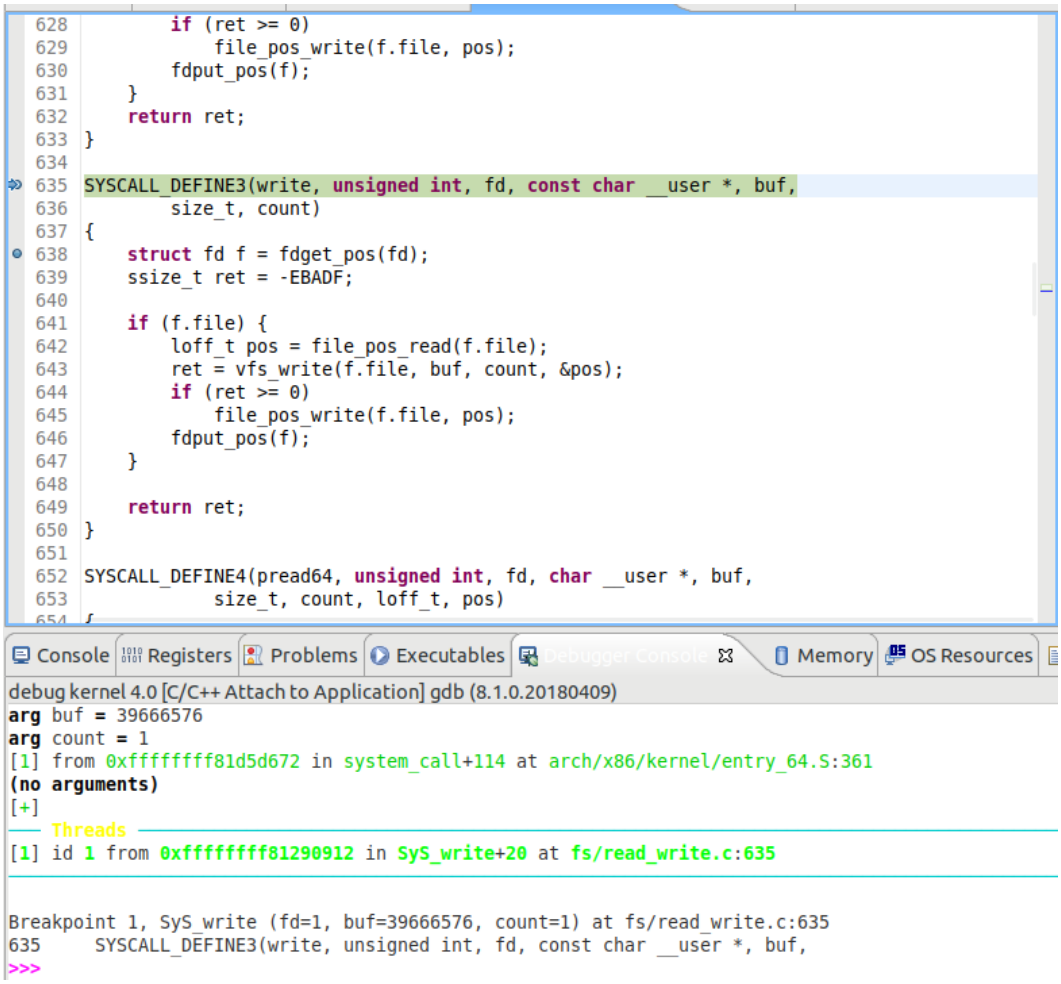
hw05. syswrite

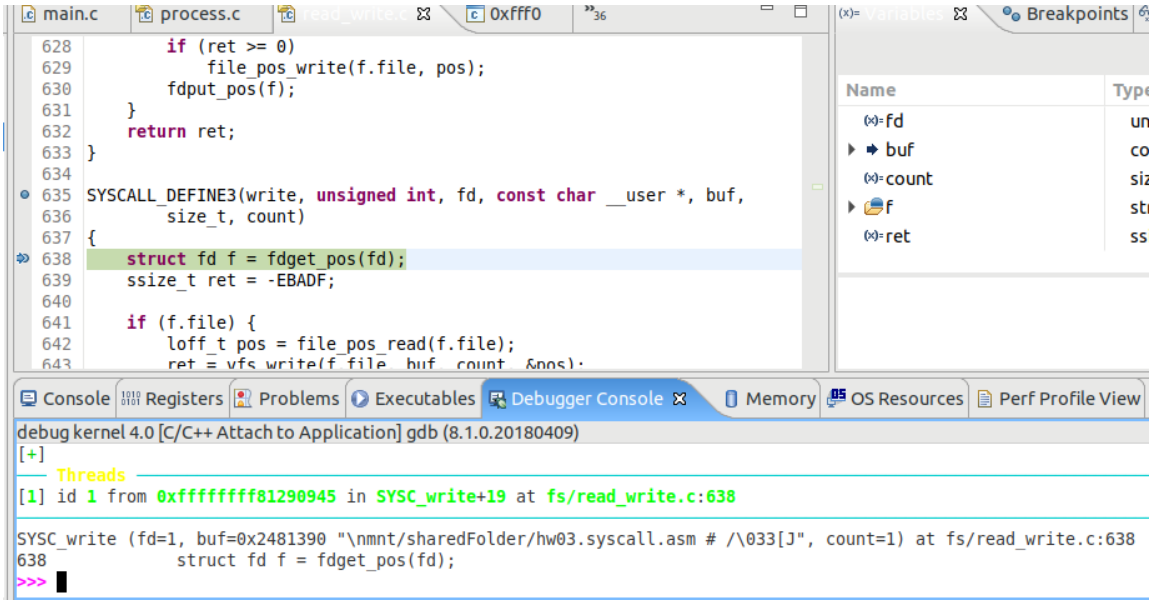
406410114 郭晏誠

1. 請將 sys_write 追蹤完畢，說明從剛開始到「於 terminal 中印出字串」為止的函數。

(1)b sys_write 後停在 635 行　在 638 行設置中斷點



(2)按 c 跳到 638 用 s 開始追蹤



(3)追蹤過程見下方表格，詳細截圖過程請見附錄

Index	File and column	Function code	Mark
1	fs/read_write.c:638	struct fd f = fdget_pos(fd);	
2	fs/read_write.c:270	return __to_fd(__fdget_pos(fd));	
3	fs/file.c:708	unsigned long v = __fdget(fd);	
4	fs/file.c:697	return __fget_light(fd, FMODE_PATH);	
5	fs/file.c:680	struct files_struct *files = current->files;	
6	./arm/x86/include/asm/current.h:14	return this_cpu_read_stable(current_task);	
7	fs/file.c:680	struct files_struct *files = current->files;	
8	fs/file.c:683	if (atomic_read(&files->count) == 1) {	
9	./arm/x86/include/asm/atomic.h:27	return ACCESS_ONCE((v)->counter);	
10	fs/file.c:684	file = __fcheck_files(files, fd);	
11	include/linux/fdtable:77	struct fdtable *fdt = rcu_dereference_raw(files->fdt);	
12	include/linux/fdtable:79	if (fd < fdt->max_fds)	
13	include/linux/fdtable:80	return rcu_dereference_raw(fdt->fd[fd]);	
14	fs/file.c:685	if (!file || unlikely(file->f_mode & mask))	
15	fs/file.c:687	return (unsigned long)file;	
16	fs/file.c:709	struct file *file = (struct file *)(v & ~3);	
17	fs/file.c:711	if (file && (file->f_mode & FMODE_ATOMIC_POS)) {	
18	fs/file.c:717	return v;	
19	include/linux/file.h:50	return (struct fd){(struct file *)(v & ~3),v & 3};	
20	fs/read_write.c:639	ssize_t ret = -EBADF;	
21	fs/read_write.c:641	if (f.file) {	
22	fs/read_write.c:642	loff_t pos = file_pos_read(f.file);	
23	fs/read_write.c:612	return file->f_pos;	
24	fs/read_write.c:643	ret = vfs_write(f.file, buf, count, &pos);	
25	fs/read_write.c:580	if (!(file->f_mode & FMODE_WRITE))	
26	fs/read_write.c:582	if (!(file->f_mode & FMODE_CAN_WRITE))	
27	fs/read_write.c:584	if (unlikely(!access_ok(VERIFY_READ, buf, count)))	
28	./arm/x86/include/asm/thread_info.h:162	ti = (void *)(this_cpu_read_stable(kernel_stack) +	
29	./arm/x86/include/asm/thread_info.h:163	KERNEL_STACK_OFFSET - THREAD_SIZE);	
30	./arm/x86/include/asm/thread_info.h:164	return ti;	
31	./arm/x86/include/asm/uaccess.h:57	addr += size;	
32	./arm/x86/include/asm/uaccess.h:58	if (addr < size)	
33	./arm/x86/include/asm/uaccess.h:60	return addr > limit;	
34	fs/read_write.c:587	ret = rw_verify_area(WRITE, file, pos, count);	
35	fs/read_write.c:391	int retval = -EINVAL;	

36	fs/read_write.c:393	inode = file_inode(file);	
37	include/linux/fs.h:1963	return f->f_inode;	
38	fs/read_write.c:394	if (unlikely((ssize_t) count < 0))	
39	fs/read_write.c:396	pos = *ppos;	
40	fs/read_write.c:397	if (unlikely(pos < 0)) {	
41	fs/read_write.c:402	} else if (unlikely((loff_t) (pos + count) < 0)) {	
42	fs/read_write.c:407	if (unlikely(inode->i_flctx && mandatory_lock(inode))) {	
43	fs/read_write.c:414	retval = security_file_permission(file,	
44	security/security.c:713	ret = security_ops->file_permission(file, mask);	
45	security/selinux/hooks.c:3215	struct inode *inode = file_inode(file);	
46	include/linux/fs.h:1963	return f->f_inode;	
47	security/selinux/hooks.c:3216	struct file_security_struct *fsec = file->f_security;	
48	security/selinux/hooks.c:3217	struct inode_security_struct *isec = inode->i_security;	
49	security/selinux/hooks.c:218	const struct task_security_struct *tsec = current_security();	
50	./arm/x86/include/asm/current.h:14	return this_cpu_read_stable(current_task);	
51	security/selinux/hooks.c:218	const struct task_security_struct *tsec = current_security();	
52	security/selinux/hooks.c:220	return tsec->sid;	
53	security/selinux/hooks.c:3220	if (!mask)	
54	security/selinux/avc.c:774	return avc_cache.latest_notif;	
55	security/selinux/hooks.c:3227	return 0;	
56	security/security.c:714	if (ret)	
57	fs/read_write.c:635	SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,	
1~57 行會重複做 2-3 次不等，但是螢幕沒有印出第一行，猜測可能是有呼叫到 sys_write 但是是把一些東西寫道 cache，並非螢幕上，例如：寫指令到 history			
58	include/linux/fsnotify.h:40	struct path *path = &file->f_path;	
59	include/linux/fs.h:1963	return f->f_inode;	
60	include/linux/fsnotify.h:42	__u32 fsnotify_mask = 0;	
61	include/linux/fsnotify.h:45	if (file->f_mode & FMODE_NONOTIFY)	
62	fs/read_write.c:416	if (retval)	
63	fs/read_write.c:588	if (ret >= 0) {	
64	fs/read_write.c:589	count = ret;	
65	fs/read_write.c:590	file_start_write(file);	
66	include/linux/fs.h:2381	if (!S_ISREG(file_inode(file)->i_mode))	
67	include/linux/fs.h:1963	return f->f_inode;	
68	include/linux/fs.h:2383	__sb_start_write(file_inode(file)->i_sb, *SB_FREEZE_WRITE*, true);	

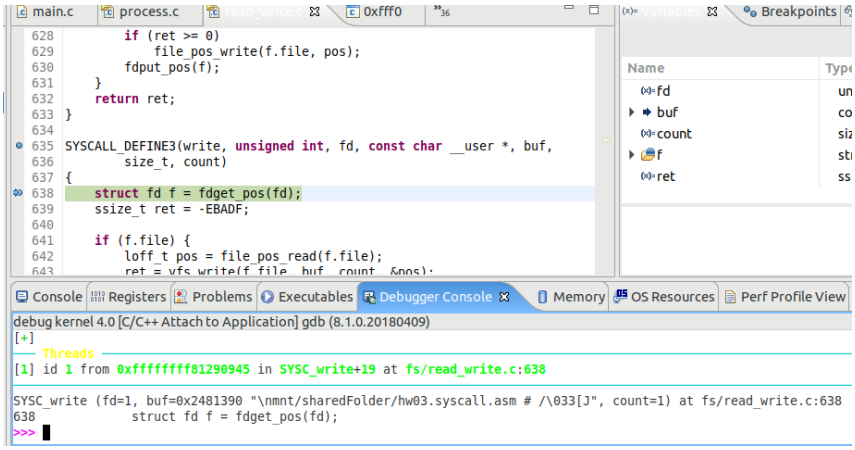
69	include/linux/fs.h:1963	return f->f_inode;	
70	include/linux/super.c:1194	if (unlikely(sb->s_writers.frozen >= level)) {	
71	include/linux/super.c:1204	percpu_counter_inc(&sb->s_writers.counter[level-1]);	
72	include/linux/percpu_counter.h:116	percpu_counter_add(fbc, 1);	
73	include/linux/percpu_counter.h:48	__percpu_counter_add(fbc, amount, percpu_counter_batch);	
74	include/linux/percpu_counter.:80	count = __this_cpu_read(*fbc->counters) + amount;	
75	include/linux/percpu-defs.:300	static inline void __this_cpu_preempt_check(const char *op) { }	
76	include/linux/percpu_counter.:81	if (count >= batch || count <= -batch) {	
77	include/linux/percpu_counter.:88	this_cpu_add(*fbc->counters, amount);	
78	include/linux/super.c:1204	smp_mb();	
79	include/linux/super.c:1210	if (unlikely(sb->s_writers.frozen >= level)) {	
80	include/linux/super.c:1214	return 1;	
81	fs/read_write.c:591	if (file->f_op->write)	
82	fs/read_write.c:592	ret = file->f_op->write(file, buf, count, pos);	
83	fs/read_write.c:526	struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len };	
84	fs/read_write.c:531	init_sync_kiocb(&kiocb, filp);	
85	./arm/x86/include/asm/aio.h:67	.ki_obj.tsk = current,	
86	./arm/x86/include/asm/current.h:14	return this_cpu_read_stable(current_task);	
87	./arm/x86/include/asm/aio.h:67	.ki_obj.tsk = current,	
88	./arm/x86/include/asm/aio.h:64	*kiocb = (struct kiocb) {	
89	fs/read_write.c:532	kiocb.ki_pos = *ppos;	
90	fs/read_write.c:533	kiocb.ki_nbytes = len;	
91	fs/read_write.c:534	iov_iter_init(&iter, WRITE, &iov, 1, len);	
92	Lib/iov_iter.c :325	if (segment_eq(get_fs(), KERNEL_DS)) {	
93	./arm/x86/include/asm/thread_info.h:162	ti = (void *)(this_cpu_read_stable(kernel_stack) +	
94	Lib/iov_iter.c :330	i->type = direction;	
95	fs/read_write.c:534	ret = filp->f_op->write_iter(&kiocb, &iter);	
96	mm/filemap.c:2656	struct file *file = iocb->ki_filp;	
97	Kernel/locking/mutex:97	might_sleep();	
98	Kernel/sched/core.c:4206	if (should_resched()) {	
99	./arm/x86/include/asm/preempt.h:95	return unlikely(!raw_cpu_read_4(__preempt_count));	
100	Kernel/sched/core.c:4206	if (should_resched()) {	
101	Kernel/sched/core.c:2845	__preempt_count_add(PREEMPT_ACTIVE);	
102	./arm/x86/include/asm/preempt.h:72	raw_cpu_add_4(__preempt_count, val);	

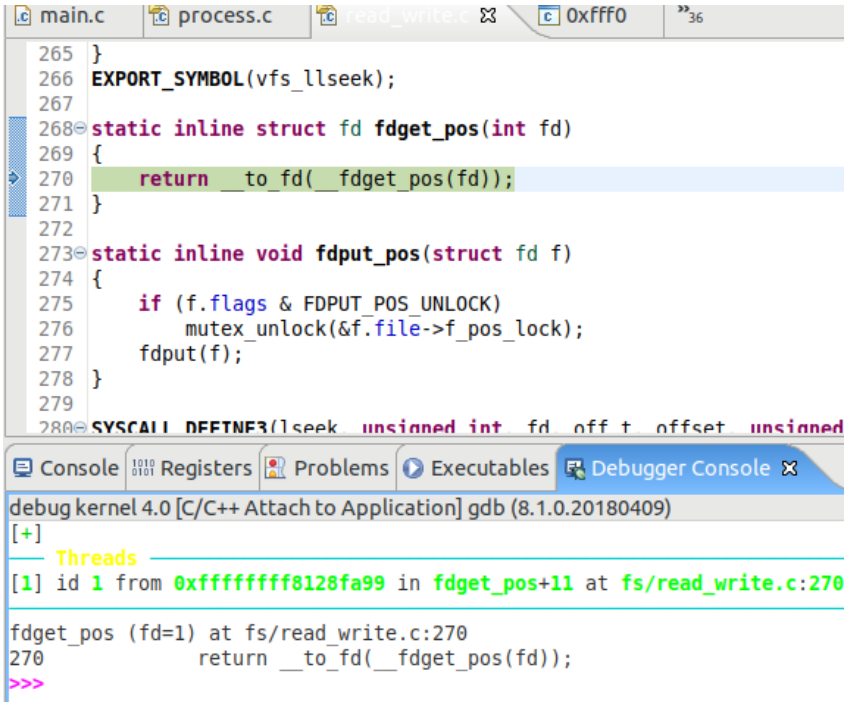
103	Kernel/sched/core.c:2845	__schedule();	
104	Kernel/sched/core.c:2721	cpu = smp_processor_id();	
105	Kernel/rcu/tree.c:285	trace_rcu_utilization(TPS("Start context switch"));	
… 中間開始進入要搶 lock 階段會經過 spinlock, ticklock, fair.c 等 function ，並且會常常進出 tree.c 做資料結構處理，這段非常的冗長，詳情請見附錄(p37-p73)			
106	fs/read_write.c:536	ret = filp->f_op->write_iter(&kiocb, &iter);	
107	fs/read_write.c:537	if (-EIOCBQUEUED == ret)	
108	fs/read_write.c:537	*ppos = kiocb.ki_pos;	
109	fs/read_write.c:540	return ret;	
110	fs/read_write.c:597	if (ret > 0) {	
111	fs/read_write.c:601	inc_syscw(current);	
112	fs/read_write.c:635	SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,	

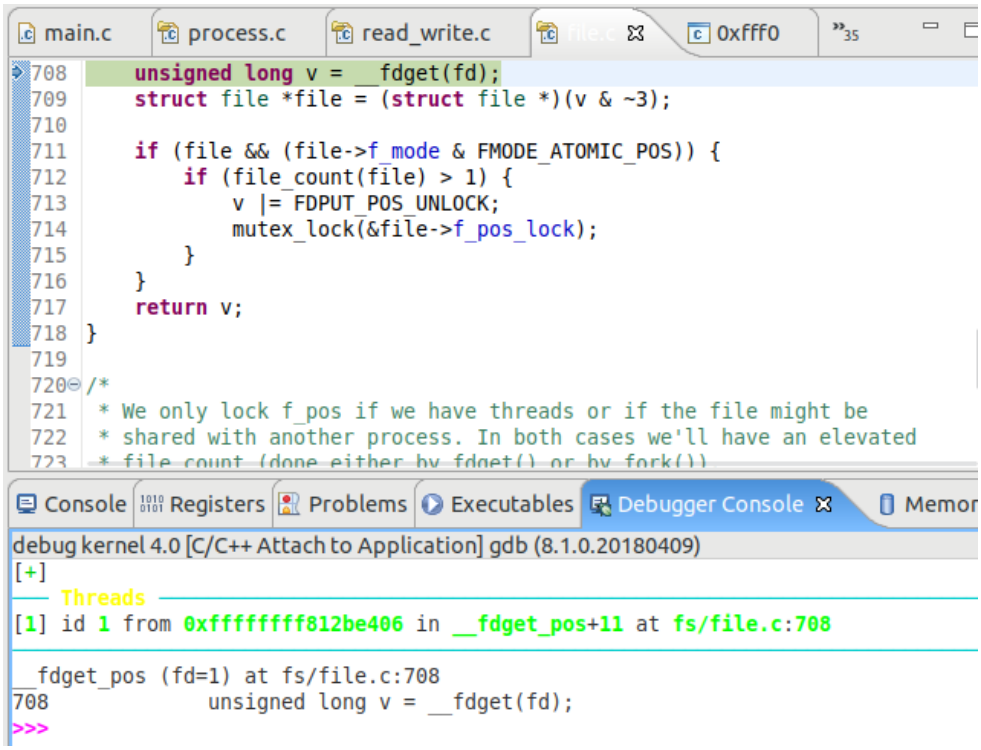
(4)印出第一行：

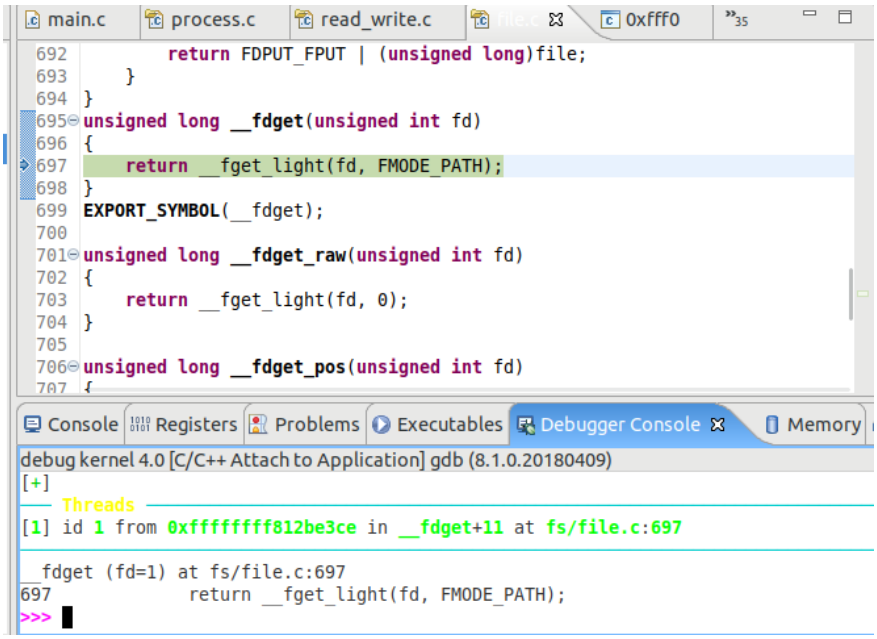
```
[   17.414524] ls (902) used greatest stack depth: 12440 bytes left
/mnt/sharedFolder # cd hw03.syscall.asm/
/mnt/sharedFolder/hw03.syscall.asm # ls
makefile    syscall     syscall.c
/mnt/sharedFolder/hw03.syscall.asm # ./syscall
使用 'int 0x80' 呼叫 system call
```

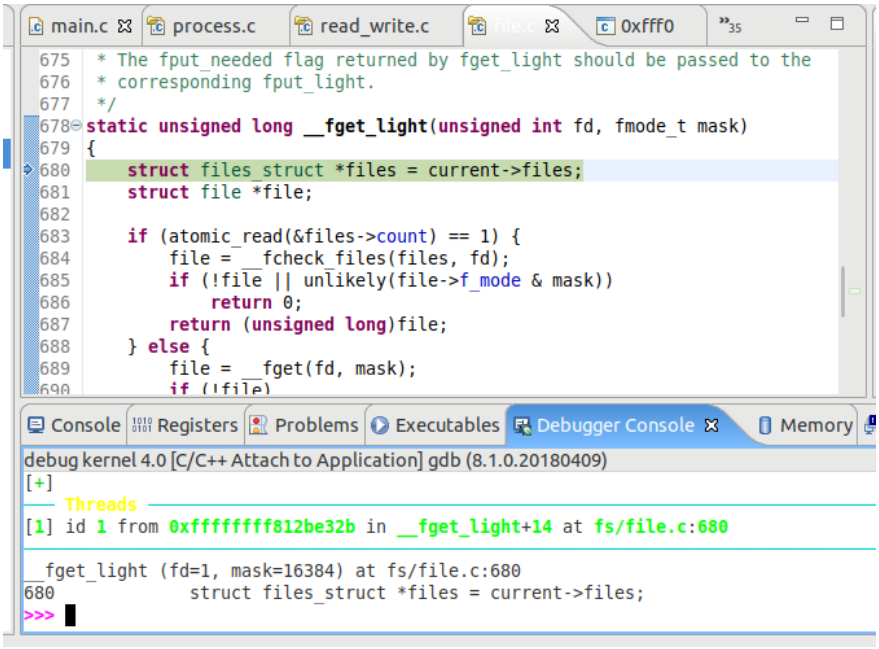
附錄：

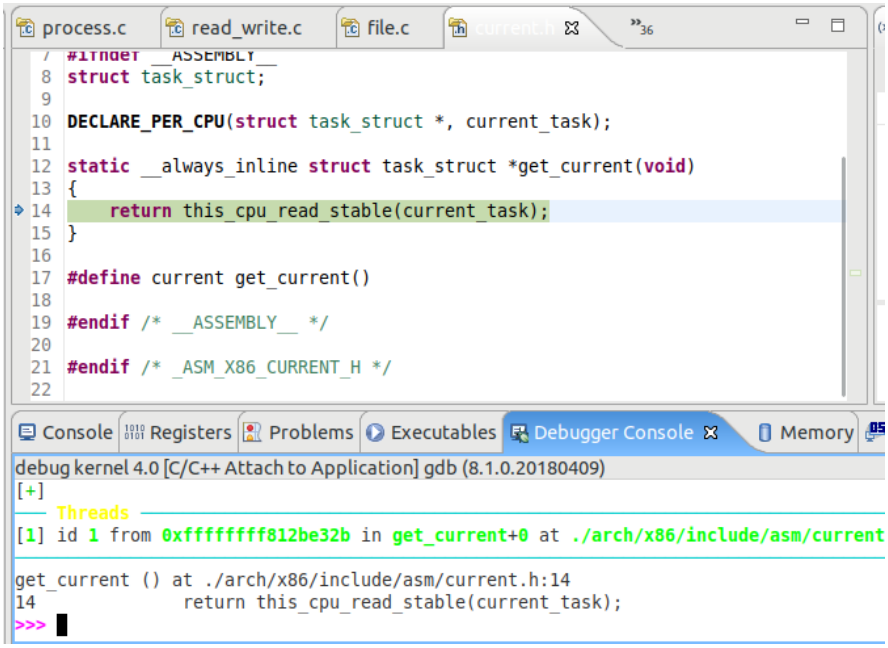

main.c
process.c
0xfff0
Breakpoints
Name
fd
buf
count
f
ret
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf Profile View
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290945 in SYSC_write+19 at fs/read_write.c:638
SYSC_write (fd=1, buf=0x2481390 "\nmnt/sharedFolder/hw03.syscall.asm # /\033[J", count=1) at fs/read_write.c:638
638 struct fd f = fdget_pos(fd);
>>>

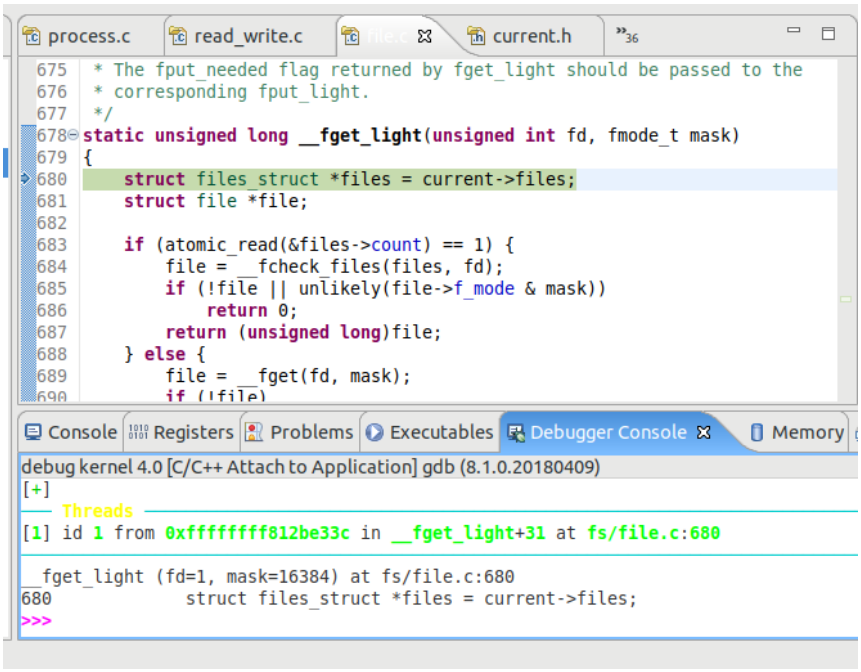

main.c
process.c
0xfff0
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128fa99 in fdget_pos+11 at fs/read_write.c:270
fdget_pos (fd=1) at fs/read_write.c:270
270 return __to_fd(__fdget_pos(fd));
>>>

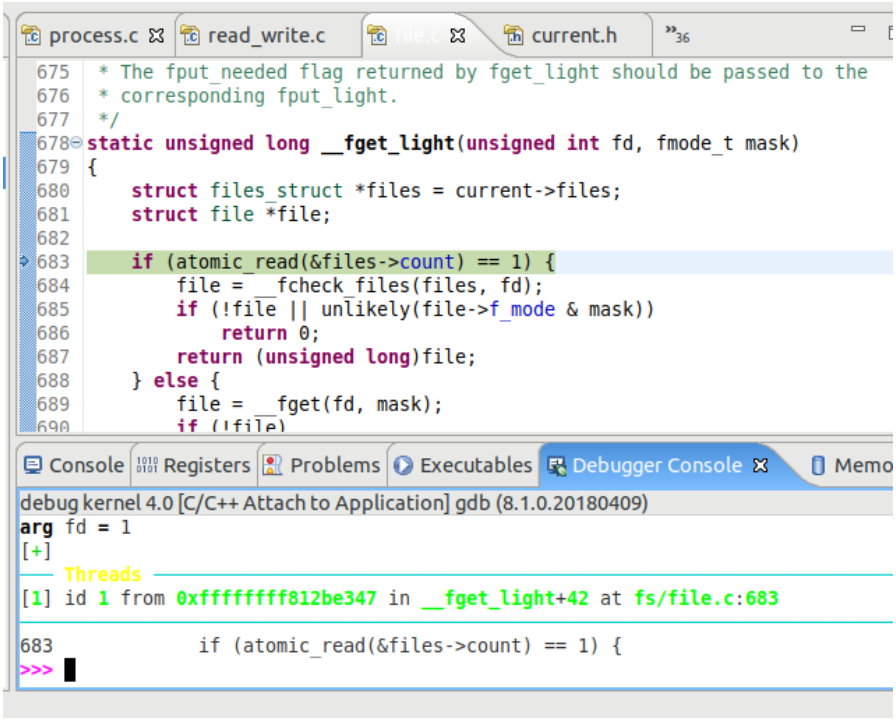

main.c
process.c
read_write.c
0xfff0
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be406 in __fdget_pos+11 at fs/file.c:708
__fdget_pos (fd=1) at fs/file.c:708
708 unsigned long v = __fdget(fd);
>>>

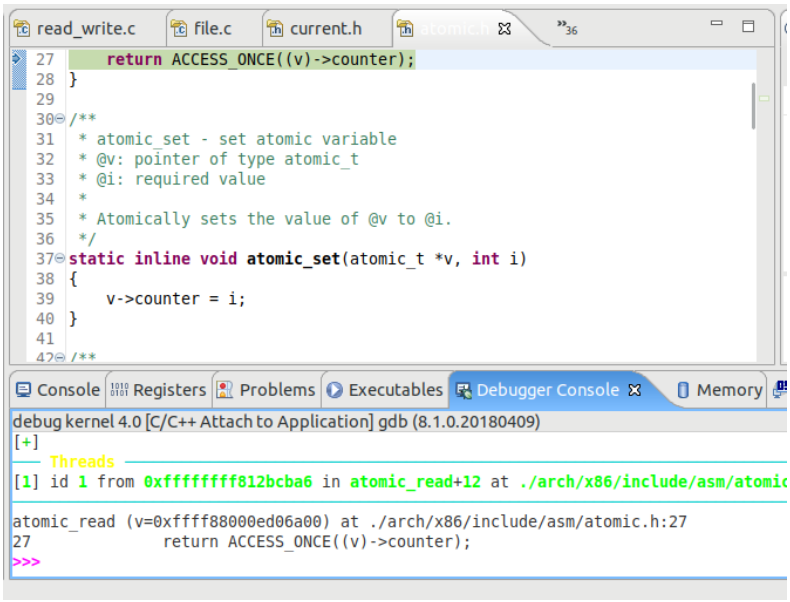

main.c
process.c
read_write.c
0xfff0
692 return FDPUT_FPUT | (unsigned long)file;
693 }
694 }
695 unsigned long __fdget(unsigned int fd)
696 {
697 return __fget_light(fd, FMODE_PATH);
698 }
699 EXPORT_SYMBOL(__fdget);
700
701 unsigned long __fdget_raw(unsigned int fd)
702 {
703 return __fget_light(fd, 0);
704 }
705
706 unsigned long __fdget_pos(unsigned int fd)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be3ce in __fdget+11 at fs/file.c:697
__fdget (fd=1) at fs/file.c:697
697 return __fget_light(fd, FMODE_PATH);
>>>

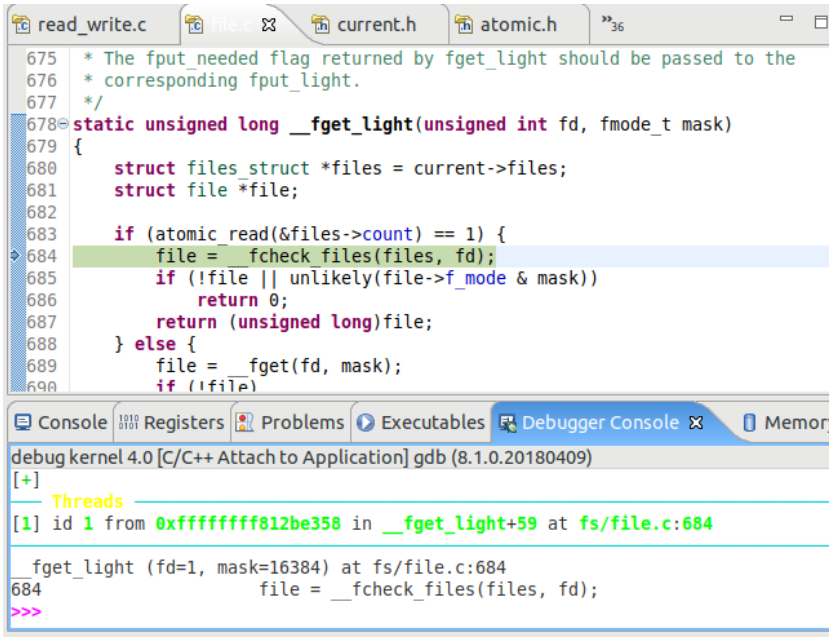

main.c
process.c
read_write.c
0xfff0
675 * The fput_needed flag returned by fget_light should be passed to the
676 * corresponding fput_light.
677 */
678 static unsigned long __fget_light(unsigned int fd, fmode_t mask)
679 {
680 struct files_struct *files = current->files;
681 struct file *file;
682
683 if (atomic_read(&files->count) == 1) {
684 file = __fcheck_files(files, fd);
685 if (!file || unlikely(file->f_mode & mask))
686 return 0;
687 return (unsigned long)file;
688 } else {
689 file = __fget(fd, mask);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be32b in __fget_light+14 at fs/file.c:680
__fget_light (fd=1, mask=16384) at fs/file.c:680
680 struct files_struct *files = current->files;
>>>

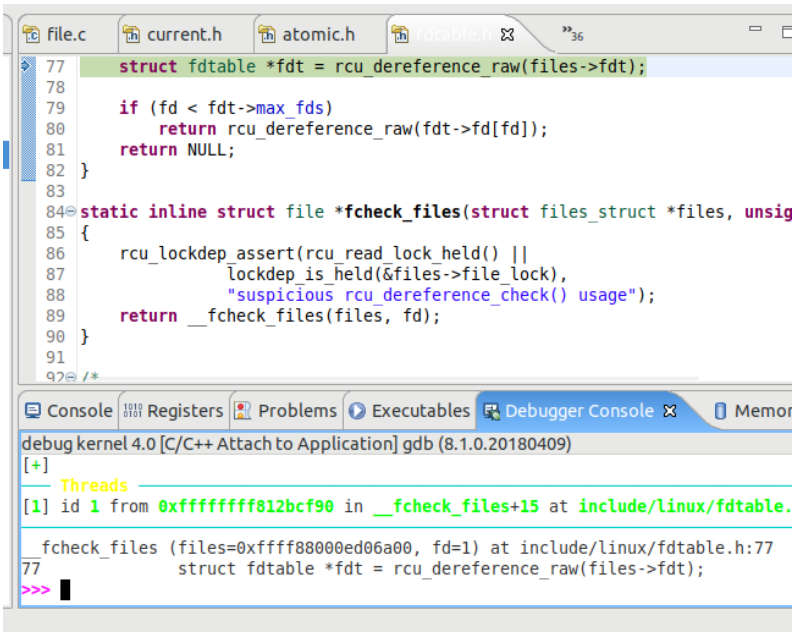

process.c
read_write.c
file.c
current.h
8 struct task_struct;
9
10 DECLARE_PER_CPU(struct task_struct *, current_task);
11
12 static __always_inline struct task_struct *get_current(void)
13 {
14 return this_cpu_read_stable(current_task);
15 }
16
17 #define current get_current()
18
19 #endif /* __ASSEMBLY__ */
20
21 #endif /* _ASM_X86_CURRENT_H */
22
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be32b in get_current+0 at ./arch/x86/include/asm/current
get_current () at ./arch/x86/include/asm/current.h:14
14 return this_cpu_read_stable(current_task);
>>>

