RED HAT :: NASHVILLE :: 2006

SUMMIT

Open Source from a Proprietary Perspective

Larry “ootpa” Troan

Corporate Preparation

It is **imperative** that people used to working with proprietary code and their management be educated on the differences between it and open source before they begin working with open source and participating in the open source community.

Furthermore, it is **critical** that a company be aware of all open source work going on within its walls and that this work be reviewed and approved prior to project kickoff and again prior to product announcement and ship.

Open Source Education

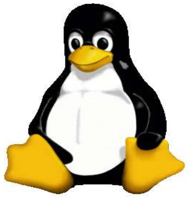
- **Key to working in a company with both proprietary and open source software is open source education…**
  - IBM requires software engineers, their management and documentation personnel to complete a 1-2 hour class on open source before accessing it
    - Class taught by an attorney or an acknowledged expert in this area
    - Engineers would often ask "…so what did they say" after completing education
    - Presenter created a second class that brought open source legal concepts to the engineer's level
  - Working on outside open source projects also requires prior management approval
- **Class covers**
  - Licensing and copyright
  - Contamination
  - Packaging and distribution
  - Required availability of source code
  - Practical examples

Agenda

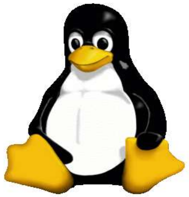
- Presenter's Background
- Corporate Preparation
- Overview of Linux Structure
- Legal Aspects
- Contamination
- Red Hat
- Examples
- Packaging
- Redistribution Requirements
- Searching for Problems

Presenter's Background

- Retired from IBM after 33 years as a Senior Software Engineer working in Development on a variety of hardware and software products
  - Last two years on an embedded Linux Network Attached Storage (NAS) product
  - Project’s primary interface to IBM Legal, IBM Open Source Steering Committee and IBM Linux Technology Center and with Red Hat
- Four years working for Red Hat
  - Senior Technical Account Manager
  - Currently, Senior Partner Manager
  - Working on a series of articles and hopefully a book on Open Source Software Licensing with Mark Webbink, Deputy General Counsel at Red Hat
- This presentation reflects the presenter's opinions based on his knowledge and experience working with both proprietary and open source software and is not intended to represent either IBM's or Red Hat's position on open source
  - First presented subject at the **Licensing Executives Society** meeting in May 2005



Robustness versus Acceptance



An operating system's ***robustness*** is often measured by the number of days, weeks or months between reboots, its raw number crunching power or its ability to support a large number of users running several tasks simultaneously.

An operating system's ***acceptance***, however, is based on the quantity and robustness of the application software available to run on it…

Larry Troan

May 2002

Corporate Control of Open Source Projects

- IBM Open Source Steering Committee
  - Initial conception
  - Completion of open source education class
  - Technical checkpoints
  - Management checkpoints
  - Legal checkpoints
  - Approval prior to product announcement

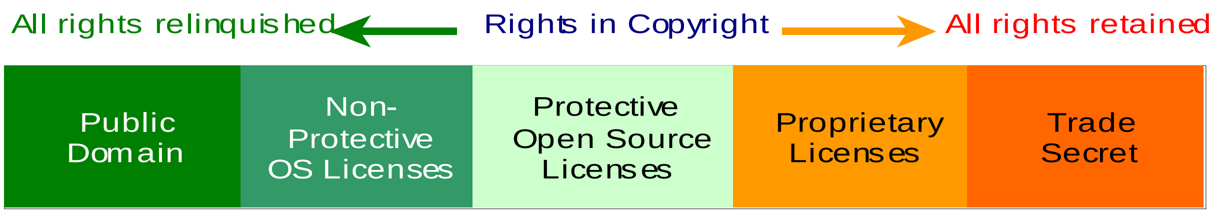
- HP has Open Source Review Board
  - Similar function as IBM OSSC

Legal Aspects

Legally, very little in open source is black or white, but there are many shades of gray since there is no legal precedent set by the courts. Therefore.....

Decisions must be made and actions taken based on license interpretation and the amount of perceived risk.

Copyright



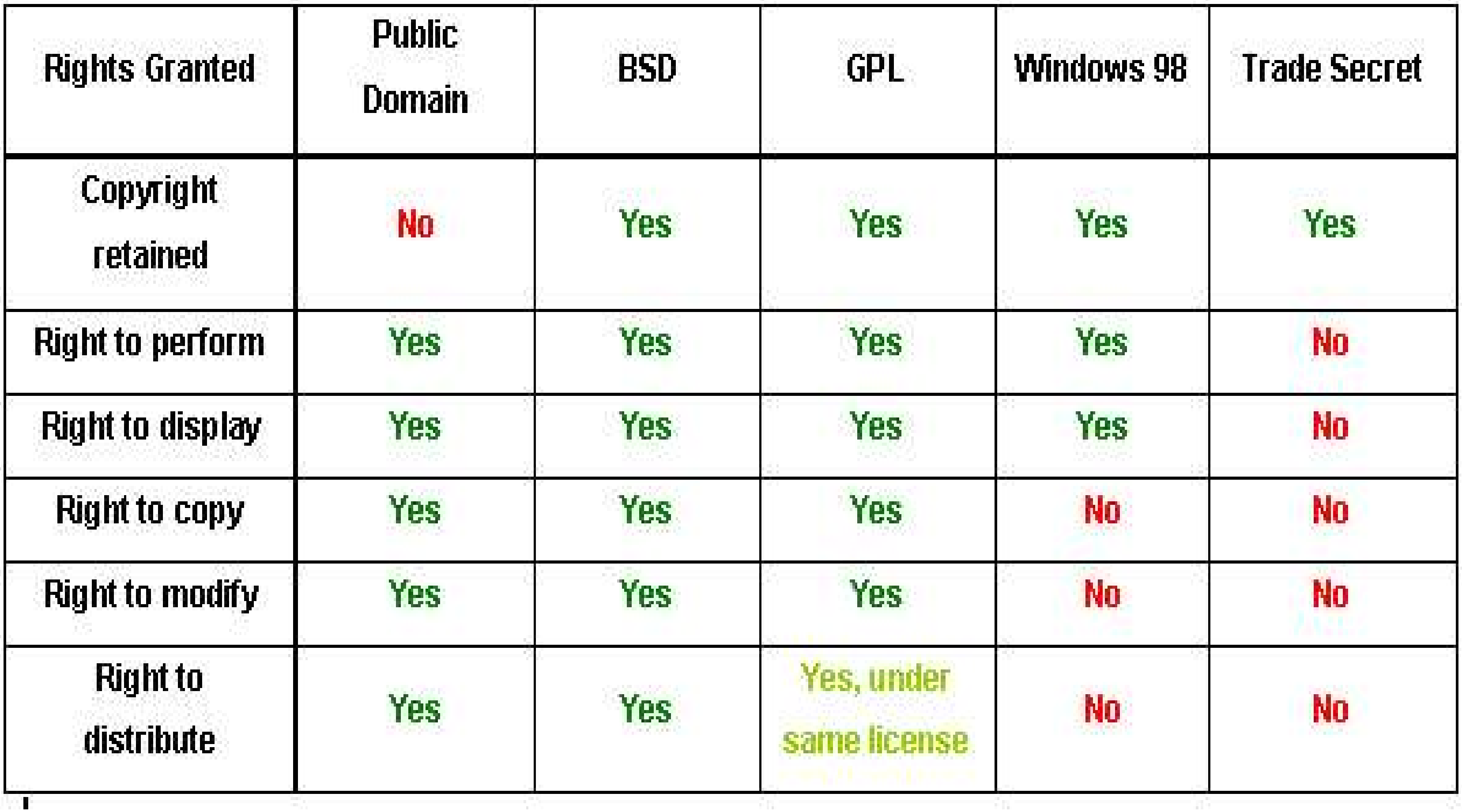
All rights relinquished ← Rights in Copyright → All rights retained

Public Domain | Non-Protective OS Licenses | Protective Open Source Licenses | Proprietary Licenses | Trade Secret

Copyright Mark Webbink 2005. Used with permission.

