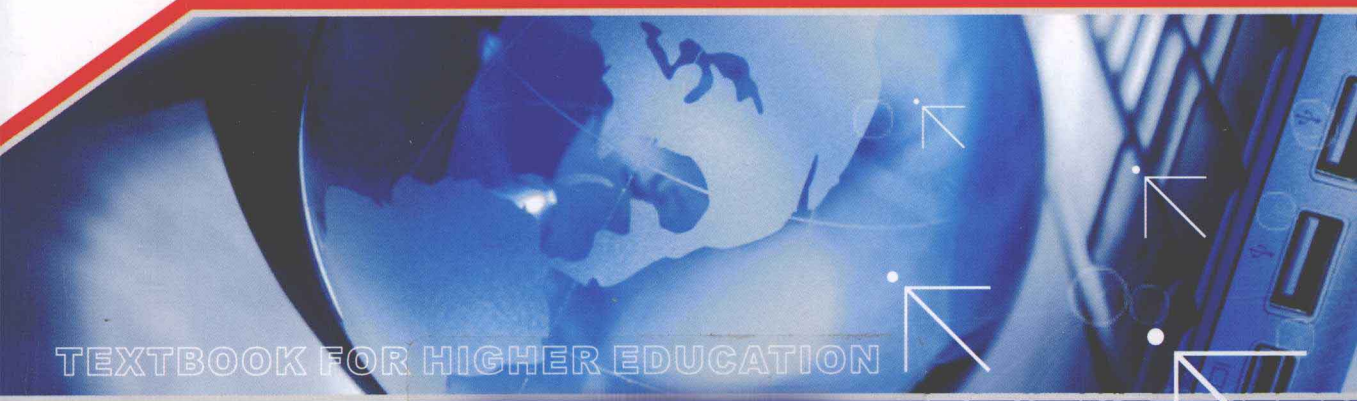
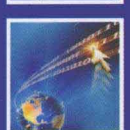
软件系统开发指导教程系列丛书

系统级编程

主编　李丹程　朱志良





西北工业大学出版社
NORTHWESTERN POLYTECHNICAL UNIVERSITY PRESS

软件系统开发指导教程系列丛书

系统级编程

主编　李丹程　朱志良

编者　刘国奇　姜琳颖　刘　莹

西北工业大学出版社

【内容简介】 本书以 Visual C++作为开发工具和平台，首先介绍了程序的调试、数据和方法的调用原理以及在计算机中数据的存储格式；其次介绍了存储空间的布局、分配和对于一般内存错误的处理办法；最后介绍了计算机操作系统的基本原理，主要涉及线程、进程及它们之间的调度等知识。

本书兼顾理论和实践两方面，让读者可以通过动手实践来掌握和理解知识要点。

本书既可作为高等院校计算机及其相关专业的教材，也可作为广大工程技术人员和自学者的参考用书。

图书在版编目(CIP)数据

系统级编程/李丹程，朱志良主编．—西安：西北工业大学出版社，2011.12
（软件系统开发指导教程系列丛书）
ISBN 978-7-5612-3245-3

Ⅰ.①系… Ⅱ.①李… ②朱… Ⅲ.①程序设计—教材 Ⅳ.①TP311

中国版本图书馆 CIP 数据核字(2011)第 249290 号

出版发行：西北工业大学出版社
通信地址：西安市友谊西路 127 号 **邮编**：710072
电　　话：(029)88493844　88491757
网　　址：www.nwpup.com
印 刷 者：陕西宝石兰印务有限责任公司
开　　本：787 mm×960 mm　1/16
印　　张：14.25
字　　数：304 千字
版　　次：2011 年 12 月第 1 版　2011 年 12 月第 1 次印刷
定　　价：27.00 元

《软件系统开发指导教程系列丛书》编委会

出版说明

2001年12月，教育部批准成立了35所国家示范性软件学院，旨在为国家软件产业的发展培养多层次、实用型、高水平、具有国际竞争力的专业人才，以适应社会对软件高端人才的需要。各软件学院在这样的大环境下，纷纷挖掘自身的优势，采用各种先进的教学模式，注重教育与教学改革，已形成了各具特色的软件工程教育体系。广大教师也在这样的教育教学中不断对传统软件工程教学进行总结，并汲取国外先进教学中的精髓，取长补短，积累了丰富的教学实践经验。

