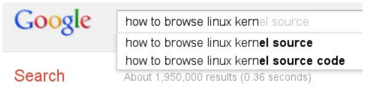
Browsing and Hacking the Linux Kernel with KDevelop

Alexandre Courbot, NVIDIA

June 8, 2012

About This Guy...

- Former academic in embedded OSes
- KDE user since 2003
- Employed by NVIDIA as a Tegra CE engineer
- Need to browse, study and understand kernel code quickly...
- ...and I know I am not alone


how to browse linux kernel source

how to browse linux kernel **source**

how to browse linux kernel **source code**

About 1,950,000 results (0.36 seconds)

My Journey with Code Editors

Like most people, I spent some time using the most popular code editors:

- Emacs user for 5 years
- Vim user for 4 years

Both are very nice and powerful. But the '80s called, and they want their UI back.

Why are we still using tools that are between 15 and 30 years old to browse and edit kernel code?

CTags/CScope

Supported by the kernel Makefile:

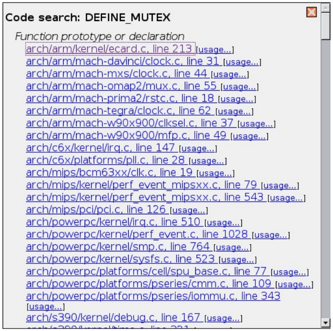
```
$ make tags
$ make cscope
```

- Output usable by virtually every editor
- Do a good job at filtering arch-relevant files. . .
- . . . but unfortunately include all mach
- . . . and not `.config`-aware
- No incremental update of tags
- Takes 40-seconds on this machine when kernel source is cached
- Sometimes not so smart

LXR, Linux Cross Reference

A HTML, browseable version of the Linux source code (`http://lxr.linux.no`).

- Web-based, not integrated into your editor
- Works with some reference source, not *yours*
- Parses *everything* and returns references about everything
- Parsing results sometimes totally wrong: every `DEFINE_MUTEX` use is interpreted as a re-declaration!



Code search: DEFINE_MUTEX

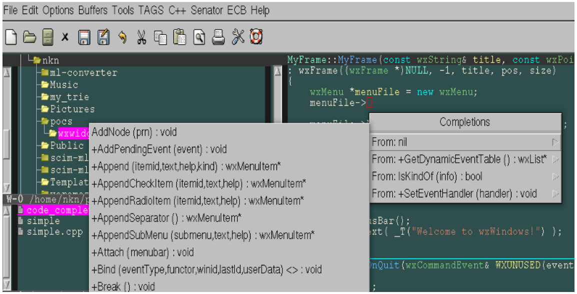
Function prototype or declaration

arch/arm/kernel/ecard.c, line 213 [usage...]
arch/arm/mach-davinci/clock.c, line 31 [usage...]
arch/arm/mach-mxs/clock.c, line 44 [usage...]
arch/arm/mach-omap2/mux.c, line 55 [usage...]
arch/arm/mach-prima2/rstc.c, line 18 [usage...]
arch/arm/mach-tegra/clock.c, line 62 [usage...]
arch/arm/mach-w90x900/clksel.c, line 37 [usage...]
arch/arm/mach-w90x900/mfp.c, line 49 [usage...]
arch/c6x/kernel/irq.c, line 147 [usage...]
arch/c6x/platforms/pll.c, line 28 [usage...]
arch/mips/bcm63xx/clk.c, line 19 [usage...]
arch/mips/kernel/perf_event_mipsxx.c, line 79 [usage...]
arch/mips/kernel/perf_event_mipsxx.c, line 543 [usage...]
arch/mips/pci/pci.c, line 126 [usage...]
arch/powerpc/kernel/irq.c, line 510 [usage...]
arch/powerpc/kernel/perf_event.c, line 1028 [usage...]
arch/powerpc/kernel/smp.c, line 764 [usage...]
arch/powerpc/kernel/sysfs.c, line 523 [usage...]
arch/powerpc/platforms/cell/spu_base.c, line 77 [usage...]
arch/powerpc/platforms/pseries/cmm.c, line 109 [usage...]
arch/powerpc/platforms/pseries/iommu.c, line 343 [usage...]
arch/s390/kernel/debug.c, line 167 [usage...]

CEDET (Emacs IDE)

Emacs-only Integrated Development Environment.

