Linux Kernel: An Introduction

Ankita Garg (ankita@in.ibm.com)
RealTime Linux Team
IBM Linux Technology Center

16/02/2008

IBM
Linux Technology Center

- World-wide distributed team working with open-source communities
- Mission
  - Make Linux better
  - Accelerate its growth as an enterprise OS
  - Expand Linux Reach
  - Make it mature OS ready for mission critical workloads
- LTC India
  - Development
  - Internal support
  - Testing

Objective

Introduce the Linux kernel and the development community.

Inspire you to play with the kernel and contribute towards making it better

Agenda

- Play with the kernel
- How you can contribute
- Debugging tips

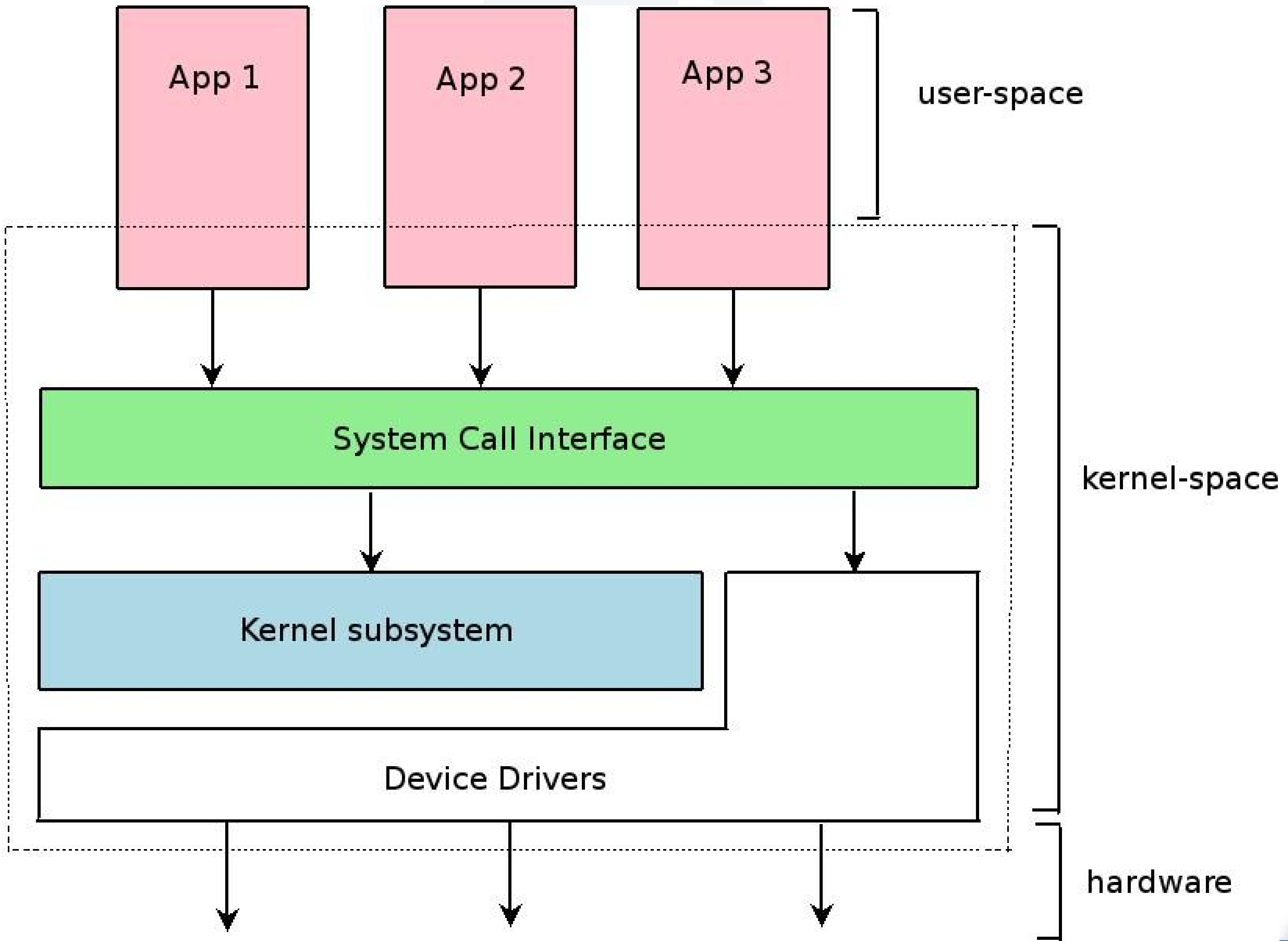
Preliminaries

Preliminaries

- Linux as an Operating System
- A look at the Kernel source tree

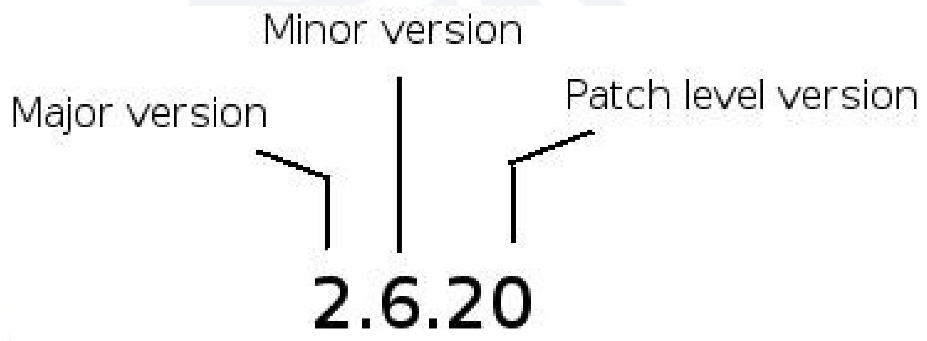
What is an Operating System??

