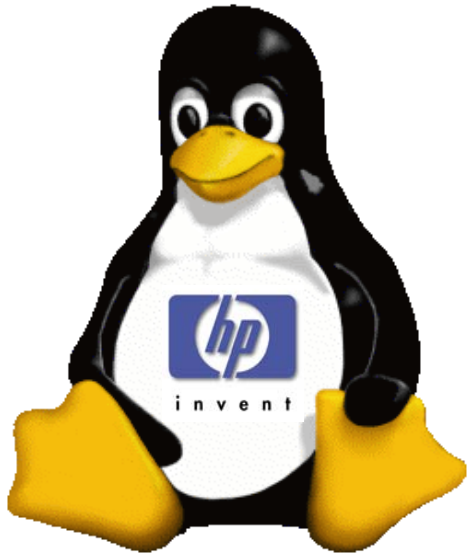
HP World/Interex 2002

Linux Kernel Configuration and Patching

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

Building a Linux Kernel

Version A.00

U2794S Module 21 Slides

Building a Kernel

- Get the source image you wish to build (new release, patch bundle, modules, and so on).
- If your current kernel's source tree is in **/usr/src/linux-2.4**, move it to **/usr/src/linux-V.R.p** (using current Version.Release.patch numbers).
- Unpack the new source image into **/usr/src/linux.V.R.p** (using new Version.Release.patch numbers). You may want to symbolic link this tree to **/usr/src/linux**-2.4
- In the new **/usr/src/linux.V.R.p**, be sure to read the **README** file!

Building a Kernel (Continued)

- Now that you have a new source tree first make sure there are no stale config or .o files laying about (this may seem silly but don't skip this step!)
  Run **# make mrproper** to accomplish this task

- Use **# make config**, **make menuconfig**, **make xconfig**, or **make oldconfig** to select your kernel configuration options interactively.
  (Be careful! There are more than 1000 kernel options.)

- Use the **# make dep** and **# make clean** macros next

- Now it is time to build. You can choose to build the kernel or modules and to install either one or both following the build.
  # make bzImage modules modules_install install

- Update your **lilo.conf** file to allow booting both the new kernel and the old kernel. Reboot and test your new kernel!

Configuration Options

Regardless of the utility you choose to help you with the build your **.config**, you will have to make decisions about your kernel's content and features.

Modules can be statically linked into the kernel, or they can be defined as dynamically loaded when needed.

Let's examine some of the options. (The options we will discuss are a common subset of options. There are well over 1000 configuration options!)

Kernel Modules

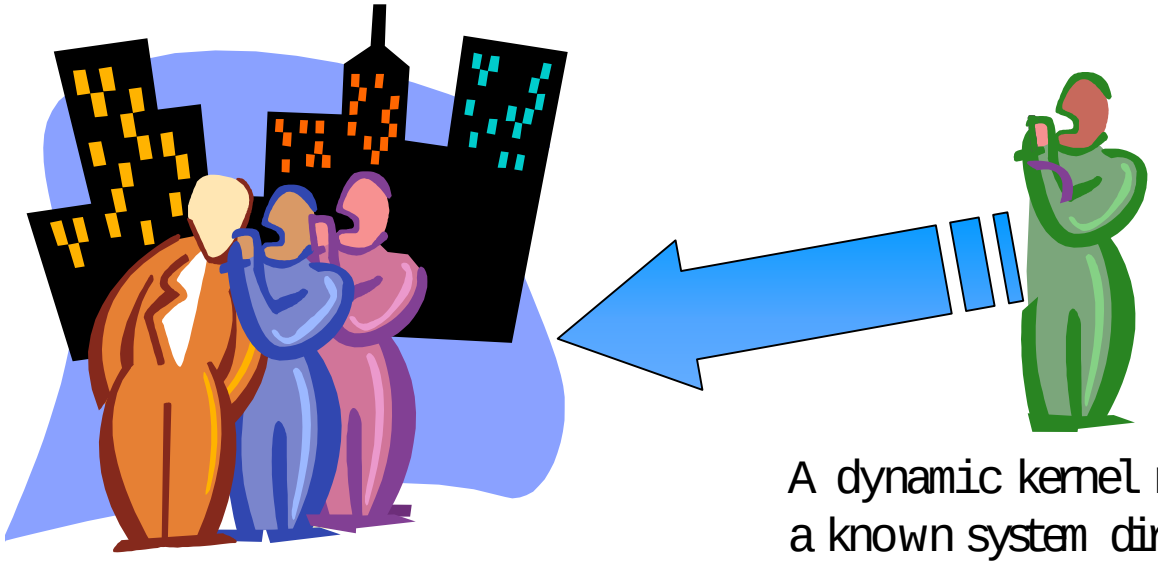
- Linux uses a monolithic kernel:
  - It's simpler to manage than a micro kernel.
  - But it can become very large.

