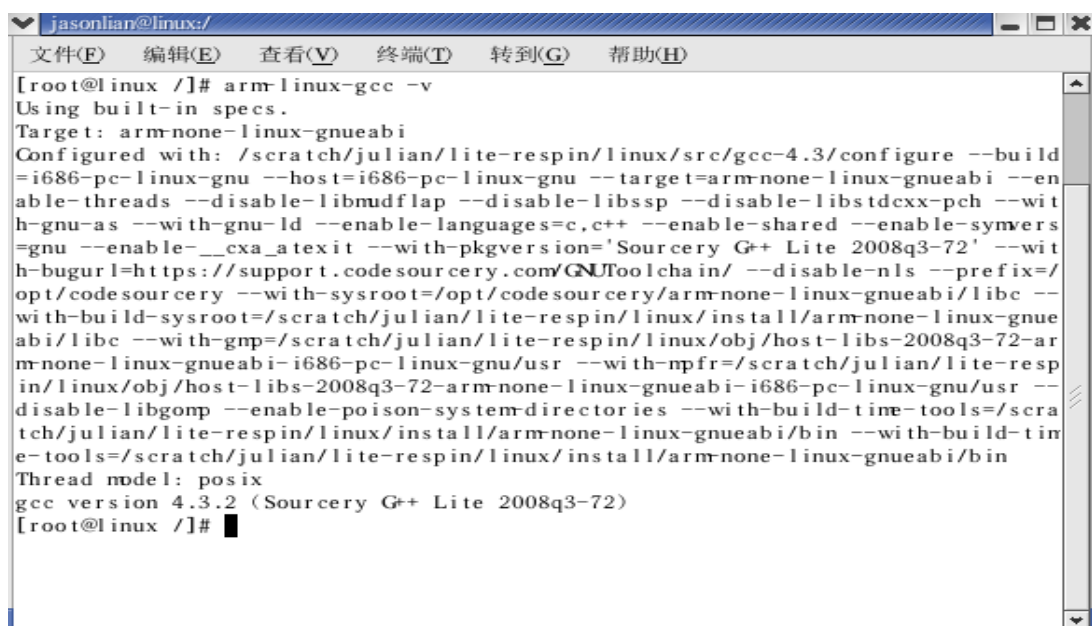
```
#cd \tmp
```

```
#tar xvzf arm-linux-gcc-4.3.2.tgz - C /
```

执行该命令，将把 arm-linux-gcc 安装到/usr/loca/arm/4.3.2 目录。把编译器路径加入系统环境变量，运行命令

```
#vim ~/.bashrc 编辑/root/.bashrc 文件，在最后一行添加如下的代码 export PATH=$PATH:/usr/local/arm/4.3.2/bin 并保存退出。
```

重新登录系统使以上设置生效，在命令行输入 arm-linux-gcc -v，如果能看到屏幕上打印的 gcc 版本信息就表示交叉编译器编译安装成功。这说明交叉编译环境已经成功安装。如图 2.8 所示：



```
jasonlian@linux:/  
文件(E) 编辑(E) 查看(V) 终端(T) 转到(G) 帮助(H)  
[root@linux /]# arm-linux-gcc -v  
Using built-in specs.  
Target: arm-none-linux-gnueabi  
Configured with: /scratch/julian/lite-respin/linux/src/gcc-4.3/configure --build=i686-pc-linux-gnu --host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi --enable-threads --disable-libmudflap --disable-libssp --disable-libstdcxx-pch --with-gnu-as --with-gnu-ld --enable-languages=c,c++ --enable-shared --enable-symvers=gnu --enable-__cxa_atexit --with-pkgversion='Sourcery G++ Lite 2008q3-72' --with-bugurl=https://support.codesourcery.com/GNUToolchain/ --disable-nls --prefix=/opt/codesourcery --with-sysroot=/opt/codesourcery/arm-none-linux-gnueabi/libc --with-build-sysroot=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/libc --with-gmp=/scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu/usr --with-mpfr=/scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu/usr --disable-libgomp --enable-poison-system-directories --with-build-time-tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin --with-build-time-tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin  
Thread model: posix  
gcc version 4.3.2 (Sourcery G++ Lite 2008q3-72)  
[root@linux /]#
```

图 2.8 交叉编译环境安装成功

Fig.2.8 A successful installation of cross-compiler environment

2.4.2 建立引导加载程序

操作系统的启动是离不开引导程序的，嵌入式操作系统也如此。因此，需要向嵌入式开发板移植加载引导程序，即 BootLoader。用户可以自己编写系统引导程序，

但由于引导程序是最贴近硬件的软件，所以要求编写人员非常熟悉相应的硬件平台。一般可以从网络上下载一些开源代码的 BootLoader 程序，常见的 BootLoader 如 U-Boot、BLOB、VIVI、LILO、ARM-Boot、RedBoot 等，根据自己的芯片进行移植修改，这也是目前最常采用的方法。

引导加载程序（BootLoader）是系统加电后运行的第一段代码，是在操作系统内核运行之前运行的一段小程序。通过这段小程序，可以初始化硬件设备，并建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，为最终调用系统内核准备好合适的环境。BootLoader 程序相当于 PC 中的引导加载程序 BIOS 和位于主引导记录（MBR）中的系统引导程序（对于 Linux 来说是 LILO）。

BootLoader 作为嵌入式的引导加载程序，主要完成下面的功能^[11]：

- （1）初始化硬件设备，如 CPU 的主频、SDRAM、中断和串口等；
- （2）准备内核启动参数；
- （3）启动内核；
- （4）与主机进行交互，从串口、USB 口或网络接口下载映像文件，并对 Flash 等存储设备进行管理。

本开发板采用的 BIOS 是基于三星原来的 bootloader 之 vivi 改进而来，名为 Supervivi，它采用功能菜单的方式，并可以和原来的命令交互模式互相切换。在保留原始 vivi 功能的基础上，整合了诸多其他实用功能，如支持 CRAMFS，YAFFS 文件系统，USB 下载，自动识别并启动 Linux，WinCE, uCos, Vxwork 等多种嵌入式操作系统，下载程序到内存中执行，并独创了系统备份和恢复功能，是目前 2440/2410 系统中功能最强大最好用的 BootLoader。

因为引导程序是应用最先被烧写到开发板上的程序，Supervivi 的菜单模式主要为烧写系统和调试而用，也可以设置参数和进行分区等，它采用 USB 下载的方式，因此搭建烧写环境极为简单，并且下载速度快，使用十分方便。在这里就不做详细的介绍了。

2.4.3 Linux 内核移植

如果在嵌入式开发板上使用操作系统，就必须为它制定一个适合于开发板的内核。通常都是从网上下载一个合适的内核版本。下载后根据自己的系统要求裁减、配置，在添加自己的特定硬件的驱动程序，进行调试修改，最终得到一个适合于自己平台的内核，将其进行交叉编译，就可以得到符合要求的内核映像文件，最后将其下载到 Flash 存储器芯片的相应分区中运行。

由于嵌入式系统存储单元十分有限，精简内核显得尤为重要。内核裁减主要是去掉无用模块，增加所需要模块，使之更符合目标系统。这主要包括：选择硬件平台的类型、选择内核对 MTD 和 Flash 存储器的支持、选择内核对网络的支持、选择内核对文件系统的支持等。

Linux 操作系统的移植主要包含内核移植、根文件系统的制作两部分。Linux 核心源程序的文件是按树形结构进行组织的，在源程序树的最上层，即目录 `/usr/src/linux` 下有这样一些目录和文件。以下是 linux 内核的文件组织结构^[6]：

`/include` 子目录包含编译建立内核代码时所需要的大部分库文件，这个模块利用其他模块重建内核。该目录也包括不同平台需要的库文件。

`/arch` 子目录包含了所有硬件体系结构相关的内核代码。它的每一个子目录都代表一种支持的体系结构，如：`arm`、`i386`、`alpha`。

`/drivers` 子目录包含了内核中所有的设备驱动程序。

`/fs` 所有的文件系统代码和各种类型的文件操作代码，它的每一个子目录支持一个文件系统。如 `vfat`，`ext3` 等。

`/init` 这个目录包含了内核的初始化代码（并不是系统的引导代码）。

`/ipc` 这个目录包含了进程间通信的代码。

`/kernel` 子目录包含了主内核的代码。

`/mm` 这个目录包含了所有独立于 `cpu` 体系结构的内存管理代码。如页式存储管理内存的分配和释放等等。

`/net` 子目录包含了和网络相关的代码。

Linux 内核的编译选项有如下几种^[11]^[6]：

- (1) `make config`: 进行命令行，可以一行一行地配置。
- (2) `make menuconfig`: 开发人员比较熟悉的 `menuconfig` 配置菜单。
- (3) `make xconfig`: 在 2.4X 以及以前版本中的 `xconfig` 图形配置菜单。

通常情况下，第二种方式（即 `menuconfig`）最稳定，也是开发人员最常用的配置方式。

具体的裁剪和配置步骤如下：

- (1) 进入内核目录。

首先创建工作目录 `/opt/FriendlyARM/mini2440` 。

下载 Linux 内核源代码：

从 <http://www.kernel.org/pub/linux/kernel/v2.6/> 下载并解压安装 Linux 内核源代码，在工作目录 `/opt/FriendlyARM/mini2440` 中执行：

```
#cd /opt/FriendlyARM/mini2440
#tar xvzf /tmp/linux/ linux-2.6.29.4.tar.bz2
```

将创建生成 `linux-2.6.29` 目录，里面包含了完整的 linux 内核源代码。

(2) 执行 `make menuconfig` 命令，进入配置菜单，对菜单选项进行选择 and 配置。配置方法如下：

```
#make menuconfig
System Type—>
[*]S3C2410 DMA support
S3C2440 Machines —>
[*]FriendlyArm Mini2440/QQ2440 development board
Device Drivers —>
Graphics support—>
<*>Support for framebuffer devices —>
<*> S3C2410 LCD framebuffer support
LCD select(3.5 inch 240*320 NEC LCD) —>
(X)3.5 inch 240*320 NEC LCD
Input device support —>
[*]Touchscreens —>
```

<*>Samsung S3C2410 touchscreen input driver
 [*]HID Devices —>
 <*>USB Human Interface Device (full HID) support
 SCSI device support —>
 <*>SCSI disk support
 [*]USB support—>
 <*>USB Mass Storage support
 [*]Networking support—>
 [*]TCP/IP networking
 [*]Network device support—>
 [*]Ethernet(10 or 100Mbit) —>
 -* Generic Media Independent Interface device support
 <*>DM9000 support
 Character devices—>
 <*>LED Support for Mini2440/QQ2440 GPIO LEDs
 <*>Memory Technology Device (MTD) support —>
 [*]MTD partitioning support
 <*>NAND Device Support—>
 <*>NAND Flash support for S3C2410/S3C2440 SoC
 File systems —>
 [*]Miscellaneous filesystems—>
 <*>YAFFS2 file system support
 [*]Autoselect yaffs2 format
 [*]Network File Systems —>
 <*>NFS client support
 [*]NFS client support for NFS version 3
 [*]Root file system on NFS

其中有几点需要说明：

(1) 带有“—>”的表示该选项包含子选项。

(2) 每个选项前面有[]或<*>，括号表示仅有两种选择（*或空），尖括号表示有 3 种选择（M、*、空），按空格键可切换这几种选择。

(3) M 表示以模块方式编译进内核，在内核启动后，需要手工执行 insmod 命令才能使用该项驱动：*表示直接编译进内核：空表示不编译进内核。

进行完内核裁减和配置以后，即可对配置后的内核进行交叉编译。具体的编译步骤如下：

输入#make zImage 命令，生成经压缩以后的内核映像文件 zImage。

编译结束后，会在 arch/arm/boot 目录下生成 linux 内核映象文件：zImage

linux 内核的移植需要修改的主要有根目录下的 Makefile 文件，我们知道 Linux 内核是根据 Makefile 的指示来进行编译的，它用来组织内核的各个模块，是记录各个模块间的相互依赖的关系，可以在这个文件中指定使用的编译器。因为我们针对的 arm-linux 平台，所以修改 Linux 内核的 Makefile 文件，设置 CPU 体系结构和交叉编译工具。

#vi Makefile

找到 ARCH 和 CROSS_COMPILE，修改如下：

ARCH：指定目标平台，需改为 arm；

CROSS_COMPILE：指定交叉编译工具，改为 arm-linux-；

然后设置 PATH 环境变量，让它能够使其找到交叉编译工具链。

#echo \$PATH

/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/jasonlian/bin:/usr/local/arm/4.3.2/bin

#vi ~/.bashrc

export PATH=\$PATH:/usr/local/arm/4.3.2/bin

修改对 nand 的分区信息。要让内核知道 nand flash 的分区信息，设置成跟 bootloader 一致。

在 arch/arm/plat-s3c24xx/common-smdk.c 中修改 smdk_default_nand_part[]，注意这个一定要跟 bootloader 的一致。在我的板子中修改如下：

```
static struct mtd_partition smdk_default_nand_part[] = {
    [0]={
        .name = "boot",
        .size = SZ_128K + SZ_64K,          //192k 0x0030000
```

```

        .offset = 0,
    },
    [1]={
        .name = "kernel",
        .size = SZ_16K * 116,          //2M - 192K   0x01e00000
        .offset = SZ_16K *12,          //192K
    },
    [2]={
        .name = "rootfs",
        .offset = SZ_2M,
        .size = SZ_2M * 15,             //30M
    },
    [3]={
        .name = "User",
        .offset = SZ_32M,
        .size = SZ_32M,
    }
};

```

name:表示分区名字。

size:表示 flash 分区大小（单位：字节）。

offset:表示 flash 分区的起始地址（代表 0X0 的偏移量）。

由于 Linux 内核比较庞大，所以在这里不做过多陈述。

2.4.4 建立根文件系统

有了 Linux 内核，还需要建立文件系统，对于 Linux 来说，就是根文件系统。在嵌入式系统中，有一个非常重要的创建根文件系统的工具——BusyBox，它能产生一个最基本的根文件系统。