Application, the kernel and hardware



A few things before we get started

- Linux Vs Classic Unix kernels
- Kernel versions

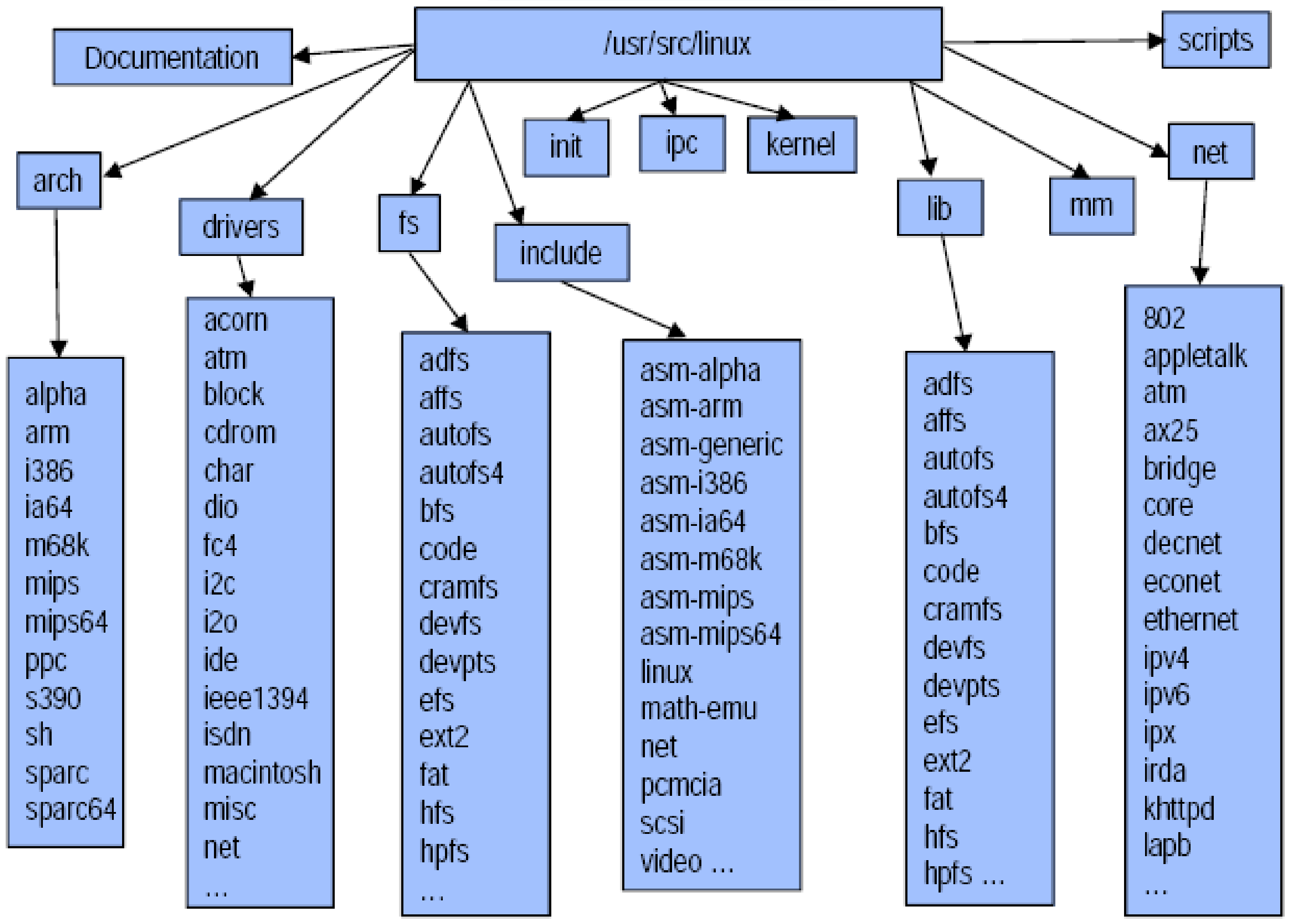


- Hardware dependency
- A beast of a different kind
- Linux distributions – RedHat, SUSE, Fedora, Debian, Ubuntu, Mandriva, YellowDog Linux, Puppy Linux, Gentoo, Slackware Linux

Obtaining the Kernel Source

- ftp from kernel.org
  - ftp ftp.<country>.kernel.org
- clone the linux git tree
  - git clone git://address/linux-2.6.git linux-2.6
- Install kernel source from the distribution CDs
  - rpm -ivh <path-to-src-rpm>/kernel-src.rpm

Getting Started – Kernel Source layout



/usr/src/linux
Documentation
scripts
arch
drivers
fs
include
init
ipc
kernel
lib
mm
net
alpha arm i386 ia64 m68k mips mips64 ppc s390 sh sparc sparc64
acorn atm block cdrom char dio fc4 i2c i2o ide ieee1394 isdn macintosh misc net …
adfs affs autofs autofs4 bfs code cramfs devfs devpts efs ext2 fat hfs hpfs …
asm-alpha asm-arm asm-generic asm-i386 asm-ia64 asm-m68k asm-mips asm-mips64 linux math-emu net pcmcia scsi video …
adfs affs autofs autofs4 bfs code cramfs devfs devpts efs ext2 fat hfs hpfs …
802 appletalk atm ax25 bridge core decnet econet ethernet ipv4 ipv6 ipx irda khttpd lapb …

Let's build our own kernel

Configuration

- First step is to configure the kernel

```
$ make config
$ make oldconfig
$ make menuconfig
$ make gconfig
$ make xconfig
$ make defconfig
```

- Resultant .config file
- Tricky
- Tips
  - /proc/cpuinfo for type of cpu
  - lspci -v
  - dmesg | less
  - enable module loading

Compilation

- Build core kernel image

```
$ make -j<no of jobs> bzImage
```

- Build and install modules

```
$ make modules
$ make modules_install
```

- Recompile kernel

```
$ make clean
$ make mrproper
```

Install & boot into new kernel

- Copy kernel to required location

```
$ cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.24
$ cp System.map /boot/System.map-2.6.24
```

