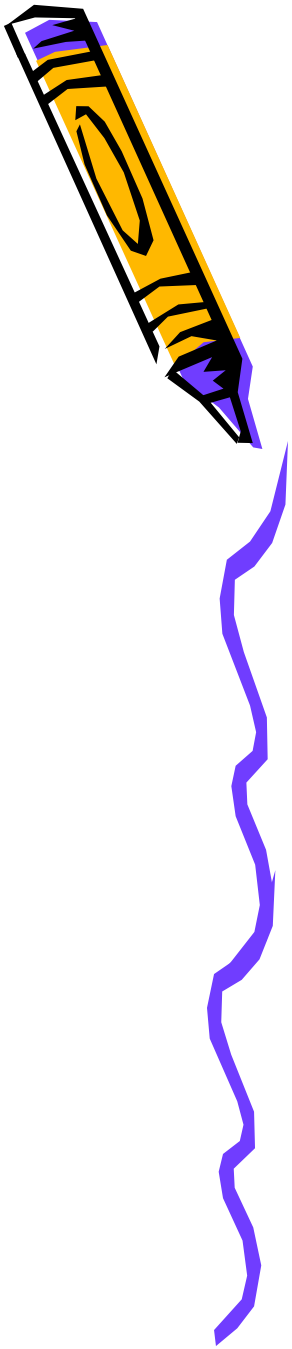
《Java程序设计实用教程》

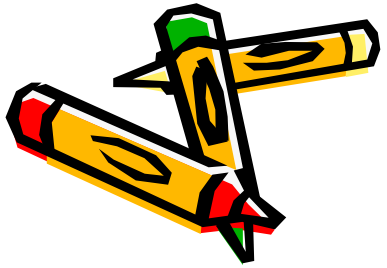
《Java程序设计实用教程》

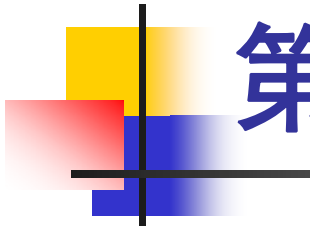
- 第1章 Java概述
- 第2章 Java语言基础
- 第3章 类的封装、继承和多态
- 第4章 接口、内部类和Java API基础
- 第5章 异常处理
- 第6章 图形用户界面
- 第7章 多线程
- 第8章 输入/输出流和文件操作
- 第9章 网络通信
- 第10章 数据库应用
- 第11章 Web应用
- 第12章 综合应用设计



第1章 Java概述

- 1.1 了解Java
- 1.2 JDK
- 1.3 Eclipse





第1章 Java概述

内容和要求：

① 了解**Java**语言特点；

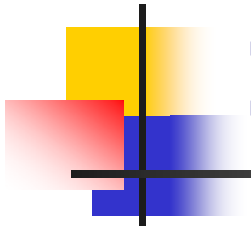
② 理解**Application**应用程序的运行原理和方法；

③ 理解由**Java**虚拟机支持的程序运行机制；

④ 包，导入包；

⑤ 掌握在**JDK**和**Eclipse**环境中编译、运行和调试程序的操作。

- 难点：包，**Eclipse**的工作区和项目。



1.1 了解Java

- 1.1.1 Java的诞生和发展
- 1.1.2 Java的特点
- 1.1.3 Java的核心技术
- 1.1.4 Application应用

计算机语言发展史

计算机语言发展历程：逐渐趋向人类能理解的语言

1. 机器语言：即计算机能理解的语言。由**0**和**1**代码组成。
2. 汇编语言：使用英文缩写的助记符来表示基本计算机操作，主要通过操作指令来进行对计算机的操作。
3. 高级语言：面向过程**(C、Fortran、COBOL、PASCAL、ADA)**、面向对象**(C++、Java、C#、Python、PHP、Object-C、Swift…)**

在显示器上输出“welcome to masm”，机器码如下：

```
00011110
10111000000000000000000000
01010000
10111000110001100000111 1
10001110111011000
10110100000000110
10110000000000000
10110111000000111
10111001000000000000000000
10110110000011000
10110010010011111
11001101000100000
10110101000000010
10110111000000000
10110110000000000
10110010000000000
11001101000100000
10110100000001001
10001101000101100010101000000000
11001101001000001
10110100000001010
10001101000101100011000100000000
11001101001000001
10110100000000110
10110000000010100
10110111000110001
10110101000010111
10110001000100111
10110110000011101
10110010001111100
11001101000100000
11010100000000010
11010111000000000
11010010000001100
11010010000101000
11001101000100000
10110100000001001
10001101000101100000000000000000
11001101001000001
11001011
```

```
C_S   SEGMENT
      ASSUME   CS: C_S, DS: C_S
S_T:
      MOV   AX, C_S
      MOV   DS, AX
      LEA   DX, P_S
      MOV   AH, 9
      INT   21H
      MOV   AH, 4CH
      INT   21H
P_S   DB    'Hello World!', 36
C_S   ENDS
      END   S_T
```

计算机系统简述

理解底层计算机系统中的硬件和系统软件如何协作来实现应用程序的运行；认识计算机体系结构、操作系统。

code/intro/hello.c

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

code/intro/hello.c

图 1-1　hello 程序

目标：理解当在计算机系统中执行hello程序时，系统发生了什么以及为什么会这样。

计算机系统简述

hello 程序的生命周期从一个源文件开始，即程序员利用编辑器创建并保存的文本文件，文件名是hello.c。源程序实际上就是一个由值0 和1 组成的位（bit）序列，8 个位被组织成一组，称为字节。每个字节表示程序中某个文本字符。大部分的现代系统都使用ASCII 标准来表示文本字符，这种方式实际上就是用一个唯一的单字节大小的整数值来表示每个字符。例如图1-2 中给出了hello.c 程序的ASCII 码表示。

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

图 1-2　hello.c 的 ASCII 文本表示

计算机系统简述

hello 程序的生命周期是从一个高级C 语言程序开始的，因为这种形式能够被人读懂。然而，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的格式打好包，并以二进制磁盘文件的形式存放起来。在Unix 系统上，从源文件到目标文件的转化是由编译器驱动程序完成的：

```
unix> gcc -o hello hello.c
```

