

<<从Windows到Linux的应用移植>>

图书基本信息

书名：<<从Windows到Linux的应用移植实现>>

13位ISBN编号：9787118082524

10位ISBN编号：711808252X

出版时间：2013-1

出版时间：国防工业出版社

作者：兰雨晴，洪雪玉 著

页数：240

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<从Windows到Linux的应用移植>>

内容概要

应用系统的开发、运行与部署依赖于所选择平台的工具、语言、环境、开发技术、服务及其兼容的第三方软件。

因此进行移植时，需要考虑目标平台对上述因素的支持和实现情况。

具体地，可以从应用系统组件、用户工作环境、开发工具 / 语言 / 平台、应用开发技术等几个角度整体考虑。

《从windows到linux的应用移植实现：平台技术与接口篇》主要关注应用开发技术在Windows平台和Linux平台的实现情况，研究分析网络通信、多进程多线程、图形界面等技术在两个平台的实现差异性，并提供对应的实现方案。

在内容编排上，为了让读者对应用移植涉及的工作和过程有清晰的认识，《从windows到linux的应用移植实现：平台技术与接口篇》在第一部分定义了应用移植过程、主要阶段、各阶段主要工作、应用移植的几个角度等，然后分为不同的部分，介绍多项应用开发技术的移植实现。

其中，第二部分从第2章到第5章，介绍基于套接字网络通信技术的应用移植实现；第三部分从第6章到第9章，介绍基于并发多任务开发技术的应用移植实现；第四部分从第10章到第12章，介绍图形用户界面的应用移植实现。

<<从Windows到Linux的应用移>>

书籍目录

第一部分 基本理论 第1章 应用系统迁移移植基础 1.1 必要性分析 1.2 迁移移植原则 1.3 迁移移植过程
1.3.1 过程定义 1.3.2 迁移需求分析 1.3.3 关键技术迁移设计 1.3.4 编码迁移移植 1.4 平台差异性与移植对
策 1.4.1 应用系统组件 1.4.2 用户工作环境 1.4.3 开发工具/语言/平台 1.4.4 应用开发技术 1.4.5 本书主要内
容 第二部分 基于套接字网络通信技术的应用系统迁移移植 第2章 可移植性分析 2.1 平台实现 2.2 差异
性分析 2.2.1 SOCKET数据类型 2.2.2 fd set宏定义 2.2.3 错误码 2.2.4 资源限制 2.2.5 函数名称 第3章 基本功
能迁移移植 3.1 创建和连接 3.1.1 接口差异性 3.1.2 移植实现 3.2 通信和关闭 3.2.1 接口差异性 3.2.2 移植实
现 3.3 字节序 3.3.1 接口差异性 3.3.2 移植实现 第4章 高级功能迁移移植 4.1 socket选项 4.1.1 接口差异性
4.1.2 移植实现 4.2 I/O控制 4.2.1 接口差异性 4.2.2 移植实现 4.3 名称与地址转换 4.3.1 接口差异性 4.3.2 移
植实现 第5章 I/O模型迁移移植 5.1 模型概述 5.2 异步阻塞 5.2.1 实现概述 5.2.2 接口差异性 5.2.3 移植实现
5.3 异步选择 5.3.1 实现概述 5.3.2 接口差异性 5.3.3 移植实现 5.4 事件选择 5.4.1 实现概述 5.4.2 接口差异性
5.4.3 移植实现 5.5 重叠I/O 5.5.1 实现概述 5.5.2 接口差异性 5.5.3 移植实现 5.6 完成端口 5.6.1 实现概述
5.6.2 接口差异性 5.6.3 移植实现 第三部分 基于并发多任务开发技术的应用系统迁移移植 第6章 可移植
性分析 6.1 线程模型 6.2 差异性分析 6.2.1 进程创建 6.2.2 线程创建 6.2.3 IPC通信机制 第7章 进程编程迁
移移植 7.1 进程创建与退出 7.1.1 创建进程 7.1.2 进程退出 7.2 进程控制 7.2.1 终止进程 7.2.2 进程同步 7.3
进程属性 7.3.1 环境变量 7.3.2 进程优先级 7.3.3 当前进程ID 7.4 移植实现 第8章 线程编程迁移移植 8.1 线
程创建与退出 8.1.1 创建线程 8.1.2 线程退出 8.2 线程控制 8.2.1 终止线程 8.2.2 挂起/恢复线程 8.2.3 线程同
步 8.3 线程属性 8.3.1 线程优先级 8.3.2 线程ID 8.4 线程本地存储 8.4.1 分配与释放索引 8.4.2 获取与设置数
据 8.5 移植实现 第9章 IPC机制迁移移植 9.1 匿名管道 9.1.1 创建与关闭 9.1.2 读写操作 9.1.3 移植实现 9.2
内存映射文件 9.2.1 创建与关闭 9.2.2 连接与分离 9.2.3 移植实现 9.3 邮槽 9.3.1 创建与关闭 9.3.2 读写操作
9.3.3 移植实现 9.4 互斥锁 9.4.1 创建与销毁 9.4.2 加锁与解锁 9.4.3 实现 9.5 信号量 9.5.1 创建与销毁 9.5.2
获取与释放 9.5.3 移植实现 9.6 事件 9.6.1 创建与销毁 9.6.2 等待事件 9.6.3 设置与重置 9.6.4 移植实现 9.7
临界区 9.7.1 初始化与销毁 9.7.2 进入与离开 9.7.3 移植实现 第四部分 图形用户界面迁移移植 第10章 应
用开发框架 10.1 框架模式 10.1.1 MFC文档/视图 10.1.2 Ot模型/视图 10.1.3 相关开发类 10.2 应用类型 10.3
消息响应机制 第11章 图形绘制基础设施 第12章 图形界面开发类 参考文献