有鉴于此，我们组织策划了《软件系统开发指导教程系列丛书》。本系列丛书共10册，包括《计算机编程导论》《计算机系统导论》《面向对象编程基础——Java语言描述》《交互式用户界面设计与测试》《数据库系统——设计与应用》《数据结构与算法》《系统级编程》《网络与分布式计算》《软件规范测试与维护》《软件项目组织与管理》。本系列丛书的出版意欲将广大教师在培养国际化、应用型软件工程化人才的教育教学中积累的经验进行推广与传播。特别是将这种教学理念在一些外语基础薄弱，还不能适应双语教学的学生中推广。

本系列丛书的分册主编均是各软件学院活跃在教学第一线的教师。他们都具有多年的教学经验、深厚的专业功底和丰富的软件开发实践经验，因而保证了这套丛书理论与实践兼备，教与学互动，特色鲜明。

为确保本系列丛书的质量，我们邀请了软件学院的一些专家、教授，成立了《软件系统开发指导教程系列丛书》的编委会。他们当中有全国知名的计算机专家、科学院院士，也有在软件工程教育中有着丰富工作经验的教授、博导，也不乏具有出国留学经历的教师，他们对国际与国内该领域的技术状况、应用环境都十分了解。这些均有利于从总体上把握本系列丛书内容向着适应国内需要并与国际接轨的方向发展。

我们将以高度的社会责任感，投入满腔的工作热忱，精益求精，为广大读者提供高质量的精品图书。

西北工业大学出版社
2009年3月

前　言

本书将向读者提供一个更加宽广的视角来观察和学习计算机处理器、计算机网络以及计算机操作系统。读者将清晰地从本书学习到汇编程序和汇编码、程序执行性能评估和优化问题，以及计算机存储结构和存储层级，网络协议以及操作和同步问题。本书还向读者提供了书中所讲概念的实际项目应用，这也是让读者更好地理解知识的一种方式。本书中的项目都是运用C或C++语言来进行测试编写的。

本书的宗旨：

(1)学习程序执行中的重点细节，并理解程序的性能、对资源的需求和程序运行中的错误。

(2)学习当前处理器支持的基本数据类型以及它们所使用的操作符。

(3)了解计算机存储器的组织方式和性能。

(4)了解计算机指令集和面向对象程序的执行过程。

(5)了解程序执行性能的评估方法和改进方法。

(6)了解应用软件和操作系统之间相互交互，尤其是线程、进程、调度以及同步控制。

(7)了解异步输入/输出。

读者通过学习本书将具有如下能力：

1.编程能力方面

(1)能够操作比特位。

(2)能够编写存储分配器，用来发现存储器相关的编程错误。

(3)能够发现大型程序中的性能瓶颈。

(4)能够诊断并且改进由于非正规的内存读取方式而导致的系统错误。

(5)能够编写简单的并发程序，以便处理延迟问题。

2.工具使用方面

(1)能够使用C语言去编写一些底层的存储和数据操作程序。

(2)会使用程序概要分析工具去评估并改进程序的性能。

(3)能够使用调试工具去分析存储访问中的错误。

(4)能够使用底层循环计数器去统计系统中断和并发操作。

3.系统层次编程方面

(1)了解现代处理器的比特位和算术操作以及对于算术精度的限制。

(2)了解虚拟内存和缓存的基本概念和性能效果。

(3)了解一般的与存储器相关的编程错误。

(4)了解进程、线程和并发的相关概念。

4.成为一名高级编程人员

完成本书学习的读者可以作为高级编程人员,生产出开发周期短并且质量非常可靠的软件产品。具体说,这些编程人员能够做以下的事情:

(1)采用程序的主要部分来描述其性能。

(2)能够从本质上提高应用软件的性能,尤其是在速度上的提升。

(3)使用线程去提高程序处理用户请求的能力。

本书由李丹程、朱志良主编。全书共分7章,具体编写分工如下:第1,2章由刘国奇编写,第3,4章由刘莹编写,第5~7章由姜琳颖编写。石凯、李昕、毛克明、邓卓夫参与校对。