这个转化的过程可分为四个阶段完成，如图1-3 所示。执行这四个阶段的程序（预处理器、编译器、汇编器和链接器）一起构成了编译系统。

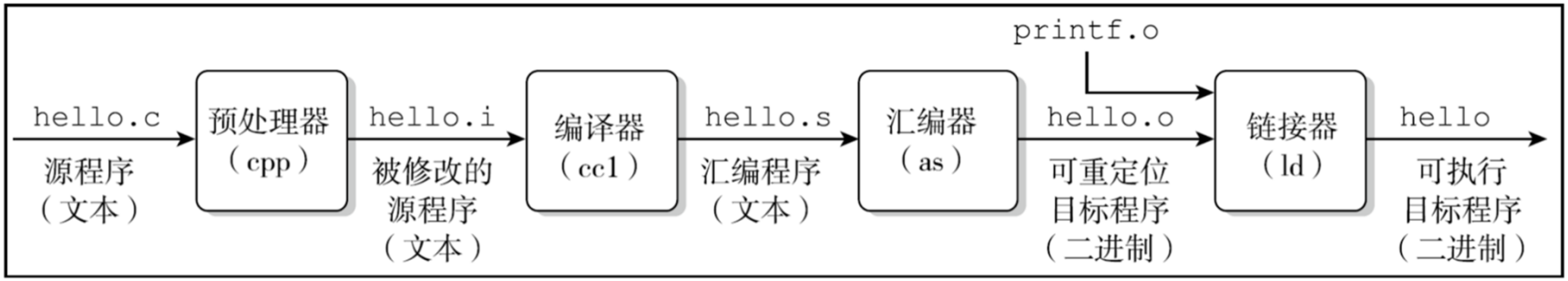
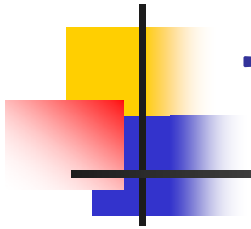


图 1-3 编译系统



计算机系统简述

hello.c 源程序被编译系统翻译成了可执行目标文件hello，并存放在磁盘上。要想在Unix 系统上运行该可执行文件，我们将它的文件名输入到称为外壳（shell）的应用程序中：

```
unix> ./hello
hello, world
unix>
```

在此例中，外壳将加载并运行hello 程序，然后等待程序终止。hello 程序在屏幕上输出它的信息，然后终止。外壳随后输出一个提示符，等待下一个输入的命令行。

计算机系统简述

系统的硬件组成：1、**总线**（贯穿整个系统的是一组电子管道，携带信息字节并负责在各个部件间传递。）；2、**I/O 设备**（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）；3、**主存**（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，主存是由一组动态随机存取存储器DRAM芯片组成的。从逻辑上来说，存储器是一个线性的字节数组。）；4、**处理器**（解释执行存储在主存中指令的引擎。包括程序计数器PC、寄存器文件和算术/ 逻辑单元ALU）。

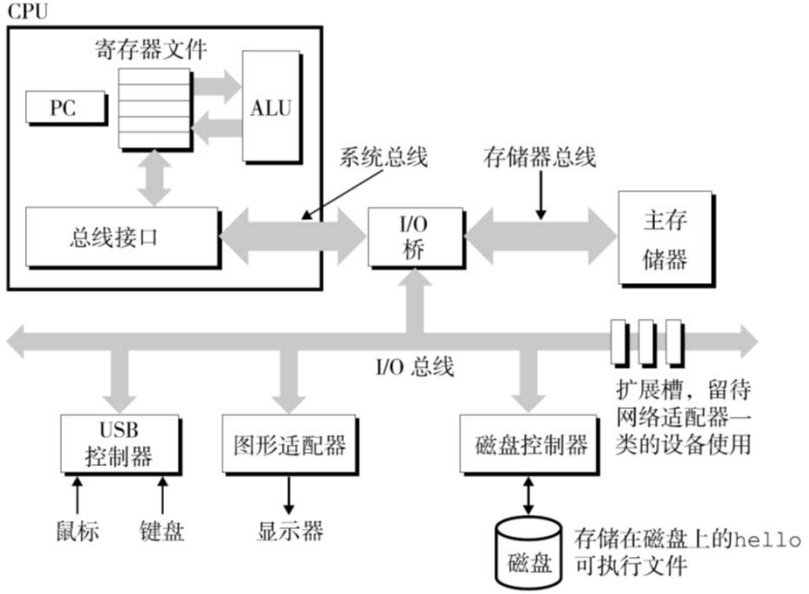


图 1-4　一个典型系统的硬件组成

计算机系统简述

初始时，外壳程序执行它的指令，等待我们输入一个命令。当我们在键盘上输入字符串“./hello”后，外壳程序将字符逐一读入寄存器，再把它存放到存储器中。

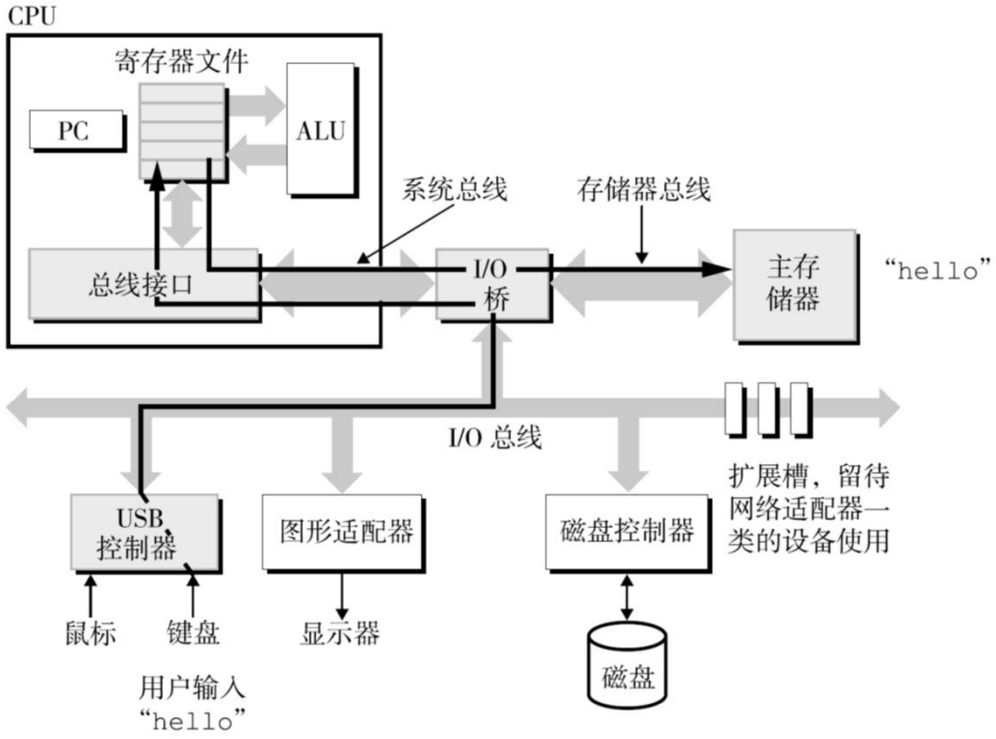


图 1-5　从键盘上读取 hello 命令

计算机系统简述

当我们在键盘上敲回车键时，外壳程序就知道我们已经结束了命令的输入。然后外壳执行一系列指令来加载可执行的hello 文件，将hello 目标文件中的代码和数据从磁盘复制到主存。数据包括最终会被输出的字符串“hello, world\n”。

