HP World/Interex 2002

Linux Partitions and Boot Loaders

Chris Cooper
(734) 805-2172
chris_cooper@hp.com

George Vish II
(404) 648-6403
george_vish@hp.com

hp education services
education.hp.com

Disk Partition Management Options (LVM)

Version A.00

U2794S Module 23-1 Slides

Linux LVM

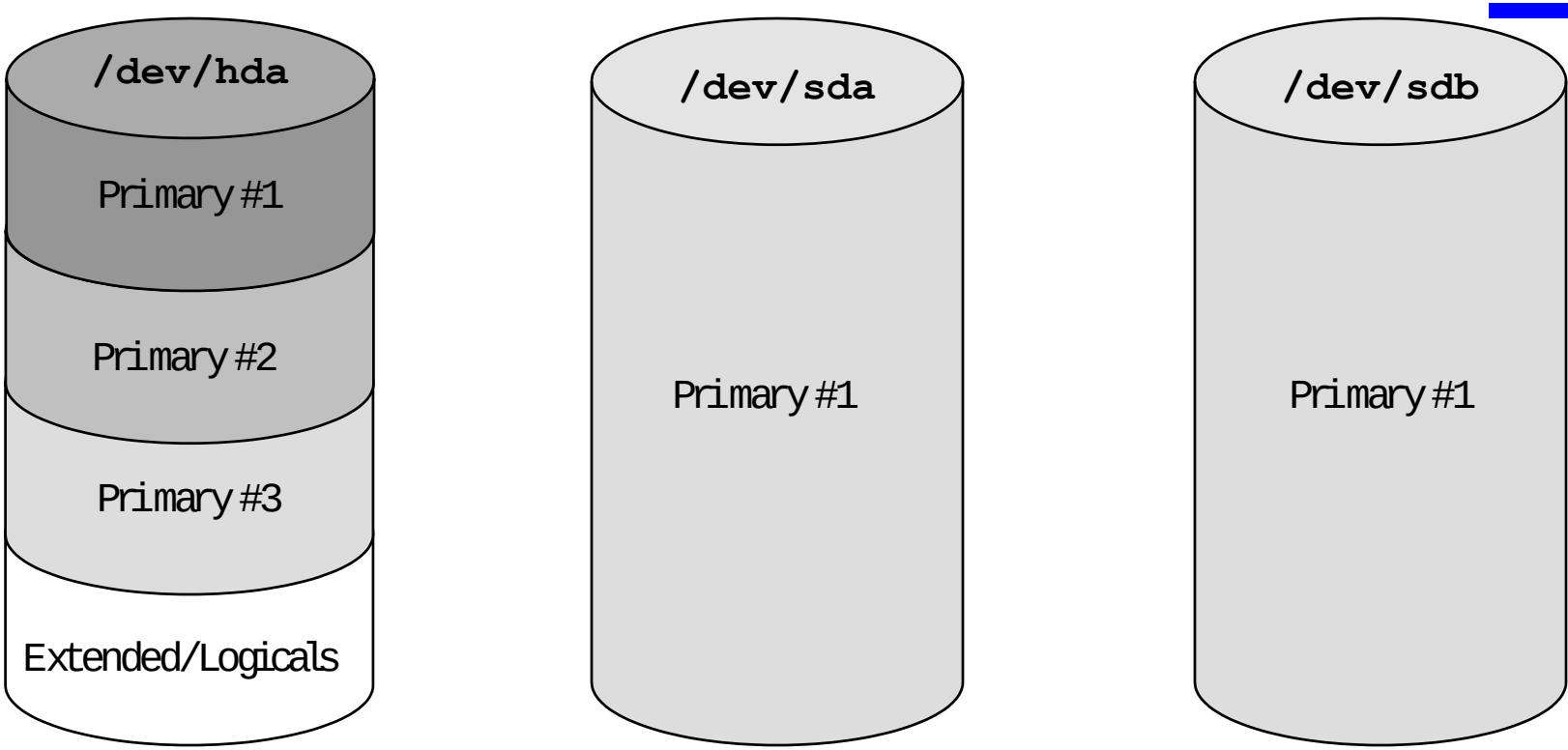
- Disk partitioning allows a single hard disk to be divided into up to 15 different sections. Each section can be used for any of a number of purposes. The Logical Volume Manager allows the Linux operating system to combine one or more partitions into a volume group, which may then be divided into logical volumes.

- There is a wide range of kernels where LVM is available. In Linux 2.4, LVM will be fully integrated. From kernel 2.3.47 and onwards, LVM is in the process of being merged into the main kernel.

- The LVM implementation for Linux strongly resembles the LVM in Hewlett-Packard's HP-UX OS.

- The 2.4 Linux kernel release also contains a software RAID capability.

Disk Management



For the purpose of our discussion, let us assume that the system has three hard drives attached,

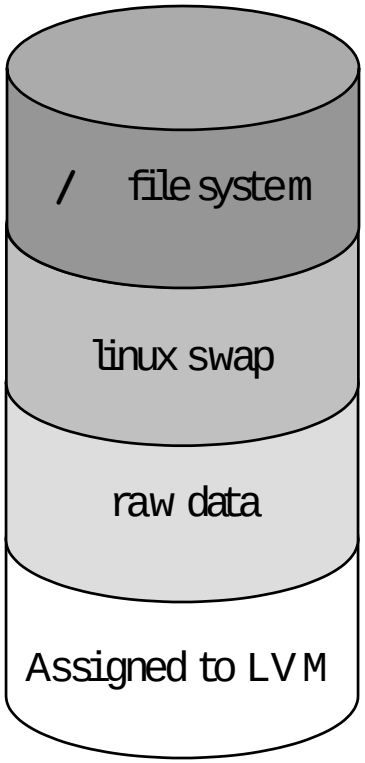
- One on an IDE interface, which contains the Linux OS and swap
- Two more connected to an SCSI controller, currently unused

