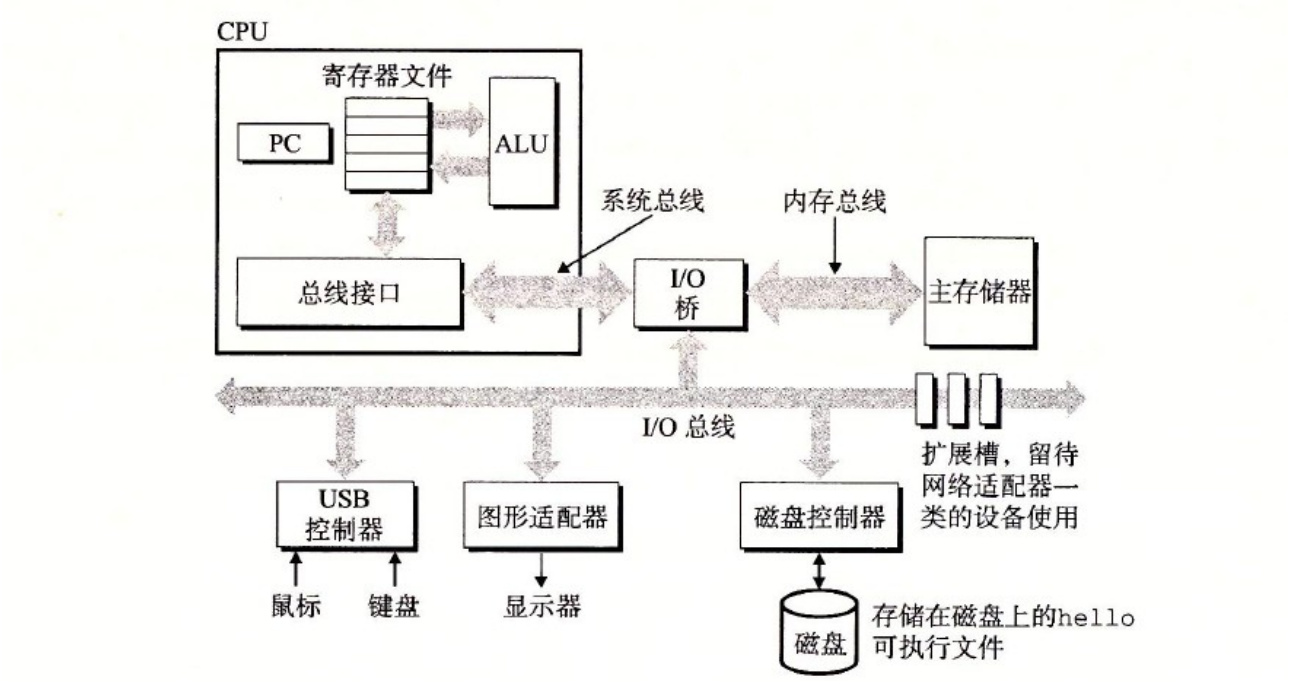
hello 程序的生命周期是从一个源程序开始的，通过编辑器创建并保存的文本文件，hello.c
实际是用一个唯一的单字节大小的整数值来表示每个字符。（ASCII)

系统中的所有信息——包括磁盘文件、内存中的程序、内存中存放的用户数据以及网络上传送的数据，都是一串比特表示的。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。

gcc 编译器分为四个阶段：预处理阶段：hello.c → hello.i (修改了的源程序）
编译阶段： hello.i → hello.s（将文件转为汇编程序）
汇编阶段： hello.s → hello.o (可重定位目标程序）
链接阶段：hello.o\printf.o …. → hello(可执行目标程序）

1.4.1 系统硬件组成



1、总线

贯穿整个系统的是一组电子管道，称作总线，它携带信息字节并负责各个部件间的传递。传送的是定长的字节块，也就是字。字的字节数要么是 4 个字节，要么是 8 个字节。

2、I/O 设备

I/O 设备是系统与外部世界的联系通道。每个 I/O 设备都通过一个控制器或适配器与 I/O 总线相连。（控制器和适配器的区别在于封装方式： 控制器是主板上的芯片组，而适配器则是插在主板插槽的卡。

3、主存

主存是一个临时存储设备，在处理器执行程序时，用来存放程序和程序处理的数据。从物理上来说，主存是一组动态随机存取存储器（DRAM）；从逻辑上来说，存储器是一个线性的字节数组，每一个字节都有唯一地址。

