Chasing the Penguin: State and Evolution of the Kernel

Wolfgang Mauerer

MPRG IOIP & linux-kernel.net

10. October 2008

Dynamics of Kernel Development

Kernel Documentation

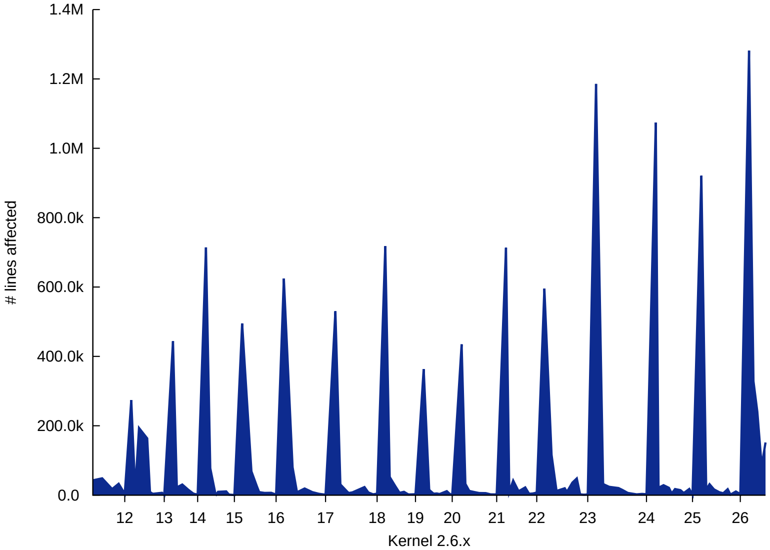
Understanding the Kernel

- Documenting new Features
- Analysis Tools

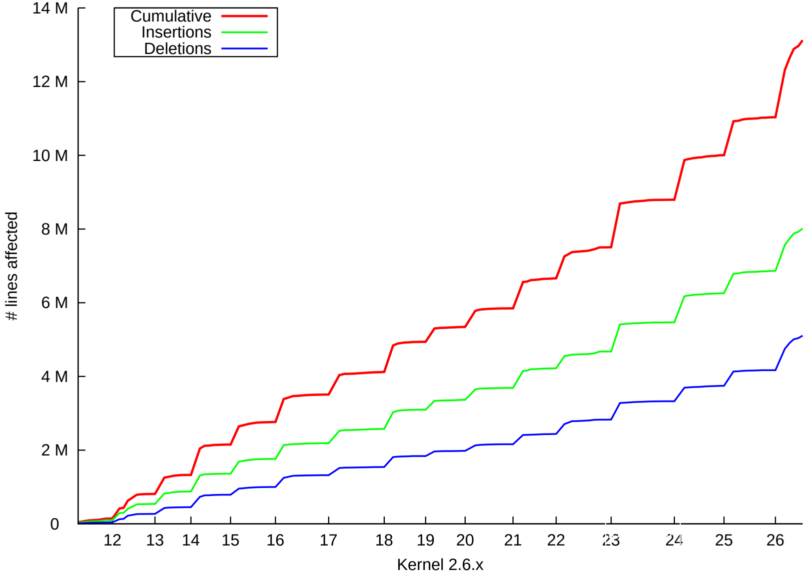
Social Aspects

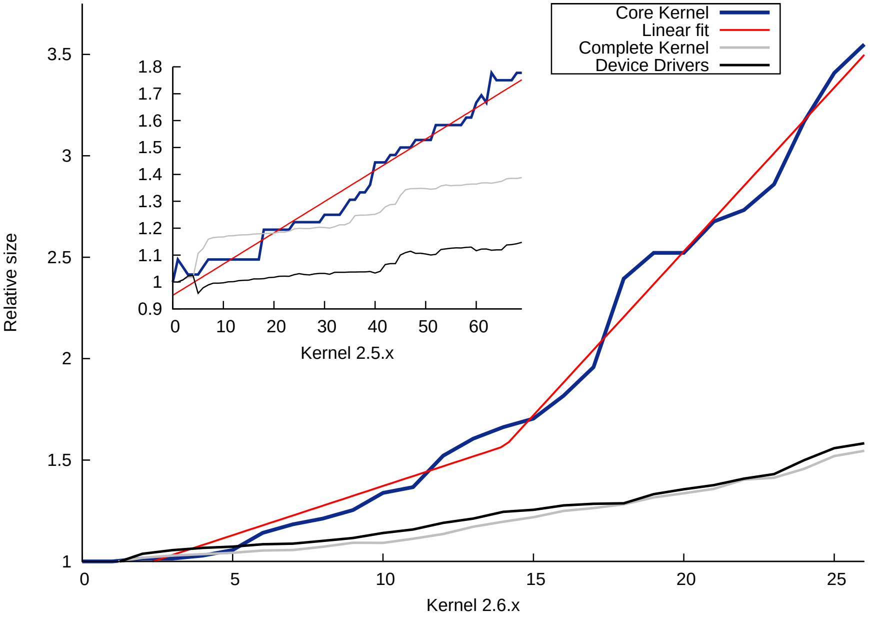
Outline

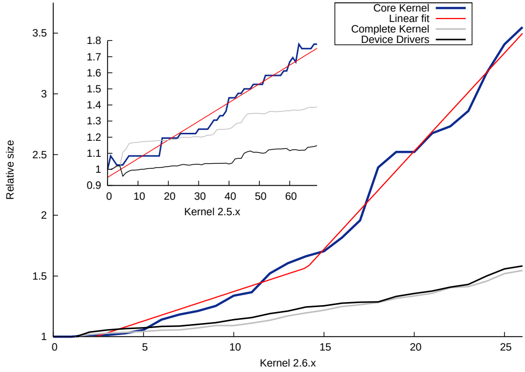
Dynamics of Kernel Development



Dynamics of Kernel Development

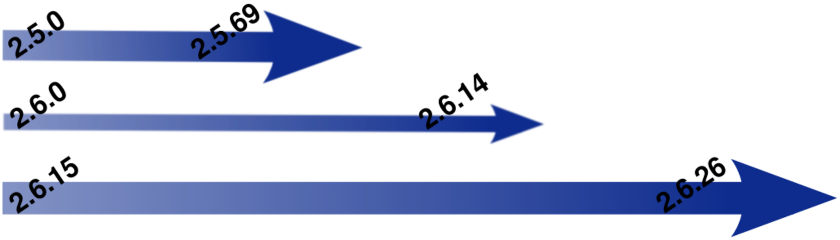


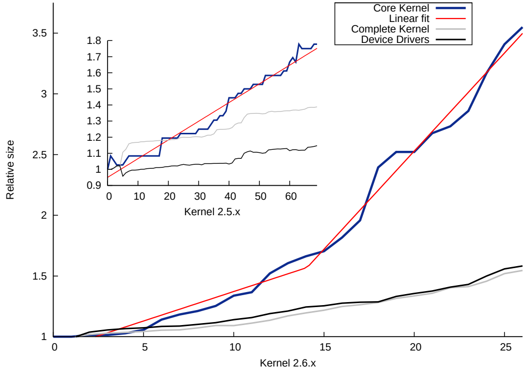

Core Kernel
Linear fit
Complete Kernel
Device Drivers
Relative size
3.5
3
2.5
2
1.5
1
0
5
10
15
20
25
Kernel 2.6.x
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0
10
20
30
40
50
60
Kernel 2.5.x