- Code completion
- Code browser
- ASCII UML diagrams
- . . .


File Edit Options Buffers Tools TAGS C++ Senator ECB Help
nkn
ml-converter
Music
my_trie
Pictures
pocs
Public
W-0 /home/nkn/
code_comple
simple
simple.cpp
AddNode (prn) : void
+AddPendingEvent (event) : void
+Append (itemid,text,help,kind) : wxMenuItem*
+AppendCheckItem (itemid,text,help) : wxMenuItem*
+AppendRadioItem (itemid,text,help) : wxMenuItem*
+AppendSeparator () : wxMenuItem*
+AppendSubMenu (submenu,text,help) : wxMenuItem*
+Attach (menubar) : void
+Bind (eventType,functor,winid,lastId,userData) <> : void
+Break () : void
Completions
From: nil
From: +GetDynamicEventTable () : wxList*
From: IsKindOf (info) : bool
From: +SetEventHandler (handler) : void

```
MyFrame::MyFrame(const wxString& title, const wxPoi
: wxFrame((wxFrame *)NULL, -1, title, pos, size)
{
    wxMenu *menuFile = new wxMenu;
    menuFile->
```

```
sBar();
xt( _T("Welcome to wxWindows!") );

nQuit(wxCommandEvent& WXUNUSED(event
;
```

CEDET (Emacs IDE)

A “Gentle” Introduction to CEDET

```
(defun my-cedet-hook ()
  (local-set-key [(control return)] 'semantic-ia-complete-symbol)
  (local-set-key "\C-c?" 'semantic-ia-complete-symbol-menu)
  (local-set-key "\C-c>" 'semantic-complete-analyze-inline)
  (local-set-key "\C-cp" 'semantic-analyze-proto-impl-toggle))
(add-hook 'c-mode-common-hook 'my-cedet-hook)
;; automatic name completion
(defun my-c-mode-cedet-hook ()
 (local-set-key "." 'semantic-complete-self-insert)
 (local-set-key ">" 'semantic-complete-self-insert))
(add-hook 'c-mode-common-hook 'my-c-mode-cedet-hook)
```

Now open a file, and admire your screen frozen for one minute.

What To Expect From a Modern IDE?

Things have changed since last century. We have multi-core machines, filesystem notifiers, modern UIs, SSD disks.

- Take advantage of modern hardware and kernel features: `inotify`, multi-core, ...
- Background parsing
- Incremental updates of modified files
- Easy to setup, minimal configuration
- Integrated into a decent editor
- Modern UI yet allow both keyboard and mouse control

Eclipse does that well for Java - why don't we have the same for C and kernel?

About KDE

A modern, slick set of libraries, core services, and applications based on Qt.

- Highly-configurable
- Reusable components (KParts)
- DBUS-controllable
- Very dynamic and listening community
- Kate text editor
  - Syntax highlighting for 200+ files types
  - Code folding
  - Split screen
  - Very configurable, extensible through plugins and scripts
  - VI mode!

About KDevelop

Project started in 1998, went through several rewrites.
KDevelop 4:

- In development since 2005, first stable release in 2010
- Leverages most KDE technologies
- Extremely modular and extensible
- Very powerful code analysis/browsing capabilities
- Probably the most overlooked Linux/C++ Linux IDE

What happens if we open the kernel source with it?

KDevelop and the Kernel

Most features working out of the box, but:

- KDevelop's parser is C++ **only**
- Code parsing extremely long, many unnecessary files parsed
- Include paths incorrect
- No kernel configuration awareness

Tuning KDevelop for the Kernel

Two-fronts work:

1. Make KDevelop more kernel-friendly
   - Make the parser capable of handling pure C
2. `kdev-kernel` plugin
   - Kernel project manager:
     - Parse configuration and Makefile to only consider source files relevant to the current configuration
     - Declare configuration macros to guide the C parser
     - Setup the correct include paths
   - Kernel builder
     - Integrate kernel configuration GUI
     - Handle out-of-source building and cross-compiling automatically

Guided Tour

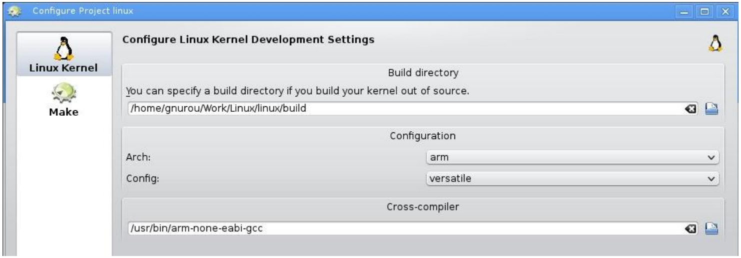
Let's see how this works!