- A modularized kernel can be smaller:
  - Only "bare necessities" are included in the base kernel.
  - Other functions may be added (by **modprobe** and **insmod**) or removed (by **rmmod**) as needed.
  - The **kerneld** daemon can automate additions and removals.

Working Together

Once loaded, all modules in the kernel must work together harmoniously. Each dynamic module must be configured, and its image stored in a location known to the kernel. When the module is loaded, its code is brought into memory and must be added to the kernel's symbol table to allow transparent interoperation with the rest of the kernel. This allows what has become a fairly complex mix of features and services to be managed on an easier modular basis. Almost all components of the kernel have been converted to modules.

Static and Dynamic Kernel Modules



A static kernel module is permanently linked and loaded into the kernel's running image. The big plus here is that when the code is needed, it is ready to go at a moment's notice. The down side is that it uses system memory on a permanent basis.

A dynamic kernel module is safely held in a known system directory and is loaded into system memory only if it is needed. The plus is that system resources are conserved until the module is loaded. The down side is that it takes longer for the kernel to use a dynamic module initially, because the module must be loaded.

New 2.4 Features

With the release of the Linux 2.4 kernel, several new features have been introduced, this is a partial list of things to look for:

- Resource management subsystem
- Remapped device files
- Many changes to the **/proc** interface
- Support for 4.2 billion users
- Memory limit 16 GB
- I2O/PCI enhancements
- Improved PnP support
- Improved Parport I/F
- USB support
- Extended joystick
- IA64, S/390, PA-Risc, and SuperH ports
- Shared memory VFS
- Logical Volume Manager
- Single buffer file caching
- Raw I/O device
- Large-file file systems (files > 2GB)
- UDF and UFS file systems
- SCSI subsystem
- Module Access Control
- Network Address Translation (NAT)
- DECNet and ARCNet support
- Direct rendering manager
- Voice synthesizer support
- Kernel-level HTTPD
- Documentation/DocBook and docgen

Getting the Source

Via **ftp** from: **ftp://ftp.kernel.org/** or any mirror site.

For general information on Linux:

http://www.linux.org/

http://slashdot.org

For general information on the Linux kernel:

http://www.kernel.org/

Unpacking the Source

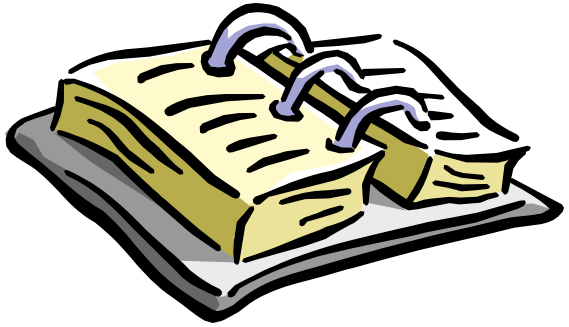
- Move **`/usr/src/linux`** to **`/usr/src/linux.V.R.p`**
  - For example: ***`V.R.p`*** --> 2.4.7
  - Major kernel version number 2, minor release 4, patch 7
- Unpack new source using
  - **`tar zxpvf linux-`*`V.R.p`*`.tar.gz`**
- Read the **`README`** file.
- Then read **`README`** again!

make arguments

- Change directory to **/usr/src/linux-2.4**

- Set a unique **EXTRAVERSION** name
  make mrproper to clean the source tree and rid it of stale .o files
  make config or menuconfig or xconfig or oldconfig
  make dep
  make clean
  make bzImage
  make modules
  make modules_install
  next
- Next move the new kernel into /boot (make sure you save the old first!) or
  make install
- Edit /etc/lilo.conf (or /etc/grub.conf)
- If using lilo run **# lilo** to store the new configuration in the MBR
- Cross your fingers and **# reboot**

Your System Log Book



- Include hard copies of your customized system configuration file.
- Keep a chronological diary of system administration decisions, including the driving factor, the proposed solutions, and the final resolutions.
- Maintain a record of hardware configuration, repairs, and additions.
- Record software package installations and patch level, and dates
- Note your observation of the overall health and well being of the system.
- This is your first-best place to turn to when things get interesting!

