

# 北京理工大学

## 本科生毕业设计（论文）

代理内核操作系统实验 PKE 在 K210 开发板  
上的移植和改进

Transplantation and improvement of Proxy Kernel operating  
system experiment (PKE) on K210 board

学    院：	计算机学院
专    业：	计算机科学与技术
学生姓名：	张国安
学    号：	1120181447
指导教师：	陆慧梅

2022 年 5 月 16 日

## 原创性声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

特此申明。

本人签名：

日期：

年

月

日

## 关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名：

日期：

年

月

日

指导老师签名：

日期：

年

月

日

# 代理内核操作系统实验 PKE 在 K210 开发板上的移植和改进

## 摘 要

本文……。

摘要正文选用模板中的样式所定义的“正文”，每段落首行缩进 2 个字符；或者手动设置成每段落首行缩进 2 个汉字，字体：宋体，字号：小四，行距：固定值 22 磅，间距：段前、段后均为 0 行。阅后删除此段。

摘要是一篇具有独立性和完整性的短文，应概括而扼要地反映出本论文的主要内容。包括研究目的、研究方法、研究结果和结论等，特别要突出研究结果和结论。中文摘要力求语言精炼准确，本科生毕业设计（论文）摘要建议 300-500 字。摘要中不可出现参考文献、图、表、化学结构式、非公知公用的符号和术语。英文摘要与中文摘要的内容应一致。阅后删除此段。

**关键词：**北京理工大学；本科生；毕业设计（论文）

## **Transplantation and improvement of Proxy Kernel operating system experiment (PKE) on K210 board**

### **Abstract**

In order to study……

Abstract 正文设置成每段落首行缩进 2 字符，字体：Times New Roman，字号：小四，行距：固定值 22 磅，间距：段前、段后均为 0 行。阅后删除此段。

**Key Words: BIT; Undergraduate; Graduation Project (Thesis)**

## 目 录

摘 要 .....	I
Abstract .....	II
第 1 章 研究现状 .....	1
1.1 代理内核操作系统实验 .....	1
1.1.1 代理内核的概念 .....	1
1.1.2 代理内核的思想 .....	1
1.1.3 代理内核实验的基本介绍 .....	1
1.2 RISC-V 新型开放指令集和精简指令集介绍 .....	1
1.2.1 RISC-V 的基本介绍 .....	1
1.2.2 RISC-V 的特权级 .....	1
1.2.3 RISC-V 的中断、异常委托 .....	1
1.3 现有 K210 板子内核移植工作的参考 .....	1
1.3.1 K210 的基本信息 .....	1
1.3.2 uCore 的移植过程 .....	1
第 2 章 环境搭建 .....	2
2.1 软件环境 .....	2
2.1.1 编译工具链 .....	2
2.1.2 代码准备 .....	3
2.1.3 编辑器、IDE 选择 .....	4
2.1.4 K210 环境 .....	4
2.2 硬件环境 .....	5
2.2.1 K210 硬件要求 .....	5
第 3 章 编译流程及内核启动流程改造 .....	6
3.1 引入 RustSBI .....	6
3.1.1 SBI 背景与现状 .....	6
3.1.2 RustSBI 的中断和异常委托 .....	6
3.1.3 RustSBI 提供 bootLoader 功能与运行时服务 .....	6
3.1.4 使用 RustSBI 兼容 K210 旧版指令集 .....	6
3.2 编译流程改造 .....	6
3.2.1 编译流程改造的背景 .....	6
3.2.2 内存布局改造 .....	6

3.2.3	Makefile 改造 .....	8
3.2.4	编译自动化脚本编写 .....	9
3.3	内核启动流程改造 .....	9
3.3.1	用户程序加载 .....	9
3.3.2	内核程序入口点修改 .....	9
第 4 章	接口移植 .....	10
4.1	接口移植的背景 .....	10
4.2	移植所需接口梳理 .....	10
4.3	接口移植的技术方案 .....	10
4.4	接口移植的具体实现 .....	10
第 5 章	系统调用、中断、异常处理 .....	11
5.1	二级题目 .....	11
5.1.1	三级题目 .....	11
第 6 章	内存管理 .....	12
6.1	二级题目 .....	12
6.1.1	三级题目 .....	12
第 7 章	进程管理 .....	13
7.1	二级题目 .....	13
7.1.1	三级题目 .....	13
第 8 章	实验指导书编写及管理 .....	14
8.1	二级题目 .....	14
8.1.1	三级题目 .....	14
结 论	.....	15
参考文献	.....	16
附 录	.....	18
附录 A	L <sup>A</sup> T <sub>E</sub> X 环境的安装 .....	18
附录 B	BIThesis 使用说明 .....	18
致 谢	.....	19

