LinuxBIOS简介

唐 欢

htang@ncic.ac.cn

2003年3月28日

Outline

- LinuxBIOS简介
- 标准BIOS工作原理
- LinuxBIOS工作原理
- LinuxBIOS的应用方向
- Pink简介
- 未来方向

结束

LinuxBIOS简介

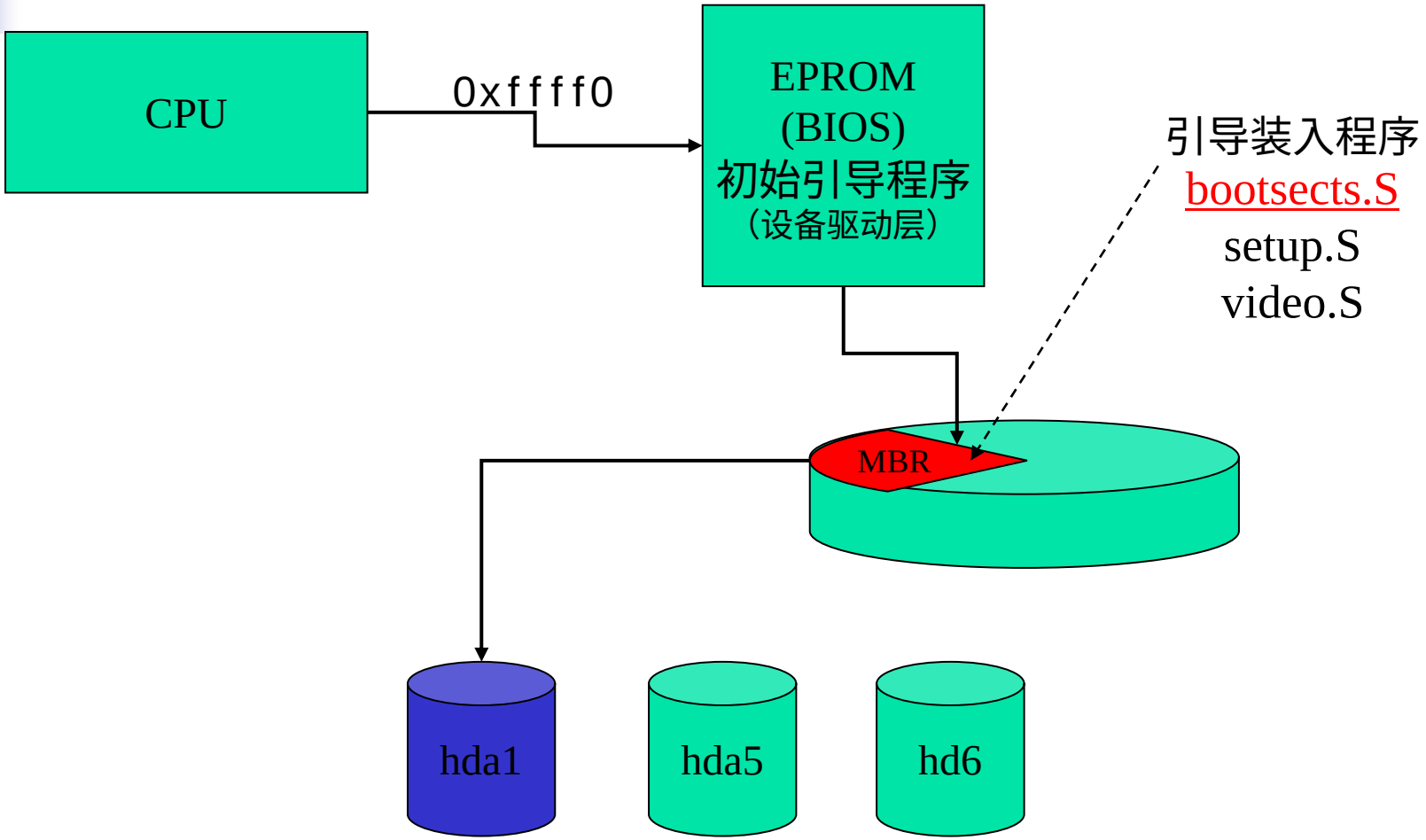
- LinuxBIOS 意在替代现有的标准BIOS，成为启动Linux内核的更为有效的工具。
- 主要目标：
  - 更快捷的引导Linux内核；
  - 减少在启动过程中对不可靠设备（软盘、硬盘）的依赖；
  - 加强机群的可维护性；
  - 简化网络启动过程，增加网络启动的途径；