Hare and Tortoise

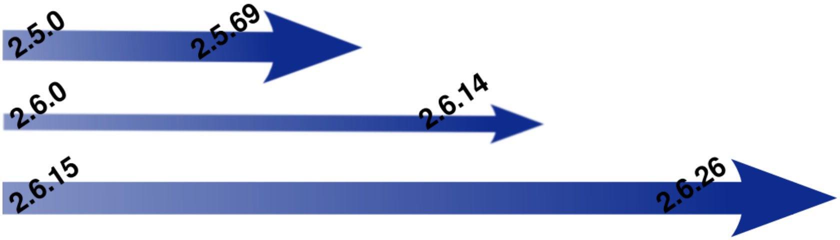
- Code sufficient?
- How to document?
- Which parts?





Hare and Tortoise

- Code sufficient?
- How to document?
- Which parts?



Outline

What's available?

In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles

What's available?

In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles

```
/**
* clocksource_khz2mult - calculates mult from khz and shift
* @khz:  Clocksource frequency in KHz
* @shift_constant:  Clocksource shift factor
*
* Helper functions that converts a khz counter frequency to a timsource
* multiplier, given the clocksource shift value
*/
static inline u32 clocksource_khz2mult(u32 khz, u32 shift_constant)
...
```

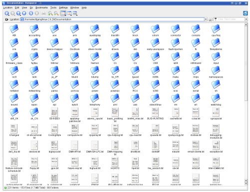
What’s available?

In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles



```
sx.txt - specialix SX/SI multiport serial driver readme.
Copyright (C) 1997 Roger Wolff (R.E.Wolff@BitWizard.nl)
...
Introduction
============

This file contains some random information, that I like to have online
instead of in a manual that can get lost.  Ever misplace your Linux
kernel sources?  And the manual of one of the boards in your computer?
...
```

What’s available?

In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles

```
commit 2087a1ad822cd3a68b73338457047fcc54da726b
Author:  Gregory Haskins <ghaskins@novell.com>
Date:  Fri Jun 27 14:30:00 2008 -0600

sched:  add avg-overlap support to RT tasks

We have the notion of tracking process-coupling (a.k.a.  buddy-wake) via
the p->se.last_wake / p->se.avg_overlap facilities, but it is only used
for cfs to cfs interactions.  There is no reason why an rt to cfs
interaction cannot share in establishing a relationhip in a similar
manner.

Because PREEMPT_RT runs many kernel threads as FIFO priority, we often
times have heavy interaction between RT threads waking CFS applications.
This patch offers a substantial boost (50-60%+) in perfomance under those
circumstances.
```

What's available?

In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles

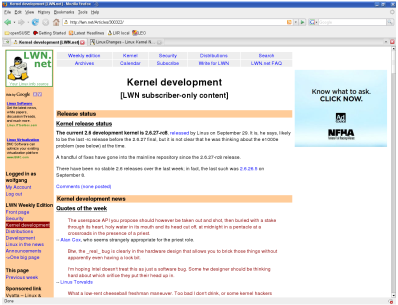
What’s available?

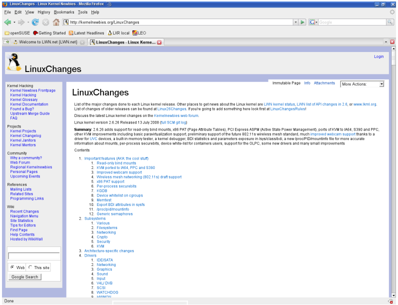
In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles





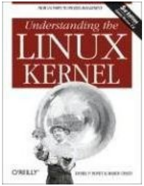
What's available?

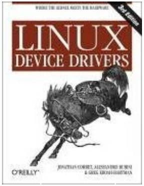
In-Tree

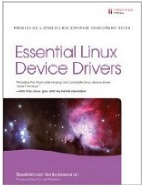
- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles







What's available?

In-Tree

- Comments and Kerneldoc
- `Documentation/`
- Git commit messages

External

- LKML and others
- Websites: `lwn.net`, `kernelnewbies.org`, ...
- Books and Articles

Problems

- Available? Location?
- Uptodate? Complete?

Summary

✓ What's good

- Huge amount of documentation available
- Implicit documentation in git
- Documentation infrastructure available

✗ What's bad

- Focus on people already intimate with the code
- Implicit documentation in git
- No consistent style
- Very fragmented and scattered

Outline

Dynamics of Kernel Development

Kernel Documentation

Understanding the Kernel
- Documenting new Features
- Analysis Tools

Social Aspects

Documenting new features

Completely Fair Scheduler

- Turbulent emergence
- Completely *replaces* old scheduler
- Considerable in-tree development after merge

High Resolution Timers

- Long external development
- New foundation for *existing* framework
- Merged at very mature state

Opposite strategies. . .

. . . also with respect to documentation!

Documenting new features

Completely Fair Scheduler

- Turbulent emergence
- Completely *replaces* old scheduler
- Considerable in-tree development after merge

High Resolution Timers

- Long external development
- New foundation for *existing* framework
- Merged at very mature state

Opposite strategies…

…also with respect to documentation!

Documenting new features

Completely Fair Scheduler

- Turbulent emergence
- Completely *replaces* old scheduler
- Considerable in-tree development after merge

High Resolution Timers

- Long external development
- New foundation for *existing* framework
- Merged at very mature state

Opposite strategies…

…also with respect to documentation!