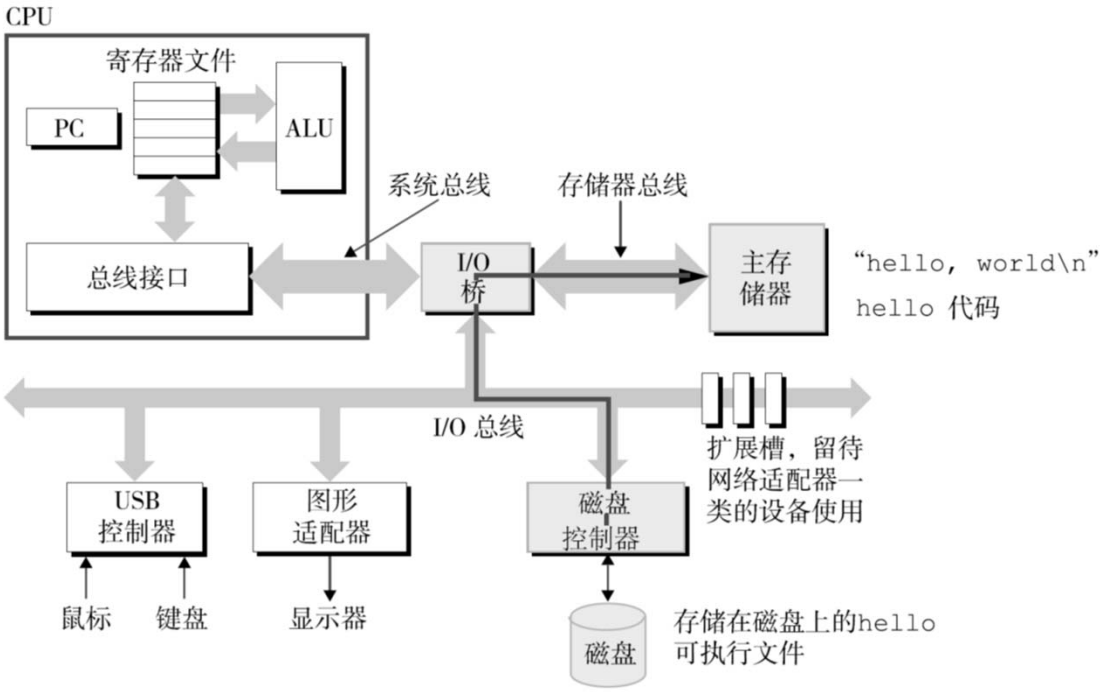


图 1-6　从磁盘加载可执行文件到主存

计算机系统简述

一旦目标文件hello 中的代码和数据被加载到主存，处理器就开始执行hello 程序的main 程序中的机器语言指令。这些指令将“hello, world\n”字符串中的字节从主存复制到寄存器文件，再从寄存器文件中复制到显示设备，最终显示在屏幕上。

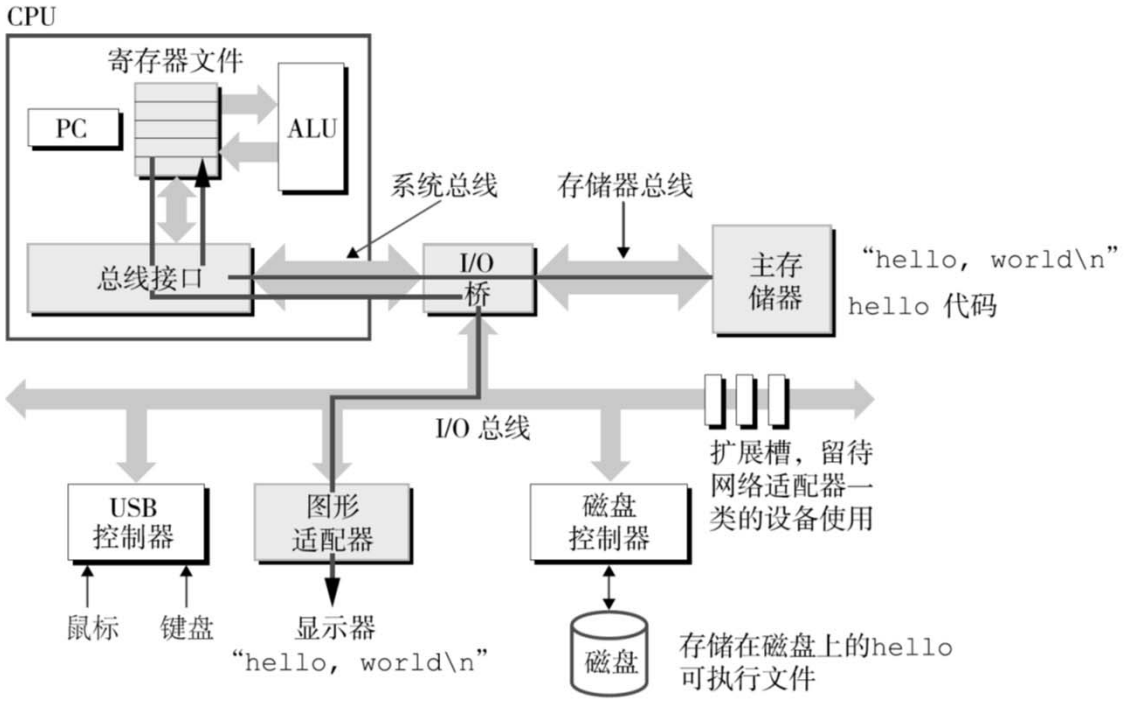


图 1-7　将输出字符串从内存写到显示器

计算机系统简述

当外壳加载和运行hello 程序，以及hello 程序输出自己的消息时，外壳和hello 程序都没有直接访问键盘、显示器、磁盘或者主存。取而代之的是，它们依靠操作系统提供的服务。可以把操作系统看成是应用程序和硬件之间插入的一层软件，所有应用程序对硬件的操作尝试都必须通过操作系统。