Busybox 是一个轻型的 linux 命令工具集，在此使用的是 busybox-1.13.3 版本。我们可以从 <http://www.busybox.net> 下载到最新版本。根文件系统的构建过程可以通过命令进行建立，在宿主机上创建一个目录作为目标系统的根目录，命令为：

```
#mkdir rootfs
```

在“rootfs”目录下建立各个子目录，命令为：

```
#mkdir bin sbin dev etc lib root tmp usr var proc mnt
```

解压安装 busybox 源代码：

在工作目录/opt/FriendlyARM/mini2440 中执行：

```
#cd /opt/FriendlyARM/mini2440
```

```
#tar xvzf /tmp/linux/busybox-1.13.3.tgz
```

```
#cd busybox-1.13.3
```

修改 Makefile，

```
#vi Makefile
```

将 Makefile 中的 CROSS_COMPILE ?=

将其修改为：

```
CROSS_COMPILE ?= arm-linux-
```

```
ARCH=arm
```

```
#make defconfig
```

```
#make menuconfig
```

设置静态编译方式

```
Busybox Settings --->
```

```
Build Options --->
```

```
[*] Build BusyBox as a static binary (no shared libs)
```

```
Installation Options-->
```

```
[*]Don't use/usr
```

退出并保存配置

编译：

```
# make
```

安装：

```
# make install
```

此外还要注意用 `chmod 777 linuxrc` 改变 linuxrc 属性，否则可能无法执行。

根文件系统的目录及其作用如表 2.2 所示：

表 2.2Linux 常用目录及其作用

Tab.2.2 Linux common directories and its function

目录名称	功能与作用
/bin 和/sbin	对 Linux 系统进行维护操作的实用命令基本上都包含在/bin和/sbin中。/bin目录经常存放最常用的一些基本命令，包括对目录和文件操作的一些实用程序、系统实用程序、压缩工具、RPM包管理程序等。/sbin目录中存放的是只允许系统管理员运行的一些系统维护程序，即只有用户 root 登录后，才能执行/sbin下的命令。
/boot	用于存放与系统启动相关的各种文件，包括系统的引导程序使用的静态文件和系统内核程序。
/dev	用于存放设备文件及其他特殊文件。
/etc	用于存放系统管理时要用到的各种配置文件，包括网络配置、设备配置信息、用户信息等。
*/home	系统中所有普通用户的宿主目录，新建用户账户后，系统会自动在该目录中创建一个与账户同名的子目录。
/lib	用于存放系统的动态链接库，几乎所有的应用程序都会用到这个目录下的共享库，如 C 链接库、内核模块。
/mnt	用于存放临时挂载的文件系统的挂载点，如 CD-ROM、软盘、U 盘之类的设备的挂载点都在此目录下。
*/opt	附加软件的安装目录。
/proc	内容是系统自动产生的，其内容是当前系统运行的进程的一个虚拟镜像，以及记录当前内存内容的 kernel 文件。
*/root	root 用户主目录
/tmp	用于存放临时文件，如程序执行期间产生的临时文件。
/var	用于存放经常变化的文件，对于存取频繁或内容经常变化的文件，可放在该目录下。
/usr	存放与用户直接相关的程序或文件。用户安装的程序或自行建立的目录，一般应该放在该目录下，它是占用硬盘空间最大的一个目录。

2.5 嵌入式 GUI 的平台设计

2.5.1 Qt/Embedded 架构

Qt/Embedded 是挪威 Trolltech 公司的图形化界面开发工具 Qt 的嵌入式版本，Qt/Embedded 以原始的 Qt 为基础，并做了许多出色的调整以适用于嵌入式环境。它通过 Qt API 与 Linux I/O 设施以及 Framebuffer 直接交互，具有较高的运行效率，而

且整体采用面向对象编程，拥有良好地体系架构和编程模式^[12]。同 Qt/X11 相比，Qt/Embedded 很省内存，因为它不需要一个 X 服务器或是 Xlib 库，它在底层摒弃了 X lib，采用 Framebuffer（帧缓冲）作为底层图形接口。同时，将外部输入设备抽象为 keyboard 和 mouse 输入事件。Qt/Embedded 的应用程序可以直接写到内核缓冲帧，这避免开发者使用繁琐的 Xlib/Server 系统。

Qt/Embedded 的底层图形引擎基于 FrameBuffer^{[11][12][14]}。FrameBuffer 是 Linux 推出的标准显示设备接口，它将显示设备抽象为帧缓冲区。在 Linux 下对应的设备节点为 /dev/fb0，使用 mmap 系统调用，对用户而言，它和 /dev 下的其他设备没有什么区别，用户可以将 FrameBuffer 的显示缓存映射为可连续访问的一段虚拟内存空间，既可以从这块内存中读取数据，也可以向其中写入数据，而写操作立即反应在屏幕上。Qt/Embedded 的实现结构如下图所示。

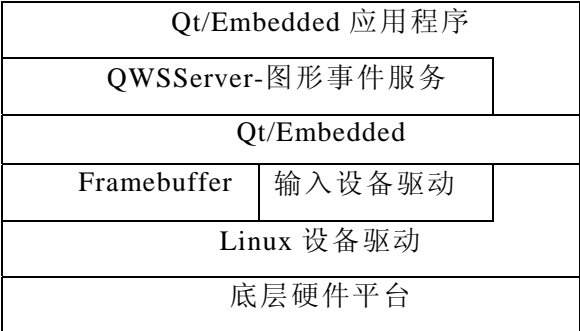


图 2.10 Qt/Embedded 的实现结构

Fig. 2.10 Implementation structure of Qt / Embedded

目前比较高级的嵌入式 CPU 中大多集成了 LCD 控制模块。FrameBuffer 驱动程序的实现分为两个方面：一方面是对 LCD 及相关硬件的初始化，包括图形在缓冲区的创建和对 DMA 通道的设置；另一方面是对画面缓冲区的读写，具体到代码为 read、write、lseek 等系统调用接口。至于将画面缓冲区的内容输出到 LCD 显示屏上，则由硬件自动完成。对于软件来说是透明的。

在 Qt/Embedded 中，Qscreen 类为抽象出的底层显示设备基类，其中声明了显示设备的基本描述和操作方式，如打开、关闭、获得显示能力等。另外一个重要的基类是 QGfx 类，该类抽象出显示设备的具体操作接口，如选择画刷、画线等操作。

以上两个基类是 Qt/Embedded 图形引擎的底层抽象。其中，所有具体函数基本都是虚函数，Qt/Embedded 对于具体的显示设备，如 Linux 的 FrameBuffer、Qt Virtual FrameBuffer 做的抽象接口类全部都由此继承并重载基类中的虚函数实现。

Qt/Embedded 在体系上为 C/S 结构，任何一个 Qt/Embedded 程序都可以作为系统中唯一的一个 GUI Server 存在。当应用程序首次以系统 GUI Server 的方式加载时，将建立 QWSServer 实体。此时调用 QWSServer::OpenDisplay() 函数创建窗体，设置 QScreen 指针 qt_screen，并调用 connet() 打开显示设备 /dev/fb0。在 QWSServer 中所有对于显示设备的调用都由 qt_screen 发起。当系统中建立好 GUI Server 后其他 Qt/Embedded 程序采用共享内存的进程通信方式获得对共享资源 FrameBuffer 设备的访问权。

2.5.2 信号与槽的机制原理

信号和槽机制^[16]是 Qt 的一个中心特征并且也许是 Qt 与其它工具包的最不相同的部分。信号与插槽机制提供了对象间的通信机制，它易于理解和使用，并完全被 Qt 图形设计器所支持。信号和槽机制是 Qt 编程的基础。它可以让应用程序人员把这些互不了解的对象绑定在一起。

在以前，当我们使用回调函数机制来把某段响应代码和一个按钮的动作相关联时，我们通常把那段响应代码写成一个函数，然后把这个函数的地址指针传给按钮，当那个按钮被按下时，这个函数就会被执行。对于这种方式，以前的开发包不能够确保回调函数被执行时所传递进来的函数参数就是正确的类型，因此容易造成进程崩溃，另外一个问题是，回调这种方式紧紧的绑定了图形用户接口的功能元素，因而很难把开发进行独立的分类^[15]。

Qt 的信号与插槽机制是不同的。Qt 的窗口在事件发生后会激发信号。例如一个按钮被点击时会激发一个 “clicked” 信号。程序员通过建立一个函数（称作一个插槽），然后调用 connect() 函数把这个插槽和一个信号连接起来，这样就完成了一个事件和响应代码的连接。信号与插槽机制并不要求类之间互相知道细节，这样就可以相对容易的开发出代码可高重用的类。信号与插槽机制是类型安全的，它以警告的方式报告类型错误，而不会使系统产生崩溃^[11]。信号与槽的抽象图如下所示：

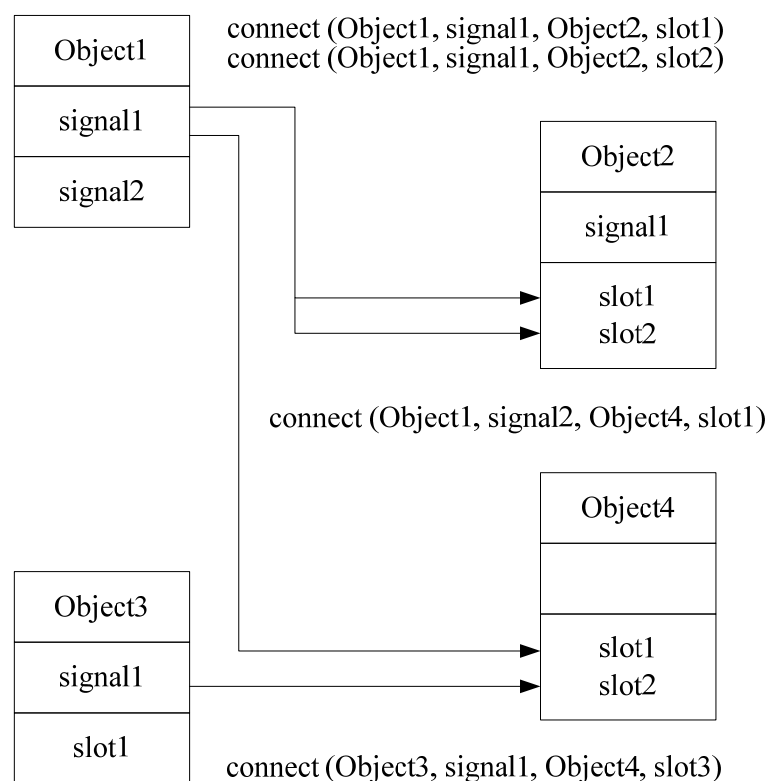


图 2.11 信号与槽的连接原理如图

Fig. 2.11 Connection of signal and slot schematic

所有从 `QObject` 或其子类（例如 `QWidget`）派生的类都是能够包含信号和槽机制的。当对象改变状态时，信号就由该对象发射（`emit`）出去了，这就是对象所要做的全部工作，它不知道另一端是谁在接收这个信号，这就是真正的信息封装，它确保对象被当做一个真正的软件组件来使用。槽用于接收信号，但它们是普通的对象成员函数。一个槽并不知道是否有任何信号与自己相连接。而且，对象并不了解具体的通信机制。

槽和普通的 C++ 成员函数几乎是一样的可以是虚函数、可以被重载、可以使公有的、保护的或者私有的，并且也可以被其他 C++ 成员函数直接调用；还有，它们的参数可以是任意类型。唯一的不同是：槽还可以和信号连接在一起，在这种情况下，每当发射这个信号的时候，就会自动调用这个槽^[14]。

`connect()` 语句看起来会是如下的样子：

```
connect(sender, SIGNAL(signal), receiver, SLOT(slot));
```

这里的 sender 和 receiver 是指向 QObject 的指针，signal 和 slot 是不带参数的函数名。

信号和槽的连接非常自由，甚至可以取消已经连接的信号和槽^[17]。总体可以归纳为以下几类：

(1) 一个信号可以连接多个槽。当信号被激发时，槽被一个接一个的调用，但是调用的顺序是不确定的。

(2) 多个信号可以连接到一个槽上。当任何一个信号被激发时，槽将被调用。

(3) 一个信号可以连接到另一个信号上。当第一个信号被激发时，第二个信号也被激发。

(4) 连接可以被删除，具体是使用 QObject::disconnect()函数。该方法很少被用到，因为对象被删除时 Qt 将自动删除和对象关联的连接。

2.5.3 Qt/Embedded 的移植

mini2440 开发板中提供包含了三个 Qt/Embedded 源代码包：

x86-qtopia.tgz: 用于在 X86 上运行的 Qtopia 开发包

arm-qtopia.tgz: 用于在 ARM 上运行的 Qtopia 开发包，支持 USB 鼠标

ipaq-qtopia.tgz: 用于在 ARM 上运行的 Qtopia 开发包，支持触摸屏

三个版本的不同仅在于各自目录中的 build 编译脚本，其他的源代码都是一样的。解压源代码：

把开发板提供的 x86-qtopia.tgz, arm-qtopia.tgz, ipaq-qtopia.tgz 复制到一个目录，进入该目录执行以下命令：

```
#tar xvzf x86-qtopia.tgz -C /opt/FriendlyARM/mini2440
```

```
#tar xvzf arm-qtopia.tgz -C /opt/FriendlyARM/mini2440
```

```
#tar xvzf ipaq-qtopia.tgz -C /opt/FriendlyARM/mini2440
```

该命令将把三种版本的 Qt/Embedded 包解压的到/opt/FriendlyARM/mini2440 目录。

为了在 PC 上模拟运行 Qtopia，需要用到对应 Qt 版本的库文件，因此需要修改/etc/ld.so.conf 文件以适应自己将要安装的 Qt 开发平台(Redhat 安装时带有 Qt 库)，修改后的 ld.so.conf 文件内容如下：

```
/opt/FriendlyARM/mini2440/x86-qtopia/qt/lib
/opt/FriendlyARM/mini2440/x86-qtopia/qtopia/lib
/usr/kerberos/lib
/usr/X11R6/lib
/usr/lib/sane
/usr/lib/mysql
```

因为配置编译 Qt/Embedded 的过程比较复杂，为了方便，我们把配置和编译的步骤制作成一个 build 脚本，执行该脚本即可“一键搞定”。

```
#cd /opt/FriendlyARM/mini2440/x86-qtopia
#./build-all
#./konq_to_qtopia
#ldconfig      (运行 ldconfig 是为了使生成的 qt 和 qtopia 库有效,运行一次即可)
```

