

ART

【博客藏经阁丛书】

程序分析与设计

王宇行 编著



北京航空航天大学出版社

TP332/132

2008

博客藏经阁丛书

ARM[®]程序分析与设计

王宇行 编著

ARM[®]程序分析与设计

王宇行 著

青景萍 译

北京航空航天大学出版社

北京航空航天大学出版社

内 容 简 介

并丛函登藏客新

以实例阐述知识点,从易到难,系统阐述 ARM 嵌入式开发的知识和技能。内容包括:ARM 开发工具,ARM 映像文件分析,ARM 汇编语言,ARM C 语言,标准 C 库的应用,ARM 汇编语言和 C 语言混合编程,引导代码分析,ARM 中断处理原理和实现方法,调试的基本原理,scatter 文件分析,位置无关代码和数据。

本书可作为学习 ARM 嵌入式技术的培训教材或 ARM 嵌入式系统开发人员的参考书,也可作为高等院校电子工程、自动控制和计算机等专业的参考书。

图书在版编目(CIP)数据

ARM 程序分析与设计/王宇行编著. —北京:北京航空航天大学出版社, 2008. 3

ISBN 978-7-81124-252-2 著者 王宇行

I. A… II. 王… III. 微处理器, ARM—系统设计 IV. TP332

中国版本图书馆 CIP 数据核字(2008)第 010089 号

© 2008, 北京航空航天大学出版社, 版权所有。

未经本书出版者书面许可,任何单位和个人不得以任何形式或手段复制或传播本书内容。
侵权必究。

ARM® 程序分析与设计

王宇行 编著

责任编辑 张冀青

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话:010-82317024 传真:010-82328026

<http://www.buaapress.com.cn> E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本:787 mm×960 mm 1/16 印张:21 字数:470千字

2008 年 3 月第 1 版 2008 年 3 月第 1 次印刷 印数:5 000 册

ISBN 978-7-81124-252-2 定价:32.00 元

前言

嵌入式技术与 ARM

在现实生活中,嵌入式设备随处可见,成为人们生活的一部分,例如手机、数码相机、个人数字助理(PDA)等。中国作为嵌入式设备消费和制造大国,对嵌入式技术人才的需求是巨大的。ARM 处理器(它的全称是 Advanced RISC Machines)是目前世界上最为流行的和应用最为广泛的 32 位处理器之一,它被嵌入到日常生活使用的各种产品中,从手机到汽车里的定位仪器。它体积小,质量轻,成本低,可靠性高,而且高性能、低功耗,是嵌入式设备的核心。社会对基于 ARM 嵌入式技术的软硬件设计人才、应用开发人才的需求越来越大,同时基于 ARM 系统的开发人员也越来越庞大,有初学者,也有身经百战的资深工程师。这就是本书诞生的根本缘由。本书的目的,就是为读者提供学习和开发 ARM 系统所需要的技能和经过消化的知识点;使读者掌握基础知识的同时,系统地掌握基于 ARM 的嵌入式系统的开发方法,以最快的速度 and 最短的时间掌握 ARM 系统下的开发技术。

本书特点

(1) 基础与实践的结合。本书详细地以实例阐述了开发基于 ARM 的嵌入式系统所需要的基础知识,而采用的实例都来源于实际的工程开发,所以可以在实际的工程项目中直接采用。对于入门者,可以学习基础知识;对于资深工程师,可以强化基础知识,也可以在实际的开发中参考借鉴。

(2) 分散与系统的结合。本书独立地阐述了基于 ARM 的嵌入式系统基础知识,同时也用实例系统地阐述了嵌入式系统的开发流程,采用点面结合的方法,帮助读者系统地掌握 ARM 嵌入式系统的开发技术。

主要内容

全书共分 11 章,用实例体现知识点,从易到难,系统地阐述了基于 ARM 的嵌入式开发知识和技能。

第 1 章:描述 ARM 开发工具及其开发的整个过程,包括编译、链接以及最终可执行映像文件的产生。对常用的编译参数和 makefile 作了简单的描述。目的是,给出一个总体开发轮

廓和概念,为理解以后的章节做准备。

第2章:系统地介绍 ARM 编译器 armcc、armasm 输出的目标文件,以及 ARM 链接器输出的可执行映像文件的物理结构和逻辑结构,详细描述与之相关的一些技术和实现实例:装载域和运行域的存储映射,分散装载技术。