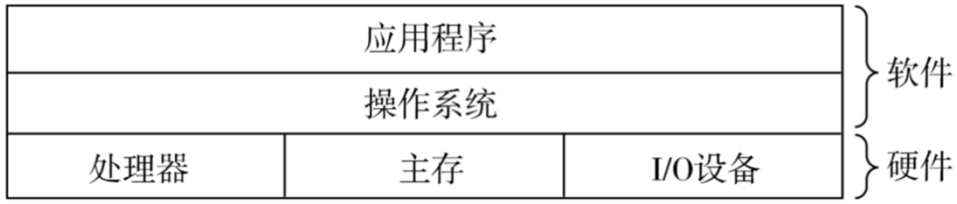


图 1-10　计算机系统的分层视图

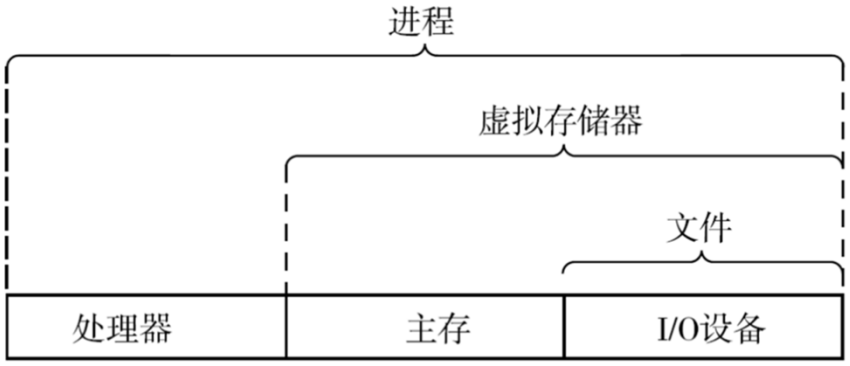


图 1-11　操作系统提供的抽象表示

计算机系统简述

进程是操作系统对一个正在运行的程序的一种抽象。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。而并发运行，则是说一个进程的指令和另一个进程的指令是交错执行的。操作系统保持跟踪进程运行所需的所有状态信息。在任何一个时刻，单处理器系统都只能执行一个进程的代码。当操作系统决定要把控制权从当前进程转移到某个新进程时，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，然后将控制权传递到新进程。

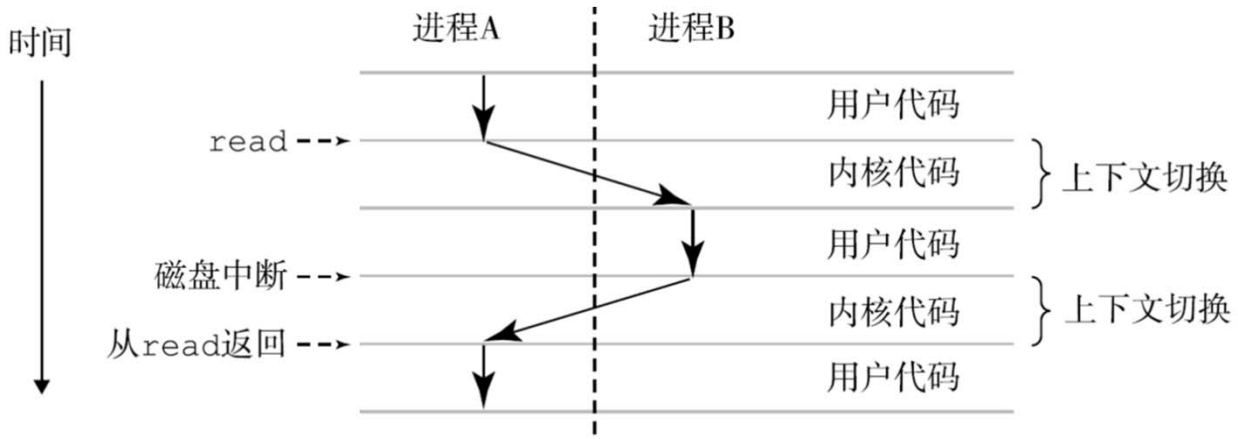
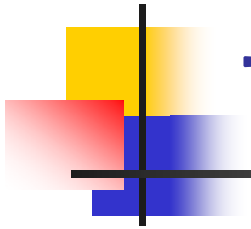


图 1-12　进程的上下文切换



计算机系统简述

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。由于网络服务器对并行处理的需求，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，也因为线程一般来说都比进程更高效。当有多处理器可用的时候，多线程也是一种使程序可以更快运行的方法。

计算机系统简述

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地使用主存。每个进程看到的是一致的存储器，称为虚拟地址空间。图1-13 所示的是Linux进程的虚拟地址空间。在Linux 中，地址空间最上面的区域是为操作系统中的代码和数据保留的，这对所有进程来说都是一样的。地址空间的底部区域存放用户进程定义的代码和数据。

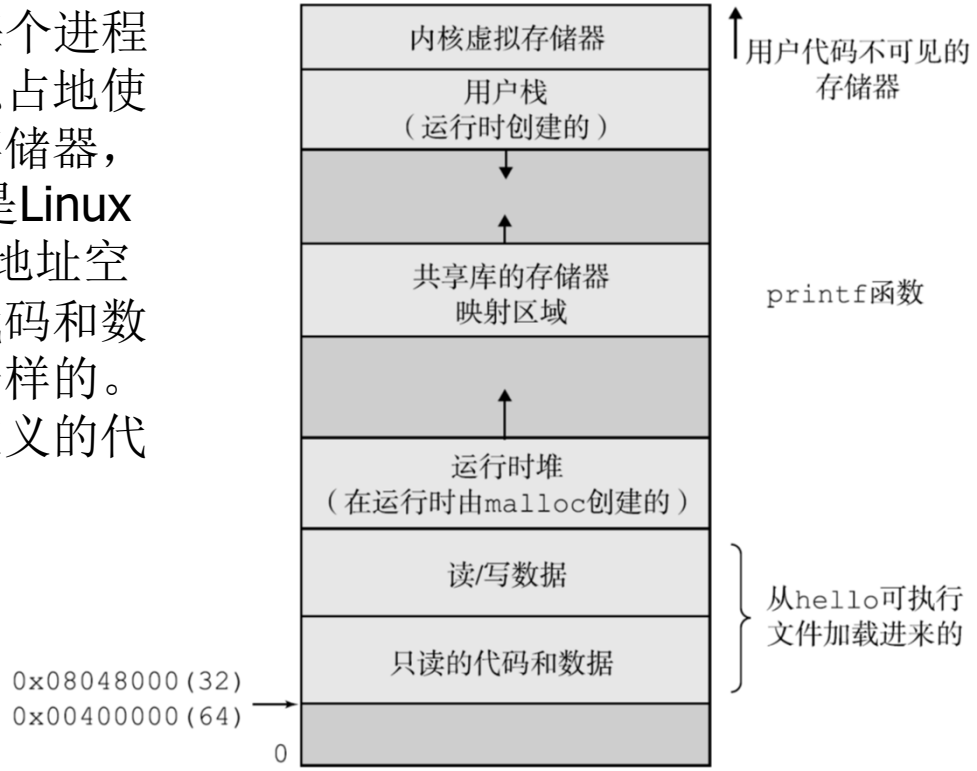
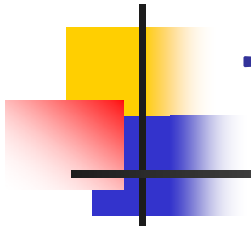