在 PC 机上模拟运行 Qtopia

```
#. set-env
#qvfb&
#qpe
```

在 PC 上使用 QVFB 模拟运行 Qtopoia 如下图所示：



图 2.12 PC 上使用 QVFB 模拟运行 Qtopia

Fig. 2.12 Using QVFB to simulate Qtopia run on PC

2.5.4 嵌入式数据库 SQLite

嵌入式数据库是嵌入式系统的重要组成部分，是嵌入式软件系统开发的重要环节之一。而嵌入式数据库的产生使得在嵌入式系统中，对数据的管理更可靠、更稳定、更高效。

嵌入式数据库^[11]属于程序驱动式，而其他的数据库属于引擎相应式。嵌入式数据库通常不需要独立运行数据库的引擎，一般都是和操作系统及其具体的应用集成在一起的。只要在运行的操作系统平台上有相应的嵌入式数据库的驱动，应用程序直接调用相应的 API 去实现对数据的存取操作就可以了。因此，可以把嵌入式数据库理解为一种具备了基本数据库特性的数据文件。

嵌入式数据库^[19]以下一些特点：

(1) 体积小是嵌入式数据库的一个很重要的特点，嵌入式数据库编译后的产品也不过几十千的字节。占用内存小使得它可以支持嵌入式 Linux、Windows CE、Palm OS 等多种嵌入式操作系统。

(2) 可靠性是嵌入式数据库的另一个特点。对嵌入式数据库的操作、系统的大小、性能都应该可以预知，这样才能保证嵌入式系统在没有人看守的情况下，能够正常地、长时间的运行。

(3) 从目前嵌入式应用的发展趋势来看，嵌入式数据库的实现必须充分体现系统的可定制性，即系统选择的技术路线要面向具体的行业应用。因此，研究源代码开发的嵌入式数据库具有特殊的意义。

(4) 嵌入式数据库应支持 SQL 查询语言，提供数据库及数据表的管理功能，能够方便地实现对数据的操作，不需要花很多的时间来重新学习嵌入式数据库。

(5) 嵌入式数据库提供了接口函数，以供在高级语言中调用，可方便地实现对数据库的操作及管理。

(6) 嵌入式系统的实时性，使得嵌入式数据库的操作具有定时限制的特性。

(7) 管理嵌入式数据库的时候，要有一定的底层控制能力，因为嵌入式系统和底层的硬件环境密不可分，如对磁盘的操作等。底层控制的能力是决定数据库管理操作的关键。

SQLite 第一个 Alpha 版本诞生于 2000 年 5 月。至今已经有 9 个年头了。SQLite 也迎来了一个版本 SQLite 3 已经发布^[17]。

SQLite 是一款轻型的数据库，是遵守 ACID 的关联式数据库管理系统，它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百 K 的内存就够了。它能够支持 Windows/Linux/Unix 等等主流的操作系统，同时能够跟很多程序语言相结合，比如 Tcl、PHP、Java 等，还有 ODBC 接口，同样比起 Mysql、PostgreSQL 这两款开源世界著名的数据库管理系统来讲，它的处理速度比他们都快。

SQLite 的源代码是 C 语言形式的，并且完全的开放。SQLite 的特性^[21]如下：

- (1) 支持 ACID 事务。
- (2) 零配置，即无需安装和管理配置。
- (3) 储存在单一磁盘文件中的一个完整的数据库。
- (4) 数据库文件可以在不同字节顺序的机器间自由的共享。
- (5) 支持数据库大小至 2TB。

- (6) 程序体积足够小，大致 3 万行 C 代码, 250K。
- (7) 比一些流行的大多数数据库在对部分普通数据的操作更快捷。
- (8) 简单，轻松的 API 。
- (9) 包含 TCL 绑定，同时通过 Wrapper 支持其他语言的绑定。
- (10) 良好注释的源代码，并且有着 90% 以上的测试覆盖率。
- (11) 程序独立运行，没有额外依赖。
- (12) Source 完全的 Open，你可以用于任何用途，包括出售它。
- (13) 支持多种开发语言，C，PHP，Perl，Java，ASP.NET，Python。

SQLite 的 SQL 语言很大程度上实现了 ANSI SQL 92 标准，特别是支持视图、触发器事务，支持嵌套 SQL。它通过 SQL 编译器来实现 SQL 语言对数据库进行操作，支持大部分的 SQL 命令。

SQLite 最大的特点在于其数据类型为无数据类型。无论表中每列声明的数据类型是什么，SQLite 并不做任何检查，可以保存任何类型的数据到所想要保存的任何列中。开发人员主要靠自己的程序控制输入与输出数据的类型。这里有一个特例，当主键为整型值时，如果要插入一个非整型值就会产生异常。SQLite 允许忽略数据类型，但是，仍然建议在创建表时指定数据类型，有利于增强程序的可读性。目前 SQLite 的版本为 SQLite3。

2.5.5 SQLite3 交叉编译

(1) SQLite3 本地安装

由于 SQLite3 为开源代码，因此可以直接从网上下载免费的 SQLite3 源代码安装程序包，本机下载的 SQLite3 软件包版本为 `sqlite-3.5.4.tar.gz`。该版本的 SQLite 在本地安装的过程如下：

解压 SQLite：

把压缩包解压到当前目录/home 下，解压以后生成 `sqlite-3.5.4` 目录。

```
[root@linux home]#tar zxvf sqlite-3.5.4.tar.gz
```

```
[root@linux home]#cd /home/sqlite-3.5.4
```

进入解压后的源代码包中，建立文件夹 `sqlitetest`，进入 `sqlitetest` 文件夹。

```
[root@linux sqlite-3.5.4]#mkdir sqlitetest
```

```
[root@linux sqlite-3.5.4]#cd /home/sqlite-3.5.4/sqlitetest
```

配置 SQLite:

configure 时加--disabe-tcl 参数，屏蔽掉 tcl 库。配置成功后之后，会在当前目录下生成 Makefile 文件。

```
[root@linux sqlitetest]#./configure --disable-tcl
```

编译并安装 SQLite:

```
[root@linux sqlitetest]#make
```

```
[root@linux sqlitetest]#make install
```

在 make 和 make install 之后，库文件编译并安装在/usr/local/lib 目录下，可执行文件 sqlite3 安装在/usr/local/bin 目录下，头文件安装在/usr/local/include 目录下。在链接程序时，为了能找到库文件，需要把库文件所在路径加到系统文件/etc/ld.so.conf 中，如下图所示。在文件后面追加/usr/local/lib 一行内容。保存文件并退出，重新启动系统之后，设置生效，如果不想启动系统，可运行如下命令：

```
[root@linux sqlitetest]#/sbin/ldconfig
```

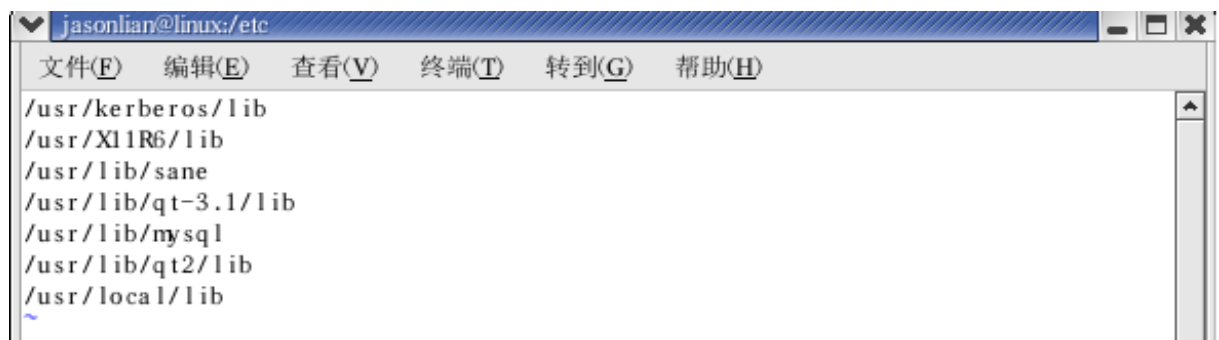


图 2.13 配置/etc/ld.so.conf 文件

Fig.2.13 Configure the file of / etc / ld.so.conf

为了验证 SQLite3 是否安装成功。可以创建一个新的数据库文件，名字为“test.db”，并测试数据库。

```
[root@linux sqlitetest]#sqlite3 test.db
```

如果出现下面的图像，表明编译安装已经成功了。如下图所示：

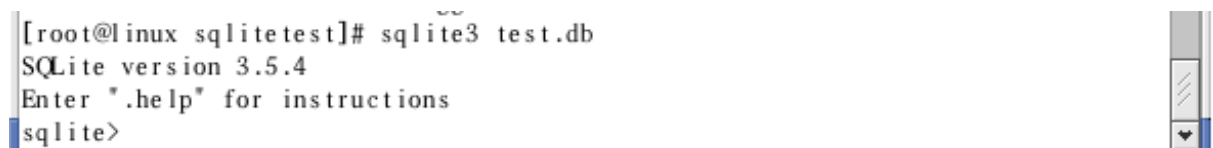


图 2.14 SQLite3 编译安装成功

Fig. 2.14 A successful installation of SQLite3

如果要把 SQLite3 运行在嵌入式体系上，则需要对 SQLite3 进行交叉编译。以编译在 ARM 体系运行的 SQLite3 为例，介绍相应过程。前面操作同本地安装，只是更改了配置信息。

(1) 更改配置

```
#../configure --disable-tcl --host=arm-linux-gcc --prefix=/usr/local
```

-prefix 参数指定安装的路径。在这里 /usr/local 为指定的安装 bin、lib、include 的路径。-host 参数指定交叉编译器。这里的 arm-linux-gcc 为交叉编译器。

(2) 交叉编译与安装

```
#make
```

```
#make install
```

交叉编译及安装之后，库文件、可执行文件、头文件就安装在指定的目录下。

2.6 本章小结

首先分析嵌入式系统的特点及嵌入式系统的设计流程。研究了目前嵌入式 Linux 下比较流行的几种嵌入式 GUI 系统，对嵌入式 GUI 的优缺点进行了比较。在搭建硬件平台同时也分析了 Linux 操作系统平台的构建过程和构建方法。首先介绍了 Linux 环境下如何建立交叉编译环境，接着在嵌入式软件平台的基础上分析研究了 BootLoader 的移植、Linux 内核配置和移植、驱动程序的加载、根文件系统的制作等。并对嵌入式图形开发工具 QT 以及 Qt/Embedded 的架构进行了深入分析，针对其特性分析了信号与槽的机制原理，研究了 Qt/Embedded 的移植以及开发平台。与此同时对嵌入式数据库进行了分析，研究了 SQLite3 的移植、安装过程。为下一章节的系统构建打好基础。

3 嵌入式码头物流管理手持终端的设计

3.1 码头物流管理界面程序的基本架构

根据功能的需求分析，设计构建了码头物流管理 GUI 的层次化人机界面结构，如下图所示。将码头物流管理主界面分为：系统管理、作业控制、箱务管理、船舶管理、堆场管理、泊位管理、入库管理、出库管理、手持终端等几大模块。其中系统管理模块负责系统中用户的建立和管理、用户权限的管理和用户角色的管理；作业控制模块根据生产作业计划、堆场情况和机械情况安排调度作业机械，监控生产作业过程；箱务管理主要对进去码头集装箱的相关信息进行管理。船舶管理用于记载集装箱船舶的数据管理，可方便地产生合理的生产作业计划；堆场管理可浏览、查询进场集装箱的场位，保障最大程度地利用堆场的空间；泊位管理针对船舶停靠信息、泊位计划进行管理，并对泊位使用情况进行统计分析；入库与出库管理主要是新建、查询、删除、编辑货场入库与出库的信息。手持终端模块是我们重点设计的模块将在下面详细说明。系统的框架图如下图所示：

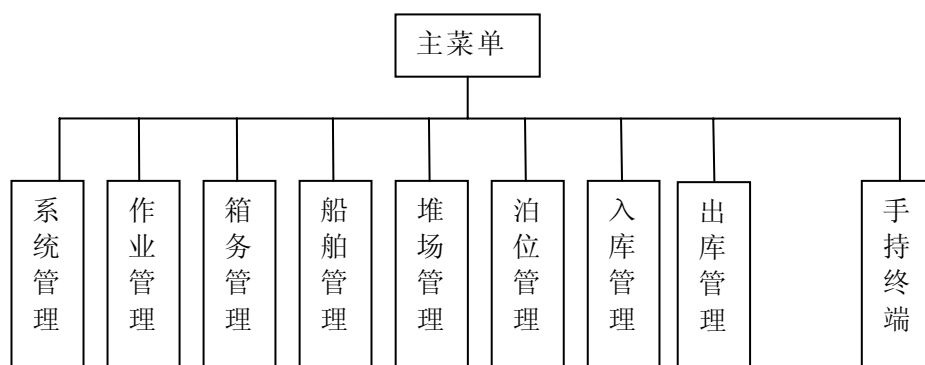


图 3.1 码头物流管理界面的主架构示意图

Fig. 3.1 The main structure diagram of Terminal Logistics Management Interface

3.2 码头物流管理手持终端界面设计

Qt 提供了非常强大的 GUI 编辑工具 Qt Designer，它的操作界面类似于 Windows 下的 Visual Studio，而且它还提供了相当多的部件资源。Qt Designer 图形设计器是一个具有可视化用户接口设计工具。在编写 Qt 源代码的时候，也同时生成了可执