4、处理器

中央处理单元（CPU），解释存储在主存中指令的引擎。处理核心是一个大小为一个字的存储设备，称为程序计数器（PC）。
PC 指向主存中的某条机器语言。
它们围绕这主存、寄存器文件、算术逻辑单元 ALU 进行。寄存器文件是一个小的存储设备，由一些单字长的寄存器组成。ALU 计算新的地址值和数据。CPU 可能执行的一些操作：

- 加载： 从主存复制一个字节或者一个字到寄存器，以覆盖寄存器原来的内容。
- 存储：从寄存器复制一个字节或者一个字到主存的某个位置，以覆盖这个位置原来的内容。

- 操作：把两个寄存器的内容复制到 ALU，ALU 对这两个字做算术运算，并将结果存放到一个寄存器中。
- 跳转：从指令本身中抽取一个字，并将这个字复制到 PC 中。

1.5 高速缓存至关重要

简单的 hello 程序揭示了一个重要的问题，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。

针对处理器与主存之间的差异，系统设计者采用更小更快的存储设备，称为高速缓存存储器（简称 cache 或高速缓存）。硬件上由静态随机访问存储器实现。

1.6 存储设备形成层次结构

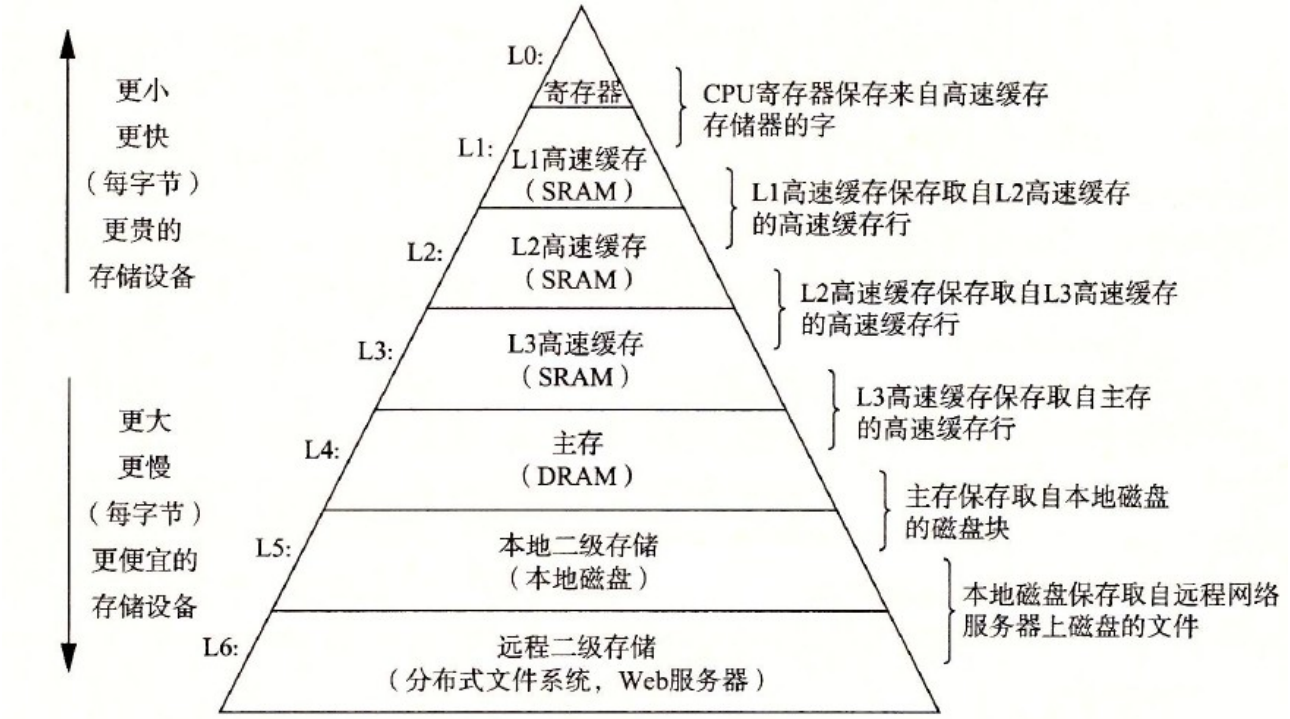


图 1-9　一个存储器层次结构的示例

在处理器和一个较大较慢的设备（例如主存）之间插入一个更小更快的存储设备，已经成为一个普遍的观念。每一个操作系统的存储设备都组织成一个存储器层次结构。这个层次结构从上至下，设备的访问速度越来越慢、容量越来越大。