由于水平有限,书中难免存在不当和疏漏之处,恳请读者批评指正。

编　者

2011年6月

目　录

第1章 编程初探

计算机系统是由硬件和系统软件组成的，它们共同工作来运行应用程序。其实计算机本身只能完成一些很简单的基本操作，如加法、减法、传送数据、发控制电压脉冲等，这些简单的基本工作叫做计算机指令。一台计算机不过几十条指令，把它们合起来叫计算机的指令系统。计算机无论做多么复杂和高级的工作，都是靠着用指令适当地排列成一个序列，逐条地执行指令，最后完成整个工作。这种把指令排列成一定的执行顺序并能完成一定目标的工作的指令序列，就叫做程序。

以一个 hello 程序为例：

```
1. #include <stdio.h>
2.
3. int main()
4. {
5. printf("hello, world\n");
6. }
```

通过了解 hello 程序如何在计算机上工作来开始对计算机系统知识的学习。hello 程序从它被程序员创建开始，到它在计算机上被执行，进而输出简单的信息，最后终止，我们将沿着 hello 程序的生命周期展开学习。

1.1 电脑的最小信息单位

hello 程序的生命周期是从源程序 hello.c 开始的，该源程序实际上就是一个由 0 和 1 组成的位(bit)序列，但“位”这个单位太小，所以每 8 个“位”一组，组成字节 (Byte)。现在大部分系统使用 ASCII 标准来表示 8 位的文本字符。

hello.c 程序是以字节序列的方式存储在文件中的，每个字节都有一个整数值来对应于某个字符，如 # 是 35，i 是 105，n 是 110，c 是 99，换行符\n 是 10……如表 1.1 所示。hello.c 这种由 ASCII 字符构成的文件称为文本文件，其他则称为二进制文件。

表 1.1 hello.c 的 ASCII 文本文件

#	i	n	c	l	u	d	e	<sp>	<	s	t	d	i	o	.	h	>	\n	\n
35	105	110	99	108	117	100	101	32	60	115	116	100	105	111	46	104	62	10	10
i	n	t	<sp>	m	a	i	n	(	)	\n	{	\n	p	r	i	n	t	f	(
105	110	116	32	109	97	105	110	40	41	10	123	10	112	114	105	110	116	102	40
"	h	e	l	l	o	,	<sp>	w	o	r	l	d	\	n	"	)	;	\n	}
34	104	101	108	108	111	44	32	119	111	114	108	100	92	110	34	41	59	10	125

hello.c 的表示方法说明了一个基本的思想：系统中所有的信息，包括磁盘文件、存储器中的程序、存储器中存储的用户数据以及网络上传输的数据，都是由一串二进制数据表示的，区分不同数据对象的唯一方法是读到这些数据对象时的上下文，比如在不同的上下文中，同样的字节序列可能表示一个整数、浮点数、字符串或者机器指令。

电脑世界是由 0 与 1 组成的，其中有数以万计的线路，一条线路传递一个信号，而 0 代表没有信号，1 代表有信号，就像电源开关一样，同一时间只可能有一种状态，所以电脑最基本的单位就是一条线路的信号，把它称作“位”，英文叫做 bit，缩写为 b。

“位”(bit)虽然是电脑中最基本的单位，但“位”这个单位太小，所以由 8 个“位”单位组成一个更大的单位，叫字节(Byte)，它也是电脑存储容量的基本计量单位。Byte 可简写为 B，一个字节由 8 个二进制位组成，其最小值为 0，最大值为 11111111，一个存储单元能存储一个字节的内容。现在的计算机可以快速并发地处理多个二进制位。多个二进制位又构成了一个单位，称作字节或者字。一个字是由 32 个二进制位组成的。

在 C 语言里，可以通过声明整型变量来为一个字分配存储单元，如：

```
int my_word; //分配一个字的内存
```

1.2 程序转化成计算机能读懂的指令

在 hello 程序生命周期的一开始，是一个高级的文本 C 程序，处于这种形式时，它是能够被人读懂的。但为了在系统上运行 hello.c 程序，每条 C 语句都必须被其他程序转化为一系列的低级机器语言指令。然后这些指令按照一种称为可执行目标程序(executable object program)的格式打好包，并以二进制磁盘文件的形式存放起来，目标程序也称为可执行目标文件(executable object file)。

从源程序到目标文件是由编译器驱动程序(compiler driver)完成的。在这里编译器驱动程序读取源程序文件 hello.c，并把它翻译成一个可执行的目标文件 hello，这个过程分 4 个阶段，如图 1.1 所示。执行这 4 个阶段的程序(预处理器、编译器、汇编器和链接器)构成编译系统。