Patching the Kernel

Version A.00

U2794S Module 22 Slides

Kernel Release Model

2 . 4 . 7

Major Version . Minor Version . Patch Release

Even # → Stable

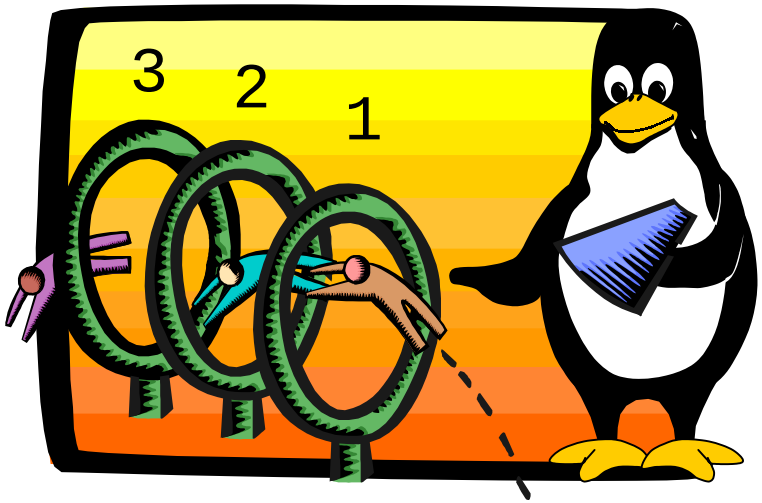
Odd # → Develop

Patches, by the numbers

- The basic steps are:
  - Research the reasons for the patch and determine if you need to proceed
  - Acquire the patch from one of the official kernel web sites
  - Make a backup of your current source build directory tree
  - Unpack the new patch image
  - Apply the patch to your source build tree using the `patch` command
  - Go through the basic steps of re-building your kernel
  - Install the new kernel and re-boot !

Order is important

- Kernel patches MUST be applied in order, that is to say, don't try applying the 2.4.10 patch bundle directly to a 2.4.7 build tree.
  - First apply the 2.4.7
  - Next apply the 2.4.8
  - Next apply the 2.4.9
  - Now apply the 2.4.10



- There are no short-cuts !

Linux (non-RedHat) Kernel Patching

- Use the Linux patch command to apply a diff file
  to kernel existing kernel source tree:

Download the Linux patch from the following Linux kernel website:

http://www.kernel.org/pub/linux/kernel/v2.4/patch-<version>.gz

```
# gunzip patch-<version>.gz
```

```
# patch -bd /usr/src/linux-<version> -p1 < patch-<version+1>
```

then rename the patched kernel source tree directory

and build a new kernel from the patched kernel source tree ...

RedHat (rpm) Kernel "Patching"

- Use the RedHat Package Manager (rpm command) to install all of the new kernel dependencies (there will be many) and then install a new precompiled, ready-to-boot kernel:

```
# rpm -i kernel-<version>.architecture.rpm
```

which installs the following kernel binary files on the Linux system:

```
/boot/System.map-<version>
/boot/module-info-<version>
/boot/vmlinuz-<version>
/lib/modules/<version>/*
```

and configures the Linux boot loader ...

<version>=2.4.7-10, for example

RedHat (rpm) Kernel Source "Patching"

- Use the RedHat Package Manager (rpm command) to install all of the new kernel dependencies (there will be many) and then install the kernel source code tree to build (customize) your own new kernel:

 # rpm -i kernel-source-<version>.architecture.rpm

which installs the kernel source tree on the Linux system:
/usr/src/linux-<version>/*

 # rpm -i kernel-headers-<version>.architecture.rpm

which installs the following kernel headers files on the Linux system:
/boot/kernel.h-<version>
/usr/include/asm/*
/usr/include/linux/*

(Note: kernel-headers rpm is no longer needed for RedHat Linux 7.3)

 # rpm -i kernel-doc-<version>.architecture.rpm

which installs the kernel documentation on the Linux system:
/usr/share/doc/kernel-doc-<version>/*