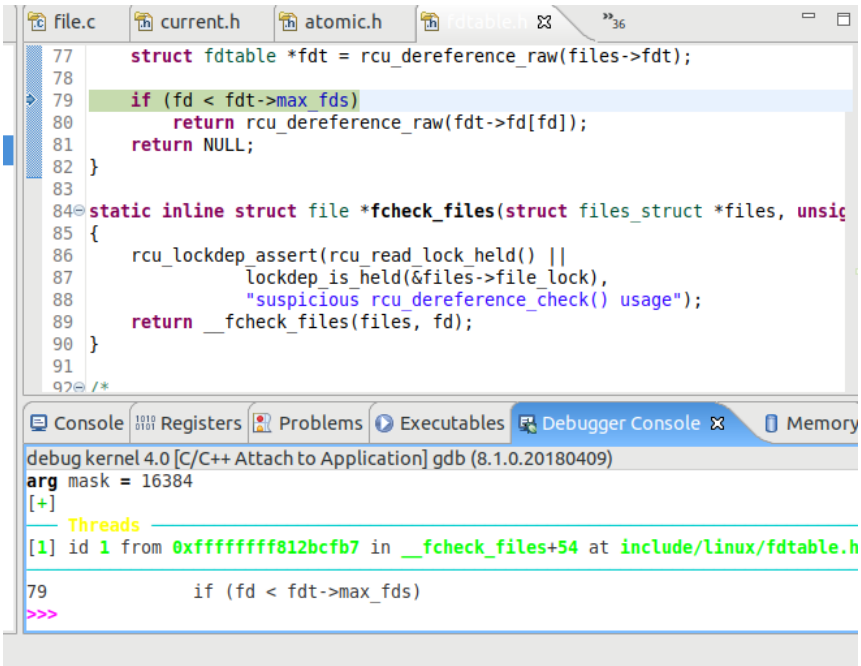

process.c
read_write.c
current.h
675 * The fput_needed flag returned by fget_light should be passed to the
676 * corresponding fput_light.
677 */
678 static unsigned long __fget_light(unsigned int fd, fmode_t mask)
679 {
680 struct files_struct *files = current->files;
681 struct file *file;
682
683 if (atomic_read(&files->count) == 1) {
684 file = __fcheck_files(files, fd);
685 if (!file || unlikely(file->f_mode & mask))
686 return 0;
687 return (unsigned long)file;
688 } else {
689 file = __fget(fd, mask);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be33c in __fget_light+31 at fs/file.c:680
__fget_light (fd=1, mask=16384) at fs/file.c:680
680 struct files_struct *files = current->files;
>>>

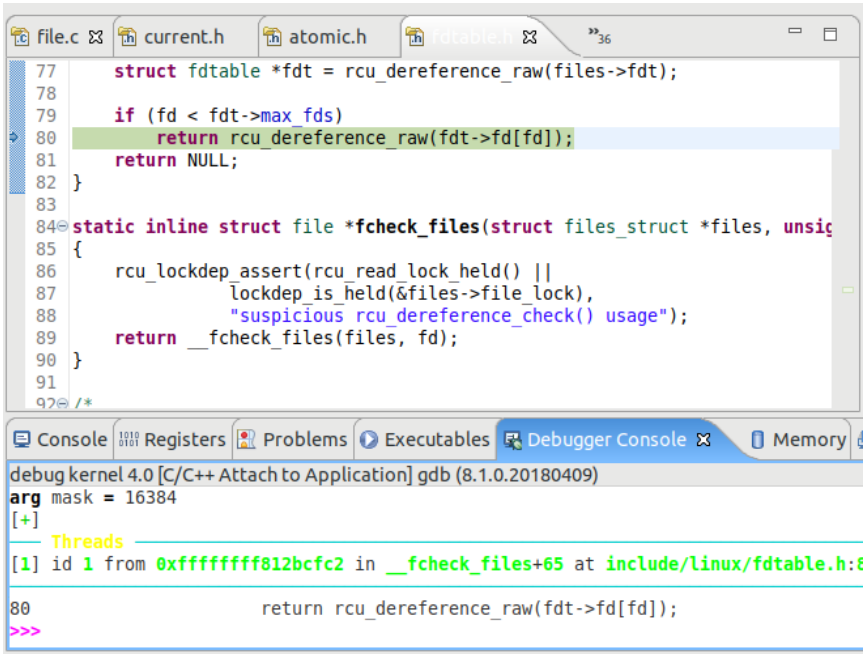

process.c
read_write.c
current.h
675 * The fput_needed flag returned by fget_light should be passed to the
676 * corresponding fput_light.
677 */
678 static unsigned long __fget_light(unsigned int fd, fmode_t mask)
679 {
680 struct files_struct *files = current->files;
681 struct file *file;
682
683 if (atomic_read(&files->count) == 1) {
684 file = __fcheck_files(files, fd);
685 if (!file || unlikely(file->f_mode & mask))
686 return 0;
687 return (unsigned long)file;
688 } else {
689 file = __fget(fd, mask);
Console
Registers
Problems
Executables
Debugger Console
Memo
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg fd = 1
[+]
Threads
[1] id 1 from 0xffffffff812be347 in __fget_light+42 at fs/file.c:683
683 if (atomic_read(&files->count) == 1) {
>>>

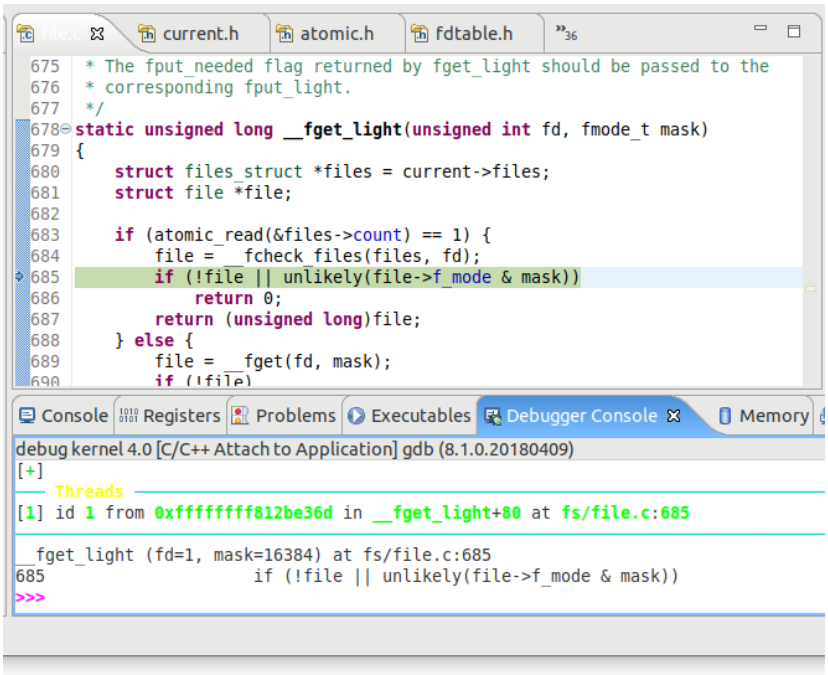

read_write.c
file.c
current.h
27 return ACCESS_ONCE((v)->counter);
28 }
29
30 /**
31 * atomic_set - set atomic variable
32 * @v: pointer of type atomic_t
33 * @i: required value
34 *
35 * Atomically sets the value of @v to @i.
36 */
37 static inline void atomic_set(atomic_t *v, int i)
38 {
39 v->counter = i;
40 }
41
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812bcba6 in atomic_read+12 at ./arch/x86/include/asm/atomic
atomic_read (v=0xffff88000ed06a00) at ./arch/x86/include/asm/atomic.h:27
27 return ACCESS_ONCE((v)->counter);
>>>

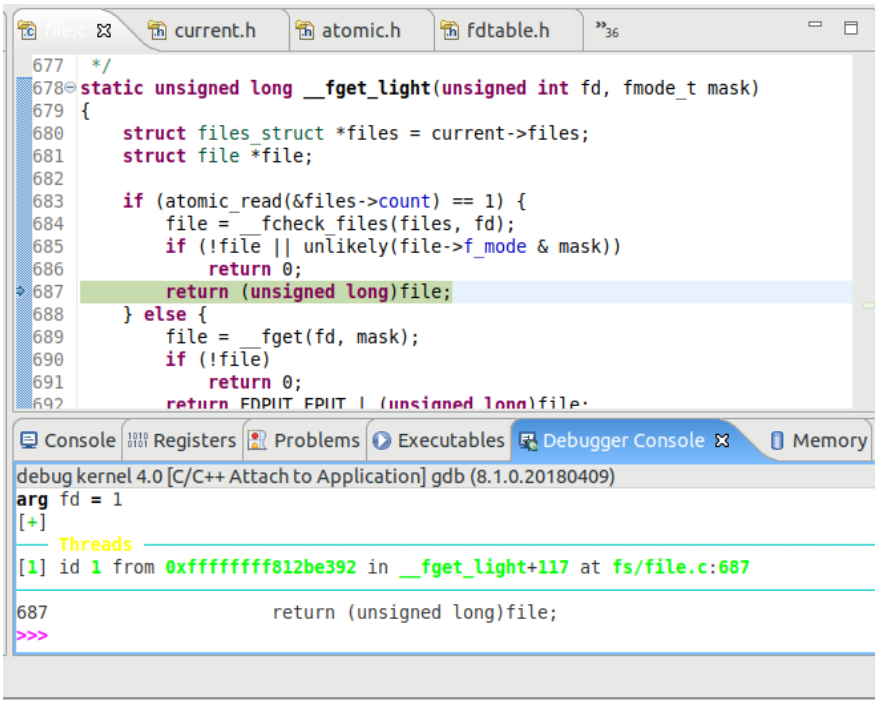

read_write.c
current.h
atomic.h
675 * The fput_needed flag returned by fget_light should be passed to the
676 * corresponding fput_light.
677 */
678 static unsigned long __fget_light(unsigned int fd, fmode_t mask)
679 {
680 struct files_struct *files = current->files;
681 struct file *file;
682
683 if (atomic_read(&files->count) == 1) {
684 file = __fcheck_files(files, fd);
685 if (!file || unlikely(file->f_mode & mask))
686 return 0;
687 return (unsigned long)file;
688 } else {
689 file = __fget(fd, mask);
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be358 in __fget_light+59 at fs/file.c:684
__fget_light (fd=1, mask=16384) at fs/file.c:684
684 file = __fcheck_files(files, fd);
>>>

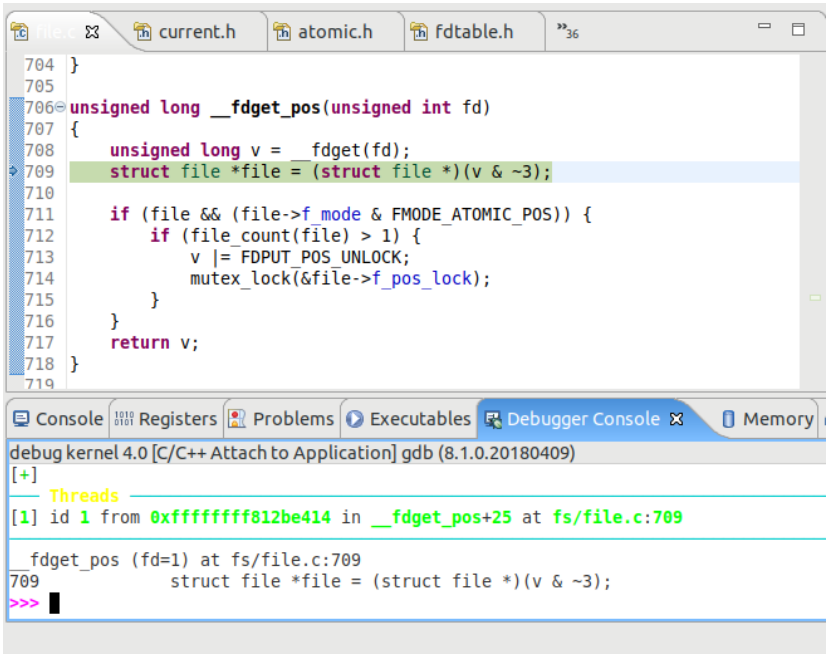

file.c
current.h
atomic.h
77 struct fdtable *fdt = rcu_dereference_raw(files->fdt);
78
79 if (fd < fdt->max_fds)
80 return rcu_dereference_raw(fdt->fd[fd]);
81 return NULL;
82 }
83
84 static inline struct file *fcheck_files(struct files_struct *files, unsig
85 {
86 rcu_lockdep_assert(rcu_read_lock_held() ||
87 lockdep_is_held(&files->file_lock),
88 "suspicious rcu_dereference_check() usage");
89 return __fcheck_files(files, fd);
90 }
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812bcf90 in __fcheck_files+15 at include/linux/fdtable.
__fcheck_files (files=0xffff88000ed06a00, fd=1) at include/linux/fdtable.h:77
77 struct fdtable *fdt = rcu_dereference_raw(files->fdt);
>>>

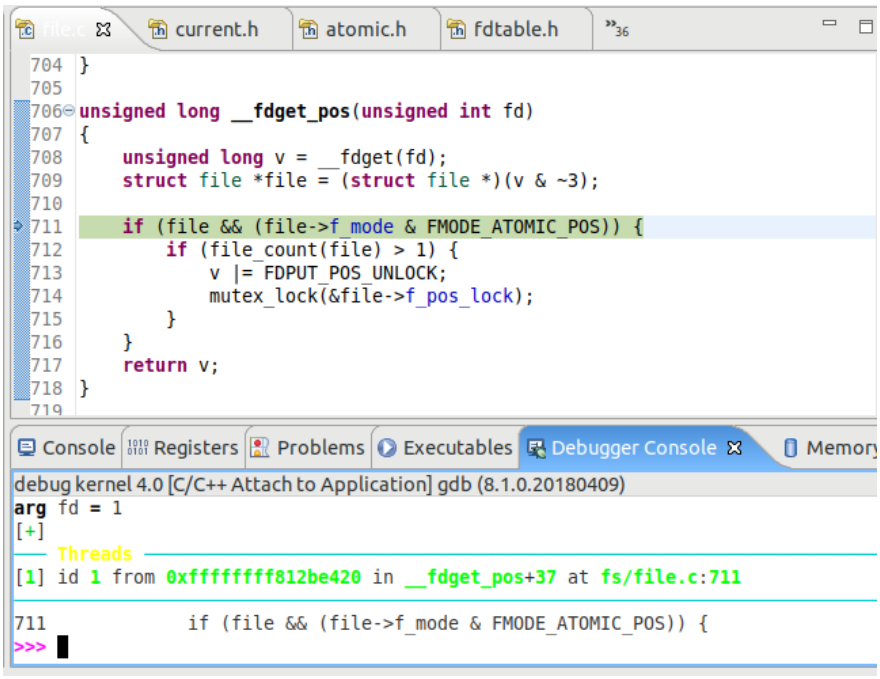

file.c
current.h
atomic.h
77 struct fdtable *fdt = rcu_dereference_raw(files->fdt);
78
79 if (fd < fdt->max_fds)
80 return rcu_dereference_raw(fdt->fd[fd]);
81 return NULL;
82 }
83
84 static inline struct file *fcheck_files(struct files_struct *files, unsig
85 {
86 rcu_lockdep_assert(rcu_read_lock_held() ||
87 lockdep_is_held(&files->file_lock),
88 "suspicious rcu_dereference_check() usage");
89 return __fcheck_files(files, fd);
90 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg mask = 16384
[+]
Threads
[1] id 1 from 0xffffffff812bcfb7 in __fcheck_files+54 at include/linux/fdtable.h
79 if (fd < fdt->max_fds)
>>>

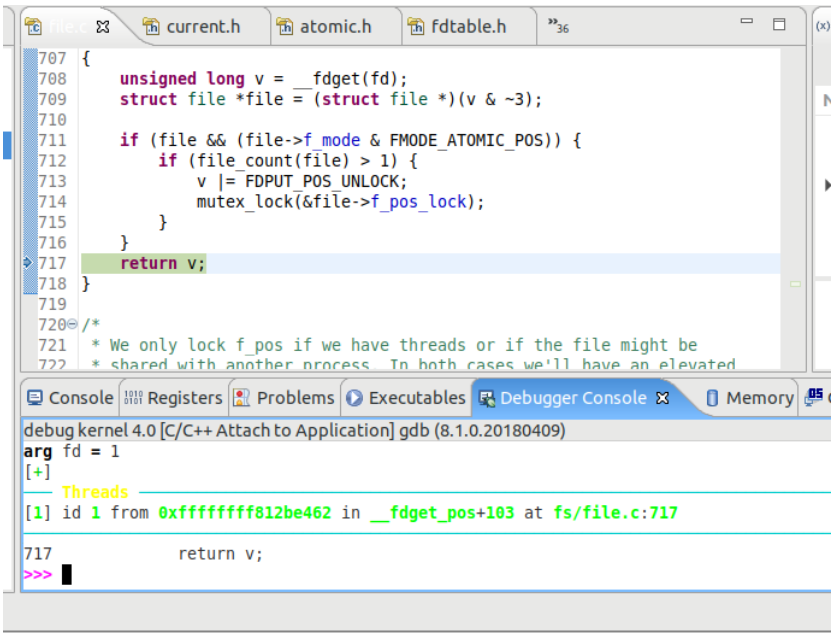

file.c
current.h
atomic.h
fdtable.h
77 struct fdtable *fdt = rcu_dereference_raw(files->fdt);
78
79 if (fd < fdt->max_fds)
80 return rcu_dereference_raw(fdt->fd[fd]);
81 return NULL;
82 }
83
84 static inline struct file *fcheck_files(struct files_struct *files, unsig
85 {
86 rcu_lockdep_assert(rcu_read_lock_held() ||
87 lockdep_is_held(&files->file_lock),
88 "suspicious rcu_dereference_check() usage");
89 return __fcheck_files(files, fd);
90 }
91
92 /*
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg mask = 16384
[+]
Threads
[1] id 1 from 0xffffffff812bcfc2 in __fcheck_files+65 at include/linux/fdtable.h:8
80 return rcu_dereference_raw(fdt->fd[fd]);
>>>

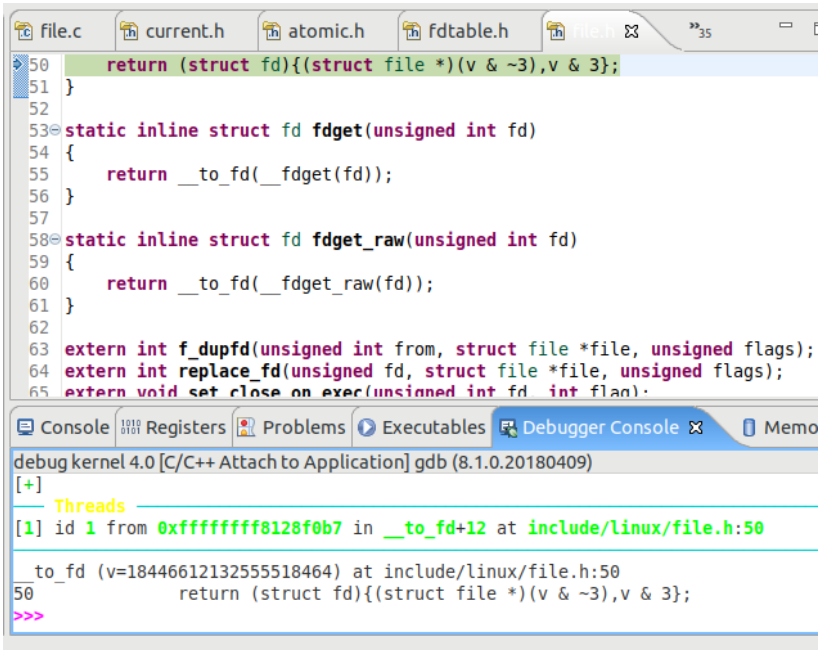

current.h
atomic.h
fdtable.h
675 * The fput_needed flag returned by fget_light should be passed to the
676 * corresponding fput_light.
677 */
678 static unsigned long __fget_light(unsigned int fd, fmode_t mask)
679 {
680 struct files_struct *files = current->files;
681 struct file *file;
682
683 if (atomic_read(&files->count) == 1) {
684 file = __fcheck_files(files, fd);
685 if (!file || unlikely(file->f_mode & mask))
686 return 0;
687 return (unsigned long)file;
688 } else {
689 file = __fget(fd, mask);
690 if (!file)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be36d in __fget_light+80 at fs/file.c:685
__fget_light (fd=1, mask=16384) at fs/file.c:685
685 if (!file || unlikely(file->f_mode & mask))
>>>

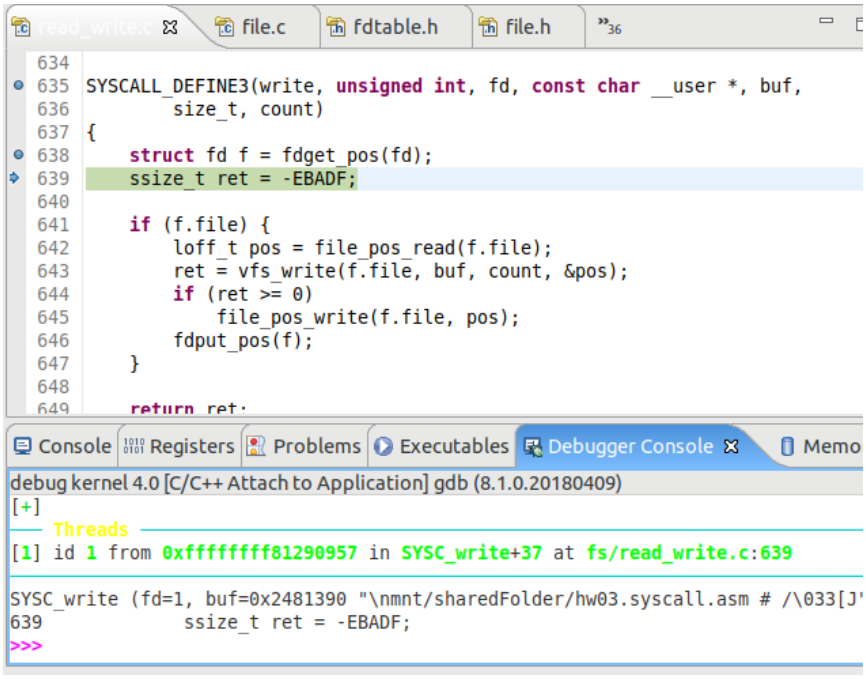

current.h
atomic.h
fdtable.h
677 */
678 static unsigned long __fget_light(unsigned int fd, fmode_t mask)
679 {
680 struct files_struct *files = current->files;
681 struct file *file;
682
683 if (atomic_read(&files->count) == 1) {
684 file = __fcheck_files(files, fd);
685 if (!file || unlikely(file->f_mode & mask))
686 return 0;
687 return (unsigned long)file;
688 } else {
689 file = __fget(fd, mask);
690 if (!file)
691 return 0;
692 return FDPUT_FPUT | (unsigned long)file;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg fd = 1
[+]
Threads
[1] id 1 from 0xffffffff812be392 in __fget_light+117 at fs/file.c:687
687 return (unsigned long)file;
>>>

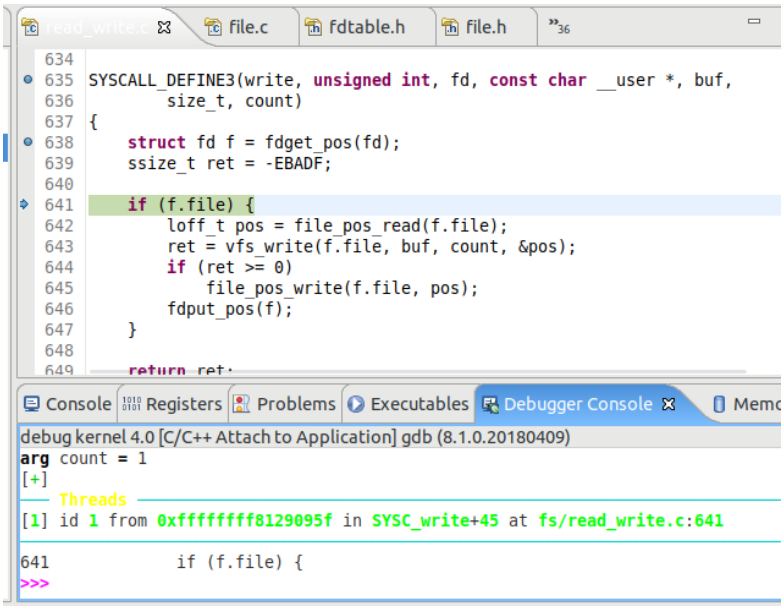

current.h
atomic.h
fdtable.h
704 }
705
706 unsigned long __fdget_pos(unsigned int fd)
707 {
708 unsigned long v = __fdget(fd);
709 struct file *file = (struct file *)(v & ~3);
710
711 if (file && (file->f_mode & FMODE_ATOMIC_POS)) {
712 if (file_count(file) > 1) {
713 v |= FDPUT_POS_UNLOCK;
714 mutex_lock(&file->f_pos_lock);
715 }
716 }
717 return v;
718 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812be414 in __fdget_pos+25 at fs/file.c:709
__fdget_pos (fd=1) at fs/file.c:709
709 struct file *file = (struct file *)(v & ~3);
>>>

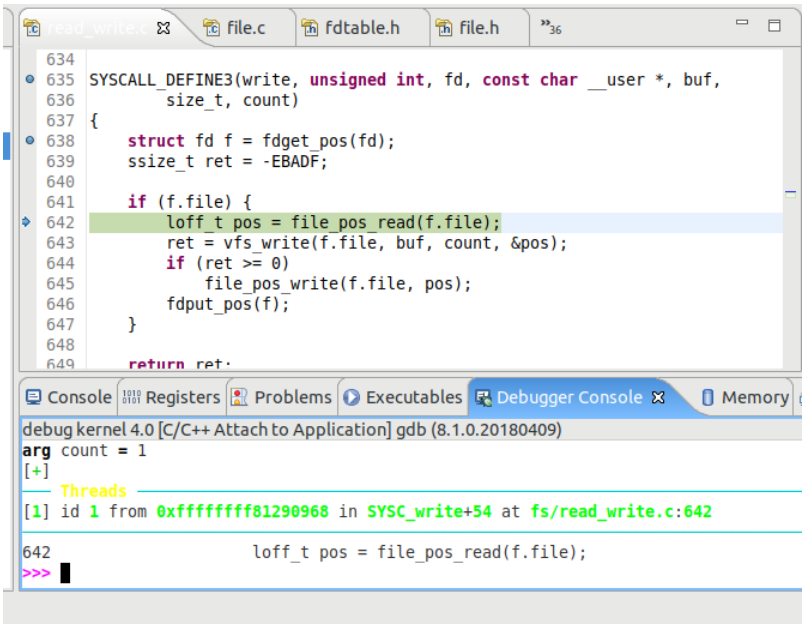

current.h
atomic.h
fdtable.h
704 }
705
706 unsigned long __fdget_pos(unsigned int fd)
707 {
708 unsigned long v = __fdget(fd);
709 struct file *file = (struct file *)(v & ~3);
710
711 if (file && (file->f_mode & FMODE_ATOMIC_POS)) {
712 if (file_count(file) > 1) {
713 v |= FDPUT_POS_UNLOCK;
714 mutex_lock(&file->f_pos_lock);
715 }
716 }
717 return v;
718 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg fd = 1
[+]
Threads
[1] id 1 from 0xffffffff812be420 in __fdget_pos+37 at fs/file.c:711
711 if (file && (file->f_mode & FMODE_ATOMIC_POS)) {
>>>

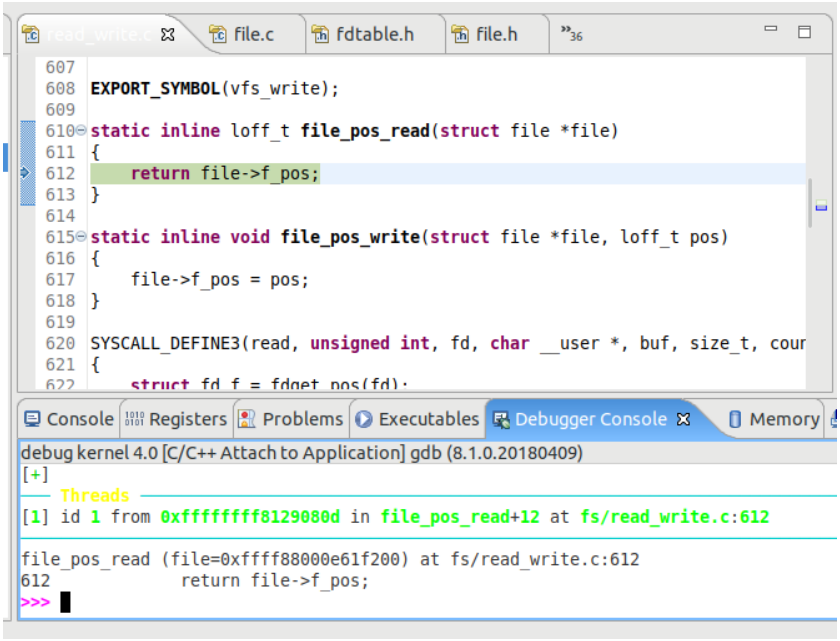

current.h
atomic.h
fdtable.h
707 {
708 unsigned long v = __fdget(fd);
709 struct file *file = (struct file *)(v & ~3);
710
711 if (file && (file->f_mode & FMODE_ATOMIC_POS)) {
712 if (file_count(file) > 1) {
713 v |= FDPUT_POS_UNLOCK;
714 mutex_lock(&file->f_pos_lock);
715 }
716 }
717 return v;
718 }
719
720 /*
721 * We only lock f_pos if we have threads or if the file might be
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg fd = 1
[+]
Threads
[1] id 1 from 0xffffffff812be462 in __fdget_pos+103 at fs/file.c:717
717 return v;
>>>

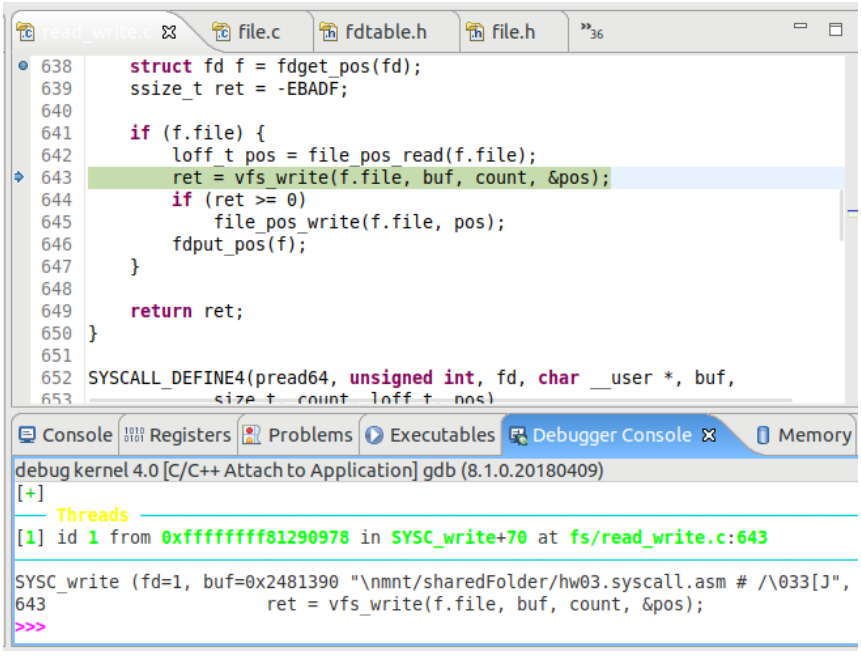

file.c
current.h
atomic.h
fdtable.h
50 return (struct fd){(struct file *)(v & ~3),v & 3};
51 }
53 static inline struct fd fdget(unsigned int fd)
54 {
55 return __to_fd(__fdget(fd));
56 }
58 static inline struct fd fdget_raw(unsigned int fd)
59 {
60 return __to_fd(__fdget_raw(fd));
61 }
63 extern int f_dupfd(unsigned int from, struct file *file, unsigned flags);
64 extern int replace_fd(unsigned fd, struct file *file, unsigned flags);
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f0b7 in __to_fd+12 at include/linux/file.h:50
__to_fd (v=18446612132555518464) at include/linux/file.h:50
50 return (struct fd){(struct file *)(v & ~3),v & 3};
>>>

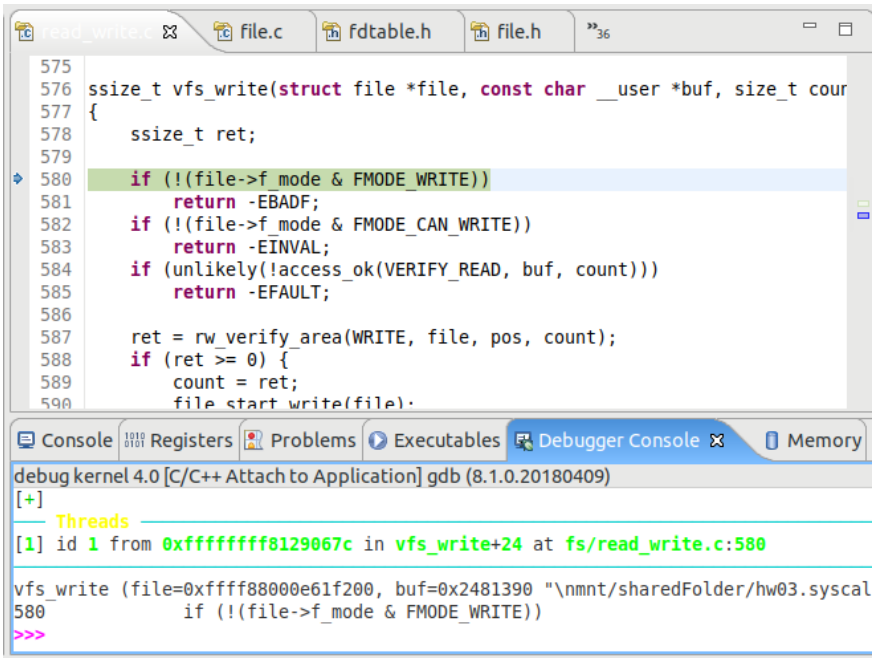

file.c
fdtable.h
file.h
634
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
636 size_t, count)
637 {
638 struct fd f = fdget_pos(fd);
639 ssize_t ret = -EBADF;
640
641 if (f.file) {
642 loff_t pos = file_pos_read(f.file);
643 ret = vfs_write(f.file, buf, count, &pos);
644 if (ret >= 0)
645 file_pos_write(f.file, pos);
646 fdput_pos(f);
647 }
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290957 in SYSC_write+37 at fs/read_write.c:639
SYSC_write (fd=1, buf=0x2481390 "\nmnt/sharedFolder/hw03.syscall.asm # /\033[J"
639 ssize_t ret = -EBADF;
>>>

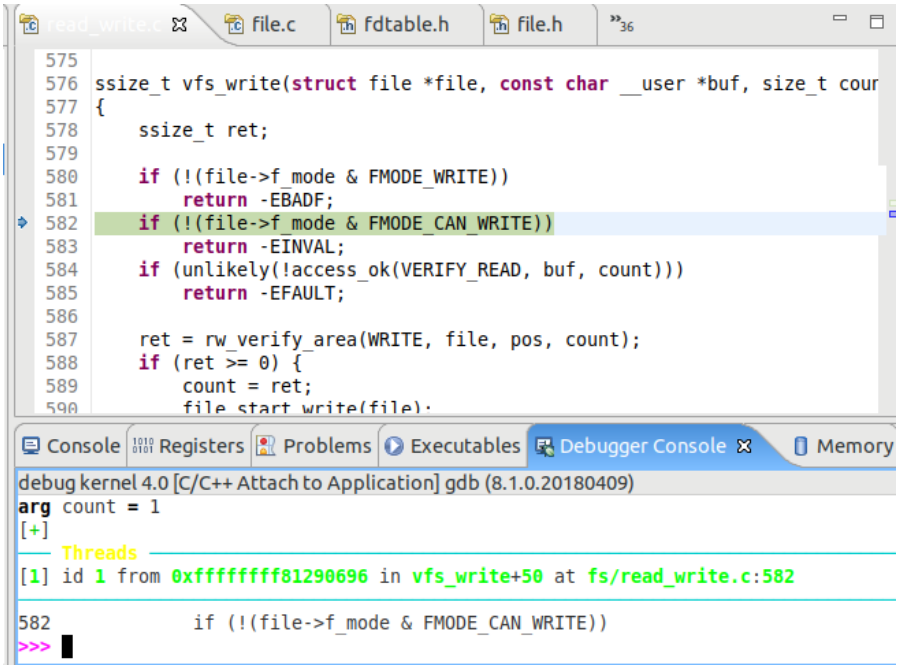

file.c
fdtable.h
file.h
634
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
636 size_t, count)
637 {
638 struct fd f = fdget_pos(fd);
639 ssize_t ret = -EBADF;
640
641 if (f.file) {
642 loff_t pos = file_pos_read(f.file);
643 ret = vfs_write(f.file, buf, count, &pos);
644 if (ret >= 0)
645 file_pos_write(f.file, pos);
646 fdput_pos(f);
647 }
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 1
[+]
Threads
[1] id 1 from 0xffffffff8129095f in SYSC_write+45 at fs/read_write.c:641
641 if (f.file) {
>>>