Disk Partitioning

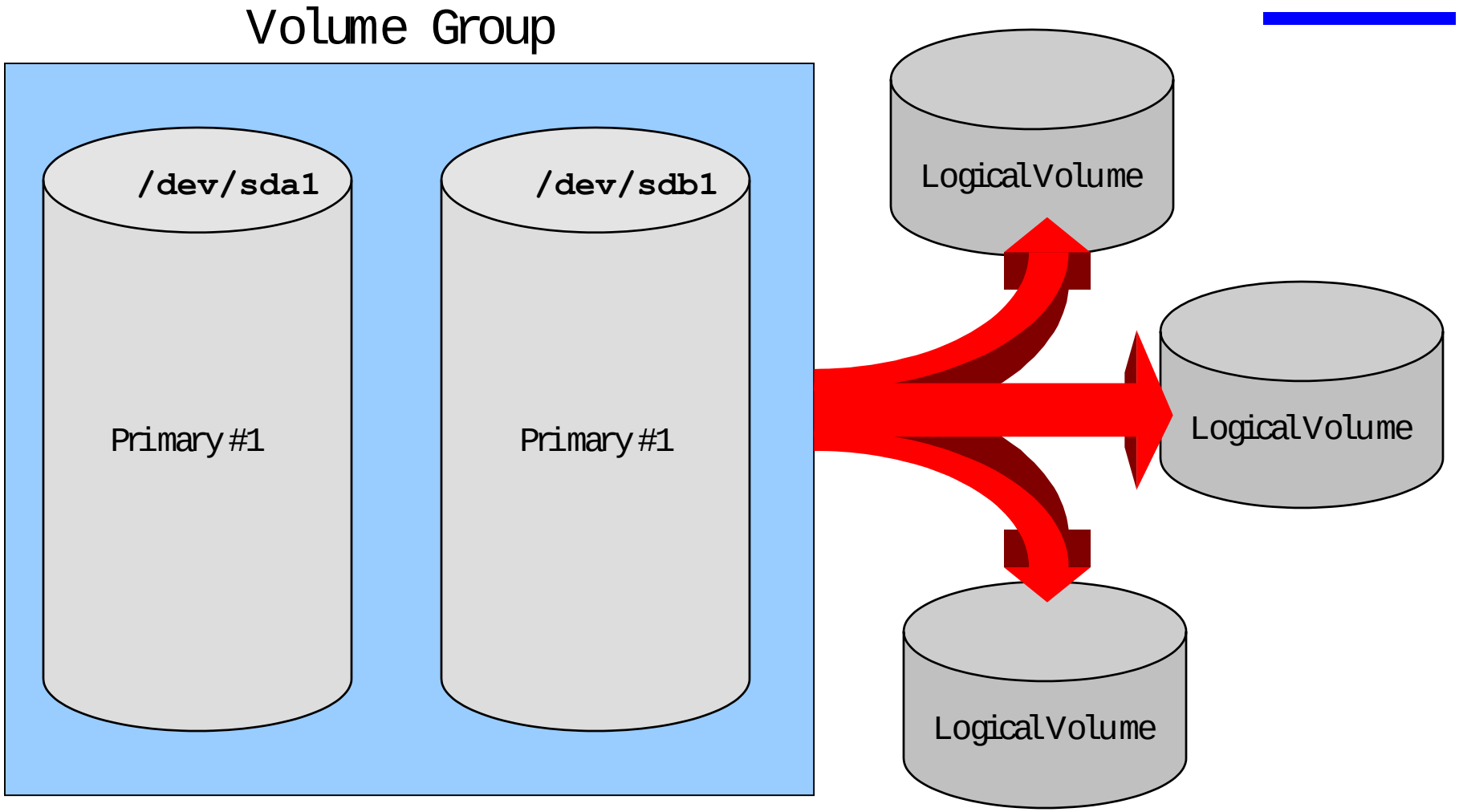
- Each Linux disk can have one or more partitions, created with **`fdisk`** or equivalent (with a maximum of 15 partitions, 3 primary and 1 extended, containing up to 12 logical).

- Each partition can be:
  - used as a file system
  - used as swap space
  - used for raw application data
  - assigned to the Logical Volume Manager

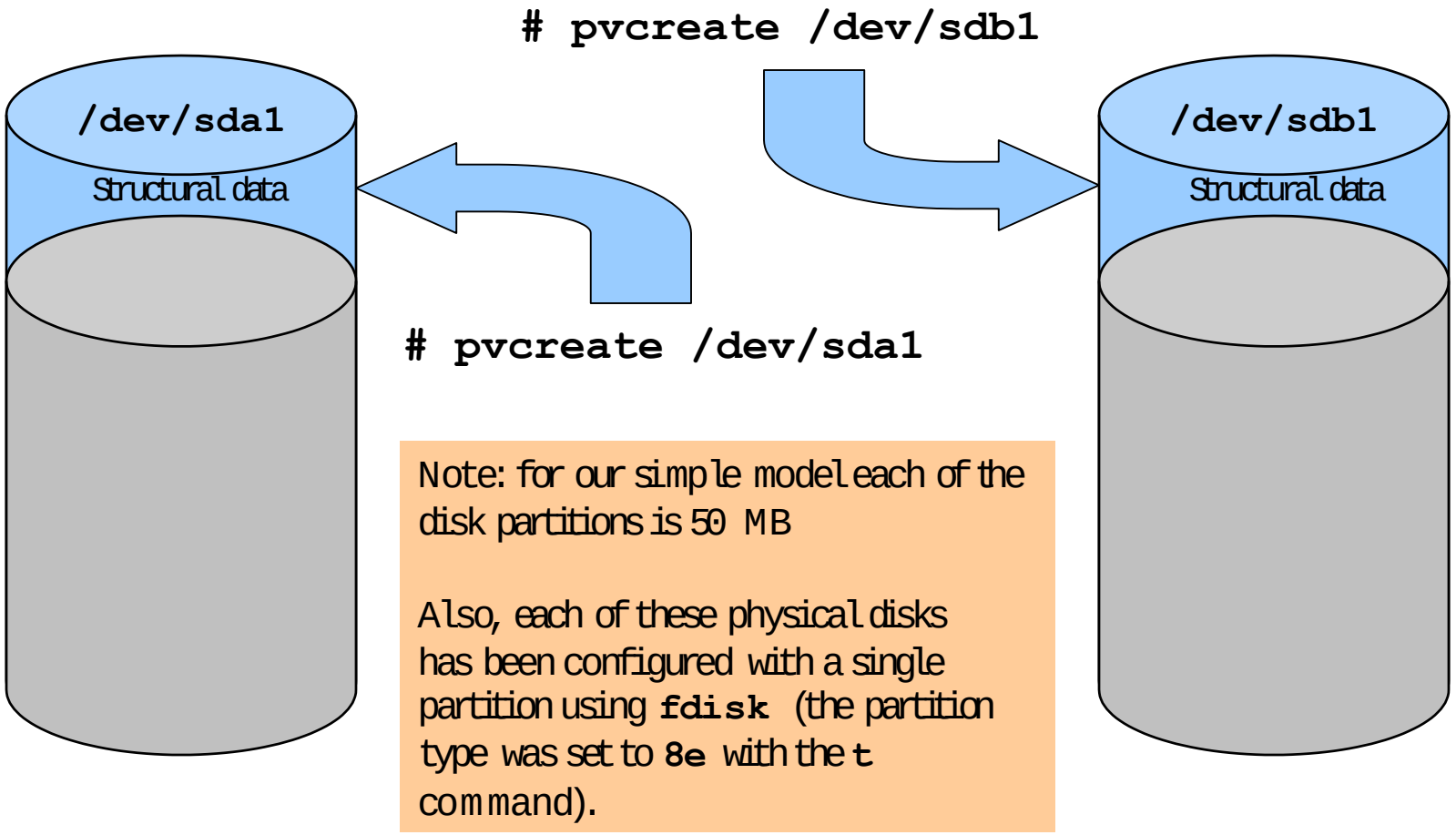
- You may examine your system's current partitioning buy running either **`parted`** or **`fdisk`** from the command line. (Caution! Make no changes to your current configuration unless you are very sure about what you are doing!)



