深入理解计算机系统—计算机系统漫游

|  |  |
| --- | --- |
|  | 🢂 内容概览 |
|  | Why：此文档用来做什么？它存在的意义是什么？为解决什么问题？   |  | | --- | |  |   What：当前包含了那些内容？   |  | | --- | |  |   How：此文档应如何参考？   |  | | --- | |  |   Who：此文档适用于那些人员阅读参考？   |  | | --- | |  | |

目录

[1 信息=位+上下文 4](#_Toc448153561)

[2 编译系统的组成&编译过程 4](#_Toc448153562)

[3 shell 4](#_Toc448153563)

[4 系统的硬件组成 4](#_Toc448153564)

[4.1 总线 5](#_Toc448153565)

[4.2 I/O设备 5](#_Toc448153566)

[4.3 主存 5](#_Toc448153567)

[4.4 处理器 5](#_Toc448153568)

[4.4.1 Cpu在指令要求下可能的操作 6](#_Toc448153569)

[4.4.2 cpu指令集结构&微体系结构 6](#_Toc448153570)

[5 程序执行过程 6](#_Toc448153571)

[5.1 读取键盘输入 6](#_Toc448153572)

[5.2 加载程序 7](#_Toc448153573)

[5.3 执行程序&输出 8](#_Toc448153574)

[6 高速缓存 8](#_Toc448153575)

[6.1 背景 9](#_Toc448153576)

[6.2 高速缓存 9](#_Toc448153577)

[6.3 存储设备层次结构 9](#_Toc448153578)

[7 操作系统 10](#_Toc448153579)

[7.1 基本功能 10](#_Toc448153580)

[7.2 抽象概念 10](#_Toc448153581)

[7.2.1 进程 11](#_Toc448153582)

[7.2.2 线程 11](#_Toc448153583)

[7.2.3 虚拟存储器 12](#_Toc448153584)

[7.2.4 文件 13](#_Toc448153585)

[7.2.5 并发&并行 13](#_Toc448153586)

[7.3 cpu概念 13](#_Toc448153587)

[7.3.1 多核处理器 14](#_Toc448153588)

[7.3.2 超线程 14](#_Toc448153589)

[7.3.3 i7处理器组织结构 14](#_Toc448153590)

[8 抽象的重要性 15](#_Toc448153591)

# 信息=位+上下文

1. 文本文件&二进制文件：只由ASCII构成的文件称为文本文件，其它非文本文件称为二进制文件；
2. 信息表示的基本思想：系统中所有的信息（包括：磁盘文件、存储器中的程序和用户数据、网络上传送的数据等）都是由一串位表示的。区分不同数据对象的唯一方法是这些数据对象的上下文；

# 编译系统的组成&编译过程

|  |
| --- |
| 编译系统 |

1. 预处理阶段：解析源代码中的预处理命令（以#开头的命令，如#define），插入替换文本；作用类似于bash中的source命令、java中的import命令；
2. 编译阶段：将修改后的源代码翻译成汇编语言代码；
3. 汇编阶段：将汇编语言代码生成目标程序；
4. 链接阶段：将目标文件链接，生成可执行目标文件；
5. 执行阶段：可执行目标文件加载到内存后，执行；

# shell

在unix中执行可执行文件，需要将文件输入到shell中。shell是一个命令行解释器，用来执行输入命令并给出输出结果；如果输入字符不是shell命令，则默认视作可执行程序，执行加载和运行；

# 系统的硬件组成

|  |
| --- |
|  |

## 总线

1. 贯穿整个系统的电子管道；
2. 携带信息字节并负责在各个部件间传递；
3. 字：通常总线被设计为传送定长的字节块，一个字节块称为一个字；
4. 字长：一个字包含的字节数称为字长；

## I/O设备

1. 系统与外部世界的连接通道；
2. 每个I/O设备都通过一个控制器/适配器与I/O总线相连；控制器/适配器之间的区别主要在于其封装方式；
3. 控制器是集成主板上的芯片；而适配器是主板上可插拔的卡；