Documenting new features

Completely Fair Scheduler

- Turbulent emergence
- Completely *replaces* old scheduler
- Considerable in-tree development after merge

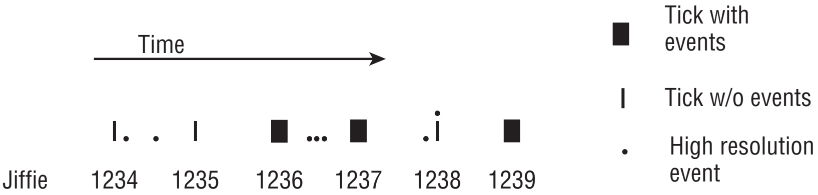
High Resolution Timers

- Long external development
- New foundation for *existing* framework
- Merged at very mature state

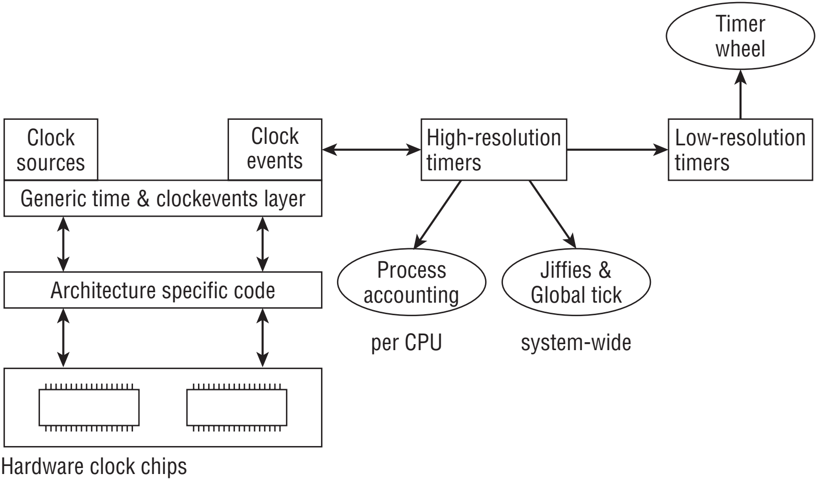
Opposite strategies. . .

. . . also with respect to documentation!

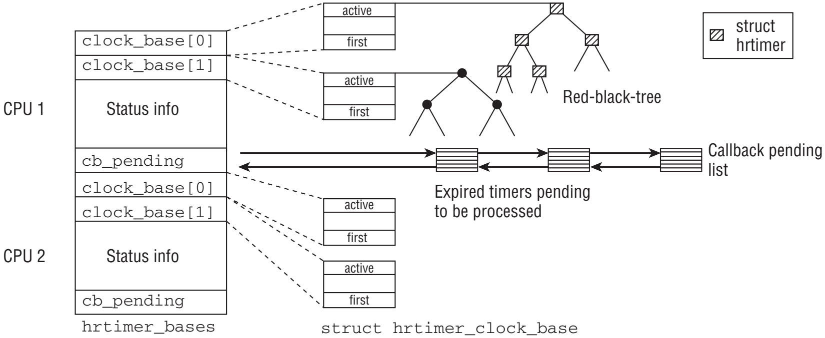
High Resolution Timers



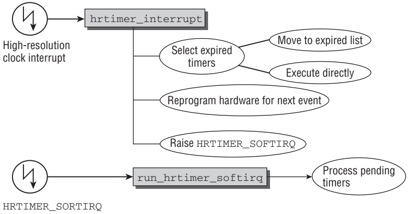
High Resolution Timers



High Resolution Timers



```c
void hrtimer_interrupt(struct clock_event_device *dev)
{
        struct hrtimer_cpu_base *cpu_base = &__get_cpu_var(hrtimer_bases);
        struct hrtimer_clock_base *base;
        ktime_t expires_next, now;
        int i, raise = 0;

        BUG_ON(!cpu_base->hres_active);
        cpu_base->nr_events++;
        dev->next_event.tv64 = KTIME_MAX;

 retry:
        now = ktime_get();

        expires_next.tv64 = KTIME_MAX;

        base = cpu_base->clock_base;

        for (i = 0; i < HRTIMER_MAX_CLOCK_BASES; i++) {
                ktime_t basenow;
                struct rb_node *node;

                spin_lock(&cpu_base->lock);

                basenow = ktime_add(now, base->offset);

                while ((node = base->first)) {
                        struct hrtimer *timer;

                        timer = rb_entry(node, struct hrtimer, node);

                        if (basenow.tv64 < timer->expires.tv64) {
                                ktime_t expires;

                                expires = ktime_sub(timer->expires,
                                                    base->offset);
                                if (expires.tv64 < expires_next.tv64)
                                        expires_next = expires;
                                break;
                        }

                        /* Move softirq callbacks to the pending list */
                        if (timer->cb_mode == HRTIMER_CB_SOFTIRQ) {
                                __remove_hrtimer(timer, base,
                                                 HRTIMER_STATE_PENDING, 0);
                                list_add_tail(&timer->cb_entry,
                                              &base->cpu_base->cb_pending);
                                raise = 1;
                                continue;
                        }

                        __run_hrtimer(timer);
                }
                spin_unlock(&cpu_base->lock);
                base++;
        }

        cpu_base->expires_next = expires_next;

        /* Reprogramming necessary ? */
        if (expires_next.tv64 != KTIME_MAX) {
                if (tick_program_event(expires_next, 0))
                        goto retry;
        }

        /* Raise softirq ? */
        if (raise)
                raise_softirq(HRTIMER_SOFTIRQ);
}
```



High Resolution Timers

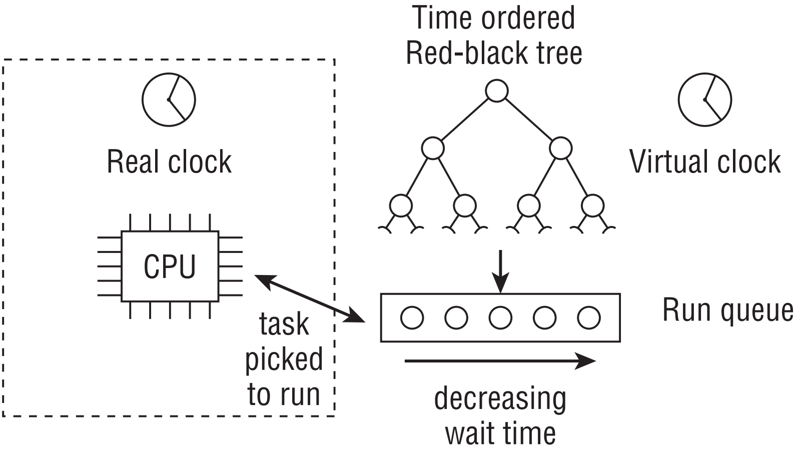
Available

- Orthogonal patch structure
- Component submission
- Design Documentation

Challenges

- Introduce conceptual parts
- Disentangle alternatives
- Prioritise important against unimportant code