Obvious Features

- Sessions
- Bookmarks
- Navigation history
- Customizable keyboard shortcuts
- Code snippets
- External scripts
- Customizable indentation rules
- Working files sets

Kernel Project Configuration

Kernel project configuration dialog: just choose your architecture and base configuration, and there you go!



Configure Project linux

Configure Linux Kernel Development Settings

Build directory

You can specify a build directory if you build your kernel out of source.

/home/gnurou/Work/Linux/linux/build

Configuration

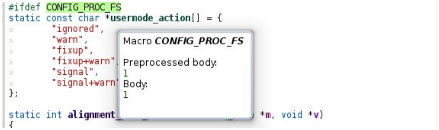
Arch: arm

Config: versatile

Cross-compiler

/usr/bin/arm-none-eabi-gcc

Code parsing then becomes configuration-aware



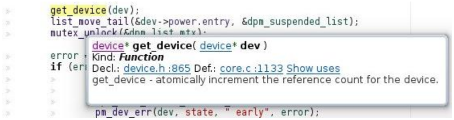
```
#ifdef CONFIG_PROC_FS
static const char *usermode_action[] = {
        "ignored",
        "warn",
        "fixup",
        "fixup+warn",
        "signal",
        "signal+warn"
};

static int alignment_                         *m, void *v)
{
```

Macro *CONFIG_PROC_FS*

Preprocessed body:
1
Body:
1

Inline Symbols Information

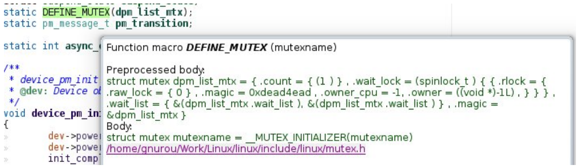
Links to symbols definitions, inline documentation



```
get_device(dev);
list_move_tail(&dev->power.entry, &dpm_suspended_list);
mutex_unlock(&dpm_list_mtx);

error
if (er

        pm_dev_err(dev, state, " early", error);
```

device* **get_device**(device* **dev**)
Kind: ***Function***
Decl.: device.h :865 Def.: core.c :1133 Show uses
get_device - atomically increment the reference count for the device.

Macro expansion and quick peeking



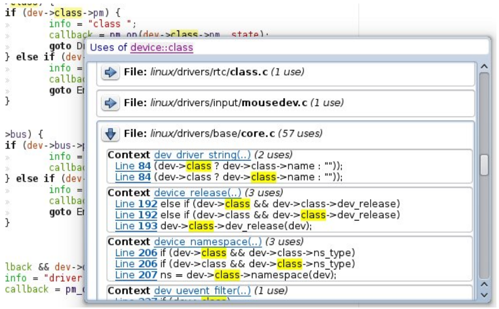
```
static DEFINE_MUTEX(dpm_list_mtx);
static pm_message_t pm_transition;

static int async_

/**
 * device_pm_init
 * @dev: Device o
 */
void device_pm_in
{
        dev->powe
        dev->powe
        init_comp
```

Function macro ***DEFINE_MUTEX*** (mutexname)

Preprocessed body:
struct mutex dpm_list_mtx = { .count = { (1) } , .wait_lock = (spinlock_t) { { .rlock = { .raw_lock = { 0 } , .magic = 0xdead4ead , .owner_cpu = -1, .owner = ((void *)-1L) , } } } , .wait_list = { &(dpm_list_mtx .wait_list), &(dpm_list_mtx .wait_list) } , .magic = &dpm_list_mtx }
Body:
struct mutex mutexname = __MUTEX_INITIALIZER(mutexname)
/home/gnurou/Work/Linux/linux/include/linux/mutex.h

Project-wide Uses of Symbols

Find exactly where a given symbol is used



```
if (dev->class->pm) {
        info = "class ";
        callback = pm_op(dev->class->pm, state);
        goto Di
} else if (dev
        info =
        callba
        goto E
}

>bus) {
if (dev->bus->
        info =
        callba
} else if (dev
        info =
        callba
        goto E
}

lback && dev->
info = "driver
callback = pm_
```