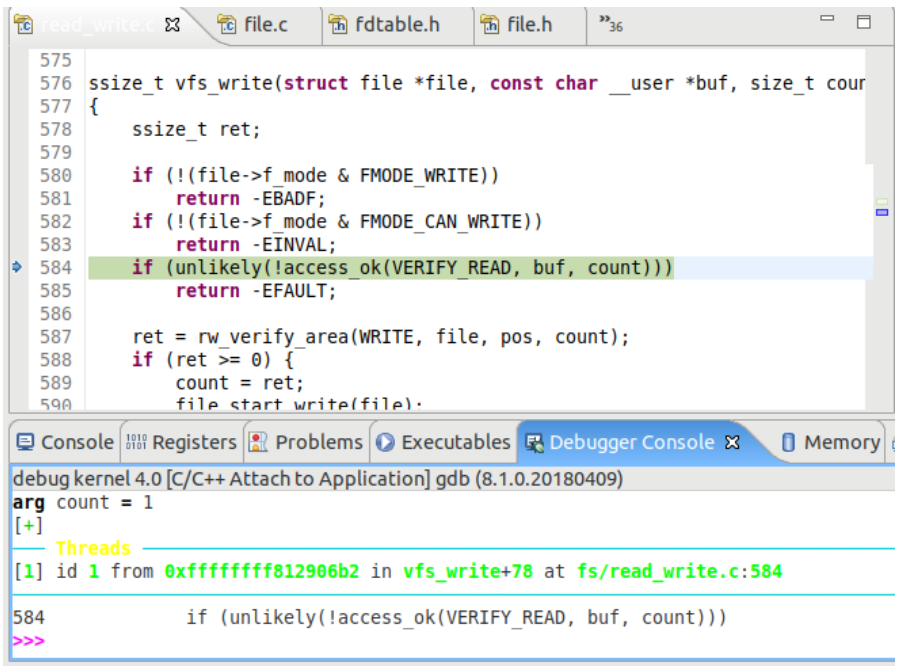

file.c
fdtable.h
file.h
SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
size_t, count)
{
struct fd f = fdget_pos(fd);
ssize_t ret = -EBADF;
if (f.file) {
loff_t pos = file_pos_read(f.file);
ret = vfs_write(f.file, buf, count, &pos);
if (ret >= 0)
file_pos_write(f.file, pos);
fdput_pos(f);
}
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 1
[+]
Threads
[1] id 1 from 0xffffffff81290968 in SYSC_write+54 at fs/read_write.c:642
642 loff_t pos = file_pos_read(f.file);
>>>

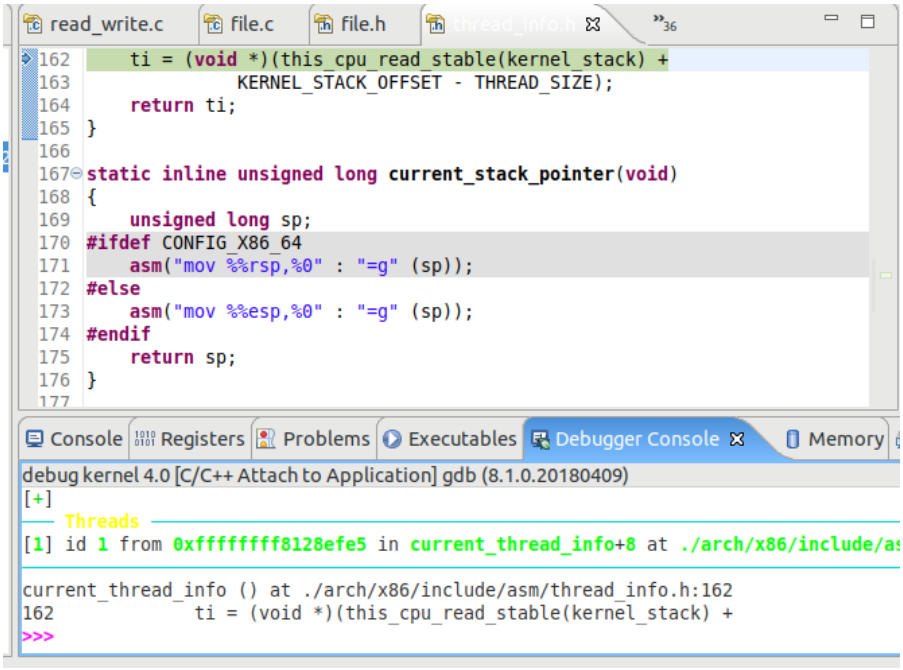

file.c
fdtable.h
file.h
EXPORT_SYMBOL(vfs_write);
static inline loff_t file_pos_read(struct file *file)
{
return file->f_pos;
}
static inline void file_pos_write(struct file *file, loff_t pos)
{
file->f_pos = pos;
}
SYSCALL_DEFINE3(read, unsigned int, fd, char __user *, buf, size_t, cour
{
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8129080d in file_pos_read+12 at fs/read_write.c:612
file_pos_read (file=0xffff88000e61f200) at fs/read_write.c:612
612 return file->f_pos;
>>>

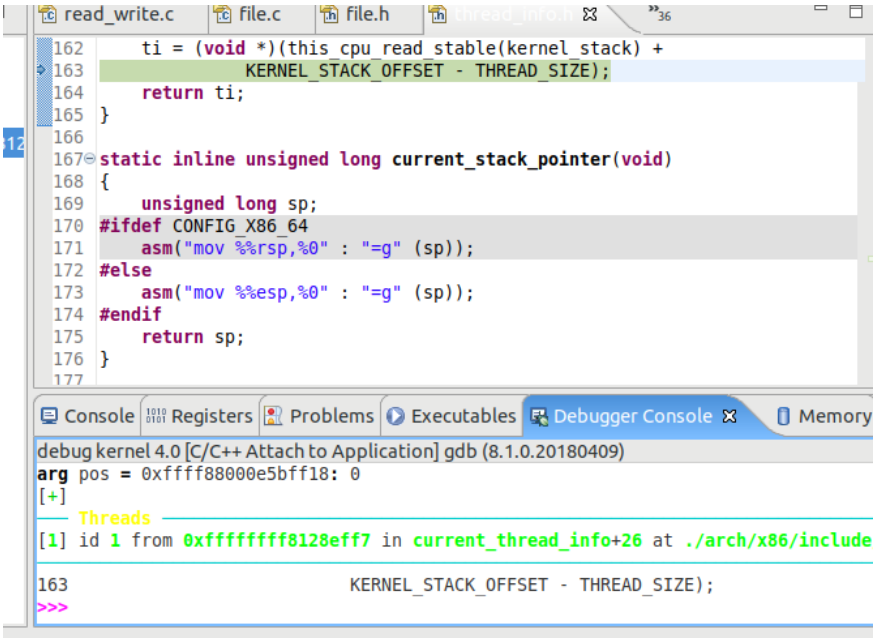

file.c
fdtable.h
file.h
struct fd f = fdget_pos(fd);
ssize_t ret = -EBADF;
if (f.file) {
loff_t pos = file_pos_read(f.file);
ret = vfs_write(f.file, buf, count, &pos);
if (ret >= 0)
file_pos_write(f.file, pos);
fdput_pos(f);
}
return ret;
}
SYSCALL_DEFINE4(pread64, unsigned int, fd, char __user *, buf,
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290978 in SYSC_write+70 at fs/read_write.c:643
SYSC_write (fd=1, buf=0x2481390 "\nmnt/sharedFolder/hw03.syscall.asm # /\033[J",
643 ret = vfs_write(f.file, buf, count, &pos);
>>>

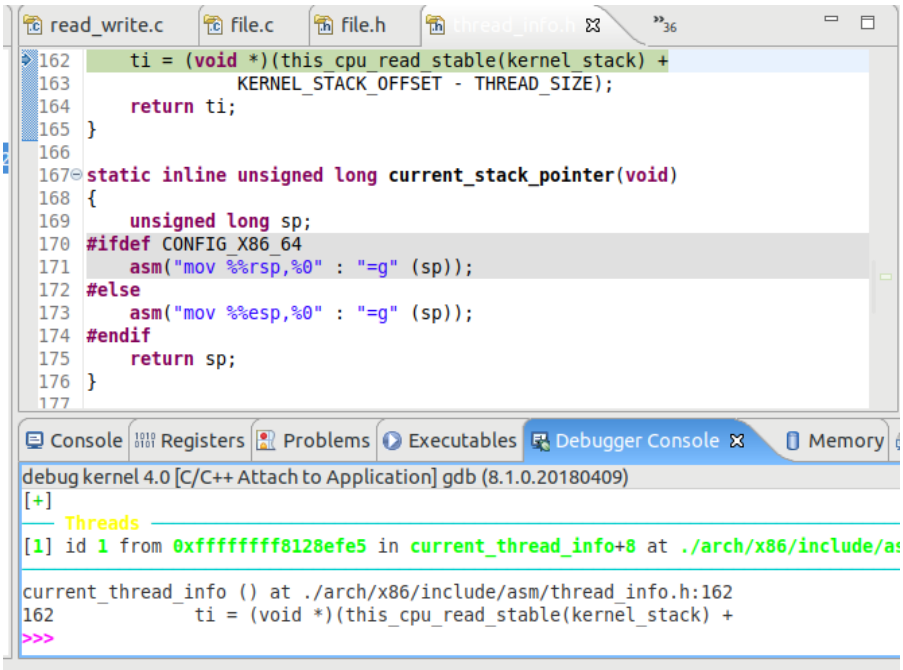

file.c
fdtable.h
file.h
575
576 ssize_t vfs_write(struct file *file, const char __user *buf, size_t cour
577 {
578 ssize_t ret;
579
580 if (!(file->f_mode & FMODE_WRITE))
581 return -EBADF;
582 if (!(file->f_mode & FMODE_CAN_WRITE))
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8129067c in vfs_write+24 at fs/read_write.c:580
vfs_write (file=0xffff88000e61f200, buf=0x2481390 "\nmnt/sharedFolder/hw03.syscal
580 if (!(file->f_mode & FMODE_WRITE))
>>>

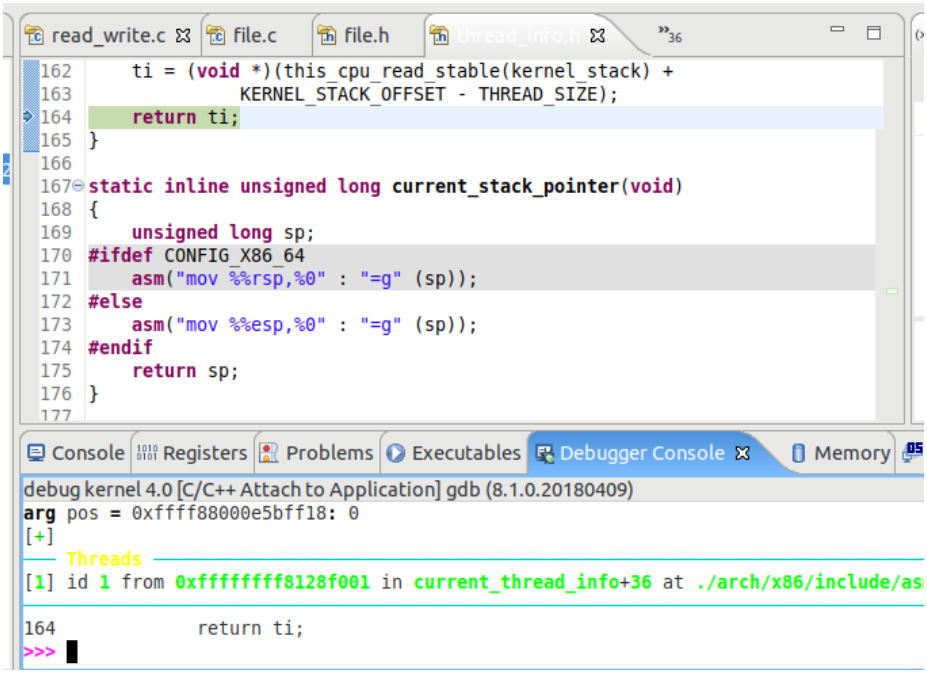

file.c
fdtable.h
file.h
575
576 ssize_t vfs_write(struct file *file, const char __user *buf, size_t cour
577 {
578 ssize_t ret;
579
580 if (!(file->f_mode & FMODE_WRITE))
581 return -EBADF;
582 if (!(file->f_mode & FMODE_CAN_WRITE))
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 1
[+]
Threads
[1] id 1 from 0xffffffff81290696 in vfs_write+50 at fs/read_write.c:582
582 if (!(file->f_mode & FMODE_CAN_WRITE))
>>>

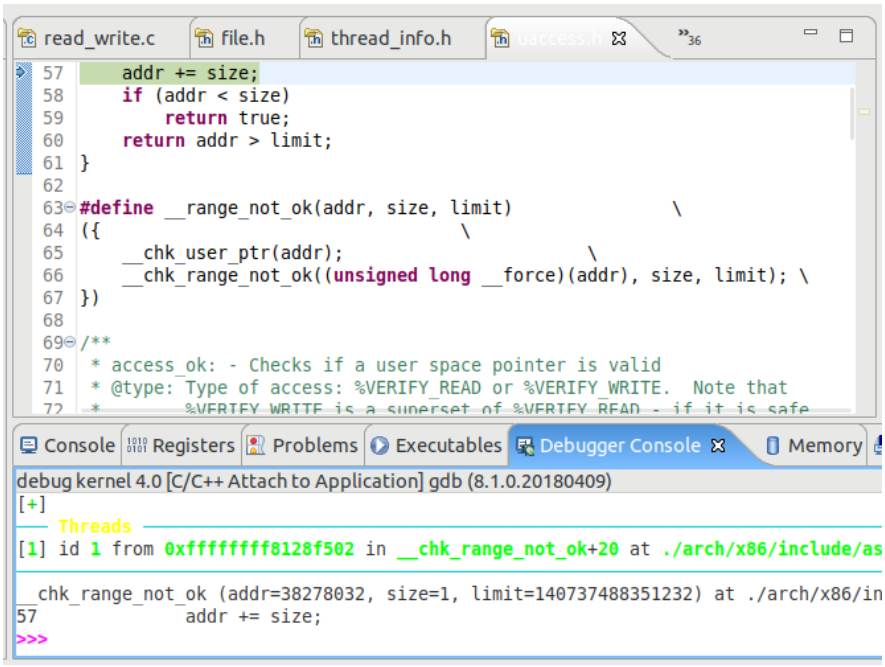

file.c
fdtable.h
file.h
575
576 ssize_t vfs_write(struct file *file, const char __user *buf, size_t cour
577 {
578 ssize_t ret;
579
580 if (!(file->f_mode & FMODE_WRITE))
581 return -EBADF;
582 if (!(file->f_mode & FMODE_CAN_WRITE))
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 1
[+]
Threads
[1] id 1 from 0xffffffff812906b2 in vfs_write+78 at fs/read_write.c:584
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
>>>

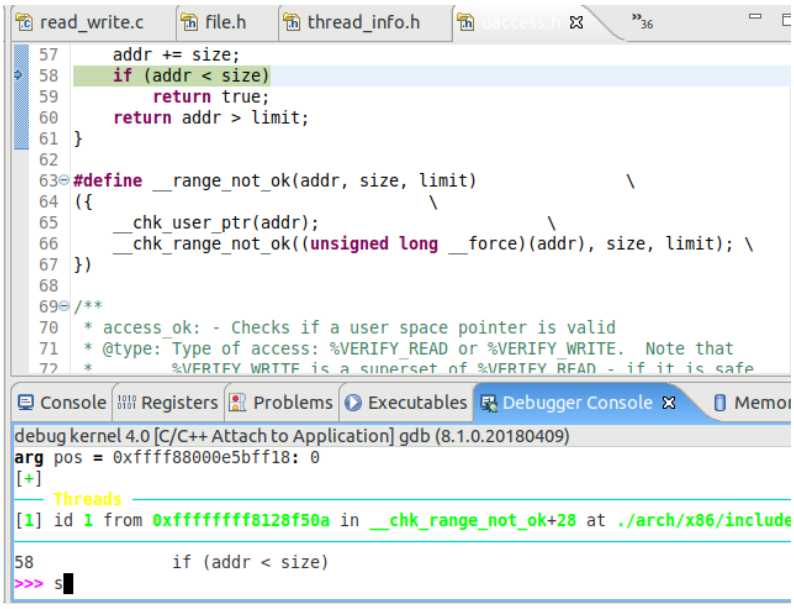

read_write.c
file.c
file.h
thread_info.h
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
163 KERNEL_STACK_OFFSET - THREAD_SIZE);
164 return ti;
165 }
166
167 static inline unsigned long current_stack_pointer(void)
168 {
169 unsigned long sp;
170 #ifdef CONFIG_X86_64
171 asm("mov %%rsp,%0" : "=g" (sp));
172 #else
173 asm("mov %%esp,%0" : "=g" (sp));
174 #endif
175 return sp;
176 }
177
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128efe5 in current_thread_info+8 at ./arch/x86/include/as
current_thread_info () at ./arch/x86/include/asm/thread_info.h:162
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
>>>

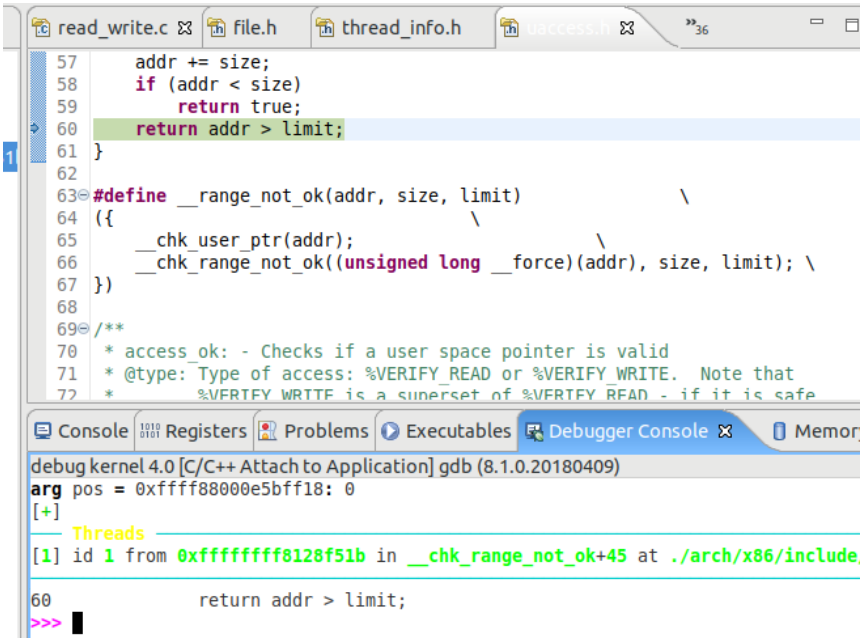

read_write.c
file.c
file.h
thread_info.h
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
163 KERNEL_STACK_OFFSET - THREAD_SIZE);
164 return ti;
165 }
166
167 static inline unsigned long current_stack_pointer(void)
168 {
169 unsigned long sp;
170 #ifdef CONFIG_X86_64
171 asm("mov %%rsp,%0" : "=g" (sp));
172 #else
173 asm("mov %%esp,%0" : "=g" (sp));
174 #endif
175 return sp;
176 }
177
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128eff7 in current_thread_info+26 at ./arch/x86/include
163 KERNEL_STACK_OFFSET - THREAD_SIZE);
>>>

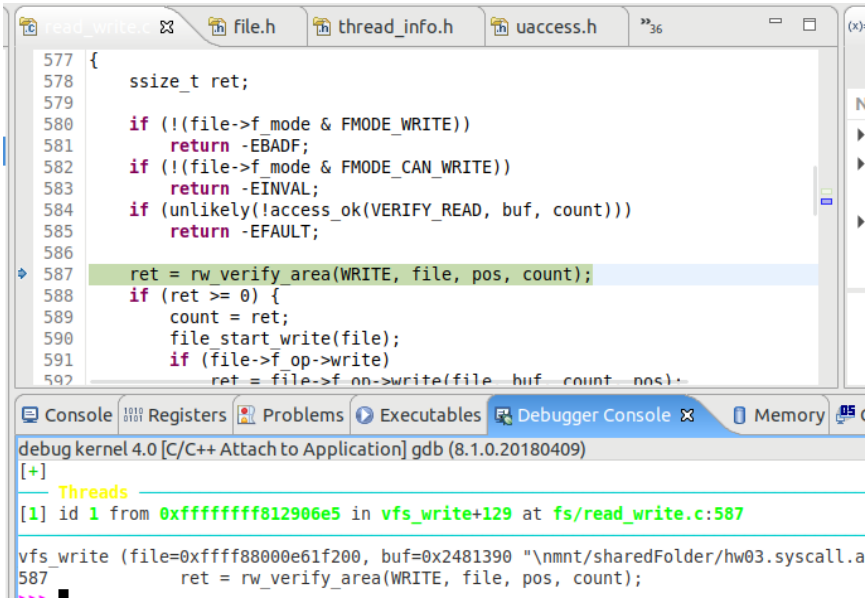

read_write.c
file.c
file.h
thread_info.h
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
163 KERNEL_STACK_OFFSET - THREAD_SIZE);
164 return ti;
165 }
166
167 static inline unsigned long current_stack_pointer(void)
168 {
169 unsigned long sp;
170 #ifdef CONFIG_X86_64
171 asm("mov %%rsp,%0" : "=g" (sp));
172 #else
173 asm("mov %%esp,%0" : "=g" (sp));
174 #endif
175 return sp;
176 }
177
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128efe5 in current_thread_info+8 at ./arch/x86/include/as
current_thread_info () at ./arch/x86/include/asm/thread_info.h:162
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
>>>

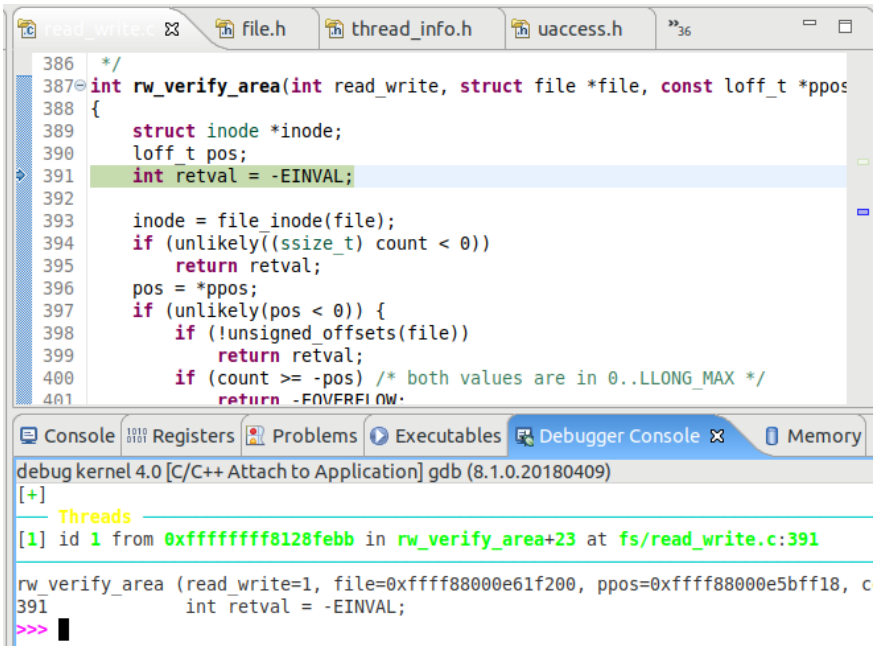

read_write.c
file.c
file.h
thread_info.h
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
163 KERNEL_STACK_OFFSET - THREAD_SIZE);
164 return ti;
165 }
166
167 static inline unsigned long current_stack_pointer(void)
168 {
169 unsigned long sp;
170 #ifdef CONFIG_X86_64
171 asm("mov %%rsp,%0" : "=g" (sp));
172 #else
173 asm("mov %%esp,%0" : "=g" (sp));
174 #endif
175 return sp;
176 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128f001 in current_thread_info+36 at ./arch/x86/include/as
164 return ti;
>>>

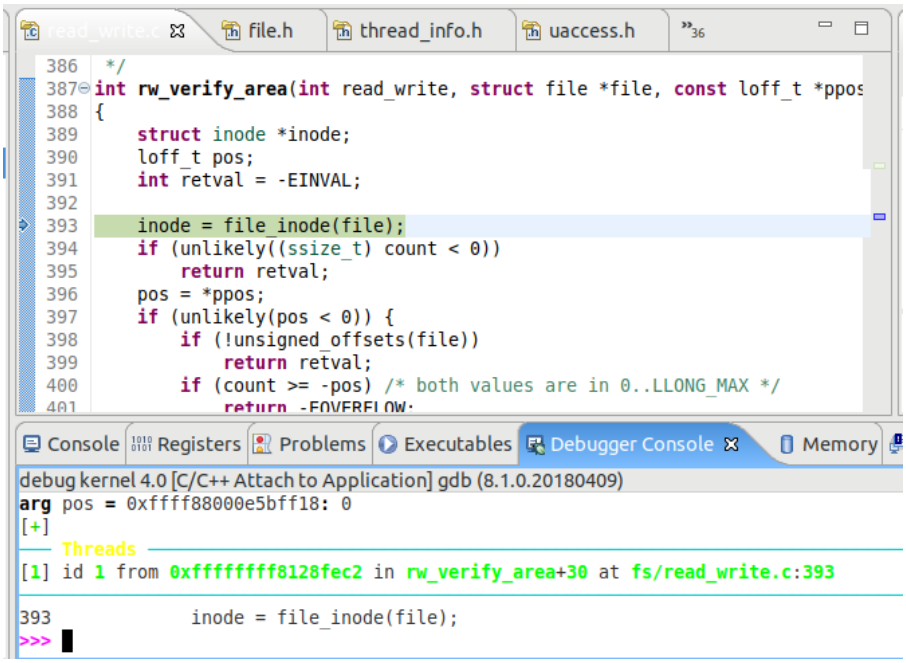

read_write.c
file.h
thread_info.h
57 addr += size;
58 if (addr < size)
59 return true;
60 return addr > limit;
61 }
62
63 #define __range_not_ok(addr, size, limit) \
64 ({ \
65 __chk_user_ptr(addr); \
66 __chk_range_not_ok((unsigned long __force)(addr), size, limit); \
67 })
68
69 /**
70 * access_ok: - Checks if a user space pointer is valid
71 * @type: Type of access: %VERIFY_READ or %VERIFY_WRITE. Note that
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f502 in __chk_range_not_ok+20 at ./arch/x86/include/as
__chk_range_not_ok (addr=38278032, size=1, limit=140737488351232) at ./arch/x86/in
57 addr += size;
>>>

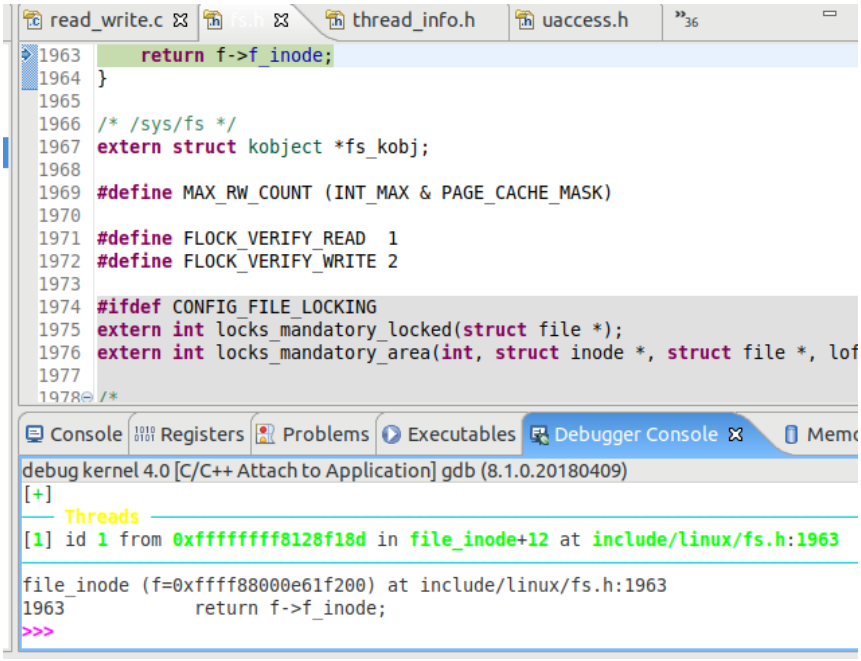

read_write.c
file.h
thread_info.h
57 addr += size;
58 if (addr < size)
59 return true;
60 return addr > limit;
61 }
62
63 #define __range_not_ok(addr, size, limit) \
64 ({ \
65 __chk_user_ptr(addr); \
66 __chk_range_not_ok((unsigned long __force)(addr), size, limit); \
67 })
68
69 /**
70 * access_ok: - Checks if a user space pointer is valid
71 * @type: Type of access: %VERIFY_READ or %VERIFY_WRITE. Note that
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128f50a in __chk_range_not_ok+28 at ./arch/x86/include
58 if (addr < size)
>>> s

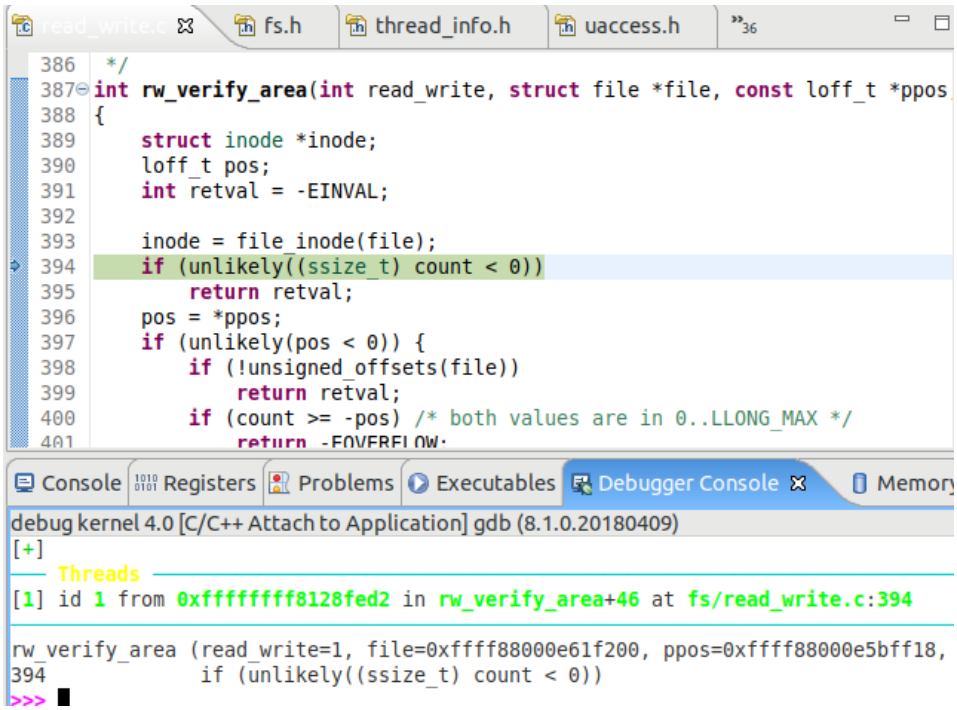

read_write.c
file.h
thread_info.h
uaccess.h
57 addr += size;
58 if (addr < size)
59 return true;
60 return addr > limit;
61 }
62
63 #define __range_not_ok(addr, size, limit) \
64 ({ \
65 __chk_user_ptr(addr); \
66 __chk_range_not_ok((unsigned long __force)(addr), size, limit); \
67 })
68
69 /**
70 * access_ok: - Checks if a user space pointer is valid
71 * @type: Type of access: %VERIFY_READ or %VERIFY_WRITE. Note that
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128f51b in __chk_range_not_ok+45 at ./arch/x86/include
60 return addr > limit;
>>>

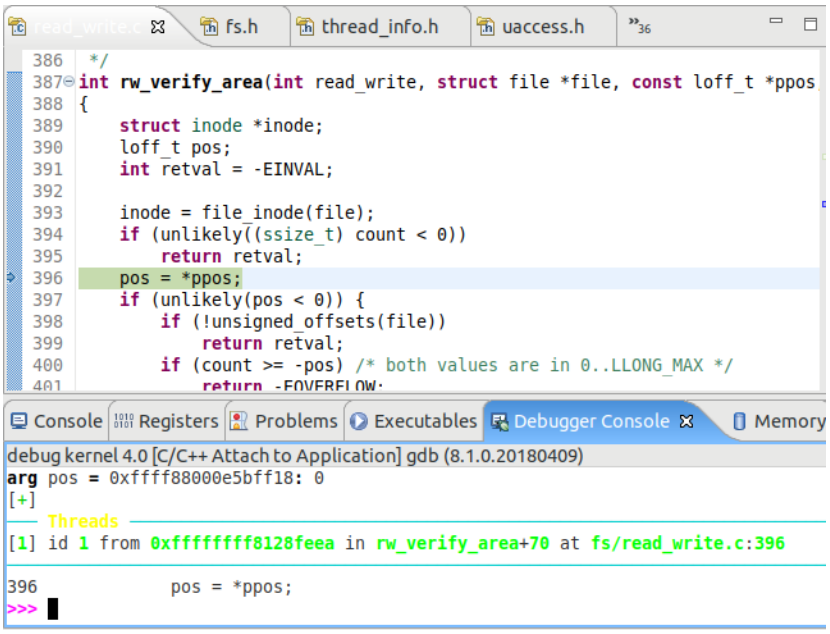

file.h
thread_info.h
uaccess.h
577 {
578 ssize_t ret;
579
580 if (!(file->f_mode & FMODE_WRITE))
581 return -EBADF;
582 if (!(file->f_mode & FMODE_CAN_WRITE))
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
591 if (file->f_op->write)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812906e5 in vfs_write+129 at fs/read_write.c:587
vfs_write (file=0xffff88000e61f200, buf=0x2481390 "\nmnt/sharedFolder/hw03.syscall.a
587 ret = rw_verify_area(WRITE, file, pos, count);

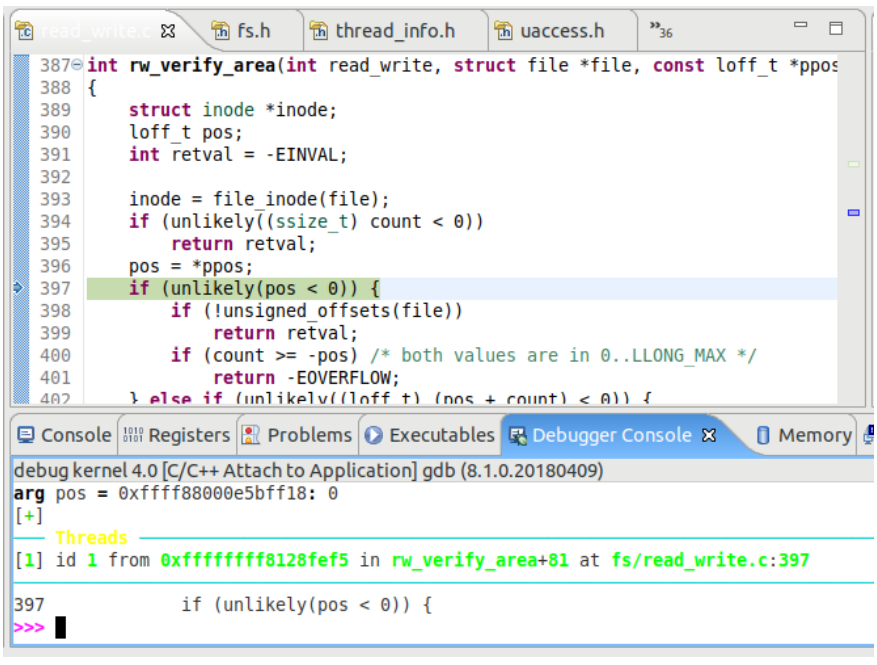

file.h
thread_info.h
uaccess.h
386 */
387 int rw_verify_area(int read_write, struct file *file, const loff_t *ppos
388 {
389 struct inode *inode;
390 loff_t pos;
391 int retval = -EINVAL;
392
393 inode = file_inode(file);
394 if (unlikely((ssize_t) count < 0))
395 return retval;
396 pos = *ppos;
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128febb in rw_verify_area+23 at fs/read_write.c:391
rw_verify_area (read_write=1, file=0xffff88000e61f200, ppos=0xffff88000e5bff18, c
391 int retval = -EINVAL;
>>>

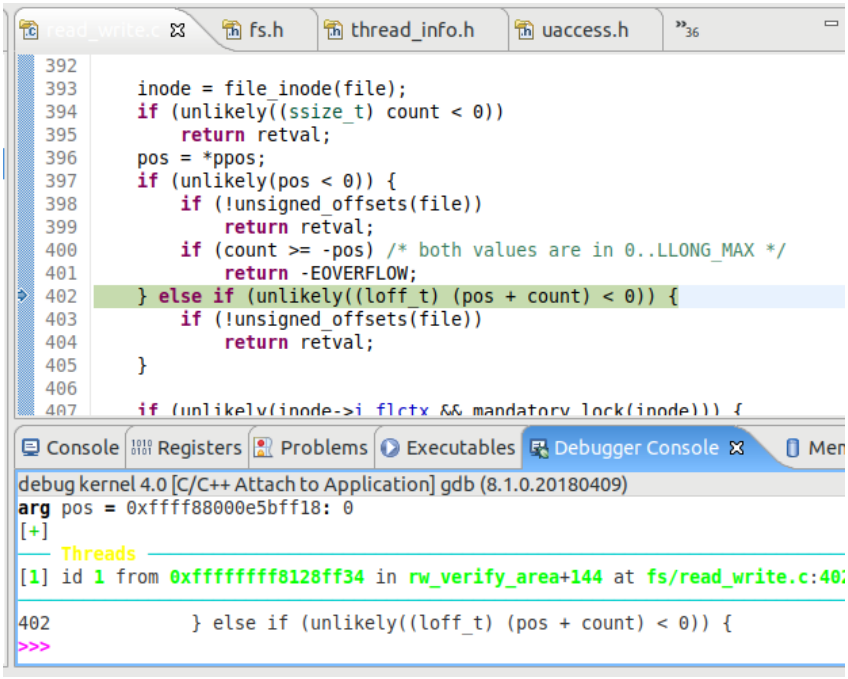

read_write.c
file.h
thread_info.h
uaccess.h
386 */
387 int rw_verify_area(int read_write, struct file *file, const loff_t *ppos
388 {
389 struct inode *inode;
390 loff_t pos;
391 int retval = -EINVAL;
392
393 inode = file_inode(file);
394 if (unlikely((ssize_t) count < 0))
395 return retval;
396 pos = *ppos;
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
401 return -EOVERFLOW;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128fec2 in rw_verify_area+30 at fs/read_write.c:393
393 inode = file_inode(file);
>>>

