Technology Tarena 达内科技

Unix系统高级编程

Primer DAY01

内容

上午	09:00 ~ 09:30	Unix系统简介
	09:30 ~ 10:20	
	10:30 ~ 11:20	Linux系统简介
	11:30 ~ 12:20	GNU编译器
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	静态库
	16:00 ~ 16:50	共享库
	17:00 ~ 17:30	总结和答疑

Unix系统简介


Unix系统简介
Unix系统的背景
Unix的三大流派
Unix家族
Unix系统的背景
Unix的三大流派
Unix家族

Technology Tarena 达内科技

Unix系统的背景

Unix系统的背景

- 史前时代：1961-1969
  - CTSS
    Compatible Time-Sharing System
    兼容分时系统，小而简单的实验室原型
  - Multics
    Multiplexed Information and Computing Service
    多路信息与计算服务，庞大复杂，不堪重负
  - Unics
    Uniplexed Information and Computing Service
    单路信息与计算服务，返朴归真，走上正道

知识讲解

Unix系统的背景（续1）

Technology Tarena 达内科技

知识讲解

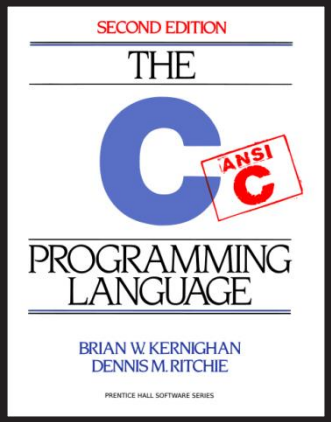
- 创世纪：1969-1971
  - 1969年，AT&T退出了Multics项目，来自贝尔实验室的Ken Thompson为了在PDP-7上实现他的星际旅行游戏，编写了一系列实用程序，成为后来Unix系统的核心
  - 1970年，Ken Thompson在BCPL语言的基础上发明了B语言，当时的Unix系统用汇编语言和B语言混合编写
  - 1971年，第一个Unix应用程序nroff诞生，为贝尔实验室的文字处理工作提供支持，Unix系统开始向PDP-11移植

Unix系统的背景（续2）

Technology Tarena 达内科技

- 出谷记：1971-1979
  - 1971年，Dennis Ritchie在B语言的基础上发明了C语言
  - 1973年，Dennis Ritchie用C语言重写了整个Unix系统，极大地提升了Unix系统的可读性、可维护性和可移植性
  - 1974年，Ritchie和Thompson在《美国计算机通信》上发表论文，第一次公开展示Unix
  - 1979年，贝尔实验室发布Unix V7版本，成为Unix世界公认的第一个真正意义上的Unix操作系统

知识讲解



Unix系统的背景（续3）

Technology Tarena 达内科技

- 第一次Unix战争：1980-1985
  - 1980年，美国国防部高级研究计划局(DARPA)决定在加州大学伯克利分校开发的BSD Unix上实现TCP/IP协议栈
  - 1983年，AT&T的拆分使其得以将Unix System V商业化
  - 1984年，第一次Unix战争在AT&T的Unix System V和伯克利的BSD Unix之间全面爆发
  - 1985年，以IEEE POSIX为核心的一系列技术标准最终弥合了Unix System V和BSD Unix之间的裂痕

知识讲解

Unix系统的背景（续4）

- 第二次Unix战争：1988-1990
  - 1988年，AT&T持有Sun公司20%的股份，宣告两家公司正式联姻
  - 1989年，IBM、DEC、HP等二线厂商创建开放软件基金会(OSF)并结成盟友，以与AT&T/Sun轴心对抗，第二次Unix战争爆发
  - 1990年，Windows 3.0发布，Unix世界从酣战中幡然醒悟，崭新的帝国正在崛起，真正的敌人来自Redmond

知识讲解

Unix系统的背景（续5）

Technology Tarena 达内科技

- 尘埃落定：1991-2000
  - 1991年，芬兰大学生Linus Torvalds宣布了Linux项目
  - 1993年，Linux已达到产品级操作系统的水准
  - 1993年，AT&T将Unix系统实验室卖给了Novell
  - 1994年，Novell将Unix商标卖给了X/Open标准组
  - 1995年，Novell将UnixWare卖给了SCO
  - 2000年，SCO将Unix源码卖给了Caldera—Linux发行商