图 1-13　进程的虚拟地址空间



计算机系统简述

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，都可以视为文件。系统中的所有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。文件向应用程序提供了一个统一的视角，来看待系统中可能含有的所有各式各样的I/O 设备。例如，处理磁盘文件内容的应用程序员非常幸福，因为他们无需了解具体的磁盘技术。进一步说，同一个程序可以在使用不同磁盘技术的不同系统上运行。

计算机系统简述

从一个单独的系统来看，网络可视为一个I/O 设备，如图1-14 所示。当系统从主存将一串字节复制到网络适配器时，数据流经过网络到达另一台机器，而不是其他地方，例如本地磁盘驱动器。相似地，系统可以读取从其他机器发送来的数据，并把数据复制到自己的主存。

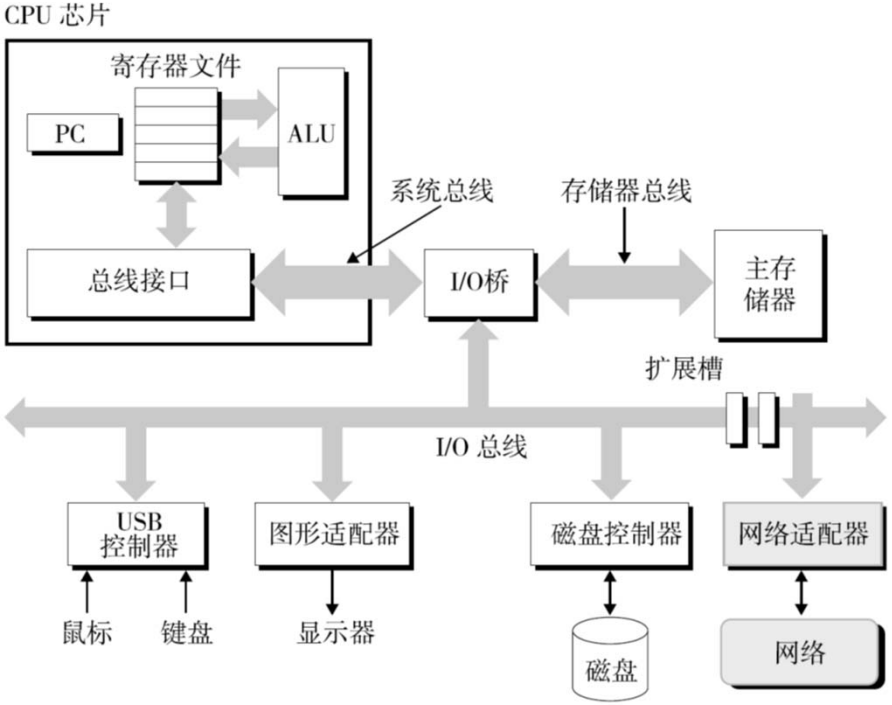


图 1-14　网络也是一种 I/O 设备

计算机系统简述

使用telnet 应用在一个远程主机上运行hello 程序。假设用本地主机上的telnet 客户端连接远程主机上的telnet 服务器。在我们登录到远程主机并运行外壳后，远端的外壳就在等待接收输入命令。当我们在telnet 客户端键入“hello”字符串并敲下回车键后，客户端软件就会将这个字符串发送到telnet 的服务器。telnet 服务器从网络上接收到这个字符串后，会把它传递给远端外壳程序。接下来，远端外壳运行hello 程序，并将输出行返回给telnet 服务器。最后，telnet 服务器通过网络把输出串转发给telnet 客户端，客户端就将输出串输出到本地终端上。

图 1-15　利用 telnet 通过网络远程运行 hello 程序

1.1.1 Java的诞生和发展

1. 前身
   - **James Gosling**，智能家用电器嵌入
2. 诞生，**1995**年，**JDK 1.0**
3. **Java 2**平台（**JDK 1.2**）
4. 三个平台（**JDK 1.3**）
   ① **Java ME**（嵌入式平台）
   ② **Java SE**（标准平台）
   ③ **Java EE**（企业级平台）

