

# Linux Kernel Walkthrough

Bart Trojanowski, bart@jukie.net

Jukie Networks Inc.

June 24, 2008

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# Table of contents

Introduction

General Info

The Code

Q&A

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
**bart@juki.net**

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

Next...

Introduction

General Info

The Code

Q&A

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
**bart@jukie.net**

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# About OCLUG

<http://oclug.on.ca>

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
[bart@jukie.net](mailto:bart@jukie.net)

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# Tutorials

4th Thursday of the month

(except for July)

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
[bart@jukie.net](mailto:bart@jukie.net)

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Tutorials

4th Thursday of the month  
(except for July)

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
[bart@jukie.net](mailto:bart@jukie.net)

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Code Walkthroughs

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
[bart@jukiie.net](mailto:bart@jukiie.net)

poetry for nerds

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# Next...

Introduction

General Info

The Code

Q&A

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
[bart@jukie.net](mailto:bart@jukie.net)

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

Introduction

General Info

**Numbers**

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# Numbers

# Releases

- ▶ old "stable" 2.4.x series
  - ▶ 2.4.36
  
- ▶ Linus' 2.6.x tree
  - ▶ 2.6.25
  
- ▶ stable 2.6.x.y series
  - ▶ 2.6.25.9
  
- ▶ special trees
  - ▶ linux-mm
  - ▶ linux-next
  - ▶ linux-staging

# Releases

- ▶ old "stable" 2.4.x series
  - ▶ 2.4.36
  
- ▶ Linus' 2.6.x tree
  - ▶ 2.6.25
  
- ▶ stable 2.6.x.y series
  - ▶ 2.6.25.9
  
- ▶ special trees
  - ▶ linux-mm
  - ▶ linux-next
  - ▶ linux-staging

# Releases

- ▶ old "stable" 2.4.x series
  - ▶ 2.4.36
  
- ▶ Linus' 2.6.x tree
  - ▶ 2.6.25
  
- ▶ stable 2.6.x.y series
  - ▶ 2.6.25.9
  
- ▶ special trees
  - ▶ linux-mm
  - ▶ linux-next
  - ▶ linux-staging

- ▶ old "stable" 2.4.x series
  - ▶ 2.4.36
  
- ▶ Linus' 2.6.x tree
  - ▶ 2.6.25
  
- ▶ stable 2.6.x.y series
  - ▶ 2.6.25.9
  
- ▶ special trees
  - ▶ linux-mm
  - ▶ linux-next
  - ▶ linux-staging

# Lines of code

## ▶ source

- ▶ 20k files
- ▶ 9.2M lines
- ▶ increases by 10% each year
- ▶ core ... 5% LOC
- ▶ drivers ... 55% LOC

## ▶ Documentation

- ▶ 1k files
- ▶ 254k lines

# Lines of code

- ▶ source
  - ▶ 20k files
  - ▶ 9.2M lines
  - ▶ increases by 10% each year
  - ▶ core ... 5% LOC
  - ▶ drivers ... 55% LOC
  
- ▶ Documentation
  - ▶ 1k files
  - ▶ 254k lines

# Lines of code

- ▶ source
  - ▶ 20k files
  - ▶ 9.2M lines
  - ▶ increases by 10% each year
  - ▶ core ... 5% LOC
  - ▶ drivers ... 55% LOC
  
- ▶ Documentation
  - ▶ 1k files
  - ▶ 254k lines

Introduction

General Info

**Numbers**

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# Rate of change

- ▶ 4.5k added, 1.8k removed, 1.5k changed *each day*
  - ▶ 2-4 changes per hour
  
- ▶ release every 2-3 months
  
- ▶ 2.4k developers
  - ▶ 25% are hobbyists

# Rate of change

- ▶ 4.5k added, 1.8k removed, 1.5k changed *each day*
  - ▶ 2-4 changes per hour
  
- ▶ release every 2-3 months
  
- ▶ 2.4k developers
  - ▶ 25% are hobbyists

# Rate of change

- ▶ 4.5k added, 1.8k removed, 1.5k changed *each day*
  - ▶ 2-4 changes per hour
  
- ▶ release every 2-3 months
  
- ▶ 2.4k developers
  - ▶ 25% are hobbyists

# Support for everything

"Linux supports more individual types of devices overall than any other operating system" - Greg KH

- ▶ 25 architectures
- ▶ 60 filesystems
- ▶ 20 net protocols (L2)
- ▶ 50+ ATA drivers
  - ▶ support for 300+ PCI devices
- ▶ 200+ Ethernet drivers

# Support for everything

"Linux supports more individual types of devices overall than any other operating system" - Greg KH