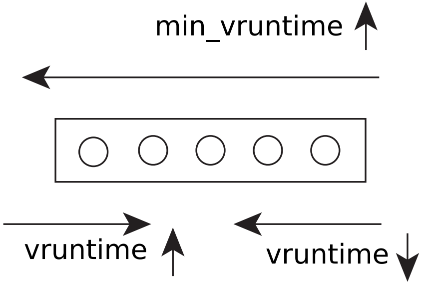
Completely Fair Scheduler



Completely Fair Scheduler

Virtual clock replacement

```
static inline s64 entity_key(struct cfs_rq *cfs_rq,
                             struct sched_entity *se) {
       return se->vruntime - cfs_rq->min_vruntime;
}
```



Completely Fair Scheduler

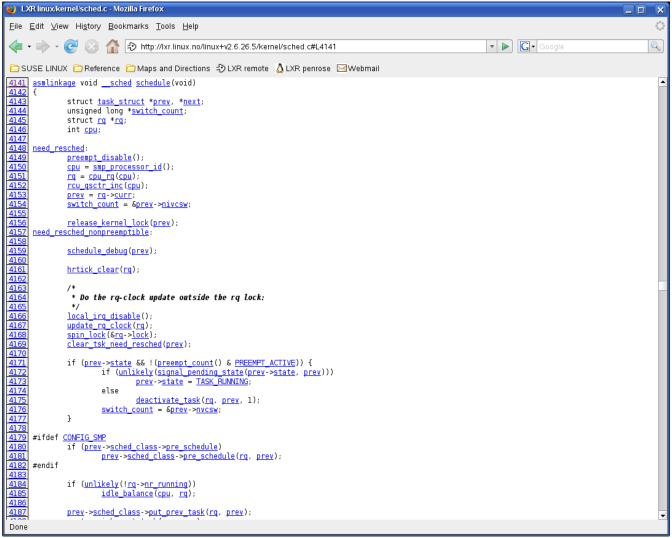
Conclusions

- Code history can ease documentation
- Identify stable components
- Reduce complexity

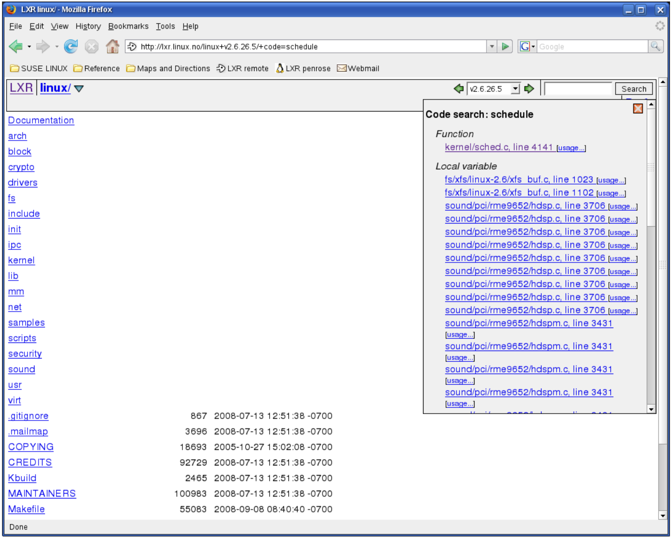
Analysis Tools

Analysis Tools

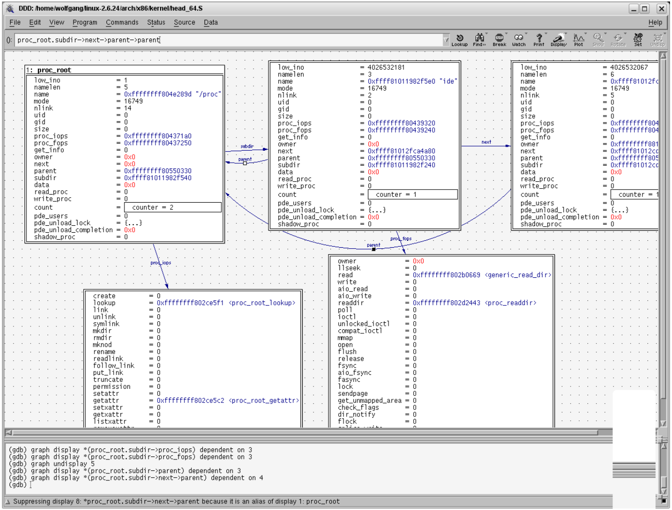
LXR Source Code Cross Reference



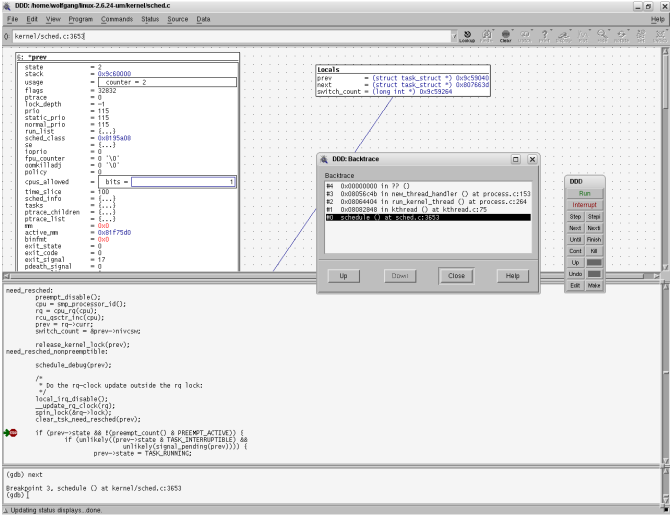
LXR Source Code Cross Reference



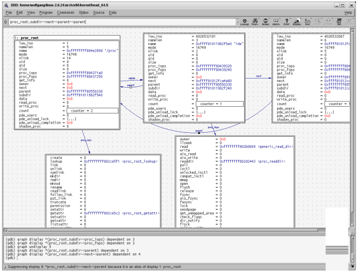
(K)GDB and DDD

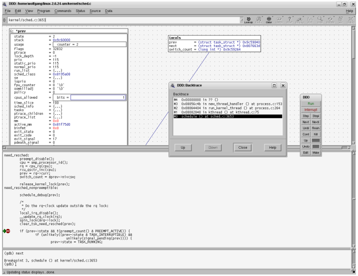

DDD: /home/wolfgang/linux-2.6.24/arch/x86/kernel/head_64.S
File Edit View Program Commands Status Source Data Help
proc_root.subdir->next->parent->parent
1: proc_root
(gdb) graph display *(proc_root.subdir->proc_iops) dependent on 3
(gdb) graph display *(proc_root.subdir->proc_fops) dependent on 3
(gdb) graph undisplay 5
(gdb) graph display *(proc_root.subdir->parent) dependent on 3
(gdb) graph display *(proc_root.subdir->next->parent) dependent on 4
(gdb)
Suppressing display 8: *proc_root.subdir->next->parent because it is an alias of display 1: proc_root