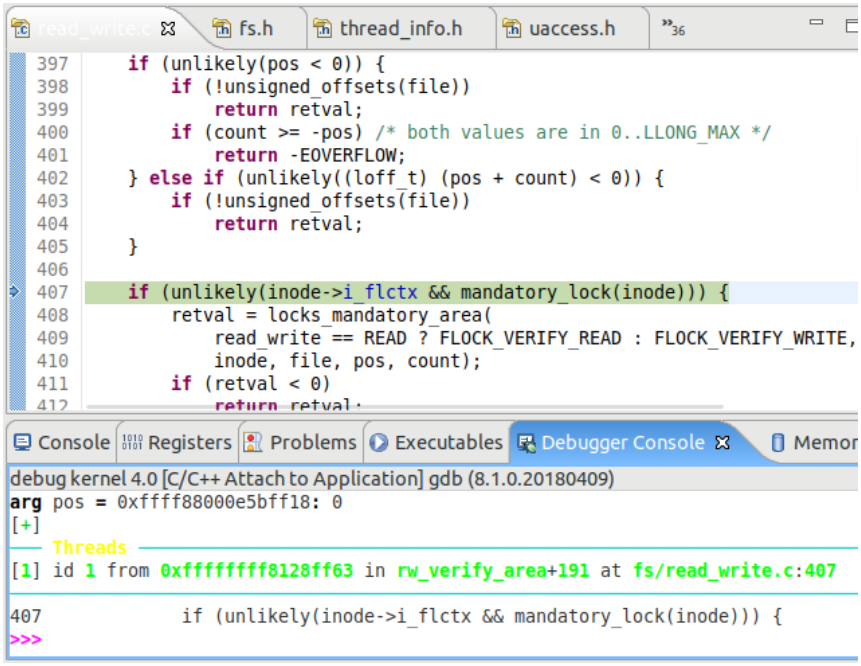

read_write.c
thread_info.h
uaccess.h
1963 return f->f_inode;
1964 }
1965
1966 /* /sys/fs */
1967 extern struct kobject *fs_kobj;
1968
1969 #define MAX_RW_COUNT (INT_MAX & PAGE_CACHE_MASK)
1970
1971 #define FLOCK_VERIFY_READ 1
1972 #define FLOCK_VERIFY_WRITE 2
1973
1974 #ifdef CONFIG_FILE_LOCKING
1975 extern int locks_mandatory_locked(struct file *);
1976 extern int locks_mandatory_area(int, struct inode *, struct file *, lof
1977
1978 /*
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f18d in file_inode+12 at include/linux/fs.h:1963
file_inode (f=0xffff88000e61f200) at include/linux/fs.h:1963
1963 return f->f_inode;
>>>

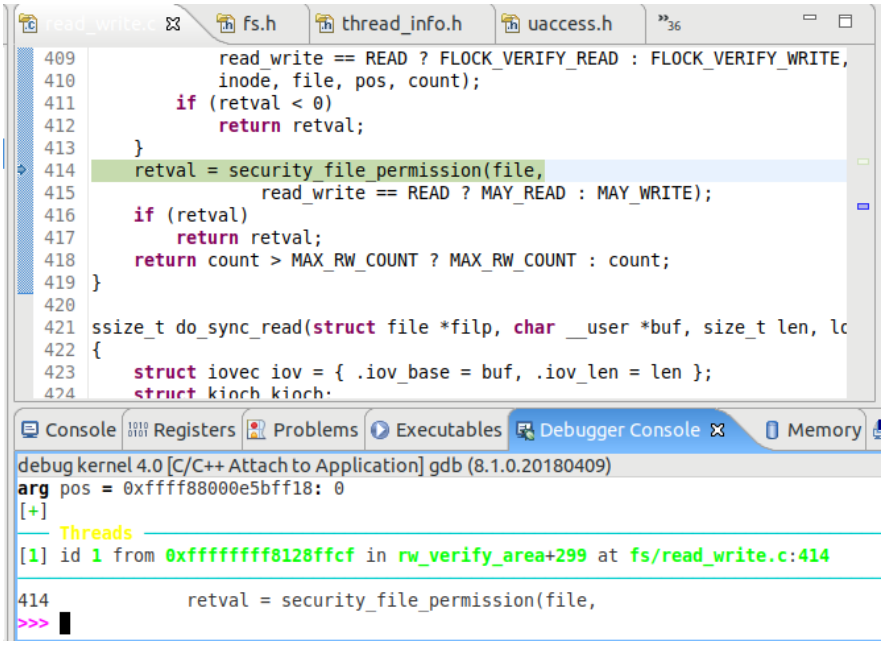

fs.h
thread_info.h
uaccess.h
386 */
387 int rw_verify_area(int read_write, struct file *file, const loff_t *ppos
388 {
389 struct inode *inode;
390 loff_t pos;
391 int retval = -EINVAL;
392
393 inode = file_inode(file);
394 if (unlikely((ssize_t) count < 0))
395 return retval;
396 pos = *ppos;
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128fed2 in rw_verify_area+46 at fs/read_write.c:394
rw_verify_area (read_write=1, file=0xffff88000e61f200, ppos=0xffff88000e5bff18,
394 if (unlikely((ssize_t) count < 0))
>>>

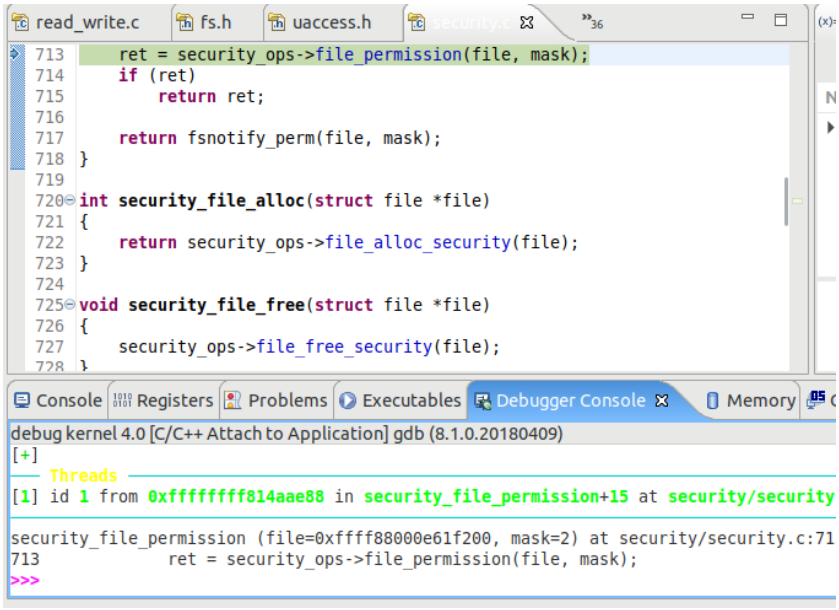

fs.h
thread_info.h
uaccess.h
386 */
387 int rw_verify_area(int read_write, struct file *file, const loff_t *ppos
388 {
389 struct inode *inode;
390 loff_t pos;
391 int retval = -EINVAL;
392
393 inode = file_inode(file);
394 if (unlikely((ssize_t) count < 0))
395 return retval;
396 pos = *ppos;
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128feea in rw_verify_area+70 at fs/read_write.c:396
396 pos = *ppos;
>>>

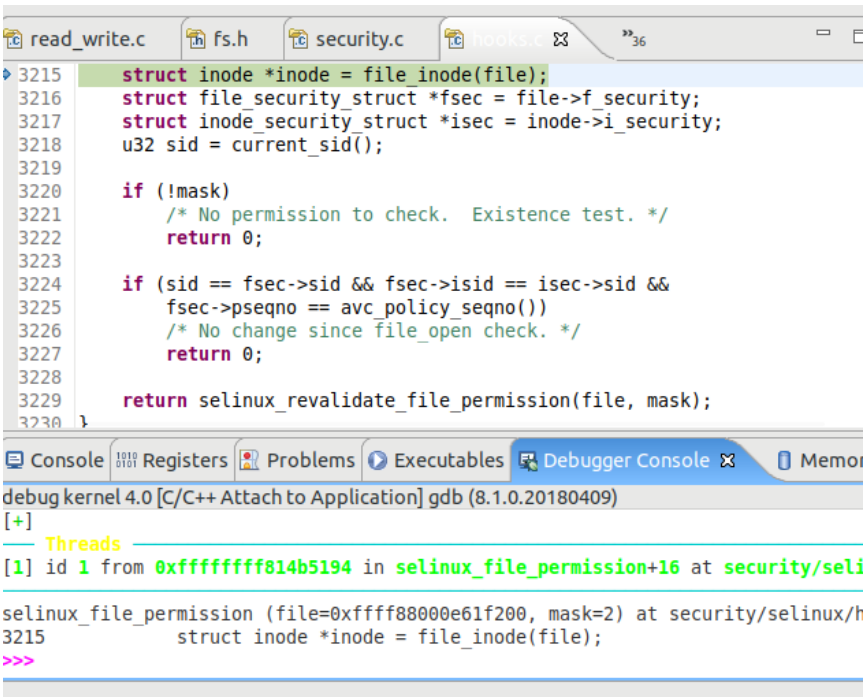

fs.h
thread_info.h
uaccess.h
387 int rw_verify_area(int read_write, struct file *file, const loff_t *ppos
388 {
389 struct inode *inode;
390 loff_t pos;
391 int retval = -EINVAL;
392
393 inode = file_inode(file);
394 if (unlikely((ssize_t) count < 0))
395 return retval;
396 pos = *ppos;
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
401 return -EOVERFLOW;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128fef5 in rw_verify_area+81 at fs/read_write.c:397
397 if (unlikely(pos < 0)) {
>>>

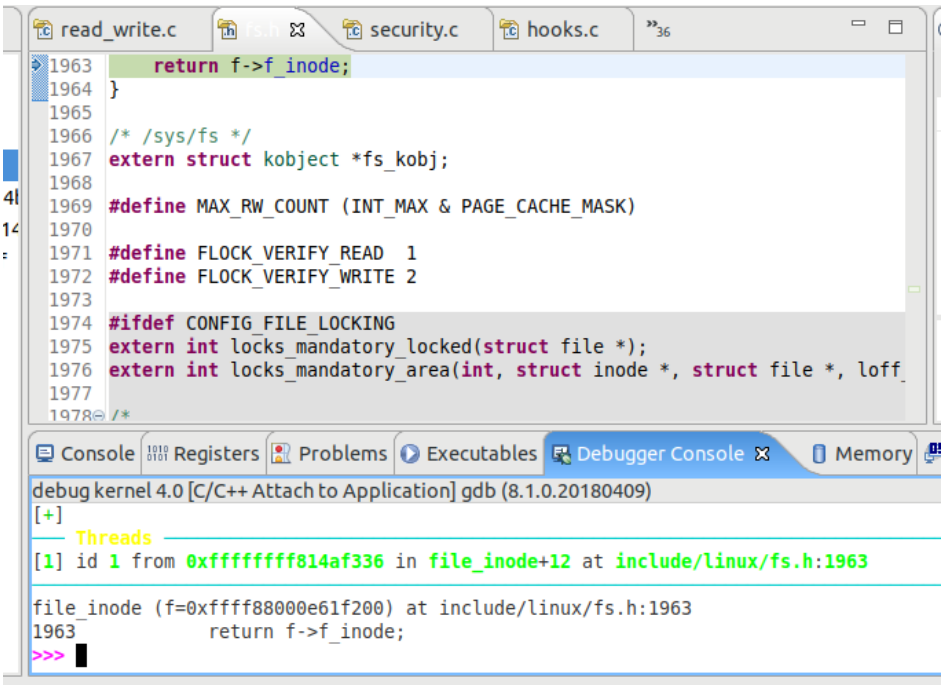

fs.h
thread_info.h
uaccess.h
392
393 inode = file_inode(file);
394 if (unlikely((ssize_t) count < 0))
395 return retval;
396 pos = *ppos;
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
401 return -EOVERFLOW;
402 } else if (unlikely((loff_t) (pos + count) < 0)) {
403 if (!unsigned_offsets(file))
404 return retval;
405 }
406
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128ff34 in rw_verify_area+144 at fs/read_write.c:40
402 } else if (unlikely((loff_t) (pos + count) < 0)) {
>>>

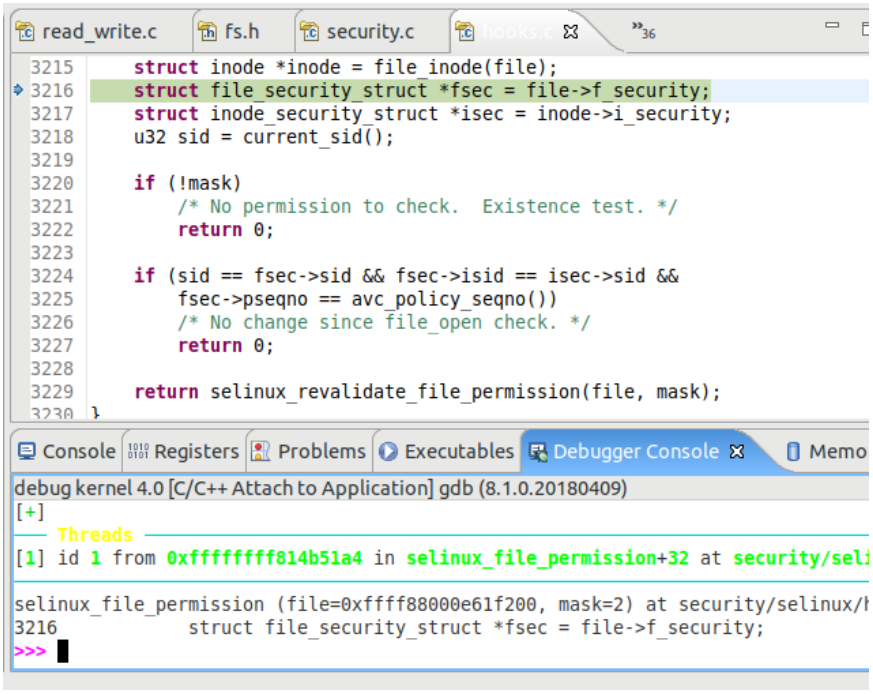

fs.h
thread_info.h
uaccess.h
397 if (unlikely(pos < 0)) {
398 if (!unsigned_offsets(file))
399 return retval;
400 if (count >= -pos) /* both values are in 0..LLONG_MAX */
401 return -EOVERFLOW;
402 } else if (unlikely((loff_t) (pos + count) < 0)) {
403 if (!unsigned_offsets(file))
404 return retval;
405 }
406
407 if (unlikely(inode->i_flctx && mandatory_lock(inode))) {
408 retval = locks_mandatory_area(
409 read_write == READ ? FLOCK_VERIFY_READ : FLOCK_VERIFY_WRITE,
410 inode, file, pos, count);
411 if (retval < 0)
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128ff63 in rw_verify_area+191 at fs/read_write.c:407
407 if (unlikely(inode->i_flctx && mandatory_lock(inode))) {
>>>

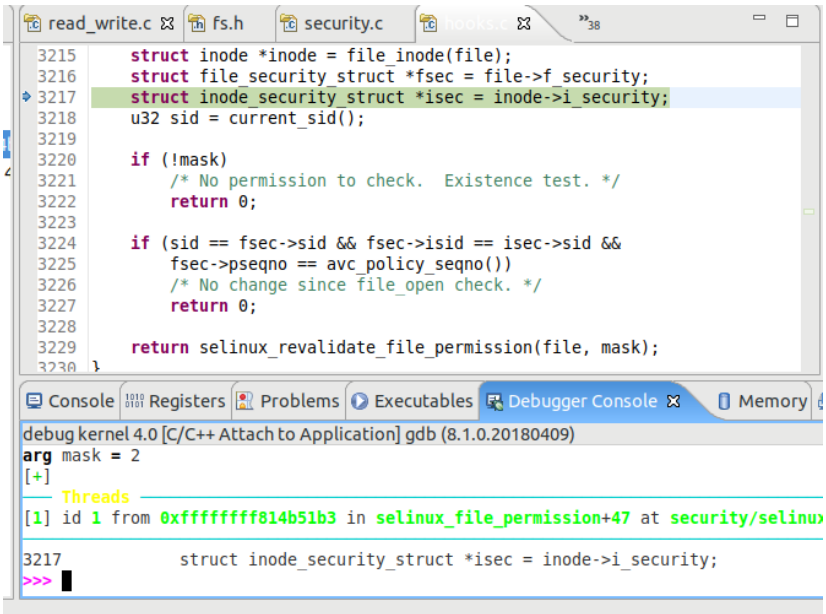

fs.h
thread_info.h
uaccess.h
409 read_write == READ ? FLOCK_VERIFY_READ : FLOCK_VERIFY_WRITE,
410 inode, file, pos, count);
411 if (retval < 0)
412 return retval;
413 }
414 retval = security_file_permission(file,
415 read_write == READ ? MAY_READ : MAY_WRITE);
416 if (retval)
417 return retval;
418 return count > MAX_RW_COUNT ? MAX_RW_COUNT : count;
419 }
420
421 ssize_t do_sync_read(struct file *filp, char __user *buf, size_t len, lo
422 {
423 struct iovec iov = { .iov_base = buf, .iov_len = len };
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 0
[+]
Threads
[1] id 1 from 0xffffffff8128ffcf in rw_verify_area+299 at fs/read_write.c:414
414 retval = security_file_permission(file,
>>>

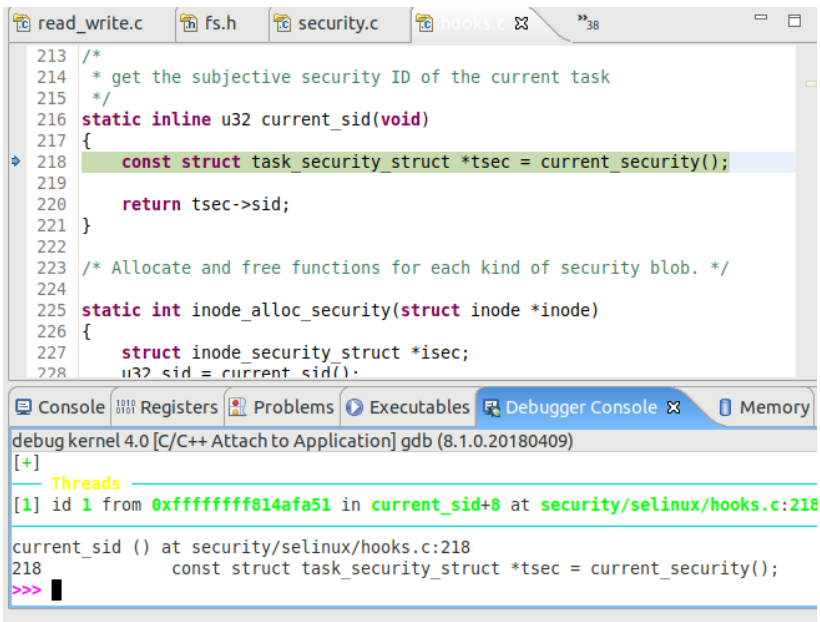

read_write.c
fs.h
uaccess.h
713 ret = security_ops->file_permission(file, mask);
714 if (ret)
715 return ret;
716
717 return fsnotify_perm(file, mask);
718 }
719
720 int security_file_alloc(struct file *file)
721 {
722 return security_ops->file_alloc_security(file);
723 }
724
725 void security_file_free(struct file *file)
726 {
727 security_ops->file_free_security(file);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814aae88 in security_file_permission+15 at security/security
security_file_permission (file=0xffff88000e61f200, mask=2) at security/security.c:71
713 ret = security_ops->file_permission(file, mask);
>>>

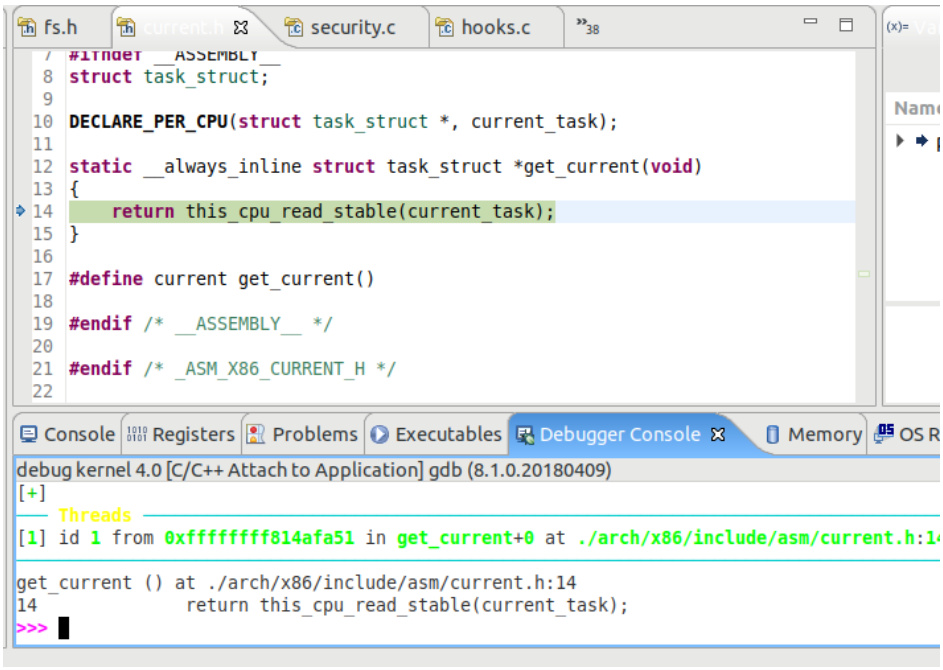

read_write.c
fs.h
security.c
3215 struct inode *inode = file_inode(file);
3216 struct file_security_struct *fsec = file->f_security;
3217 struct inode_security_struct *isec = inode->i_security;
3218 u32 sid = current_sid();
3219
3220 if (!mask)
3221 /* No permission to check. Existence test. */
3222 return 0;
3223
3224 if (sid == fsec->sid && fsec->isid == isec->sid &&
3225 fsec->pseqno == avc_policy_seqno())
3226 /* No change since file_open check. */
3227 return 0;
3228
3229 return selinux_revalidate_file_permission(file, mask);
3230 }
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814b5194 in selinux_file_permission+16 at security/seli
selinux_file_permission (file=0xffff88000e61f200, mask=2) at security/selinux/h
3215 struct inode *inode = file_inode(file);
>>>

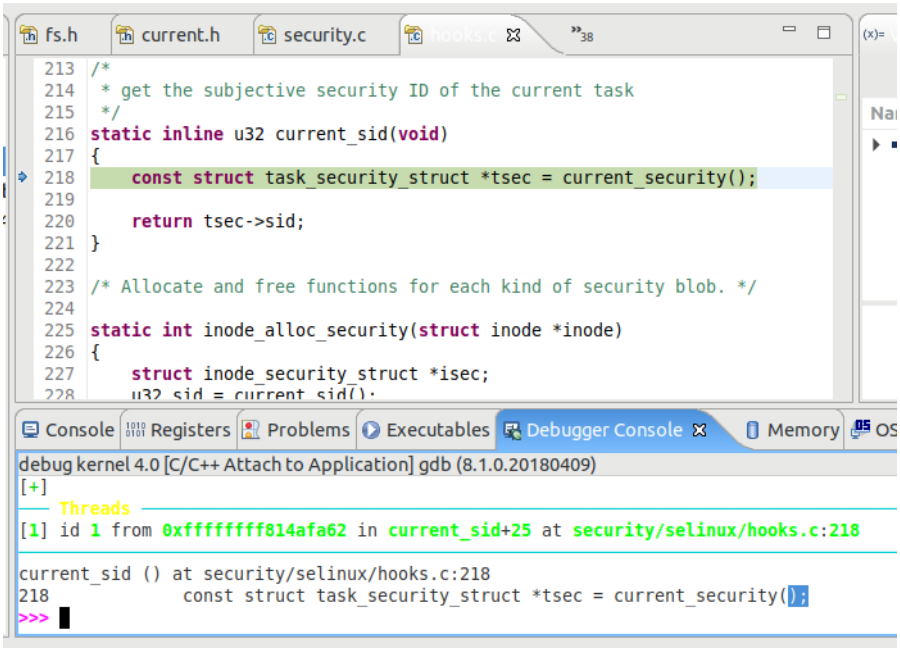

read_write.c
security.c
hooks.c
1963 return f->f_inode;
1964 }
1965
1966 /* /sys/fs */
1967 extern struct kobject *fs_kobj;
1968
1969 #define MAX_RW_COUNT (INT_MAX & PAGE_CACHE_MASK)
1970
1971 #define FLOCK_VERIFY_READ 1
1972 #define FLOCK_VERIFY_WRITE 2
1973
1974 #ifdef CONFIG_FILE_LOCKING
1975 extern int locks_mandatory_locked(struct file *);
1976 extern int locks_mandatory_area(int, struct inode *, struct file *, loff_
1977
1978 /*
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814af336 in file_inode+12 at include/linux/fs.h:1963
file_inode (f=0xffff88000e61f200) at include/linux/fs.h:1963
1963 return f->f_inode;
>>>

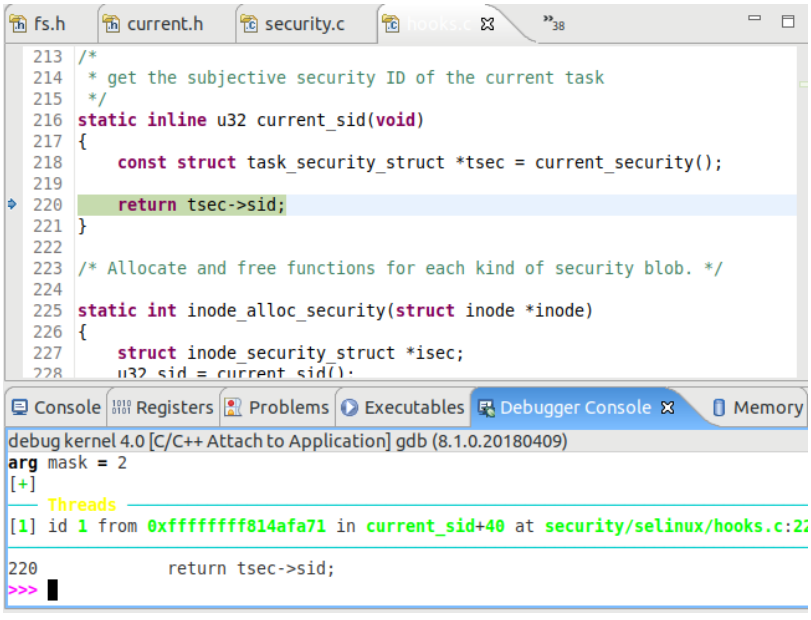

read_write.c
fs.h
security.c
hooks.c
3215 struct inode *inode = file_inode(file);
3216 struct file_security_struct *fsec = file->f_security;
3217 struct inode_security_struct *isec = inode->i_security;
3218 u32 sid = current_sid();
3219
3220 if (!mask)
3221 /* No permission to check. Existence test. */
3222 return 0;
3223
3224 if (sid == fsec->sid && fsec->isid == isec->sid &&
3225 fsec->pseqno == avc_policy_seqno())
3226 /* No change since file_open check. */
3227 return 0;
3228
3229 return selinux_revalidate_file_permission(file, mask);
3230 }
Console
Registers
Problems
Executables
Debugger Console
Memo
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814b51a4 in selinux_file_permission+32 at security/seli
selinux_file_permission (file=0xffff88000e61f200, mask=2) at security/selinux/h
3216 struct file_security_struct *fsec = file->f_security;
>>>

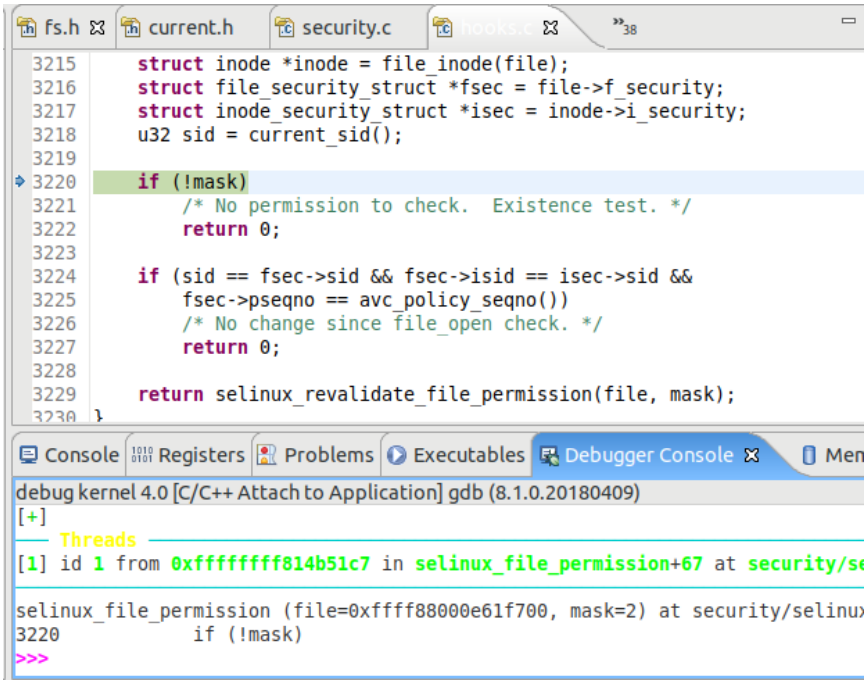

read_write.c
fs.h
security.c
hooks.c
3215 struct inode *inode = file_inode(file);
3216 struct file_security_struct *fsec = file->f_security;
3217 struct inode_security_struct *isec = inode->i_security;
3218 u32 sid = current_sid();
3219
3220 if (!mask)
3221 /* No permission to check. Existence test. */
3222 return 0;
3223
3224 if (sid == fsec->sid && fsec->isid == isec->sid &&
3225 fsec->pseqno == avc_policy_seqno())
3226 /* No change since file_open check. */
3227 return 0;
3228
3229 return selinux_revalidate_file_permission(file, mask);
3230 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg mask = 2
[+]
Threads
[1] id 1 from 0xffffffff814b51b3 in selinux_file_permission+47 at security/selinux
3217 struct inode_security_struct *isec = inode->i_security;
>>>

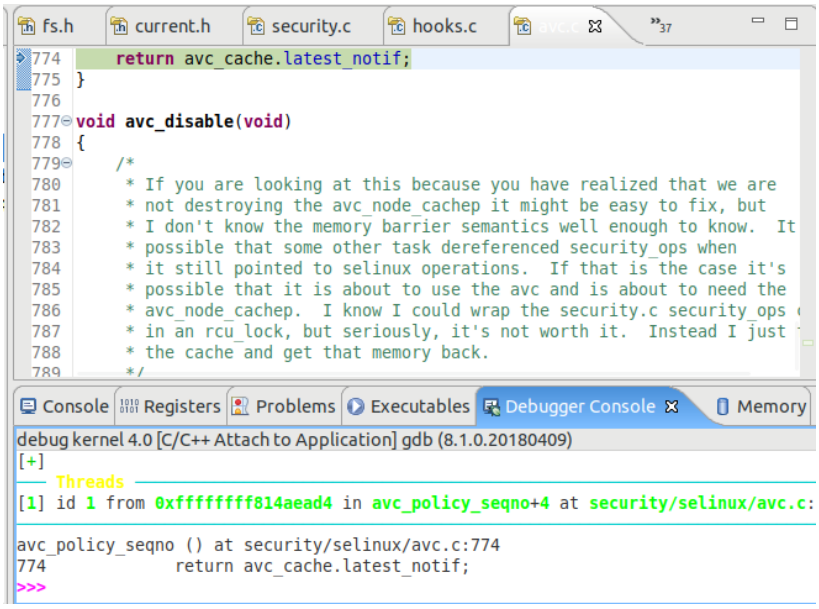

read_write.c
fs.h
security.c
hooks.c
213 /*
214 * get the subjective security ID of the current task
215 */
216 static inline u32 current_sid(void)
217 {
218 const struct task_security_struct *tsec = current_security();
219
220 return tsec->sid;
221 }
222
223 /* Allocate and free functions for each kind of security blob. */
224
225 static int inode_alloc_security(struct inode *inode)
226 {
227 struct inode_security_struct *isec;
228 u32 sid = current_sid();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814afa51 in current_sid+8 at security/selinux/hooks.c:218
current_sid () at security/selinux/hooks.c:218
218 const struct task_security_struct *tsec = current_security();
>>>

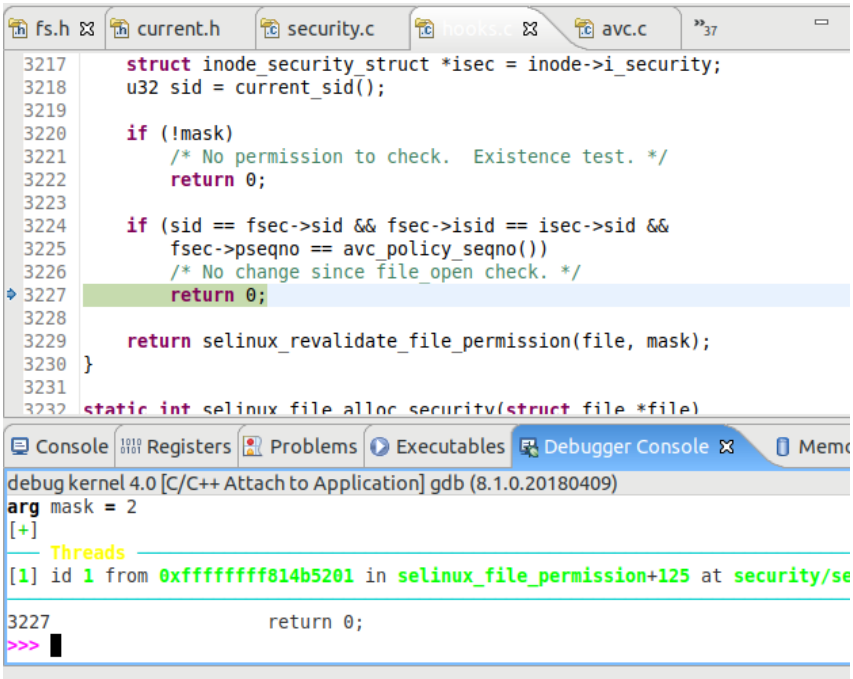

fs.h
security.c
hooks.c
#ifndef __ASSEMBLY__
struct task_struct;
DECLARE_PER_CPU(struct task_struct *, current_task);
static __always_inline struct task_struct *get_current(void)
{
return this_cpu_read_stable(current_task);
}
#define current get_current()
#endif /* __ASSEMBLY__ */
#endif /* _ASM_X86_CURRENT_H */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814afa51 in get_current+0 at ./arch/x86/include/asm/current.h:14
get_current () at ./arch/x86/include/asm/current.h:14
14 return this_cpu_read_stable(current_task);
>>>

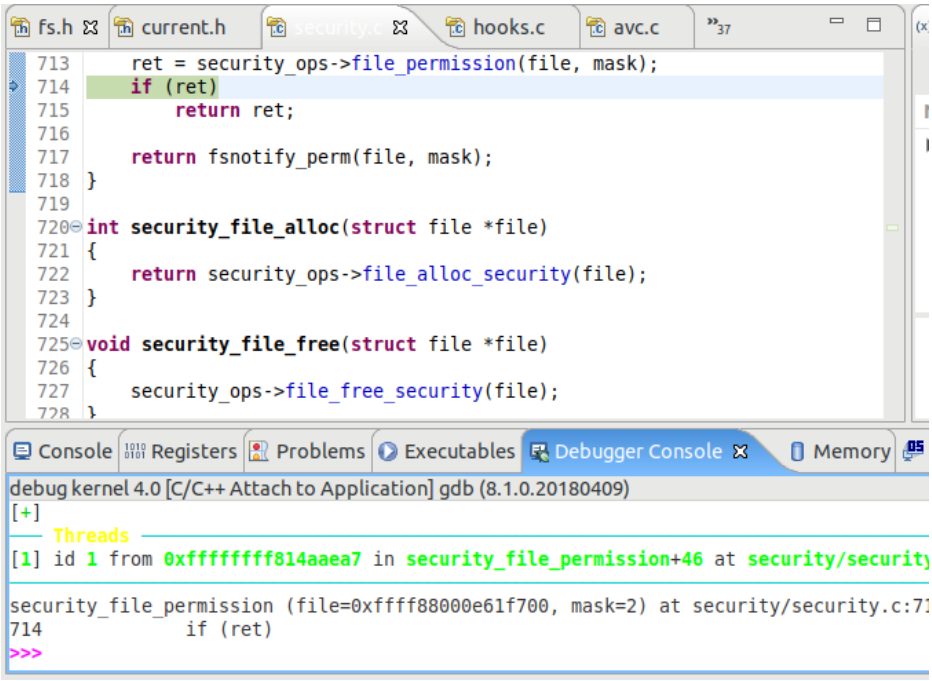

fs.h
current.h
security.c
/*
* get the subjective security ID of the current task
*/
static inline u32 current_sid(void)
{
const struct task_security_struct *tsec = current_security();
return tsec->sid;
}
/* Allocate and free functions for each kind of security blob. */
static int inode_alloc_security(struct inode *inode)
{
struct inode_security_struct *isec;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814afa62 in current_sid+25 at security/selinux/hooks.c:218
current_sid () at security/selinux/hooks.c:218
218 const struct task_security_struct *tsec = current_security();
>>>