行程序 designer。在 Linux 终端下，输入 designer 命令，就进入了 Qt Designer 的管理界面^[11]。如下图所示：

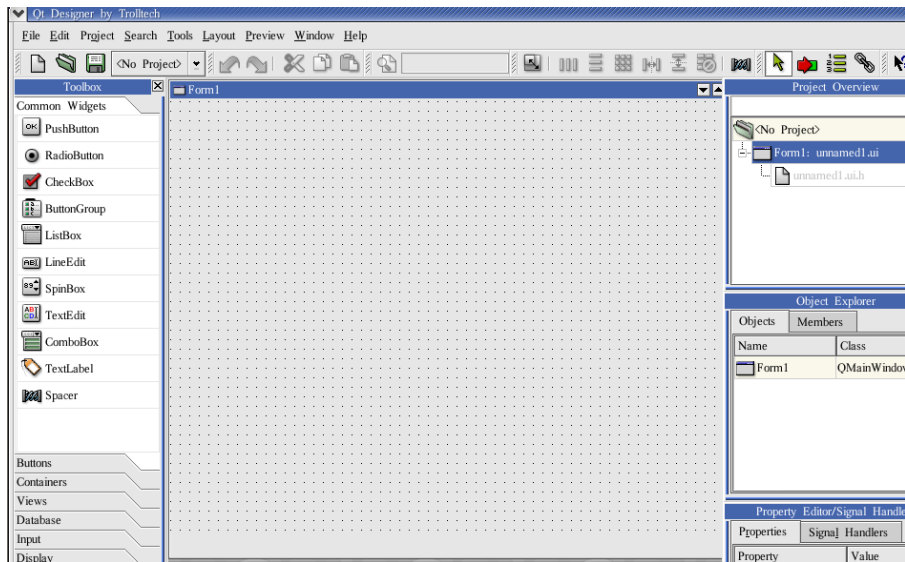


图 3.2 Qt Designer 可视化图形工具的启动界面

Fig.3.2 Graphical startup interface of vedio graphical tool

使用 Qt Designer 图形设计器进行图形用户接口的设计可以消除应用的编译、连接和运行时间，同时使用修改图形用户接口的设计变得更加的容易。Qt 图形设计器的预览功能可以使开发者能够在开发阶段看到各种样式的图形用户界面，也包括客户样式的用户界面。通过 Qt 集成功能的强大的数据库类，Qt Designer 还可以提供生动的数据库数据浏览和编辑操作。

Qt 图形设计器设计的图形界面以扩展名为 .ui 的文件进行保存，这个文件有良好的可读性，这个文件可被 uic（Qt 提供的用户接口编译工具）编译成为 C++ 的头文件和源文件。Qmake 工具在它为工程生成的 Makefile 文件中自动包含了 uic 生成头文件和源代码的规则。

3.2.1 手持终端功能框架设计

针对目前尚不完善的码头集装箱堆场管理的效率低下这一个管理现状，将嵌入式手持设备终端技术应用在码头物流的集装箱堆场管理当中。从而提高集装箱的周转效率和周转速度，提升集装箱堆场的管理水平，尽可能的提高堆场空间的利用效率。这一嵌入式手持终端主要是面向集装箱堆场的工作人员，使他们能过更好的管

理集装箱堆场，提高集装箱的调度效率。本论文结合这一管理现状设计对码头物流管理 PDA 手持终端的设计，下面介绍手持终端的框架示意图。

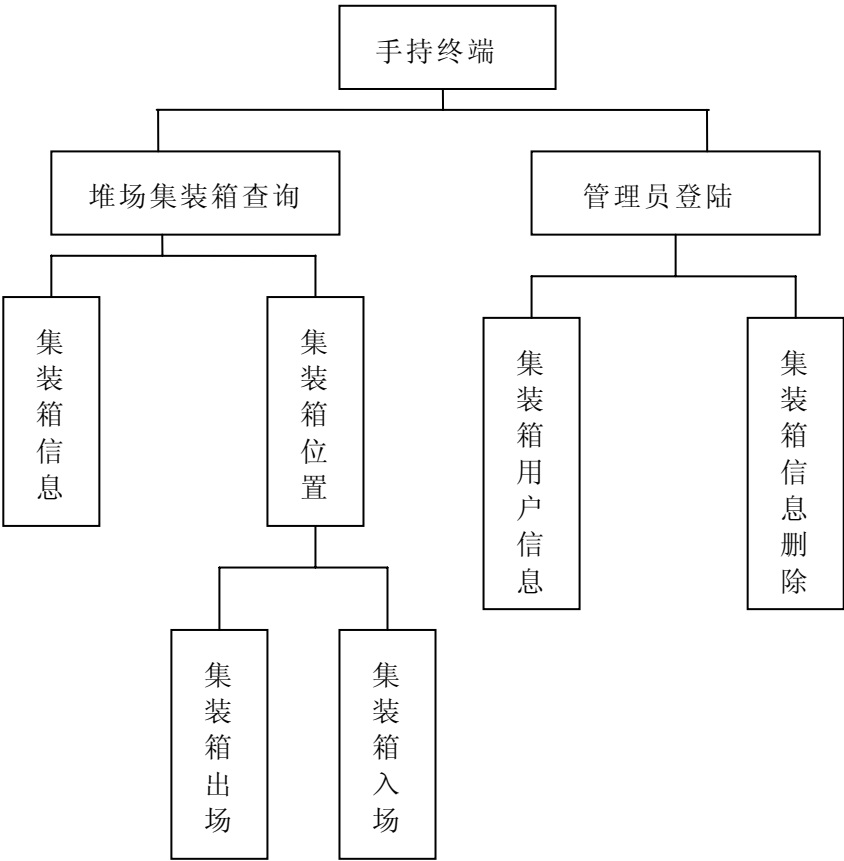


图 3.3 码头物流管理终端设计示意图
Fig.3.3 Design diagram of Terminal Logistics Management Terminal

针对码头物流集装箱管理的这一机制，提出手持终端来便于堆场工人对集装箱信息查询，位置摆放和集装箱出场区调度的管理。将整个的手持终端系统划分为两大功能模块，即堆场集装箱查询模块和管理员登录模块，同时针对堆场集装箱的查询模块下面又设有集装箱信息查询和集装箱位置查询功能模块。集装箱位置查询模块又设有集装箱出场查询模块和集装箱入场查询模块两大功能模块。同时管理员登录模块下设有集装箱用户信息模块，和集装箱信息删除模块。整个系统的功能模块设计如上图所示。

系统的系统结构为 C/S 架构，即客户机和服务器结构。它是软件系统体系结构，通过它可以充分利用两端硬件环境的优势，将任务合理分配到 Client 端和 Server 端

来实现，降低了系统的通讯开销。系统的数据库采用嵌入式数据库 SQLite，SQLite 是一款轻型的数据库，它的设计目标是嵌入式的，而且现在已经在很多的嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百 K 的内存就够了。

3.2.2 手持终端功能模块界面设计

下面我们来分析展现一下手持功能模块的各个界面，并对起各项功能进行分析。

首先是通过系统的验证功能信息进入集装箱信息查询手持终端界面。它主要有两大功能模块，即集装箱信息查询模块和管理员登录模块。堆场的工人可以通过其用户登录身份的验证，对堆场集装箱信息进行管理，下面是它的功能模块的界面设计图：



图 3.4 集装箱信息查询手持终端示意图

Fig.3.4 Container information query handheld terminal diagram

下面给出堆场集装箱管理者查询集装箱信息的时验证身份的数据流程图。堆场信息查询时针对堆场工人运输作业的时候对集装箱信息的一些查询工作。如下图所示：

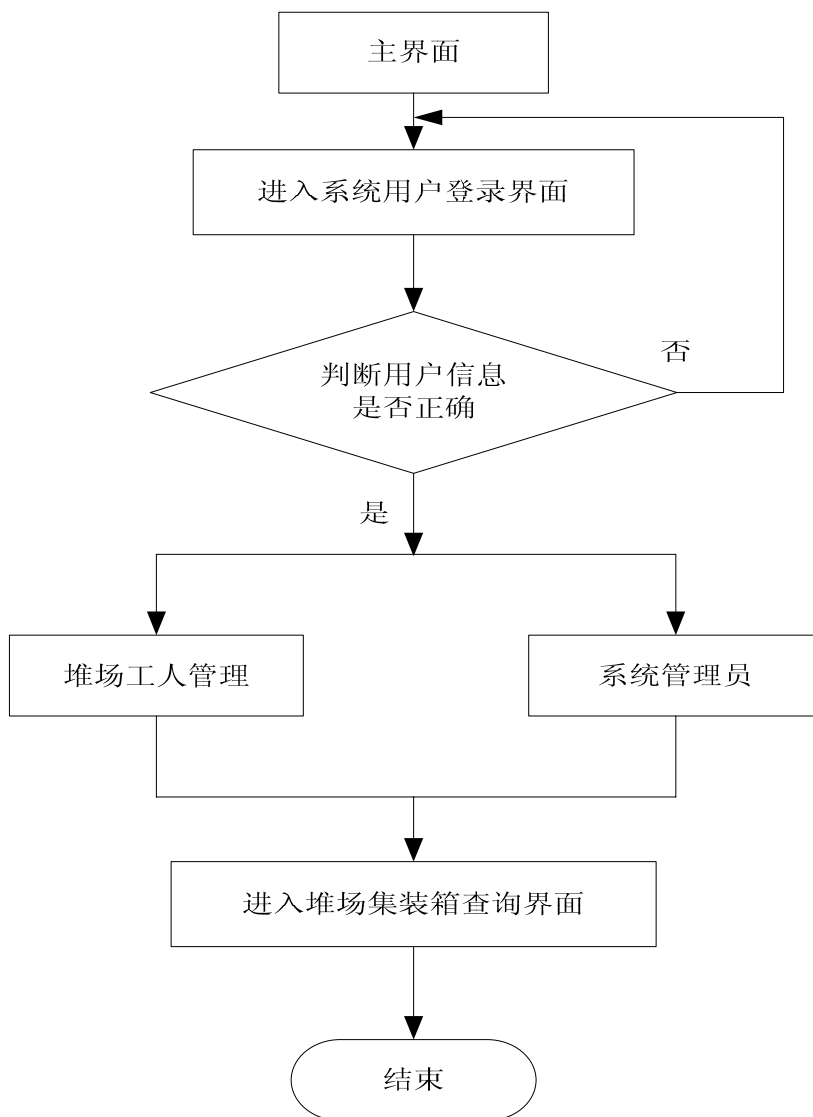


图 3.5 集装箱查询身份信息验证流程图

Fig.3.5 Container Query identity verification flowchart

集装箱堆场信息管理员登录的模块主要分为两大部分，在进入管理员登录界面后，每一个验证界面都对应的管理员的验证身份，都要输入正确的用户名和密码。才能通过对应的操作界面进行下一步的操作。根据集装箱手持终端管理员登录角色访问控制机制的这一特点，给出具体的验证过程流程图，如下图所示：

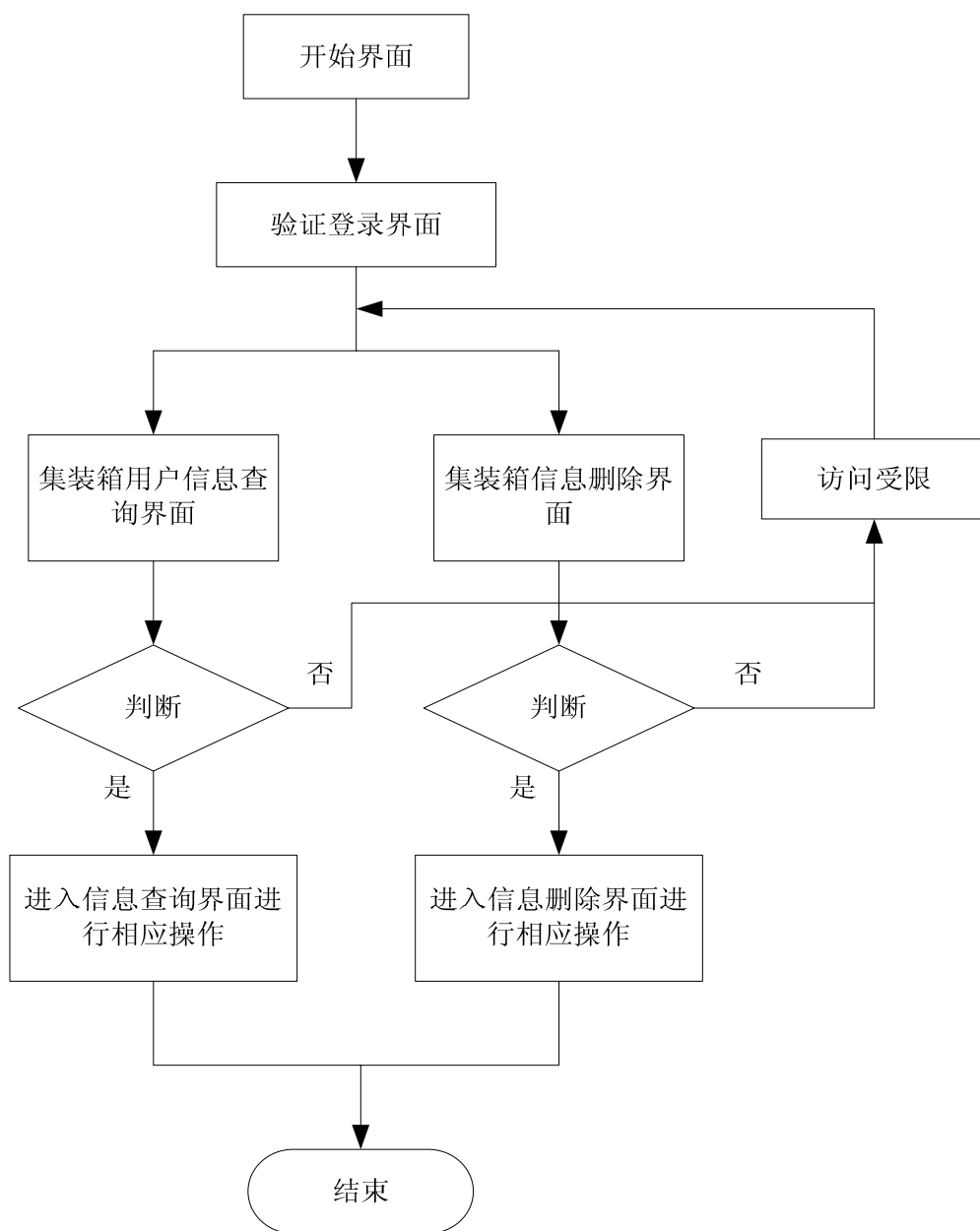


图 3.6 管理员身份验证流程图

Figure3.5The identification flowchar of the administrators

管理员信息登录模块下分为两个子功能模块，在管理员登录模块中首先要填入登录的用户名和密码，只有输入正确才能进入集装箱及用户信息模块和集装箱信息删除模块并对它们进行相应的用户信息查询和信息删除工作，模块的界面示意图如下：