## 主存

1. 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；
2. 物理上，主存是由一组动态随机存取存储器芯片（DRAM）组成的；
3. 逻辑上，主存是一个线性的字节数组，每个字节都有其唯一的地址（数组索引），地址从0开始；
4. 组成程序的每条机器指令都由不同数量的字节组成；

## 处理器

1. 解释执行存储器中指令的引擎；
2. 程序计数器：存放一个字长的存储设备，用来存放将要执行的指令地址；
3. 寄存器文件：存放少量数据的存储设备，由一些1个字长的寄存器组成；每个寄存器都有唯一的名字；
4. ALU：计算新的数据和地址值；

### Cpu在指令要求下可能的操作

1. 加载：把一个字节或者一个字复制到寄存器，覆盖寄存器原来的内容；
2. 存储：把一个字节或者一个字从寄存器复制到主存的某个位置，覆盖该位置原来的内容；
3. 操作：把两个寄存器的内容复制到ALU；ALU对这两个字做算术/逻辑操作，并将结果复制到一个寄存器中覆盖原来的内容；
4. 跳转：从指令本省抽取1个字，并将这个字复制到程序计数器（PC）中，覆盖PC中原来的值；

### cpu指令集结构&微体系结构

1. 指令集结构：描述每条机器代码指令的效果；
2. 微体系结构：描述处理器是如何实现的；

# 程序执行过程

## 读取键盘输入

当我们在键盘上输入字符串”./hello”后，shell将逐一读入寄存器，放入存储器；

|  |
| --- |
| 从键盘上读取hello命令 |

## 加载程序

当敲击回车时，视作输入结束，shell执行一系列指令，将hello程序加载至存储器；

|  |
| --- |
| 从磁盘加载可执行文件到主存 |

## 执行程序&输出

|  |
| --- |
| 将输出字符串从内存输出到显示器 |

# 高速缓存

## 背景

1. 从程序执行的过程，揭示出一个重要问题：系统花费大量的时间把信息从一个地方复制到另一个地方，这些开销减缓了真正的工作，故需要尽可能使其快速完成；
2. 根据机械原理，较大的存储设备要比较小的存储设备运行的慢，而快速的设备造价远高于同类的低速设备；
3. 处理器从磁盘驱动器读取一个字的数据，开销要比从主存中读取开销大1000万倍；
4. 处理器从寄存器文件中读取数据速率要比从主存中读取快近100倍；且差距还在加大，因为加快cpu速率要比加快主存速率更加容易和便宜；

## 高速缓存

|  |
| --- |
|  |

1. L1缓存：位于处理器芯片上，容量数万字节，速率和寄存器速率基本相同；
2. L2缓存：通过一条特殊总线连接到处理器，容量数百万字节，比L1慢5倍，比主存快5~10倍；
3. L1和L2使用SRAM（静态随机访问存储器）的硬件技术实现；
4. 高速缓存的局部性原理：通过让高速缓存里存放经常访问的数据的方法，大部分存储器操作都能够快速在高速缓存中完成；

|  |
| --- |
| 结论：利用高速缓存可以将程序的性能提高一个数量级； |

## 存储设备层次结构

|  |
| --- |
| 存储器层次结构 |

1. 每个计算机系统中的存储设备都被组织成了一个存储器层次结构；
2. 从上至下，设备访问速度减慢，造价也越来越便宜；
3. 存储器层次结构的主要思想：位于下层的存储设备将相邻的上一层存储器视作其高速缓存；

# 操作系统

## 基本功能

|  |
| --- |
| 计算机系统分层视图 |

1. 操作系统可以看做应用程序和硬件之间插入的一层软件；
2. 操作系统具有两个基本功能：
   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大相径庭的低级硬件设备；

## 抽象概念

|  |
| --- |
| 操作系统提供的抽象表示 |

1. 操作系统通过几个基本的抽象概念，如：进程、虚拟存储器、文件等，实现上述基本功能；
2. 文件：对I/O设备的抽象表示；
3. 虚拟存储器：对主存和磁盘I/O设备的抽象表示；
4. 进程：对处理器、主存和I/O设备的抽象表示；