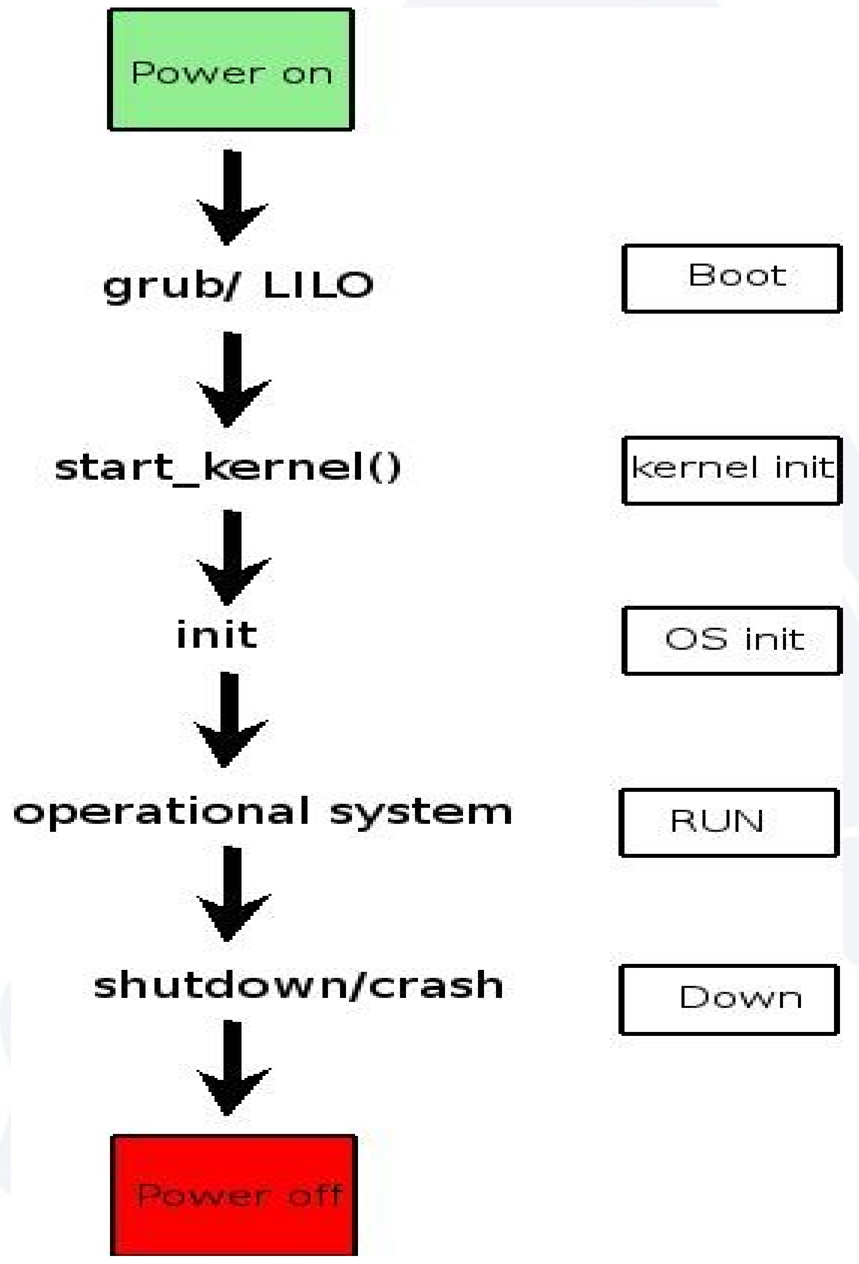
- Edit the bootloader config file
  - Grub: Edit /etc/grub.conf or /boot/grub/menu.lst
  - LILO: /etc/lilo.conf
  - Yaboot: /etc/yaboot.conf

- Recompile kernel

```
$ make clean
$ make mrproper
```

- Install kernel – manually or using a command

Boot up and kernel initialization



/sbin/init

- Process with PID 1
- Kernel forks this process
- Executes startup and shutdown scripts
- Starts/stops important system services (daemons, etc)
- Controls transition between runlevels

Using printk

How you can contribute

Kernel Development Community

- Linux kernel mailing list (lkml)
  - *Info@ http://vger.kernel.org*
  - *Subscribe today!!* :-)
- Various trees / Who's who
- Role of maintainers
- Kernel Bugzilla
- Kernel Test project