第3章:详细阐述 ARM 处理器的基本概念、ARM 汇编语言编程及大量的 ARM 汇编实例。

第4章:介绍用 C 语言开发 ARM 系统常用的一些概念,以及 C 语言编程的优化技术。

第5章:介绍标准 C 库的使用方法、使用流程以及对标准 C 库的裁剪方法,以满足实际的目标系统。

第6章:以实例阐述 ARM 汇编语言和 C 语言混合编程的技术,包括内联汇编、嵌入式汇编以及 ARM 汇编语言和 C 语言函数相互调用的编程技术。

第7章:详细阐述系统初始化原理和流程。对系统初始化过程中相关的硬件作了详细描述。

第8章:详细阐述中断处理原理和实现方法。以一个外部中断为例来介绍编写中断服务程序的流程和一些基本概念。

第9章:介绍调试的基本原理和技术。重点讨论基于 JTAG 调试的基本理论和方法。

第10章:阐述分散装载技术所需要的配置文件和分散装载技术在实际开发中的使用技术和方法。

第11章:用实例阐述位置无关代码和数据的原理和开发方法。

主要读者

本书对 ARM 处理器从基本概念到整个系统开发做了详细的阐述,涉及每个概念和知识点,不是简单的罗列,而是以通俗易懂的语言和详细的实例来阐述它,使读者易于消化和吸收。通过阅读本书,读者可以系统地掌握 ARM 开发的各方面知识,所以,本书可以作为学习 ARM 技术的培训教材或 ARM 开发人员的参考用书,也可以作为高等院校电子工程、自动控制和计算机等专业的参考书。

致谢

感谢我的很多同事提供了技术上的建议,感谢我的家人对我的支持和关心。由于时间仓促和水平所限,本书难免有疏漏和不足之处,请读者批评指正并给出宝贵意见。联系方式 E-mail:micro9229@gmail.com,或登陆我的博客 <http://micro9229.bokee.com/index.html>,与我进一步交流。

2008年1月

目 录

3.1.1	ARM 处理器模式	35
3.1.2	寄存器	36
3.1.3	处理器工作状态	42
3.1.4	ARM 数据长度	42
3.1.5	ARM 存储系统	42
3.1.6	存储和地址相关内容	43
3.2	ARM 汇编程序的结构	47
3.2.1	段	47
3.2.2	标识符	48
3.2.3	程序入口	50
3.2.4	程序结束	50
3.2.5	包含其他汇编源文件	50
3.2.6	引用外部标识符	51
3.3	ARM 汇编程序的常量和变量	52
3.3.1	常 量	52
3.3.2	变 量	53
3.4	ARM 汇编程序的运算符和表达式	57
3.4.1	数字表达式	57
3.4.2	逻辑表达式	58
3.4.3	字符串表达式	59
3.5	ARM 汇编程序的数据定义	62
3.5.1	LTORG	63
3.5.2	DCB、DCW、DCD 和 SPACE	67
3.5.3	MAP 和 FIELD	69
3.6	ARM 汇编程序的控制结构	72
3.6.1	选择结构	72
3.6.2	循环结构	72
3.6.3	选择结构和循环结构实例	73
3.7	ARM 汇编指令	74
3.7.1	数据处理指令	75
3.7.2	比较指令	84
3.7.3	存储器访问指令	89
3.7.4	堆栈操作指令	97
3.7.5	交换指令	101

3.7.6 跳转指令	103
3.7.7 条件执行指令	105
3.7.8 软件中断指令 SWI	111
3.8 ARM 程序和 Thumb 程序混合使用	114
3.8.1 混合使用的原因	114
3.8.2 ARM 和 Thumb 状态的切换方式	114
3.9 汇编语言实例	117
3.9.1 字符串处理	117
3.9.2 十进制数用七段数码管显示	119
3.9.3 系统初始化简介	121
3.9.4 七段数码管显示	124
第 4 章 ARM C 语言基础	
4.1 数据类型	131
4.1.1 基本数据类型	131
4.1.2 数据类型修饰符 signed 和 unsigned	132
4.2 常 量	133
4.3 变 量	134
4.4 操作符	137
4.5 控制结构	140
4.5.1 选 择	140
4.5.2 循 环	143
4.6 结构体	145
4.7 编译指令	147
4.7.1 #define 和 #undef	147
4.7.2 #if 和 #endif	148
4.7.3 #error	148
4.8 C 程序优化	149
4.8.1 除法和求余运算	149
4.8.2 条件执行	151
4.8.3 关系表达式	152
4.8.4 循 环	154
4.8.5 寄存器分配	155
4.8.6 函 数	158