(K)GDB and DDD


DDD: /home/wolfgang/linux-2.6.24-um/kernel/sched.c
File Edit View Program Commands Status Source Data Help
kernel/sched.c:3653
Locals
prev = (struct task_struct *) 0x9c59040
next = (struct task_struct *) 0x807663d
switch_count = (long int *) 0x9c59264
DDD: Backtrace
Backtrace
#4 0x00000000 in ?? ()
#3 0x08056c4b in new_thread_handler () at process.c:153
#2 0x08064404 in run_kernel_thread () at process.c:264
#1 0x08082848 in kthread () at kthread.c:75
#0 schedule () at sched.c:3653
Up Down Close Help
DDD
Run Interrupt Step Stepi Next Nexti Until Finish Cont Kill Up Undo Edit Make
need_resched:
preempt_disable();
cpu = smp_processor_id();
rq = cpu_rq(cpu);
rcu_qsctr_inc(cpu);
prev = rq->curr;
switch_count = &prev->nivcsw;
release_kernel_lock(prev);
need_resched_nonpreemptible:
schedule_debug(prev);
* Do the rq-clock update outside the rq lock:
local_irq_disable();
__update_rq_clock(rq);
spin_lock(&rq->lock);
clear_tsk_need_resched(prev);
if (prev->state && !(preempt_count() & PREEMPT_ACTIVE)) {
if (unlikely((prev->state & TASK_INTERRUPTIBLE) &&
unlikely(signal_pending(prev)))) {
prev->state = TASK_RUNNING;
(gdb) next
Breakpoint 3, schedule () at kernel/sched.c:3653
(gdb)
Updating status displays...done.

(K)GDB and DDD

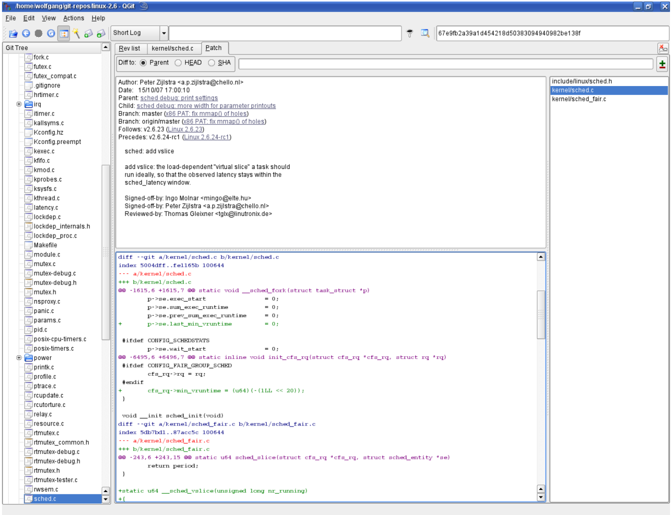




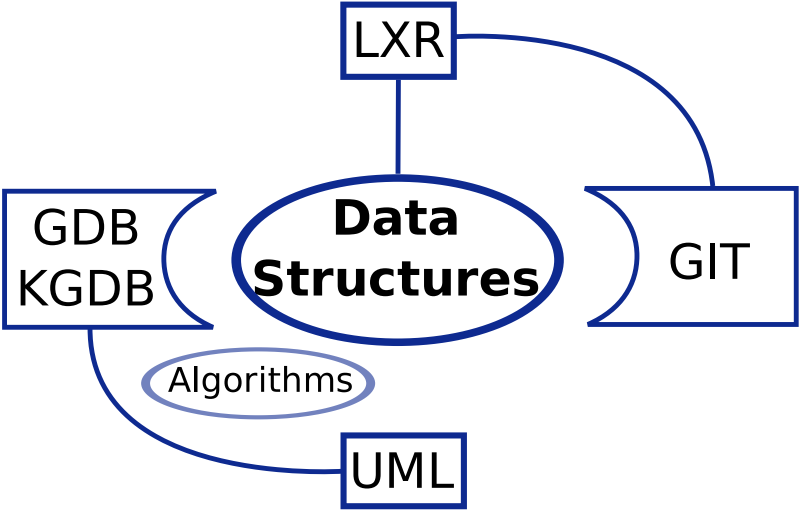
Andrew Morton on KGDB

I used kgdb continuously for 4-5 years until it broke. I don't think I ever used it much for "debugging" as such. I used it more for general observation of what's going on in the kernel. And for *confirmation* of what's going on (ie: testing that the actual state matches the expected state).