LogicalDisk Management



Preparing the Disks for LVM

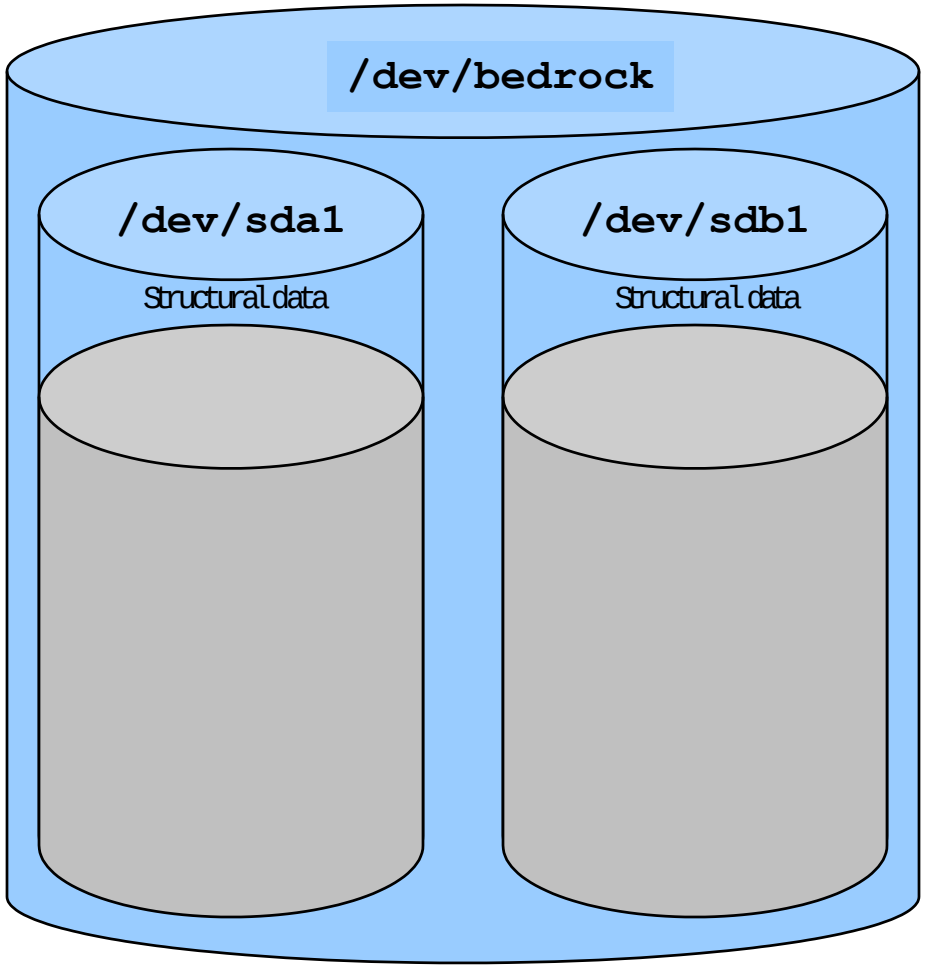


Note: for our simple model each of the disk partitions is 50 MB

Also, each of these physical disks has been configured with a single partition using **fdisk** (the partition type was set to **8e** with the **t** command).

Creating the Volume Group

```
# vgcreate bedrock /dev/sda1 /dev/sdb1
```



Now that our volume group has been created, there is a total of 96 MB of space in the extent pool:

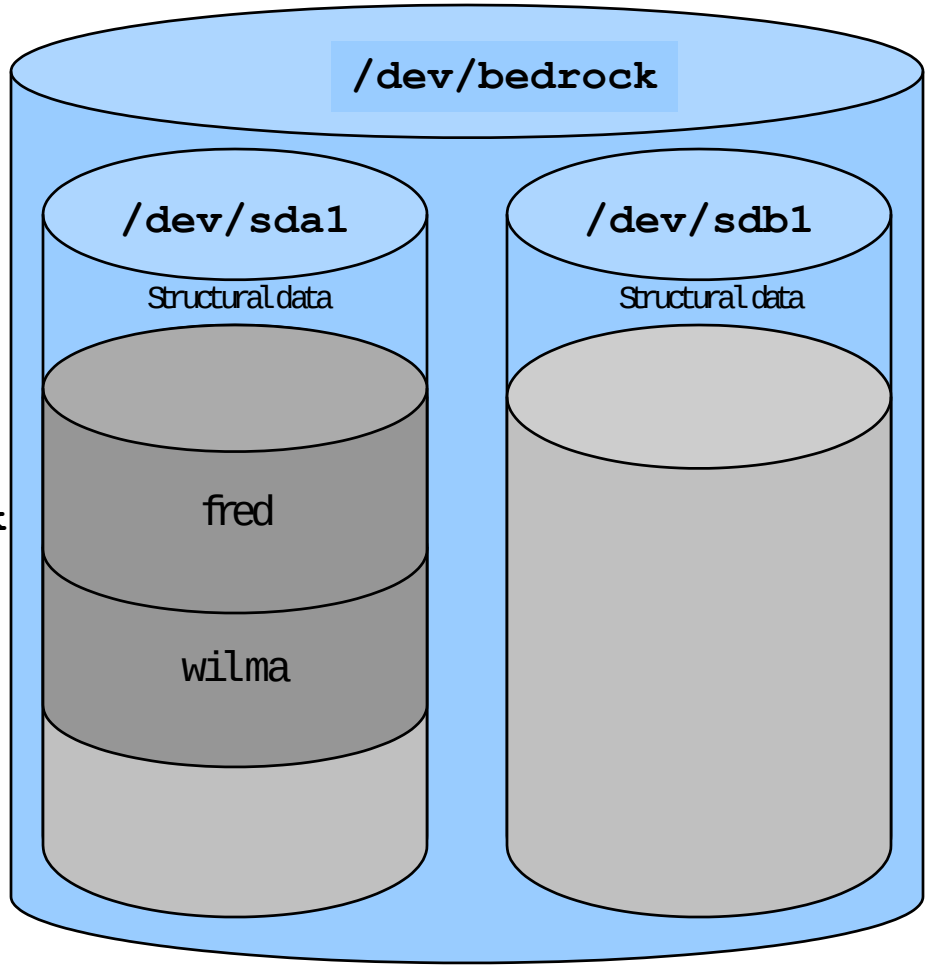
50 MB + 50 MB = 96 M ??