### 进程

1. 进程是操作系统对一个正在运行的程序的一种抽象；
2. 在一个系统上可以运行多个进程，每个进程都好像在独占的使用硬件；
3. 并发运行：一个进程的指令和另一个进程的指令交错执行；
4. 单核cpu在一个时刻只能执行一个程序，而多核cpu能够在同一时刻执行多个程序；
5. 上下文：操作系统保持跟踪进程运行所需的所有状态信息；包括：pc和寄存器文件当前值、主存内容等；
6. 上下文切换：无论单核还是多核系统中，一个cpu看上去都像是在并发的执行多个进程，这是通过处理器在进程间的切换来实现的；这种交错执行的机制称为上下文切换；当操作系统决定把控制权从当前进行切换到其它进程时，就会进行上下文切换；

|  |
| --- |
| 上下文切换示意 |

### 线程

1. 一个进程可以由多个称为线程的执行单元组成；
2. 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；
3. 线程相比进程更容器进行数据共享，线程的上下文切换消耗更小，且线程能够更好的利用多核进行运算，故线程比进程更加高效；

### 虚拟存储器

|  |
| --- |
| 进程的虚拟地址空间 |

1. 虚拟存储器是一个抽象概念，它为每个进程提供了一个假象，即每个进程都在独占的使用主存；
2. 每个进程看到的是一致的存储器，称为虚拟地址空间；虚拟地址空间自下而上由如下几个部分组成：
   1. 代码和数据区：存放程序代码和全局数据，对于所有进程而言，代码是从同一个固定地址开始的，代码和数据区直接按照可执行目标文件的内容进行初始化；紧接着是和C全局数据相对应的数据位置；在运行之前就能够确定大虾；
   2. 堆：代码和数据区之后是运行时堆。堆的大小能够在运行时动态伸缩，使用free和malloc函数；
   3. 共享库：在地址空间的中部，用来存放C标准库、数学库等共享库的代码和数据；
   4. 栈：位于虚拟地址空间顶部，编译器用它来存放局部变量，实现函数调用；能够在运行时动态伸缩；
   5. 内核虚拟存储器：内核总是驻留在内存中，是操作系统的一部分；地址空间的顶部区域是为内核保留的，不允许应用程序读写这个区域的内容或者直接调用内核代码定义的函数；

### 文件

1. 文件就是一个字节序列；每个I/O设备，包括：磁盘、键盘、显示器、网络都是视作文件；
2. 系统中的所有的输入输出都是通过使用一小组称为Unix I/O的系统函数调用读写文件来实现的；
3. 网络也是一种I/O设备；

### 并发&并行

1. 并发：做得更多；并发不一定并行；一个同时具有多个活动的系统；
2. 并行：做得更快；
3. 并发可以分为如下类型：
   1. 线程级并发：层次较高，一个系统能够执行多个进程，一个进程可以划分为多个线程执行；
   2. 指令级并行：层次较低，处理器同一时刻能够执行多条指令；使用流水线的思路，将指令执行划分为不同的步骤，将cpu的硬件执行划分为多个阶段。不同的步骤放到不同的阶段上执行，从整体上看，同一时间点，能够处理的指令可能有多条，从而提高指令执行效率；
   3. 单指令、多数据并行（SIMD, Single Instruction Multiple Data）：层次最低，一条指令能够产生多个可以并行执行的操作；

## cpu概念

|  |
| --- |
| cpu种类划分 |

### 多核处理器

多个cpu集成到一个集成电路芯片上，每个核都有自己的L1和L2高速缓存；多个核之间共享更高层次的高速缓存以及内存接口；

### 超线程

同时多线程，允许一个cpu执行多个控制流测技术。它涉及cpu某些硬件有多个备份；比如：程序计数器文和寄存器文件；而其余硬件部分只有一份；

### i7处理器组织结构

|  |
| --- |
|  |

# 抽象的重要性

|  |
| --- |
|  |

1. 万事万物皆抽象；抽象抽取共性，隐藏实现的复杂性；
2. 在cpu中，指令集结构提供了对实际处理器硬件的抽象；
3. 文件是对I/O的抽象；
4. 虚拟存储器是对程序存储器的抽象；
5. 进程是对正在运行程序的抽象；
6. 虚拟机是对整个计算机的抽象；