LinuxBIOS简介—主要组成

- about 10 lines of patches to the current Linux kernel；（用于处理未初始化的硬件）
- startup code - about 500 lines of assembly and 5000 lines of C ;
- executes 16 instructions to get into 32-bit mode ;

本小节结束

标准BIOS工作原理


CPU
0xffff0
EPROM
(BIOS)
初始引导程序
（设备驱动层）
引导装入程序
bootsects.S
setup.S
video.S
MBR
hda1
hda5
hd6

用启动扇区做什么？

启动扇区（boot sector）在接过BIOS的“接力棒”后，首先要做的是把操作系统的kernel load到我们想要的位置，还要做的就是要进入保护模式。

机器初启的时候是在实模式中的，在这个模式下，只能访问低端的1M地址，1M以上的地址不能访问。而保护模式下最多可访问的地址是4G（2^{32}）。

标准BIOS不足之处

- BIOS启动系统的过程依赖于不稳定设备。
  - 不稳定设备指软盘、硬盘等，引导装入程序贮存在其中，每次启动时被读取并运行。
- BIOS提供的设备驱动不适应于Linux。
  - Linux是多用户、多进程、保护模式、页式映射操作系统，其内核并不使用BIOS设备驱动。则BIOS在自检过程中浪费很多时间。

标准BIOS不足之处（续）

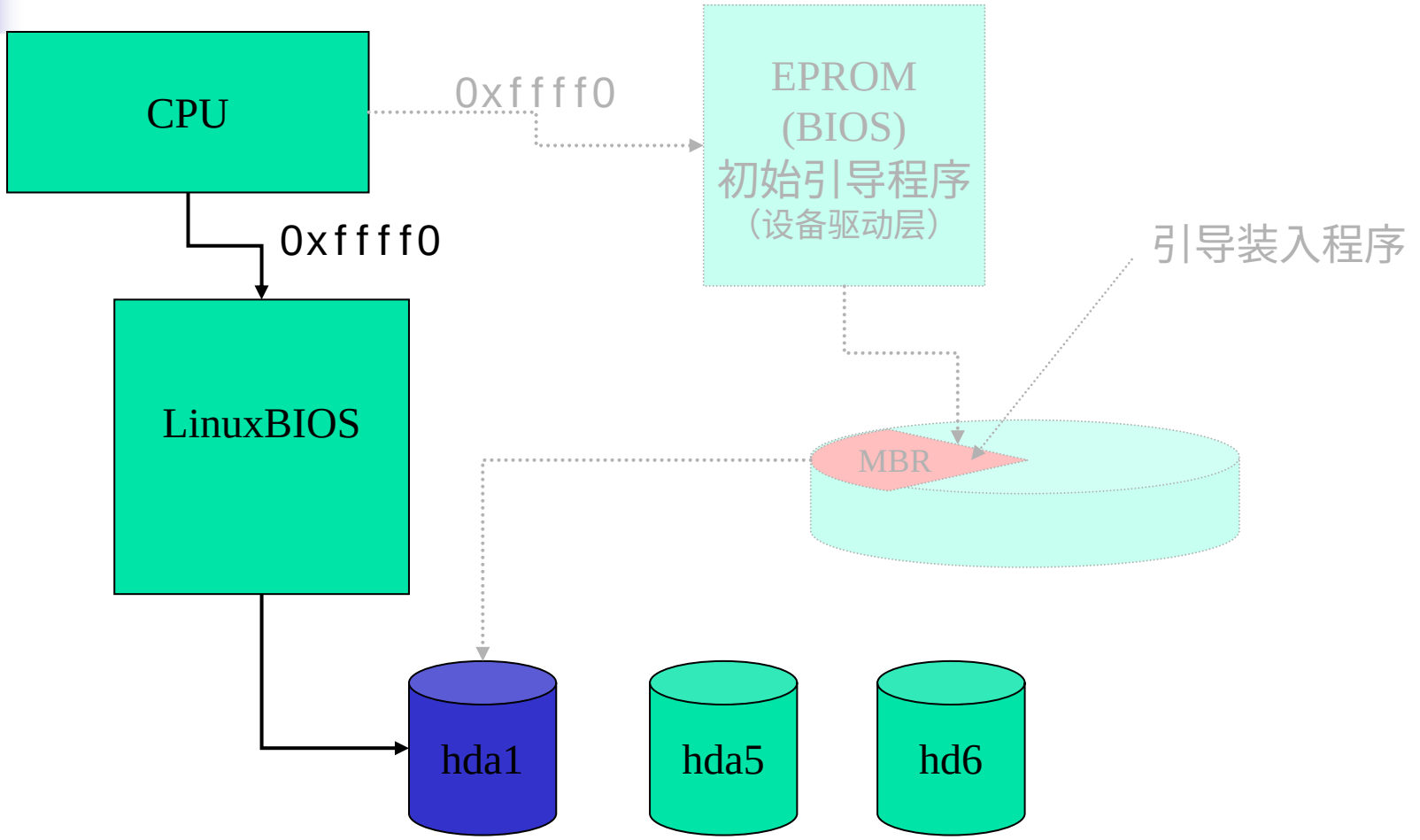
- 所有X86系列CPU都初始为16位的8086仿真模式。相应的，BIOS必须使用far-pointer 、near-pointer与之适应。
- 在机群管理中，对BIOS的维护工作很难。
- BIOS无法识别出非标准设备，为试验性工作带来了困难。
- BIOS is not Open-Source!