Creating Logical Volumes

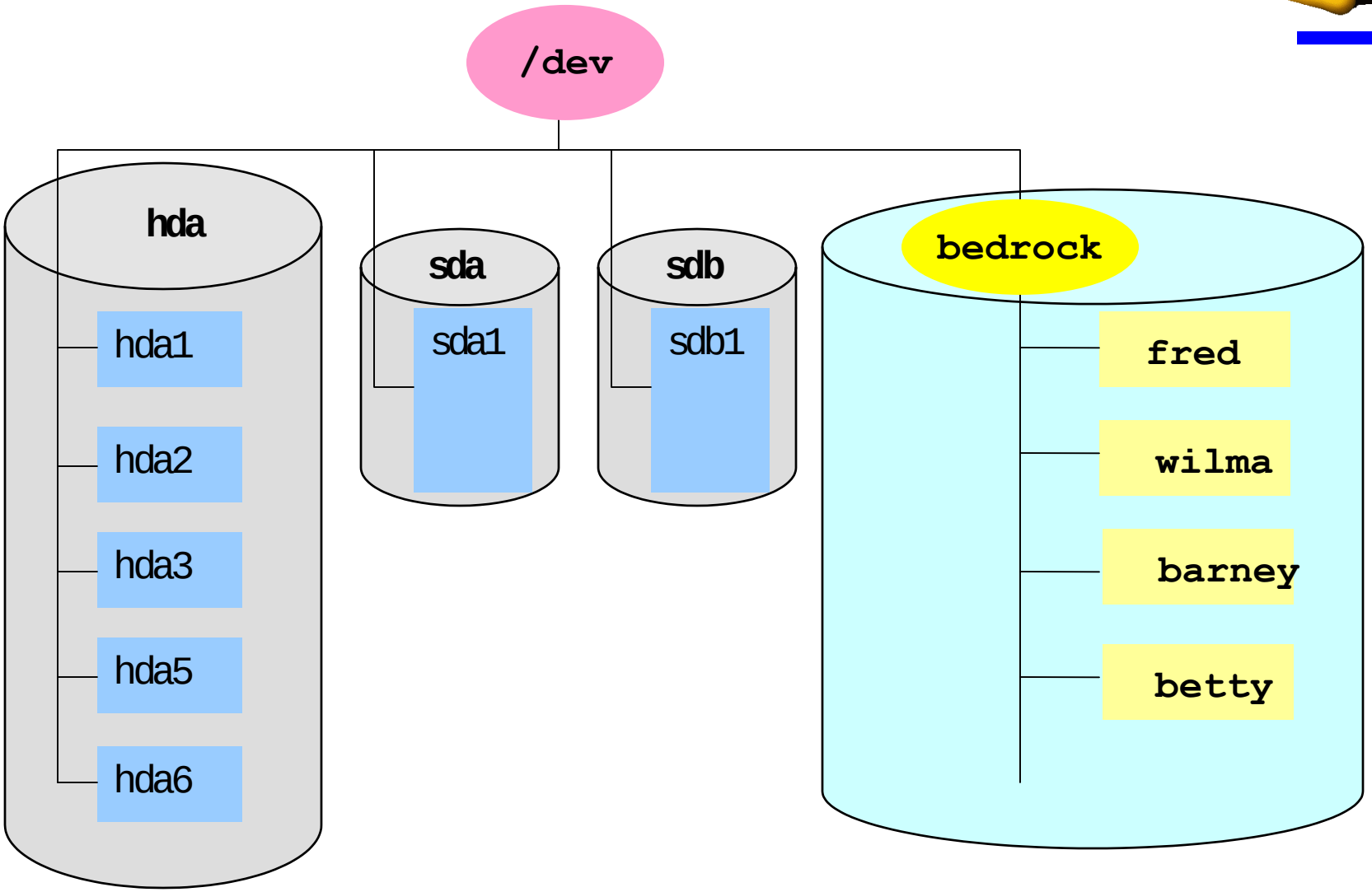
Now, let's create two logical volumes from the extent pool in "**bedrock**."

Note: Different options are used (but the end results are identical).

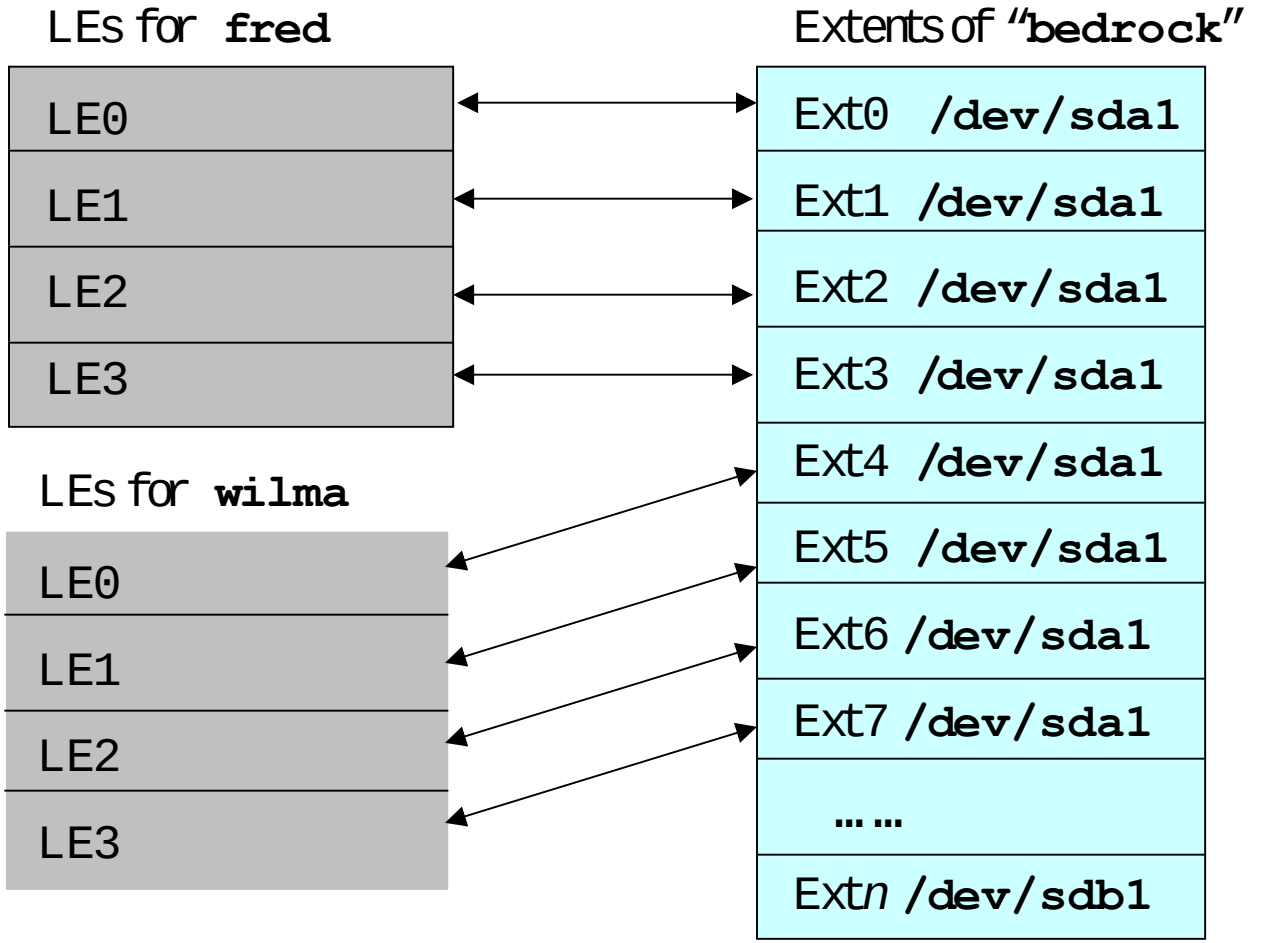
```
# lvcreate -L 14M -n fred  bedrock
# lvcreate -l 4 -n wilma  bedrock
```