- ▶ 25 architectures
- ▶ 60 filesystems
- ▶ 20 net protocols (L2)
- ▶ 50+ ATA drivers
  - ▶ support for 300+ PCI devices
- ▶ 200+ Ethernet drivers

Introduction

General Info

**Numbers**

Files

Tree

The Code

Types

OO

Macros

API

Q&A

Introduction

General Info

Numbers

**Files**

Tree

The Code

Types

OO

Macros

API

Q&A

Types of files

# Source files

- ▶ C99
- ▶ GCC extensions
- ▶ Documentation/CodingStyle
- ▶ OO
  - ▶ yes, in C

# Source files

- ▶ C99
- ▶ GCC extensions
- ▶ Documentation/CodingStyle
- ▶ OO
  - ▶ yes, in C

# Source files

- ▶ C99
- ▶ GCC extensions
- ▶ Documentation/CodingStyle
- ▶ OO
  - ▶ yes, in C

# Source files

- ▶ C99
- ▶ GCC extensions
- ▶ Documentation/CodingStyle
- ▶ OO
  - ▶ yes, in C

# Source files

- ▶ C99
- ▶ GCC extensions
- ▶ Documentation/CodingStyle
- ▶ OO
  - ▶ yes, in C

# Scripts

- ▶ 60% C
- ▶ 20% bash
- ▶ 10% perl
- ▶ >1% python

# Configuration

- ▶ `.config`
  - ▶ `CONFIG_E1000=m`
- ▶ >400 Kconfig files
- ▶ `make menuconfig`
- ▶ `make xconfig`
- ▶ `make gconfig`
  
- ▶ `make oldconfig`
- ▶ `make allmodconfig`
- ▶ `make randconfig`

# Configuration

- ▶ .config
  - ▶ CONFIG\_E1000=m
  
- ▶ >400 Kconfig files
  
- ▶ make menuconfig
- ▶ make xconfig
- ▶ make gconfig
  
- ▶ make oldconfig
- ▶ make allmodconfig
- ▶ make randconfig

# Configuration

- ▶ `.config`
  - ▶ `CONFIG_E1000=m`
- ▶ >400 Kconfig files
- ▶ `make menuconfig`
- ▶ `make xconfig`
- ▶ `make gconfig`
  
- ▶ `make oldconfig`
- ▶ `make allmodconfig`
- ▶ `make randconfig`

# Configuration

- ▶ .config
  - ▶ CONFIG\_E1000=m
  
- ▶ >400 Kconfig files
  
- ▶ make menuconfig
- ▶ make xconfig
- ▶ make gconfig
  
- ▶ make oldconfig
- ▶ make allmodconfig
- ▶ make randconfig

# Build system

- ▶ >1000 makefiles
- ▶ make help
- ▶ make oldconfig all
- ▶ sudo make modules\_install install
  
- ▶ make deb-pkg
- ▶ make rpm-pkg

# Build system

- ▶ >1000 makefiles
- ▶ make help
- ▶ make oldconfig all
- ▶ sudo make modules\_install install
- ▶ make deb-pkg
- ▶ make rpm-pkg

# Build system

- ▶ >1000 makefiles
- ▶ make help
- ▶ make oldconfig all
- ▶ sudo make modules\_\_install install
  
- ▶ make deb-pkg
- ▶ make rpm-pkg

# Build system

- ▶ >1000 makefiles
- ▶ make help
- ▶ make oldconfig all
- ▶ sudo make modules\_\_install install
  
- ▶ make deb-pkg
- ▶ make rpm-pkg

Introduction

General Info

Numbers

Files

**Tree**

The Code

Types

OO

Macros

API

Q&A

Kernel source tree

# Core system

▶ arch	.....	1.9M	...	20%
▶ x86	....	148k	...	1.5%
▶ include	...	874k	...	9.3%
▶ init	.....	4k	...	0.042%
▶ kernel	.....	96k	...	1.0%
▶ lib	.....	26k	...	0.28%
▶ mm	.....	50k	...	0.54%
▶ scripts	....	44k	...	0.47%

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Core system

▶ arch	.....	1.9M	...	20%
▶ x86	....	148k	...	1.5%
▶ include	...	874k	...	9.3%
▶ init	.....	4k	...	0.042%
▶ kernel	.....	96k	...	1.0%
▶ lib	.....	26k	...	0.28%
▶ mm	.....	50k	...	0.54%
▶ scripts	....	44k	...	0.47%

# Useful components

▶ block	.....	14k	...	0.15%
▶ crypto	.....	36k	...	0.38%
▶ drivers	...	4.3M	...	46%
▶ fs	.....	743k	...	7.9%
▶ ipc	.....	7293	...	0.078%
▶ net	.....	525k	...	5.6%
▶ security	...	36k	...	0.38%
▶ sound	.....	452k	...	4.8%
▶ virt	.....	2k	...	0.026%

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Other stuff

- ▶ Documentation ... 254k ... 2.7%
- ▶ samples ..... 1k ... 0.013%
- ▶ usr ..... 1k ... 0.0075%

Next...