1.1.1 Java的诞生和发展

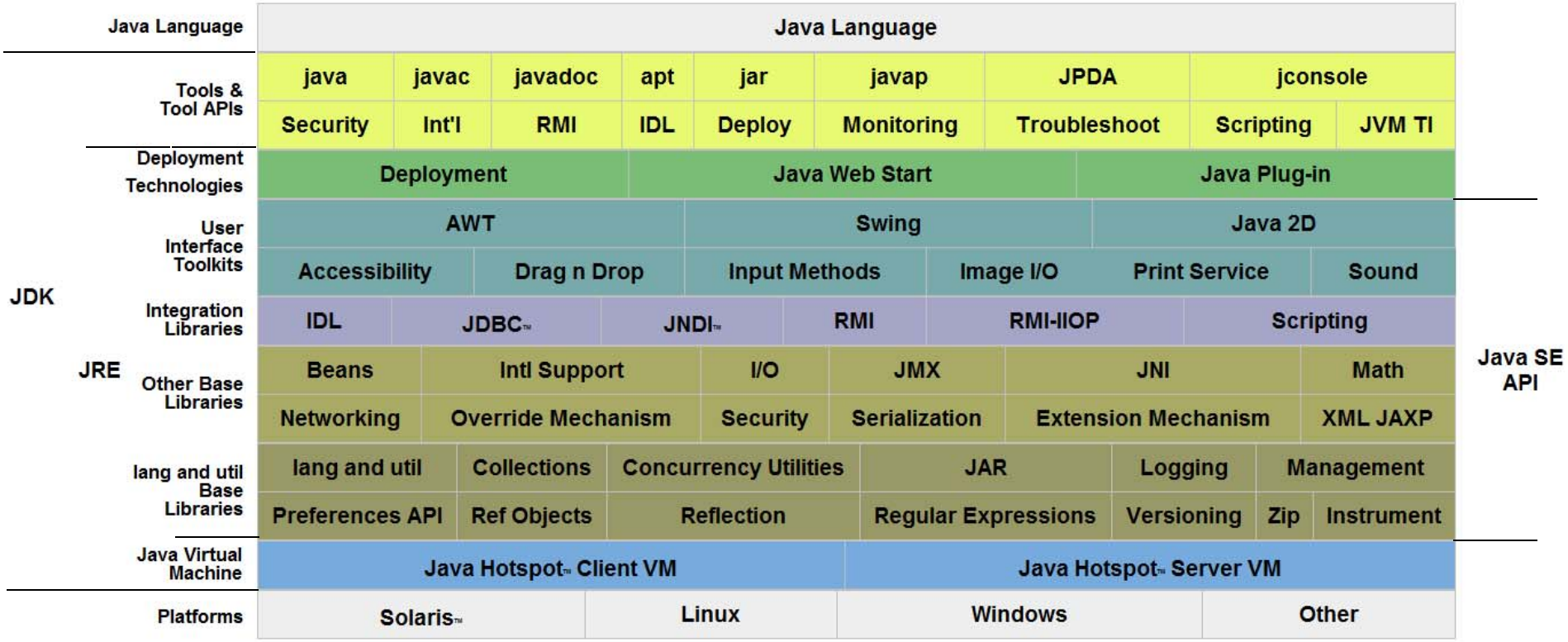
JDK&JRE&JVM

① **JDK: Java Development Kit　　Java**开发工具包

② **JRE: Java Runtime Environment　　Java**运行时环境

③ **JVM: Java Virtual Machine　　Java**虚拟机

1.1.1 Java的诞生和发展



1.1.2 Java特点

1. 跨平台特性

“Write once, run anywhere”，Java虚拟机

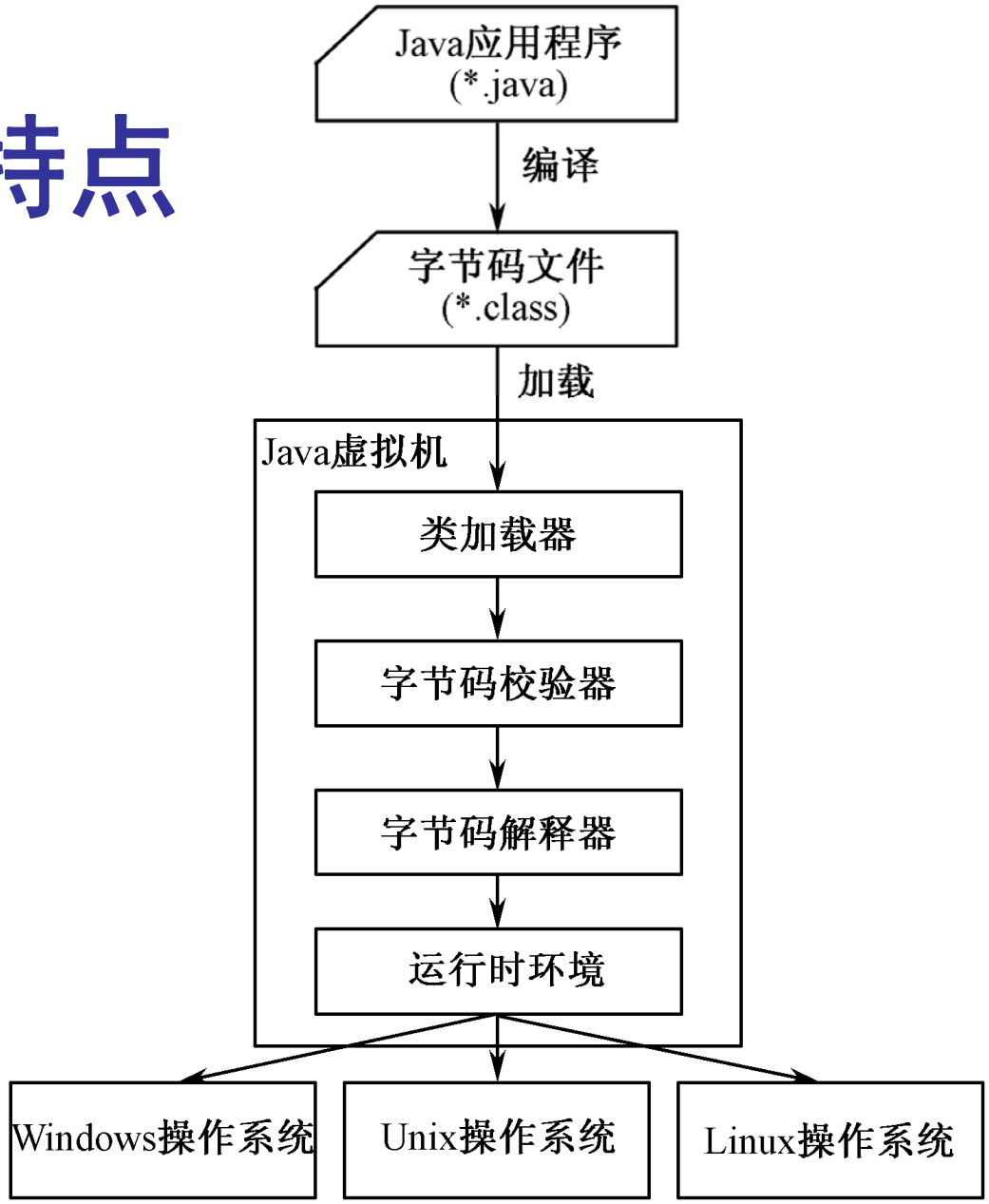


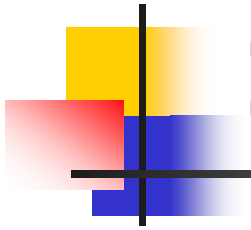
图1.1 Java虚拟机执行Java程序

1.1.2 Java特点

2. 完全面向对象和简单性

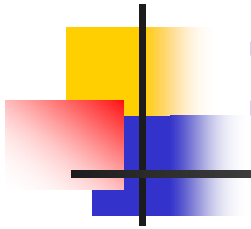
① 语法：放弃了**C++**语言的全程变量、**goto**语句、宏定义、全局函数以及结构、联合和指针数据类型。

② 面向对象概念：放弃了多重继承、友元类和运算符重载，采用“单重继承+接口”的方式代替多重继承，提供语言级的内存自动管理和异常处理方式。



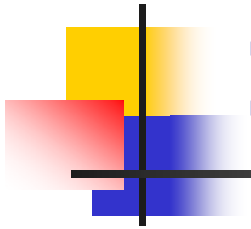
1.1.2 Java特点

3. 可靠性
   - ① 语言级别：提供严密的语法规则，在编译和运行时严格检查错误。
   - ② 运行架构级别：安全模型
4. 安全性
5. 多线程
6. 直接支持分布式的网络应用，高效的网络编程



1.1.3 Java核心技术

1. **JDBC，Java数据库连接**
2. **JSP，动态网页技术标准**
3. **JavaBean和EJB，Java的对象组件技术**
4. **JavaMail，E-mail邮件服务**