How you can Contribute

- Contribution != code only
- What you can do?
  - Test newer kernels
  - Report bugs
  - Squash bugs
- And ofcourse, contribute code once in a while ;-)

Working with Patches

- A patch is a file that describes difference between two versions of a file
- *'diff'* command to create a patch
- *'patch'* patches to apply the patch

- Or use *'quilt'*

Code contribution

- Refer kernel Documentation (/Documentation)
- Create Patches
- Test the patch, get reviews
- Dos & Dont's
  - Read through the patch, check if unwanted files included in the patch
  - watch the level of directories
  - create against latest kernel version
  - Coding Style
- Submitting Patches
  - Refer the MAINTAINERS file (though most often out-of-date)
  - Watch the mailing list
  - Ask somebody
- Distributing your patches

Writing kernel modules

Dynamically Loadable Kernel module

- Kernel is modular, allowing dynamic insertion and removal of code
- Minimal base kernel image, with features/drivers as modules
- Best way to get started !! So, let's write one

myModule.c

```
/*  myModule.c */

#include<linux/module.h>
#include<linux/init.h>
#include<linux/kernel.h>

static int __init myhello_init(void)
{
        printk(KERN_INFO “\n Hello there, mymodule loaded \n”);
        return 0;
}

static void __exit myhello_exit(void)
{
        printk(KERN_INFO “\n Have a nice day! myModule unloaded \n”);
}

module_init(myhello_init);
module_exit(myhello_exit);

MODULE_LICENSE(“GPL”);
MODULE_AUTHOR(“XYZ”);
```

Building & installing myModule.c

- Edit Makefile in the same directory..add the following line:

```
obj-m += myModule.o
```

- Compile module

```
# make -C <top level kernel src tree> SUBDIRS=$PWD modules
```

- Load the module

```
# insmod myModule.ko
```

- Verify module loaded

```
# dmesg | tail
```

Building & installing a module

- Write a makefile

```
obj-m := kprobe-example.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
        $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
clean:
        rm -f *.mod.c *.ko *.o
```

- Build using *make*
- Install using *insmod*

Linux Modules

- To make it part of the kernel source
- Module Parameters

```
module_param(name, type, perm);
```

- Pass as *name=value* while inserting
- Use lsmod, insmod, modprobe, rmmod

Debugging Methods/Tips

Debugging Tips

- Use printk
- /proc filesystem
- kprobes / SystemTap (link to stap)
- KDB
- KGDB
- QEMU
- kdump
- lockstat, logdev

Useful Tools

- Source code browsing – cscope, lxr, ctags, etc
  - Cscope
    - http://cscope.sourceforge.net/
    - http://cscope.sourceforge.net/cscope_vim_tutorial.html
  - LXR
    - http://lxr.linux.no/
- Patching the kernel – patch, quilt
- GIT

Start Contributing

- Read/Understand the kernel source
- www.kernelnewbies.com
- lwn.net
- Linux kernel Documentation
- Books
  - *Linux Kernel Development* by *Robert Love*
  - *Understanding the Linux Kernel* by *Bovet and Cesati*
  - *Linux Device Drivers* by *Alessandro Rubini*
- Follow the mailing list
- Use Linux as your primary OS

Questions ??

Legal Statement

Thank You !!

Back up Slides

A tour of www.kernel.org

Boot loader: GRUB / LILO

- Charactersitics:
  - Versatile boot manager that supports choice of kernel
  - Boot time kernel parameters
  - Booting non-Linux kernels
  - Variety of configurations
  - Lives in MBR or partition boot sector
  - Has a configuration file (eg, /etc/lilo.conf, /etc/grub.conf)
- Sample boot loader configuration

Advanced Boot concepts

- Initial ramdisk (initrd) – two stage boot for flexibility:
  - First mount “initial” ramdisk
  - Execute linuxrc for setup, configuration, etc
  - Finally mount “real” root filesystem

/proc filesystem

- virtual filesystem mounted under /proc
- Provided information on running processes
  - read-only access to kernel data structures
  - superuser can change kernel parameters at runtime
- Linux kernel Documentation