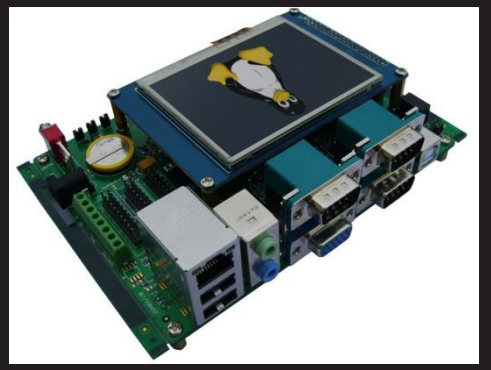
知识讲解

Technology Tarena 达内科技

Unix的三大流派

Unix的三大流派

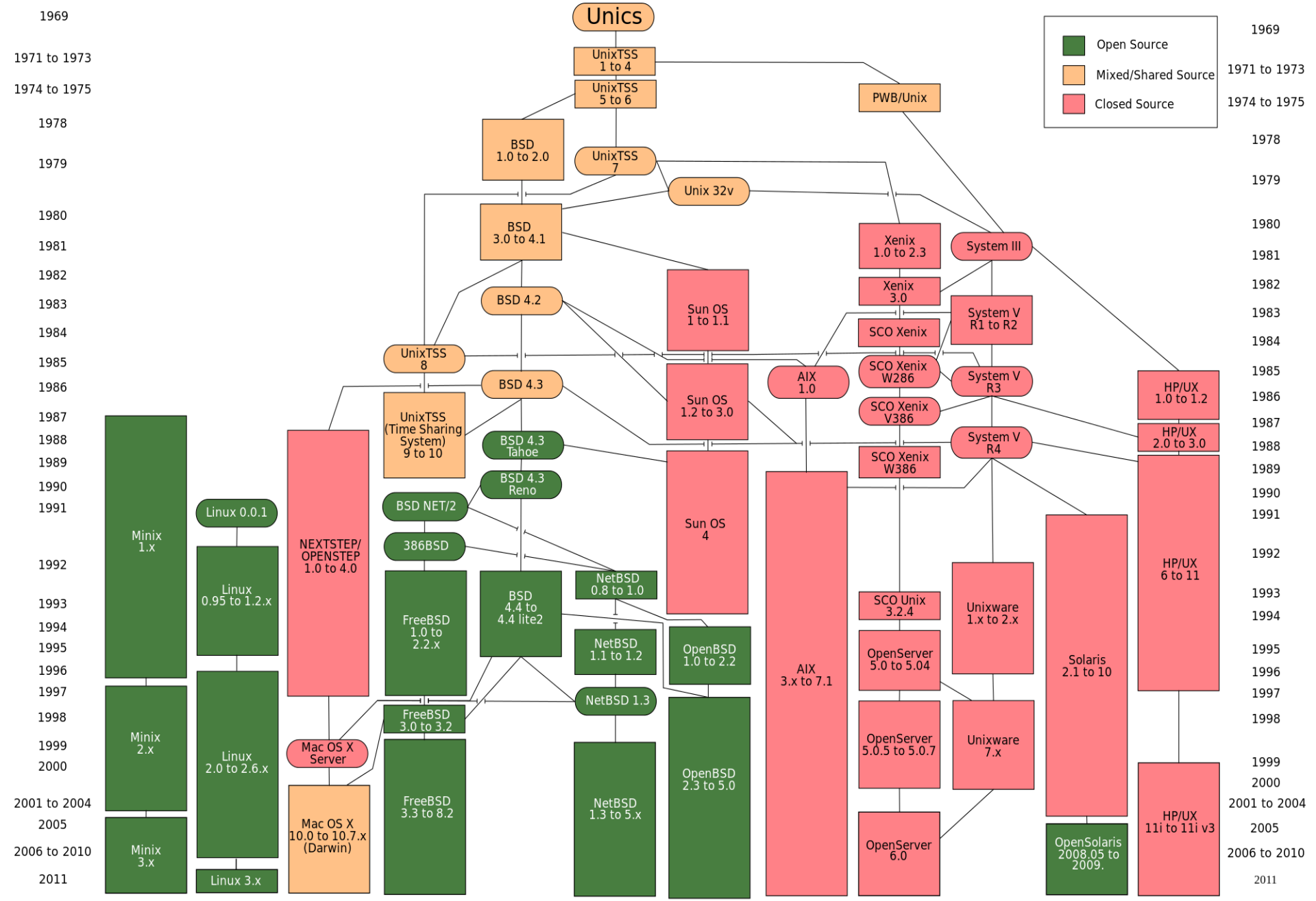
- System V
  - AIX，IBM，银行
  - Solaris，SUN，Oracle，电信
  - HP-UX
- Berkley
  - FreeBSD
  - NetBSD
  - OpenBSD，Mac OS X
- Hybrid
  - Minix，迷你版的类Unix操作系统
  - Linux，GPL，免费开源



知识讲解

Technology Tarena 达内科技

Unix家族


Unics
UnixTSS 1 to 4
UnixTSS 5 to 6
PWB/Unix
BSD 1.0 to 2.0
UnixTSS 7
Unix 32v
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
UnixTSS 8
SCO Xenix W286
AIX 1.0
System V R3
HP/UX 1.0 to 1.2
BSD 4.3
Sun OS 1.2 to 3.0
SCO Xenix V386
HP/UX 2.0 to 3.0
UnixTSS (Time Sharing System) 9 to 10
BSD 4.3 Tahoe
System V R4
SCO Xenix W386
BSD 4.3 Reno
BSD NET/2
Sun OS 4
386BSD
NetBSD 0.8 to 1.0
SCO Unix 3.2.4
Unixware 1.x to 2.x
HP/UX 6 to 11
OpenServer 5.0 to 5.04
Solaris 2.1 to 10
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
AIX 3.x to 7.1
NetBSD 1.3
OpenServer 5.0.5 to 5.0.7
Unixware 7.x
FreeBSD 3.0 to 3.2
OpenBSD 2.3 to 5.0
NetBSD 1.3 to 5.x
OpenServer 6.0
OpenSolaris 2008.05 to 2009.
HP/UX 11i to 11i v3
Minix 1.x
Linux 0.0.1
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
Minix 2.x
Linux 2.0 to 2.6.x
Mac OS X Server
FreeBSD 3.3 to 8.2
Minix 3.x
Mac OS X 10.0 to 10.7.x (Darwin)
Linux 3.x
Open Source
Mixed/Shared Source
Closed Source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2010
2011