## **第 1 章 研究现状**

### **1.1 代理内核操作系统实验**

#### **1.1.1 代理内核的概念**

#### **1.1.2 代理内核的思想**

#### **1.1.3 代理内核实验的基本介绍**

### **1.2 RISC-V 新型开放指令集和精简指令集介绍**

#### **1.2.1 RISC-V 的基本介绍**

#### **1.2.2 RISC-V 的特权级**

#### **1.2.3 RISC-V 的中断、异常委托**

### **1.3 现有 K210 板子内核移植工作的参考**

#### **1.3.1 K210 的基本信息**

#### **1.3.2 uCore 的移植过程**

## 第 2 章 环境搭建

### 2.1 软件环境

#### 2.1.1 编译工具链

step1. 访问 sifive 官网，下载 riscv gcc toolchain

<https://www.sifive.com/software>

step2. 找到 Prebuilt RISC-V GCC Toolchain。根据开发环境选择对应的版本。

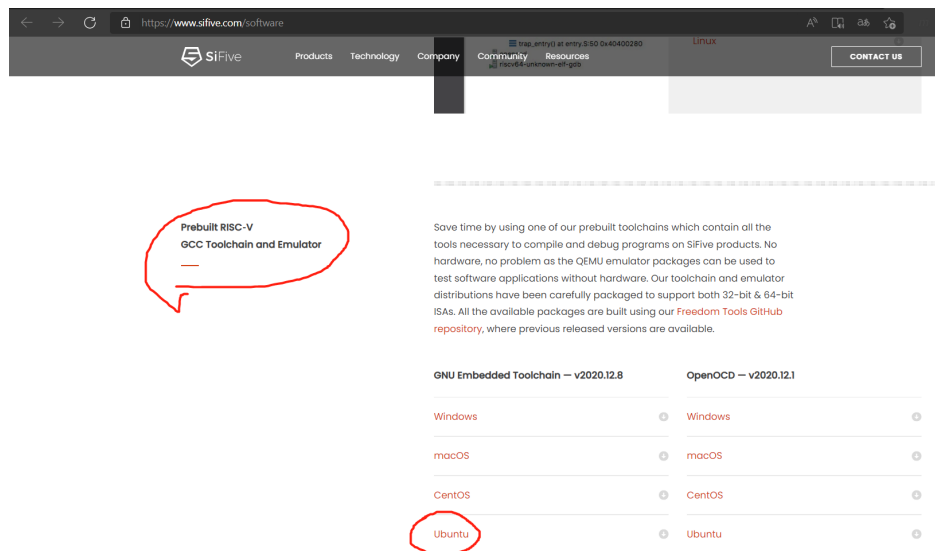


图 2-1 编译工具链列表

step3. 将下载好的 tar.gz 压缩包解压

```
1 tar -zxvf $your_tar_gz
```

代码 2-1: 解压命令

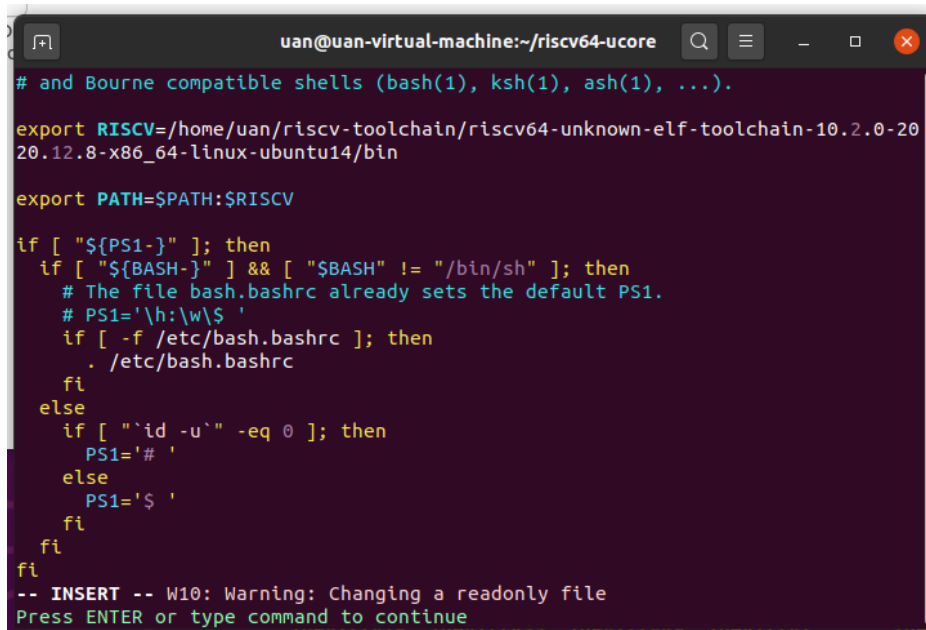
step4. 配置环境变量。解压完成得到文件夹，进入文件夹里的 bin 目录，打开 terminal，输入 pwd 获得当前路径。复制获得的路径。将复制到的路径加入系统的 PATH 环境变量。

```
1 vim /etc/profile
2 #添加以下两行到文件末尾
3 export RISCVC=$your_path
```



```
4 export PATH=$PATH:$RISCV
```

代码 2-2: 修改环境变量

A terminal window titled 'uan@uan-virtual-machine:~/riscv64-ucore' with a dark purple background and light green text. The terminal shows the following commands and output:

```
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).  
export RISCV=/home/uan/riscv-toolchain/riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-linux-ubuntu14/bin  
export PATH=$PATH:$RISCV  
if [ "${PS1-}" ]; then  
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then  
    # The file bash.bashrc already sets the default PS1.  
    # PS1='\h:\w\$ '  
    if [ -f /etc/bash.bashrc ]; then  