LVM Device Files



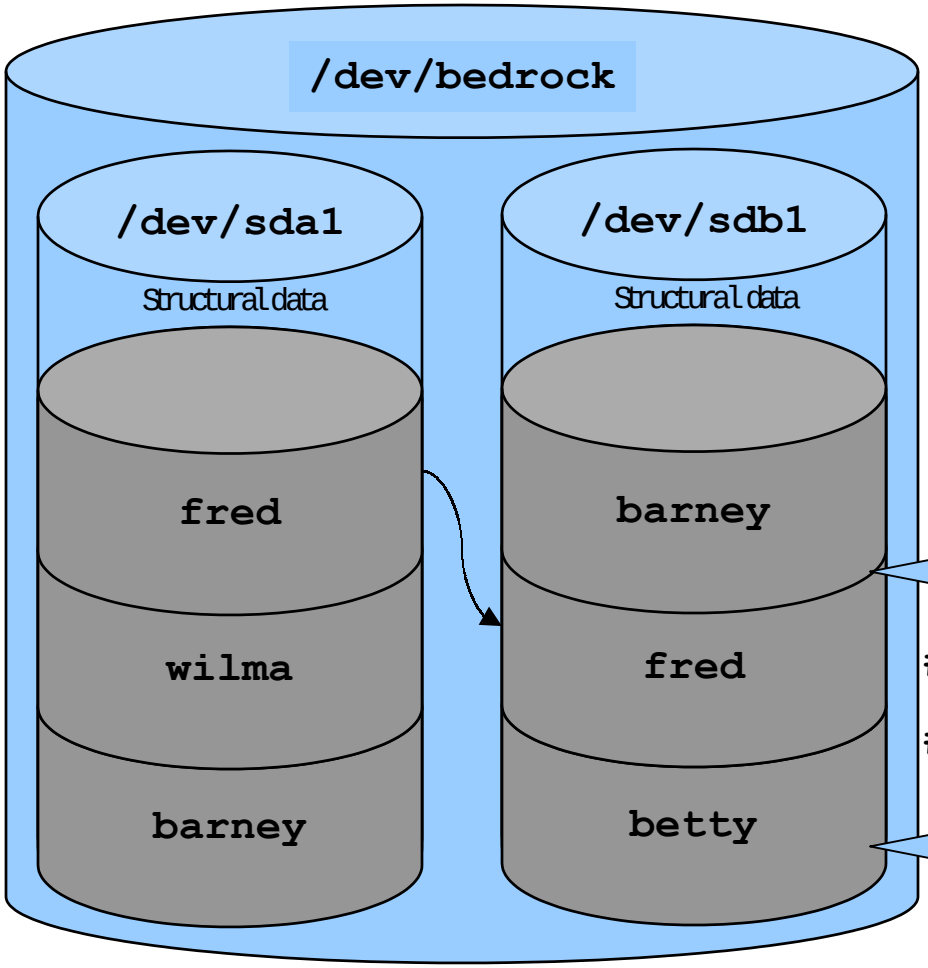
LVM Extents



Logical extents are remapped to physical extents by the LVM kernel module.

How would extent size affect these translation tables?

Extending a Logical Volume



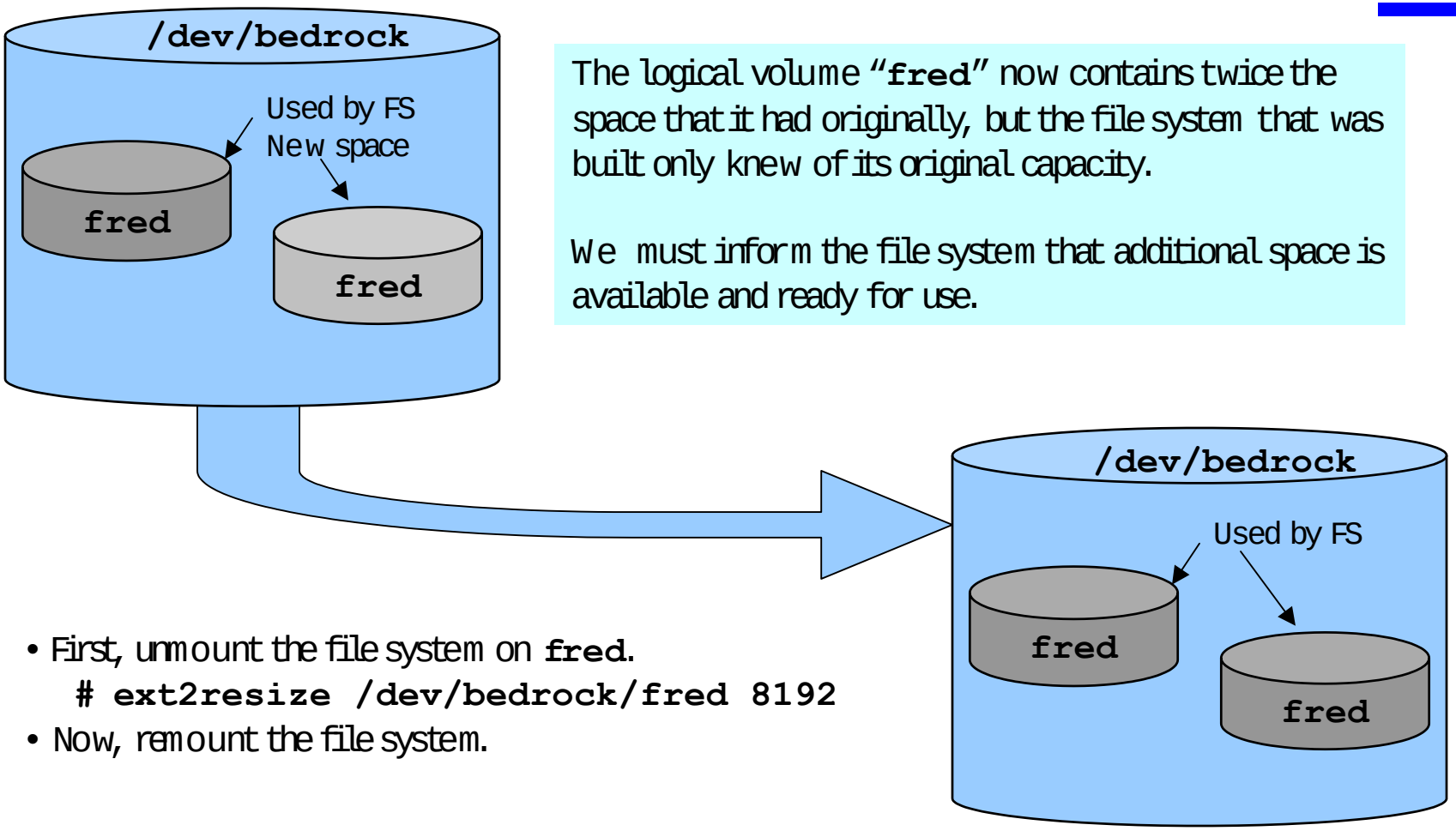
Since our last visit, a third logical volume named "**barney**" has been created and initially sized to 32 MB. Note its assigned extents.