图 3.7 管理员登录信息界面示意图

Fig.3.7 Administrator login interface diagram

由于嵌入式系统的内存资源非常有限，不能全部的记录所有的查询信息，这就需要对堆场集装箱这个特殊的环境在对集装箱的信息进行管理，有效的删减集装箱信息的操作可以减轻嵌入式系统的内存资源的占用，也更有效的便于集装箱信息的管理。集装箱信息删除模块，在正确登录密码后，进入集装箱信息删除模块，此模块的作用就在当集装箱被货主提走后，系统就需要将该集装箱的信息及时的删除，以便其他货主使用该集装箱。输入集装箱箱号后，点击“Cancel”按钮，就会对该集装箱对应的信息进行删除了。集装箱信息的删除界面如下图所示：



图 3.8 集装箱删除界面示意图

Fig.3.8 The interface of the container deleting diagram

下面是堆场集装箱查询模块，它分为堆场集装箱信息模块，堆场集装箱位置模块。如下图所示：



图 3.9 堆场集装箱查询界面示意图

Fig.3.9 Yard container query interface diagram

点击堆场集装箱位置信息就会出现两大模块集装箱的出场，入场信息界面，他们包括了集装箱编号，场区类型，集装箱尺寸大小，联系人电话，集装箱的入场时间和出场时间，还有一些备注的说明信息。其中集装箱的编号就是对应堆场集装箱中集装箱登记的编号，集装箱的尺寸大小包括 20 寸、40 寸、45 寸等，场区的类型就是堆场集装箱中集装箱箱型了种类堆放，包括空箱场区，重箱场区，冷藏箱场区，危险箱场区等等。它们的示意图如下所示：

集装箱入场信息

集装箱编号: HT43009245

场区类型: HT43009245

经手人: HT43009323

备注: HT43007392

尺寸大小: 20'

入场时间:

联系人电话:

确定入场

退出

图 3.10 集装箱入场信息界面示意图

Fig.3.10 Container admission information interface diagram

集装箱出场信息

集装箱编号:

场区类型: 空箱区

尺寸大小: 20'

联系人电话:

入场时间:

出场时间:

备注:

确定出场

退出

图 3.11 集装箱出场信息界面示意图

Fig.3.11 Container appearance information interface diagram

堆场集装箱信息模块就是集装箱的信息查询，它包含了集装箱编号、所放货品的名称、集装箱的箱型、集装箱尺寸大小、以及到达地和发货地。如下图所示：



图 3.12 集装箱信息查询界面示意图

Fig.3.12 Container information query interface diagram

3.3 嵌入式数据库的设计

3.3.1 数据库的框架设计

码头物流管理系统的数据库主要应用嵌入式数据库 Sqlite 进行存储集装箱堆场的信息，查询信息的数据库为 container.db、用户信息数据库 user.db 和查询所有数据库信息的数据库 Database_Info。container.db 数据库包括了集装箱信息、集装箱位置信息、集装箱用户信息、集装箱信息删除、集装箱出场查询、集装箱入场查询信息的内容数据表和信息的索引数据表。信息索引数据表包括 id 和 name 两个字段，id 表示查询信息中的标号，属性为 varchar(30)；name 表示信息名称，属性为 varchar(100)。信息索引的数据表名分别为 ctxx, ctwz, yhxx, xxsc, cccx 和 rccx；数据表中信息内容包括 id 和 info 两个字段，id 表示查询信息的标号，属性为 varchar(30)；info 表示信息的相关内容，属性为 text。信息内容的表名为在对应信息索引表名后

加上“info”。user.db 数据库是用来保存用户信息，其中包括数据表 user。user 包括 name、pass 和 way 三个字段，name 表示登陆用户名，pass 表示密码，way 表示权限，属性分别为 varchar（30）、varchar（30）和 varchar（1）。Database_Info.db 数据库是用来记录系统所有的数据库名和数据表名的，包括数据表 Database_Info。其中 Database_Info 包括了 tablename 和 dataname 两个字段，属性都为 varchar（30）。数据库的框架图如下图所示：

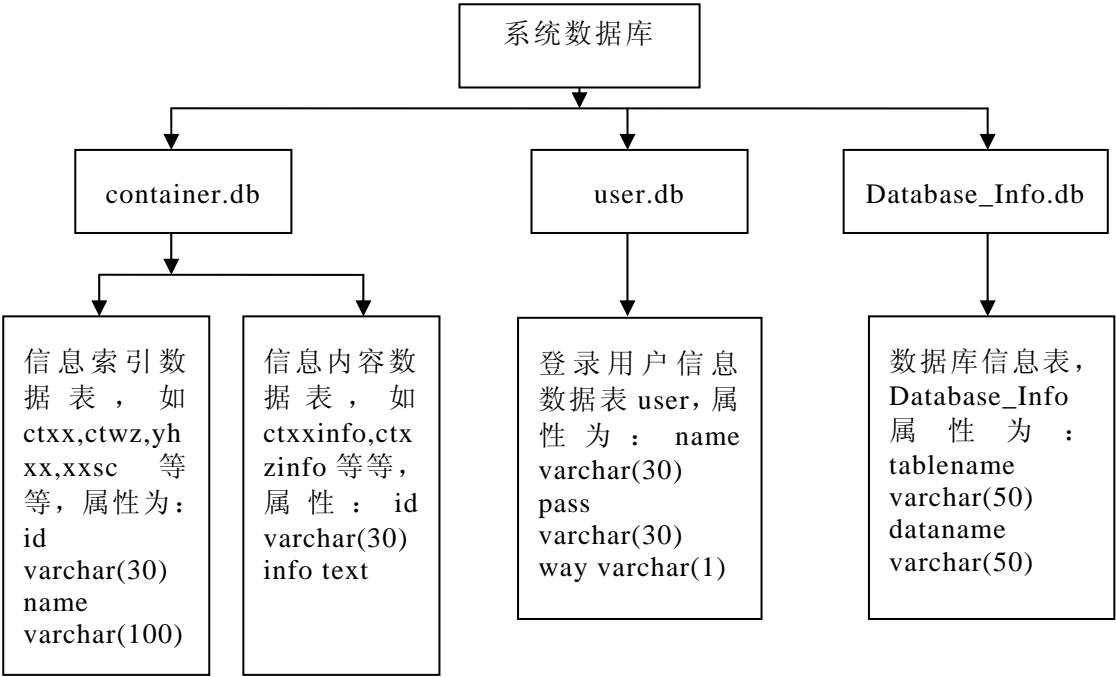


图 3.13 系统数据库的结构流程图
Fig.3.13 Flowchart of the system database structure

3.3.2 嵌入式数据库 SQLite 函数分析

针对嵌入式数据库开发就 SQLite 数据库的常用函数^[25]进行如下分析：

（1）打开数据库：

函数功能：打开一个数据库，文件名不一定要存在，如果此文件不存在，sqlite 会自动创建。第一个参数指文件名，第二个参数则是定义的 sqlite3 ** 结构体指针。如果打开或新建数据库成功，返回 SQLITE_OK 表示操作正常，否则返回响应的错误代码。

```
int sqlite3_open(
const char *filename,    /* 定义的数据库名字*/
sqlite3 **ppDb          /* 输出 SQLite 数据库句柄指针*/
);
```

（2）关闭数据库：

函数功能：如果用 `sqlite3_open` 开启了一个数据库，结尾时不要忘了用这个函数关闭数据库。关闭数据库，如果关闭成功，返回 `SQLITE_OK`，否则返回 `SQLITE_ERROR`。

```
int sqlite3_close(sqlite3*); /*参数就是刚才的结构体，也就是数据库句柄指针*/
```

（3）执行 SQL 语句：

函数功能：这个函数的功能是执行一条或者多条 SQL 语句，SQL 语句之间用“;”号隔开。建议在执行一条或者多条 SQL 语句得时候，指定第三个参数回调函数，在回调函数中可以获得执行 Sql 的详细过程，如果所有 Sql 执行完毕则应该返回 0，否则，则说明这次执行并没有完全成功。第五个参数：如果执行失败（没有返回 0）则可以查看第五个阐述得值。来查看详细错误信息。

```
int sqlite3_exec(
sqlite3*,                /* 已经打开的数据库句柄 */
const char *sql,         /* 要执行的 Sql 语句 */
sqlite_callback,         /* 回调函数 */
void *,                 /* 传递给回调函数的参数 */
char **errmsg            /* 保存错误信息 */
);
```

通常 `sqlite3_callback` 和它后面的 `void*` 这两个位置都可以填 `NULL`，表示不需要回调。比如做 `insert` 操作，做 `delete` 操作，就没有必要使用回调。而当作 `select` 时，就要使用回调，这是因为用 `sqlite3` 把数据调出来得通过回调告诉你查出了什么数据。

（4）exec 的回调

函数功能：你的回调函数必须定义为上面这个函数的类型。

```
typedef int (*sqlite3_callback)(void*, int, char**, char**);
```

（5）释放查询结果：

函数功能：释放当前查询的记录集所占用的内存

```
void sqlite3_free_table(char **result);
```

（6）非回调 select 查询：

函数功能：执行一次查询 Sql 并且返回得到一个记录集。第三个参数是查询结果，它是一维数组，内存布局为：第一行是字段名称，后面是紧接着是每个字段的值。

```
int sqlite3_get_table(  
    sqlite3*,                /*已经打开的数据库句柄*/  
    const char *sql,         /*要执行的 Sql 语句*/  
    char ***result,          /*保存返回记录集的指针*/  
    int *nrow,               /* 返回记录数（及查出多少行）*/  
    int *ncolumn,           /*返回字段数（多少列）*/  
    char **errmsg             /*返回错误信息*/  
)
```

3.4 本章小结

设计了码头物流管理界面程序的基本框架，详细分析了码头物流管理终端各个界面设计的功能，分析了各个功能模块。如堆场集装箱查询模块中的集装箱信息模块、集装箱位置模块、集装箱的入场和出场模块、以及管理员登录模块中的集装箱用户信息模块、集装箱信息删除模块等等。给出了各功能模块的流程图。并对各功能模块的界面进行设计。设计了嵌入式数据库的框架，如存储集装箱信息和查询信息的数据库 container.db 和用户信息数据库 user.db 以及查询所有数据库信息的数据库 Database_Info。并对 SQLite 的各功能接口函数进行分析研究。

4 集装箱堆场的空间分配及算法调度的研究

4.1 集装箱堆场作业模式

集装箱的出现给运输的储存带来了全新的观念，集装箱本身便是一个仓库，不需要再有传统意义上的库房。以集装箱存放的货物为例，在集装箱货场，可以直接以集装箱作为媒介，使用大型机械设备进行装卸、搬运，可以从一种运输搬运工具直接地转换到另一种运输搬运工具中。或者从发货方的仓库经由海上运输，不用开箱检验，也不用接触和移动集装箱箱内货物，可直接运输到收货人的仓库，从而省去了很多诸如，入库、出库、堆垛、验收、清点、保管等一系列仓库的管理作业。这样可以大大的提高管理的效率，而且避免了在仓储与装卸搬运的过程中对其商品的损伤。同时，在某种程度上还可以减少包装，运输的费用。因而，对改变传统储存作业有很重要的意义，是储存运输作业合理化的一种有效方式。

4.1.1 集装箱堆场

集装箱堆场^[26]，某些地方也叫场站。对于海运集装箱出口而言，堆场的作用就是在堆场中把所有出场、入场的客户集装箱在某处先集合起来（不论通关与否），等到了港口货船出港的时间时，再统一装船（此时必定已经通关）。堆场就是港口集装箱在通关装船前的统一集合的地方，在堆场的集装箱货物等待通关，便于一些船运公司，海关等进行管理。

集装箱堆场一般有两种含义：广义上的集装箱堆场可理解为进行装卸、交接和保管重箱、空箱的场地，包括前方堆场、后方堆场和码头前沿在内；狭义的集装箱堆场是指除码头前沿以外的堆场。其中也包括存放底盘车的场地在内。

集装箱前方堆场，是指集装箱作业主要在码头的前方，为了提高船舶装卸作业的效率，而暂时堆放集装箱的场地。它的主要作用是：当装有集装箱的货船到达港口时，有计划有次序地按积载要求将出口集装箱整齐地集中堆放，卸船时将进口集装箱暂时堆放在码头前方，以加速船舶装卸作业。

集装箱后方堆场，集装箱重箱或空箱进行交接、保管和堆存的场所。有些国家对集装箱堆场并不分前方堆场或后方堆场，统称为堆场。集装箱后方堆场是集装箱装卸区的组成部分。是集装箱运输“场到场”交接方式的整箱货办理交接的场所。

码头集装箱的作业流程与机械设备：

(1) 船——堆场

重箱（冷藏箱、特种箱）：船→集装箱岸边起重机→集装箱牵引半挂车→轮胎式集装箱龙门起重机→堆场；

空箱：船→集装箱岸边起重机→集装箱牵引半挂车→空箱堆高机→空箱堆场；

危险品箱：船→集装箱岸边起重机→集装箱牵引半挂车→集装箱叉车→危险品箱堆场。

(2) 堆场——拆装箱库

堆场→轮胎式集装箱龙门起重机→集装箱牵引半挂车→集装箱箱内叉车→拆装箱库。

(3) 堆场——货主

堆场→轮胎式集装箱龙门起重机→集装箱牵引半挂车→货主。

(4) 拆装箱库——货主

拆装箱库→集装箱箱内叉车→汽车→货主。

4.1.2 优化堆场的作业空间

集装箱堆场容量与集装箱堆场的集装箱可取性一直是码头运营企业在集装箱堆场管理上所必须取舍的问题。理论上，相对较低的堆存方式在置入或取出集装箱时，却具有较高的存取可及性^[27]。例如，早期美国的一些港口运营公司在土地资源充裕港口采用底盘车系统，直接将集装箱放到拖车上，需要移动时，再以牵引车将集装箱连接拖入或拖离堆场。虽然这种堆场作业方式可获得比较高的集装箱存取可及性，但是却必须将集装箱和底盘一同储存，无法使得集装箱堆叠，大大降低了堆场土地所能提供的集装箱储存容量。反之，若利用龙门起重机搭配使得集装箱的排列以紧密化，堆叠高层化的方式，将集装箱直接放置在堆场库区的地面上，并以垂直堆叠的方式堆储，可大幅提升堆场的储存容量和堆场集装箱土地的利用效率，适