图 1.1　编译系统

1. 预处理阶段

读取 C 源程序，对其中的伪指令（以 # 开头的指令）和特殊符号进行处理。例如，根据 # 开头的命令修改原始的 C 程序，#include <stdio. h>告诉预处理器读取系统头文件 stdio. h 的内容并插入到程序文本中，从而得到另一个 C 程序，通常以. i 作为文件扩展名。或者说预处理阶段是扫描源代码，对其进行初步的转换，产生新的源代码提供给编译器。预处理过程先于编译器对源代码进行处理。

在 C 语言中，并没有任何内在的机制来完成如下一些功能：在编译时包含其他源文件、定义宏、根据条件决定编译时是否包含某些代码。要完成这些工作，就需要使用预处理程序。尽管在目前绝大多数编译器都包含了预处理程序，但通常认为它们是独立于编译器的。预处理过程读入源代码，检查包含预处理指令的语句和宏定义，并对源代码进行相应的转换。预处理过程还会删除程序中的注释和多余的空白字符。

2. 编译阶段

编译阶段以源程序作为输入，以目标程序作为输出，其主要任务是将源程序翻译成目标程序。即将高级程序设计语言书写的源程序，翻译成等价的用计算机汇编语言、机器语言或某种中间语言表示的目标程序的过程。例如，将文本文件 hello. i 翻译成文本文件 hello. s，它为一个汇编语言程序。汇编语言为不同高级语言的不同编译器提供了通用的输出语言，如 C 编译器和 Fortran 编译器产生的输出文件用的是一样的汇编语言。

在编译阶段，又分为 6 个子阶段，分别是词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成，如图 1.2 所示。

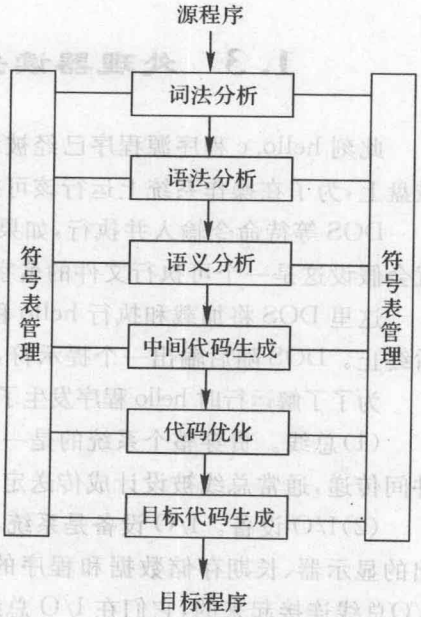


图 1.2　编译阶段的处理过程

3. 汇编阶段

将 hello. s 翻译成机器语言指令，把这些指令打包成一种叫可重定位目标程序的格式，结果保存在目标文件 hello. obj 中，hello. obj 是二进制文件，它的字节编码是机器语言指令而不是字符。

4. 链接阶段

由于 hello 程序调用了 printf 这个标准 C 库中的一个函数，而 printf 函数存在于一个名为 printf. o 的单独预编译目标文件中，链接器就负责将它并入到 hello. o 程序中，结果就得到 hello. exe 文件，它是一个可执行目标文件，可执行文件加载到存储器后由系统负责执行。

如果我们编写的程序不仅仅是一个 hello. c 程序，而是由多个源程序组成的，它们需要组织在一起，共同完成某个功能，这时候就像图 1.3 所示，同样经过预处理、编译、汇编、链接 4 个阶段，形成一个可执行文件。