Introduction

General Info

The Code

Q&A

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
bart@jukie.net

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

Introduction

General Info

Numbers

Files

Tree

The Code

**Types**

OO

Macros

API

Q&A

Variable types

# Base types

- ▶ C99 types
  - ▶ `bool` `char` `int` ...
  - ▶ `int8_t` ...
  - ▶ `uint8_t` ...
  
- ▶ kernel types
  - ▶ `size_t`
  - ▶ `s8` ...
  - ▶ `u8` ...
  
- ▶ userspace API
  - ▶ `__kernel_size_t`
  - ▶ `__s8` ...
  - ▶ `__u8` ...

# Base types

- ▶ C99 types
  - ▶ `bool` `char` `int` ...
  - ▶ `int8_t` ...
  - ▶ `uint8_t` ...
- ▶ kernel types
  - ▶ `size_t`
  - ▶ `s8` ...
  - ▶ `u8` ...
- ▶ userspace API
  - ▶ `__kernel_size_t`
  - ▶ `__s8` ...
  - ▶ `__u8` ...

# Base types

- ▶ C99 types
  - ▶ `bool` `char` `int` ...
  - ▶ `int8_t` ...
  - ▶ `uint8_t` ...
- ▶ kernel types
  - ▶ `size_t`
  - ▶ `s8` ...
  - ▶ `u8` ...
- ▶ userspace API
  - ▶ `__kernel_size_t`
  - ▶ `__s8` ...
  - ▶ `__u8` ...

# Attributes

- ▶ `const`, `pure`, `register`, `unsigned`, `volatile`, ...
- ▶ `asm` linkage, `inline`, `noinline`
- ▶ `__noreturn`
- ▶ `__aligned`, `__packed`
- ▶ `__read_mostly`

static analysis *checker* (`sparse`)

- ▶ `__user`, `__kernel`, `__iomem`
- ▶ `__must_check`, `__used`, `__nocast`
- ▶ `__force`, `__safe`
- ▶ `__deprecated`

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

**Types**

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Attributes

- ▶ `const`, `pure`, `register`, `unsigned`, `volatile`, ...
- ▶ `asm` linkage, `inline`, `noinline`
- ▶ `__noreturn`
- ▶ `__aligned`, `__packed`
- ▶ `__read_mostly`

static analysis *checker* (`sparse`)

- ▶ `__user`, `__kernel`, `__iomem`
- ▶ `__must_check`, `__used`, `__nocast`
- ▶ `__force`, `__safe`
- ▶ `__deprecated`

# Attributes

- ▶ `const`, `pure`, `register`, `unsigned`, `volatile`, ...
- ▶ `asm` linkage, `inline`, `noinline`
- ▶ `__noreturn`
- ▶ `__aligned`, `__packed`
- ▶ `__read_mostly`

static analysis *checker* (`sparse`)

- ▶ `__user`, `__kernel`, `__iomem`
- ▶ `__must_check`, `__used`, `__nocast`
- ▶ `__force`, `__safe`
- ▶ `__deprecated`

# Attributes

- ▶ `const`, `pure`, `register`, `unsigned`, `volatile`, ...
- ▶ `asm` linkage, `inline`, `noinline`
- ▶ `__noreturn`
- ▶ `__aligned`, `__packed`
- ▶ `__read_mostly`

## static analysis *checker* (`sparse`)

- ▶ `__user`, `__kernel`, `__iomem`
- ▶ `__must_check`, `__used`, `__nocast`
- ▶ `__force`, `__safe`
- ▶ `__deprecated`

# Structure initializers

```
struct foo {  
    int num;  
    const char *str;  
};
```

```
struct foo *foo = {  
    .num = 1,  
    .str = "foo"  
};
```

```
*foo = (struct foo){  
    .num = 2,  
    .str = "bar"  
};
```

# Structure initializers

```
struct foo {
    int num;
    const char *str;
};

struct foo *foo = {
    .num = 1,
    .str = "foo"
};

*foo = (struct foo){
    .num = 2,
    .str = "bar"
};
```

# Structure initializers

```
struct foo {
    int num;
    const char *str;
};

struct foo *foo = {
    .num = 1,
    .str = "foo"
};

*foo = (struct foo){
    .num = 2,
    .str = "bar"
};
```

"Object Flavored Programming"