Next we will increase the "**fred**" volume and add a fourth volume named "**betty**."

What if "**fred**" contained a file system before we extended the volume?

```
# lvextend  -L+16M /dev/bedrock/fred
# lvcreate -l 4 -n betty  bedrock
```

Resizing a File System



The logical volume "**fred**" now contains twice the space that it had originally, but the file system that was built only knew of its original capacity.

We must inform the file system that additional space is available and ready for use.

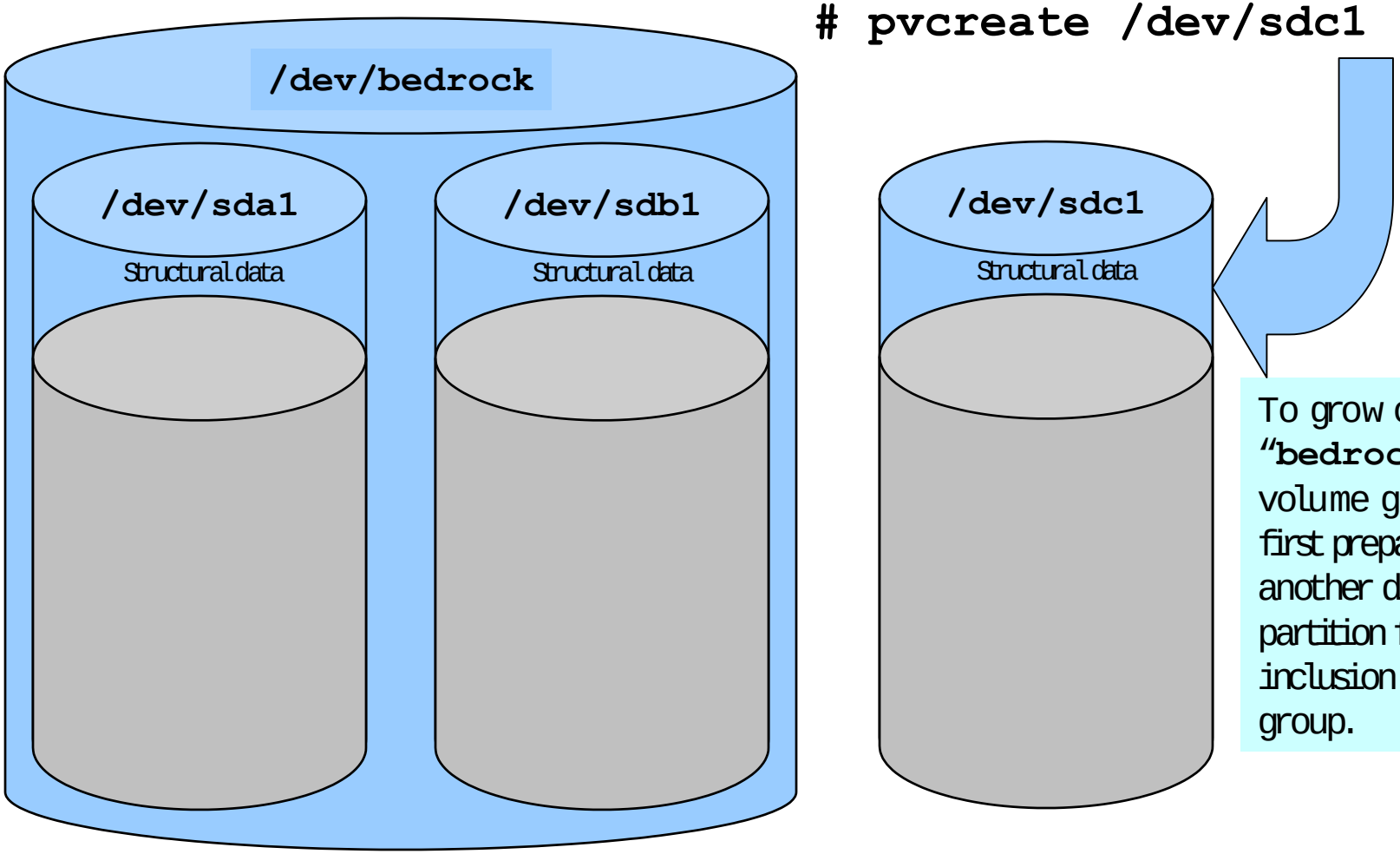
- First, unmount the file system on **fred**.

  # ext2resize /dev/bedrock/fred 8192

- Now, remount the file system.