- **Rights**
  - Public Domain – free of all restrictions
  - Non-Protective – retain copyright but grant all rights to copyright the licensee
  - Protective – non-protective plus restrictions, typically requirement to publish source
  - Proprietary – retain copy and grant few rights (perform and display)
  - Trade Secret – all rights retained
- **Clear copyright or all holders' permission required to add or change a license**
  - Samba requires submissions by individuals (not companies and copyright assignment)
  - Compatible licensing: BSD => GPL but not reverse
  - Copyright holder can license under multiple license types: BSD+GPL+Proprietary

License



Rights Granted	Public Domain	BSD	GPL	Windows 98	Trade Secret
Copyright retained	No	Yes	Yes	Yes	Yes
Right to perform	Yes	Yes	Yes	Yes	No
Right to display	Yes	Yes	Yes	Yes	No
Right to copy	Yes	Yes	Yes	No	No
Right to modify	Yes	Yes	Yes	No	No
Right to distribute	Yes	Yes	Yes, under same license	No	No

Copyright Mark Webbink 2005. Used with permission.

Warranties and Indemnification

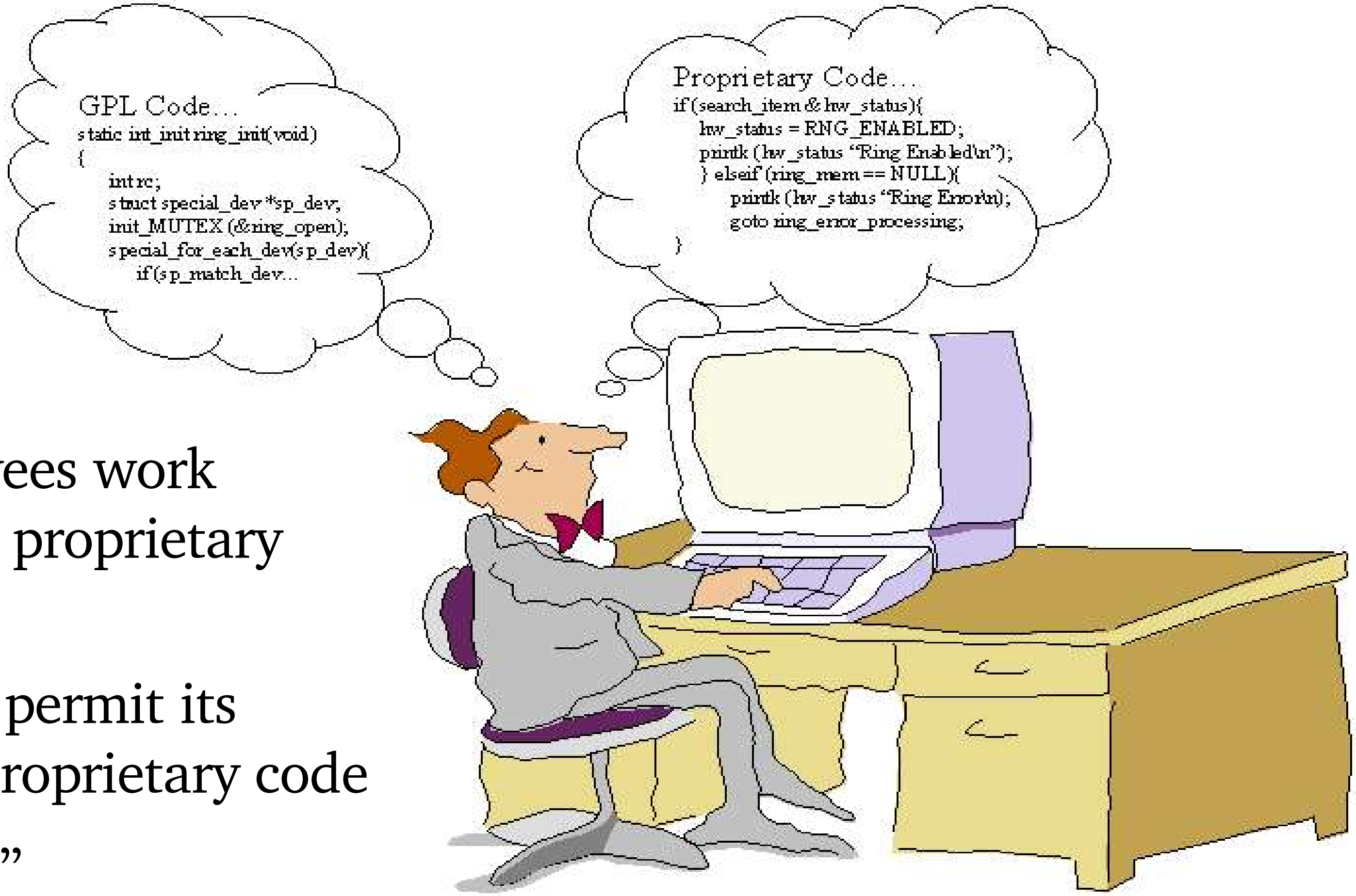
- **Most Open Source licenses disclaim all warranties, including the warranty of non-infringement.**
  - Disclaimed to protect programmers who voluntarily contribute code to Open Source projects
  - Some Open Source licenses, like the GPL, permit parties to offer and to charge for warranty coverage if they should so choose.
- **Indemnification for infringement**
  - Any software company can elect to indemnify its customers for infringement claims at any time
  - Prudent risk management practices dictate against providing such protection in all cases and/or providing it as a standard license provision
  - Times where software companies deem it to be in their own self-interest to provide such indemnification
  - Significant advantage Open Source distributors enjoy is their source code is available for inspection

Derived Works

> A work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a "derivative work” (U.S. Code §101)

- Legal Posturing – There is no legal precedent set by the courts
- User space
  - Proprietary applications using standard libraries and system calls are permissible running on Linux
  - Changes to GPL/LGPL libraries or system calls or combining with GPL code is not permitted unless the resulting work is also GPL
- Kernel space
  - Anything statically linked with the Linux kernel is a derived work and must be licensed under the GPL
  - Dynamically loaded modules inserted during runtime that use standard kernel headers and exported variables are permissible provided they do not include any GPL code
    - Strongest case if ported from another operating system
    - If new driver, where did it come from?
  - “Shim layer” using open source that can be recompiled to reflect kernel changes
    - Must do more than simply remap calls one-for-one
- Macros
  - Maximum 10 lines or less of code (lower limit of copyrightable material)
- Header files
  - Uncertain whether they can be copyrighted. Best to abide by licensing to be safe

Contamination



GPL Code…

```
static int_init ring_init(void)
{
     int rc;
     struct special_dev *sp_dev;
     init_MUTEX (&ring_open);
     special_for_each_dev(sp_dev){
          if(sp_match_dev…
```

Proprietary Code…

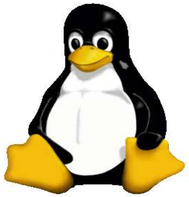
```
if(search_item & hw_status){
     hw_status = RNG_ENABLED;
     printk (hw_status "Ring Enabled\n");
     } elseif (ring_mem == NULL){
          printk (hw_status "Ring Error\n);
          goto ring_error_processing;
}
```

- Do <u>not</u> let employees work concurrently with proprietary and open source
- Red Hat does not permit its engineers to see proprietary code
- "Cubical wall rule"
- Shim layer especially important

Pedigree

- Know where the source code originated and who has touched it since it was created
  - Potential copyright infringement
  - Potential intellectual property (patent) infringement
- Chances better if your package distributed by major Distribution
- Suggest database with package history