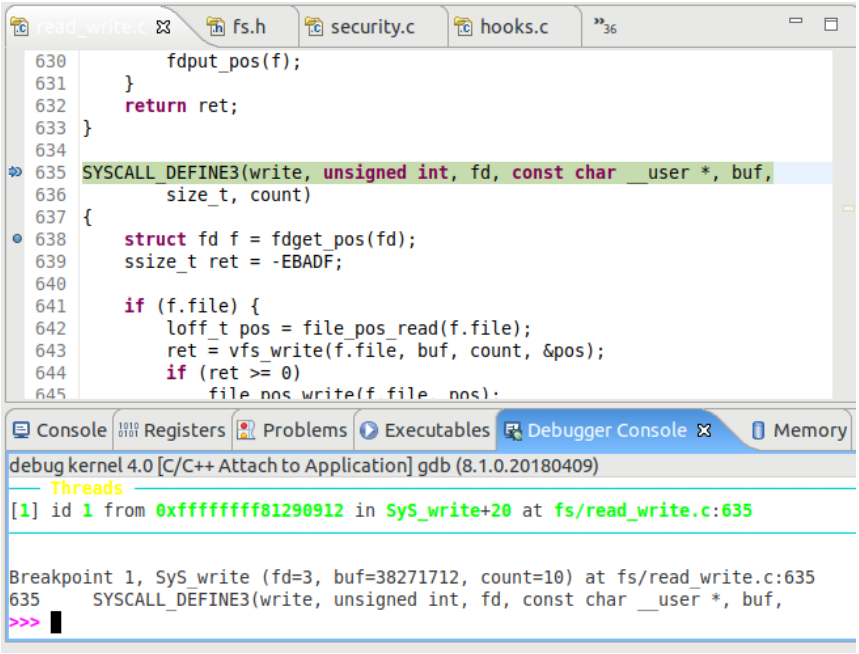

fs.h
current.h
security.c
/*
* get the subjective security ID of the current task
*/
static inline u32 current_sid(void)
{
const struct task_security_struct *tsec = current_security();
return tsec->sid;
}
/* Allocate and free functions for each kind of security blob. */
static int inode_alloc_security(struct inode *inode)
{
struct inode_security_struct *isec;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg mask = 2
[+]
Threads
[1] id 1 from 0xffffffff814afa71 in current_sid+40 at security/selinux/hooks.c:22
220 return tsec->sid;
>>>

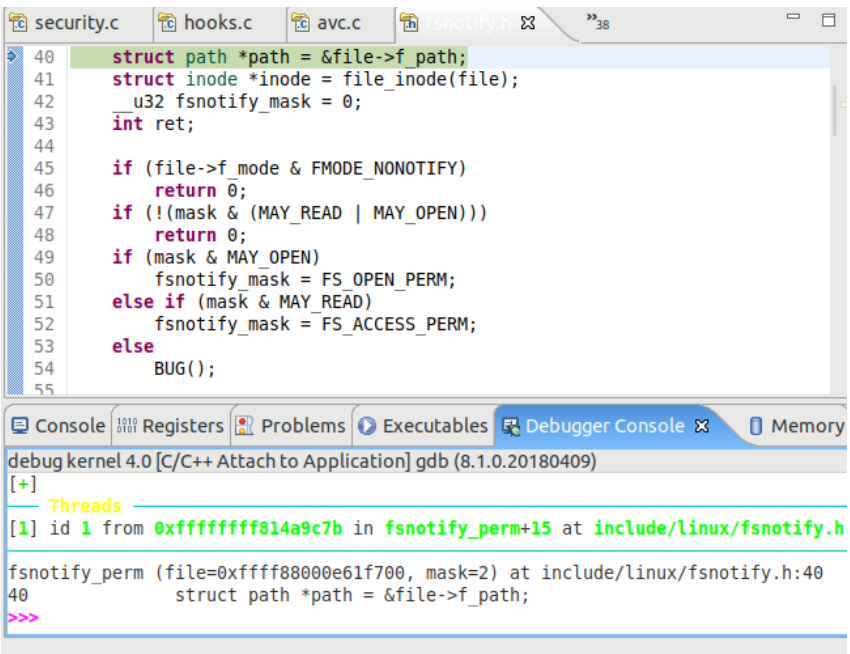

fs.h
current.h
security.c
hooks.c
3215 struct inode *inode = file_inode(file);
3216 struct file_security_struct *fsec = file->f_security;
3217 struct inode_security_struct *isec = inode->i_security;
3218 u32 sid = current_sid();
3219
3220 if (!mask)
3221 /* No permission to check. Existence test. */
3222 return 0;
3223
3224 if (sid == fsec->sid && fsec->isid == isec->sid &&
3225 fsec->pseqno == avc_policy_seqno())
3226 /* No change since file_open check. */
3227 return 0;
3228
3229 return selinux_revalidate_file_permission(file, mask);
3230 }
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814b51c7 in selinux_file_permission+67 at security/se
selinux_file_permission (file=0xffff88000e61f700, mask=2) at security/selinux
3220 if (!mask)
>>>

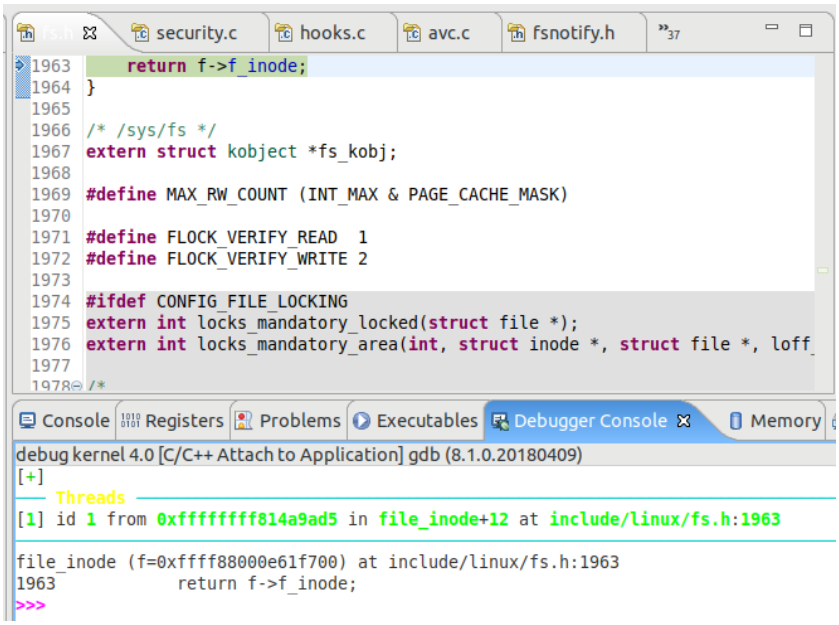

fs.h
current.h
security.c
hooks.c
774 return avc_cache.latest_notif;
775 }
776
777 void avc_disable(void)
778 {
779 /*
780 * If you are looking at this because you have realized that we are
781 * not destroying the avc_node_cachep it might be easy to fix, but
782 * I don't know the memory barrier semantics well enough to know. It
783 * possible that some other task dereferenced security_ops when
784 * it still pointed to selinux operations. If that is the case it's
785 * possible that it is about to use the avc and is about to need the
786 * avc_node_cachep. I know I could wrap the security.c security_ops
787 * in an rcu_lock, but seriously, it's not worth it. Instead I just
788 * the cache and get that memory back.
789 */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814aead4 in avc_policy_seqno+4 at security/selinux/avc.c:
avc_policy_seqno () at security/selinux/avc.c:774
774 return avc_cache.latest_notif;
>>>

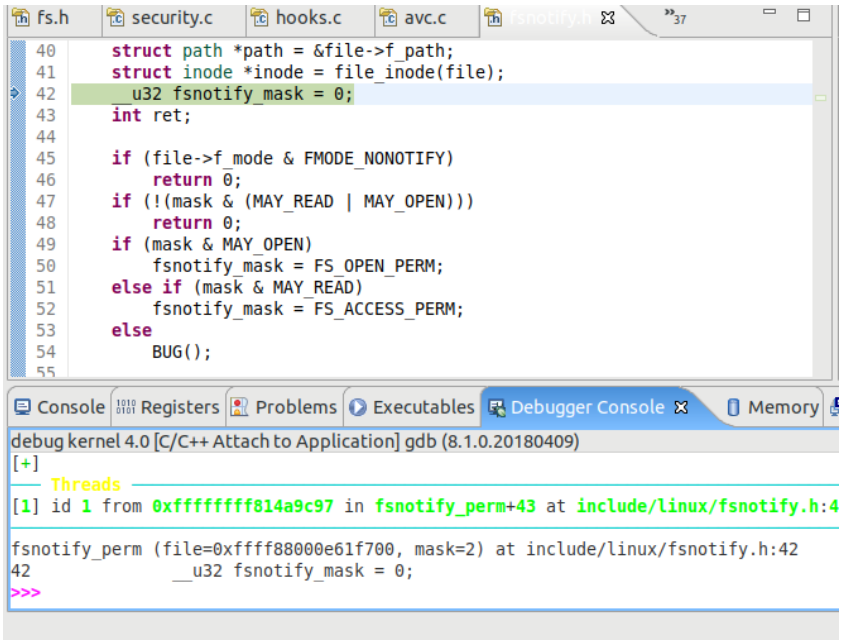

fs.h
current.h
security.c
hooks.c
avc.c
3217 struct inode_security_struct *isec = inode->i_security;
3218 u32 sid = current_sid();
3219
3220 if (!mask)
3221 /* No permission to check. Existence test. */
3222 return 0;
3223
3224 if (sid == fsec->sid && fsec->isid == isec->sid &&
3225 fsec->pseqno == avc_policy_seqno())
3226 /* No change since file_open check. */
3227 return 0;
3228
3229 return selinux_revalidate_file_permission(file, mask);
3230 }
3231
3232 static int selinux_file_alloc_security(struct file *file)
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg mask = 2
[+]
Threads
[1] id 1 from 0xffffffff814b5201 in selinux_file_permission+125 at security/se
3227 return 0;
>>>

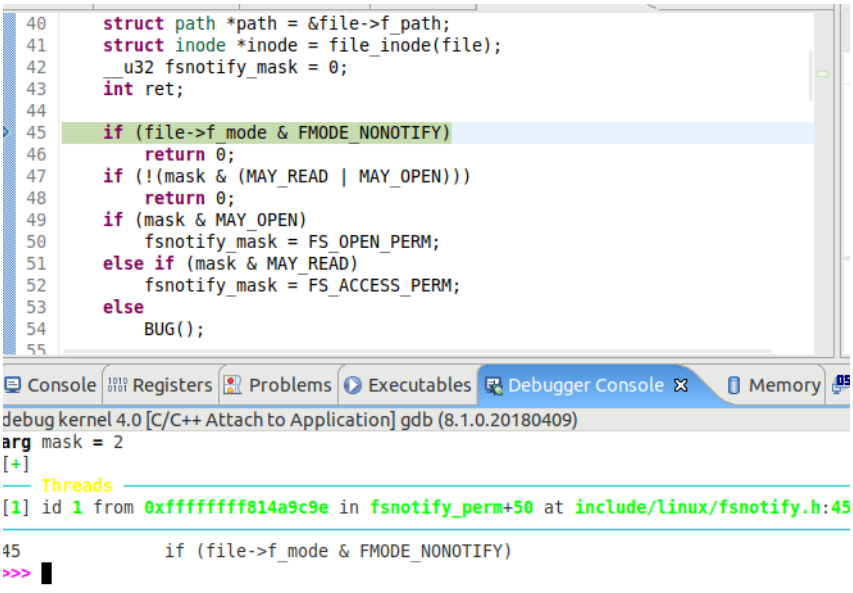

fs.h
current.h
hooks.c
avc.c
713 ret = security_ops->file_permission(file, mask);
714 if (ret)
715 return ret;
716
717 return fsnotify_perm(file, mask);
718 }
719
720 int security_file_alloc(struct file *file)
721 {
722 return security_ops->file_alloc_security(file);
723 }
724
725 void security_file_free(struct file *file)
726 {
727 security_ops->file_free_security(file);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814aaea7 in security_file_permission+46 at security/securit
security_file_permission (file=0xffff88000e61f700, mask=2) at security/security.c:7
714 if (ret)
>>>

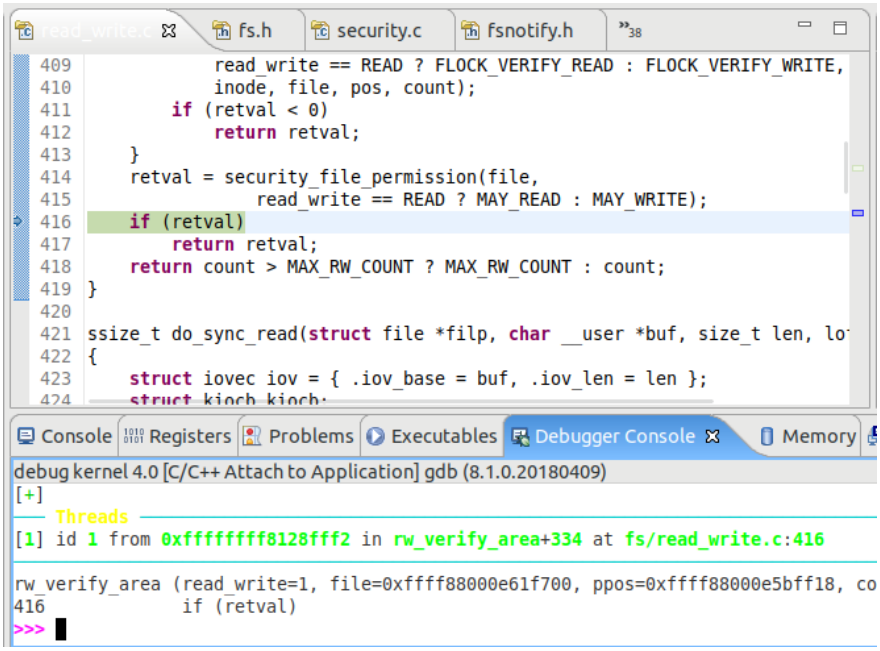

fs.h
security.c
hooks.c
630 fdput_pos(f);
631 }
632 return ret;
633 }
634
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
636 size_t, count)
637 {
638 struct fd f = fdget_pos(fd);
639 ssize_t ret = -EBADF;
640
641 if (f.file) {
642 loff_t pos = file_pos_read(f.file);
643 ret = vfs_write(f.file, buf, count, &pos);
644 if (ret >= 0)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff81290912 in SyS_write+20 at fs/read_write.c:635
Breakpoint 1, SyS_write (fd=3, buf=38271712, count=10) at fs/read_write.c:635
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
>>>

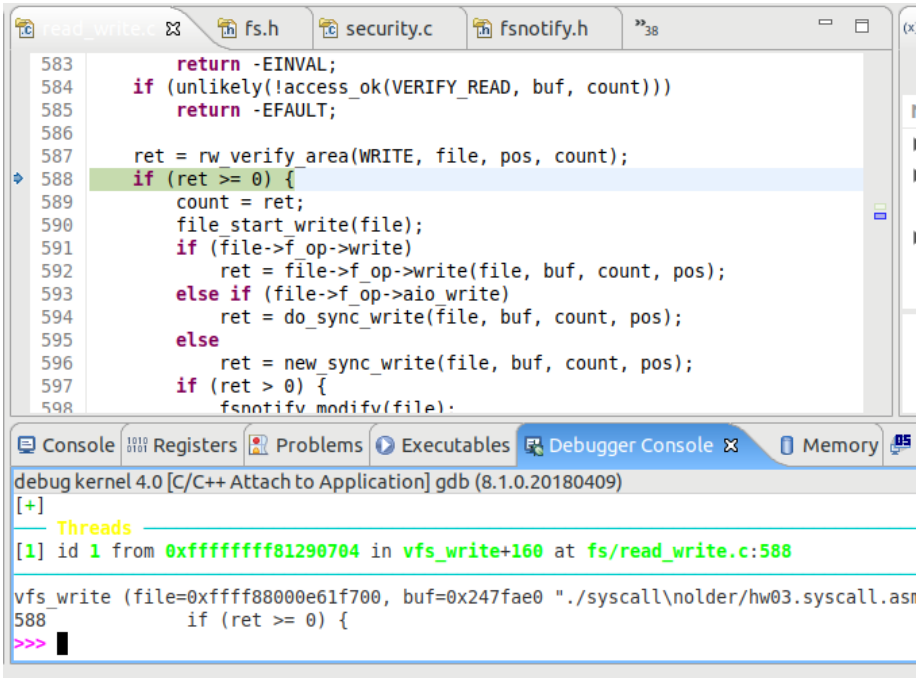

security.c
hooks.c
avc.c
40 struct path *path = &file->f_path;
41 struct inode *inode = file_inode(file);
42 __u32 fsnotify_mask = 0;
43 int ret;
44
45 if (file->f_mode & FMODE_NONOTIFY)
46 return 0;
47 if (!(mask & (MAY_READ | MAY_OPEN)))
48 return 0;
49 if (mask & MAY_OPEN)
50 fsnotify_mask = FS_OPEN_PERM;
51 else if (mask & MAY_READ)
52 fsnotify_mask = FS_ACCESS_PERM;
53 else
54 BUG();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814a9c7b in fsnotify_perm+15 at include/linux/fsnotify.h
fsnotify_perm (file=0xffff88000e61f700, mask=2) at include/linux/fsnotify.h:40
40 struct path *path = &file->f_path;
>>>

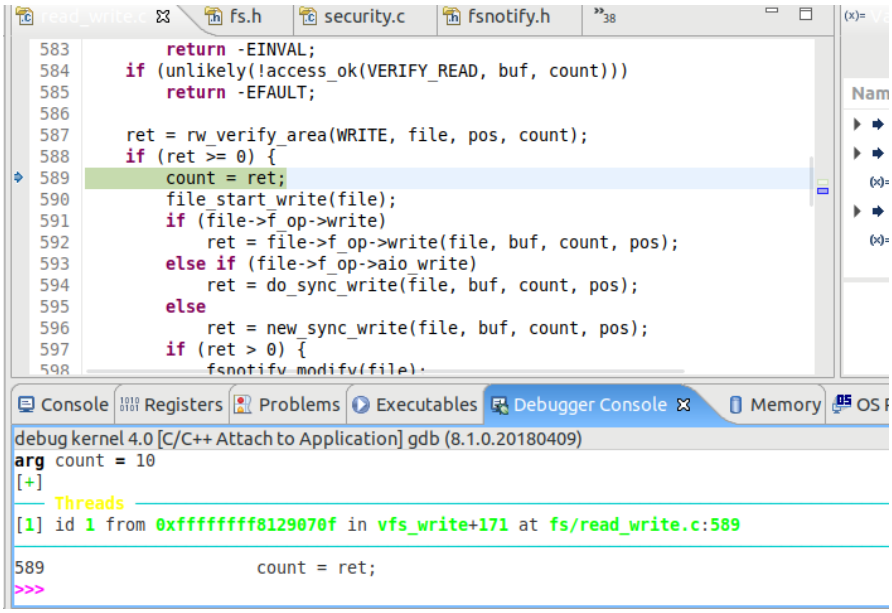

security.c
hooks.c
avc.c
fsnotify.h
1963 return f->f_inode;
1964 }
1965
1966 /* /sys/fs */
1967 extern struct kobject *fs_kobj;
1968
1969 #define MAX_RW_COUNT (INT_MAX & PAGE_CACHE_MASK)
1970
1971 #define FLOCK_VERIFY_READ 1
1972 #define FLOCK_VERIFY_WRITE 2
1973
1974 #ifdef CONFIG_FILE_LOCKING
1975 extern int locks_mandatory_locked(struct file *);
1976 extern int locks_mandatory_area(int, struct inode *, struct file *, loff_
1977
1978 /*
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814a9ad5 in file_inode+12 at include/linux/fs.h:1963
file_inode (f=0xffff88000e61f700) at include/linux/fs.h:1963
1963 return f->f_inode;
>>>

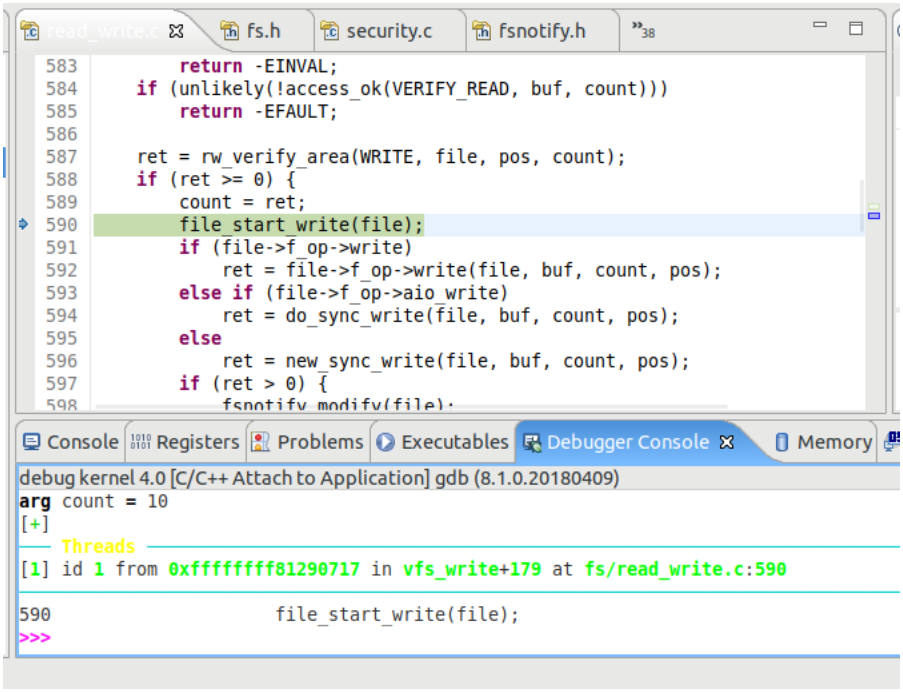

fs.h
security.c
hooks.c
avc.c
40 struct path *path = &file->f_path;
41 struct inode *inode = file_inode(file);
42 __u32 fsnotify_mask = 0;
43 int ret;
44
45 if (file->f_mode & FMODE_NONOTIFY)
46 return 0;
47 if (!(mask & (MAY_READ | MAY_OPEN)))
48 return 0;
49 if (mask & MAY_OPEN)
50 fsnotify_mask = FS_OPEN_PERM;
51 else if (mask & MAY_READ)
52 fsnotify_mask = FS_ACCESS_PERM;
53 else
54 BUG();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff814a9c97 in fsnotify_perm+43 at include/linux/fsnotify.h:4
fsnotify_perm (file=0xffff88000e61f700, mask=2) at include/linux/fsnotify.h:42
42 __u32 fsnotify_mask = 0;
>>>

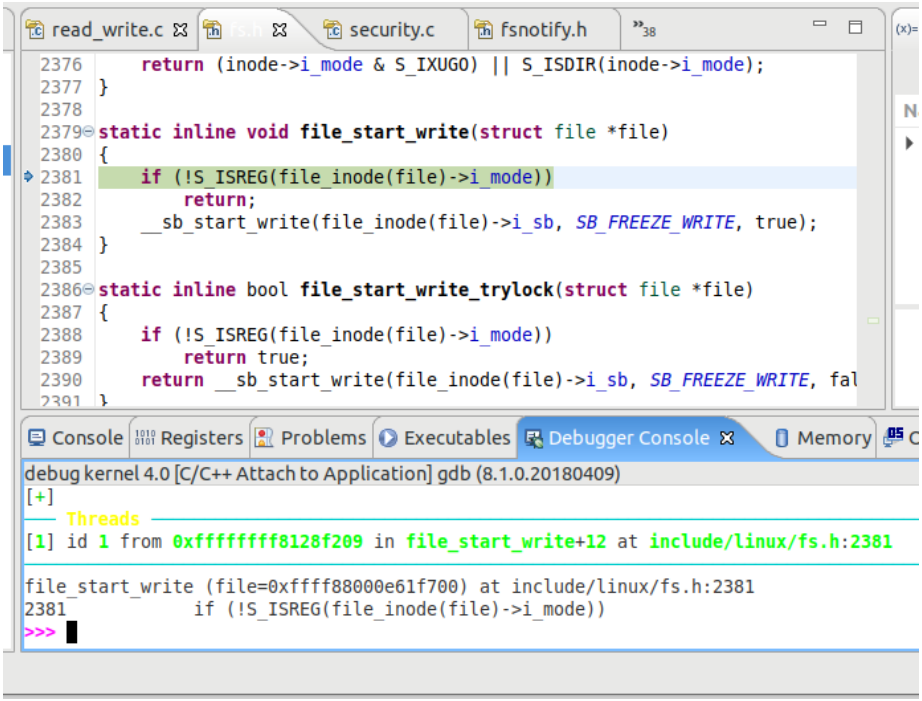

40 struct path *path = &file->f_path;
41 struct inode *inode = file_inode(file);
42 __u32 fsnotify_mask = 0;
43 int ret;
44
45 if (file->f_mode & FMODE_NONOTIFY)
46 return 0;
47 if (!(mask & (MAY_READ | MAY_OPEN)))
48 return 0;
49 if (mask & MAY_OPEN)
50 fsnotify_mask = FS_OPEN_PERM;
51 else if (mask & MAY_READ)
52 fsnotify_mask = FS_ACCESS_PERM;
53 else
54 BUG();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg mask = 2
[+]
Threads
[1] id 1 from 0xffffffff814a9c9e in fsnotify_perm+50 at include/linux/fsnotify.h:45
45 if (file->f_mode & FMODE_NONOTIFY)
>>>

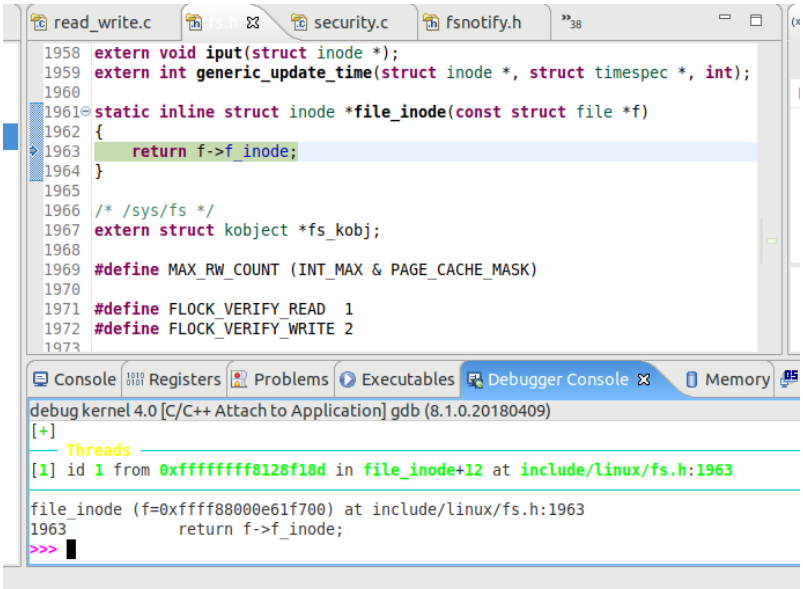

fs.h
security.c
fsnotify.h
409 read_write == READ ? FLOCK_VERIFY_READ : FLOCK_VERIFY_WRITE,
410 inode, file, pos, count);
411 if (retval < 0)
412 return retval;
413 }
414 retval = security_file_permission(file,
415 read_write == READ ? MAY_READ : MAY_WRITE);
416 if (retval)
417 return retval;
418 return count > MAX_RW_COUNT ? MAX_RW_COUNT : count;
419 }
420
421 ssize_t do_sync_read(struct file *filp, char __user *buf, size_t len, lo
422 {
423 struct iovec iov = { .iov_base = buf, .iov_len = len };
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128fff2 in rw_verify_area+334 at fs/read_write.c:416
rw_verify_area (read_write=1, file=0xffff88000e61f700, ppos=0xffff88000e5bff18, co
416 if (retval)
>>>

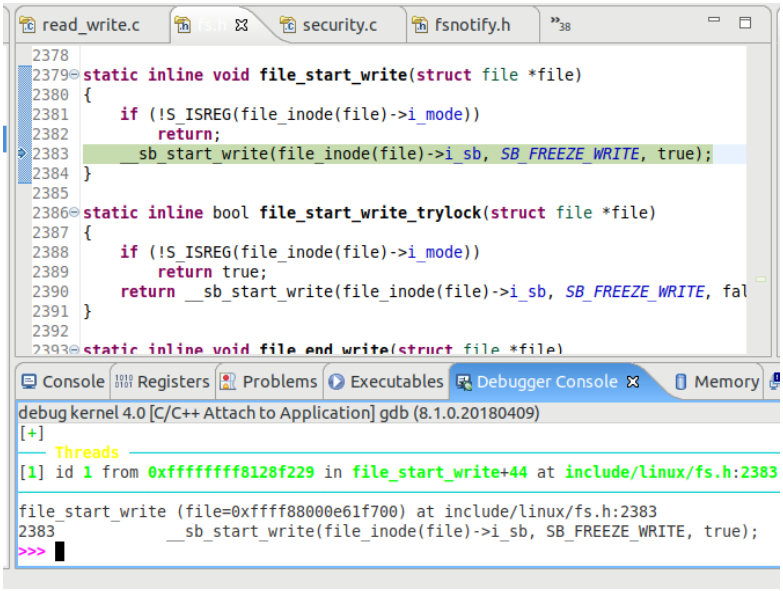

fs.h
security.c
fsnotify.h
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
591 if (file->f_op->write)
592 ret = file->f_op->write(file, buf, count, pos);
593 else if (file->f_op->aio_write)
594 ret = do_sync_write(file, buf, count, pos);
595 else
596 ret = new_sync_write(file, buf, count, pos);
597 if (ret > 0) {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290704 in vfs_write+160 at fs/read_write.c:588
vfs_write (file=0xffff88000e61f700, buf=0x247fae0 "./syscall\nolder/hw03.syscall.asm
588 if (ret >= 0) {
>>>

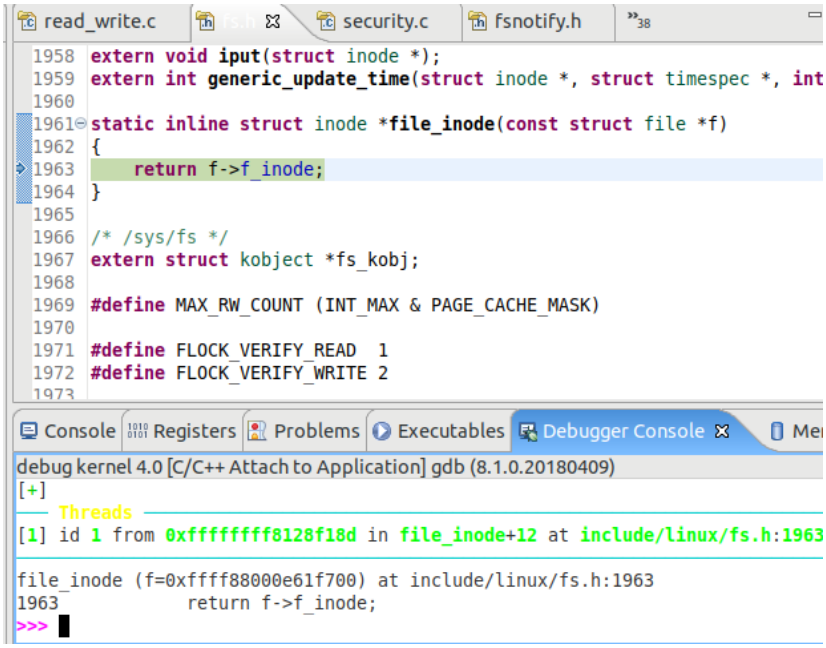

fs.h
security.c
fsnotify.h
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
591 if (file->f_op->write)
592 ret = file->f_op->write(file, buf, count, pos);
593 else if (file->f_op->aio_write)
594 ret = do_sync_write(file, buf, count, pos);
595 else
596 ret = new_sync_write(file, buf, count, pos);
597 if (ret > 0) {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 10
[+]
Threads
[1] id 1 from 0xffffffff8129070f in vfs_write+171 at fs/read_write.c:589
589 count = ret;
>>>

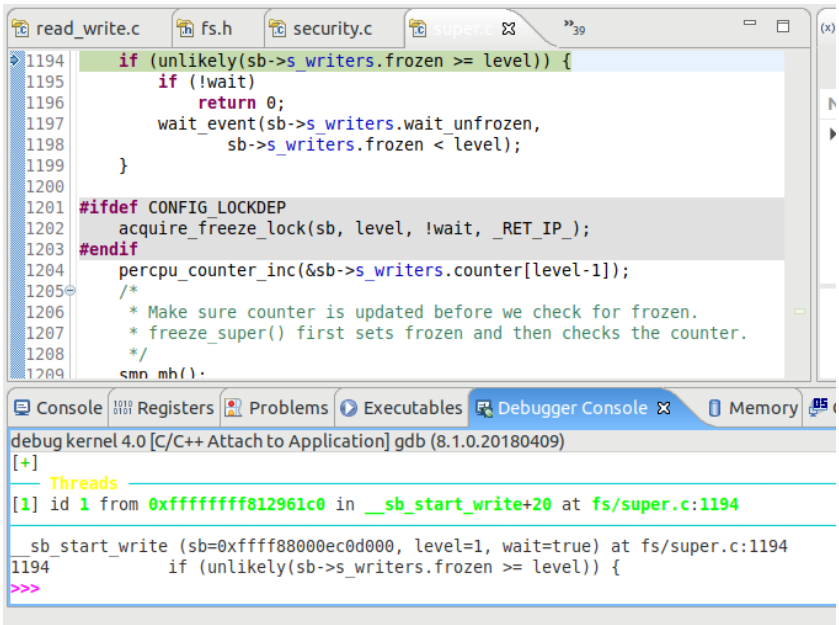

fs.h
security.c
fsnotify.h
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
591 if (file->f_op->write)
592 ret = file->f_op->write(file, buf, count, pos);
593 else if (file->f_op->aio_write)
594 ret = do_sync_write(file, buf, count, pos);
595 else
596 ret = new_sync_write(file, buf, count, pos);
597 if (ret > 0) {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 10
[+]
Threads
[1] id 1 from 0xffffffff81290717 in vfs_write+179 at fs/read_write.c:590
590 file_start_write(file);
>>>

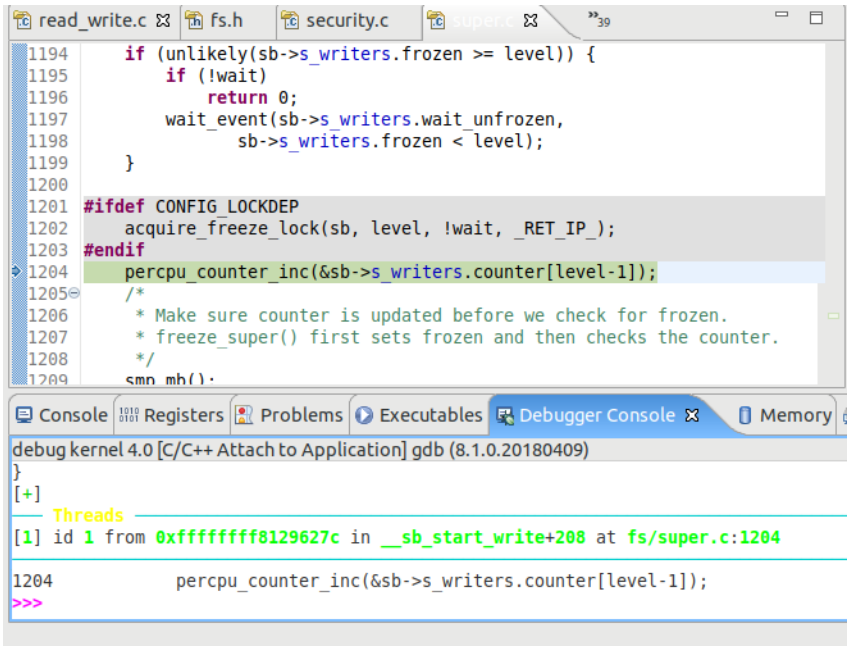

read_write.c
security.c
fsnotify.h
2376 return (inode->i_mode & S_IXUGO) || S_ISDIR(inode->i_mode);
2377 }
2378
2379 static inline void file_start_write(struct file *file)
2380 {
2381 if (!S_ISREG(file_inode(file)->i_mode))
2382 return;
2383 __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, true);
2384 }
2385
2386 static inline bool file_start_write_trylock(struct file *file)
2387 {
2388 if (!S_ISREG(file_inode(file)->i_mode))
2389 return true;
2390 return __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, fal
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f209 in file_start_write+12 at include/linux/fs.h:2381
file_start_write (file=0xffff88000e61f700) at include/linux/fs.h:2381
2381 if (!S_ISREG(file_inode(file)->i_mode))
>>>

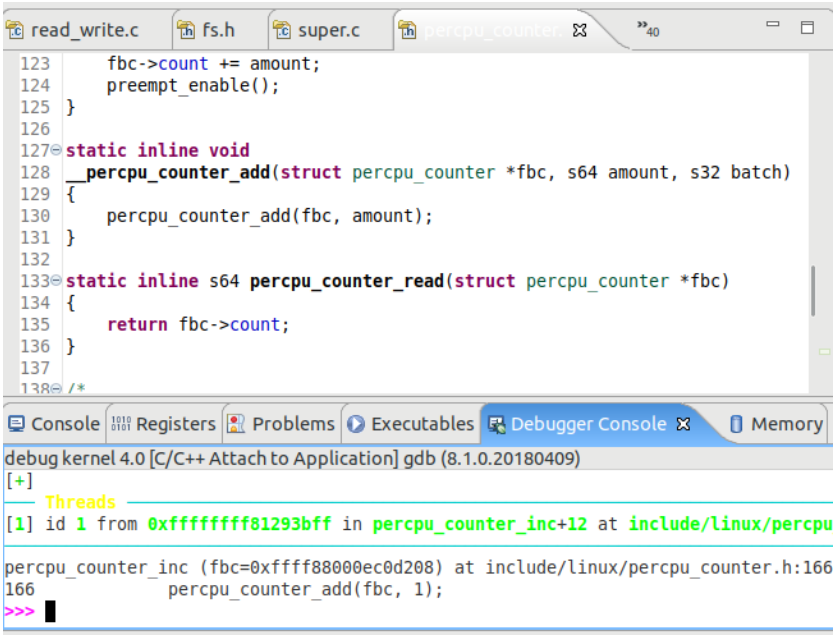

read_write.c
security.c
fsnotify.h
1958 extern void iput(struct inode *);
1959 extern int generic_update_time(struct inode *, struct timespec *, int);
1960
1961 static inline struct inode *file_inode(const struct file *f)
1962 {
1963 return f->f_inode;
1964 }
1965
1966 /* /sys/fs */
1967 extern struct kobject *fs_kobj;
1968
1969 #define MAX_RW_COUNT (INT_MAX & PAGE_CACHE_MASK)
1970
1971 #define FLOCK_VERIFY_READ 1
1972 #define FLOCK_VERIFY_WRITE 2
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f18d in file_inode+12 at include/linux/fs.h:1963
file_inode (f=0xffff88000e61f700) at include/linux/fs.h:1963
1963 return f->f_inode;
>>>