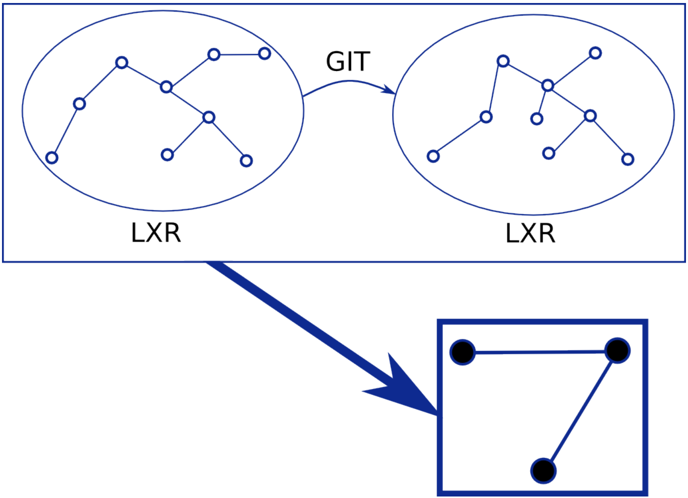
Git and GUIs


/home/wolfgang/git-repos/linux-2.6 - QGit
File Edit View Actions Help
Short Log
67e9fb2a39a1d454218d50383094940982be138f
Git Tree
Rev list
kernel/sched.c
Patch
Diff to: Parent HEAD SHA
Author: Peter Zijlstra <a.p.zijlstra@chello.nl>
Date: 15/10/07 17:00:10
Parent: sched debug: print settings
Child: sched debug: more width for parameter printouts
Branch: master (x86 PAT: fix mmap() of holes)
Branch: origin/master (x86 PAT: fix mmap() of holes)
Follows: v2.6.23 (Linux 2.6.23)
Precedes: v2.6.24-rc1 (Linux 2.6.24-rc1)
sched: add vslice
add vslice: the load-dependent "virtual slice" a task should run ideally, so that the observed latency stays within the sched_latency window.
Signed-off-by: Ingo Molnar <mingo@elte.hu>
Signed-off-by: Peter Zijlstra <a.p.zijlstra@chello.nl>
Reviewed-by: Thomas Gleixner <tglx@linutronix.de>
include/linux/sched.h
kernel/sched.c
kernel/sched_fair.c
diff --git a/kernel/sched.c b/kernel/sched.c
index 5004dff..fe1165b 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -1615,6 +1615,7 @@ static void __sched_fork(struct task_struct *p)
p->se.exec_start = 0;
p->se.sum_exec_runtime = 0;
p->se.prev_sum_exec_runtime = 0;
+ p->se.last_min_vruntime = 0;
#ifdef CONFIG_SCHEDSTATS
p->se.wait_start = 0;
@@ -6495,6 +6496,7 @@ static inline void init_cfs_rq(struct cfs_rq *cfs_rq, struct rq *rq)
#ifdef CONFIG_FAIR_GROUP_SCHED
cfs_rq->rq = rq;
#endif
+ cfs_rq->min_vruntime = (u64)(-(1LL << 20));
}
void __init sched_init(void)
diff --git a/kernel/sched_fair.c b/kernel/sched_fair.c
index 5db7bd1..87acc5c 100644
--- a/kernel/sched_fair.c
+++ b/kernel/sched_fair.c
@@ -243,6 +243,15 @@ static u64 sched_slice(struct cfs_rq *cfs_rq, struct sched_entity *se)
return period;
}
+static u64 __sched_vslice(unsigned long nr_running)
+{
fork.c
futex.c
futex_compat.c
.gitignore
hrtimer.c
irq
itimer.c
kallsyms.c
Kconfig.hz
Kconfig.preempt
kexec.c
kfifo.c
kmod.c
kprobes.c
ksysfs.c
kthread.c
latency.c
lockdep.c
lockdep_internals.h
lockdep_proc.c
Makefile
module.c
mutex.c
mutex-debug.c
mutex-debug.h
mutex.h
nsproxy.c
panic.c
params.c
pid.c
posix-cpu-timers.c
posix-timers.c
power
printk.c
profile.c
ptrace.c
rcupdate.c
rcutorture.c
relay.c
resource.c
rtmutex.c
rtmutex_common.h
rtmutex-debug.c
rtmutex-debug.h
rtmutex.h
rtmutex-tester.c
rwsem.c
sched.c

How it all fits together



How it all fits together



Outline

From: Con Kolivas

```
Ooh you have a vm patch that helps swap on the desktop!  I can help you
here with my experience from swap prefetch.

1.  Get it reviewed and have noone show any evidence it harms
2.  Find hundreds of users who can testify it helps
3.  Find a way of quantifying it.
4.  ...
5.  Merge into mainline.

There, that should get you as far as 4.  I haven't figured out what 4

is yet.  I believe it may be goto 1;
```

From: Con Kolivas

```
Ooh you have a vm patch that helps swap on the desktop!  I can help you
here with my experience from swap prefetch.

1.  Get it reviewed and have noone show any evidence it harms
2.  Find hundreds of users who can testify it helps
3.  Find a way of quantifying it.
4.  ...
5.  Merge into mainline.

There, that should get you as far as 4.  I haven't figured out what 4

is yet.  I believe it may be goto 1;
```

Linus on smelly pets

Ok, so now that I've insulted you and your pets (they're ugly!), show me wrong, and then call me a d*ckhead. ("Linus - you're a d*ckhead, and you didn't understand the problem, so you're a *stupid* d*ckhead. And my pet may be ugly, but yours *smells* bad!").

From: Rusty Russel

```
-#define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))
+#define ARRAY_SIZE(arr) (sizeof(arr) / sizeof((arr)[0])       \
+ + sizeof(typeof(int[1 - 2*!!__builtin_types_compatible_p(typeof(arr), \
+  typeof(&arr[0]))]))*0)
```

Reply from Linus Torvalds

```
Rusty, that's a work of art.
However, I would suggest that you never show it to anybody ever
again. I'm sure that in fifty years, it will be worth much more. So
please keep it tightly under wraps, to keep people from gouging their
eyes out^W^W^W^W^W^W^W make a killing in the art market.
```

Improved patch

```
OK, many people complained that it needed a comment.  Good point!
==
Add comment to ARRAY_SIZE macro.
diff -r 933e410f204f include/linux/kernel.h
--- a/include/linux/kernel.h Sat Mar 10 09:55:31 2007 +1100
+++ b/include/linux/kernel.h Sat Mar 10 09:55:53 2007 +1100
 <at>  <at>  -35,6 +35,7  <at>  <at>  extern const char linux_proc_banner[];
 #define ALIGN(x,a) __ALIGN_MASK(x,(typeof(x))(a)-1)
 #define __ALIGN_MASK(x,mask) (((x)+(mask))&~(mask))
+/* GCC is awesome. */
 #define ARRAY_SIZE(arr) (sizeof(arr) / sizeof((arr)[0])       \
  + sizeof(typeof(int[1 - 2*!!__builtin_types_compatible_p(typeof(arr), \
   typeof(&arr[0]))]))*0)
```

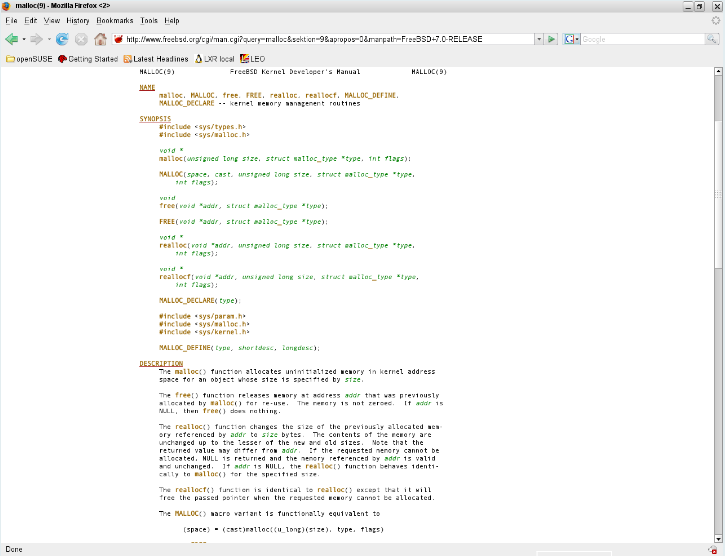
Thanks for you attention!