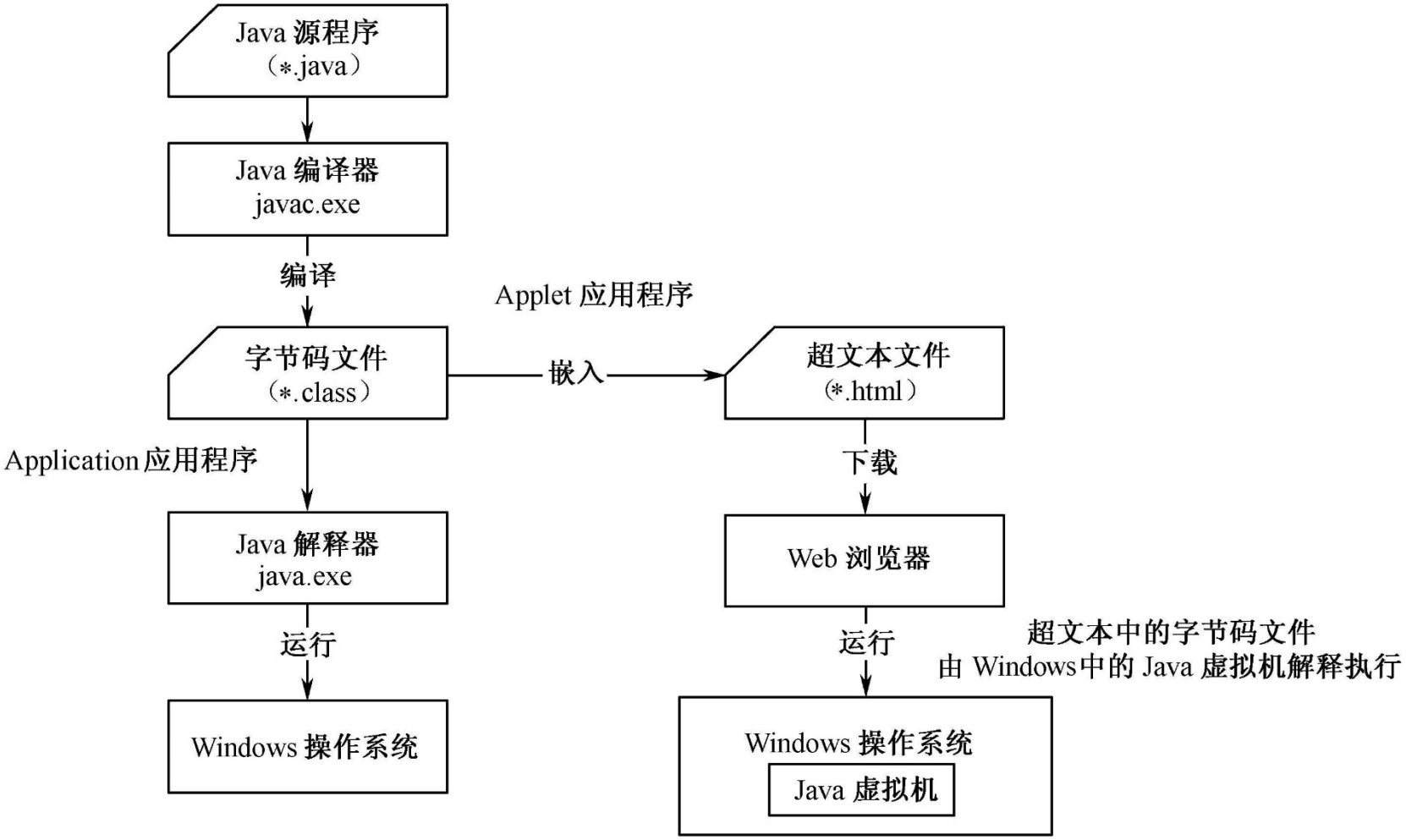
1.1.4 Application应用

1. **Application**，是能够独立运行的应用程序，有控制台和图形用户界面两种运行方式。
2. **Applet**，是可以嵌入**Web**页面的最小应用，它不能独立运行，必须嵌入超文本（***.html**）中，由浏览器中的**Java**解释器解释执行。

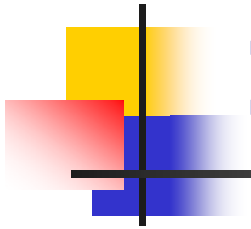
【例1.1】接受命令行参数的Application应用程序。

```
public class Hello
{
    public static void main(String args[])
    {
        if (args.length==0)
            System.out.println("Hello!");
        else
            for (int i=0; i<args.length; i++)
                System.out.println(args[i]);
    }
}
```

图1.2 Java程序运行流程



《Java程序设计实用教程(第4版)》



1.2 JDK

1. 1.2.1 JDK的安装与设置
2. 1.2.2 编译和运行Java程序
3. 1.2.3 包

1.2.1 JDK的安装与设置

1. 安装JDK
2. 设置环境变量
   ① **Windows**中设置环境变量
   ② 设置环境变量的批命令，**jdk7.bat**

```
set path=%path%;C:\Program
    Files\Java\jdk1.7.0_07\bin
set classpath=.;C:\Program
    Files\Java\jdk1.7.0_07\lib
```

1.2.2 编译和运行Java程序

1. 执行批命令设置环境变量

```
C:\>d:                          //d: 转换盘符
D:\>cd myjava                   //进入myjava文件夹
D:\myjava>jdk7          //执行批命令文件jdk7.bat
```

2. 编译

```
D:\myjava>javac Hello.java
```

3. 运行Application应用程序

```
D:\myjava>java Hello
D:\myjava>java Hello > Hello.txt
```

4. 命令行参数

```
D:\myjava>java Hello World Welcome
```

1.2.2 编译和运行Java程序

第一个**JAVA**程序的总结：

① **Java**对大小写敏感，如果出现了大小写拼写错误，程序无法运行。

② **"java"** 不是内部或外部命令，也不是可运行的程序或处理文件。

③ 关键字**public**被称作访问修饰符（**access modifier**）,用于控制程序的其它部分对这段代码的访问级别。

④ 关键字**class** 表明**Java** 程序中的全部内容都包含在类中，**Java**是一种纯面向对象的语言。

⑤ 一个源文件中至多只能有一个**public**的类声明，其它类的个数不限，如果源文件中包含一个**public** 类，源文件名必须和它中定义的**public**的类名相同，且以 **"java"**为扩展名。

⑥ 一个源文件可以有多个**class**。

⑦ 正确编译后的源文件，会得到相应的字节码文件，编译器为每个类生成独立的字节码文件，且将字节码文件自动命名为类的名字且以 **"class"**为扩展名。

⑧ **main**方法是**Java**应用程序的入口方法，它有固定的书写格式：**public static void main(String[] args) {...}**

⑨ 在**Java**中，用花括号划分程序的各个部分，任何方法的代码都必须以 **"{"**开始，以 **"}"**结束， 由于编译器忽略空格，所以花括号风格不受限制。