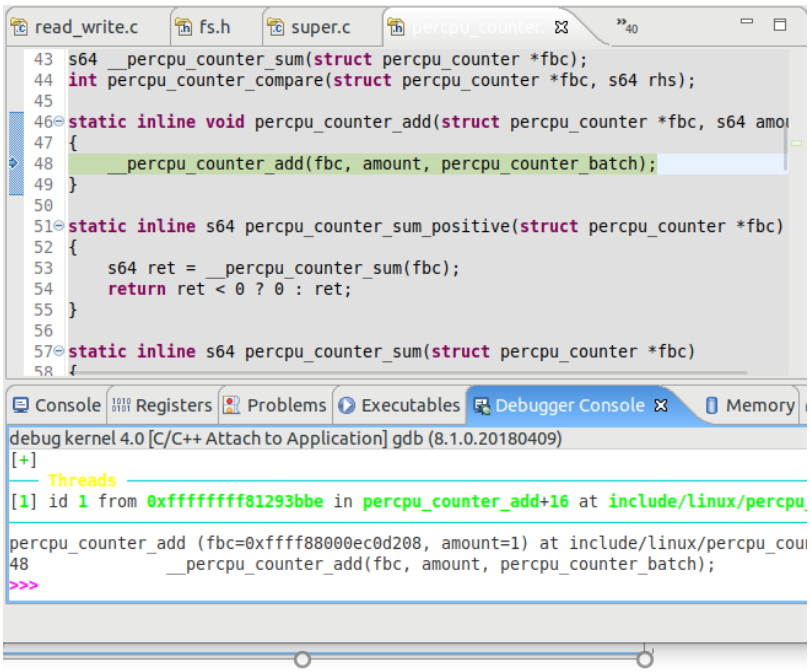

read_write.c
security.c
fsnotify.h
2378
2379 static inline void file_start_write(struct file *file)
2380 {
2381 if (!S_ISREG(file_inode(file)->i_mode))
2382 return;
2383 __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, true);
2384 }
2385
2386 static inline bool file_start_write_trylock(struct file *file)
2387 {
2388 if (!S_ISREG(file_inode(file)->i_mode))
2389 return true;
2390 return __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, fal
2391 }
2392
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f229 in file_start_write+44 at include/linux/fs.h:2383
file_start_write (file=0xffff88000e61f700) at include/linux/fs.h:2383
2383 __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, true);
>>>

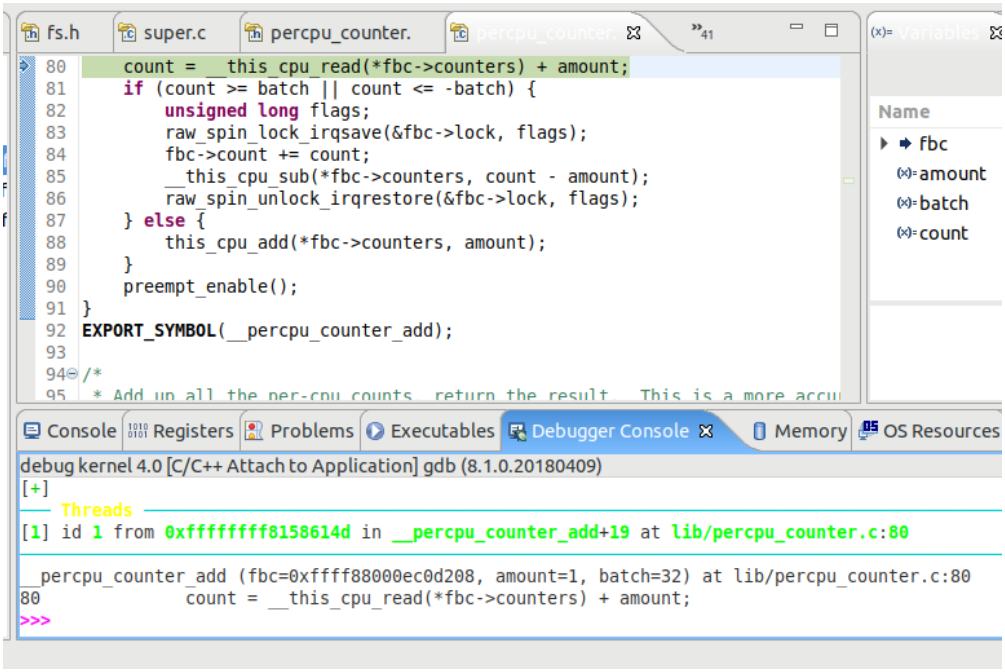

read_write.c
security.c
fsnotify.h
1958 extern void iput(struct inode *);
1959 extern int generic_update_time(struct inode *, struct timespec *, int
1960
1961 static inline struct inode *file_inode(const struct file *f)
1962 {
1963 return f->f_inode;
1964 }
1965
1966 /* /sys/fs */
1967 extern struct kobject *fs_kobj;
1968
1969 #define MAX_RW_COUNT (INT_MAX & PAGE_CACHE_MASK)
1970
1971 #define FLOCK_VERIFY_READ 1
1972 #define FLOCK_VERIFY_WRITE 2
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f18d in file_inode+12 at include/linux/fs.h:1963
file_inode (f=0xffff88000e61f700) at include/linux/fs.h:1963
1963 return f->f_inode;
>>>

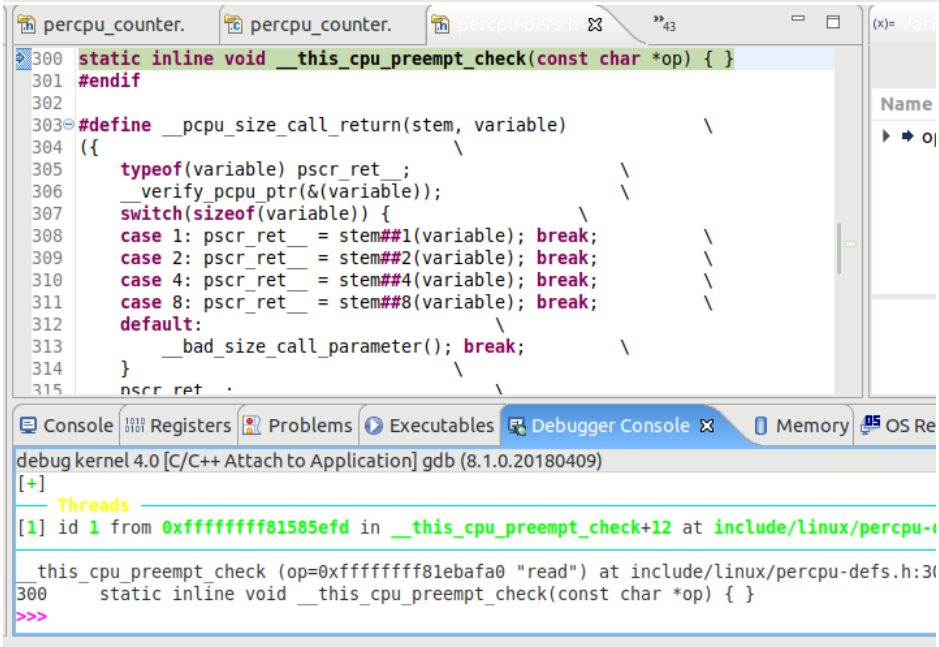

read_write.c
fs.h
security.c
1194 if (unlikely(sb->s_writers.frozen >= level)) {
1195 if (!wait)
1196 return 0;
1197 wait_event(sb->s_writers.wait_unfrozen,
1198 sb->s_writers.frozen < level);
1199 }
1200
1201 #ifdef CONFIG_LOCKDEP
1202 acquire_freeze_lock(sb, level, !wait, _RET_IP_);
1203 #endif
1204 percpu_counter_inc(&sb->s_writers.counter[level-1]);
1205 /*
1206 * Make sure counter is updated before we check for frozen.
1207 * freeze_super() first sets frozen and then checks the counter.
1208 */
1209 smp_mb();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812961c0 in __sb_start_write+20 at fs/super.c:1194
__sb_start_write (sb=0xffff88000ec0d000, level=1, wait=true) at fs/super.c:1194
1194 if (unlikely(sb->s_writers.frozen >= level)) {
>>>

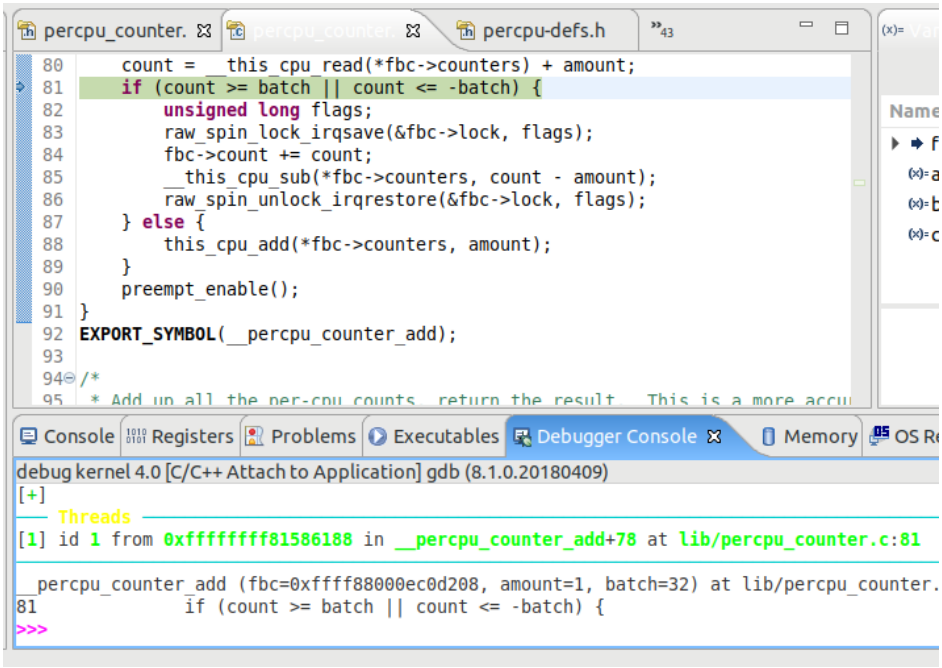

read_write.c
fs.h
security.c
super.c
1194 if (unlikely(sb->s_writers.frozen >= level)) {
1195 if (!wait)
1196 return 0;
1197 wait_event(sb->s_writers.wait_unfrozen,
1198 sb->s_writers.frozen < level);
1199 }
1200
1201 #ifdef CONFIG_LOCKDEP
1202 acquire_freeze_lock(sb, level, !wait, _RET_IP_);
1203 #endif
1204 percpu_counter_inc(&sb->s_writers.counter[level-1]);
1205 /*
1206 * Make sure counter is updated before we check for frozen.
1207 * freeze_super() first sets frozen and then checks the counter.
1208 */
1209 smp_mb();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
}
[+]
Threads
[1] id 1 from 0xffffffff8129627c in __sb_start_write+208 at fs/super.c:1204
1204 percpu_counter_inc(&sb->s_writers.counter[level-1]);
>>>

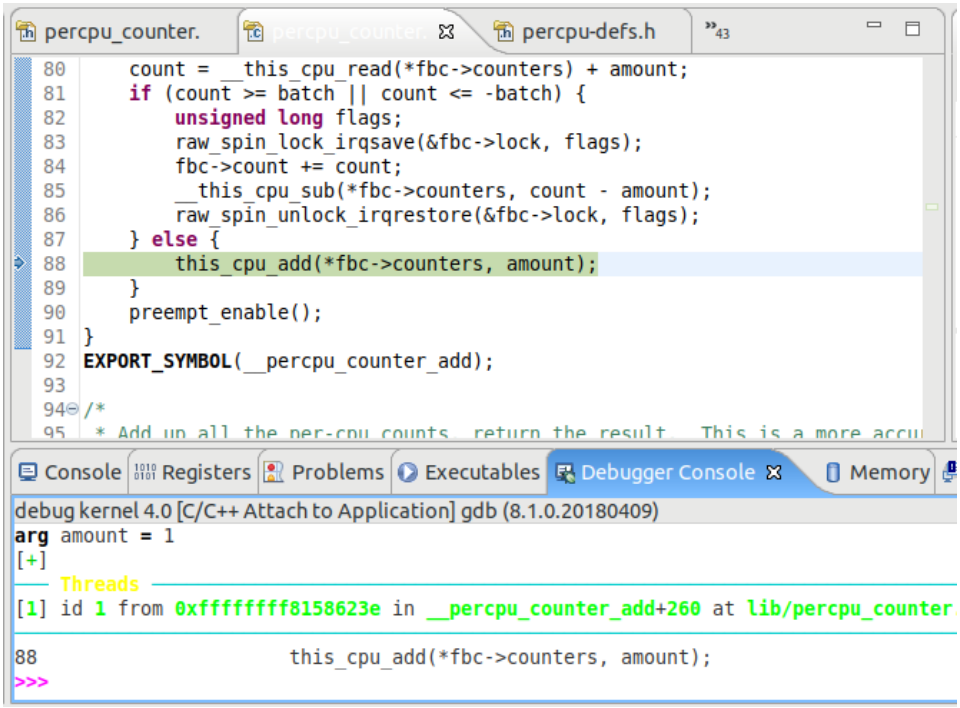

read_write.c
fs.h
super.c
123 fbc->count += amount;
124 preempt_enable();
125 }
126
127 static inline void
128 __percpu_counter_add(struct percpu_counter *fbc, s64 amount, s32 batch)
129 {
130 percpu_counter_add(fbc, amount);
131 }
132
133 static inline s64 percpu_counter_read(struct percpu_counter *fbc)
134 {
135 return fbc->count;
136 }
137
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81293bff in percpu_counter_inc+12 at include/linux/percpu
percpu_counter_inc (fbc=0xffff88000ec0d208) at include/linux/percpu_counter.h:166
166 percpu_counter_add(fbc, 1);
>>>

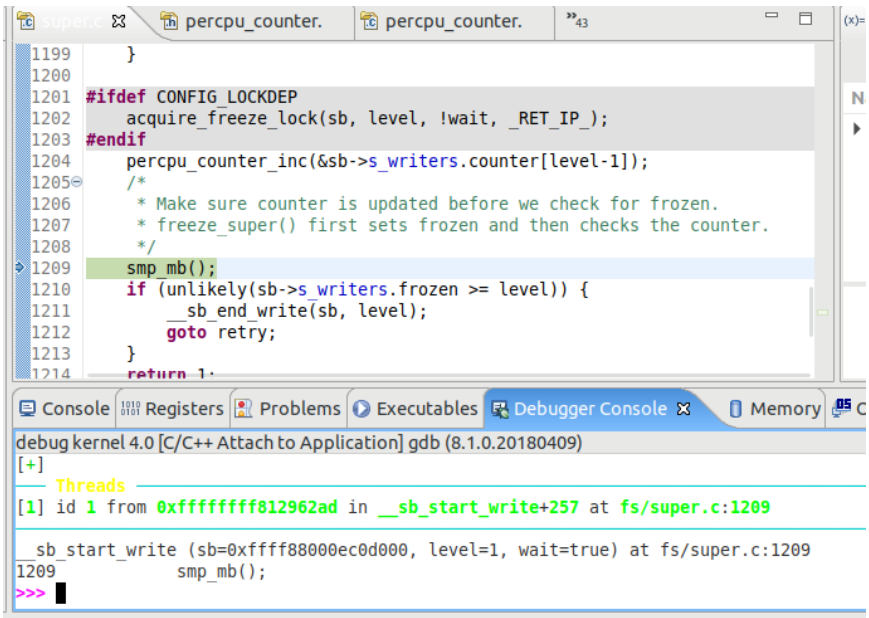

read_write.c
fs.h
super.c
43 s64 __percpu_counter_sum(struct percpu_counter *fbc);
44 int percpu_counter_compare(struct percpu_counter *fbc, s64 rhs);
45
46 static inline void percpu_counter_add(struct percpu_counter *fbc, s64 amo
47 {
48 __percpu_counter_add(fbc, amount, percpu_counter_batch);
49 }
50
51 static inline s64 percpu_counter_sum_positive(struct percpu_counter *fbc)
52 {
53 s64 ret = __percpu_counter_sum(fbc);
54 return ret < 0 ? 0 : ret;
55 }
56
57 static inline s64 percpu_counter_sum(struct percpu_counter *fbc)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81293bbe in percpu_counter_add+16 at include/linux/percpu
percpu_counter_add (fbc=0xffff88000ec0d208, amount=1) at include/linux/percpu_cou
48 __percpu_counter_add(fbc, amount, percpu_counter_batch);
>>>

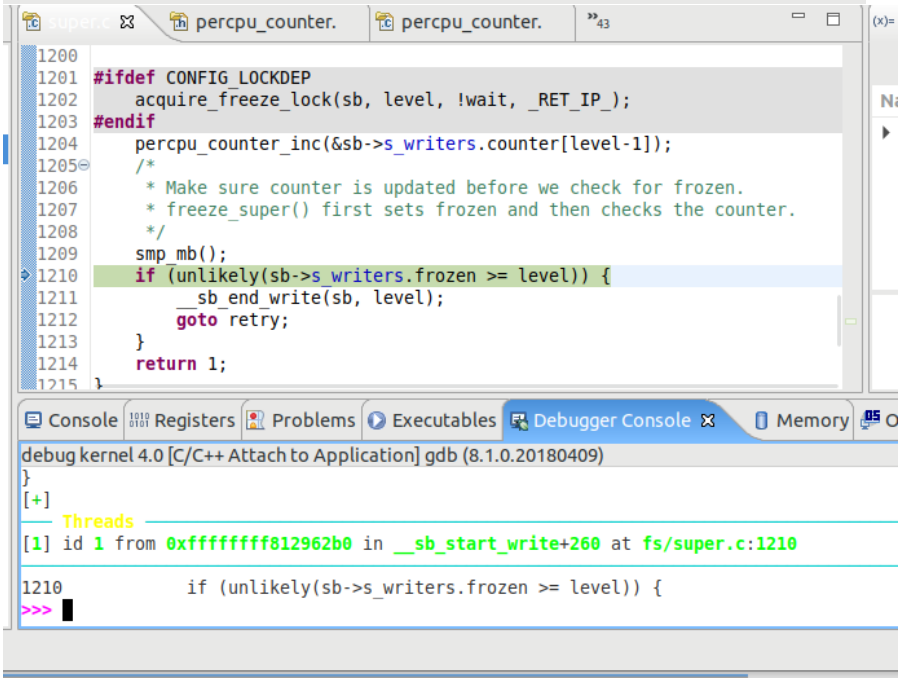

fs.h
super.c
percpu_counter.
80 count = __this_cpu_read(*fbc->counters) + amount;
81 if (count >= batch || count <= -batch) {
82 unsigned long flags;
83 raw_spin_lock_irqsave(&fbc->lock, flags);
84 fbc->count += count;
85 __this_cpu_sub(*fbc->counters, count - amount);
86 raw_spin_unlock_irqrestore(&fbc->lock, flags);
87 } else {
88 this_cpu_add(*fbc->counters, amount);
89 }
90 preempt_enable();
91 }
92 EXPORT_SYMBOL(__percpu_counter_add);
93
94 /*
Name
fbc
amount
batch
count
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8158614d in __percpu_counter_add+19 at lib/percpu_counter.c:80
__percpu_counter_add (fbc=0xffff88000ec0d208, amount=1, batch=32) at lib/percpu_counter.c:80
80 count = __this_cpu_read(*fbc->counters) + amount;
>>>

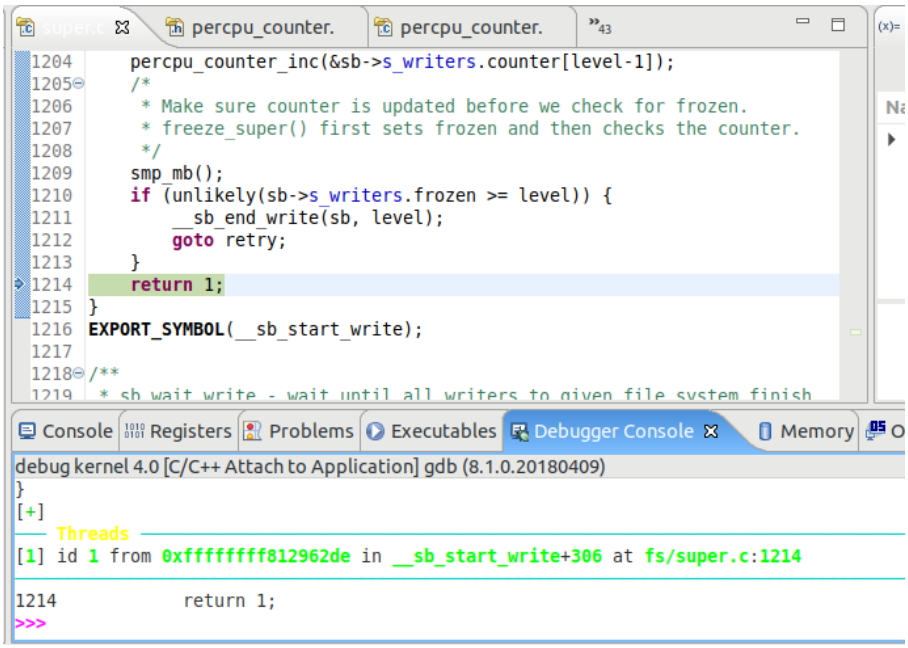

percpu_counter.
percpu_counter.
300 static inline void __this_cpu_preempt_check(const char *op) { }
301 #endif
302
303 #define __pcpu_size_call_return(stem, variable) \
304 ({ \
305 typeof(variable) pscr_ret__; \
306 __verify_pcpu_ptr(&(variable)); \
307 switch(sizeof(variable)) { \
308 case 1: pscr_ret__ = stem##1(variable); break; \
309 case 2: pscr_ret__ = stem##2(variable); break; \
310 case 4: pscr_ret__ = stem##4(variable); break; \
311 case 8: pscr_ret__ = stem##8(variable); break; \
312 default: \
313 __bad_size_call_parameter(); break; \
314 } \
Name
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81585efd in __this_cpu_preempt_check+12 at include/linux/percpu-
__this_cpu_preempt_check (op=0xffffffff81ebafa0 "read") at include/linux/percpu-defs.h:3
300 static inline void __this_cpu_preempt_check(const char *op) { }
>>>

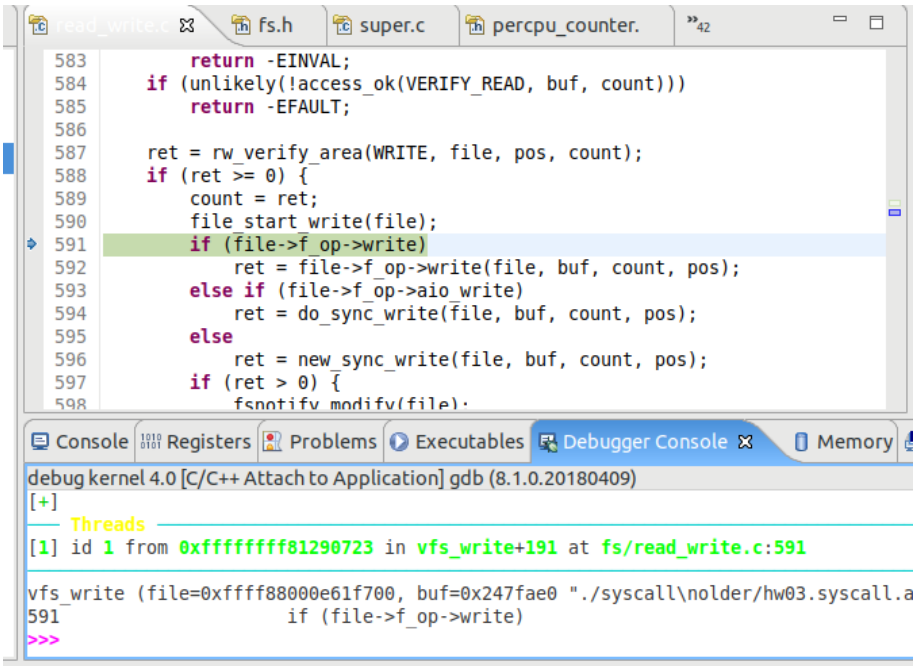

percpu_counter.
percpu-defs.h
80 count = __this_cpu_read(*fbc->counters) + amount;
81 if (count >= batch || count <= -batch) {
82 unsigned long flags;
83 raw_spin_lock_irqsave(&fbc->lock, flags);
84 fbc->count += count;
85 __this_cpu_sub(*fbc->counters, count - amount);
86 raw_spin_unlock_irqrestore(&fbc->lock, flags);
87 } else {
88 this_cpu_add(*fbc->counters, amount);
89 }
90 preempt_enable();
91 }
92 EXPORT_SYMBOL(__percpu_counter_add);
93
94 /*
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81586188 in __percpu_counter_add+78 at lib/percpu_counter.c:81
__percpu_counter_add (fbc=0xffff88000ec0d208, amount=1, batch=32) at lib/percpu_counter.
81 if (count >= batch || count <= -batch) {
>>>

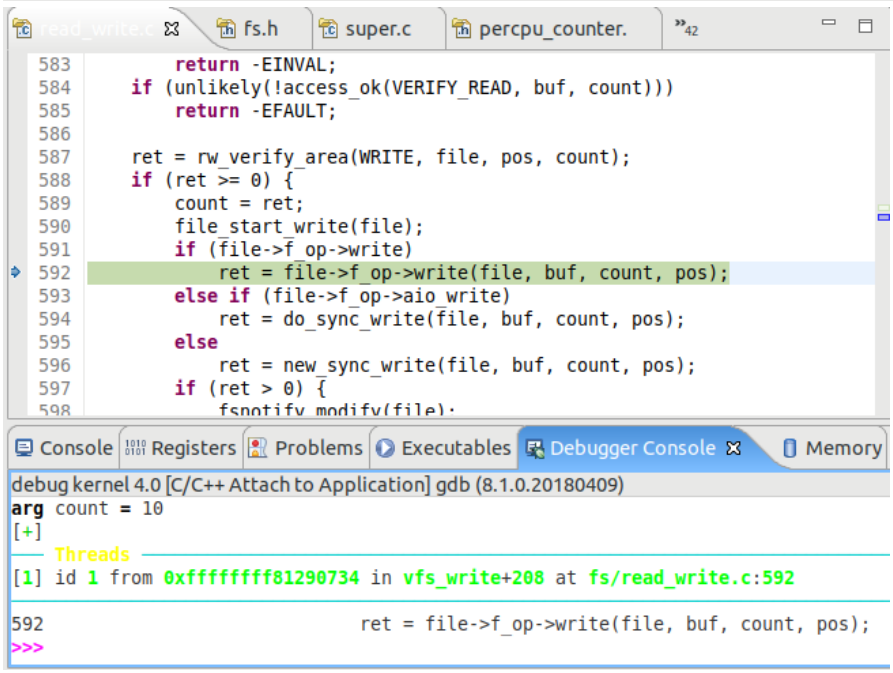

percpu_counter.
percpu-defs.h
80 count = __this_cpu_read(*fbc->counters) + amount;
81 if (count >= batch || count <= -batch) {
82 unsigned long flags;
83 raw_spin_lock_irqsave(&fbc->lock, flags);
84 fbc->count += count;
85 __this_cpu_sub(*fbc->counters, count - amount);
86 raw_spin_unlock_irqrestore(&fbc->lock, flags);
87 } else {
88 this_cpu_add(*fbc->counters, amount);
89 }
90 preempt_enable();
91 }
92 EXPORT_SYMBOL(__percpu_counter_add);
93
94 /*
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg amount = 1
[+]
Threads
[1] id 1 from 0xffffffff8158623e in __percpu_counter_add+260 at lib/percpu_counter
88 this_cpu_add(*fbc->counters, amount);
>>>

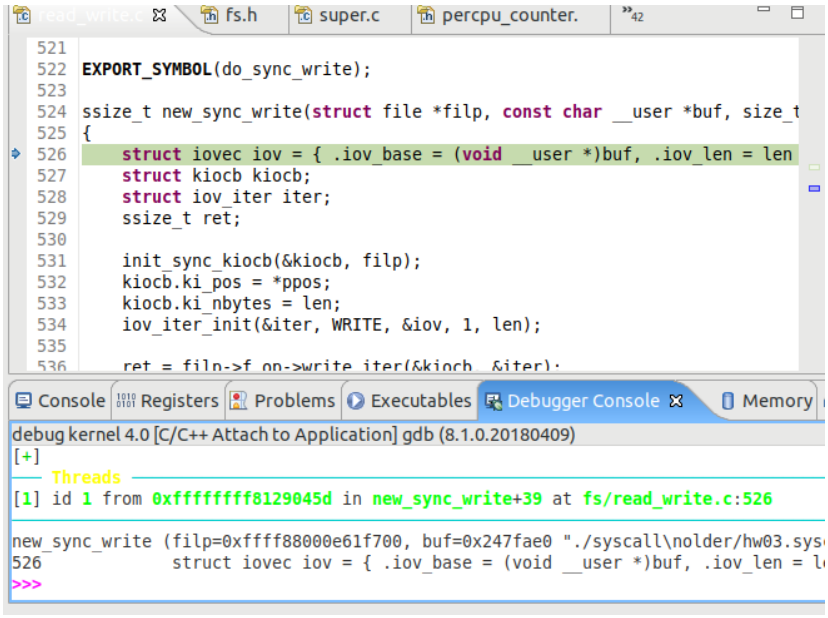

percpu_counter.
percpu_counter.
1199 }
1200
1201 #ifdef CONFIG_LOCKDEP
1202 acquire_freeze_lock(sb, level, !wait, _RET_IP_);
1203 #endif
1204 percpu_counter_inc(&sb->s_writers.counter[level-1]);
1205 /*
1206 * Make sure counter is updated before we check for frozen.
1207 * freeze_super() first sets frozen and then checks the counter.
1208 */
1209 smp_mb();
1210 if (unlikely(sb->s_writers.frozen >= level)) {
1211 __sb_end_write(sb, level);
1212 goto retry;
1213 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812962ad in __sb_start_write+257 at fs/super.c:1209
__sb_start_write (sb=0xffff88000ec0d000, level=1, wait=true) at fs/super.c:1209
1209 smp_mb();
>>>

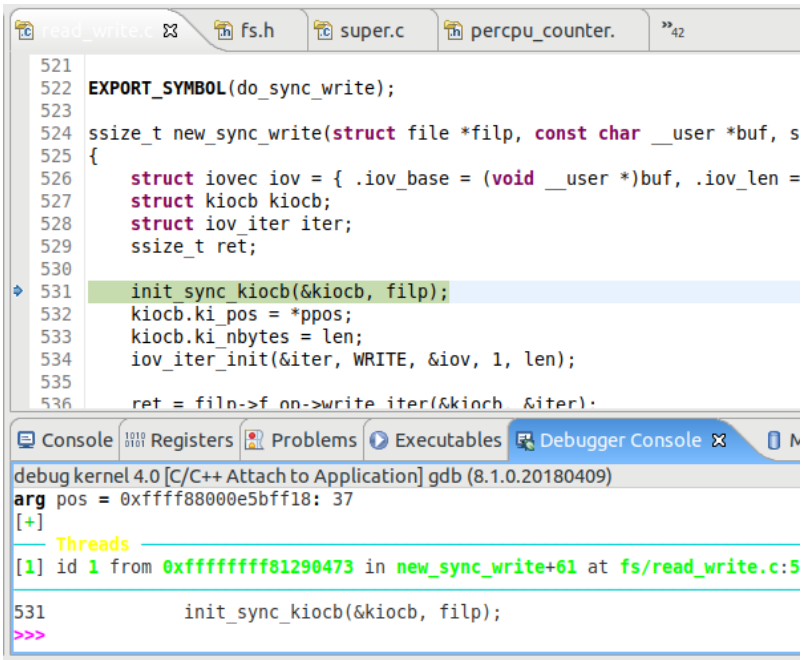

percpu_counter.
percpu_counter.
1200
1201 #ifdef CONFIG_LOCKDEP
1202 acquire_freeze_lock(sb, level, !wait, _RET_IP_);
1203 #endif
1204 percpu_counter_inc(&sb->s_writers.counter[level-1]);
1205 /*
1206 * Make sure counter is updated before we check for frozen.
1207 * freeze_super() first sets frozen and then checks the counter.
1208 */
1209 smp_mb();
1210 if (unlikely(sb->s_writers.frozen >= level)) {
1211 __sb_end_write(sb, level);
1212 goto retry;
1213 }
1214 return 1;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
}
[+]
Threads
[1] id 1 from 0xffffffff812962b0 in __sb_start_write+260 at fs/super.c:1210
1210 if (unlikely(sb->s_writers.frozen >= level)) {
>>>

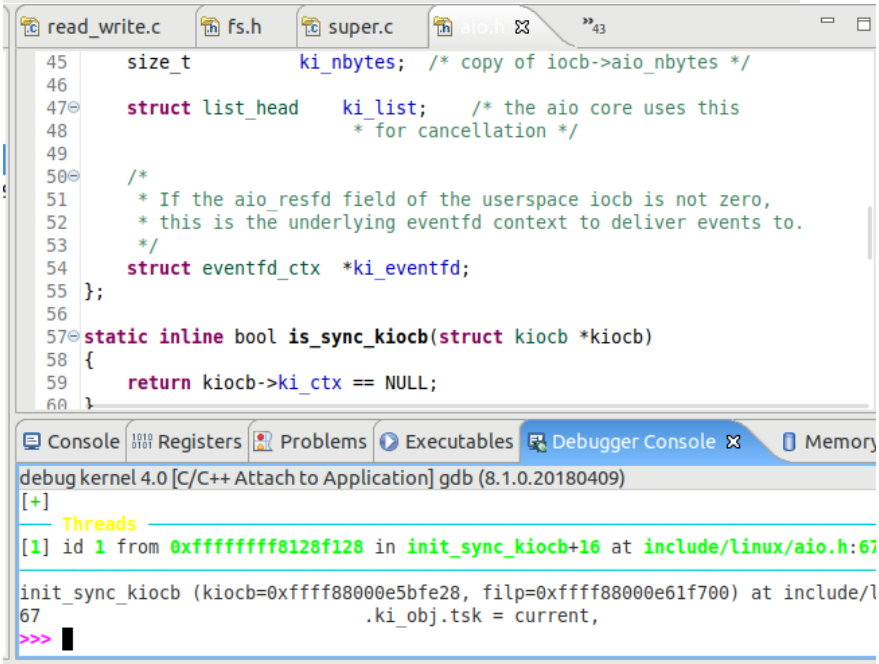

percpu_counter.
percpu_counter.
1204 percpu_counter_inc(&sb->s_writers.counter[level-1]);
1205 /*
1206 * Make sure counter is updated before we check for frozen.
1207 * freeze_super() first sets frozen and then checks the counter.
1208 */
1209 smp_mb();
1210 if (unlikely(sb->s_writers.frozen >= level)) {
1211 __sb_end_write(sb, level);
1212 goto retry;
1213 }
1214 return 1;
1215 }
1216 EXPORT_SYMBOL(__sb_start_write);
1217
1218 /**
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
}
[+]
Threads
[1] id 1 from 0xffffffff812962de in __sb_start_write+306 at fs/super.c:1214
1214 return 1;
>>>

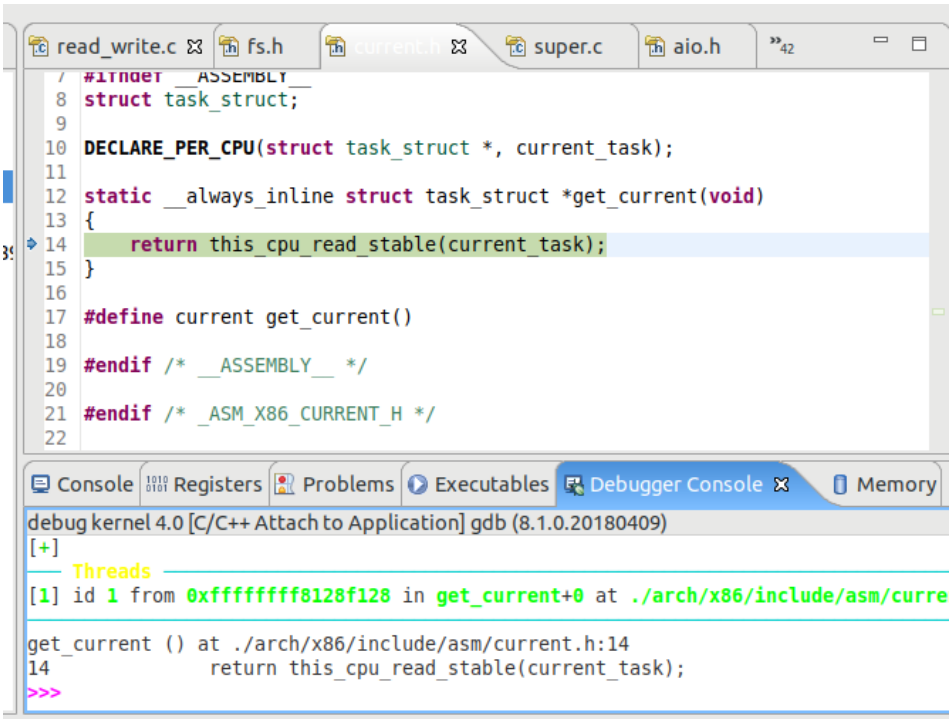

fs.h
super.c
percpu_counter.
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
591 if (file->f_op->write)
592 ret = file->f_op->write(file, buf, count, pos);
593 else if (file->f_op->aio_write)
594 ret = do_sync_write(file, buf, count, pos);
595 else
596 ret = new_sync_write(file, buf, count, pos);
597 if (ret > 0) {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290723 in vfs_write+191 at fs/read_write.c:591
vfs_write (file=0xffff88000e61f700, buf=0x247fae0 "./syscall\nolder/hw03.syscall.a
591 if (file->f_op->write)
>>>