合土地资源有限的码头堆场采用。例如，新加坡新建的世界上最先进的集装箱港口——巴西班让集装箱码头，就将集装箱垂直堆叠八至九层。但相对的，集装箱存取可及性的效率则大幅度的降低，必须进行妥善的事前规划，才能够尽量降低调度效率低的集装箱搬移作业。

在集装箱码头的堆场中，为了更好的利用集装箱堆场的使用面积，增大码头集装箱堆场的容量，提高码头集装箱运输的效率，增加码头、港口的企业竞争力，可以采用以下的方式来合理有效地对堆场的作业空间优化分配，从而提高码头集装箱的运作效率。这将极大的解决船舶大型化带来的对船舶在集装箱装卸过程中的效率问题和码头集装箱堆场空间的需求问题^[27]。

（1）集装箱堆垛高度增加。在北美的部分港口中由于采用底盘车装卸系统，集装箱仅能堆垛一层，堆场利用率非常的低。在采用了龙门起重机装卸系统的码头中，一般集装箱堆垛3-5层，有些港口可以堆到8-9层，而在空间严重缺乏的港口，甚至可以堆垛到12层。在部分港口中已经考虑了用龙门起重机替代底盘车或跨运车来提高集装箱堆垛高度。显而易见，增加集装箱的堆垛高度会提高集装箱堆场空间作业的利用率。

（2）集装箱码头全天候运作。集装箱船、列车和集卡全天候的运作，能够增加码头的使用效率，从而减少了集装箱在码头堆存的时间，提高集装箱堆场的利用效率。

（3）减少集装箱在码头的堆存时间。由于各种不确定的因素造成的集装箱货物到达港口的时间不能确定，而造成货主提货的时间被延迟。一般集装箱码头会让到达港口的集装箱货物免费在集装箱堆场堆存一段时间，在国内一般为7天，国外某些港口为3-4天。然而也有一些货主认为码头集装箱堆场堆存货物费用相对便宜和便利，也会延长货物在码头的堆存时间。集装箱在码头堆存时间越短，对码头堆场面积需求就越小，使得码头堆场的利用率就越高。

（4）码头智能化运输系统的运用。对集装箱码头的智能化运用，可以使得集装箱码头各个装卸运输环节的配合更加顺畅，从而提高了港口的运作效率，减少集装箱在码头港口的停留时间，提高集装箱堆场利用效率。码头的自动化是港口发展

的必然趋势，码头自动化的实现将使集装箱码头提高大型机械设备资源的利用率，减少劳动力费用，提高码头运作效率。

优化堆场的作业空间主要考虑的情况是在集装箱前方堆场如何最大化利用空间资源问题，结合嵌入式码头物流管理系统终端的设计主要解决在集装箱存放的空间请求，如何使得占用空间最小，最大可能的提高空间的利用率。从而解决集装箱堆场空间货位分配优化的问题。

4.2 集装箱堆场空间分配策略的研究

4.2.1 堆场箱位分配策略研究

堆场的集装箱位置的储存策略是储存区规划的一个重要原则，对于完成出入库的任务尤为的重要，必须要配合集装箱位置的分配原则才能决定储存作业实际运作的模式。堆场箱位分配策略考虑的原则有很多^[28]，主要有以下几种：

（1）巷道均匀存放。要防止因某个巷道中机械设备作业堵塞影响到其他堆场货箱的出入场区而造成的生产作业中断。

（2）上轻下重。要考虑到承载额定箱位的重量，要确保同尺寸的集装箱的堆放。

（3）箱位的种类的相关性。货箱的种类一般分为空箱、重箱、冷藏箱、危险品箱。按照分类的原则应尽可能的分开存放。空箱放在空箱堆场，重箱放在堆场，危险品箱放在危险品堆场，冷藏箱放在能确保控制冷藏箱中箱内货品温度的场区。如果调度效率高的集装箱可就近放置在出库口附近。同种类的箱位应尽可能的放在相邻的位置。

（4）先到先服务。一般这一原则是针对集卡进行调度的。为了保证快速的调度响应出场请求，一般将出场调度效率高的货箱放在出场口附近，方便机械设备快速的响应出场信息，从而提高堆场的调度响应效率。

4.2.2 堆场箱位任务优先级分配策略

集装箱码头堆场在港口码头的作用是起到扮演缓冲区的角色，是提供装卸船舶堆放集装箱的场所。同时也是向货主交接集装箱和对集装箱进行临时保管的地方，用以将集装箱暂时集中堆存，为以后各项连续作业提供便利条件。

集装箱码头堆场的调度都会对信息管理的出入堆场的集装箱进行排序，对于出入堆场集装箱任务，其基本的流程图如下：

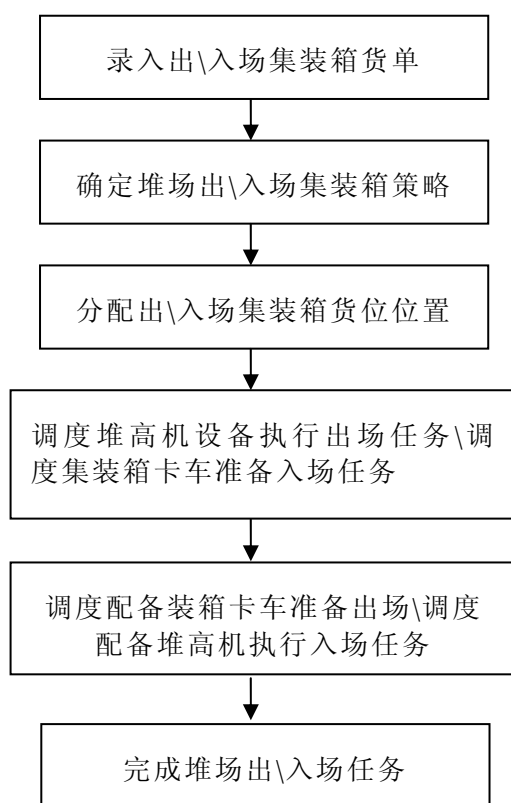


图 4.1 集装箱堆场出\入场流程图

Fig.4.1 Flowchart of container yard admission and appearance

集装箱的提箱作业是一个复杂的作业过程，它涉及了很多的制约因素。在这里为了简化问题，突出问题的核心部分，对集装箱箱位任务优化时设定以下的前提条件^[32]：

- (1) 在堆场中的一条街中给定几个 bay 位的范围进行堆场集装箱的作业。
- (2) 制定提箱作业计划时应事先指定一种类型的集装箱堆场设备进行作业。

在集装箱码头堆场执行出入场区的调度时，为了避免出入堆场集装箱任务的冲突，造成集装箱码头堆场缓冲区交通阻塞，秩序混乱。对出入堆场集装箱任务遵循以下策略调度，这样可以提高运输和堆场作业的效率。

(1) 堆场出场任务优先。当堆场的集装箱出场任务单和入场任务单同时递交的时候，应优先响应堆场集装箱的出场任务，即所有的出场任务的优先响应都应该高于入场任务的优先响应。当堆场集装箱入场任务不影响出场集装箱任务时，响应堆场集装箱的入场任务。

(2) 先到先服务。对堆场集装箱出场任务和入场任务按照它们到达的时间先后分别进行排序，先到堆场集装箱的任务要高于后到任务的优先响应。

(3) 人工干预。由于码头港口地域广阔，集装箱堆场内环境过于复杂，在实际的堆场操作过程中，难免会出现异常情况的发生，而且还存在着其他很多因素的影响。如由于天气原因造成的船舶不能按时到港，使得集装箱积压港口，或者客户的货物没有及时的从堆场中提走，或者大型机械设备出现故障，或者到港船舶的集装箱货物繁多造成拥挤等等。这都是有可能导致堆场的空间分配出现问题。这就需要丰富经验的管理工人对堆场场区的信息进行人工干预、调度。在人工进行干预的任务执行的优先响应要高于其它优先响应。

4.3 嵌入式实时数据库查询优化算法研究

嵌入式数据库系统主要针对移动计算或基于某种特定计算模式的数据库管理系统。嵌入式数据库系统通常与操作系统和具体的应用结合在一起，运行在嵌入式或移动设备上。嵌入式实时数据库技术涉及很多个学科，如数据库、实时系统、分布式计算以及移动通信等等。嵌入式数据库主要的管理范围是存放在 **SRAM**、**ROM** 或 **Flash ROM** 中的系统和用户的数据信息。因为嵌入式系统和用户的数据信息一般都存储在 **SRAM**、**ROM** 或 **Flash ROM** 中，但时由于嵌入式系统的内存较小而且 CPU 运行的速度较慢，因此，在嵌入式数据库系统中研究数据的结构和算法以及数据查询处理算法就显得非常的重要了，必需采用特殊的数据结构、算法及相关的数据库精简技术。才能更好在嵌入式系统中发挥作用。

4.3.1 手持终端与嵌入式数据库的交互分析

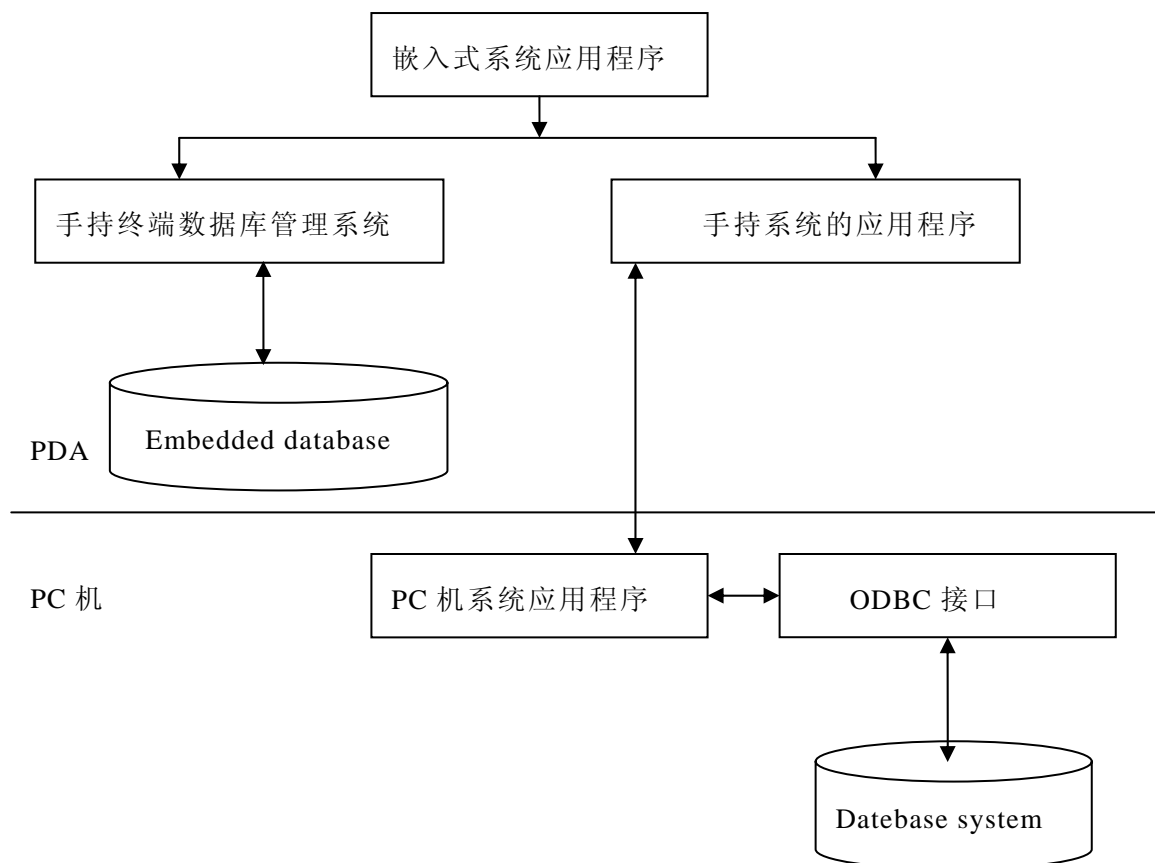


图 4.2 嵌入式手持终端系统的框架图

Fig 4.2 The framework of embedded handheld terminal system

嵌入式系统的应用程序是在 PDA 手持终端的嵌入式数据库管理系统的基础上设计的，针对 PDA 客户对需求开发的要求而设计的应用程序，由用户界面和其相关的控制模块而构成的。EDB (embedded database system) 是 PDA 上的嵌入式数据库，一般存放在 PDA 的 Flash 中，运行 PDA 时加载内存。主要的数据源是 PC 机上 ODBC 数据源管理器，及其管理器中管理支持 ODBC 接口各种数据类型的数据源^[29]。

在手持终端上建立一个嵌入式数据库管理系统，主要是实现基本的 PDA 数据管理任务。其针对数据库的操作主要是包括表的创建和删除、记录的添加、删除、修改和查询等。嵌入式数据库系统的特点就是要精简、易用、体积小、占用资源少等，

但是同时导致了在数据容量的增加和功能方面的缺陷。为了克服这一缺陷，就需要与后台数据库相进行紧密的连接。

而针对嵌入式手持终端的嵌入式数据库是基于内存的数据库，而查询的处理最终执行的结果是与内存数据库接口相连接的^[30]。为了充分的利用内存数据库的特点，要求最大限制的节约内存。从而提出了节省内存的查询处理的实现算法，传统的查询优化策略大部分是以时间为代价的，其中有很多的因素都对时间有影响：如 I/O 次数。但嵌入式实时数据库是基于内存的数据库，它无需 I/O 调用，主要是尽量减少查询所占用的内存空间。所以传统的查询优化已经不适合实时内存数据库的要求。