A Comparison: Allocating Memory

Everybody needs memory

- Core OS service
- Stable interface (introduced $\approx v0.98$)
- Documentation situation representative

FreeBSD




```
MALLOC(9)          FreeBSD Kernel Developer's Manual          MALLOC(9)

NAME
     malloc, MALLOC, free, FREE, realloc, reallocf, MALLOC_DEFINE,
     MALLOC_DECLARE -- kernel memory management routines

SYNOPSIS
     #include <sys/types.h>
     #include <sys/malloc.h>

     void *
     malloc(unsigned long size, struct malloc_type *type, int flags);

     MALLOC(space, cast, unsigned long size, struct malloc_type *type,
         int flags);

     void
     free(void *addr, struct malloc_type *type);

     FREE(void *addr, struct malloc_type *type);

     void *
     realloc(void *addr, unsigned long size, struct malloc_type *type,
         int flags);

     void *
     reallocf(void *addr, unsigned long size, struct malloc_type *type,
         int flags);

     MALLOC_DECLARE(type);

     #include <sys/param.h>
     #include <sys/malloc.h>
     #include <sys/kernel.h>

     MALLOC_DEFINE(type, shortdesc, longdesc);

DESCRIPTION
     The malloc() function allocates uninitialized memory in kernel address
     space for an object whose size is specified by size.

     The free() function releases memory at address addr that was previously
     allocated by malloc() for re-use.  The memory is not zeroed.  If addr is
     NULL, then free() does nothing.

     The realloc() function changes the size of the previously allocated mem-
     ory referenced by addr to size bytes.  The contents of the memory are
     unchanged up to the lesser of the new and old sizes.  Note that the
     returned value may differ from addr.  If the requested memory cannot be
     allocated, NULL is returned and the memory referenced by addr is valid
     and unchanged.  If addr is NULL, the realloc() function behaves identi-
     cally to malloc() for the specified size.

     The reallocf() function is identical to realloc() except that it will
     free the passed pointer when the requested memory cannot be allocated.

     The MALLOC() macro variant is functionally equivalent to

           (space) = (cast)malloc((u_long)(size), type, flags)
```

www.freebsd.org/cgi/man.cgi?query=malloc\&apropos=0\&sektion=0\&manpath=FreeBSD+7.0-RELEASE\&format=html

Mac OS

I/O Kit Device Driver Design Guidelines

PDF

Object Creation and Destruction

Because exceptions are excluded from the kernel's restricted form of C++, you cannot implement "normal" C++ constructors and destructors without jeopardy. Constructors and destructors are typed to return no value (such as an error code). Normally, if they encounter a problem, they raise an exception. But because exceptions aren't supported in the kernel's C++ runtime, there is no way for you to know when an allocation or deallocation error has occurred.

This situation prompted a design feature of the libkern's C++ runtime system that uses OSMetaClass macros to specify the structure of a class—that is, the metaclass data structures and functional interfaces—for the runtime typing system. The macros also define the primary constructor and a destructor for a class. These macro-created constructors are guaranteed not to fail because they do not themselves perform any allocations. Instead, the runtime system defers the actual allocation of objects until their initialization (usually in the `init` member function). Because the `init` function is typed to return a `bool`, it makes it possible to return an error upon any failure.

In this section:

Using the OSMetaClass Constructor Macros
Allocating Objects Dynamically
Global Initializers

Using the OSMetaClass Constructor Macros

When you create a C++ class based on OSObject, your code must call a matching pair of macros based upon the OSMetaClass class. The calls must be among the first statements in both the definition and implementation of the class. These macros are critical to your class because they enter metaclass information about it into the libkern runtime typing facility and define the static constructor and destructor for your class.

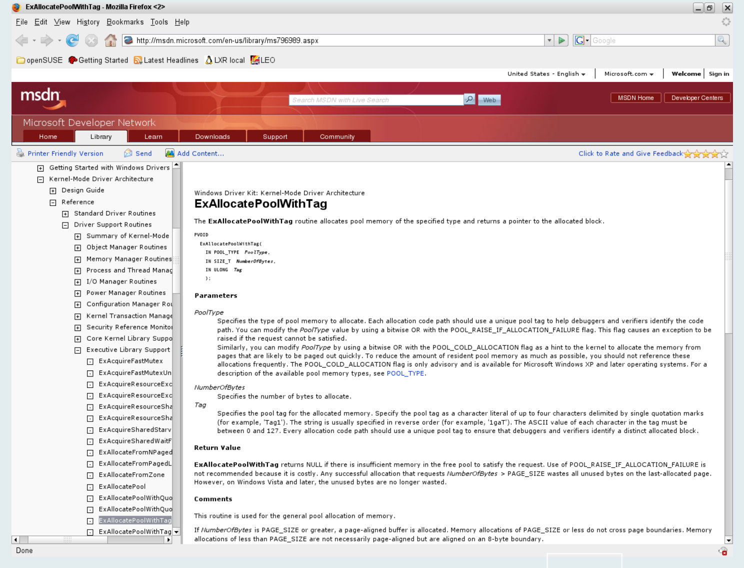
For concrete (that is, non-abstract) classes, the first macro, `OSDeclareDefaultStructors` declares the C++ constructors; by convention you insert this macro as the first element of the class declaration in the header file. For example:

```
class com_MyCompany_driver_MyDriver : public IOService
{
    OSDeclareDefaultStructors(com_MyCompany_driver_MyDriver);
    /* ... */
};
```

Your class implementation must include the companion "define" macro, `OSDefineMetaClassAndStructors`. This macro defines the constructor and destructor, implements the OSMetaClass allocation member function (`alloc`) for the class, and supplies the metaclass information for the runtime typing system. `OSDefineMetaClassAndStructors` takes as arguments the name of your driver's class and the name of its base class. It uses these to generate code that allows your driver class to be loaded and instantiated while the kernel is

developer.apple.com/documentation/DeviceDrivers/Conceptual/WritingDeviceDriver/CPluPlusRuntime/chapter_2_section_3.html#//apple_ref/doc/uid/TP30000695-BAJCCBGJ