      . /etc/bash.bashrc  
    fi  
  else  
    if [ "`id -u`" -eq 0 ]; then  
      PS1='# '  
    else  
      PS1='$ '  
    fi  
  fi  
fi  
-- INSERT -- W10: Warning: Changing a readonly file  
Press ENTER or type command to continue
```

图 2-2 配置环境变量

#### step5. 加载环境变量文件

```
1 source /etc/profile
```

代码 2-3: 加载环境变量文件

#### step6. 验证编译环境

在终端输入 `riscv64-unknown-elf-gcc -v`，如果出现以下内容，则编译环境配置成功。

```
1 Using built-in specs.  
2 ....  
3 gcc version 10.2.0 (SiFive GCC-Metal 10.2.0-2020.12.8)
```

代码 2-4: 验证编译环境

### 2.1.2 代码准备

#### step1. 下载 riscv64-pke-k210 的代码库并查看所有分支

```
1 git clone git@github.com:BITzga/riscv64-pke-k210.git
2 git branch -a
```

代码 2-5: 下载代码库

step2. 根据开发需求选择分支。

如下所示，k210 前缀的代码分支是根据 k210 环境移植完成的代码。而其他普通分支则是 PKE 原先的代码。此时，根据开发需求使用 `git checkout` 命令选择分支即可。

```
1 remotes/origin/k210/lab1_1_syscall
2 remotes/origin/k210/lab1_2_exception
3 remotes/origin/k210/lab1_3_irq
4 remotes/origin/k210/lab2_1_pagetable
5 remotes/origin/k210/lab2_2_allocatepage
6 remotes/origin/k210/lab2_3_pagefault
7 remotes/origin/k210/lab3_1_fork
8 remotes/origin/k210/lab3_2_yield
9 remotes/origin/k210/lab3_3_rrsched
10 remotes/origin/lab1_1_syscall
11 remotes/origin/lab1_2_exception
12 remotes/origin/lab1_3_irq
13 remotes/origin/lab2_1_pagetable
14 remotes/origin/lab2_2_allocatepage
15 remotes/origin/lab2_3_pagefault
16 remotes/origin/lab3_1_fork
17 remotes/origin/lab3_2_yield
18 remotes/origin/lab3_3_rrsched
19 remotes/origin/master
```