本小节结束

LinuxBIOS的出发点

- 舍弃标准BIOS中对设备的“多余的”初始化；
- 尽可能快的进入保护模式，提高系统的安全性与稳定性；
- 尽量避免汇编语言，使用效能更高的C语言；
- 代码公开，使LinuxBIOS能够在用户态下配置；
- 提供安全可行的机群管理机制。

LinuxBIOS工作原理



LinuxBIOS工作过程

(1) Protected mode setup.

进入保护模式，最大可寻址空间达到4G，运行于奔腾模式而非8086仿真模式。

(2) DRAM setup.

配置DRAM，为运行C做铺垫。

(3) Transition to C.

为实现C语言而建立堆栈并调用一个硬件配置函数。

LinuxBIOS工作过程一续

(4) Mainboard fix up.

主板的设置。

(5) Inflate the kernel. (inflate.c)

解压缩内核，内核初始化的命令以及参数被设定。内核被自动解压缩到标准位置。

(6) Jump to the kernel.

跳转至内核起始地址。很多标准内核启动代码不必执行，包括解压缩内核其他部分。不必使用LILO，直接跳转到startup_32。

本小节结束

LinuxBIOS的应用

- 嵌入式系统：
  - 没有沿袭传统BIOS结构，摒弃了冗余的代码；
  - 合理有效的应用ROM，发挥更多效能；
  - 启动速度高(0.8s)，适应高要求的场合；
  - 开放源码，可以开发有针对性的代码；

LinuxBIOS的应用—续

- 机群系统：
  - 对串行接口的提前配置，通过远程控制台访问，获得启动报告及错误报告；
  - 通过网络启动（Network Booting），通过DHCP改变配置；
  - 开发的源码可以在用户态配置管理。Anything done from user space also can be set up to be done remotely. 对同构机群尤为适合；

LinuxBIOS的应用—续

- 通过减少对硬件的依赖，提高机群的稳定性。可以避免使用磁盘、光盘等“不稳定”设备；
- 多种网络启动方式：
  - **Myrinet**
  - **Scaleable Coherent Interface (SCI)**
  - **using IP Multicast**

LinuxBIOS目前状态

- 支持芯片：
  - AMD Athlon，Duron，AMD 760, AMD 760MP
  - Pentium II, Pentium III,
  - Alpha 211264 CPUs, ALI m1631, Digital Tsunami
  - Intel 440BX, Intel 440GX, VIA VT8601,
  - SiS540, SiS550, SiS630 and SiS730