4.3.2 嵌入式系统应用查询算法的问题分析

嵌入式系统中查询算法时，必须要注意以下几个方面^[31]：

(1) 尽量避免使用动态分配内存。在嵌入式系统的应用过程中它是不支持动态内存分配的，但没有动态内存分配，将很难在链表数据中添加或删除数据节点。如果链表的长度基本保持不变的话，可使用数组进行代替，使数组的大小设为链表长度的最大值。一旦超出了这个最大值的范围，就返回操作失败。

(2) 如果要引入智能搜索就一定要考虑嵌入式硬件方面的资源限制，尽量做到算法的简单、实用。

(3) 避免递归。任何递归函数都是用迭代函数进行实现的

4.3.3 算法在嵌入式码头管理系统中的应用研究

(1) 贪婪算法

贪婪算法^[32]是一种典型的领域搜索技术，是自底向上的启发式查询优化算法。算法是从一个随机选取的可能解出发，对这个可能解的周边领域进行搜索，如果找到了更好的解，就将这个更好解作为当前解继续对周边的领域进行搜索，直到所有的领域操作都不能对这个解进行改变。算法在选择搜索顺序时，使用一种简单而严格方法，每次都选取代价最小的连接。贪婪算法总是能做出当前最好的选择，具有

很好的局部搜索最优解得能力，但是，它并不能对所有的问题在全局搜索中得到最优解。

在嵌入式实时数据库中，对于简单的实时事务搜索，使用贪婪算法的优化可以起到有效地节省系统开销。但是对与长事务就应该尽量优化，并提高搜索效率。所以，在算法的时间复杂度允许的情况下，为了能搜索到更优的解，甚至全局的最优解，就必须要考虑在贪婪搜索的基础上对其拓宽搜索区域。

（2）禁忌搜索

禁忌搜索^[33]的思想最早由 Glover 提出的，它是对局部领域搜索的一种扩展，适合于处理最优化或排列组合问题，是一种全局逐步寻优的算法。其主要发展的概念是来自登山算法，是以搜索邻近可能获取最优解的区域为基础，进行最优解的区域搜索，不断寻求最优解的过程。禁忌算法通过灵活的存储结构和对应的禁忌准则来避免陷入局部最优解搜索，而保证多样化的有效探索，最终实现全局优化。

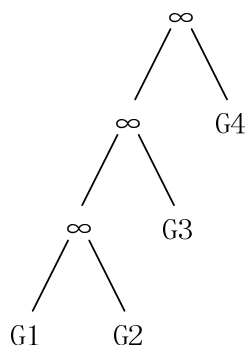
局部领域的搜索是基于贪婪思想，并且持续地在当前解的领域中进行搜索。虽然贪婪算法容易解理，通用并且易实现，但其搜索性能是完全依赖于局部领域结构和开始的起始解的，尤其是如果陷入局部极小而无法保证全局优化性。针对局部领域搜索，为了最终实现全局化最优解的需要

禁忌搜索是人工智能的体现，是对局部领域搜索的扩展。禁忌搜索算法最重要的思想是标记对应已经搜索的局部最优解的对象，并在下一步的迭代搜索中有意识的避开这些对象（但不是完全的隔绝），这就保证了对不同途径进行有效搜索的探索。

（3）提出一种适用于手持终端在码头管理系统中的算法^{[33] [35] [31]}

提出的这一算法是综合了禁忌搜索中扩展的登山算法与贪禁算法中结合的改进算法（TGI），这个算法重要可以分为两个阶段：（1）进行贪婪算法，运行输出是一个局部的极小解，这是为了成为下一步禁忌搜索阶段的一个初始状态；（2）进行禁忌搜索的迭代改进。先利用禁忌搜索迭代改进算法的局部优化改进这个初始的极小解。然后，在重新的随机生成一个新的初始状态，重复改进进行不断搜索最优解得过程，直到满足优化解得产生。

在求解的空间中，需要从已存在状态中随机移动一个新的状态，即需要得到一个新的连接顺序。而这一个新的状态需要移动到达的邻居状态区域是由一系列移动规则决定的。随着移动规则的区别，算法也将应用于不同解空间子集。例如，下图所示的左深连接树，用 $G1G2G3G4$ 表示。对该解进行一次随机性的移动（交换关系 $G2$ 和 $G4$ ），就会得到 $G1G4G3G2$ 的表示，这样所形成的叶结点排列的连接树就表征了一个新的连接顺序。



$G1-4$ 表示一个基本关系 ∞ 表示一个连接操作

图 4.3 查询树模型

Fig.4.3 Query tree model

新算法的研究目标是在完整解空间搜索中得到最优解或较优解，因此，有上图的描述中可知一个状态的邻居状态可能是任意状态形式的连接树。移动的规则是要应用在连接树的内部结点上，如果改变它们的位置，则不影响其他状态结点。算法中改进函数采用如下的移动规则描述，其中 A ， B ， C 分别为独立的连接表达式形式移动规则如下：

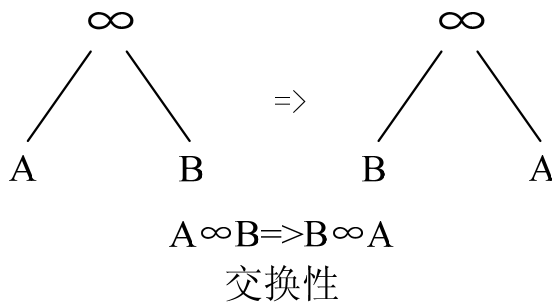
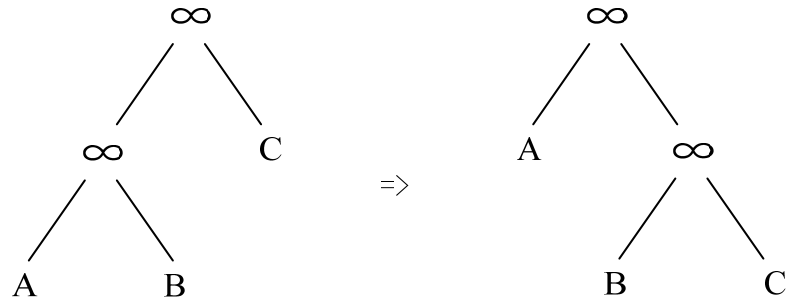


图 4.4 bushy-tree 解空间的移动规则（1）

Fig.4.4 The mobile rules of bushy-tree solution space (1)

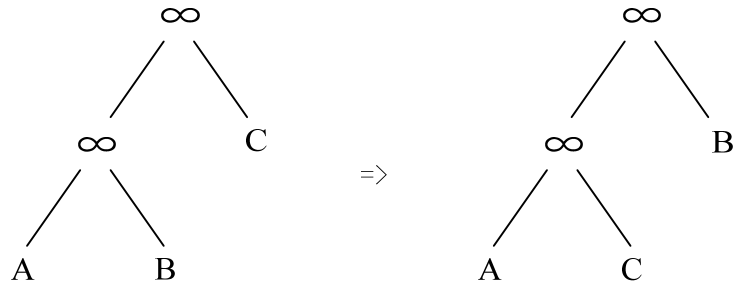


$$(A \infty B) \infty C \Rightarrow A \infty (B \infty C)$$

结合性

图 4.5 bushy-tree 解空间的移动规则 (2)

Fig.4.5 The mobile rules of bushy-tree solution space (2)

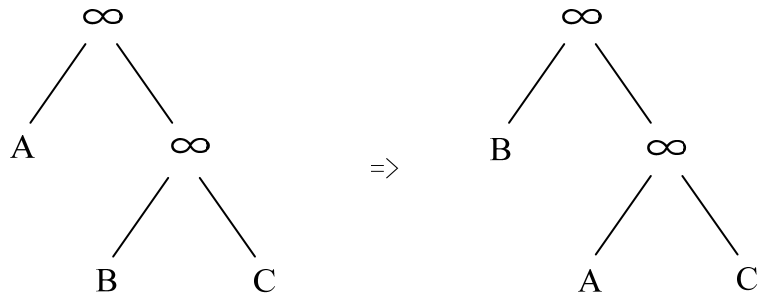


$$(A \infty B) \infty C \Rightarrow (A \infty C) \infty B$$

左连接交换性

图 4.6 bushy-tree 解空间的移动规则 (3)

Fig.4.6 The mobile rules of bushy-tree solution space (3)



$$A \infty (B \infty C) \Rightarrow B \infty (A \infty C)$$

右连接交换性

图 4.7 bushy-tree 解空间的移动规则 (4)

Fig.4.7 The mobile rules of bushy-tree solution space (4)

(4) 算法步骤分析

输入查询 P 的连接图 T，关系数 n，终止条件 stopping_condition

输出查询 Q 的最优连接树 minstate

算法描述如下：

minX = X^∞ ;

mincost = ∞ ;

state = bulidGreedyPlan(T);

/*根据上述的连接图得到贪婪算法并

生成搜索计划*/

if (stopping_condition) then

 minX=state

 return minX

end

while not(stopping_condition) do {

 newstate = improve(state , stopping_condition)

 newcost = Cost(newstate)

/*优化初始状态，并判断结

束条件 */

 if (newcost < mincost) then

 minX =newstate

 mincost =newcost

 end

 state = "Random starting "

 cost = Cost(state)

 }

return minX

4.4 本章小节

介绍了集装箱堆场的工作环境和工作流程，并对实际的工作环境提出了优化堆场作业空间的方案。对堆场箱位和堆场任务优先级的分配策略进行了研究。重点研究了应用在嵌入式实时数据库查询的优化算法。研究了嵌入式手持终端与嵌入式数

数据库的交互分析。由于手持终端设备的内存及 I/O 的特定环境限制了它在运行处理数据库查询信息时的能力。所以提出了一种算法应用在手持终端上的查询调度的算法中，并设计算法使其应用在嵌入式码头管理的手持终端系统中。

5 算法实验与分析

5.1 算法描述分析

本文所讨论的嵌入式查询算法结合了贪婪算法与禁忌搜索算法。其中贪婪算法是一种自底向上的启发式查询优化算法。在基于查询图的贪婪算法中，算法生成的时间复杂度为 $O(n^3)$ ，生成连接树的代价较小。因此可知，贪婪算法具有快速求解和执行效率高的特点。但由于，贪婪算法采取的搜索方式是不可撤回的，这就不能保证从中获得的解为最优解。为了使算法能够搜索到最优解，就应该考虑在贪婪算法的基础上拓宽搜索区域。而禁忌搜索算法的主要发展概念来自登山算法，是以搜索邻近可能获取最优解得区域为基础，进行最优解的区域搜索，不断寻求最优解的过程。针对嵌入式实时查询数据库的特点，本文将二者算法加以结合并进一步优化，并通过大量的实验进行在嵌入式实时环境下多连接查询优化问题的有效性。

提出的这一算法是综合了禁忌搜索中扩展的登山算法与贪婪算法中相结合的改进算法（TGI），这个算法主要可以分为两个阶段：（1）进行贪婪算法，运行输出是一个局部的极小解，这是为了成为下一步禁忌搜索阶段的一个初始状态；（2）进行禁忌搜索的迭代改进。先利用禁忌搜索迭代改进算法的局部优化改进这个初始的极小解。然后，在重新的随机生成一个新的初始状态，重复改进进行不断搜索最优解的过程，直到满足优化解的产生。

本算法所讨论的搜索空间为整个策略空间，新算法的研究目标是在完整解空间搜索中得到最优解或较优解。在求解的空间中，需要从已存在状态中随机移动一个新的状态，即需要得到一个新的连接顺序。而这一个新的状态需要移动到达的邻居状态区域是由一系列移动规则决定的。算法也将应用于不同的解空间子集中。由上一章给出的解空间移动规则可知一个状态的邻居状态可能是任意状态形式的连接树。移动的规则是要应用在连接树的内部结点上，如果改变它们的位置，则不影响其他状态结点。具体的算法步骤在上一章已经明确给出，在这里就不过多叙述了。

5.2 实验环境

嵌入式码头物流管理系统是一个针对在码头作业环境中集装箱堆场的管理现状而进行的提高港口集装箱调度的系统，将嵌入式数据库作为其中的底层数据，而进行采集与管理，负责实时对港口集装箱调度的数据进行处理，并接收服务器的实时查询处理等命令，在整个系统中起着重要作用。为验证算法的有效性，在嵌入式实时数据库系统针对港口集装箱的箱号调度与查询的管理将对算法进行实验测试，通过设置若干周期性实时事务，对任务的执行结果进行比较。

本文实验环境采用的是基于 ARM 9 芯片的开发板，嵌入式 linux 操作系统，arm-linux-gcc 的版本。

实验中选取的连接图为随机选取，它们都围绕 2 个参数变化：关系数为 n ，其中允许连接每个结点的最大边数为 $node_fanout$ 。每个结点至少都要与一边相连，整个连接图为连通状态。

5.3 实验结果与性能分析

下面给出了 TGI 算法与禁忌搜索、贪婪算法的实验比较结果。其中所用的值都是取自 20 次实验结果的平均值。

5.3.1 运行时间分析

下表为 TGI 算法与禁忌搜索算法运行时间的比较。当 $n>6$ 的时候，在本文模拟的实验环境下，禁忌搜索算法因占用系统的大量内存资源而无法求解。因而，如下只比较关系数 $4 \leq n \leq 6$ 的状态。表 5.1 中 n 为关系数，plans 为连接计划的数目。

表 5.1 TGI 算法与禁忌搜索算法运行时间的比较

Tab. 5.1Comparison of running time between TGI algorithm and tabu search algorithm							
n	plans	1		2		3	
		TGI	tabu	TGI	tabu	TGI	tabu
4	15	0.001	0.02	0.003	0.03	0.004	0.05
5	105	0.025	0.84	0.033	0.96	0.033	0.90
6	945	0.071	4.4	0.091	5.41	0.098	5.98

从实验的结果来看，当比较关系系数在 4 到 6 之间时，TGI 算法的运行时间远小于禁忌搜索算法的运行时间。在这种情况下可找到最优解，避免了占用系统大量的内存资源，又不用列举出所有可能的查询计划，算法的空间复杂性较低。