代码 2-6: 分支列表

### 2.1.3 编辑器、IDE 选择

### 2.1.4 K210 环境

- Python3 环境

由于烧录程序和串口调试工具都是使用 Python3 编写的。Python 是解释型语言，所以我们需要安装 Python3 解释器。除此之外我们还需要安装 Python 包管理工具 pip。

```
1      sudo apt-get install python3
2      sudo apt-get install python3-pip
3
```

代码 2-7: 安装 Python3 环境

- 烧录工具我们编写好的程序，经过编译，变成 bin 文件，还需要烧录到 K210 上才能运行。而烧录需要借助烧录工具，这里我们使用了 K-Flash。
- 串口调试工具内核在运行时，输出信息是通过串口输出的，我们需要一个串口调试工具来接收 K210 上的串口信息。这里我们需要安装并使用 miniterm。

```
1      sudo apt-get install miniterm
2
```

代码 2-8: 安装 miniterm

- RustSBI-K210 支持包 SBI 是 RISC-V 的规范之一，它规定了监管者二进制 (Supervisor Binary Interface) 接口。RustSBI-K210 是 SBI 标准的一种实现，它使用 Rust 语言进行编写，具有性能安全的特点。除此之外，RustSBI-K210 还对 K210 板子提供了特殊的支持。它还可以在 K210 上作为我们内核程序的 bootloader。我们需要使用 RustSBI-K210 支持包来支持内核移植，这里我们需要在烧录内核时引入 RustSBI-K210 支持包。

在这里，我们可以下载到 RustSBI-K210 的 release 版本

<https://github.com/rustsbi/rustsbi-k210/releases>

## 2.2 硬件环境

### 2.2.1 K210 硬件要求

硬件环境较为简单，我们只需要一块具有串口功能的 K210 板子和一根数据线即可。

## 第 3 章 编译流程及内核启动流程改造

### 3.1 引入 RustSBI

#### 3.1.1 SBI 背景与现状

移植 K210 时，PKE 需要依赖 SBI（Supervisor Binary Interface）提供 BOOTLOADER 和 RUNTIME 功能，所以烧录内核时需要带上 SBI 固件。通过调研发现，OpenSBI 与 RustSBI（用 Rust 语言实现的 SBI）均按照 SBI 标准实现。这两种也是业内使用最多的开源 SBI。qemu 就是用了 OpenSBI 为 RISC-V 提供了环境支持。此外，RustSBI 还对 K210 做了特殊支持。所以目前暂定使用 RustSBI 当作 SBI 固件，为 PKE 提供 BOOTLOADER 功能和 RUNTIME 运行时服务。移植过程中我们不需要关心 RustSBI 的具体实现，只需要根据 SBI 标准调用其接口即可。

除此之外，引入 SBI，可以便于内核在其他板子上运行。当更换板子时，不需要改变内核代码，只需要更改 RustSBI 的支持包，获取对应硬件平台的支持包即可。这也简化了后续内核在其他芯片的移植工作。

#### 3.1.2 RustSBI 的中断和异常委托

#### 3.1.3 RustSBI 提供 bootLoader 功能与运行时服务

#### 3.1.4 使用 RustSBI 兼容 K210 旧版指令集

RustSBI 在 K210 兼容了高版本的指令。K210 实现的 RISC-V 指令集是 1.9.1 标准的。目前最新的特权级标准已经达到 1.11。如果我们的内核代码里有用到更高级的 RISC-V 汇编指令，可能会在 K210 上无法运行。这种情况下，就要改动内核的代码，会带来许多工作量。因此，使用 RustSBI 可以使我们免去处理 RISC-V 汇编版本的麻烦。