本小节结束

Pink

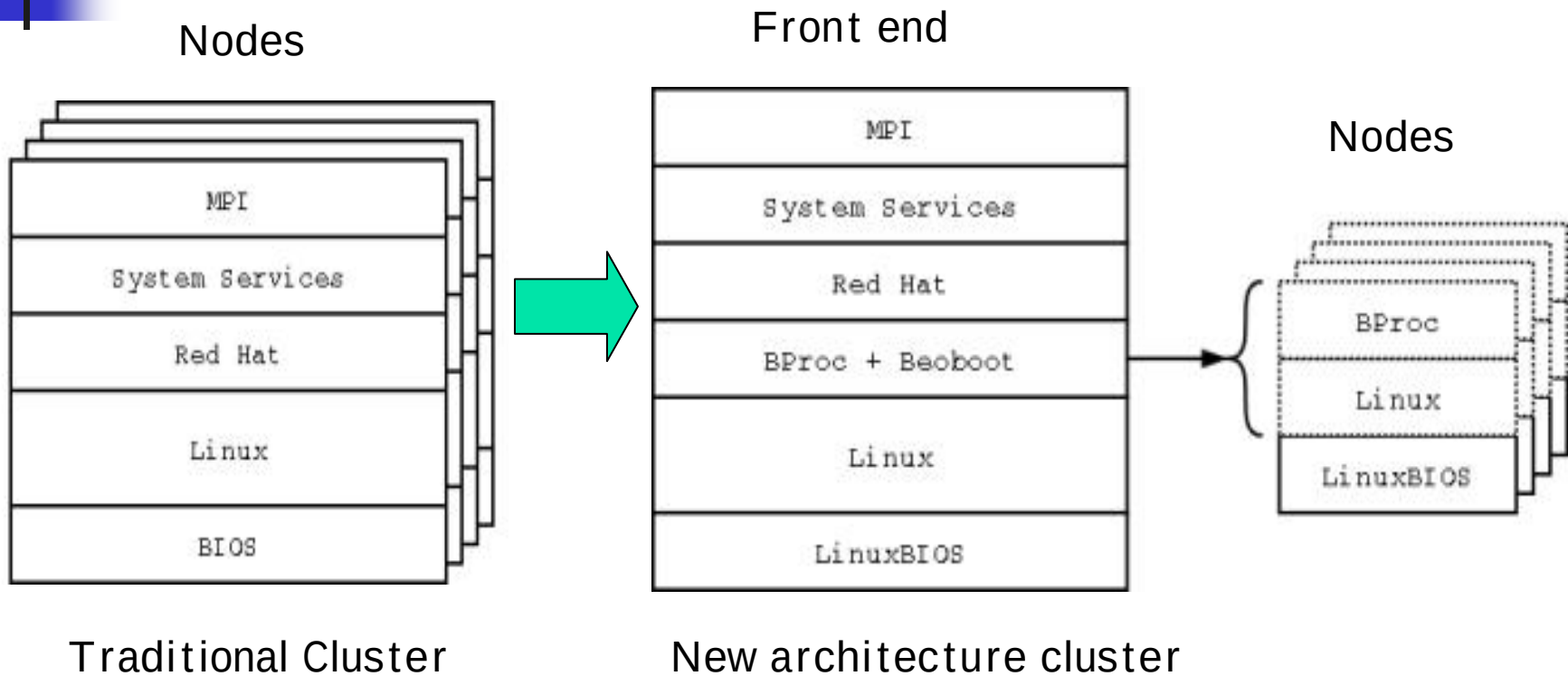
注：Pink完成于2003－2－14

- the largest single-system image Linux cluster in the world
- 1024-node (2048 processor) dual P4
- no local disk, no NFS-mounted root file system
- connected with Myrinet 2000.

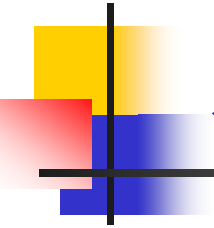
Pink

- node boot in under 10 seconds (LinuxBIOS)
- cluster boot *and upgrade* in under 2 minutes (LinuxBIOS, BProc)
- manageable nodes from power-on (LinuxBIOS, BProc)
- single system image of the entire cluster (BProc)
- fast process migration (BProc)
- fast cluster monitoring (Supermon)

Redesigning the Cluster Architecture



本小结结束



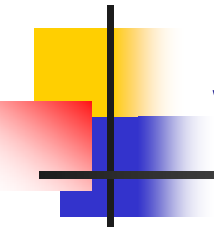
未来方向

- LinuxBIOS：
  - embedded computing platforms,
  - graphics hardware initialization,
  - booting other operating systems
- BProc：
  - process management,
  - single-system image size,
  - complete node-to-node process migration

未来方向—续

- Supermon ：
  - improving cluster-wide sampling rates
  - extracting hardware sensor information such as CPU temperature and fan speeds.