Field	Entry
Module/Package Name	Full name of package
Description	Short description of package
Function	Function of package
User/kernel Space	Package contains files that are user or kernel space or both
Origin	Location where package was obtained
Download Date	Date download into project database
Download Person	Name of person downloading package
License	Type of license
License Comments	Comments or location of additional license information
Modified Source	Yes/No (Used to determine if source must be externalized)
Potential Patent Concerns	Legal or technical staff comments regarding patent concerns
Major Distributions	Major Distributions shipping this package
Approved by Legal	Name of attorney
Approval Date	Date approved by Legal Department
Additional Comments	Other relevant comments



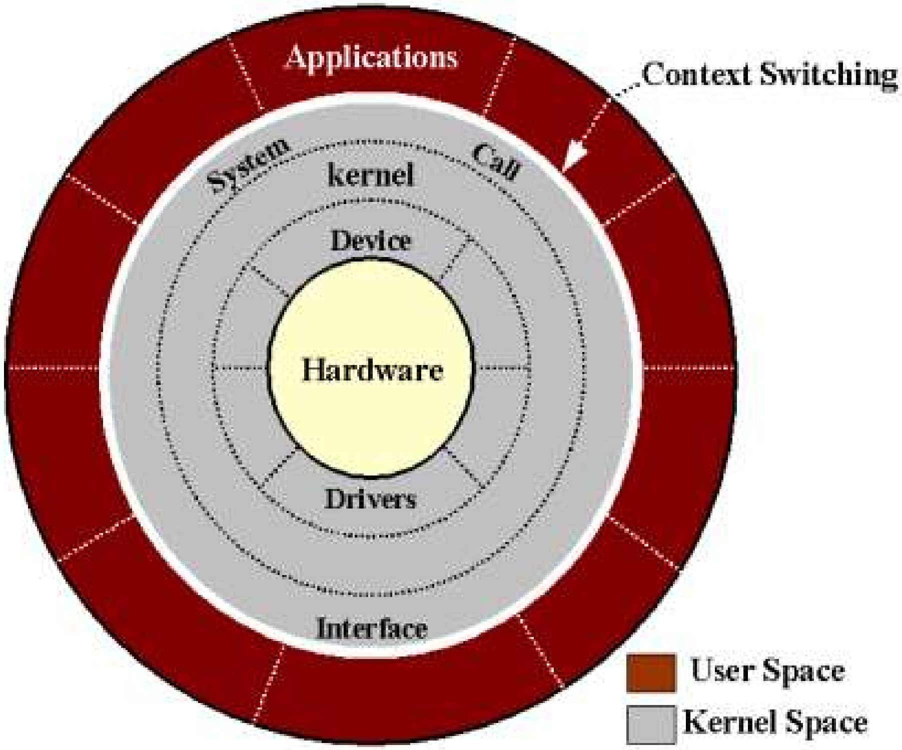
The Community

- The Cathedral and the Bazaar*
  - In a proprietary model, products driven by schedules with corporate resources. Licensing usually determined by Legal department and not an issue. Shipped as binary only.
  - In an open source model, products driven by community interest with some resources provided by distributors. Licensing determined by authors and source is usually readily available.
- Questions:
  - What is a Distribution?
  - What is Community Participation?
  - How is Unix/Linux structured?

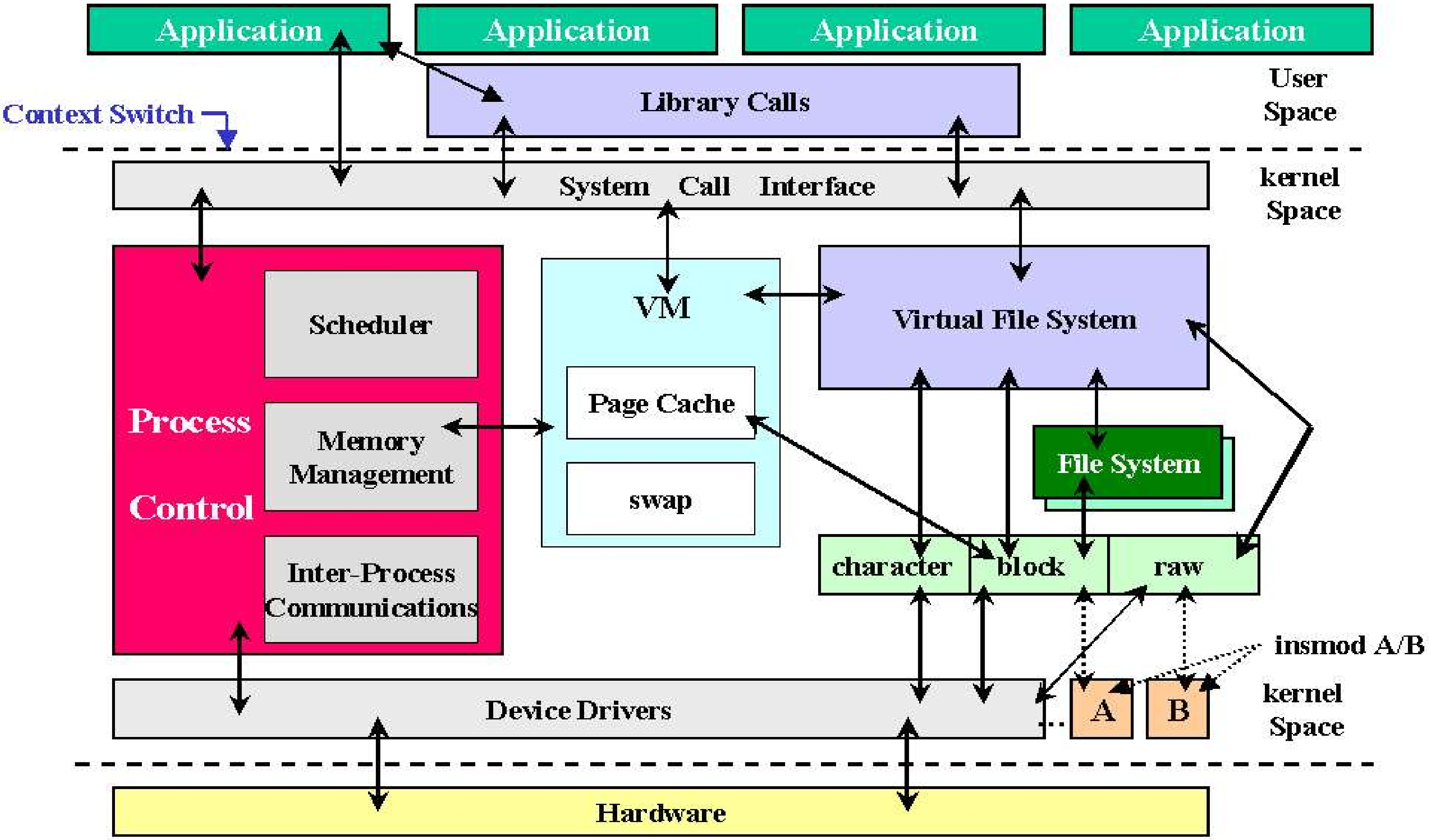
* **"The Cathedral and the Bazaar,"** by Eric S Raymond, O'Reilly, January 2001, ISBN 0-596-00108-8

Unix/Linux Organization

- **User space**
  - Applications
    - Oracle DB
    - Apache web server
  - Desktop
  - Daemons
- **Kernel space**
  - Operating System kernel
  - Device Drivers
    - Video
    - Network
    - Sound
    - CD/DVD
    - Hard Drives