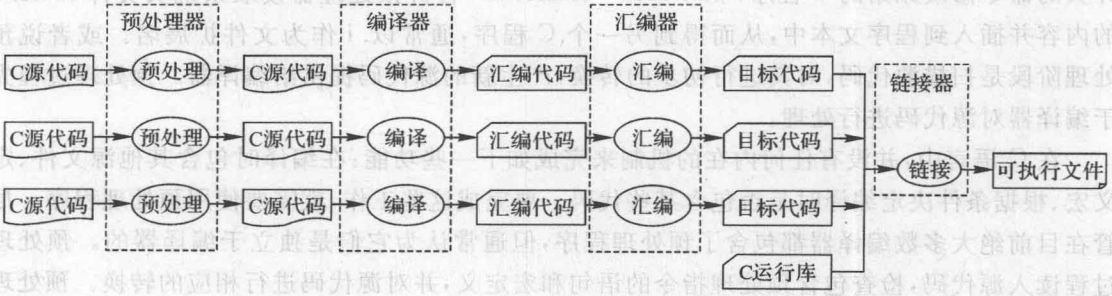


图 1.3　多文件系统的转化过程

1.3　处理器读并解释储存在存储器中的指令

此刻 hello. c 程序源程序已经被编译系统转换成了可执行的目标文件 hello，并被存放在磁盘上，为了在操作系统上运行该可执行文件，用以下步骤：

DOS 等待命令输入并执行，如果命令的第一个单词不是一个内置的 DOS 命令，那么 DOS 就会假设这是一个可执行文件的名字，要加载和执行该文件。

这里 DOS 将加载和执行 hello 程序，然后等待程序终止，hello 程序在屏幕上输出信息，然后终止。DOS 随后输出一个提示符，等待下一个输入的命令。

为了了解运行时 hello 程序发生了什么，需要了解一个典型系统的硬件组织，如图 1.4 所示。

(1)总线。贯穿整个系统的是一组电子管道，称做总线，它携带信息字节并负责在各个部件间传递，通常总线被设计成传送定长的字节块，也就是字。

(2)I/O 设备。I/O 设备是系统与外界的联系通道，例如用户输入的键盘和鼠标、用户输出的显示器、长期存储数据和程序的磁盘。每个 I/O 设备都是通过一个控制器或适配器与 I/O总线连接起来的，它们在 I/O 总线和 I/O 设备之间传递信息。

(3)主存。主存是一个临时存储设备，在处理器执行程序时，它被用来存放程序和程序处理的数据，物理上来说就是 DRAM 芯片，逻辑上来说存储器是由一个线性的字节数组组成的，每个字节都有自己唯一的地址(数组索引)，这些地址从零开始。