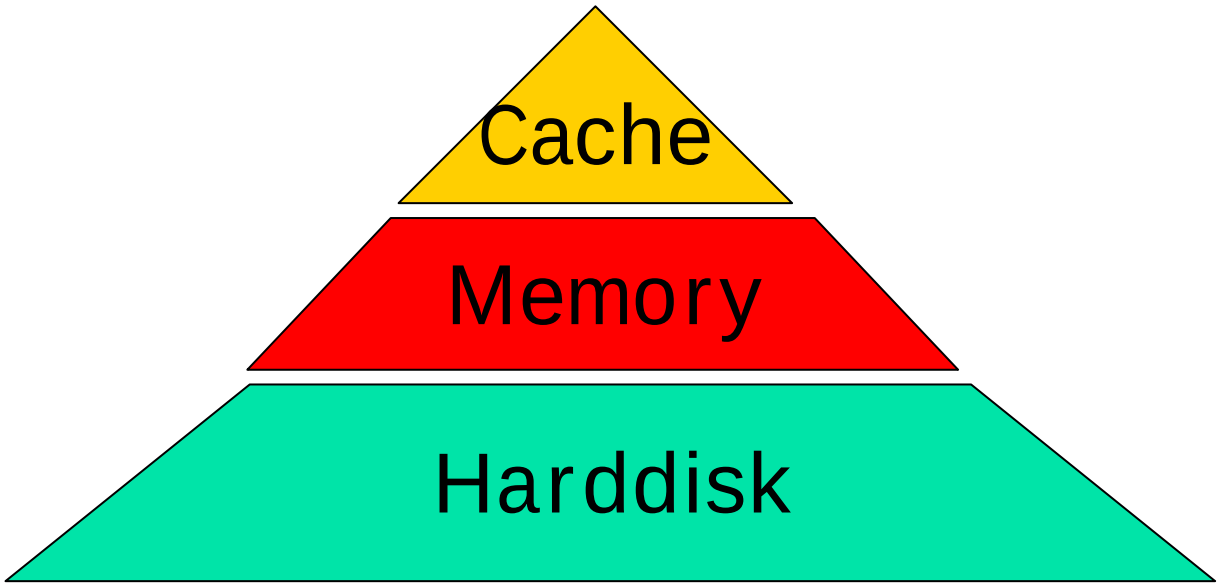
本小节结束



我的问题

- Supercomputer与Cluster的关系。
- “机群管理”是否是在承认现有硬件设备的基础上对资源？
- 何为“异构机群”？为什么要研究“异构机群”？

假想——把内存也拿出去



```
/*
 *      bootsect.S              Copyright (C) 1991, 1992 Linus Torvalds
 *
 *      modified by Drew Eckhardt
 *      modified by Bruce Evans (bde)
 *      modified by Chris Noe (May 1999) (as86 -> gas)
 *
 * 360k/720k disk support: Andrzej Krzysztofowicz <ankry@green.mif.pg.gda.pl>
 *
 * BIG FAT NOTE: We're in real mode using 64k segments.  Therefore segment
 * addresses must be multiplied by 16 to obtain their respective linear
 * addresses. To avoid confusion, linear addresses are written using leading
 * hex while segment addresses are written as segment:offset.
 *
 * bde - should not jump blindly, there may be systems with only 512K low
 * memory.  Use int 0x12 to get the top of memory, etc.
 *
 * It then loads 'setup' directly after itself (0x90200), and the system
 * at 0x10000, using BIOS interrupts.
 *
 * NOTE! currently system is at most (8*65536-4096) bytes long. This should
 * be no problem, even in the future. I want to keep it simple. This 508 kB
 * kernel size should be enough, especially as this doesn't contain the
 * buffer cache as in minix (and especially now that the kernel is
 * compressed :-)
 *
 * The loader has been made as simple as possible, and continuous
 * read errors will result in a unbreakable loop. Reboot by hand. It
 * loads pretty fast by getting whole tracks at a time whenever possible.
 */
```

back

```
/* inflate.c*/

/* Tables for deflate from PKZIP's appnote.txt. */
const unsigned border[] = {    /* Order of the bit length code lengths */
        16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};
const ush cplens[] = {         /* Copy lengths for literal codes 257..285 */
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31, 35, 43, 51, 59,
    67, 83, 99, 115, 131, 163, 195, 227, 258, 0, 0};
                    /* note: see note #13 above about the 258 in this list. */
const ush cplext[] = {         /* Extra bits for literal codes 257..285 */
        0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5,
    5, 5, 0, 99, 99}; /* 99==invalid */
const ush cpdist[] = {         /* Copy offsets for distance codes 0..29 */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193, 257, 385,
    513, 769, 1025, 1537, 2049, 3073, 4097, 6145, 8193, 12289, 16385,
    24577};
const ush cpdext[] = {         /* Extra bits for distance codes */
        0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10,
    11, 11, 12, 12, 13, 13};
```

const	Read-only text segment
static	RAM

back

BProc

The Beowulf Distributed Process Space

BProc 能够在整个机群中为进程提供一个单一的进程空间。进程在前台被创建，并由BProc的进程迁移机制转向某个结点。在机群结点上运行的任务可以在前台被监视到 (via ps and the like)并且被控制 (via standard UNIX signals)。

back

Supermon

High speed cluster monitoring ：

- 6000 samples/s with supermon (275 RPC)
- 采用基于符号表达式的数据协议，同时具有数据传输与数据监控功能；
- 系统故障汇报:

((portDown (switch m1c1) (slot 35) (XBar16 11) (XBarPort 172) (port 11) (value 1))

back

谢谢参与！