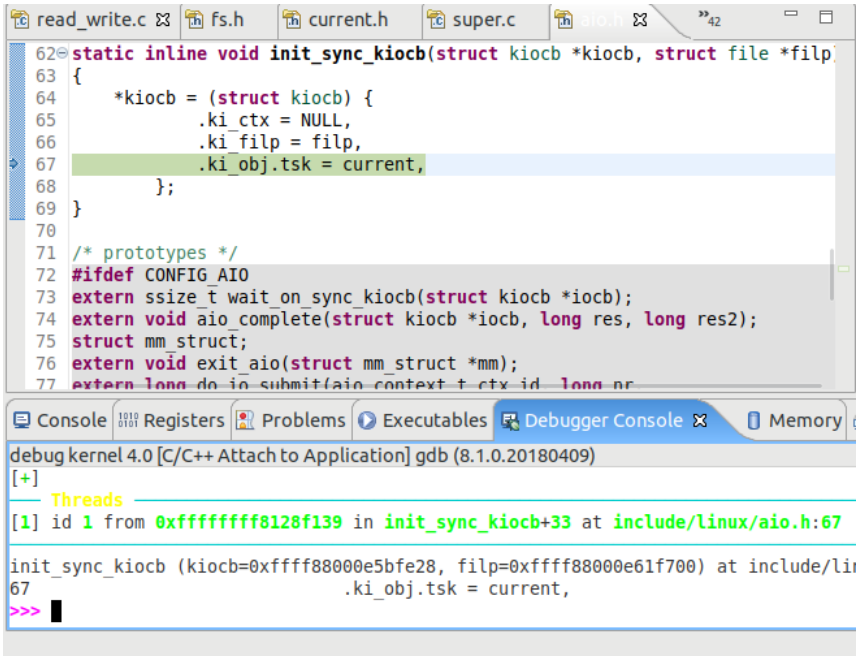

fs.h
super.c
percpu_counter.
583 return -EINVAL;
584 if (unlikely(!access_ok(VERIFY_READ, buf, count)))
585 return -EFAULT;
586
587 ret = rw_verify_area(WRITE, file, pos, count);
588 if (ret >= 0) {
589 count = ret;
590 file_start_write(file);
591 if (file->f_op->write)
592 ret = file->f_op->write(file, buf, count, pos);
593 else if (file->f_op->aio_write)
594 ret = do_sync_write(file, buf, count, pos);
595 else
596 ret = new_sync_write(file, buf, count, pos);
597 if (ret > 0) {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg count = 10
[+]
Threads
[1] id 1 from 0xffffffff81290734 in vfs_write+208 at fs/read_write.c:592
592 ret = file->f_op->write(file, buf, count, pos);
>>>

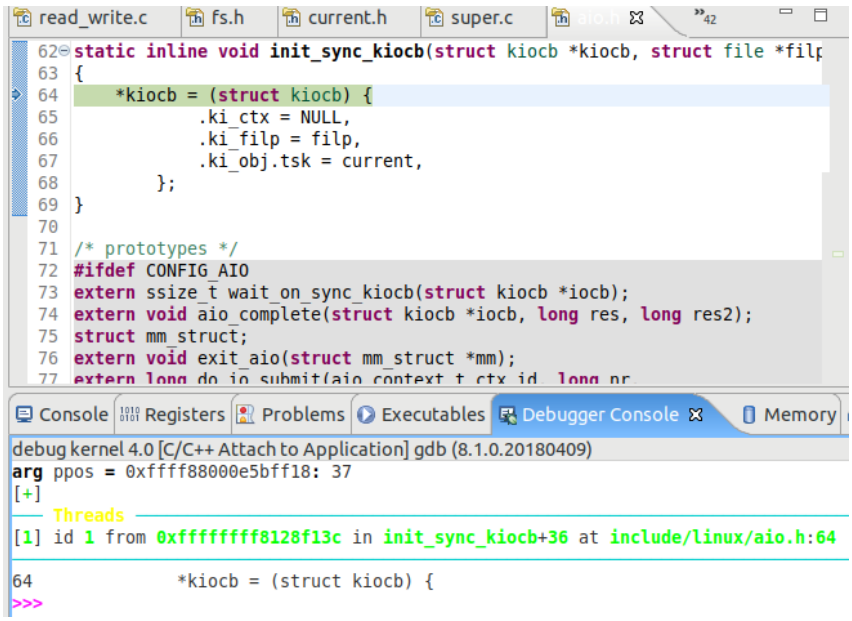

fs.h
super.c
percpu_counter.
EXPORT_SYMBOL(do_sync_write);
ssize_t new_sync_write(struct file *filp, const char __user *buf, size_t
{
struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len
struct kiocb kiocb;
struct iov_iter iter;
ssize_t ret;
init_sync_kiocb(&kiocb, filp);
kiocb.ki_pos = *ppos;
kiocb.ki_nbytes = len;
iov_iter_init(&iter, WRITE, &iov, 1, len);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8129045d in new_sync_write+39 at fs/read_write.c:526
new_sync_write (filp=0xffff88000e61f700, buf=0x247fae0 "./syscall\nolder/hw03.sys
526 struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = l
>>>

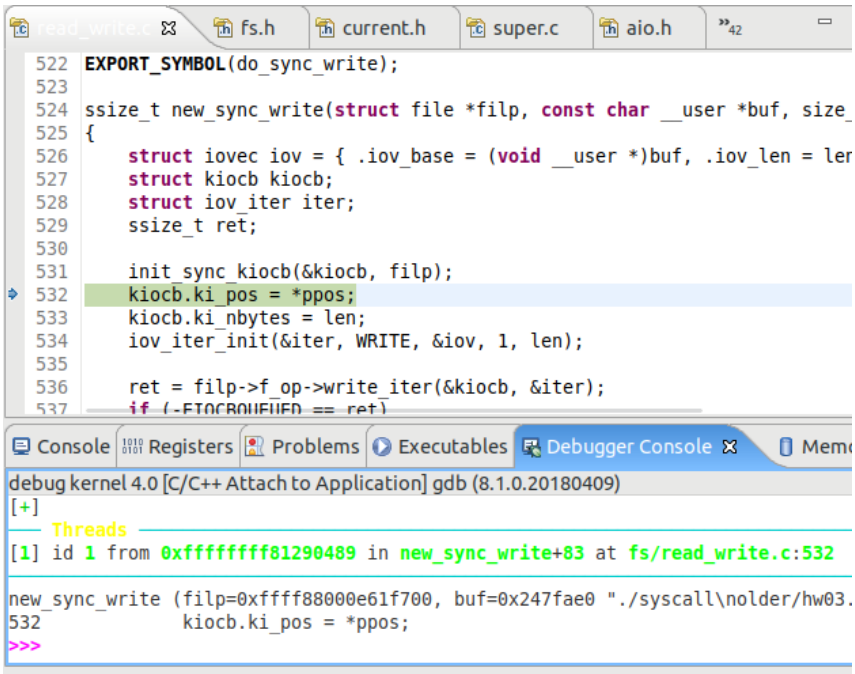

fs.h
super.c
percpu_counter.
EXPORT_SYMBOL(do_sync_write);
ssize_t new_sync_write(struct file *filp, const char __user *buf, s
{
struct iovec iov = { .iov_base = (void __user *)buf, .iov_len =
struct kiocb kiocb;
struct iov_iter iter;
ssize_t ret;
init_sync_kiocb(&kiocb, filp);
kiocb.ki_pos = *ppos;
kiocb.ki_nbytes = len;
iov_iter_init(&iter, WRITE, &iov, 1, len);
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 37
[+]
Threads
[1] id 1 from 0xffffffff81290473 in new_sync_write+61 at fs/read_write.c:5
531 init_sync_kiocb(&kiocb, filp);
>>>

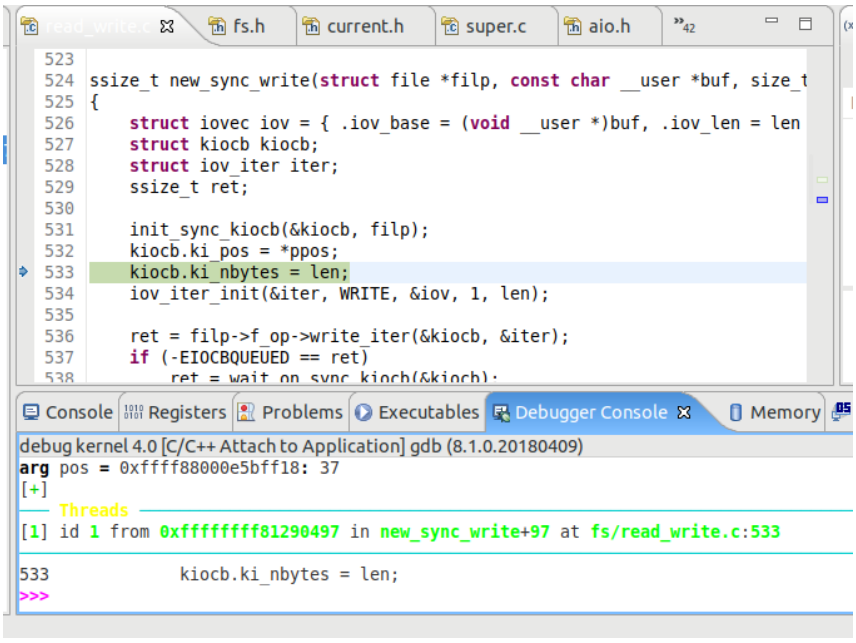

read_write.c
fs.h
super.c
size_t ki_nbytes; /* copy of iocb->aio_nbytes */
struct list_head ki_list; /* the aio core uses this
* for cancellation */
/*
* If the aio_resfd field of the userspace iocb is not zero,
* this is the underlying eventfd context to deliver events to.
*/
struct eventfd_ctx *ki_eventfd;
};
static inline bool is_sync_kiocb(struct kiocb *kiocb)
{
return kiocb->ki_ctx == NULL;
}
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f128 in init_sync_kiocb+16 at include/linux/aio.h:67
init_sync_kiocb (kiocb=0xffff88000e5bfe28, filp=0xffff88000e61f700) at include/l
67 .ki_obj.tsk = current,
>>>

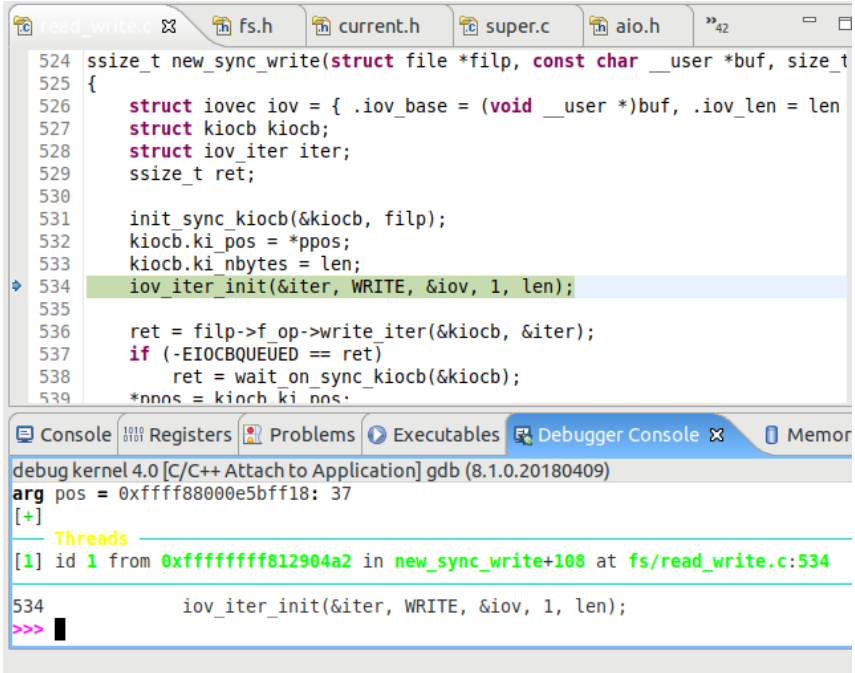

read_write.c
fs.h
current.h
super.c
aio.h
#ifndef __ASSEMBLY__
struct task_struct;
DECLARE_PER_CPU(struct task_struct *, current_task);
static __always_inline struct task_struct *get_current(void)
{
return this_cpu_read_stable(current_task);
}
#define current get_current()
#endif /* __ASSEMBLY__ */
#endif /* _ASM_X86_CURRENT_H */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f128 in get_current+0 at ./arch/x86/include/asm/curre
get_current () at ./arch/x86/include/asm/current.h:14
14 return this_cpu_read_stable(current_task);
>>>

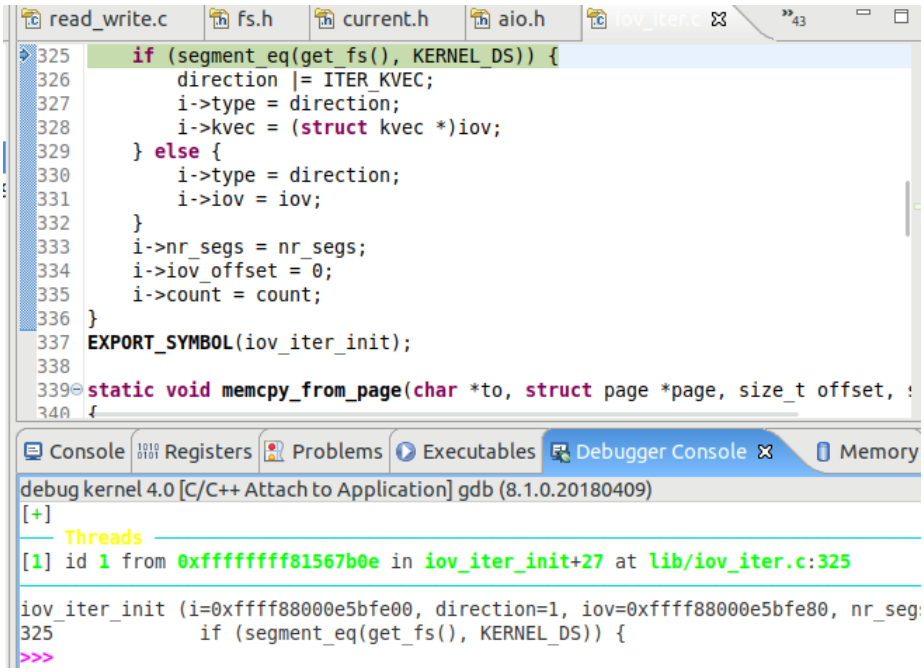

read_write.c
fs.h
current.h
super.c
static inline void init_sync_kiocb(struct kiocb *kiocb, struct file *filp
{
*kiocb = (struct kiocb) {
.ki_ctx = NULL,
.ki_filp = filp,
.ki_obj.tsk = current,
};
}
/* prototypes */
#ifdef CONFIG_AIO
extern ssize_t wait_on_sync_kiocb(struct kiocb *iocb);
extern void aio_complete(struct kiocb *iocb, long res, long res2);
struct mm_struct;
extern void exit_aio(struct mm_struct *mm);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8128f139 in init_sync_kiocb+33 at include/linux/aio.h:67
init_sync_kiocb (kiocb=0xffff88000e5bfe28, filp=0xffff88000e61f700) at include/li
67 .ki_obj.tsk = current,
>>>

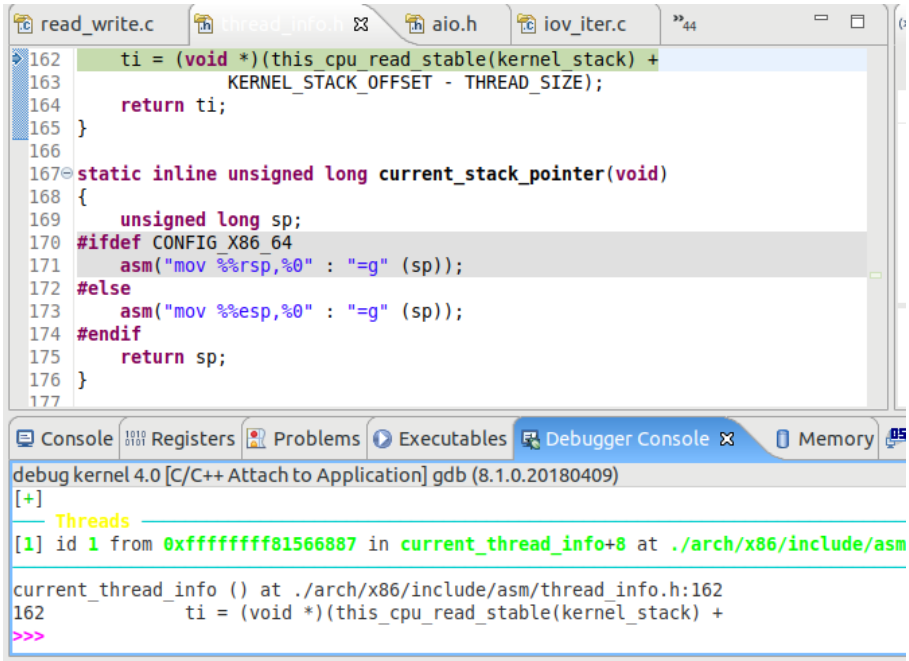

read_write.c
fs.h
current.h
super.c
static inline void init_sync_kiocb(struct kiocb *kiocb, struct file *filp
{
*kiocb = (struct kiocb) {
.ki_ctx = NULL,
.ki_filp = filp,
.ki_obj.tsk = current,
};
}
/* prototypes */
#ifdef CONFIG_AIO
extern ssize_t wait_on_sync_kiocb(struct kiocb *iocb);
extern void aio_complete(struct kiocb *iocb, long res, long res2);
struct mm_struct;
extern void exit_aio(struct mm_struct *mm);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg ppos = 0xffff88000e5bff18: 37
[+]
Threads
[1] id 1 from 0xffffffff8128f13c in init_sync_kiocb+36 at include/linux/aio.h:64
64 *kiocb = (struct kiocb) {
>>>

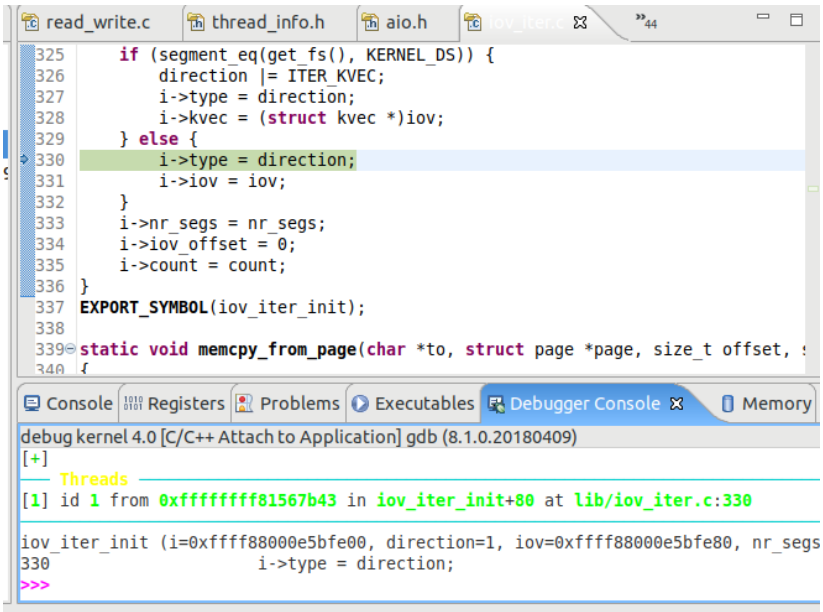

fs.h
current.h
super.c
aio.h
522 EXPORT_SYMBOL(do_sync_write);
523
524 ssize_t new_sync_write(struct file *filp, const char __user *buf, size_
525 {
526 struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = ler
527 struct kiocb kiocb;
528 struct iov_iter iter;
529 ssize_t ret;
530
531 init_sync_kiocb(&kiocb, filp);
532 kiocb.ki_pos = *ppos;
533 kiocb.ki_nbytes = len;
534 iov_iter_init(&iter, WRITE, &iov, 1, len);
535
536 ret = filp->f_op->write_iter(&kiocb, &iter);
Console
Registers
Problems
Executables
Debugger Console
Memo
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290489 in new_sync_write+83 at fs/read_write.c:532
new_sync_write (filp=0xffff88000e61f700, buf=0x247fae0 "./syscall\nolder/hw03.
532 kiocb.ki_pos = *ppos;
>>>

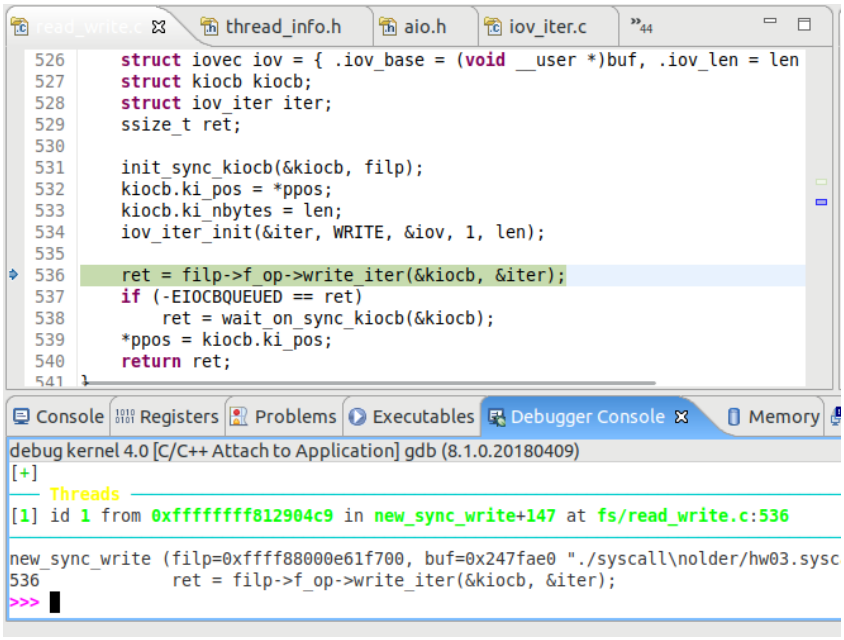

fs.h
current.h
super.c
aio.h
524 ssize_t new_sync_write(struct file *filp, const char __user *buf, size_t
525 {
526 struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len
527 struct kiocb kiocb;
528 struct iov_iter iter;
529 ssize_t ret;
530
531 init_sync_kiocb(&kiocb, filp);
532 kiocb.ki_pos = *ppos;
533 kiocb.ki_nbytes = len;
534 iov_iter_init(&iter, WRITE, &iov, 1, len);
535
536 ret = filp->f_op->write_iter(&kiocb, &iter);
537 if (-EIOCBQUEUED == ret)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 37
[+]
Threads
[1] id 1 from 0xffffffff81290497 in new_sync_write+97 at fs/read_write.c:533
533 kiocb.ki_nbytes = len;
>>>

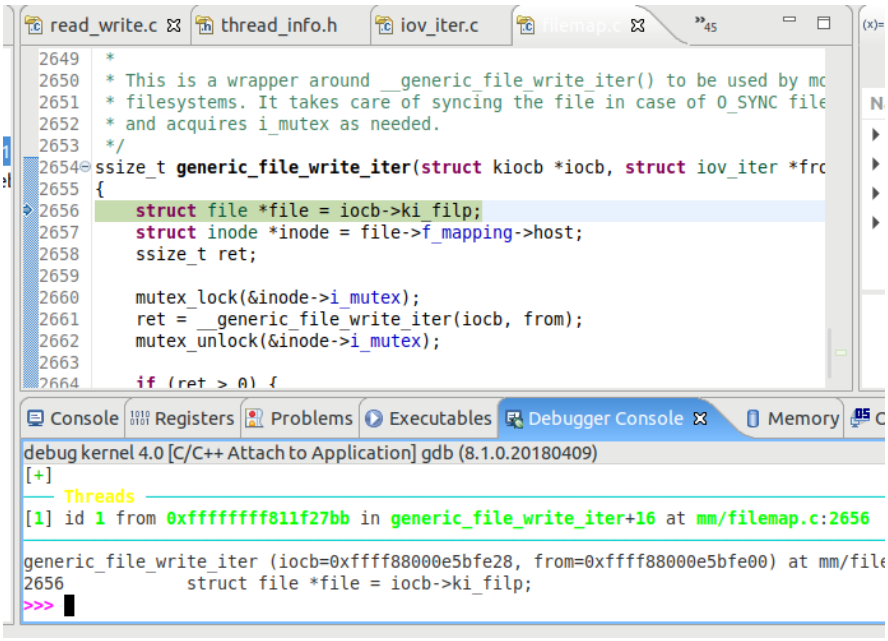

fs.h
current.h
super.c
aio.h
524 ssize_t new_sync_write(struct file *filp, const char __user *buf, size_t
525 {
526 struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len
527 struct kiocb kiocb;
528 struct iov_iter iter;
529 ssize_t ret;
530
531 init_sync_kiocb(&kiocb, filp);
532 kiocb.ki_pos = *ppos;
533 kiocb.ki_nbytes = len;
534 iov_iter_init(&iter, WRITE, &iov, 1, len);
535
536 ret = filp->f_op->write_iter(&kiocb, &iter);
537 if (-EIOCBQUEUED == ret)
538 ret = wait_on_sync_kiocb(&kiocb);
Console
Registers
Problems
Executables
Debugger Console
Memor
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg pos = 0xffff88000e5bff18: 37
[+]
Threads
[1] id 1 from 0xffffffff812904a2 in new_sync_write+108 at fs/read_write.c:534
534 iov_iter_init(&iter, WRITE, &iov, 1, len);
>>>

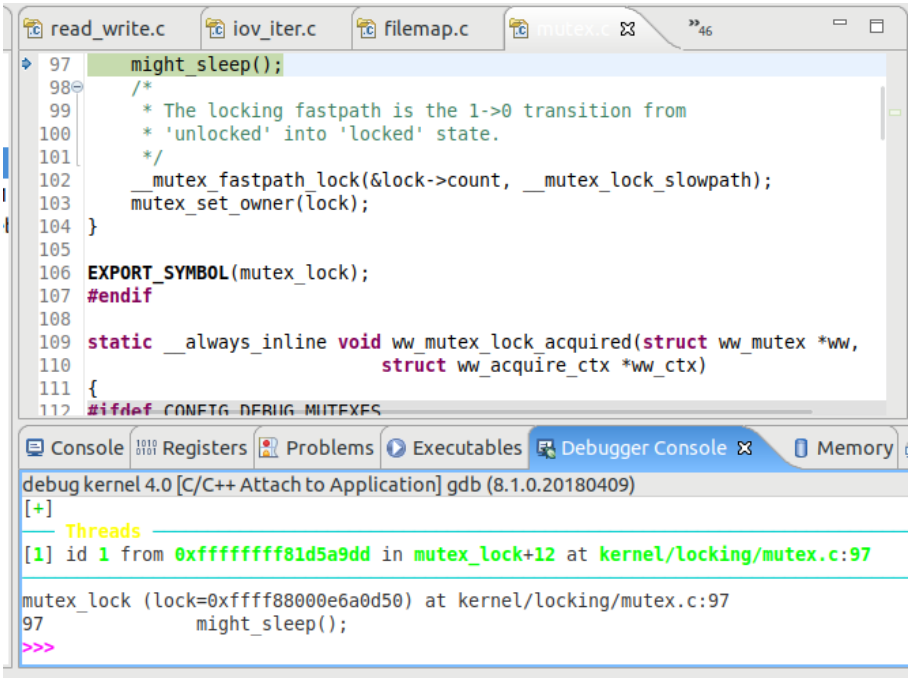

read_write.c
fs.h
current.h
aio.h
iov_iter.c
325 if (segment_eq(get_fs(), KERNEL_DS)) {
326 direction |= ITER_KVEC;
327 i->type = direction;
328 i->kvec = (struct kvec *)iov;
329 } else {
330 i->type = direction;
331 i->iov = iov;
332 }
333 i->nr_segs = nr_segs;
334 i->iov_offset = 0;
335 i->count = count;
336 }
337 EXPORT_SYMBOL(iov_iter_init);
338
339 static void memcpy_from_page(char *to, struct page *page, size_t offset,
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81567b0e in iov_iter_init+27 at lib/iov_iter.c:325
iov_iter_init (i=0xffff88000e5bfe00, direction=1, iov=0xffff88000e5bfe80, nr_seg
325 if (segment_eq(get_fs(), KERNEL_DS)) {
>>>

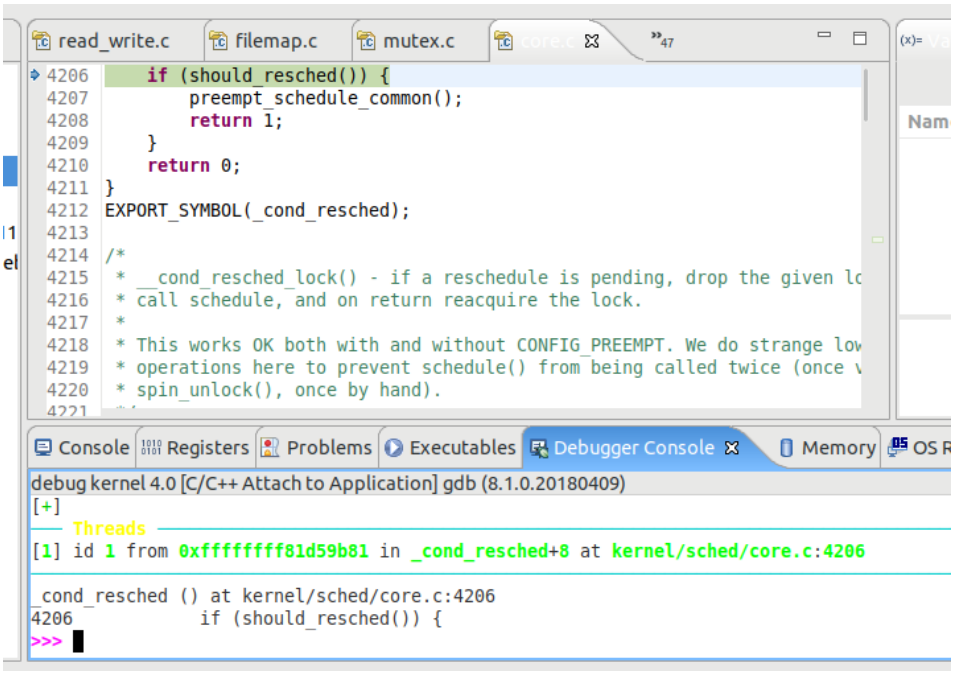

read_write.c
aio.h
iov_iter.c
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
163 KERNEL_STACK_OFFSET - THREAD_SIZE);
164 return ti;
165 }
166
167 static inline unsigned long current_stack_pointer(void)
168 {
169 unsigned long sp;
170 #ifdef CONFIG_X86_64
171 asm("mov %%rsp,%0" : "=g" (sp));
172 #else
173 asm("mov %%esp,%0" : "=g" (sp));
174 #endif
175 return sp;
176 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81566887 in current_thread_info+8 at ./arch/x86/include/asm
current_thread_info () at ./arch/x86/include/asm/thread_info.h:162
162 ti = (void *)(this_cpu_read_stable(kernel_stack) +
>>>

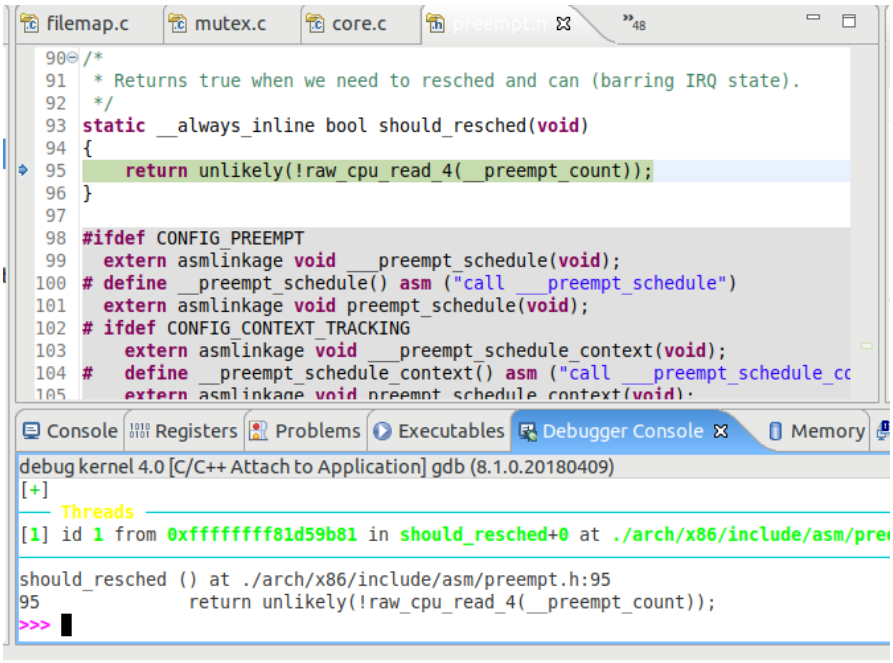

read_write.c
thread_info.h
aio.h
325 if (segment_eq(get_fs(), KERNEL_DS)) {
326 direction |= ITER_KVEC;
327 i->type = direction;
328 i->kvec = (struct kvec *)iov;
329 } else {
330 i->type = direction;
331 i->iov = iov;
332 }
333 i->nr_segs = nr_segs;
334 i->iov_offset = 0;
335 i->count = count;
336 }
337 EXPORT_SYMBOL(iov_iter_init);
338
339 static void memcpy_from_page(char *to, struct page *page, size_t offset,
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81567b43 in iov_iter_init+80 at lib/iov_iter.c:330
iov_iter_init (i=0xffff88000e5bfe00, direction=1, iov=0xffff88000e5bfe80, nr_segs
330 i->type = direction;
>>>

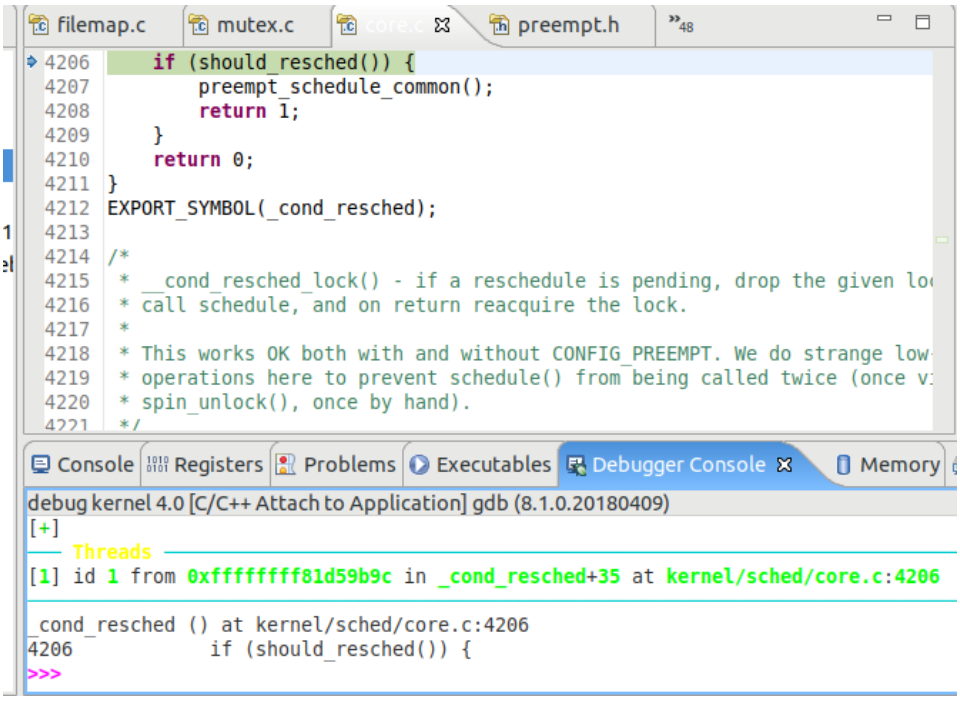

thread_info.h
aio.h
iov_iter.c
526 struct iovec iov = { .iov_base = (void __user *)buf, .iov_len = len
527 struct kiocb kiocb;
528 struct iov_iter iter;
529 ssize_t ret;
530
531 init_sync_kiocb(&kiocb, filp);
532 kiocb.ki_pos = *ppos;
533 kiocb.ki_nbytes = len;
534 iov_iter_init(&iter, WRITE, &iov, 1, len);
535
536 ret = filp->f_op->write_iter(&kiocb, &iter);
537 if (-EIOCBQUEUED == ret)
538 ret = wait_on_sync_kiocb(&kiocb);
539 *ppos = kiocb.ki_pos;
540 return ret;
541 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff812904c9 in new_sync_write+147 at fs/read_write.c:536
new_sync_write (filp=0xffff88000e61f700, buf=0x247fae0 "./syscall\nolder/hw03.sysc
536 ret = filp->f_op->write_iter(&kiocb, &iter);
>>>