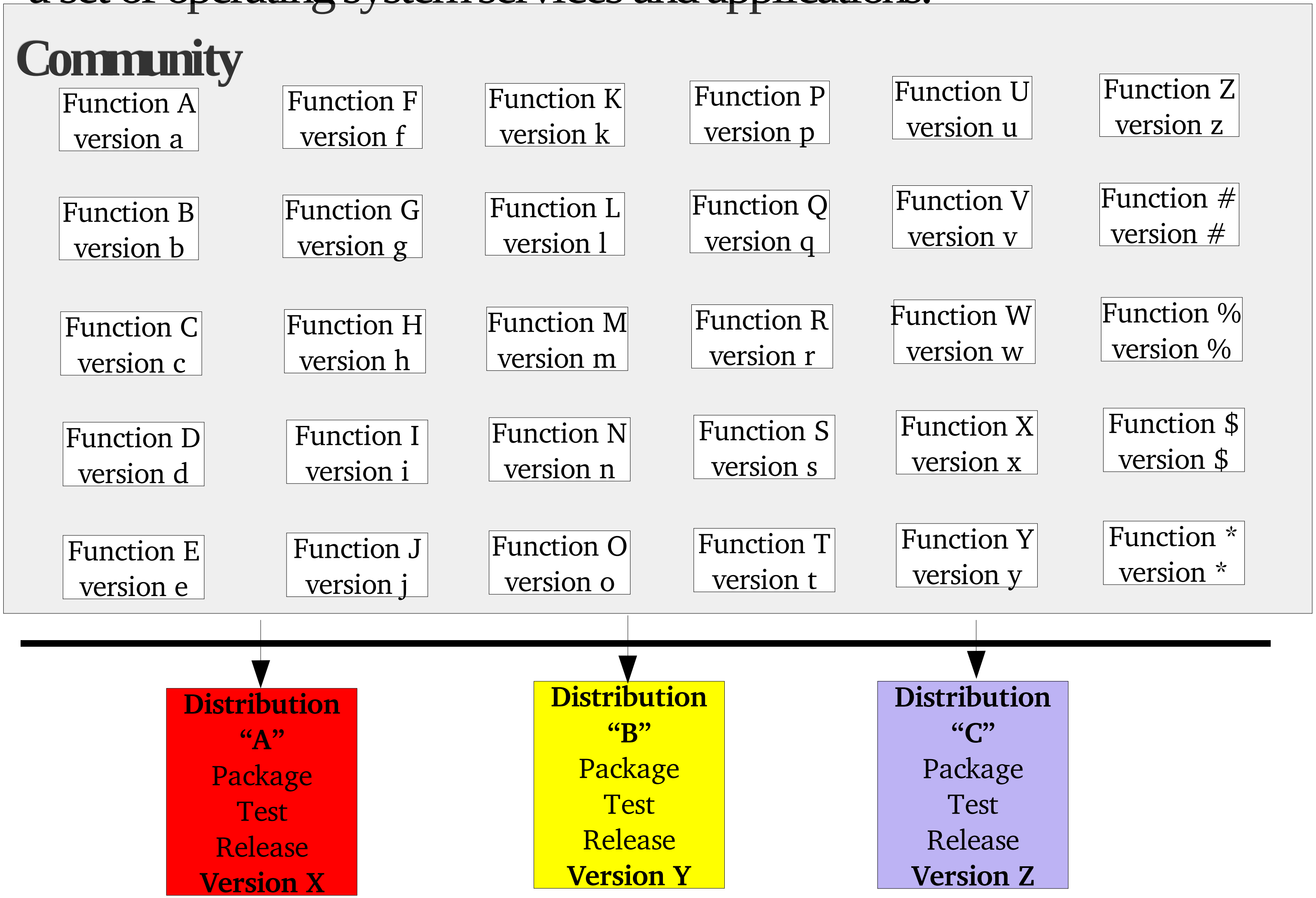


Unix/Linux Organization...



Definition of a Distribution

In Linux, a “distribution” is a collection of packages that provide a set of operating system services and applications.



Community

Function A version a	Function F version f	Function K version k	Function P version p	Function U version u	Function Z version z
Function B version b	Function G version g	Function L version l	Function Q version q	Function V version v	Function # version #
Function C version c	Function H version h	Function M version m	Function R version r	Function W version w	Function % version %
Function D version d	Function I version i	Function N version n	Function S version s	Function X version x	Function $ version $
Function E version e	Function J version j	Function O version o	Function T version t	Function Y version y	Function * version *

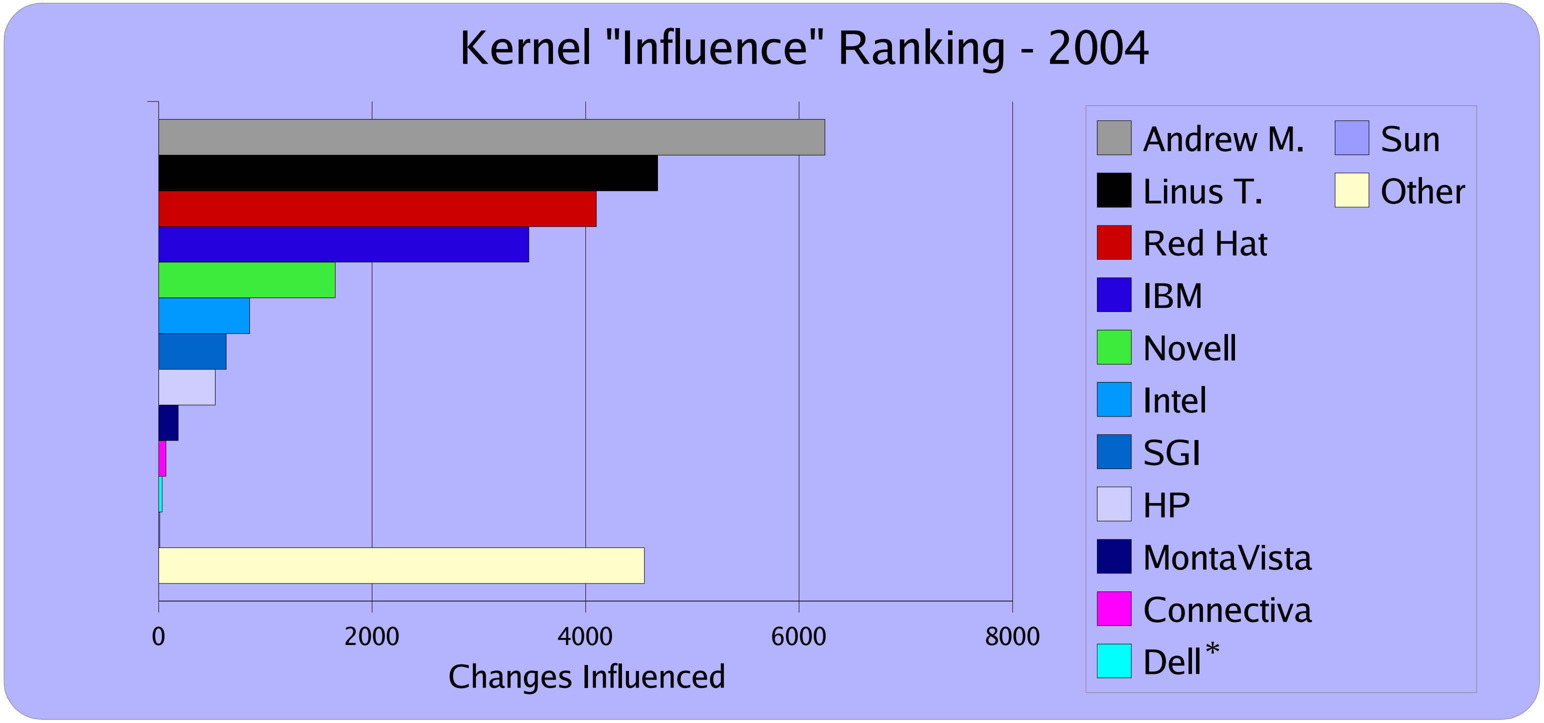
Distribution “A”
Package
Test
Release
Version X