5.3.2 生成计划的质量分析

在下图的表示中 n 为关系系数， fanout 为每个结点允许连接的最大边数。当 n 由 10 到 60 时，TGI 算法与贪婪算法比较而生成的计划质量改进情况。假设 t 为 TGI 算法生成的计划代价， t_1 为贪婪算法生成的计划代价。则曲线个个点表示为 $(t_1 - t) * 100 / t_1$ 的值。如图所示，当 $\text{fanout}=3$ 时，TGI 算法的解平均比贪婪算法的解改进了 30%；当 $\text{fanout}=4$ 时， n 小于 35 时，TGI 的解能比贪婪算法改进 20%，随着 n 的数量的增加，其改进比降低。这在实际的嵌入式码头物流管理系统中出现的大规模数量的查询情况比较少。所以，TGI 算法有一定的优势。

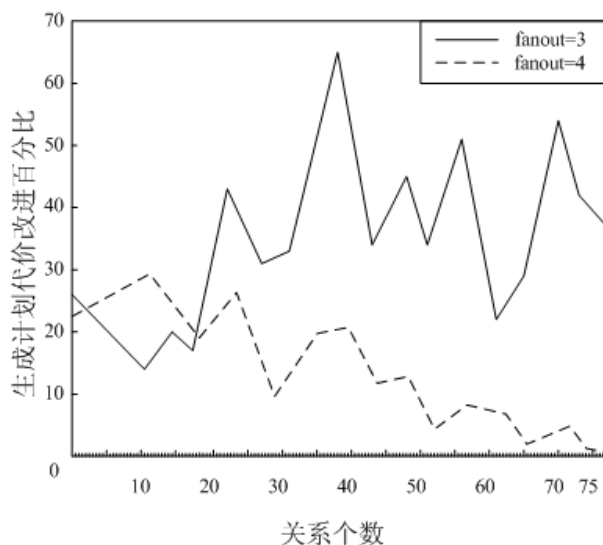


图 5.1 TGI 算法与贪婪算法计划质量改进比

Fig.5.1 Quality improvement ratio of TGI algorithm and greedy algorithm scheme

5.3.3 算法模块的查询质量分析

针对本文模拟实验采用的开发平台，应用在本文设计的嵌入式关系数据库管理系统（EDBMS，Embedded Database Manage System）的基础上，对其中的新算法模块进行分析。新算法的模块在执行时需要占用系统的内存资源，为了验证该模块执

行的效率，实验将没有实现 TGI 模块的 EDBMS，及采用该体系结构的 EDBMS 的效率进行了比较,本模块中定义了不同数目的主动规则，对于每个 EDBMS 都递交相同数目的 SQL 查询，其执行时间如下图所示。由图中可以看出，虽然 TGI 算法模块在执行中会占用一些资源，但由于能够根据系统状况来动态配置查询优化的方式，因此实现了 TGI 模块的 EDBMS 执行效率是优于没有实现 TGI 算法模块的 EDBMS。

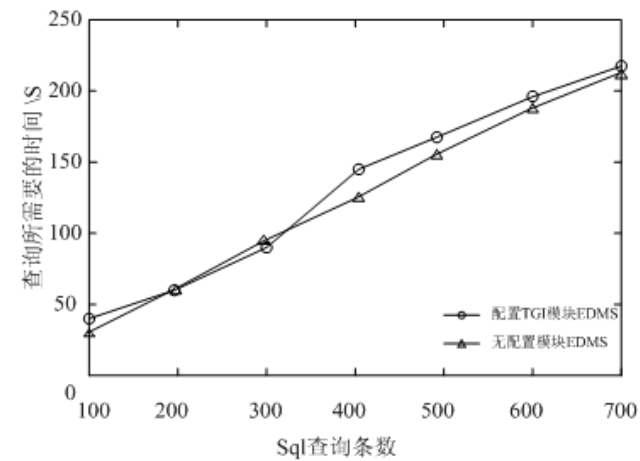


图 5.2 配置基于系统的模块规则对查询效率的影响

Fig.5.2 Influence of rules based on system configuration module on efficiency of the query
通过算法的描述将这一算法应用在码头物流管理手持终端的控制界面中，如下

图所示：

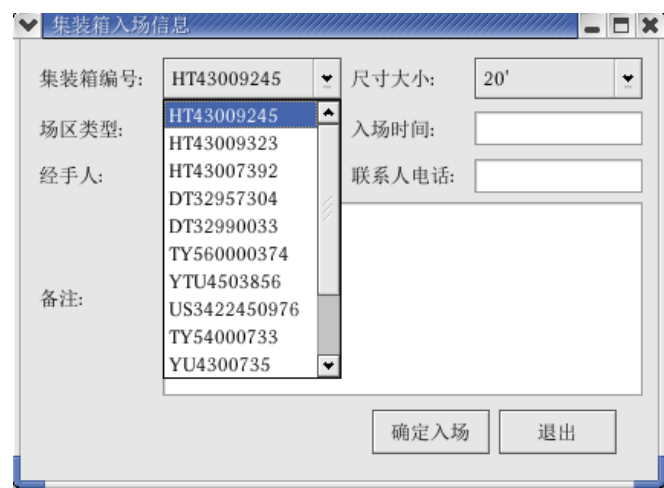


图 5.3 算法在码头物流管理界面上的应用

Fig.5.3The application of Algorithms in terminal logistics management interface

5.4 本章小结

本章主要是通过举例和实验进一步证明了结合禁忌搜索算法和贪婪算法提出改进连接顺序优化算法的可行性。举例用具体的数值来说明嵌入式查询算法 TGI 与贪婪算法和禁忌搜索算法的不同，直观的说明了 TGI 算法的优点。与禁忌搜索算法和贪婪算法的模拟实验表明，本算法在系统的运行时间、计划的质量改进和查询的效率都有一定的优势。通过算法的实验与分析表明，在嵌入式实时数据库查询系统中 TGI 算法更适于应用的环境，也更适于系统的性能。

结论与展望

本论文是嵌入式系统在码头物流管理上的应用，将嵌入式手持终端应用在码头集装箱管理这一领域，使嵌入式手持终端技术能更好的应用在码头集装箱堆场的管理中。提高作业工人的运作效率，提高堆场空间管理的调度效率。并提出应用在嵌入式手持终端上的查询算法。

本论文主要做了以下的工作：

建立了嵌入式 linux 的开发环境，基于 Qt/Embedded 的图形用户界面的开发与移植、BootLoader 的分析、内核的配置与编译系统的分析与研究。阐述了交叉编译环境的搭建、如嵌入式系统开发中广泛使用的 NFS 服务器的建立等。分析了嵌入式数据库 SQLite3 的安装及交叉编译的方法。针对码头物流管理系统的终端设计进行了重点研究，并且研究了嵌入式实时数据库的查询算法，并将其应用在码头管理的手持终端中。

通过做这个课题的研究分析及实践开发，获益良多，加深了对嵌入式系统的认识和理解，对驱动程序及图形用户界面开发有了深入了解，熟悉了嵌入式系统的整个开发流程。

经验证，对嵌入式手持终端各个功能模块进行了性能测试，达到了预期的设计要求。本文对嵌入式系统手持终端的设计做了一定的工作，取得了一些成果，但是还存在着很多的不足和需要改进的地方：

- （1）手持终端的设计需要进一步的完善和优化。
- （2）在开发中应该加载无线通信模块，使嵌入式手持终端系统可以更加完善。
- （3）算法的复杂度及设计思想有待进一步的优化。算法的设计比较简单，应进一步的改进算法实现更复杂实时响应度更好的查询功能。

参 考 文 献

- [1] <http://www.roboticfan.com/blog/index.html>
- [2] 杨中华. 基于 Qt/Embedded 的 SQLite 数据库研究及应用[D]. 四川: 西华大学, 2008: 12-13
- [3] 2004 中国计算机科学技术年度发展报告 - 嵌入式系统部分, 2005, 5
<http://www.bol-system.com>
- [4] 林艺明. 集装箱码头调度问题研究[D]. 上海: 上海海事大学, 2007, 03: 12-13
- [5] 郑灵翔. 嵌入式系统设计与应用开发[M]. 北京: 北京航空航天大学出版社, 2006, 1: 2-4
- [6] 孙琼. 嵌入式 Linux 应用程序开发详解[M]. 北京: 人民邮电出版社, 2008, 8: 107-110
- [7] 徐广毅, 张晓林, 崔迎炜, 杨欣昕, 吴小伟. 嵌入式 Linux 系统中 GUI 系统的研究与移植[J], 单片机与嵌入式系统应用, 2004(10): 11-17
- [8] MiniGUI 用户手册, 北京飞漫软件技术有限公司。
<http://www.minigui.com/index.php?id=minigui-doc-dev-package&L=1>, 2007, 8
- [9] OpenGUI. document. <http://www.tutok.sk/index2.php>
- [10] Qt/Embedded. <http://www.trolltech.com/products/qt/embedded>, 2007, 5
- [11] 滕英岩. 嵌入式系统开发基础—基于 ARM 微处理器和 Linux 操作系统[M]. 北京: 电子工业出版社, 2008, 10: 69-71, 91, 164, 212
- [12] <http://www.qteverywhere.com/>
- [13] MINI2440 用户手册, 广州友善之臂计算机科技有限公司, 2008, 12
- [14] 纪君峰. 基于 QT/E 的嵌入式系统及应用 [D]. 北京: 北京邮电大学 2008, 5: 18-20
- [15] Jasmin Blanchette, Mark Summerfield. C++ GUI Programming with Qt 4, 2nd Edition. Prentice Hall, 2008: 15-16
- [16] 陈英革, 马力, 王小英. Qt/E 中信号和槽机制的分析及教学实践[J]. 常熟理工学院学报 (自然科学), 2008(10): 108-110
- [17] 张广斌, 宫金林, 陈爽. SQLite 嵌入式数据库系统的研究与实现[J], 西南交通大学, 2008: 11-13
- [18] 彭侃. 基于 ARM9 的嵌入式软件平台的研究与实现[D]. 江西: 东华大学, 2008: 22-25
- [19] SQLite Documentation, www.sqlite.org/docs.html
- [20] 张涛. 集装箱堆场物流系统规划研究[D]. 大连: 大连理工大学. 2007(6): 11-13
- [21] 尧有平, 薛小波, 基于 ARM-Linux 的 SQLite 嵌入式数据库的研究[J]. 微计算机信息, 2008 (24): 64-66

- [22]刘小春, 张有为, 向伟. 嵌入式 Linux 下 Qt/Embedded 应用关键技术研究[J]. 微计算机信息, 2007 (23) :62-63
- [23]胡英多. 基于 S3C2440 的嵌入式 Linux 的应用[D], 电子科技大学. 2009: 22-25
- [24]黄东. 基于 SQLite 的移动嵌入式数据库同步系统的研究和开发[D]. 武汉: 华中师范大学, 2009: 12-15
- [25]孙磊, 基于 ARM Linux 的嵌入式数据库 SQLite 的移植开发[D]. 昆明: 昆明理工大学, 2008: 38-45
- [26]朱新民. 物流仓储[M]. 北京: 清华大学出版社, 2007(11): 45-60
- [27]王维圳. 集装箱堆场分配问题的启发式方法研究[D]. 天津: 天津大学, 2008(5): 27—30
- [28]王孟昌. 集装箱码头堆场箱位动态分配优化策略研究[D]. 武汉: 武汉理工大学, 2007(5): 27-29
- [29]张涛. 嵌入式实时数据库关键技术研究实现[D]. 北京: 电子科技大学, 2005(5):13-15
- [30]蔡芸, 霍永忠. 集装箱港口物流系统仿真和优化研究综述[J]. 系统仿真学报, 2009(4)
- [31]张益生. 嵌入式的智能路径遍历算法[J]. 广东公安科技, 2005(2): 45-47
- [32]严太山. 用基于贪婪算法的混合遗传算法求解 0/1 背包问题[J]. 现代计算机, 2007(8): 14-16
- [33]戴冬, 王江晴. 一种基于禁忌搜索算法的车辆路径问题的改进算法[J]. 中南民族大学学报(自然科学版), 2007(5): 4-5
- [34]刘云生, 迟岩. 基于遗传算法的实时内存数据库查询优化[J]. 小型微型计算机系统, 2005(3): 465-468
- [35] 宋静静, 贾智平. 一种嵌入式实时数据库系统查询优化算法[J]. 计算机工程, 2007(6): 90-92

作者简历

一、基本情况

姓名：连照亮 性别：男 民族：汉 出生年月：1982-02-27 籍贯：辽宁省沈阳市

2001-09—2005-07 东北大学成人教育学院学士；

2007-09—2010-07 辽宁工程技术大学电子与信息工程学院硕士

二、在学期间发表的学术论文

1.连照亮，徐世国. 基于 Qt/Embedded 在嵌入式 Linux 下的应用研究[J]. 微计算机信息，2010，06.

学位论文数据集

关键词*	密级*	中图分类号*	UDC	论文资助
嵌入式 GUI ; Qt/Embedded ; SQLite3 ; 集装箱堆场	公开	TP315	004	
学位授予单位名称*	学位授予单位代码*	学位类别*	学位级别*	
辽宁工程技术大学	10147	工学	硕士	
论文题名*	并列题名*			论文语种*
基于 ARM Linux 的码头集装箱堆场的应用研究				中文
作者姓名*	连照亮	学号*	47072246	
培养单位名称*	培养单位代码*	培养单位地址	邮编	
辽宁工程技术大学	10147	辽宁省阜新市	123000	
学科专业*	研究方向*	学制*	学位授予年*	
计算机应用技术	嵌入式系统应用	2.5 学年制	2010 年	
论文提交日期*	2010 年 5 月			
导师姓名*	徐世国	职称*	教授	
评阅人	答辩委员会主席*	答辩委员会成员		
电子版论文提交格式 文本 (<input checked="" type="checkbox"/>) 图像 (<input type="checkbox"/>) 视频 (<input type="checkbox"/>) 音频 (<input type="checkbox"/>) 多媒体 (<input type="checkbox"/>) 其他 (<input type="checkbox"/>) 推荐格式: application/msword; application/pdf				
电子版论文出版 (发布) 者	电子版论文出版 (发布) 地		权限声明	
论文总页数*	70			
注: 共 33 项, 其中带*为必填数据, 为 22 项。				