### 3.2 编译流程改造

#### 3.2.1 编译流程改造的背景

#### 3.2.2 内存布局改造

由于我们引入了 RustSBI，RustSBI 需要占用 0x80000000-0x8001FFFF 的物理内存空间。所以，内核的程序入口点由此发生了变化。我们需要修改内核 lds 文件，更

改了程序的入口点以保证内核可以正常运行。

首先，我们现在 `mentry.S` 中加入以下两行代码，用以确保 `_mentry` 是内核的程序入口点。

```
1  .globl _mentry
2  .section .text.prologue, "ax"
3  _mentry:
```

代码 3-1: 修改内核程序入口点

确定了内核的程序入口点，还需要把程序入口点的地址设置为 `0x80020000`，这需要我们对内存布局进行改造。修改 `BASE_ADDRESS`，赋值为 `0x80020000`。并设置代码段的起始地址为 `BASE_ADDRESS`，自此，内存布局就修改完成了。

```
1  OUTPUT_ARCH(riscv)
2  ENTRY(_mentry)
3  BASE_ADDRESS = 0x80020000;
4  SECTIONS
5  {
6      /*-----*/
7      /* Code and read-only segment */
8      /*-----*/
9
10     /* Begining of code and text segment, starts from DRAM_BASE to be
effective before enabling paging */
11     . = BASE_ADDRESS;
12
13     /* text: Program code section */
14     .text :
15     {
16         stext = .;
17         *(.text.prologue);
18         *(.text .stub .text.* .gnu.linkonce.t.*);
19         . = ALIGN(0x1000);
20         .....
21     }
```

代码 3-2: 修改内存布局

### 3.2.3 Makefile 改造

由于我们需要引入 RustSBI，并需要将其打包进入内核。除此之外，还需要将内核烧录到 K210，并与 K210 进行串口通讯。现有的 Makefile 并不支持这些工作。因此，我们需要修改 MakeFile。

```
1 $(KERNEL_K210_TARGET): $(KERNEL_TEMP_TARGET) $(BOOTLOADER)
2 $(COPY) $(BOOTLOADER) $@
3 $(V)dd if=$(KERNEL_TEMP_TARGET) of=$@ bs=128K seek=1
```

代码 3-3: 修改 Makefile

整体的编译流程为：

#### 1. 打包内核

```
1 $(KERNEL_K210_TARGET): $(KERNEL_TEMP_TARGET) $(BOOTLOADER)
2 $(COPY) $(BOOTLOADER) $@
3 $(V)dd if=$(KERNEL_TEMP_TARGET) of=$@ bs=128K seek=1
```

代码 3-4: 打包内核

以上步骤是为了把 rust-sbi.bin 和 pke.img 打包成 kernel.img。bs=128k 意味着输入/输出的 block 大小为 128k，seek=1 意味着跳过第零个 block 进行复制操作。也就是说，内核镜像里第零个 block 存放着 rust-sbi.bin，第一个 block 才开始存放 pke.img。128k 对应着十六进制 0x20000，也就是二进制的 0010 0000 0000 0000 0000。

我们的 kernel.img 放置在 0x80000000 处，再加上以上原因，pke 的地址自然就是 0x8020000。所以，我们需要在链接脚本 kernel.lds 里指定内核起始地址为 0x8020000，这也相当于告诉 SBI 这是内核的起始地址。当 SBI 在行使 bootloader 的功能时，会跳转到 0x8020000，将控制权转接给内核。

#### 2. 烧录

用数据线将 K210 与上位机连接，再使用 kflash，指定好相关参数即可完成烧录。

```
1 $(PYTHON) compile_tool/kflash.py -p $(PORT) -b 1500000 $(KERNEL_K210_TARGET)
```

代码 3-5: 烧录

#### 3. 运行 minitem，与 K210 进行串口通讯

```
1 $(TERM) --eol LF --dtr 0 --rts 0 --filter direct $(PORT) 115200
```

代码 3-6: 运行 minitem

打开 miniterm, 接收 K210 的串口打印输出。

#### 4. 总结

最小可执行内核在 K210 的运行流程是：指定内核起始地址-> 打包完整内核镜像-> 烧录到 flash-> 引导程序加载和运行 RustSBI-> RustSBI 运行并跳转到内核的指定地址-> RustSBI 将控制权交接给内核-> 内核运行