## struct as object

- ▶ data fields
- ▶ pointer to *ops* structure
  - ▶ function pointers
- ▶ calling member functions
  - ▶ `ret = filp → f_op → close(filp, owner);`

## struct as object

- ▶ data fields
- ▶ pointer to *ops* structure
  - ▶ function pointers
- ▶ calling member functions
  - ▶ `ret = filp → f_op → close(filp, owner);`

- ▶ manual reference counting

```
kobj = kobject_get(kobj);  
if (kobj) {  
    // do stuff here  
    kobject_put(kobj);  
}
```

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

**Macros**

API

Q&A

Macro tricks

Useful macros

```
#define do_two_things(x) do { \  
    do_first_thing(x); \  
    do_second_thing(x); \  
} while(0)
```

- ▶ gcc also supports `{ ... }`

```
#define do_two_things(x) do { \  
    do_first_thing(x); \  
    do_second_thing(x); \  
} while(0)
```

- ▶ gcc also supports (`{ ... }`)

# Data type info

▶ `offsetof(type, member)`

▶ `container_of(ptr, type, member)`

```
struct foo {
    struct bar {
        ...
    } bar;
    ...
} foo;
...
void work_on_bar(struct bar *bar)
{
    struct foo *foo;
    foo = container_of(bar, struct foo, bar);
    ...
}
```

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Data type info

- ▶ `offsetof(type, member)`
- ▶ `container_of(ptr, type, member)`

```
struct foo {
    struct bar {
        ...
    } bar;
    ...
} foo;
...
void work_on_bar(struct bar *bar)
{
    struct foo *foo;
    foo = container_of(bar, struct foo, bar);
    ...
}
```

[Introduction](#)[General Info](#)[Numbers](#)[Files](#)[Tree](#)[The Code](#)[Types](#)[OO](#)[Macros](#)[API](#)[Q&A](#)

▶ `typecheck(type, x)`

```
struct foo;  
#define operate_on_foo(foo_ptr) \  
    typecheck(struct foo*, foo_ptr) \  
    ...
```

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

**API**

Q&A

Useful functions

# Almost like libc

```
printk(KERN_INFO "Hello world\n");
```

```
ptr = kmalloc(1024, GFP_USER);
```

```
ptr = kzalloc(1024, GFP_ATOMIC);
```

```
<linux/string.h> == <string.h>
```

# Bit operations

```
#include <linux/bitops.h>

set_bit(nr, ptr);

nr = find_first_bit(ptr, size);

if (test_and_clear_bit(nr, ptr))
    /* bit was set before */
```

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

**[API](#)**

[Q&A](#)

# Linked lists

```
#include <linux/list.h>

LIST_HEAD(list);

struct item { struct list_head link; } *item;

list_add(item, &list);

list_for_each(item, &list)
    /* operate on each item */
```

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

[API](#)

[Q&A](#)

# Concurrency

```
#include <linux/mutex.h>
DEFINE_MUTEX(mutex);

mutex_lock(&mutex);
mutex_unlock(&mutex);

if (mutex_trylock(&mutex) {
    ...
    mutex_unlock(&mutex);
}
```

[Introduction](#)

[General Info](#)

[Numbers](#)

[Files](#)

[Tree](#)

[The Code](#)

[Types](#)

[OO](#)

[Macros](#)

**[API](#)**

[Q&A](#)

# Spin locks

```
#include <linux/spinlock.h>
spinlock_t lock;

spin_lock(&lock);
spin_unlock(&lock);

spin_lock_bh(&lock);
spin_unlock_bh(&lock);

unsigned long flags;
spin_lock_irqsave(&lock, flags);
spin_unlock_irqrestore(&lock, flags);
```

# Spin locks

```
#include <linux/spinlock.h>
spinlock_t lock;

spin_lock(&lock);
spin_unlock(&lock);

spin_lock_bh(&lock);
spin_unlock_bh(&lock);

unsigned long flags;
spin_lock_irqsave(&lock, flags);
spin_unlock_irqrestore(&lock, flags);
```

# Spin locks

```
#include <linux/spinlock.h>
spinlock_t lock;

spin_lock(&lock);
spin_unlock(&lock);

spin_lock_bh(&lock);
spin_unlock_bh(&lock);

unsigned long flags;
spin_lock_irqsave(&lock, flags);
spin_unlock_irqrestore(&lock, flags);
```

Next...

Introduction

General Info

The Code

Q&A

Linux Kernel  
Walkthrough

**Bart**  
Trojanowski,  
**bart@jukie.net**

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

Q&A

# Q&A

**Bart**  
Trojanowski,  
[bart@jukiie.net](mailto:bart@jukiie.net)

Introduction

General Info

Numbers

Files

Tree

The Code

Types

OO

Macros

API

**Q&A**

What do you want to see?