Distribution “B”
Package
Test
Release
Version Y

Distribution “C”
Package
Test
Release
Version Z

Community Contributions

- The strength of Linux is based on having a large, vigorous, development community
  - Not overpowered by a single commercial entity
- Red Hat is the leading commercial engineering contributor



Kernel BitKeeper analysis, January 2005   Influence = feature creator, reviewer, or acceptor

* Some companies primarily contribute through Andrew Morton's tree

Red Hat Development Model

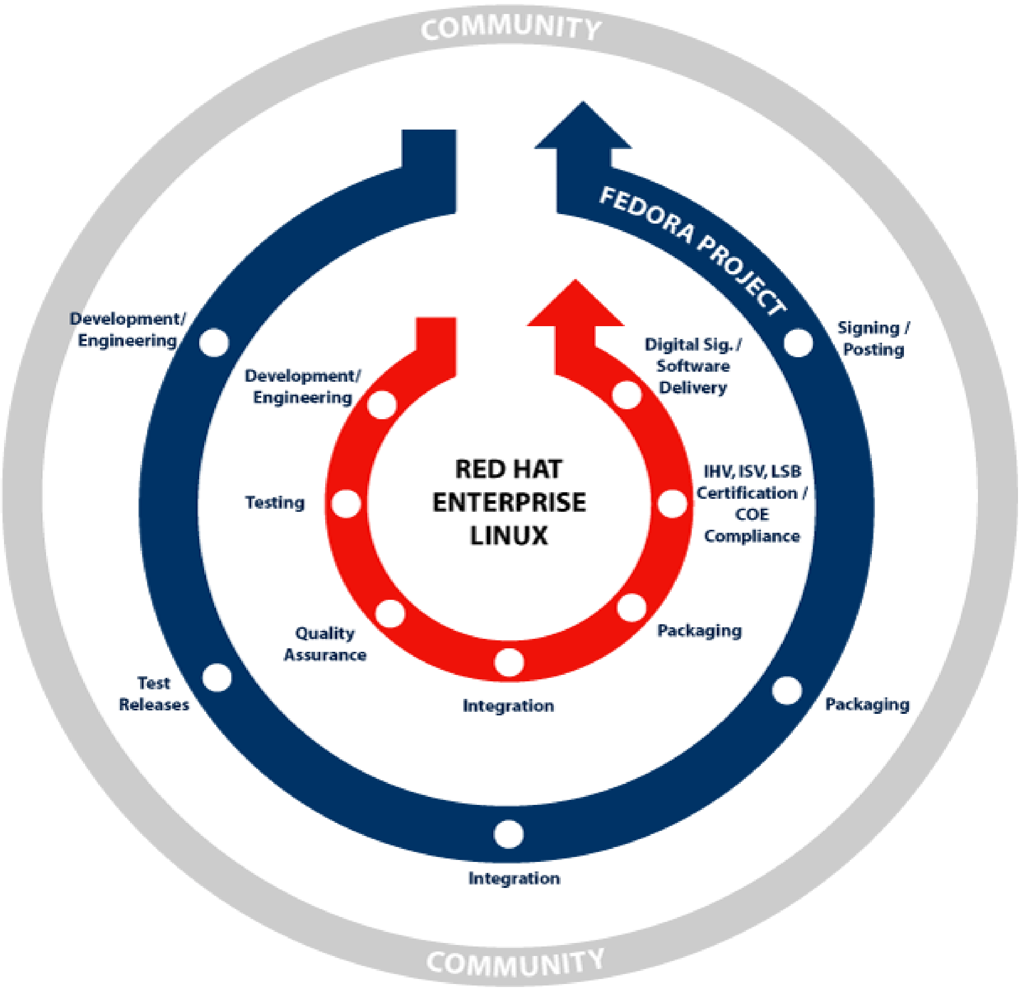
#1 - Red Hat contributes directly to open source projects & sponsors the Fedora distribution project

#2 - Red Hat adds the engineering required to make open source projects deployable in commercial enterprise environments

- Additional development
- Integration & Hardening
- QA testing & Benchmarking
- Delivery
- Hardware & Software Certifications

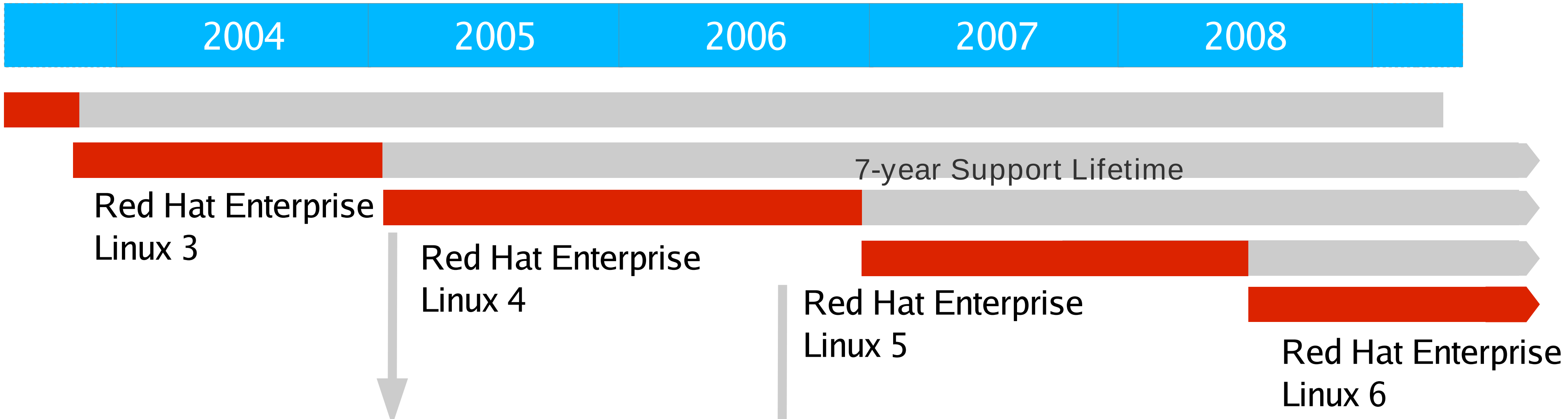
#3 - Red Hat provides services to commercial customers:

- 7 years Maintenance and Updates
- Technical Support
- Training & Consulting Services



RED HAT :: NASHVILLE :: 2006
SUMMIT

Red Hat Enterprise Linux
Delivering consistent commercial solutions


2004
2005
2006
2007
2008
7-year Support Lifetime
Red Hat Enterprise Linux 3
Red Hat Enterprise Linux 4
Red Hat Enterprise Linux 5
Red Hat Enterprise Linux 6

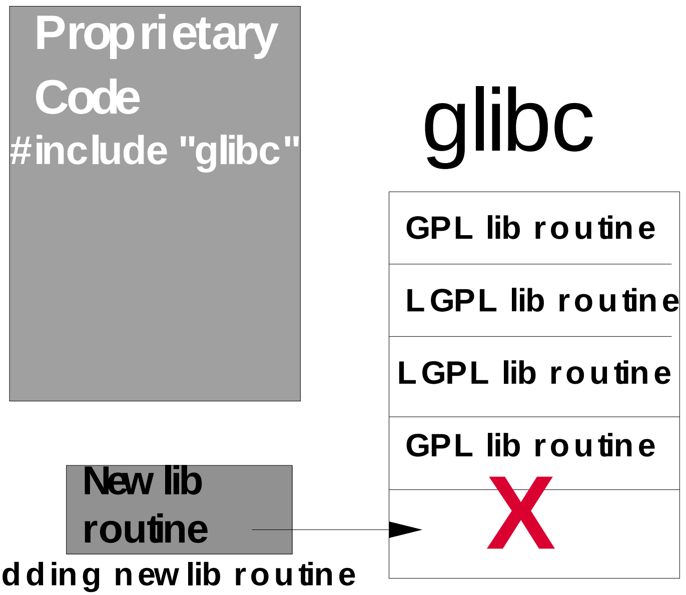
- Linux 2.6 kernel
- SELinux
- Performance/scalability
- 32-bit and 64-bit workloads

- Virtualization (Server/Storage)
- Security: Expanded MAC; compiler/runtime technologies
- Directory integration
- Multi-core/platform features
- Client: Wireless/Stateless

Examples

Practical examples of what can and can not be done.