#### 3.2.4 编译自动化脚本编写

### 3.3 内核启动流程改造

#### 3.3.1 用户程序加载

在原先 pke 的中，是通过调用 spike 接口，进而调用 linux 的文件系统接口来加载用户程序的。而在 K210 上，我们没有 spike 的环境支持，不能直接调用 spike 接口。因此，加载用户程序就需要自行实现文件系统，或者使用其他办法。

由于文件系统的实现较为繁琐，其工作量会阻塞这个移植进度。因此，我们暂时不实现文件系统，而是采用获取用户程序地址，再加载的办法来实现这个需求。

具体的做法是：

1. 将用户程序、内核和 RustSBI 一起编译打包到 kernel.img
2. 使用 objdump 命令查找到用户程序 main 函数的地址
3. 得到地址以后，把地址的值赋值到内核加载用户程序处

通过这种技术方案，我们可以用较低的开发成本实现用户程序加载。

#### 3.3.2 内核程序入口点修改

由于 RustSBI 已经运行在 M 态，并且为我们提供了许多运行时服务。有了 RustSBI，在 K210 上，pke 运行在 M 态会破坏 RustSBI 的设计，因此 pke 只需要运行在 S 态即可。

这样，我们就可以直接将 pke 的 M 态代码根据自身需求迁移到 S 态代码。迁移完成后，需要更改内核程序入口点至 S 态入口

修改 mentry.S 文件，将 call m\_start 替换成 call s\_start

## 第 4 章 接口移植

### 4.1 接口移植的背景

正文……<sup>[1]</sup>

### 4.2 移植所需接口梳理

### 4.3 接口移植的技术方案

### 4.4 接口移植的具体实现

正文……<sup>[2]</sup>



## 第 5 章 系统调用、中断、异常处理

### 5.1 二级题目

正文……<sup>[1]</sup>

#### 5.1.1 三级题目

正文……<sup>[2]</sup>

## 第 6 章 内存管理

### 6.1 二级题目

正文……<sup>[1]</sup>

#### 6.1.1 三级题目

正文……<sup>[2]</sup>

## 第 7 章 进程管理

### 7.1 二级题目

正文……<sup>[1]</sup>

#### 7.1.1 三级题目

正文……<sup>[2]</sup>

## 第 8 章 实验指导书编写及管理

### 8.1 二级题目

正文……<sup>[1]</sup>

#### 8.1.1 三级题目

正文……<sup>[2]</sup>

## 结 论

本文结论……。<sup>[3]</sup>

结论作为毕业设计（论文）正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。阅后删除此段。

结论正文样式与文章正文相同：宋体、小四；行距：22 磅；间距段前段后均为 0 行。阅后删除此段。

## 参考文献

### 参考文献书写规范

参考国家标准《信息与文献参考文献著录规则》【GB/T 7714—2015】，参考文献书写规范如下：

#### 1. 文献类型和标识代码

普通图书：M      会议录：C      汇编：G      报纸：N

期刊：J      学位论文：D      报告：R      标准：S

专利：P      数据库：DB      计算机程序：CP      电子公告：EB

档案：A      舆图：CM      数据集：DS      其他：Z

#### 2. 不同类别文献书写规范要求

##### 期刊

[序号] 主要责任者. 文献题名 [J]. 刊名, 出版年份, 卷号 (期号): 起止页码.

[1] 余雄庆. 飞机总体多学科设计优化的现状与发展方向[J]. 南京航空航天大学学报, 2008(04): 417-426.

[2] Hajela P, Bloebaum C L, Sobieszczanski-Sobieski J. Application of global sensitivity equations in multidisciplinary aircraft synthesis[J]. Journal of Aircraft, 1990, 27(12): 1002-110.

##### 普通图书

[序号] 主要责任者. 文献题名 [M]. 出版地: 出版者, 出版年. 起止页码. [4]

[3] 李成智, 李小宁, 田大山. 飞行之梦: 航空航天发展史概论[M]. 北京: 北京航空航天大学, 2004.