Adding a Disk

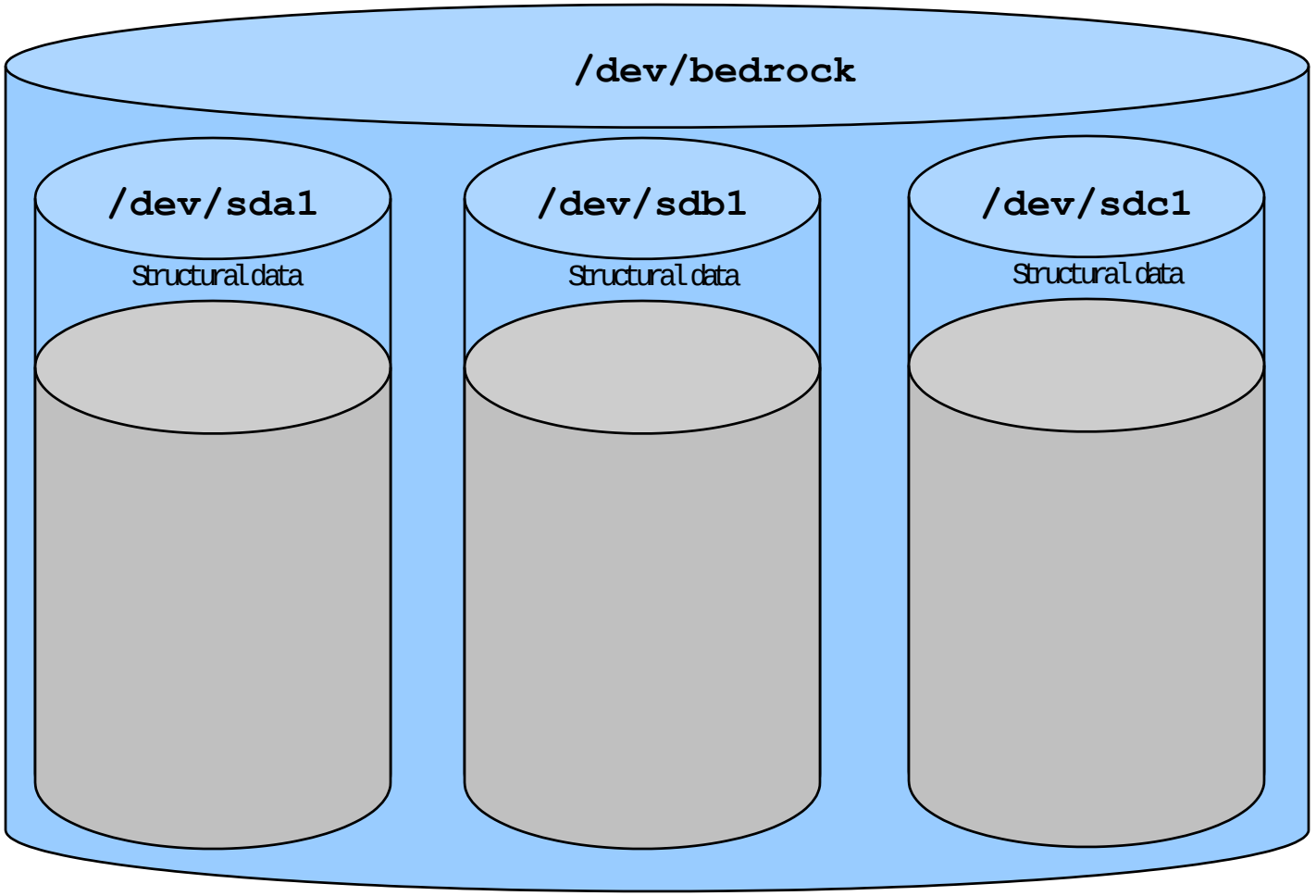
```
# pvcreate /dev/sdc1
```



To grow our "**bedrock**" volume group, we first prepare another disk partition for inclusion in the group.

Growing the Volume Group

```
# vgextend  bedrock /dev/sdc1
```



Then we extend the volume group to include the new **pvcreate**'d partition.

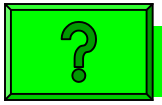
Now we can create new or extend existing logical volumes.

Additional LVM Commands

- To examine configuration information:
  - **vgdisplay**
  - **pvdisplay**
  - **lvdisplay**
- To activate/deactivate a volume group:
  - **vgchange**
- To move a volume group:
  - **vgexport**
  - **vgimport**
- And many more:
  - **vgcfgrestore**, various **extend**, **reduce**, and **delete** commands

LVM's Other Functionality

- There are several other configuration options for LVM including:
  - Volume stripping
  - Making snapshots for consistent backups
  - Mirror volumes



For additional information -> http://www.cistina.com/products_lvm.htm

hp education services
education.hp.com

Boot Loaders
LILO and Grub

Version A.00

U2794S Module 24 Slides

LILO and the Boot Process

- Boot sector either contains or loads the LInux Looader.
- Each partition on a hard drive has a boot sector:
  - Frst boot sector on entire disk is called MBR.
  - LILO can be installed as the MBR, or can be loaded from the boot sector of the active partition.
- LILO loads the Linux kernel.
  - Configured by **/etc/lilo.conf**.
  - Allows you to set hardware parameters at **boot** prompt.
- The kernel starts the **init** process.
  - Image statement in **lilo.conf** identifies default kernel.
- Components:
  - **/sbin/lilo** (map installer)
  - **/etc/lilo.conf** (configuration file)
  - **/boot/map** (map file)
  - **/boot/boot.b** (boot program)

lilo.conf, Global Options

- Global options
  - boot=boot_device
  - default=name
  - delay=tsecs
  - install=file
  - map=map_file
  - password=a_password
  - restricted
  - timeout=tsecs

lilo.conf, Image Options

- Image options
  - alias=name
  - image=pathname
  - label=name
  - password=a_password
  - table=device

- You can boot from different images. The table=device option is for non-Linux operating system.

lilo.conf, KernelOptions

- Kernel options
  - **inetrd**=filename
  - read-only
  - root=root_device
  - vga=mode
  - number
- A list of available modes for your video card can be obtained at the LILO boot prompt
  - **boot:** linux vga=ask

Simple `lilo.conf`

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
delay=50
image=/boot/vmlinuz-2.2.14-5.0
  label=linux
  root=/dev/hda1
  read-only
```

Additional `lilo.conf` options

```
boot=/dev/sda3
prompt
timeout=10
image=/boot/vmlinuz-2.2.5-15
  label=linux
  alias=1
  root=/dev/sda3
  read-only
  password=secret
  restricted
```

Safe `lilo.conf`

```
boot=/dev/hda3
map=/boot/map
install=/boot/boot.b
prompt
image=/boot/vmlinuz
  label=linux
  root=/dev/hda3
  read-only
image=/boot/vmlinuz-2-2.14-5.0
  label=backup
  root=/dev/hda3
  read-only
```

Interactive LILO arguments

- At the LILO boot prompt, you can change the normal startup procedure or supply the kernel with hardware parameters.
- Examples:

```
lilo boot: linux  <single or 1>
lilo boot: linux  <run_level>
lilo boot: linux rescue
lilo boot: linux root=device
lilo boot: linux vga=mode
lilo boot: linux init=/bin/sh
lilo boot: linux ro
```

Dual Booting

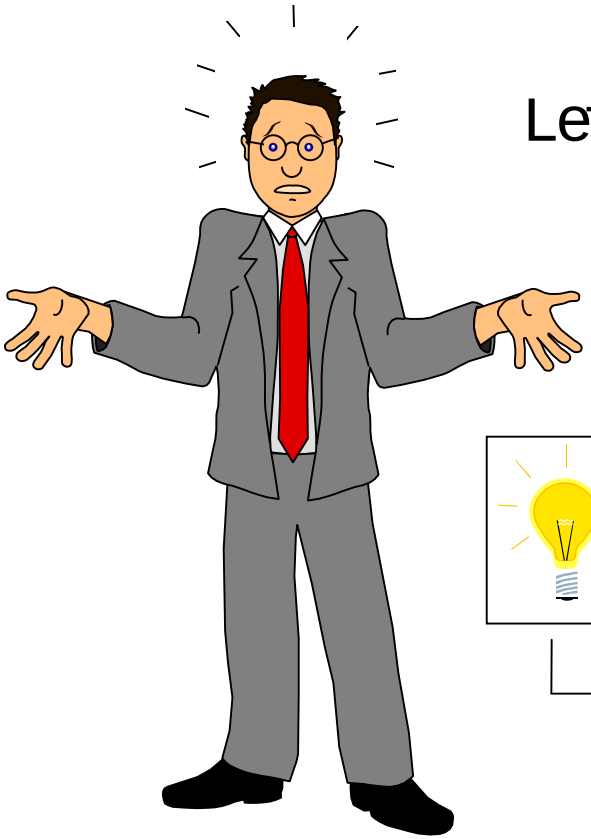
- A PC can be configured with more than one operating system.
- There are several ways to accomplish dual booting:
  - Linux
  - Linux and Windows NT
  - Linux and Windows 95/98
  - Linux with Windows NT and Windows 95/98 and 2000

- FIPS: A program that allows you to repartition your disk without destroying data.

Dual Boot with Windows NT

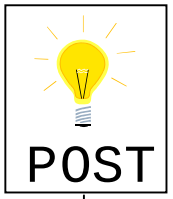
- Scenario #1:
  - Windows NT installed first and Linux installed second.
  - Use Windows NT's loader to load both operating systems, NT's loader installs to the MBR.
- Scenario #2:
  - Windows NT installed first and Linux installed second.
  - Use Linux loader to load both operating systems, Linux's loader installs to the MBR.

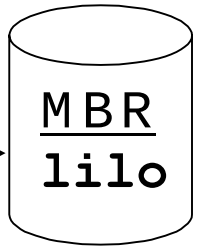
Problems at Boot!



Let's revisit startup.

Power On





Recognizing a Boot Failure

- Where is the problem?
- How far through the boot process does the system get?
  - Hardware failure?
  - Partition misconfigured?
  - File system corrupted?
  - Error in startup scripts?
- Check **/var/log** for error messages.

- Kernel Boot Messages
  - An exhaustive series of messages is printed to the console at boot time.
  - Messages are stored in **/var/log/dmesg**.
  - It is useful for debugging or detecting boot problems.

Hardware Failure

- Power on: No beep codes, system does not come up.
  - Power supply.
- POST: Beep codes, system does not come up.
  - Hardware failed boot power-on self test.
- OS Loading: Boot or disk errors.
  - Corrupted media or file system.
- LILO hangs:
  - Configuration errors in LILO or partition table.
- Kernel boot: Various errors.
  - Read **/var/log/dmesg**.

LILO Error Codes

No display	**LILO** not installed or boot sector not active
L error	Media failure or wrong disk geometry
LI	Media failure or **/boot/boot.b** not mapped correctly
LIL	Media failure or wrong disk geometry
LIL?	Wrong disk geometry or **/boot/boot.b** not mapped correctly
LIL-	Corrupt descriptor table

Crash Dumps

- A crash dump is a core image stored to the dump area upon system panic.
  - The dump area is the current working directory.
- Core dumps are usually generated by segmentation faults.
- The GNU debugger (**gdb**) can be used to analyze a core dump — but only if all running programs were compiled with debugging code turned on, which is rarely the case outside development environments.
- To effectively eliminate core dumps, set the core dump size to **0**:
  - **bash shell: ulimit -c 0**
  - **tcsh shell: limit coredumpsize 0k**

Backing Up and Restoring the MBR

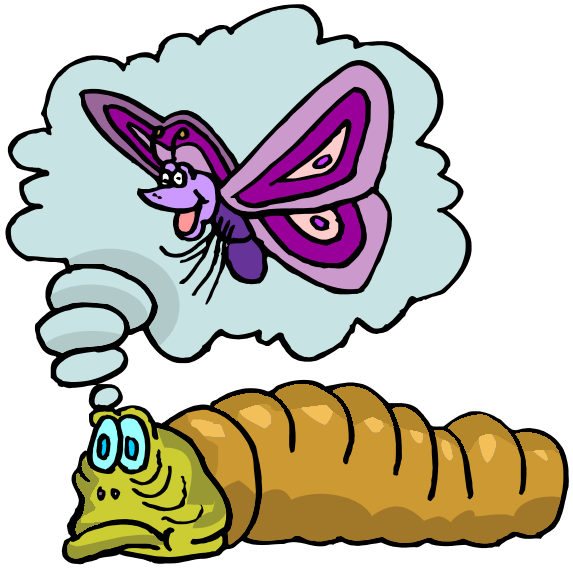
- Boot image copies are stored in:
  - **/boot/boot.0300** (for IDE drives)
  - **/boot/boot.0800** (for SCSI)
- Restore the Master Boot Record with:
  - **dd if=/boot/boot.0300 of=/dev/hda bs=512 count=1**
- Restore the DOS boot record with:
  - **fdisk /mbr**

New from GNU -- GRUB !

- GRUB the GRand Unified Bootloader command shell
- GRUB is a "Multiboot Specification" compliant boot loader
- Features "chain loading" capabilities
- Supports loading a wide variety of "free" operating systems
- Is offered as an option by some recent Linux distribution installers

"Of things to be"

As the lowly caterpillar dreams of becoming a butterfly so the boot loader dreams of becoming an OS ! (or as one of the GRUB developers puts it; "from maggot to house fly!")



GRUB may either be installed "natively" (into the MBR of the boot disk) or it may be chain-loaded by another boot loader.

Use caution if you choose to install it natively as this will erase any pre-existing boot loader !

The GRUB shell

- Unlike other boot loaders, GRUB offers an interactive shell interface.
- There are many commands and options which may be used during the boot process. These can greatly increase flexibility and control.
- There are many security, default, and password configurations available

For information on grub try:

```
# info grub
```

(be prepared for a sizeable document)

GRUB device naming conventions

The GRUB loader uses the following naming convention for devices:

(fd0) <-- All device names must be enclosed in (...)

(fd0) <-- Floppy Disk

(fd0) <-- The drive number (counting form 0)

For a hard disk:

(hd0,1) <-- would be Hard Disk #0, 2nd partition (counted from 0 !)

Note! GRUB does not differentiate SCSI from IDE drives, it merely counts the drive numbers from zero ! (boot device drive order is determined by your BIOS, in most cases IDE drives precede SCSI drives but that it not a given !)

Device specification

- In addition to specifying the device name you must also tell grub what type of device you are attempting to access.

- The most common specifier is "root". In order to access logical partition # 6 on your first disk the specification would be: **root (hd0,5)**

- To make life a bit easier GRUB provides argument completion, simply enter "root <TAB>" and GRUB will display the list of drives, partitions, or file names to choose from.

Creating a GRUB boot floppy

You should always create a GRUB boot floppy !

```
# cd /user/share/grub/i386-pc
# dd  if=stage1  of=/dev/fd0  bs=512  count=1
1+ 0 records in
1+ 0 records out
# dd  if=stage  of=/dev/fd0  bs=1  seek=1
153 + 1  records  in
153 + 1  records  out
#
```

Caution! This will destroy all current data on the floppy

Native installation

To locate GRUB on the MBR of a disk you may use **grub-install**

If your kernel image is under the "/" directory you will only need to add a single argument:

```
# grub-install   /dev/hda     <- Linux naming
          or
# grub-install '(hd0)'        <- GRUB naming
```

If the kernel is located under "/boot" then use the following:

```
# grub-install --root-directory=/boot /dev/hda
```

The menu configuration file

To utilize a menu with the GRUB shell place a grub.conf file under the boot directory (ie. /boot/grub.conf)

```
# Sample boot menu configuration file
# By default boot the first entry
default 0
# Boot automatically after 30 sec.
timeout 30
title GNU/Linux
kernel (hd0,1)/root/vmlinuz root=/dev/hda2
```

Remember that GRUB is actually an interactive boot loader shell, the grub.conf file is run as a script !

GRUB and LILO

GRUB

- Is Multi-Boot Compliant
- re-reads it's configuration file when it runs
- Understands many file system layouts
- Works in multi-boot scenarios
- Flexible but many options to master
- Starting to be offered as a choice during installs

LILO

- Is specific to Linux
- Must be re-installed if the configuration is changed
- Must have fixed pointers to kernel images
- Works in Dual boot scenarios
- Is the established "norm" for most Linux installations