<<从Windows到Linux的应用移>>

章节摘录

版权页：插图：2.Linux对应接口 Linux共享内存的创建/打开操作相比Windows要简单一些，System V使用shmget（）函数来创建新的或取得已有的共享内存。

它在Linux系统库linux/shm.h中的定义是这样的：`#include #include int shmget (key_t key , int size , int shmflg)`；参数key指定了共享内存的关键字。

第二个参数size是创建的共享内存的大小，以字节为单位。

第三个参数shmflg是控制函数行为的标志量。

如果操作成功，函数返回共享内存的标识符。

而POSIX中调用函数shin_open（）创建/打开共享内存。

函数原型为：`#include #include #include int shm_open (const char name , int oflag , mode_t mode)`；参数name指定了将要创建/打开的共享内存对象。

参数oflag用来指定访问标识、创建标识和状态标识等，取值如下：`O_RDONLY`、`O_RDWR`、`O_CREAT`、`O_EXCL`、`O_TRUNC`。

mode指定了该共享内存的权限。

System V关闭共享内存是通过调用函数shmctl（）来实现的，它在Linux系统函数库include/linux/shm.h中的函数声明是这样的：`#include #include int shmctl (int shmqid , int cmd , struct shmid_ds buf)`；其中，参数shmqid是shmget（）函数返回的共享内存标识符。

第二个参数cmd是要执行的操作，可取的值有：`IPC_STAT`，表示将buf指向的shmid_ds结构中的数据设置为共享内存的当前关联值；`IPC_SET`，表示如果进程有足够的权限，就把共享内存当前关联的值设置为buf指向的结构中的值；`IPC_RMID`，表示删除共享内存。

参数buf指向一个包含共享内存模式和访问权限的结构。

需要说明的是，当执行IPC_RMID操作时，系统并不是立即将其删除，而只是将其标为待删，然后等待与其连接的进程断开连接。

只有当所有的连接都断开以后系统才执行真正的删除操作。

当然，如果执行IPC_RMID的时候没有任何的连接，删除操作将会被立即执行。

POSIX调用函数shm_unlink（）关闭共享内存。

函数原型为：`#include #include #include int shm_unlink (const char name)`；该函数类似于unlink（），它根据参数name删除指定共享内存对象的连接；当所有进程都删除了该共享内存对象的连接时，系统将删除共享内存并释放相应的内存空间。

<<从Windows到Linux的应用移植>>

编辑推荐

《从Windows到Linux的应用移植实现:平台技术与接口篇》主要关注应用开发技术在Windows平台和Linux平台的实现情况,研究分析网络通信、多进程多线程、图形界面等技术在两个平台的实现差异性,并提供对应的实现方案。

<<从Windows到Linux的应用移>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>