⑩ **Java**中每个语句必须以分号结束，回车不是语句的结束标志，所以一个语句可以跨多行。

⑪ 编程风格：注意缩进！成对编程（括号、引号都应该写完后，再往里加内容）！见名知意！

1.2.2 编译和运行Java程序

Java注释：

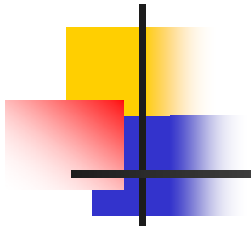
① 注释就是程序员为读者作的说明，是提高程序可读性的一种手段。

② 注释不会出现在字节码文件中。即**java**编译器编译时会跳过注释语句。

③ **//** 单行注释（注释内容从**//** 到本行结束）。

④ **/* */** 多行注释 ——注释不能嵌套。

⑤ **/** ** */**文档注释，用于源代码自动生成文档。执行**javadoc**命令，可根据源代码中的内容生成**API**文档。



1.2.3 包

package 包（作用类似于文件夹）：为了解决类之间重名问题；为了便于管理类：合适的类位于合适的包。通常是类的第一句非注释性语句；用打包语句：”**package** 包名**;**”；包名：域名倒着写即可再加上模块名，并与内部管理类。写项目时都要加包，不要使用默认包；**edu.whut**和**edu.whut.info**，这两个包没有包含关系，是两个完全独立的包，只是逻辑上看起来后者是前者的一部分。

1.2.3 包

import：如果不使用**import**，我们如果用到其他包的类时，只能这么写：**java.util.Date**，代码量太大，不利于编写和维护。通过**import**可以导入其他包下面的类，从而可以在本类中直接通过类名来调用。

- **java**会默认导入**java.lang**包下所有的类，因此这些类我们可以直接使用。
- 如果导入两个同名的类，只能用包名**+**类名来显示调用相关类。

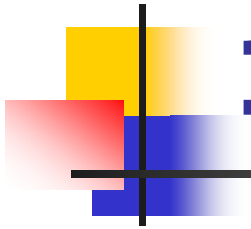
static import 静态导入（用于导入指定类的静态属性）。使用方法：

1. **import static java.lang.Math.*;//**导入**Math**类的所有静态属性
2. **import static java.lang.Math.PI;//**导入**Math**类的**PI**属性
3. 然后，可以在程序中直接使用：**System.out.println(PI);**

1.2.3 包

JDK 中主要的包：

- **java.lang**：包含一些**Java**语言的核心类，如**String**、**Math**、**Integer**、**System**和**Thread**，提供常用功能。
- **java.awt**：包含了构成抽象窗口工具集（**abstract window toolkits**）的多个类，这些类被用来构建和管理应用程序的图形用户界面**(GUI)**。
- **java.net**：包含执行与网络相关的操作的类。
- **java.io**：包含能提供多种输入/输出功能的类。
- **java.util**：包含一些实用工具类，如定义系统特性、使用与日期日历相关的函数。



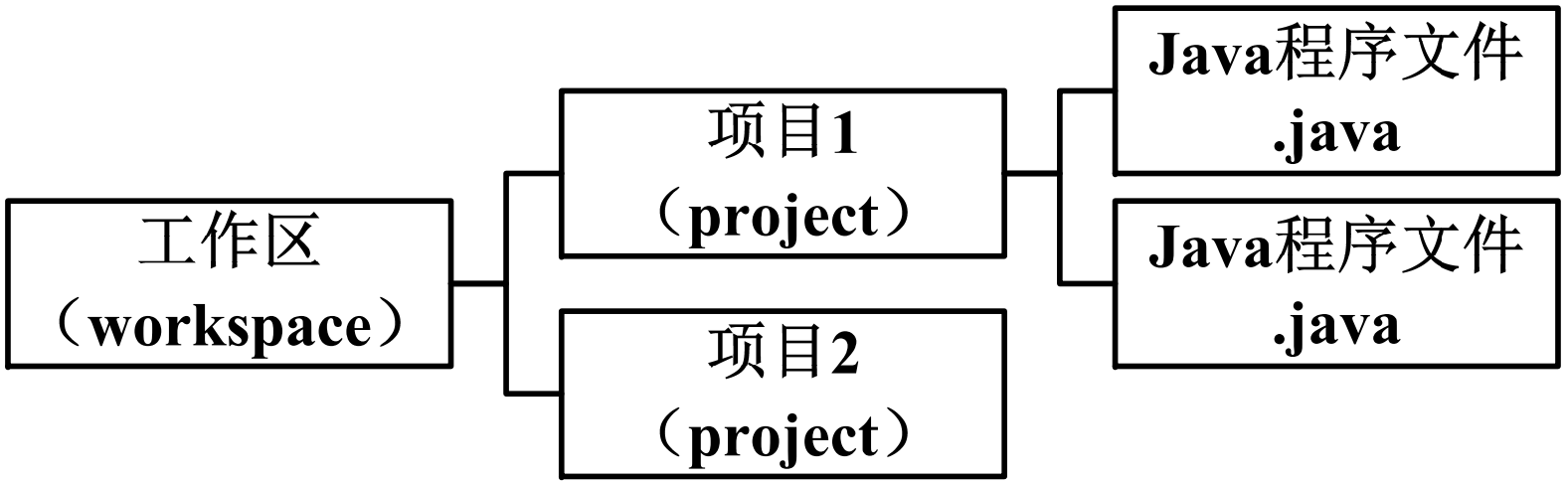
1.3 Eclipse

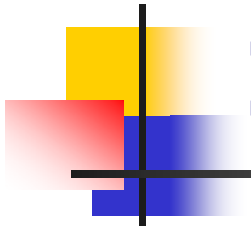
Eclipse提供**Java**程序的编辑、编译和运行的集成开发环境。

1. **1.3.1 Eclipse**集成开发环境
2. **1.3.2** 创建**Java**项目并运行
3. **1.3.3** 程序调试技术

1.3.1 Eclipse集成开发环境

1. 安装**Eclipse**并启动
2. 界面
3. 代码提示和源代码查看
4. 项目和工作区





1.3.2 创建Java项目并运行

1. 新建**Java**项目
2. 新建**Java**类
3. 编辑、编译和运行
4. 重构
5. 切换工作区
6. 创建包
7. 导入项目
8. 设置注释行

1.3.2 创建Java项目并运行

9. 设置项目属性

① 设置运行属性

② 配置编译路径

③ 添加**JAR**包

10.设置环境属性

① 更新**JDK**

② 修改编辑区的字体和颜色

③ 设置默认字符集

1.3.3 程序调试技术

1. 程序错误、发现时刻及错误处理原则

   语法错、语义错、逻辑错。

2. 程序运行方式

   正常运行、单步运行、分段运行

3. 调试过程

   设置断点、调试界面、单步或分段运行 、查看变量的当前值

实验1 Application程序的编译和运行

- 要求：掌握在**JDK**和**Eclipse**中编译和运行**Application**程序的操作。