User Space Example



Proprietary Code
#include "glibc"

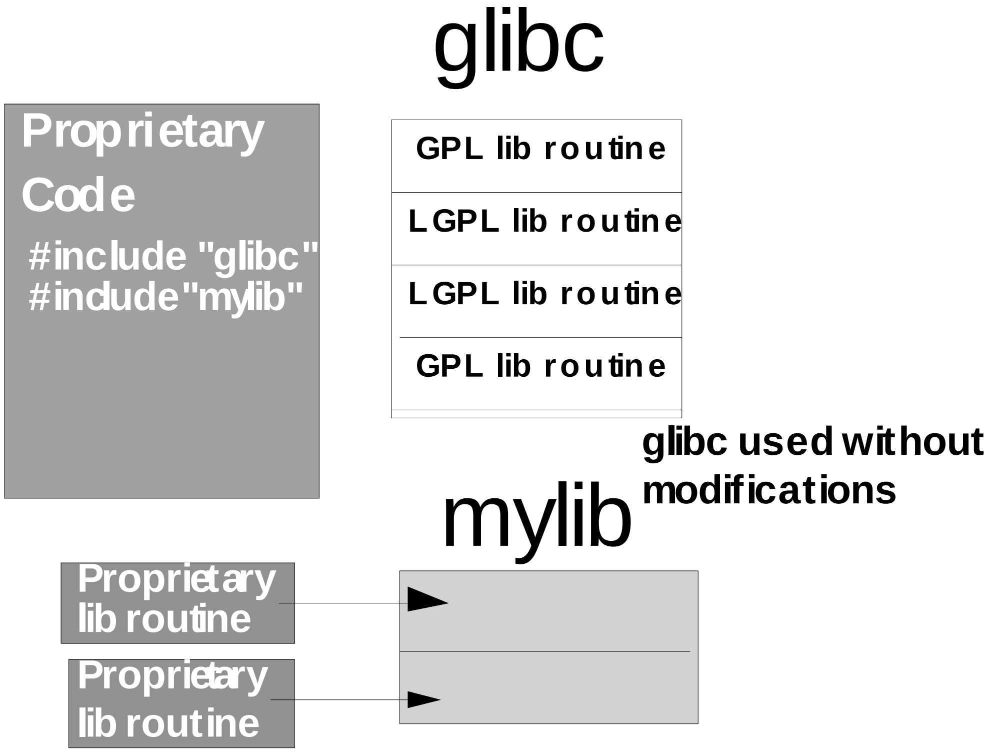
glibc

GPL lib routine
LGPL lib routine
LGPL lib routine
GPL lib routine
X

New lib routine

Adding new lib routine directly to GPL/LGPL library

New lib routine must be GPL licensed



Proprietary Code
#include "glibc"
#indude"mylib"

glibc

GPL lib routine
LGPL lib routine
LGPL lib routine
GPL lib routine

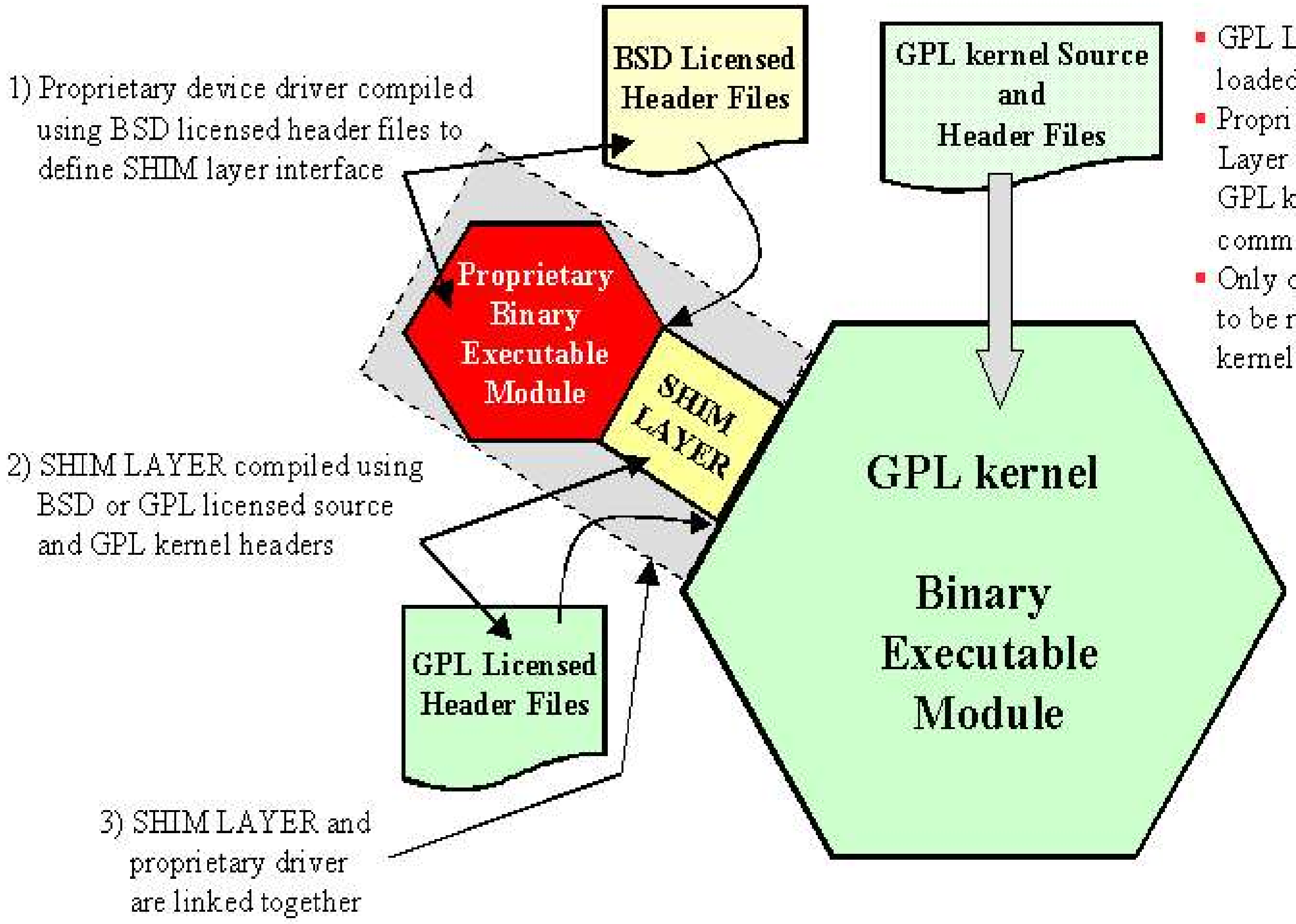
glibc used without modifications

mylib

Proprietary lib routine
Proprietary lib routine

Proprietary lib routines may remain non-GPL

Kernel Example



1) Proprietary device driver compiled using BSD licensed header files to define SHIM layer interface