Linux系统简介


Linux系统简介
Linux系统的背景
相关知识
Linux系统的版本
Linux系统的特点
Linux发行版本
Linux系统的背景
相关知识
Linux系统的版本
Linux系统的特点
Linux发行版本

Technology Tarena 达内科技

Linux系统的背景

Linux系统的背景

知识讲解

- Linux是一款类Unix操作系统，免费开源
- Linux的不同发行版本都使用相同的内核
- Linux可以运行在手机、平板、路由器、视频游戏控制器、个人电脑、大型计算机、超级计算机等多种硬件平台上
- 严格意义上的Linux仅指操作系统内核，但一般被用于指称某个具体的发行版本
- Linux隶属于GNU工程，整套GNU工具包从一开始就内置其中，可提供高质量的开发工具
- Linux的发明人是Linus Torvalds，同时他也是Linux商标的合法持有者

Technology Tarena 达内科技

相关知识

相关知识

- Minix操作系统
  - Andrew S. Tanenbaum
  - 荷兰阿姆斯特丹Vrije大学
  - 数学与计算机科学系教授
  - ACM和IEEE的资深会员

- GNU工程
  - GNU is GNU Not Unix
  - Richard Stallman发起于1984年，由自由软件基金会(FSF)提供支持
  - GNU的基本原则就是共享，其主旨在于发展一个有别于一切商业Unix系统的，免费且完整的类Unix系统

知识讲解

相关知识（续1）

知识讲解

- POSIX标准
  - Portable Operating System Interface for Computing Systems，统一系统编程接口规范
  - 由IEEE和ISO/IEC开发
  - 保证应用程序源代码级的可移植性
  - Linux完全遵循POSIX标准

- GPL
  - General Public License，通用公共许可证
  - 如果发布了可执行的二进制代码，就必须同时发布可读的源代码，并且在发布任何基于GPL许可的软件时，不能添加任何限制性的条款

Technology Tarena 达内科技

Linux系统的版本

Linux系统的版本

- 早期版本
  - 0.01
    0.02
    ...
    0.99
    1.00
- 旧计划
  - 介于1.0.1和2.6.0之间，A.B.C
    A：主版本号，内核大幅更新
    B：次版本号，内核重大修改，奇数测试版，偶数稳定版
    C：补丁序号，内核轻微修订