第 5 章 标准 C 库的应用

5.1 标准 C 库的组成	162
5.1.1 与目标硬件无关的库函数	162
5.1.2 与目标硬件相关的库函数	162
5.2 标准 C 库的位置	163
5.3 标准 C 库的使用流程	163
5.4 标准 C 库函数的移植和重定向实例	166
5.5 ARM 编译器对 C 语言的扩展	170
5.5.1 __irq	171
5.5.2 __swi	172
5.5.3 __asm	173
5.5.4 __inline	174
5.5.5 __weak	174
5.5.6 __register	175
5.5.7 __int64	175
5.5.8 __pure	176
5.5.9 __value_in_regs	176

第 6 章 ARM 汇编语言和 C 语言混合编程

6.1 C 语言里嵌入汇编指令	177
6.1.1 内联汇编	177
6.1.2 嵌入式汇编	179
6.2 C 程序调用汇编语言函数	181
6.3 汇编程序调用 C 语言函数	182
6.3.1 汇编程序中调用用户自定义的 C 函数	182
6.3.2 汇编程序中调用标准库函数	182
6.4 汇编程序访问 C 语言定义的全局变量	183
6.5 APCS 概述	183
6.5.1 寄存器的使用规则	184
6.5.2 堆栈的使用规则	184
6.5.3 参数的传递规则	184
6.5.4 函数返回值规则	185
6.5.5 参数传递和函数返回值总结	185

6.6 混合编程实例	187
6.6.1 生成伪随机数	187
6.6.2 数字转换成字符串	191
6.7 编写基于 ROM 的程序	195
6.7.1 系统初始化	195
6.7.2 裁剪库函数	201
6.7.3 中断处理	203
6.7.4 串口驱动	206
6.7.5 I/O 地址映射	207
6.7.6 生成并装载 ROM 映像文件	209
6.7.7 在 scatter 文件里定义堆栈	212
第 7 章 引导代码分析	
7.1 BOOT 相关硬件:Flash ROM	213
7.2 BOOT 的主要功能	214
7.3 主要功能分析	217
7.3.1 中断处理	217
7.3.2 初始化硬件	222
7.3.3 应用程序执行环境的初始化并跳到主程序	225
7.3.4 BOOT 流程图	226
7.3.5 地址重映射	232
第 8 章 ARM 中断处理	
8.1 中断概述	235
8.1.1 中断分类	235
8.1.2 响应时间	235
8.1.3 屏蔽和使能中断	236
8.1.4 可重入的中断处理程序	237
8.2 硬件中断处理流程	238
8.2.1 ARM 内核自动完成的动作	238
8.2.2 跳转到中断处理函数	238
8.2.3 中断处理完成后返回	239
8.2.4 中断处理流程图	240
8.3 硬件中断实现	240

目 录

8.3.1	定义中断服务程序	241
8.3.2	主程序	242
8.4	中断服务子程序中关键的变量类型	243
8.4.1	volatile	243
8.4.2	__irq	246
8.4.3	访问绝对地址的内存位置	246
第 9 章 调试基础		
9.1	调试构架	247
9.1.1	调试主机	247
9.1.2	协议转换器	248
9.1.3	调试目标	248
9.2	调试逻辑结构	252
9.3	JTAG 简介	252
9.3.1	基本概念	253
9.3.2	举 例	254
第 10 章 scatter 文件		
10.1	scatter 文件的作用	255
10.2	scatter 文件简介	255
10.2.1	对装载域的描述	256
10.2.2	对运行域的描述	256
10.2.3	对输入段的描述	257
10.3	scatter 文件的使用	257
10.3.1	系统挂接不同类型的存储器	257
10.3.2	复杂的内存映射	258
10.3.3	映射外部控制寄存器的地址	259
10.3.4	映射堆和栈的地址	260
10.3.5	scatter 文件实例	261
第 11 章 位置无关代码和数据		
11.1	只读段(RO)位置无关(PI)	263
11.1.1	编写位置无关的代码	263
11.1.2	位置无关的代码编译和链接参数	264