存储起层次结构的主要思想是上一层的存储起作为低一层存储器的高速缓存。

1.7 操作系统管理硬件

shell 程序和 hello 都没有直接访问键盘、显示器、磁盘或主存。依靠的是操作系统提供的服务。操作系统可以看成是应用程序和硬件之间插入的一层软件。

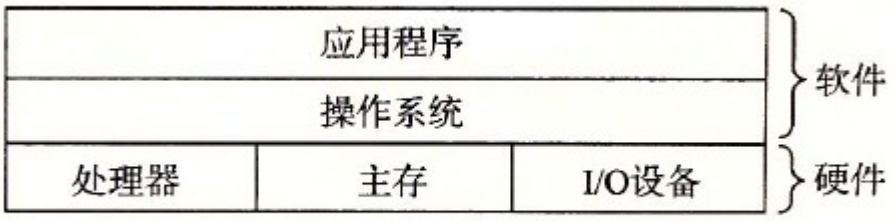


图 1-10　计算机系统的分层视图

操作系统有两个基本功能：（1）防止硬件被失控的应用程序滥用；（2）向应用程序提供简单一致的机制来控制复杂而又通常大不相同的低级硬件设备。

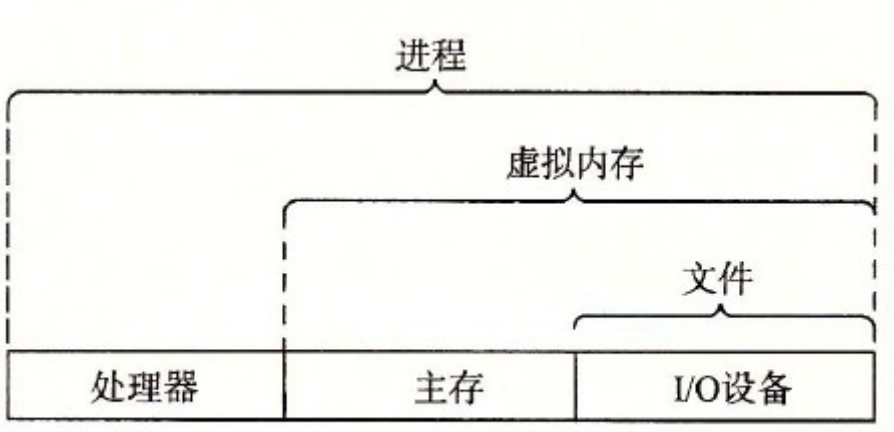


图 1-11　操作系统提供的抽象表示

1.7.1 进程

1. 操作系统会提供一种假象，好像系统上只有这个程序独占使用处理器、主存和 I/O 设备。这些假象是通过进程的概念来实现的。
2. 进程是操作系统对于一个正在运行的程序的一种抽象。
3. 在一个系统上可以同时运行多个进程，而每个进程好像独占使用硬件。并发运行，是说一个进程的指令和另一个进程的指令交错执行。这是通过处理器在进程间的切换来实现的，操作系统实现这种交错执行的机制称为上下文切换。
4. 操作系统保持跟踪进程运行所需的所有状态信息。这种状态，也就是上下文（包括各种资源：PC、寄存器文件、ALU）。
5. 从一个进程到另一个进程的转换是由操作系统内核管理的。内核是操作系统代码常驻主存的部分。内核是系统管理全部进程所用代码和数据结构的集合。