Uses of device::class

File: *linux*/drivers/rtc/**class.c** *(1 use)*

File: *linux*/drivers/input/**mousedev.c** *(1 use)*

File: *linux*/drivers/base/**core.c** *(57 uses)*

Context dev_driver_string(..) *(2 uses)*
Line **84** (dev->class ? dev->class->name : ""));
Line **84** (dev->class ? dev->class->name : ""));

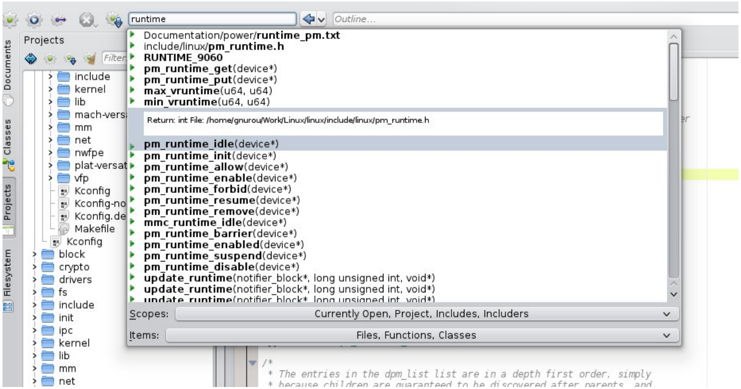
Context device_release(..) *(3 uses)*
Line **192** else if (dev->class && dev->class->dev_release)
Line **192** else if (dev->class && dev->class->dev_release)
Line **193** dev->class->dev_release(dev);

Context device_namespace(..) *(3 uses)*
Line **206** if (dev->class && dev->class->ns_type)
Line **206** if (dev->class && dev->class->ns_type)
Line **207** ns = dev->class->namespace(dev);

Context dev_uevent_filter(..) *(1 use)*

Quick Open

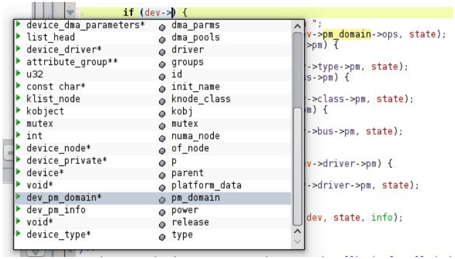
Quickly find any file, function of struct across the project



```
Documentation/power/runtime_pm.txt
include/linux/pm_runtime.h
RUNTIME_9060
pm_runtime_get(device*)
pm_runtime_put(device*)
max_vruntime(u64, u64)
min_vruntime(u64, u64)
Return: int File: /home/gnurou/Work/Linux/linux/include/linux/pm_runtime.h
pm_runtime_idle(device*)
pm_runtime_init(device*)
pm_runtime_allow(device*)
pm_runtime_enable(device*)
pm_runtime_forbid(device*)
pm_runtime_resume(device*)
pm_runtime_remove(device*)
mmc_runtime_idle(device*)
pm_runtime_barrier(device*)
pm_runtime_enabled(device*)
pm_runtime_suspend(device*)
pm_runtime_disable(device*)
update_runtime(notifier_block*, long unsigned int, void*)
update_runtime(notifier_block*, long unsigned int, void*)
update_runtime(notifier_block*, long unsigned int, void*)
Scopes: Currently Open, Project, Includes, Includers
Items: Files, Functions, Classes
/*
 * The entries in the dpm_list list are in a depth first order, simply
 * because children are guaranteed to be discovered after parents, and
```

Code Editing

Smart code completion

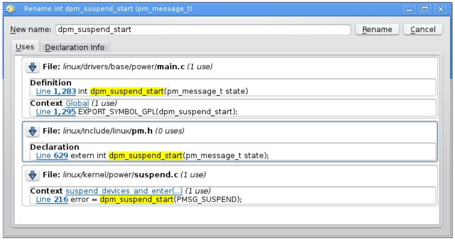


```
if (dev->) {
```

device_dma_parameters*	dma_parms
list_head	dma_pools
device_driver*	driver
attribute_group**	groups
u32	id
const char*	init_name
klist_node	knode_class
kobject	kobj
mutex	mutex
int	numa_node
device_node*	of_node
device_private*	p
device*	parent
void*	platform_data
dev_pm_domain*	pm_domain
dev_pm_info	power
void*	release
device_type*	type

```
";
v->pm_domain->ops, state);
>pm) {
->type->pm, state);
s->pm) {
->class->pm, state);
m) {
->bus->pm, state);
v->driver->pm) {
->driver->pm, state);
dev, state, info);
```