知识讲解

Linux系统的版本（续1）

- 2003年12月发布2.6.0以后
  - 缩短发布周期，A.B.C-D.E

    D：构建次数，反映极微小的更新

    E：描述信息，如：

    rc/r - 候选版本，后跟候选版本号，越大越接近正式版

    smp - 对称多处理器版本

    pp - Red Hat Linux的测试版本

    EL - Red Hat Linux的企业版本

    mm - 测试新技术或新功能

    fc - Red Hat Linux的Fedora Core版本
- 查看系统版本

```
$ cat /proc/version
```

知识讲解

Linux系统的特点

Linux系统的特点

- 遵循GNU/GPL
- 开放性
- 多用户
- 多任务
- 设备独立性
- 丰富的网络功能
- 可靠的系统安全
- 良好的可移植性

知识讲解

Technology Tarena 达内科技

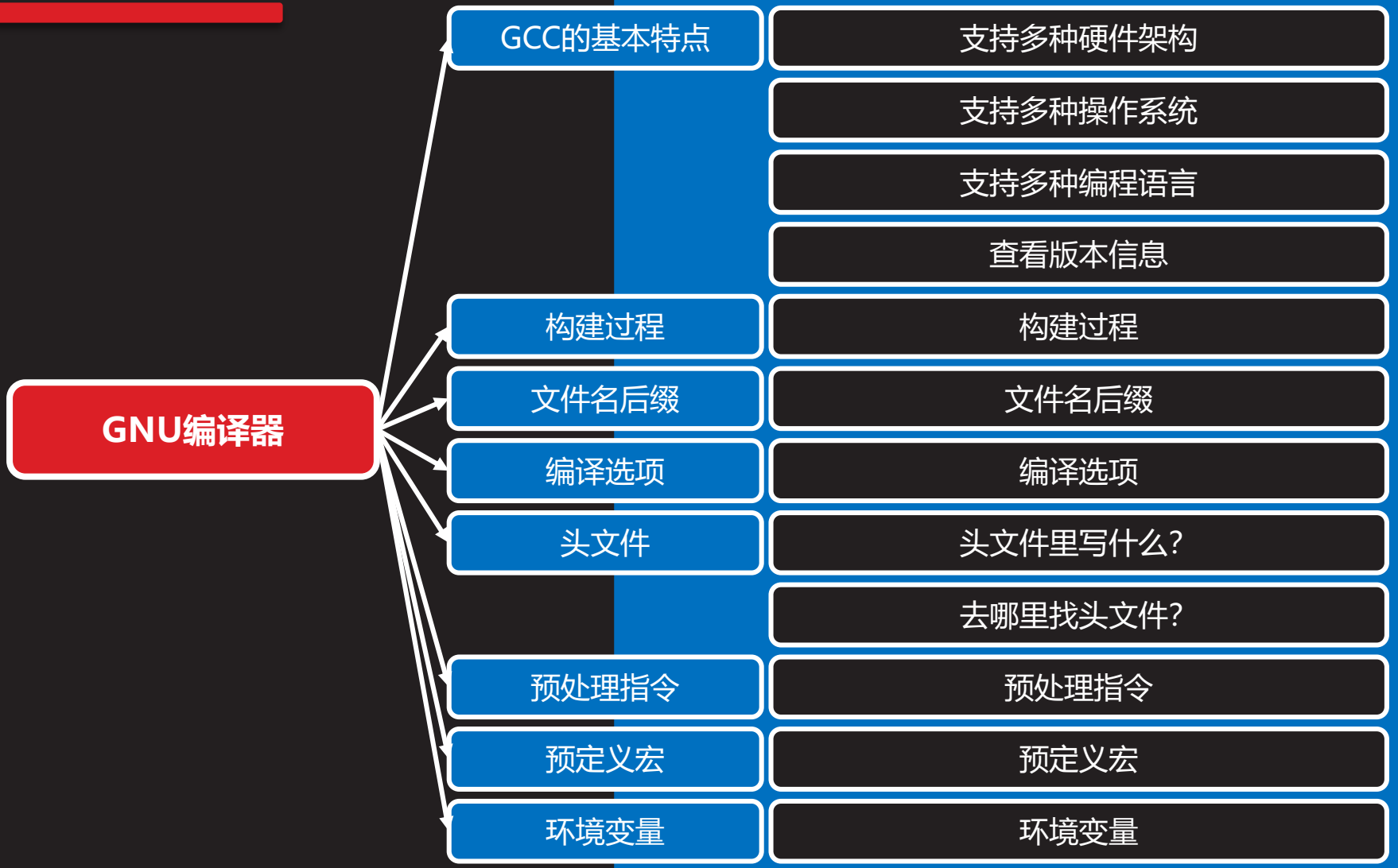
Linux发行版本

Linux发行版本

- 大众的Ubuntu
- 优雅的Linux Mint
- 锐意的Fedora
- 华丽的openSUSE
- 自由的Debian
- 简洁的Slackware
- 老牌的RedHat

知识讲解

GNU编译器

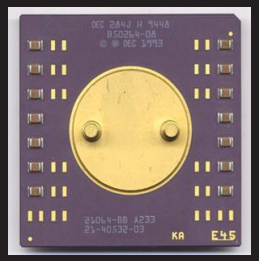

GNU编译器
GCC的基本特点
支持多种硬件架构
支持多种操作系统
支持多种编程语言
查看版本信息
构建过程
构建过程
文件名后缀
文件名后缀
编译选项
编译选项
头文件
头文件里写什么?
去哪里找头文件?
预处理指令
预处理指令
预定义宏
预定义宏
环境变量
环境变量

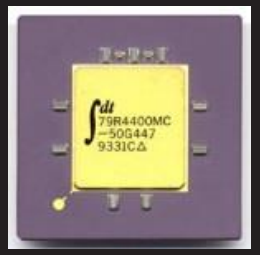
Technology Tarena 达内科技

GCC的基本特点

支持多种硬件架构

- x86-64
- Alpha
- ARM
- Motorola 68000
- MIPS
- PDP-10/11
- PowerPC
- System/370-390
- SPARC
- VAX

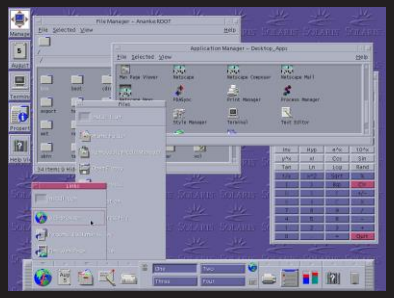


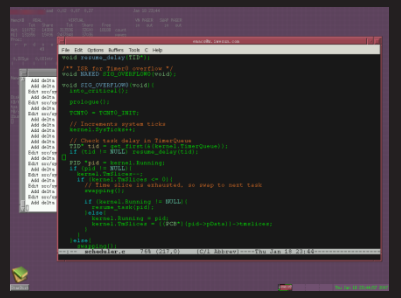


知识讲解

支持多种操作系统

- Unix
- Linux
- BSD
- Android
- Mac OS X
- iOS
- Windows





支持多种编程语言

- C
  - 以简洁高效的方式控制系统底层，无需环境支持便能运行
- C++
  - 兼容C语言，集面向过程、面向对象和泛型编程于一身
- Objective-C
  - 扩充C语言的面向对象特性，主要用于Mac OS X和GNUstep这两个使用OpenStep标准的系统
- Java
  - 面向对象且跨平台， 需要虚拟机支持，号称一次编译到处运行