1.7.2 线程

进程只有单一的控制流，一个进程由多个线程的执行单元组成，每个线程都运行在进程的上下文。

1.7.3 虚拟内存

虚拟内存是一个抽象概念，为每个进程提供一个假象，即每个进程独占使用内存。每个进程看到的内存都是一致的，称为虚拟地址空间。

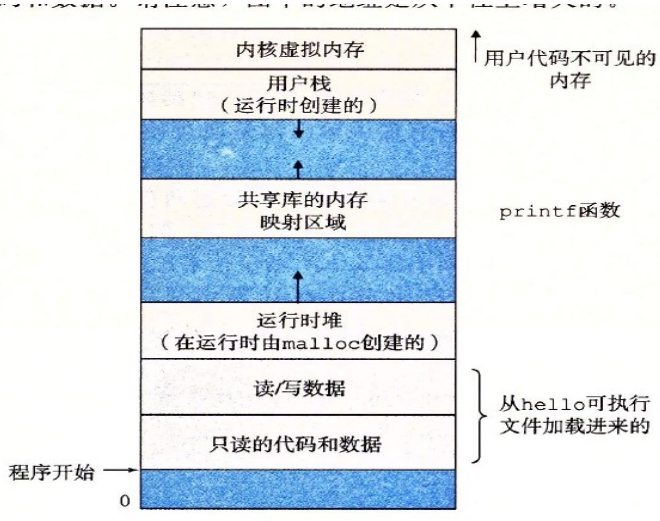


图 1-13　进程的虚拟地址空间

在 linux 中，地址空间最上面的区域是保留给操作系统的代码和数据的。每个虚拟地址空间由大量准确定义的区域构成：

- 程序代码和数据。代码和数据去是直接按照可执行目标文件的内容初始化。
- 堆。当调用 malloc 和 free 这样的动态函数。
- 共享库。 大约在地址空间的中间部分。
- 栈。编译器用它实现函数调用。
- 内核虚拟内存。不允许应用程序读写这个区域或直接调用内核代码定义的函数。

1.7.4 文件

文件就是字节序列，仅此而已。每个 i/o 设备，包括磁盘、键盘、显示器，甚至网络，都可以看成文件。

1.9 并发和并行

并发是一个通用概念，一个同时就多个活动的系统。而并行指用并发来使一个系统运行的更快。

1. 线程级并发

传统意义上，这种并发执行只是模拟出来的，是通过使一台计算机在它正在执行的进程间快速切换来实现的。

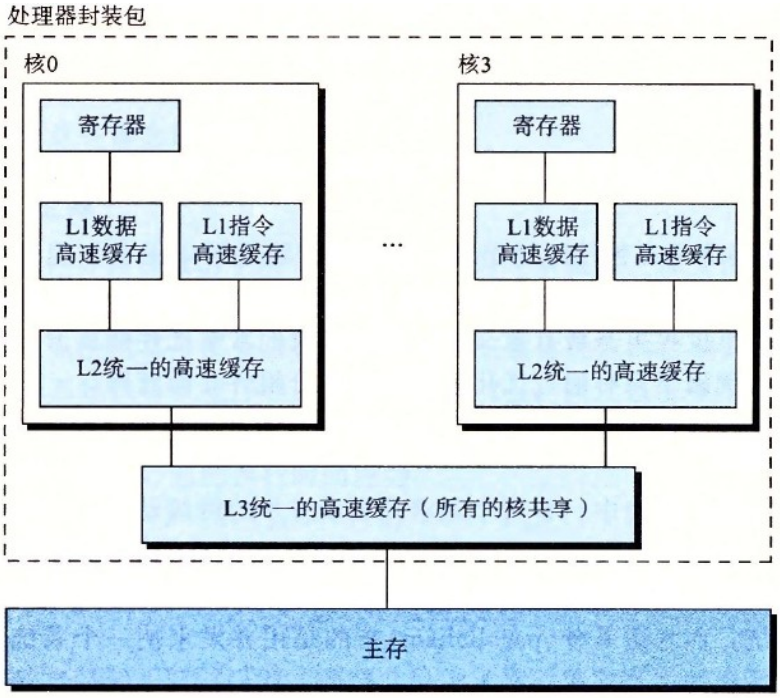


图 1-17　多核处理器的组织结构。4 个处理器核集成在一个芯片上

2. 指令级并行

现代处理器可以同时执行多条指令的属性称为指令级并行。

3. 单指令、多数据并行

允许一条指令产生多个可以并行执行的操作。

1.9.2 计算机系统抽象的重要性

抽象是计算机科学中最为重要的概念之一。

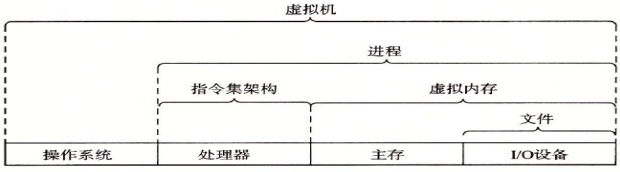


图 1-18　计算机系统提供的一些抽象。计算机系统中的一个重大主题就是提供不同层次的抽象表示，来隐藏实际实现的复杂性