[4] Raymer, Daniel P. Aircraft design: A Conceptual Approach[M]. Reston, Virginia: American Institute of Aeronautics, 1992.

##### 会议论文集

[序号] 析出责任者. 析出题名 [A]. 见 (英文用 In): 主编. 论文集名 [C]. (供选择项: 会议名, 会址, 开会年) 出版地: 出版者, 出版年. 起止页码. [5]

[5] 孙品一. 高校学报编辑工作现代化特征[C]//张为民. 中国高等学校自然科学学报研究会. 科技编辑学论文集 (2). 北京: 北京师范大学出版社, 1998: 10-22.

##### 专著中析出的文献

[序号] 析出责任者. 析出题名 [A]. 见 (英文用 In): 专著责任者. 书名 [M]. 出版地: 出版者, 出版年. 起止页码. [6]

[6] 罗云. 安全科学理论体系的发展及趋势探讨[M]//白春华, 何学秋, 吴宗之. 21 世纪安全科学与技术的发展趋势. 北京: 科学出版社, 2000: 1-5.

##### 学位论文

[序号] 主要责任者. 文献题名 [D]. 保存地: 保存单位, 年份. [7][8]

[7] 张和生. 嵌入式单片机系统设计[D]. 北京: 北京理工大学, 1998.

[8] Sobieski I P. Multidisciplinary Design Using Collaborative Optimization[D]. United States – California: Stanford University, 1998.

### 报告

[序号] 主要责任者. 文献题名 [R]. 报告地: 报告会主办单位, 年份. [9][10]

[9] 冯西桥. 核反应堆压力容器的 LBB 分析[R]. 北京: 清华大学核能技术设计研究院, 1997.

[10] Sobieszczanski-Sobieski J. Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems[R]. NASA CP-3031, 1989.

### 专利文献

[序号] 专利所有者. 专利题名 [P]. 专利国别: 专利号, 发布日期. [11]

[11] 姜锡洲. 一种温热外敷药制备方案: 88105607[P]. 中国. 1989-07-26.

### 国际、国家标准

[序号] 标准代号. 标准名称 [S]. 出版地: 出版者, 出版年. [12]

[12] GB/T 16159—1996. 汉语拼音正词法基本规则[S]. 北京: 中国标准出版社, 1996.

### 报纸文章

[序号] 主要责任者. 文献题名 [N]. 报纸名, 出版年, 月 (日): 版次. [13]

[13] 谢希德. 创造学习的思路[N]. 人民日报, 1998-12-25(10).

### 电子文献

[序号] 主要责任者. 电子文献题名 [文献类型/载体类型]. 电子文献的出版或可获得地址 (电子文献地址用文字表述), 发表或更新日期/引用日期 (任选). [14]

[14] 姚伯元. 毕业设计 (论文) 规范化管理与培养学生综合素质[EB/OL]. 中国高等教育网教学研究. [2013-03-26]. <http://www.cnnic.net.cn/hlwfzyj/hlwxyzbg/201201/P020120709345264469680>.

关于参考文献的未尽事项可参考国家标准《信息与文献参考文献著录规则》(GB/T 7714—2015)

## 附 录

附录相关内容...

### 附录 A L<sup>A</sup>T<sub>E</sub>X 环境的安装

L<sup>A</sup>T<sub>E</sub>X 环境的安装。

### 附录 B B<sub>I</sub>Thesis 使用说明

B<sub>I</sub>Thesis 使用说明。

附录是毕业设计（论文）主体的补充项目，为了体现整篇文章的完整性，写入正文又可能有损于论文的条理性、逻辑性和精炼性，这些材料可以写入附录段，但对于每一篇文章并不是必须的。附录依次用大写正体英文字母 A、B、C……编序号，如附录 A、附录 B。阅后删除此段。

附录正文样式与文章正文相同：宋体、小四；行距：22 磅；间距段前段后均为 0 行。阅后删除此段。



## 致 谢

值此论文完成之际，首先向我的导师……

致谢正文样式与文章正文相同：宋体、小四；行距：22 磅；间距段前段后均为 0 行。阅后删除此段。