支持多种编程语言（续1）

- Fortran
  - 最早的高级程序设计语言，主要用于科学和工程计算
- Pascal
  - 最早的结构化程序设计语言，语法严谨，层次分明，擅于描述算法和数据结构
- Ada
  - 迄今为止最复杂最完备的程序设计语言，代表了全世界程序设计语言领域的最高成就，甚至突破了冯诺依曼思维模式的桎梏，标志着软件工程已经发展到国家和国际的规模
  - 美国国防部指定的唯一一种可应用于军事系统开发的语言
  - 我国军方也将其作为军内系统开发标准：GJB1383-1998

知识讲解

查看版本信息

- GCC的早期版本只能编译C语言程序
  - GNU C Compiler
- GCC现在的版本已可以编译多种语言程序
  - GNU Compiler Collection
- 查看编译器版本

```
$ gcc -v
```

知识讲解

Technology Tarena 达内科技

构建过程

构建过程

- 从源代码到可执行程序的构建过程
  1. 预编译(编译预处理)：头文件扩展、宏扩展
     $ gcc -E hello.c -o hello.i -> hello.i
  2. 编译：将高级语言翻译成汇编语言，得到汇编文件
     $ gcc -S hello.i -> hello.s
  3. 汇编：将汇编语言翻译成机器指令，得到目标模块
     $ gcc -c hello.s -> hello.o
  4. 链接：将目标模块和标准库链接，得到可执行程序
     $ gcc hello.o -o hello -> hello
- 运行

```
$ hello
```

知识讲解

Hello, World !

【参见：hello.c】

课堂练习

- Hello, World !

Technology Tarena 达内科技

文件名后缀

文件名后缀

- `.h` ：C语言源代码头文件
- `.c` ：预处理前的C语言源代码文件
- `.i` ：预处理后的C语言源代码文件
- `.s` ：汇编语言文件
- `.o` ：目标文件
- `.a` ：静态库文件
- `.so`：共享库(动态库)文件

知识讲解

Technology Tarena 达内科技

编译选项

编译选项

$ gcc [选项] [参数] 文件1 文件2 ...

- `-o`：指定输出文件
  - $ gcc hello.c -o hello
- `-E`：预编译，缺省输出到屏幕，用-o指定输出文件
  - $ gcc -E hello.c -o hello.i
- `-S`：编译，将高级语言文件编译成汇编语言文件
  - $ gcc -S hello.c
- `-c`：汇编，将汇编语言文件汇编成机器语言文件
  - $ gcc -c hello.c

知识讲解

编译选项（续1）

- `-Wall` ：产生尽可能多的警告
  - $ gcc -Wall wall.c
- `-Werror` ：将警告作为错误处理
  - $ gcc -Werror werror.c
- `-x` ：显式指定源代码的语言
  - $ gcc -x c++ cpp.c -lstdc++
- `-pedantic` ：对非标准的扩展语法产生警告
- `-g` ：产生调试信息
- `-O0/O1/O2/O3` ：指定优化等级，O0不优化，缺省O1

知识讲解

产生尽可能多的警告

【参见：wall.c】

课堂练习

- 产生尽可能多的警告

将警告作为错误处理

【参见：werror.c】

课堂练习

- 将警告作为错误处理

指定源代码的语言

【参见：cpp.c】

课堂练习

- 指定源代码的语言

Technology Tarena 达内科技

头文件

头文件里写什么?

- 头文件卫士
  - `#ifndef __GEOMETRY_H__`
    `#define __GEOMETRY_H__`
    `...`
    `#endif`
- 包含其它头文件
  - #include <sys/types.h>
- 宏定义
  - #define PI 3.14159
- 自定义类型
  - struct Circle { double x; double y; double r; };

知识讲解

头文件里写什么？（续1）

- 类型别名
  - typedef struct Circle CIRCLE;
- 外部变量声明
  - extern double e /* = 2.71828 */;
- 函数声明
  - double circleArea (CIRCLE const*);
- 函数定义不要写在头文件里
  - 一个头文件可能会被多个源文件包含，写在头文件里的函数定义也会因此被预处理器扩展到多个包含该头文件的源文件中，并在编译阶段被编译到多个不同的目标文件中，这将直接导致链接错误：multiple definition，多重定义

去哪里找头文件?

- 通过gcc的-I选项指定头文件的附加搜索路径
  - gcc math.c calc.c -I.
- #include <calc.h>
  - 先找-I指定的目录，再找系统目录
- #include "calc.h"
  - 先找-I指定的目录，再找当前目录，最后找系统目录
- 头文件的系统目录
  - /usr/include
  - /usr/local/include
  - /usr/lib/gcc/i686-linux-gnu/<版本号>/include
  - /usr/include/c++/<版本号> (C++编译器)

知识讲解

分文件声明、定义和调用函数

【参见：calc.h、calc.c、math.c】

课堂练习

- 分文件声明、定义和调用函数

Technology Tarena 达内科技

预处理指令

预处理指令

- `#include` ：将指定文件的内容插至此指令处
- `#define` ：定义宏
- `#undef` ：删除宏
- `#if` ：如果
- `#ifdef` ：如果宏已定义
- `#ifndef` ：如果宏未定义
- `#else` ：否则，与#if/#ifdef/#ifndef配合使用
- `#elif` ：否则如果，与#if/#ifdef/#ifndef配合使用
- `#endif` ：结束判定，与#if/#ifdef/#ifndef配合使用