2) SHIM LAYER compiled using BSD or GPL licensed source and GPL kernel headers

3) SHIM LAYER and proprietary driver are linked together

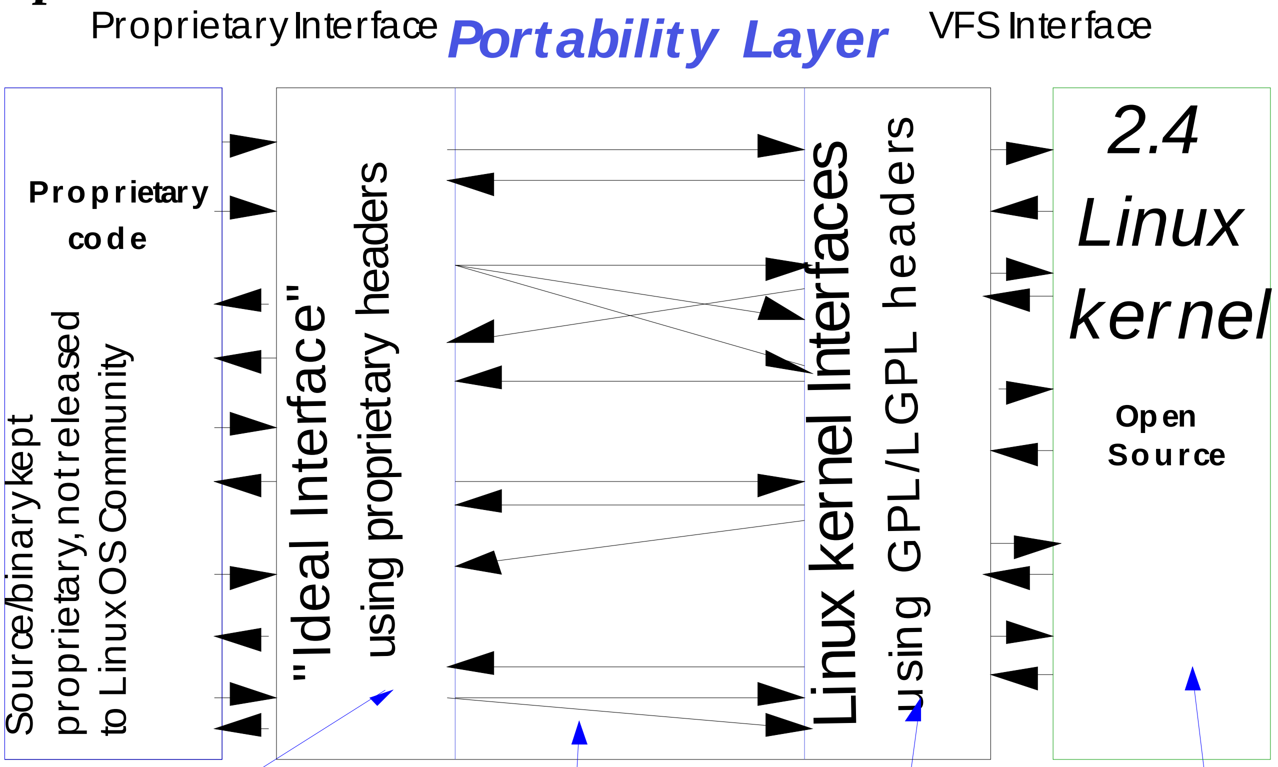
BSD Licensed Header Files
GPL kernel Source and Header Files
Proprietary Binary Executable Module
SHIM LAYER
GPL kernel
Binary Executable Module
GPL Licensed Header Files

- GPL Linux binary kernel module is loaded and running
- Proprietary Module including SHIM Layer is dynamically linked to the GPL kernel using depmod or insmod commands
- Only open source SHIM Layer needs to be recompiled against updated kernel source when kernel is updated

Prudent posturing rather than legal precedent

Example of Isolation Technique

Layer Explanation


Proprietary Interface
Portability Layer
VFS Interface
Proprietary code
Source/binary kept proprietary, not released to Linux OS Community
"Ideal Interface" using proprietary headers
Linux kernel Interfaces using GPL/LGPL headers
2.4 Linux kernel
Open Source

Headers created independent of Linux kernel header files

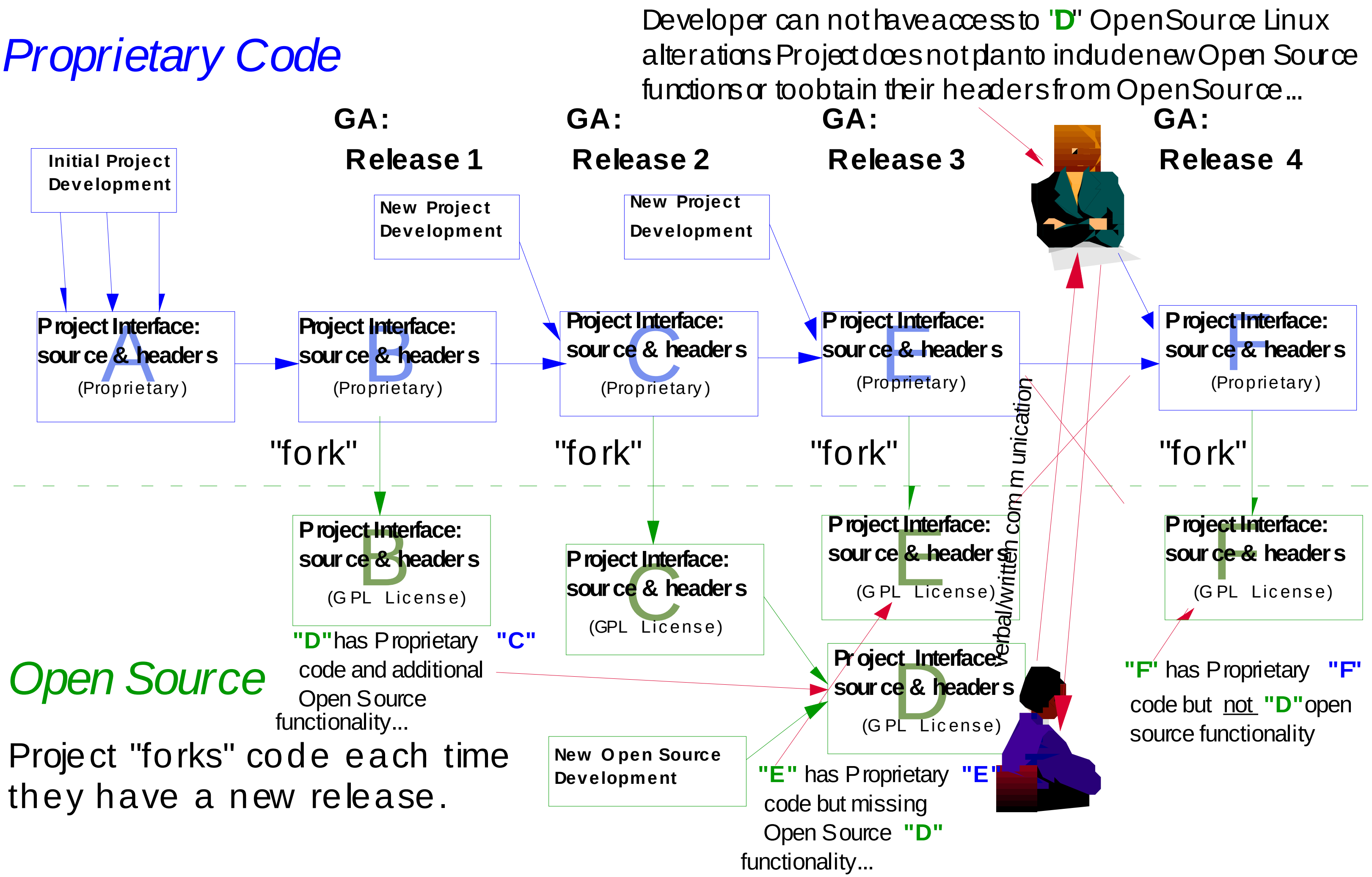
- Not one-for-one mapping to Linux structures or headers
- Not mixed with Open Source code
- Not contaminated by GPL/LGPL

Standard Linux kernel header files

No proprietary hooks or patches

Vendor developed source code (must be returned to Open Source Community)

Example of Isolation Technique



Proprietary Code

Developer can not have access to 'D" OpenSource Linux alterations. Project does not plan to include new Open Source functions or to obtain their headers from OpenSource...