11.1.3 位置无关的代码段分析.....	265
11.2 读写段(RW)位置无关(PI)	273
11.2.1 可重入代码.....	274
11.2.2 位置无关的数据编译和链接参数.....	277
11.3 位置无关(PI)代码段和数据段编写实例分析.....	277
11.3.1 实例简介.....	277
11.3.2 symdefs 文件	278
11.3.3 实例分析.....	282
附录 A BOOT 源代码	307
附录 B ARM 处理器系列	320
参考文献.....	324

第 1 章

ARM 开发工具

工欲善其事，必先利其器。做嵌入式系统的开发，必须了解和熟知开发流程和开发工具，这是开发人员必须具备的基本能力。本章简要描述这两个专题。

1.1 开发流程

本节介绍从源程序到运行在硬件设备上的二进制代码的整个演变流程。整个流程包括下面 5 个步骤：

- ① 用 C 语言或汇编语言编写源代码。
- ② 用编译器把源代码编译成 ELF (Executable and Linking Format, 对于更为详细的解释, 请参见 http://www.arm.com/pdfs/DUI0101A_Elf.pdf) 的目标文件。
- ③ 用链接器把目标文件链接成 ELF 格式的可执行映像文件。
- ④ 用 fromelf 工具把可执行的映像文件转换成二进制映像文件。
- ⑤ 把二进制映像文件烧写到目标系统

二进制映像文件就是最终在 RAM (Random Access Memory) 运行的、完成特定功能的二进制代码。俗话说：一张图胜过于言万语。下面用图 1.1 来展示这个流程。

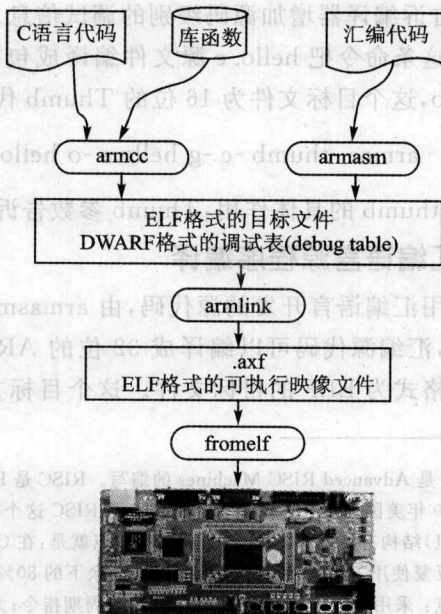


图 1.1 开发流程图

第1章 ARM 开发工具

1.2 开发工具

1.2.1 编译器简介

1. C 语言源程序编译

ARM^① 编译器 armcc 负责编译用 C 语言或 C++ 语言编写的源代码。根据编译参数的选择,可以编译成 32 位的 ARM 代码,也可以编译成 16 位的 Thumb^② 代码。最终输出格式为 ELF 的目标文件。这个目标文件可以包括 DWARF^③ 格式的调试表,也可以不包括,完全由编译参数控制。armcc 的使用方法如下:

编译 armcc -c -g hello.c -o hello.o。

这条命令把 hello.c 源文件编译成包括 DWARF 格式调试表的 ELF 格式的目标文件(hello.o)。这个目标文件为 32 位的 ARM 代码。参数 c 和 g 的具体作用如下:

-c: 告诉编译器只编译,不链接。

-g: 告诉编译器增加源码级别的调试信息。

下面这条命令把 hello.c 源文件编译成包括 DWARF 格式调试表的 ELF 格式的目标文件:hello.o,这个目标文件为 16 位的 Thumb 代码。

编译 armcc -thumb -c -g hello.c -o hello.o。

参数 thumb 的具体作用:Thumb 参数告诉编译器生成 16 位的 Thumb 代码。

2. 汇编语言源程序编译

对于用汇编语言开发的源代码,由 armasm 编译器负责编译。与 armcc 一样,根据编译参数的选择,汇编源代码可以编译成 32 位的 ARM 代码,也可以编译成 16 位的 Thumb 代码。最终输出格式为 ELF 的目标文件。这个目标文件可以包括 DWARF 格式的调试表,也可以