知识讲解

预处理指令（续1）

- **#error** ：产生错误，结束预处理
  - #error "版本太低！"
- **#warning**：产生警告，不结束预处理
  - #warning "版本太高！"
- **#line** ：指定行号
  - #line 100
- **#pragma** ：设定编译器的状态或指示编译器的操作
  - #pragma GCC dependency "dep.c"
  - #pragma GCC poison goto
  - #pragma pack(1)

知识讲解

Technology Tarena 达内科技

产生错误和警告

【参见：error.c】

课堂练习

- 产生错误和警告

指定行号

【参见：line.c】

课堂练习

- 指定行号

编译器指示

【参见：pragma.c、dep.c】

课堂练习

- 编译器指示

Technology Tarena 达内科技

预定义宏

预定义宏

- `__BASE_FILE__` ：正在编译的源文件名
- `__FILE__` ：所在文件名
- `__LINE__` ：行号
- `__FUNCTION__` ：函数名
- `__func__` ：同__FUNCTION__
- `__DATE__` ：日期
- `__TIME__` ：时间
- `__INCLUDE_LEVEL__` ：包含层数，从0开始
- `__cplusplus` ：C++有定义，C无定义

知识讲解

打印预定义宏

【参见：print.h、predef.h、predef.c】

课堂练习

- 打印预定义宏

Technology Tarena 达内科技

环境变量

环境变量

- C_INCLUDE_PATH
  - C头文件的附加搜索路径，相当于gcc的-I选项
- CPATH
  - 同C_INCLUDE_PATH
- CPLUS_INCLUDE_PATH
  - C++头文件的附加搜索路径
- LIBRARY_PATH
  - 链接时查找静态库和共享库的路径
- LD_LIBRARY_PATH
  - 运行时查找共享库的路径

知识讲解

环境变量（续1）

- 举例
  - 通过-I选项将当前目录作为C头文件附加搜索路径

```
$ gcc calc.c cpath.c -I.
```

  - 将当前目录添加到CPATH环境变量中

```
$ export CPATH=$CPATH:.
```

  - 将对CPATH环境变量的设置写到登录脚本中

```
$ vi ~/.bashrc
```

或

```
$ vi ~/.bash_profile
```

写入：export CPATH=$CPATH:.

环境变量（续2）

- 举例
  - 重新登录，或手动执行登录脚本

```
$ source ~/.bashrc
```

或

```
$ source ~/.bash_profile
```

  - 在登录脚本中设置环境变量，每次登录都会自动生效

环境变量（续3）

- 头文件的三种定位方式
  - #include "<目录>/xxx.h"
    头文件路径发生变化，必须修改源程序
  - export CPATH=$CPATH:<目录>
    同时构建多个工程，容易引发冲突
  - gcc -I<目录>
    既不需要修改源程序，也不会有任何冲突，推荐使用此方法