Refactoring



Rename int dpm_suspend_start (pm_message_t)

New name: dpm_suspend_start Rename Cancel

Uses Declaration Info

File: *linux/drivers/base/power/***main.c** *(1 use)*

Definition
Line 1,283 int dpm_suspend_start(pm_message_t state)

Context Global *(1 use)*
Line 1,295 EXPORT_SYMBOL_GPL(dpm_suspend_start);

File: *linux/include/linux/***pm.h** *(0 uses)*

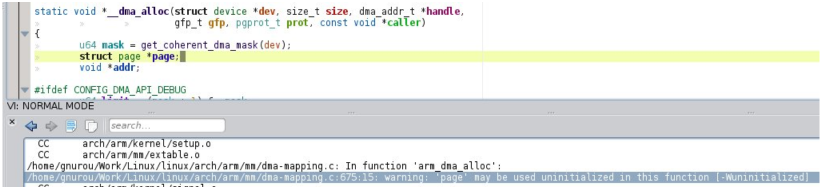
Declaration
Line 629 extern int dpm_suspend_start(pm_message_t state);

File: *linux/kernel/power/***suspend.c** *(1 use)*

Context suspend_devices_and_enter(..) *(1 use)*
Line 216 error = dpm_suspend_start(PMSG_SUSPEND);

Integrated Configuration and Compilation

Integrated `make` invoked with the right parameters and linked output

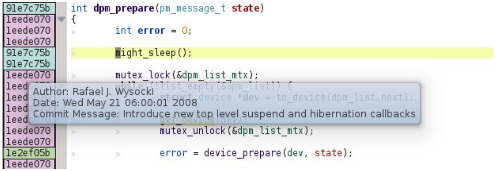


```
static void *__dma_alloc(struct device *dev, size_t size, dma_addr_t *handle,
                         gfp_t gfp, pgprot_t prot, const void *caller)
{
        u64 mask = get_coherent_dma_mask(dev);
        struct page *page;
        void *addr;

#ifdef CONFIG_DMA_API_DEBUG
```

```
  CC      arch/arm/kernel/setup.o
  CC      arch/arm/mm/extable.o
/home/gnurou/Work/Linux/linux/arch/arm/mm/dma-mapping.c: In function 'arm_dma_alloc':
/home/gnurou/Work/Linux/linux/arch/arm/mm/dma-mapping.c:675:15: warning: 'page' may be used uninitialized in this function [-Wuninitialized]
```

Git Integration

`git blame` support



```
91e7c75b  int dpm_prepare(pm_message_t state)
1eede070  {
1eede070          int error = 0;
1eede070
91e7c75b          might_sleep();
91e7c75b
1eede070          mutex_lock(&dpm_list_mtx);
leed              while (!list_empty(&dpm_list)) {
leed                      struct device *dev = to_device(dpm_list.next);
leed
leed                      get_device(dev);
1eede070                  mutex_unlock(&dpm_list_mtx);
1eede070
1e2ef05b                  error = device_prepare(dev, state);
1eede070
```

Author: Rafael J. Wysocki
Date: Wed May 21 06:00:01 2008
Commit Message: Introduce new top level suspend and hibernation callbacks

Future Work

Both KDevelop and `kdev-kernel` are works in progress.

- Complete C support / Fix parser errors
- Improve background parsing speed
- Function pointers analysis
- Debugger integration
- Support for out-of-tree kernel modules
- Static analysis tools (call graphs, . . .)
- . . .

KDevelop is a fun project to hack on

- Usable on a daily basis and first-of-his-class on a lot of features
- Yet potential to implement many cool things!

Thank you!

KDevelop Official Website

- `http://www.kdevelop.org`

Kernel Tailored Branch and `kdev-kernel` Plugin

- `https://github.com/Gnurou/kdevelop`
- `https://github.com/Gnurou/kdev-kernel`

Feel free to contact me on Github if you have trouble setting up!
Bug reports, features requests, and patches are very welcome.