Windows



Windows Driver Kit: Kernel-Mode Driver Architecture

ExAllocatePoolWithTag

The **ExAllocatePoolWithTag** routine allocates pool memory of the specified type and returns a pointer to the allocated block.

```
PVOID
  ExAllocatePoolWithTag(
    IN POOL_TYPE  PoolType,
    IN SIZE_T  NumberOfBytes,
    IN ULONG  Tag
    );
```

Parameters

PoolType
Specifies the type of pool memory to allocate. Each allocation code path should use a unique pool tag to help debuggers and verifiers identify the code path. You can modify the *PoolType* value by using a bitwise OR with the POOL_RAISE_IF_ALLOCATION_FAILURE flag. This flag causes an exception to be raised if the request cannot be satisfied.
Similarly, you can modify *PoolType* by using a bitwise OR with the POOL_COLD_ALLOCATION flag as a hint to the kernel to allocate the memory from pages that are likely to be paged out quickly. To reduce the amount of resident pool memory as much as possible, you should not reference these allocations frequently. The POOL_COLD_ALLOCATION flag is only advisory and is available for Microsoft Windows XP and later operating systems. For a description of the available pool memory types, see POOL_TYPE.

NumberOfBytes
Specifies the number of bytes to allocate.

Tag
Specifies the pool tag for the allocated memory. Specify the pool tag as a character literal of up to four characters delimited by single quotation marks (for example, 'Tag1'). The string is usually specified in reverse order (for example, '1gaT'). The ASCII value of each character in the tag must be between 0 and 127. Every allocation code path should use a unique pool tag to ensure that debuggers and verifiers identify a distinct allocated block.

Return Value

ExAllocatePoolWithTag returns NULL if there is insufficient memory in the free pool to satisfy the request. Use of POOL_RAISE_IF_ALLOCATION_FAILURE is not recommended because it is costly. Any successful allocation that requests *NumberOfBytes* > PAGE_SIZE wastes all unused bytes on the last-allocated page. However, on Windows Vista and later, the unused bytes are no longer wasted.

Comments

This routine is used for the general pool allocation of memory.

If *NumberOfBytes* is PAGE_SIZE or greater, a page-aligned buffer is allocated. Memory allocations of PAGE_SIZE or less do not cross page boundaries. Memory allocations of less than PAGE_SIZE are not necessarily page-aligned but are aligned on an 8-byte boundary.

msdn.microsoft.com/en-us/library/ms796989.aspx

Perl

Memory Allocation

It is suggested that you use the version of malloc that is distributed with Perl. It keeps pools of various sizes of unallocated memory in order to satisfy allocation requests more quickly. However, on some platforms, it may cause spurious malloc or free errors.

```
New(x, pointer, number, type);
Newc(x, pointer, number, type, cast);
Newz(x, pointer, number, type);
```

These three macros are used to initially allocate memory.

The first argument `x` was a "magic cookie" that was used to keep track of who called the macro, to help when debugging memory problems. However, the current code makes no use of this feature (most Perl developers now use run-time memory checkers), so this argument can be any number.

The second argument `pointer` should be the name of a variable that will point to the newly allocated memory.

The third and fourth arguments `number` and `type` specify how many of the specified type of data structure should be allocated. The argument `type` is passed to `sizeof`. The final argument to `Newc`, `cast`, should be used if the `pointer` argument is different from the `type` argument.

Unlike the `New` and `Newc` macros, the `Newz` macro calls `memzero` to zero out all the newly allocated memory.

```
Renew(pointer, number, type);
Renewc(pointer, number, type, cast);
Safefree(pointer)
```

These three macros are used to change a memory buffer size or to free a piece of memory no longer needed. The arguments to `Renew` and `Renewc` match those of `New` and `Newc` with the exception of not needing the "magic cookie" argument.

```
Move(source, dest, number, type);
Copy(source, dest, number, type);
Zero(dest, number, type);
```

These three macros are used to move, copy, or zero out previously allocated memory. The `source` and `dest` arguments point to the source and destination starting points. Perl will move, copy, or zero out `number` instances of the size of the `type` data structure (using the `sizeof` function).

PerlIO

The most recent development releases of Perl has been experimenting with removing Perl's dependency on the "normal" standard I/O suite and allowing other stdio implementations to be used. This involves creating a new abstraction layer that then calls whichever implementation of stdio Perl was compiled with. All XSUBs should now use the functions in the PerlIO abstraction layer and not make any assumptions about what kind of stdio is being used.

Linux

Chapter 5. Memory Management in Linux

Prev   Next

Chapter 5. Memory Management in Linux

Table of Contents

The Slab Cache

kcalloc — allocate memory for an array. The memory is set to zero.
kmalloc_node — allocate memory from a specific node
kzalloc — allocate memory. The memory is set to zero.
kzalloc_node — allocate zeroed memory from a particular memory node.
kmem_cache_create — Create a cache.
kmem_cache_shrink — Shrink a cache.
kmem_cache_destroy — delete a cache
kmem_cache_alloc — Allocate an object
kmem_cache_free — Deallocate an object
kfree — free previously allocated memory

Prev		Next
crc_ccitt	Home	kcalloc

www.kernel.org/doc/htmldocs/kernel-api/ch05.html

Linux

kmalloc_node

[Prev](#) The Slab Cache [Next](#)

Name

kmalloc_node — allocate memory from a specific node

Synopsis

```
void * kmalloc_node (size_t  size,
                     gfp_t   flags,
                     int     node);
```

Arguments

size

how many bytes of memory are required.

flags

the type of memory to allocate (see kcalloc).

node

node to allocate from.

Description

`kmalloc` for non-local nodes, used to allocate from a specific node if available. Equivalent to `kmalloc` in the non-NUMA single-node case.

Prev	Up	Next
kcalloc	Home	kzalloc

www.kernel.org/doc/htmldocs/kernel-api/ch05.html

Linux

kcalloc

Prev | The Slab Cache | Next

Name

kcalloc — allocate memory for an array. The memory is set to zero.

Synopsis

```
void * kcalloc (size_t n,
                size_t size,
                gfp_t  flags);
```

Arguments

n

number of elements.

size

element size.

flags

the type of memory to allocate.

Description

The `flags` argument may be one of:

`GFP_USER` - Allocate memory on behalf of user. May sleep.

`GFP_KERNEL` - Allocate normal kernel ram. May sleep.

`GFP_ATOMIC` - Allocation will not sleep. May use emergency pools. For example, use this inside interrupt handlers.

`GFP_HIGHUSER` - Allocate pages from high memory.

`GFP_NOIO` - Do not do any I/O at all while trying to get memory.

www.kernel.org/doc/htmldocs/kernel-api/ch05.html

That’s what you also get. . .