知识讲解

头文件的附加搜索路径

【参见：calc.h、calc.c、cpath.c】

课堂练习

- 头文件的附加搜索路径

静态库

Technology Tarena 达内科技

何为静态库

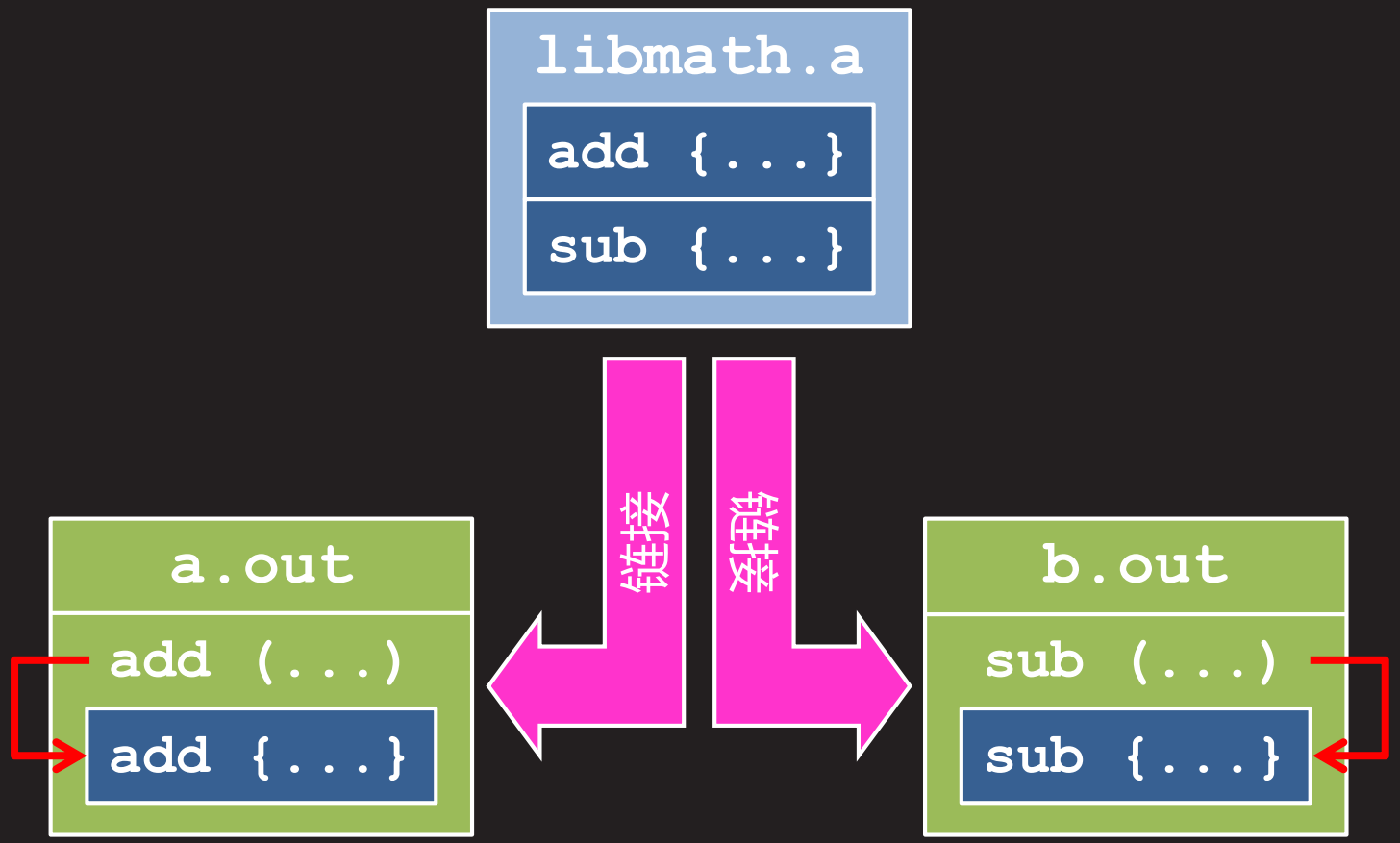
何为静态库

- 为什么要把一个程序分成多个源文件，并由每个源文件编译生成独立的目标文件?
  - 合久必分，化整为零，易于维护
- 为什么要把多个目标文件合并成一个库文件?
  - 分久必合，化零为整，方便使用
- 静态库的本质就是将多个目标文件打包成一个文件
- 链接静态库就是将库中被调用的代码复制到调用模块中
- 静态库占用空间大，库中代码一旦修改必须重新链接
- 使用静态库的代码在运行时无需依赖库，且执行效率高
- 静态库的缺省扩展名是.a

知识讲解

何为静态库（续1）

知识讲解



Technology Tarena 达内科技

构建静态库

构建静态库

知识讲解

- 编辑库的实现代码和接口声明
  - 计算模块：calc.h、calc.c
  - 显示模块：show.h、show.c
  - 接口文件：math.h
- 编译成目标文件

```
$ gcc -c calc.c
$ gcc -c show.c
```

- 打包成静态库文件

```
$ ar -r libmath.a calc.o show.o
```

- 向用户(库的使用者)提供libmath.a和math.h即可

构建静态库（续1）

- ar命令

```
$ ar [选项] <静态库文件> <目标文件列表>
```

-r - 将目标文件插入到静态库中，已存在则更新

-q - 将目标文件追加到静态库尾

-d - 从静态库中删除目标文件

-t - 列表显示静态库中的目标文件

-x - 将静态库展开为目标文件

知识讲解

Technology Tarena 达内科技

使用静态库

使用静态库

- 编辑库的使用代码
  - main.c
- 编译并链接静态库
  - 显示指定库文件的路径

```
$ gcc main.c libmath.a
```

  - 用-l选项指定库名，用-L选项指定库路径

```
$ gcc main.c -lmath -L.
```

  - 用-l选项指定库名，用LIBRARY_PATH环境变量指定库路径

```
$ export LIBRARY_PATH=$LIBRARY_PATH:.
$ gcc main.c -lmath
```