GA: Release 1

GA: Release 2

GA: Release 3

GA: Release 4

Initial Project Development

New Project Development

New Project Development

Project Interface: source & headers (Proprietary) A

Project Interface: source & headers (Proprietary) B

Project Interface: source & headers (Proprietary) C

Project Interface: source & headers (Proprietary) E

Project Interface: source & headers (Proprietary) F

"fork"

"fork"

"fork"

"fork"

Project Interface: source & headers (GPL License) B

Project Interface: source & headers (GPL License) C

Project Interface: source & headers (GPL License) E

Project Interface: source & headers (GPL License) F

Project Interface: source & headers (GPL License) D

verbal/written communication

"D" has Proprietary "C" code and additional Open Source functionality...

Open Source

Project "forks" code each time they have a new release.

New Open Source Development

"E" has Proprietary "E" code but missing Open Source "D" functionality...

"F" has Proprietary "F" code but not "D" open source functionality

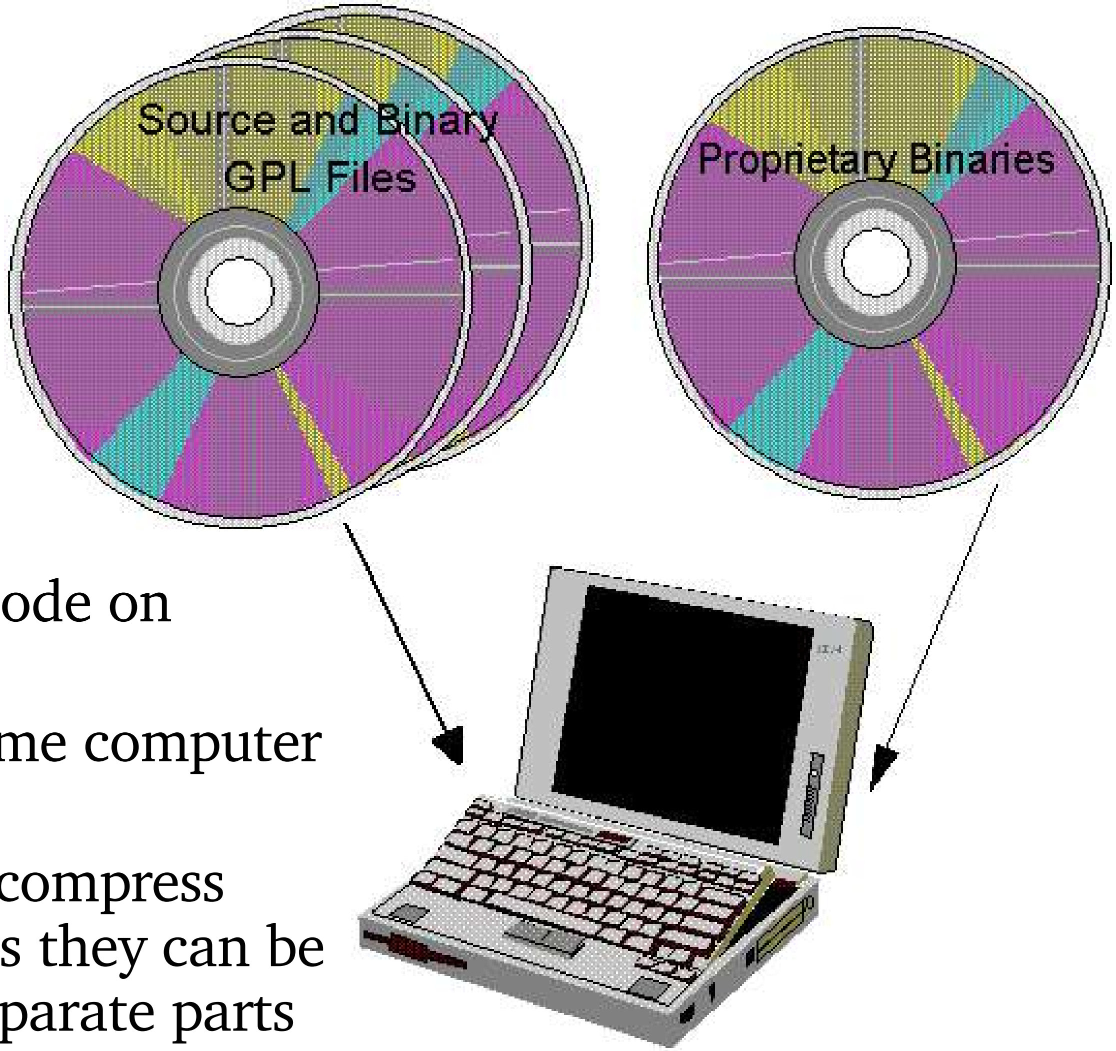
Dark Side of Binary kernel Modules

- Kernel modules may require recompilation for kernel updates
  - IBM first released afs (Andrew File System) as binary
  - Problems and bad press in attempting to keep up with ever changing kernel
  - Decided to released as open source
- Security fixes are asynchronous and unannounced and they may include kernel changes
- “Shim Layer” helps
  - nVidia
  - IBM

Dynamic Kernel Module Support

- DKMS
- Framework allowing individual kernel modules such as device drivers to be upgraded without changing whole kernel
- Makes it easy to rebuild modules as you upgrade kernels
- Allows Linux vendors to provide drops without having to wait for new kernel releases
- Invented by Gary Lerhaupt at Dell
- References:
  - http://linux.dell.com/projects.shtml
  - http://linux.dell.com/dkms
  - http://freshmeat.net/projects/dkms

Packaging



- Keep GPL and proprietary code on separate media
- Both can be installed on same computer hard drive
- You can “tar” or otherwise compress GPL and non-GPL as long as they can be extracted back into their separate parts
- A non-redistributing end user can never violate the GPL…

Requirements to Redistribute

- BSD license allows you to redistribute binaries without making source available
- GPL code may be used without sharing the source as long as it remains within your premises
  - Redistributing it outside firewall requires the source to be made available
- GPL source availability
  - With binaries on media or via Internet
    - If via Internet, then web site must be easily available for period of 5 years
  - Upon request
    - May charge reasonable redistribution fee to cover cost media and shipping
  - Non-commercial only: accompanied with the offer the redistributing party received as to the availability of the source code
- Beware of incompatible licenses
  - NDMP and GPL
  - Only provide source through ndmp.org website vs GPL “must provide source”

Searching for Problems

- Command line
  - rpm2cpio …/filename | cpio –id
    - Expands Red Hat rpm package
  - grep –r string > tmpfile
    - Where “string” is icense, opyright, ommercial, …
    - Greps for the specified string down through the tree
- Black Duck
  - Provides products and services for understanding and safely managing software licensing compliance
  - http://www.blackducksoftware.com/

Questions ?

Daily Trivia

- What's with this “**ootpa**” nickname?
- My son Erik was an early participant in the Linux movement
  - One of the first engineers to work at Red Hat
  - His initials are “ewt”
  - Alan Cox (Red Hat Fellow) started calling him “oot” based on his initials
- I joined Red Hat in 2002
  - Someone asked me if I was related to oot during my company orientation, perhaps his father (or “pa”)
  - The nickname stuck as “ootpa”