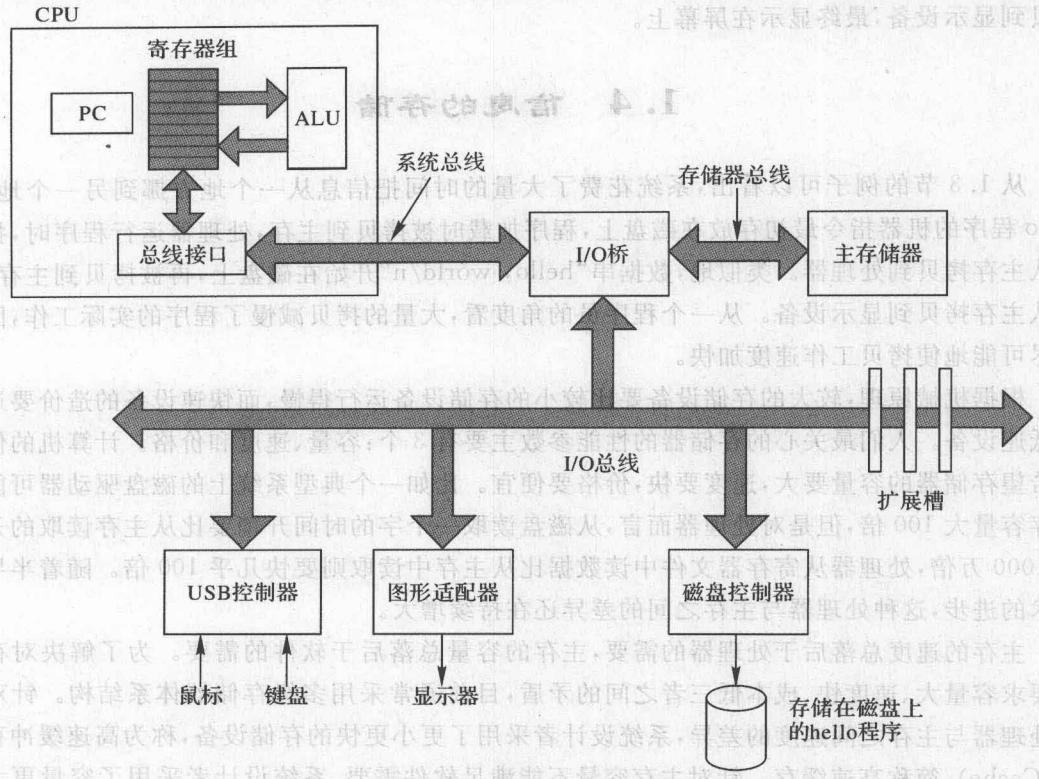


图 1.4　一个典型的计算机硬件系统组成

(4)处理器。中央处理单元(CPU)简称处理器,是解释(或执行)存储在主存中指令的引擎,处理器的核心是一个被称为程序计数器(PC)的字长大小的存储设备(寄存器),任何时间点上,PC 都指向主存中的某条机器语言指令(内含其地址)。从系统通电开始直到系统断电,处理器一直重复执行相同的基本任务:从程序计数器指向的存储器处读取指令,解释指令中的位,执行指令指示的简单操作,然后更新程序计数器指向下一条指令,而这条指令并不一定在存储器中和刚刚执行的指令相邻。

程序执行过程:

DOS 等待输入字符串“./hello.exe”后,逐一读取字符到寄存器,然后放到存储器中。当敲回车键时,DOS 知道已经结束了命令的输入,然后执行一系列指令。

这些指令将 hello 目标文件的代码和数据从磁盘拷贝到主存,从而加载 hello 文件,数据包括最终被输出的字符串“hello, world\n”。

一旦 hello 目标文件中的代码和数据被加载到存储器,处理器就开始执行 hello 程序的主程序中的机器语言指令。

这些指令将“hello, world\n”串中的字节从存储器中拷贝到寄存器文件,再从寄存器文件

拷贝到显示设备，最终显示在屏幕上。

1.4 信息的存储

从 1.3 节的例子可以看出，系统花费了大量的时间把信息从一个地方挪到另一个地方，hello 程序的机器指令最初存放在磁盘上，程序加载时被拷贝到主存，处理器运行程序时，指令又从主存拷贝到处理器。类似地，数据串“hello，world/n”开始在磁盘上，再被拷贝到主存，然后从主存拷贝到显示设备。从一个程序员的角度看，大量的拷贝减慢了程序的实际工作，因此要尽可能地使拷贝工作速度加快。

根据机械原理，较大的存储设备要比较小的存储设备运行得慢，而快速设备的造价要远高于低速设备。人们最关心的存储器的性能参数主要有 3 个：容量、速度和价格。计算机的使用者希望存储器的容量要大，速度要快，价格要便宜。比如一个典型系统上的磁盘驱动器可能比主存容量大 100 倍，但是对处理器而言，从磁盘读取一个字的时间开销要比从主存读取的开销大 1000 万倍，处理器从寄存器文件中读数据比从主存中读取则要快几乎 100 倍。随着半导体技术的进步，这种处理器与主存之间的差异还在持续增大。

主存的速度总落后于处理器的需要，主存的容量总落后于软件的需要。为了解决对存储器要求容量大、速度快、成本低三者之间的矛盾，目前通常采用多级存储器体系结构。针对这种处理器与主存之间速度的差异，系统设计者采用了更小更快的存储设备，称为高速缓冲存储器(Cache)，简称高速缓存。针对主存容量不能满足软件需要，系统设计者采用了容量更大的外存储器，形成多层次的存储体系结构，如图 1.5 所示。