① ARM 是 Advanced RISC Machines 的缩写。RISC 是 Reduced Instruction Set Computer 的缩写,意思为精简指令集计算机。1979 年美国加州大学伯克利分校提出了 RISC 这个概念。传统的 CISC(Complex Instruction Set Computer,复杂指令集计算机)结构有其固有的缺点,最明显的缺点就是:在 CISC 指令集的各种指令中,其使用频率相差悬殊,大约有 20% 的指令会被反复使用,占整个程序代码的 80%;而余下的 80% 的指令却不经常使用,在程序设计中只占 20%。RISC 体系结构具有的特点:采用固定长度的指令格式;使用单周期指令;大量使用寄存器。

② Thumb 指令长度为 16 位。Thumb 指令集为 ARM 指令集的功能子集,但与等价的 ARM 代码相比较,可节省 30%~40% 的存储空间,同时具备 32 位代码的所有优点。

③ 调试器调试程序,需要调试信息。如果在编译程序的时候加上参数 g,那么编译输出的目标文件就会包含调试信息。调试信息的表示格式有很多种,在 ARM 系统里使用 DWARF 格式。DWARF(Debugging With Attributed Record Formats)现在已经有 DWARF 1、DWARF 2 和 DWARF 3 三个版本。

不包括,完全由编译参数控制。

下面两个实例演示了 `armasm` 的使用方法。

① 编译成带调试信息的 32 位 ARM 代码:

```
编译 armasm -32 -g hello.s -o hello.o.
```

② 编译成带调试信息的 16 位 Thumb 代码:

```
编译 armasm -16 -g hello.s -o hello.o.
```

3. 目标文件

源代码经过 `armcc` 和 `armasm` 的编译,输出了 ELF 格式的目标文件。这个目标文件由编译参数 `g` 来决定是否包含 DWARF* 格式的调试表。调试表里的信息就是调试器调试程序时需要的信息,简称调试信息。如果在编译的时候,使用了参数 `g`,则在编译的时候,编译器会从源文件中收集变量名、变量类型、变量所在行号、函数名、函数参数、函数的地址范围、行号和地址的对应关系等调试器需要的信息;然后按照一种特定的格式写入到编译后的目标文件中。调试的时候,调试器便从文件中读取并解析这些信息,从而达到调试程序的目的。调试信息一般都是按照什么样的格式存放的呢?在 ARM 开发系统中使用 DWARF 格式来存放调试信息。为什么调试信息要用其他格式来表示呢?这是因为 ELF 标准没有定义调试信息的表示格式,所以才选择其他的标准来表示调试信息。当 ELF 文件里包含 DWARF 格式的调试信息时,常常称这个文件为 ELF/DWARF 1 或 ELF/DWARF 2 格式的文件。

`dwarf` 英文意思为有魔法的小矮人;`elf` 为小精灵。

编译器生成的目标文件由有魔法的小矮人和小精灵构成,这两个小人究竟是哪路神仙,在第 2 章会详细介绍。

1.2.2 链接器简介

在 ARM 系统开发中使用的链接工具为 `armlink`,`armlink` 涉及了几乎 ARM 开发的全部内容。它的每一个参数几乎都可以独立成章。如果用户掌握了它的全部,那么用户对 ARM 的掌握就达到了一个很高的层次。本节丢开细节,从总体上对它予以介绍。

1. 主要功能

简单地说,`armlink` 的工作就是合并一个或多个 ELF 格式的目标文件的内容,生成一个 ELF 格式的可执行文件(executable images),可以细分为下面几个任务:

① 解析目标文件之间的符号引用关系。

② 如果目标文件包含 C/C++ 库函数的调用,从 C/C++ 运行时库(run-time library)中

* 通过访问网站 <http://dwarf.freestandards.org> 可以了解 DWARF 标准的详细信息。

第1章 ARM 开发工具

提取相应的功能模块。

- ③ 将输入段按规则组成相应的输出段。
- ④ 如果目标文件里包含调试信息,则删除重复的调试信息。
- ⑤ 根据用户指定的分组和定位信息,建立映像文件的地址映射关系。
- ⑥ 重新定位需要重定位的变量的值。
- ⑦ 生成可执行的映像文件。