知识讲解

静态库的构建和使用

【参见：static/】

课堂练习

- 静态库的构建和使用

共享库

Technology Tarena 达内科技

何为共享库

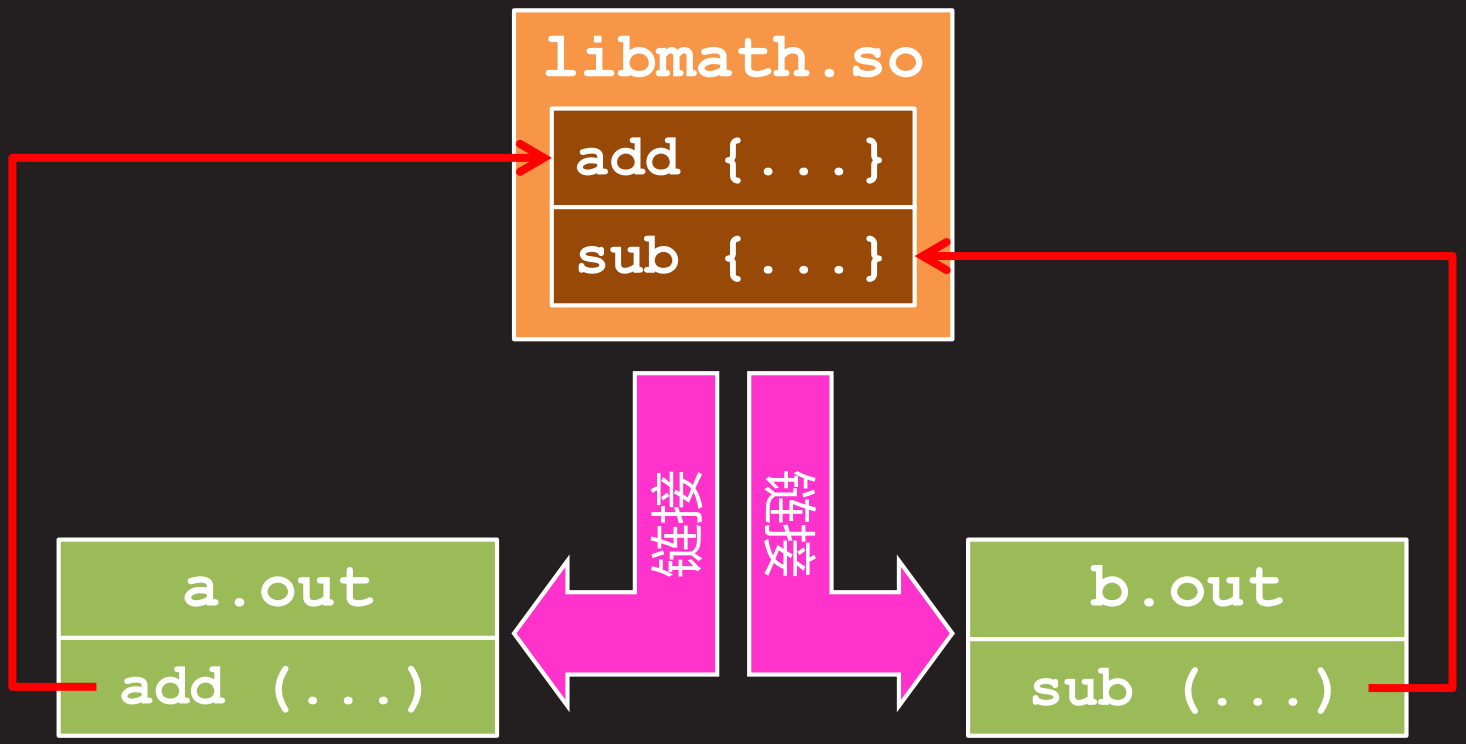
何为共享库

- 共享库和静态库最大的不同就是，链接共享库并不需要将库中被调用的代码复制到调用模块中，相反被嵌入到调用模块中的仅仅是被调用代码在共享库中的相对地址
- 如果共享库中的代码同时为多个进程所用，共享库的实例在整个内存空间中仅需一份，这正是共享的意义所在
- 共享库占用空间小，即使修改了库中的代码，只要接口保持不变，无需重新链接
- 使用共享库的代码在运行时需要依赖库，执行效率略低
- 共享库的缺省扩展名是.so

知识讲解

何为共享库（续1）

知识讲解



Technology Tarena 达内科技

构建共享库

构建共享库

- 编辑库的实现代码和接口声明
  - 计算模块：calc.h、calc.c
  - 显示模块：show.h、show.c
  - 接口文件：math.h
- 编译成目标文件

```
$ gcc -c -fpic calc.c
$ gcc -c -fpic show.c
```

- 连接成共享库文件

```
$ gcc -shared calc.o show.o -o libmath.so
```

- 向用户(库的使用者)提供libmath.so和math.h即可

知识讲解

构建共享库（续1）

- 编译和链接也可以合并为一步完成

```
$ gcc -shared -fpic calc.c show.c -o libmath.so
```

- PIC (Position Independent Code，位置无关代码)
  - 调用代码通过相对地址标识被调用代码的位置，模块中的指令与该模块被加载到内存中的位置无关
  - `-fPIC`：大模式，生成代码比较大，运行速度比较慢，所有平台都支持
  - `-fpic`：小模式，生成代码比较小，运行速度比较快，仅部分平台支持

知识讲解

Technology Tarena 达内科技

使用共享库

使用共享库

- 编辑库的使用代码
  - main.c
- 编译并链接共享库
  - 显示指定库文件的路径

```
$ gcc main.c libmath.so
```

  - 用-l选项指定库名，用-L选项指定库路径

```
$ gcc main.c -lmath -L.
```

  - 用-l选项指定库名，用LIBRARY_PATH环境变量指定库路径

```
$ export LIBRARY_PATH=$LIBRARY_PATH:.
$ gcc main.c -lmath
```

知识讲解

使用共享库（续1）

- 运行时需要保证LD_LIBRARY_PATH环境变量中包含共享库所在的路径

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
```

- 在可执行程序的链接阶段，并不将所调用函数的二进制代码复制到可执行程序中，而只是将该函数在共享库中的地址嵌入到调用模块中，因此运行时需要依赖共享库
- gcc缺省链接共享库，可通过-static选项强制链接静态库

```
$ gcc -static hello.c
```

知识讲解

共享库的构建和使用

【参见：shared/】

课堂练习

- 共享库的构建和使用

Technology Tarena 达内科技

总结和答疑