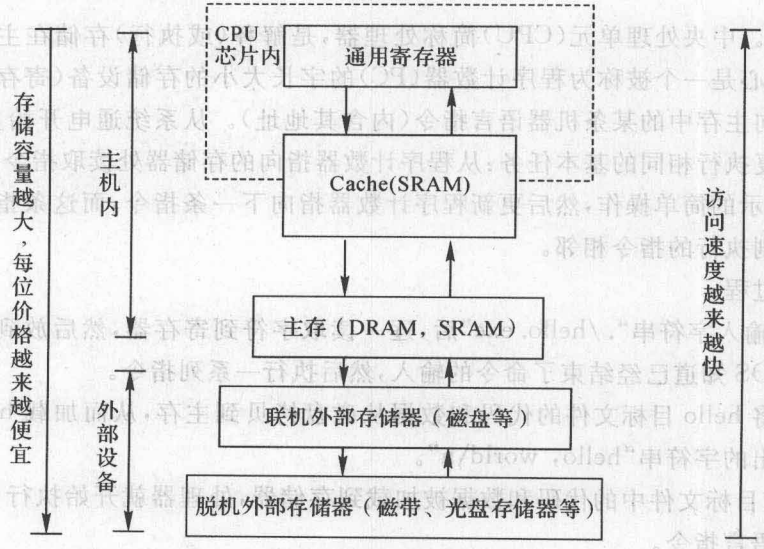


图 1.5 多层次的存储体系结构

位于处理器芯片上的 L1 高速缓存的容量可达数万字节，访问速度几乎和寄存器文件一样快。容量为数十万到数百万的更大的 L2 高速缓存是通过一条特殊的总线连接到处理器的。访问 L2 的时间开销比 L1 大 5 倍，但仍然比访问主存快 5～10 倍。L1 和 L2 高速缓存是用一种叫静态随机访问存储器(SRAM)的硬件技术实现的。

每个计算机的存储设备组织成一个存储器层次模型，从上至下设备变得更慢、更大，并且每字节的造价也更便宜。寄存器文件位于层次模型的最顶部，第 0 级记为 L0，L1 高速缓存为第一层，L2 高速缓存为第二层，主存为 L3，本地磁盘等本地二级存储为 L4，分布式文件系统、Web 服务器等远程二级存储为 L5。层次结构的主要思想是一个层次上的存储器作为下一层次上存储器的高速缓存，寄存器文件是 L1 的高速缓存，L1 是 L2 的高速缓存，L2 是主存的高速缓存，主存是磁盘的高速缓存，本地磁盘是分布式文件系统的高速缓存。

1.5 操作系统的作用

仍然以 hello 程序为例，shell 加载和运行 hello 程序，当 hello 程序输出自己的消息时，程序并没有直接访问键盘、显示器、磁盘或者主存储器，而是依靠操作系统提供的服务。可以把操作系统看做是应用程序和硬件之间插入的一层软件，如图 1.6 所示。所有应用程序对硬件的操作尝试都必须通过操作系统。

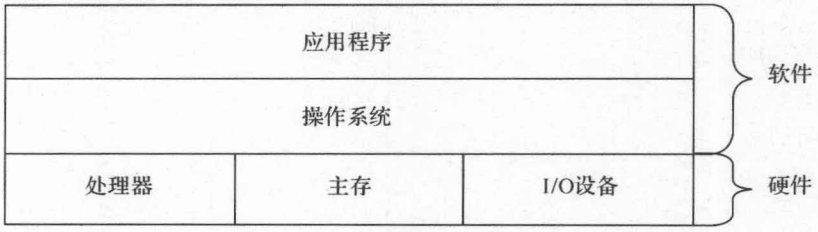


图 1.6 计算机系统的分层视图

操作系统有两个基本功能：防止失控的应用程序滥用硬件；在控制复杂而又各不相同的低级硬件设备访问方面，为应用程序提供简单一致的方法。操作系统通过几个基本的抽象概念(进程、虚拟存储器和文件)来实现这两个功能。文件是对 I/O 设备的抽象表示，虚拟存储器是对主存和磁盘 I/O 设备的抽象表示，进程则是对处理器、主存和 I/O 设备的抽象表示。

进程是操作系统对运行程序的一种抽象，操作系统保存进程运行所需的所有状态信息，这种状态就是上下文(context)，包括许多信息，比如 PC 和寄存器文件的当前值，以及主存的内容。在一个系统上可以同时运行多个进程，而每个进程都好像在独占地使用硬件，我们称之为并发运行，实际上是指一个进程的指令和另一个进程的指令是交错执行的。操作系统实现这种交错执行的机制称为上下文切换(context switching)。

在任何一个时刻，系统都只有一个进程正在运行。当操作系统决定从当前进程转移控制

权到某个新进程时，它就会进行上下文切换，即保存当前进程的上下文、恢复新进程的上下文，然后将控制权转移到新进程，新进程就会从它上次停止的地方开始运行。

进程这个抽象概念还暗示着由于不同的进程交错执行，打乱了时间的概念，使得程序员很难获得运行时间的准确和可重复度量。

尽管通常我们认为一个进程只有单一的控制流，但是在现代系统中，一个进程实际上可以由多个称为线程的执行单元组成，每个线程都运行在进程的上下文中，并共享同样的代码和全局数据。由于网络服务器中对并行处理的要求，线程成为越来越重要的编程模型，因为多线程之间比多进程之间更容易共享数据，也因为线程一般都比进程更高效。