下面这条命令演示了 armlink 的基本用法,它把两个目标文件 test_1.o 和 test_2.o 链接成一个可执行的映像文件 test.axf。

```
链接 armlink -ro-base 0x8000 test_1.o test_2.o -o test.axf。
```

其中,参数 ro-base 和 o 的具体作用如下:

-ro-base 表示代码段(RO)在装载域(load view)和运行域(execution view)里的起始地址;

-o 表示输出文件为 test.axf。

对于装载域和运行域的具体含义,在第2章有详细的阐述。

1.2.3 fromelf 工具

ELF 格式的可执行的映像文件如 test.axf,在调试通过后,如果需要下载到正式的产品上,必须经过 fromelf 工具处理,掐头去尾,留下真正有用的执行代码。它把 armlink 生成的 ELF 格式的映像文件转换成适合在 ROM 和 RAM 里运行的各种格式的二进制代码。在 ARM 系统里用的二进制格式是:Plain binary format。如果在编译源代码时,使用 g 参数,那么在编译器输出的目标文件和 armlink 输出的映像文件里将包含调试信息,这些信息可以帮助开发人员来调试代码、修改程序错误。但对于最终产品来说,它就成了占用资源的垃圾了。因此,在调试成功之后,在下载到最后的产品之前,必须去掉这些调试信息。fromelf 就是用来完成这个功能的。

下面这个命令演示了 fromelf 的基本用法,它把 ELF 格式的映像文件 test.axf 转换成 bin (Plain binary format)格式的二进制代码。

```
fromelf test.axf --bin -o test.bin
```

其中,--bin 表示输出格式为 Plain binary。

二进制映像文件 test.bin 诞生了,下一步要做的,就是把它下载到最终产品。

fromelf 有一个参数:no_debug,可以去掉一个包含调试信息的可执行映像文件的调试信息。下面这条命令演示了 fromelf 的用法:去掉包含调试信息的可执行映像文件里的调试信息。

```
C:\arm\fromelf --no_debug --elf -o nodebug.axf debug.axf
```

当然,在默认情况下,fromelf 在把可执行映像文件转换成二进制映像文件时,去掉了调试信息。

1.2.4 常用的编译和链接参数

1. 编译参数

(1) armcc*

用户在 Windows 的命令行执行 armcc 或者 TCC 编译器命令 C:\arm>armcc(TCC),就会列出主要的编译参数,如下所示:

ARM/Thumb C/C++ Compiler, RVCT2.2 [Build 349] 编译器的版本号

Usage: armcc [option] file1 file2... file n 编译器的用法,其中的 options 就是各种编译参数。

主要[参数]的含义如下:

- --arm 编译器输出的为 ARM 指令集。
- --thumb 编译器输出的为 Thumb 指令集。
- --c90 编译的为 C 语言源程序。
- --cpp 编译的为 C++ 语言源程序。
- -O0 最小的优化级别,关闭了除代码变换外的所有优化功能。
- -O1 关闭了那些影响调试效果的优化功能。
- -O2 高的优化级别,提供所有的优化功能。
- -Ospace 从空间上优化代码,目的是追求紧凑的目标代码。
- -Otime 从时间(执行效率)上优化代码,目的是使代码的执行效率最好。
- --cpu <cpu> 处理器的型号或者 CPU 的体系版本号,这样可以让编译器根据处理器的特点编译出最优的代码。
- --cpu list 输出可供选择的处理器列表。
- -o <file> 编译器输出的文件名字,它和下面三个参数结合使用,输出不同类型的文件。
- -c 对源程序只编译,不进行链接,与-o 参数结合,编译输出目标文件,目标文件的名字就是-o 参数指定的名字。如果没有使用-o 参数,那么输出的目标文件名和源文件名相同。比如,源文件为 hello.c,则编译输出的目标文件为 hello.o。
- --asm 输出目标文件的同时,也输出相应的汇编代码。
- -S 只输出汇编代码,不输出目标代码。
- --interleave 和-asm 或-S 组合使用,同时输出汇编代码和源代码,如 C:\mix>armcc

* 对于 armcc 更为详细的用法,可以参考文档: http://www.arm.com/pdfs/DUI0205G_rvct_compiler_and_libraries_guide.pdf。