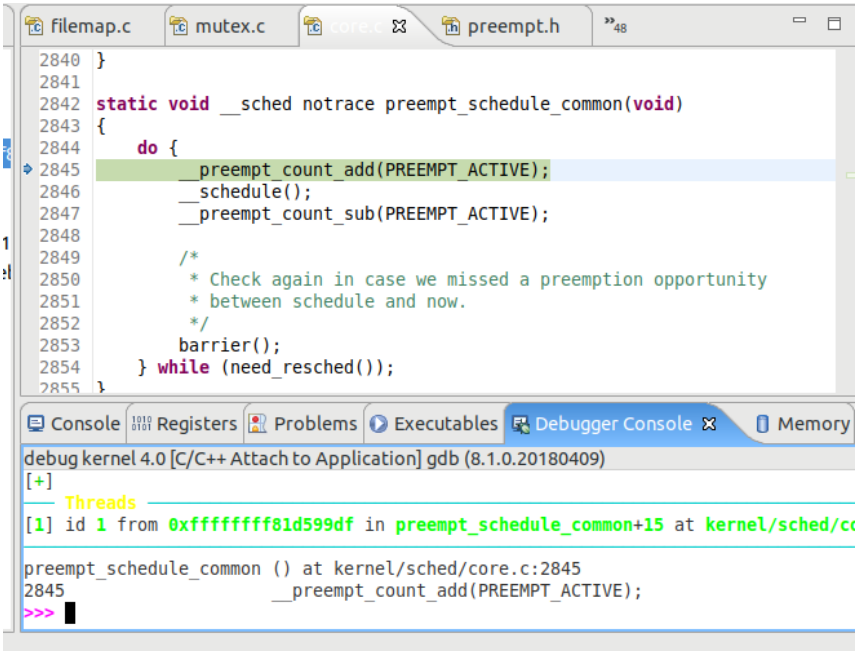

read_write.c
thread_info.h
iov_iter.c
2649 *
2650 * This is a wrapper around __generic_file_write_iter() to be used by mo
2651 * filesystems. It takes care of syncing the file in case of O_SYNC file
2652 * and acquires i_mutex as needed.
2653 */
2654 ssize_t generic_file_write_iter(struct kiocb *iocb, struct iov_iter *fro
2655 {
2656 struct file *file = iocb->ki_filp;
2657 struct inode *inode = file->f_mapping->host;
2658 ssize_t ret;
2659
2660 mutex_lock(&inode->i_mutex);
2661 ret = __generic_file_write_iter(iocb, from);
2662 mutex_unlock(&inode->i_mutex);
2663
2664 if (ret > 0) {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811f27bb in generic_file_write_iter+16 at mm/filemap.c:2656
generic_file_write_iter (iocb=0xffff88000e5bfe28, from=0xffff88000e5bfe00) at mm/file
2656 struct file *file = iocb->ki_filp;
>>>

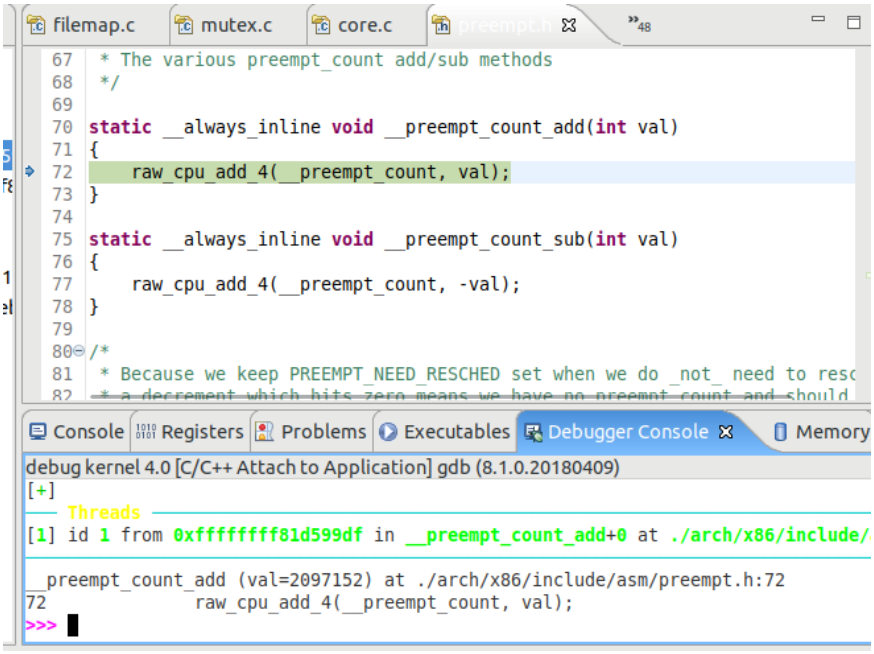

read_write.c
iov_iter.c
filemap.c
97 might_sleep();
98 /*
99 * The locking fastpath is the 1->0 transition from
100 * 'unlocked' into 'locked' state.
101 */
102 __mutex_fastpath_lock(&lock->count, __mutex_lock_slowpath);
103 mutex_set_owner(lock);
104 }
105
106 EXPORT_SYMBOL(mutex_lock);
107 #endif
108
109 static __always_inline void ww_mutex_lock_acquired(struct ww_mutex *ww,
110 struct ww_acquire_ctx *ww_ctx)
111 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d5a9dd in mutex_lock+12 at kernel/locking/mutex.c:97
mutex_lock (lock=0xffff88000e6a0d50) at kernel/locking/mutex.c:97
97 might_sleep();
>>>

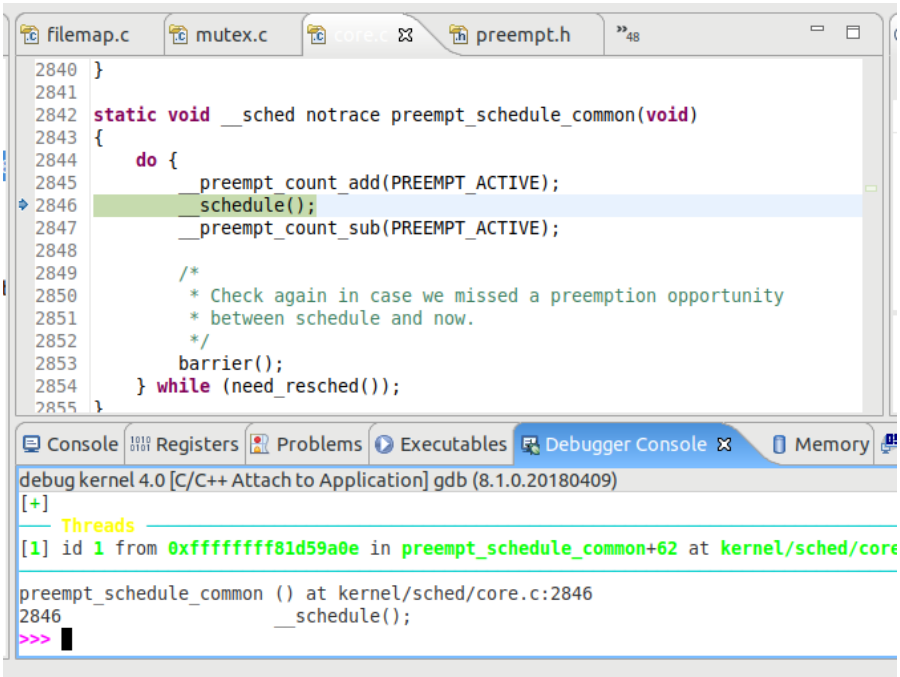

read_write.c
filemap.c
mutex.c
4206 if (should_resched()) {
4207 preempt_schedule_common();
4208 return 1;
4209 }
4210 return 0;
4211 }
4212 EXPORT_SYMBOL(_cond_resched);
4213
4214 /*
4215 * __cond_resched_lock() - if a reschedule is pending, drop the given lo
4216 * call schedule, and on return reacquire the lock.
4217 *
4218 * This works OK both with and without CONFIG_PREEMPT. We do strange low
4219 * operations here to prevent schedule() from being called twice (once v
4220 * spin_unlock(), once by hand).
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d59b81 in _cond_resched+8 at kernel/sched/core.c:4206
_cond_resched () at kernel/sched/core.c:4206
4206 if (should_resched()) {
>>>

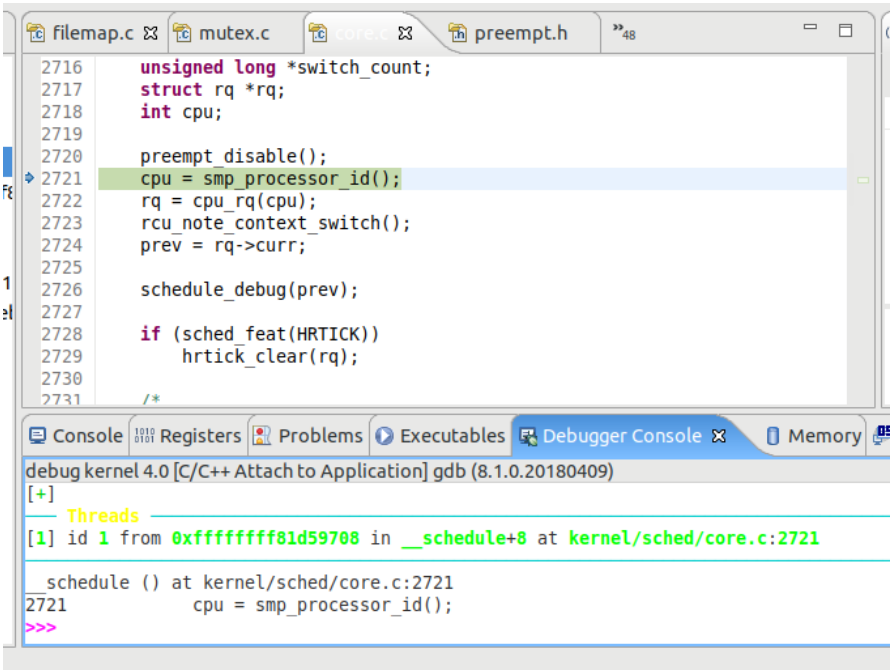

filemap.c
mutex.c
core.c
preempt.h
90 /*
91 * Returns true when we need to resched and can (barring IRQ state).
92 */
93 static __always_inline bool should_resched(void)
94 {
95 return unlikely(!raw_cpu_read_4(__preempt_count));
96 }
97
98 #ifdef CONFIG_PREEMPT
99 extern asmlinkage void ___preempt_schedule(void);
100 # define __preempt_schedule() asm ("call ___preempt_schedule")
101 extern asmlinkage void preempt_schedule(void);
102 # ifdef CONFIG_CONTEXT_TRACKING
103 extern asmlinkage void ___preempt_schedule_context(void);
104 # define __preempt_schedule_context() asm ("call ___preempt_schedule_cc
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d59b81 in should_resched+0 at ./arch/x86/include/asm/pre
should_resched () at ./arch/x86/include/asm/preempt.h:95
95 return unlikely(!raw_cpu_read_4(__preempt_count));
>>>

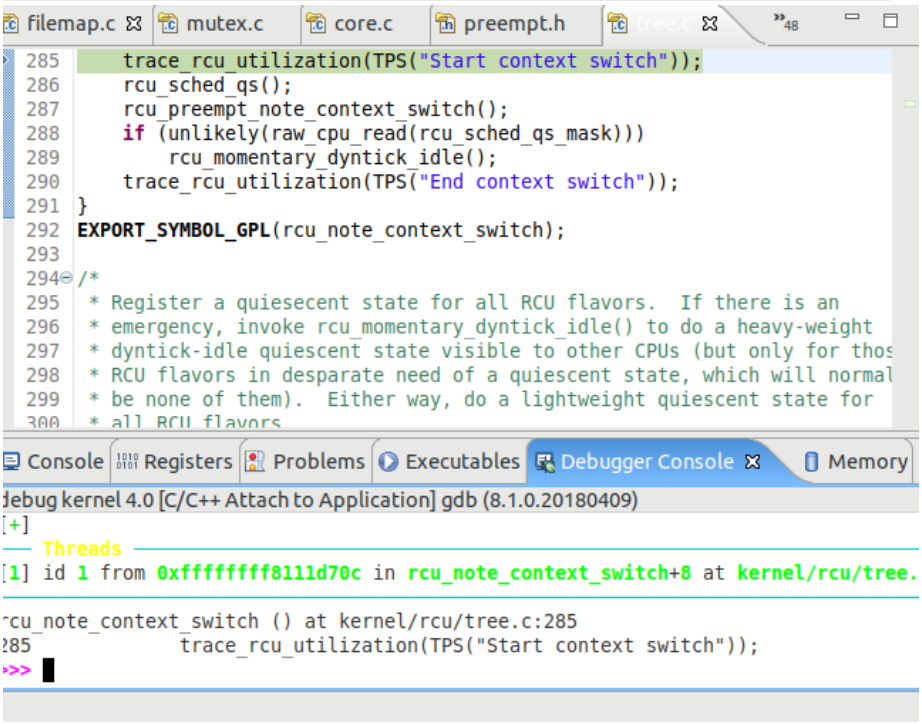

filemap.c
mutex.c
preempt.h
4206 if (should_resched()) {
4207 preempt_schedule_common();
4208 return 1;
4209 }
4210 return 0;
4211 }
4212 EXPORT_SYMBOL(_cond_resched);
4213
4214 /*
4215 * __cond_resched_lock() - if a reschedule is pending, drop the given lo
4216 * call schedule, and on return reacquire the lock.
4217 *
4218 * This works OK both with and without CONFIG_PREEMPT. We do strange low
4219 * operations here to prevent schedule() from being called twice (once v
4220 * spin_unlock(), once by hand).
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d59b9c in _cond_resched+35 at kernel/sched/core.c:4206
_cond_resched () at kernel/sched/core.c:4206
4206 if (should_resched()) {
>>>

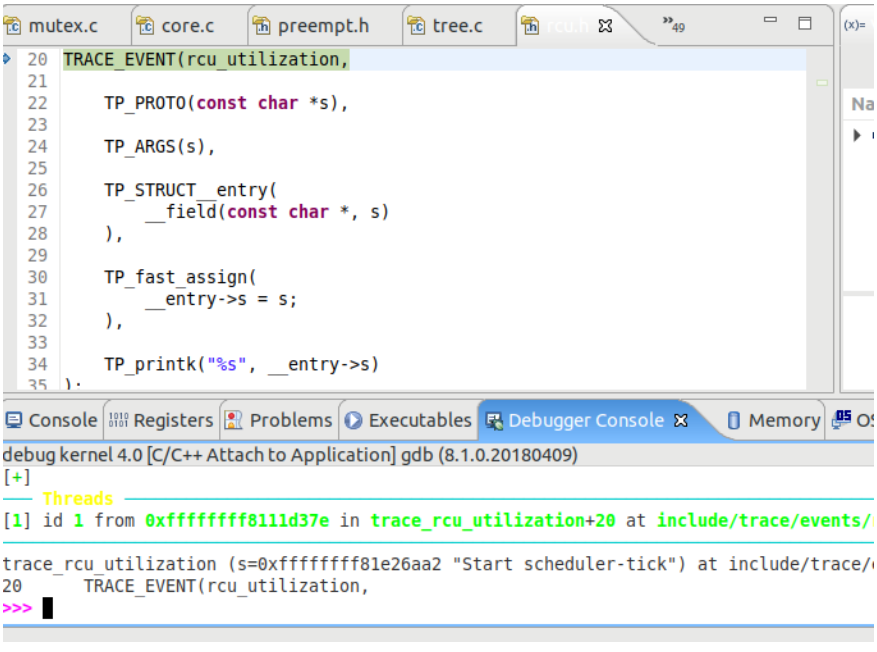

filemap.c
mutex.c
preempt.h
2840 }
2841
2842 static void __sched notrace preempt_schedule_common(void)
2843 {
2844 do {
2845 __preempt_count_add(PREEMPT_ACTIVE);
2846 __schedule();
2847 __preempt_count_sub(PREEMPT_ACTIVE);
2848
2849 /*
2850 * Check again in case we missed a preemption opportunity
2851 * between schedule and now.
2852 */
2853 barrier();
2854 } while (need_resched());
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d599df in preempt_schedule_common+15 at kernel/sched/c
preempt_schedule_common () at kernel/sched/core.c:2845
2845 __preempt_count_add(PREEMPT_ACTIVE);
>>>

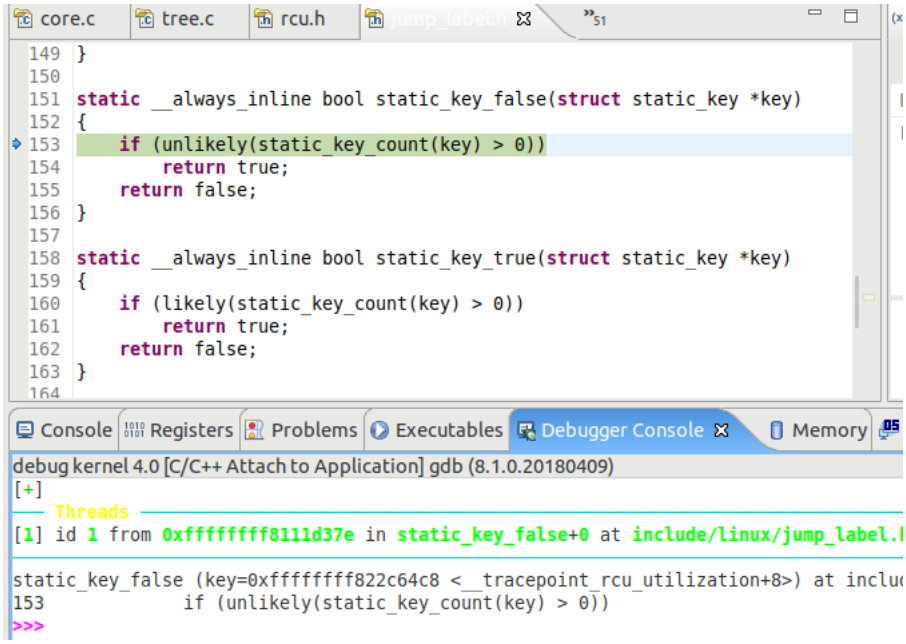

filemap.c
mutex.c
core.c
67 * The various preempt_count add/sub methods
68 */
69
70 static __always_inline void __preempt_count_add(int val)
71 {
72 raw_cpu_add_4(__preempt_count, val);
73 }
74
75 static __always_inline void __preempt_count_sub(int val)
76 {
77 raw_cpu_add_4(__preempt_count, -val);
78 }
79
80 /*
81 * Because we keep PREEMPT_NEED_RESCHED set when we do _not_ need to resc
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d599df in __preempt_count_add+0 at ./arch/x86/include/
__preempt_count_add (val=2097152) at ./arch/x86/include/asm/preempt.h:72
72 raw_cpu_add_4(__preempt_count, val);
>>>

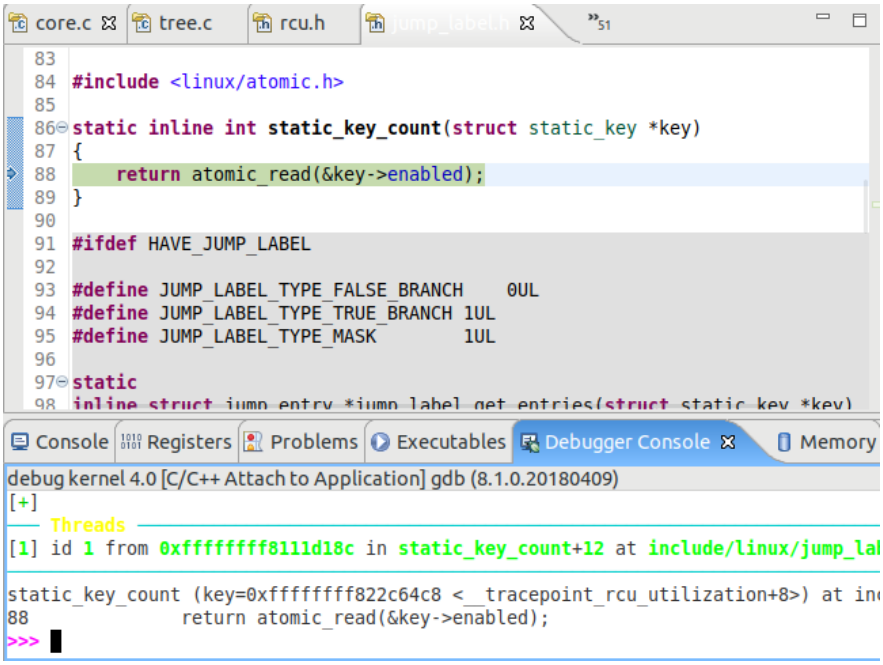

filemap.c
mutex.c
preempt.h
2840 }
2841
2842 static void __sched notrace preempt_schedule_common(void)
2843 {
2844 do {
2845 __preempt_count_add(PREEMPT_ACTIVE);
2846 __schedule();
2847 __preempt_count_sub(PREEMPT_ACTIVE);
2848
2849 /*
2850 * Check again in case we missed a preemption opportunity
2851 * between schedule and now.
2852 */
2853 barrier();
2854 } while (need_resched());
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d59a0e in preempt_schedule_common+62 at kernel/sched/core
preempt_schedule_common () at kernel/sched/core.c:2846
2846 __schedule();
>>>

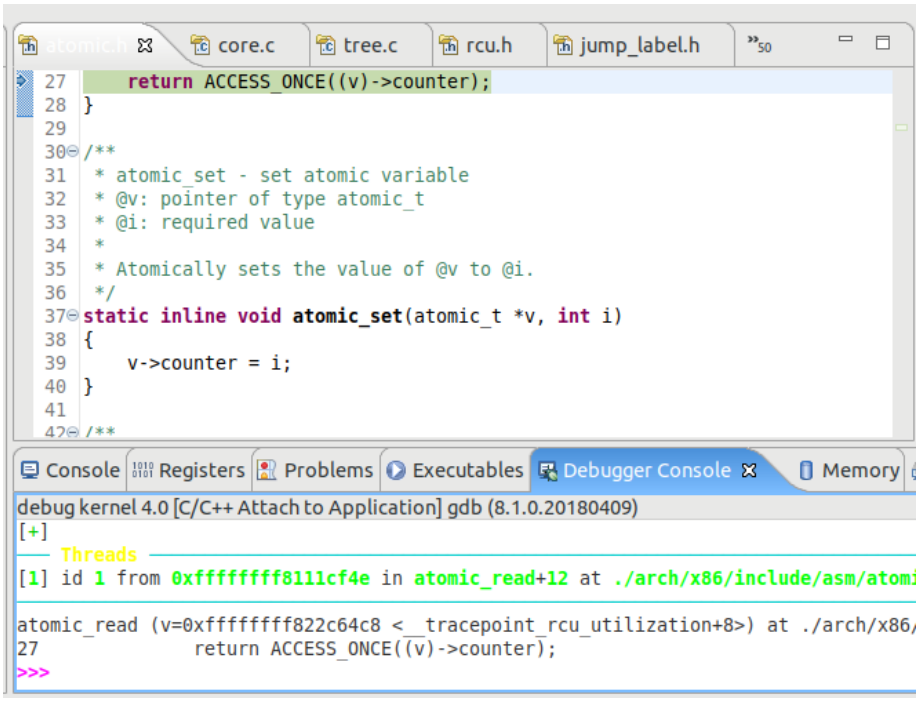

filemap.c
mutex.c
core.c
preempt.h
2716 unsigned long *switch_count;
2717 struct rq *rq;
2718 int cpu;
2719
2720 preempt_disable();
2721 cpu = smp_processor_id();
2722 rq = cpu_rq(cpu);
2723 rcu_note_context_switch();
2724 prev = rq->curr;
2725
2726 schedule_debug(prev);
2727
2728 if (sched_feat(HRTICK))
2729 hrtick_clear(rq);
2730
2731 /*
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d59708 in __schedule+8 at kernel/sched/core.c:2721
__schedule () at kernel/sched/core.c:2721
2721 cpu = smp_processor_id();
>>>

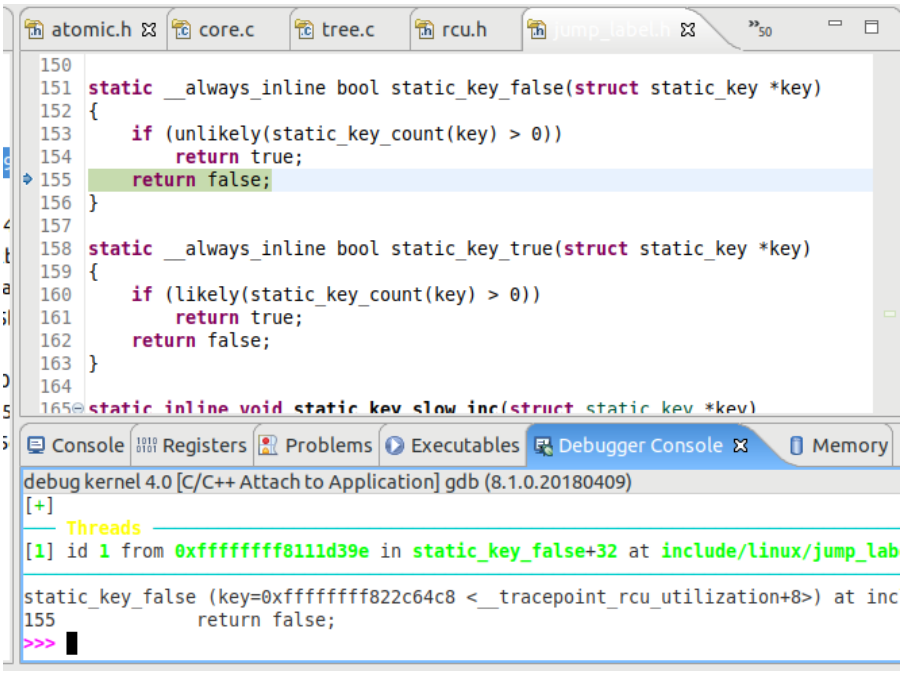

filemap.c
mutex.c
core.c
preempt.h
285 trace_rcu_utilization(TPS("Start context switch"));
286 rcu_sched_qs();
287 rcu_preempt_note_context_switch();
288 if (unlikely(raw_cpu_read(rcu_sched_qs_mask)))
289 rcu_momentary_dyntick_idle();
290 trace_rcu_utilization(TPS("End context switch"));
291 }
292 EXPORT_SYMBOL_GPL(rcu_note_context_switch);
293
294 /*
295 * Register a quiesecent state for all RCU flavors. If there is an
296 * emergency, invoke rcu_momentary_dyntick_idle() to do a heavy-weight
297 * dyntick-idle quiescent state visible to other CPUs (but only for thos
298 * RCU flavors in desparate need of a quiescent state, which will normal
299 * be none of them). Either way, do a lightweight quiescent state for
300 * all RCU flavors
Console
Registers
Problems
Executables
Debugger Console
Memory
Threads
[1] id 1 from 0xffffffff8111d70c in rcu_note_context_switch+8 at kernel/rcu/tree.
rcu_note_context_switch () at kernel/rcu/tree.c:285
285 trace_rcu_utilization(TPS("Start context switch"));
>>>

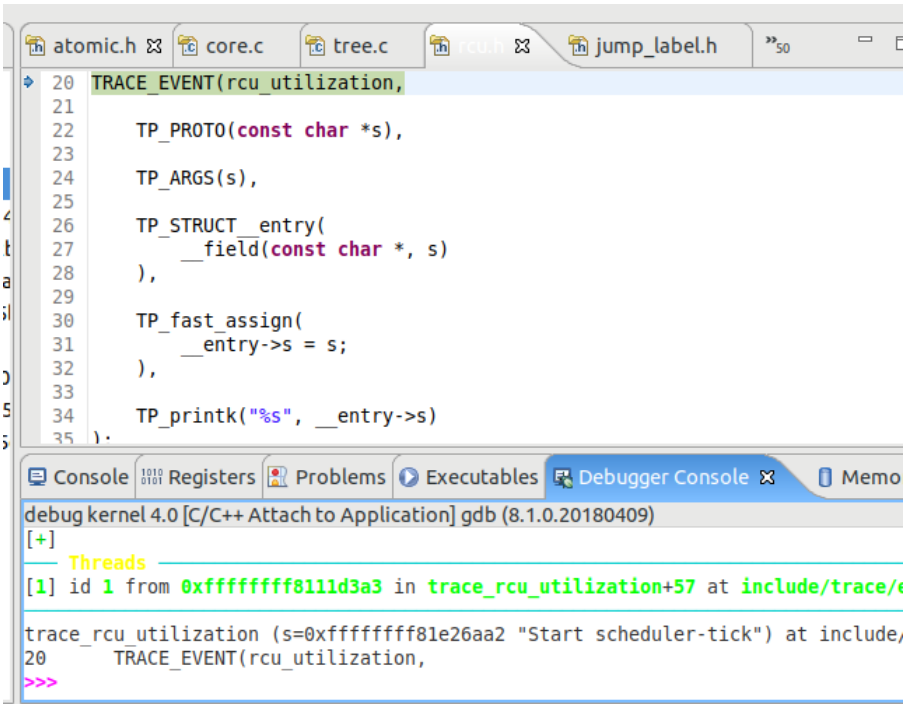

mutex.c
core.c
preempt.h
tree.c
20 TRACE_EVENT(rcu_utilization,
21
22 TP_PROTO(const char *s),
23
24 TP_ARGS(s),
25
26 TP_STRUCT__entry(
27 __field(const char *, s)
28),
29
30 TP_fast_assign(
31 __entry->s = s;
32),
33
34 TP_printk("%s", __entry->s)
35);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d37e in trace_rcu_utilization+20 at include/trace/events/
trace_rcu_utilization (s=0xffffffff81e26aa2 "Start scheduler-tick") at include/trace/
20 TRACE_EVENT(rcu_utilization,
>>>

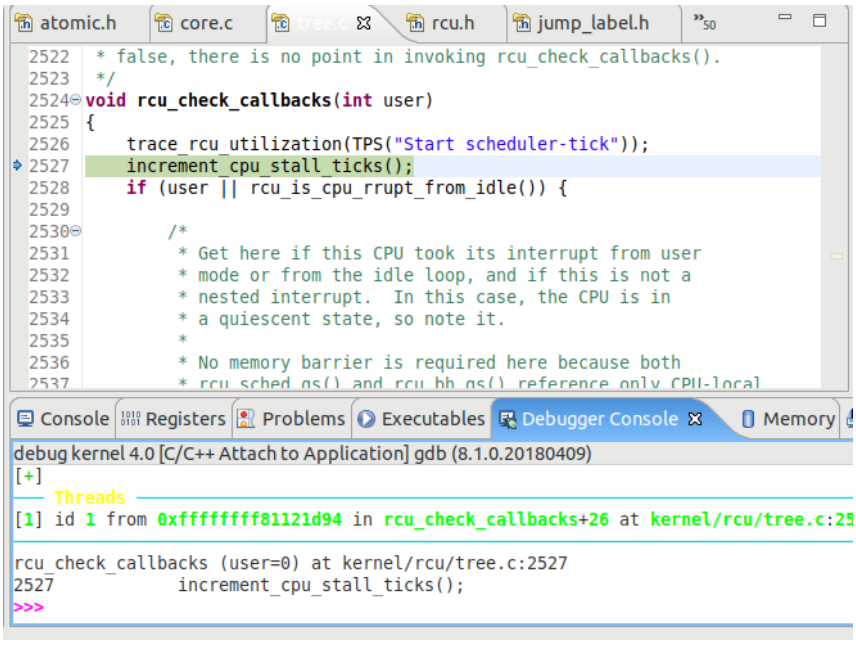

core.c
tree.c
rcu.h
jump_label.h
149 }
150
151 static __always_inline bool static_key_false(struct static_key *key)
152 {
153 if (unlikely(static_key_count(key) > 0))
154 return true;
155 return false;
156 }
157
158 static __always_inline bool static_key_true(struct static_key *key)
159 {
160 if (likely(static_key_count(key) > 0))
161 return true;
162 return false;
163 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d37e in static_key_false+0 at include/linux/jump_label.h
static_key_false (key=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at inclu
153 if (unlikely(static_key_count(key) > 0))
>>>

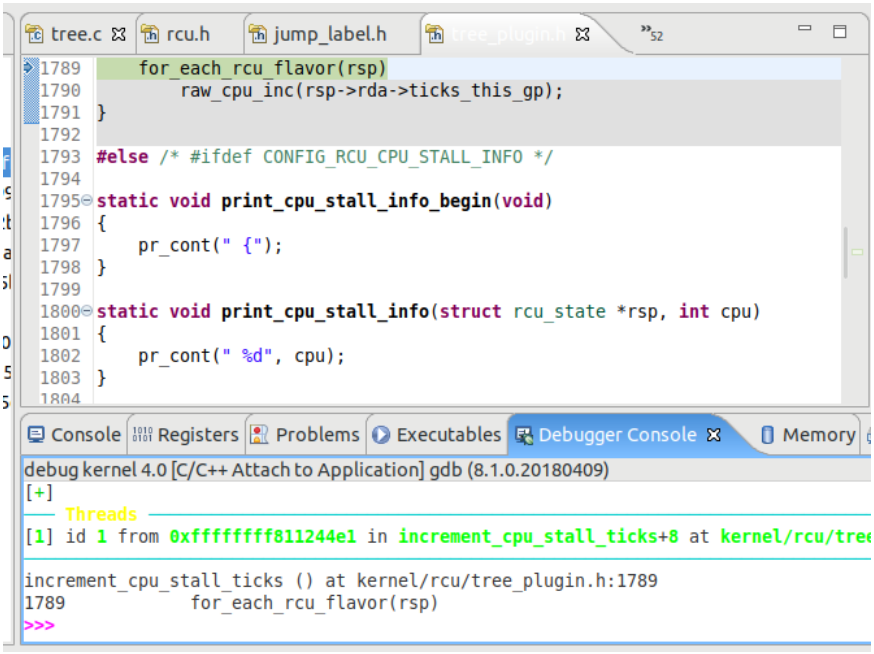

core.c
tree.c
rcu.h
jump_label.h
83
84 #include <linux/atomic.h>
85
86 static inline int static_key_count(struct static_key *key)
87 {
88 return atomic_read(&key->enabled);
89 }
90
91 #ifdef HAVE_JUMP_LABEL
92
93 #define JUMP_LABEL_TYPE_FALSE_BRANCH 0UL
94 #define JUMP_LABEL_TYPE_TRUE_BRANCH 1UL
95 #define JUMP_LABEL_TYPE_MASK 1UL
96
97 static
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d18c in static_key_count+12 at include/linux/jump_lab
static_key_count (key=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at in
88 return atomic_read(&key->enabled);
>>>

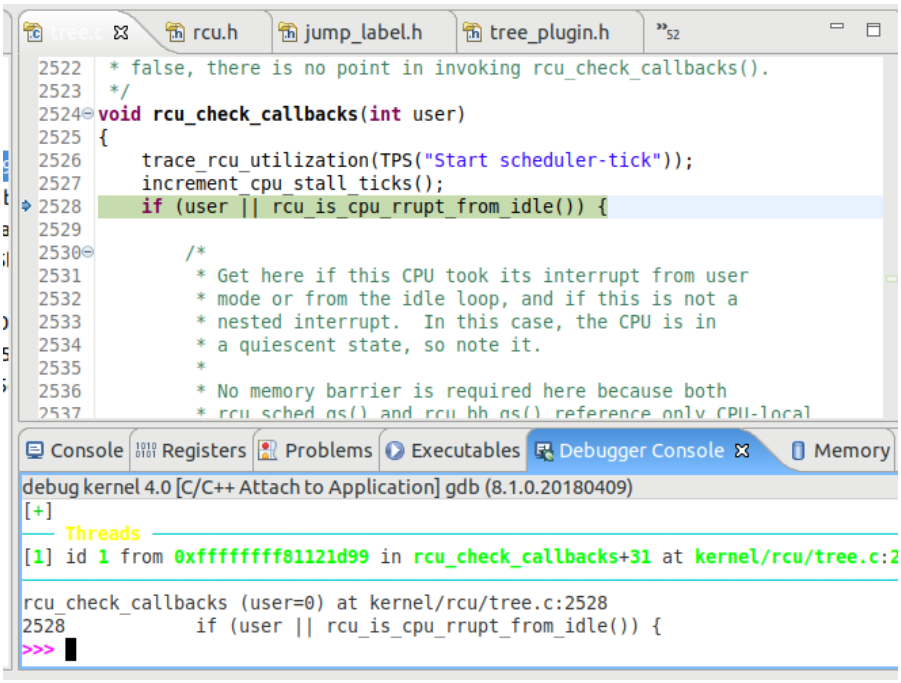

atomic.h
core.c
tree.c
rcu.h
jump_label.h
27 return ACCESS_ONCE((v)->counter);
28 }
29
30 /**
31 * atomic_set - set atomic variable
32 * @v: pointer of type atomic_t
33 * @i: required value
34 *
35 * Atomically sets the value of @v to @i.
36 */
37 static inline void atomic_set(atomic_t *v, int i)
38 {
39 v->counter = i;
40 }
41
42 /**
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111cf4e in atomic_read+12 at ./arch/x86/include/asm/atomi
atomic_read (v=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at ./arch/x86/
27 return ACCESS_ONCE((v)->counter);
>>>

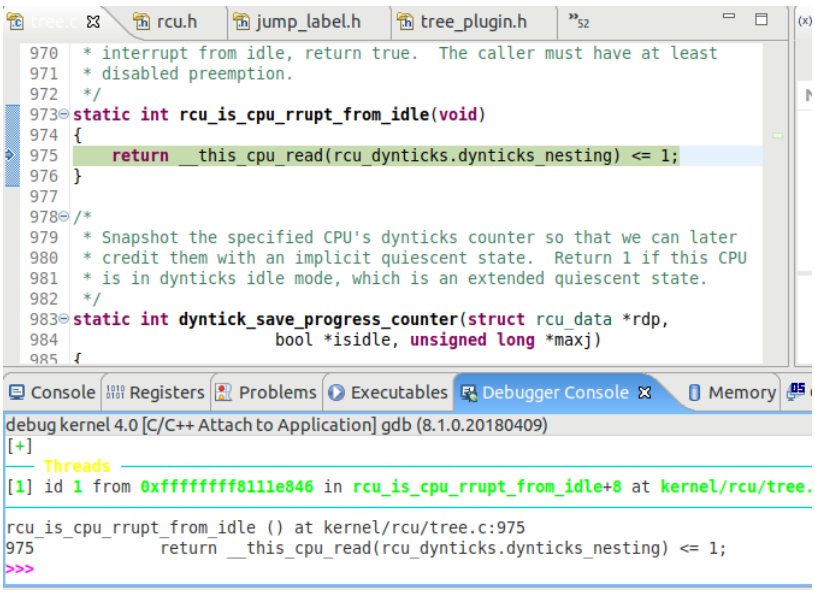

atomic.h
core.c
tree.c
rcu.h
jump_label.h
150
151 static __always_inline bool static_key_false(struct static_key *key)
152 {
153 if (unlikely(static_key_count(key) > 0))
154 return true;
155 return false;
156 }
157
158 static __always_inline bool static_key_true(struct static_key *key)
159 {
160 if (likely(static_key_count(key) > 0))
161 return true;
162 return false;
163 }
164
165 static inline void static_key_slow_inc(struct static_key *key)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d39e in static_key_false+32 at include/linux/jump_lab
static_key_false (key=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at inc
155 return false;
>>>

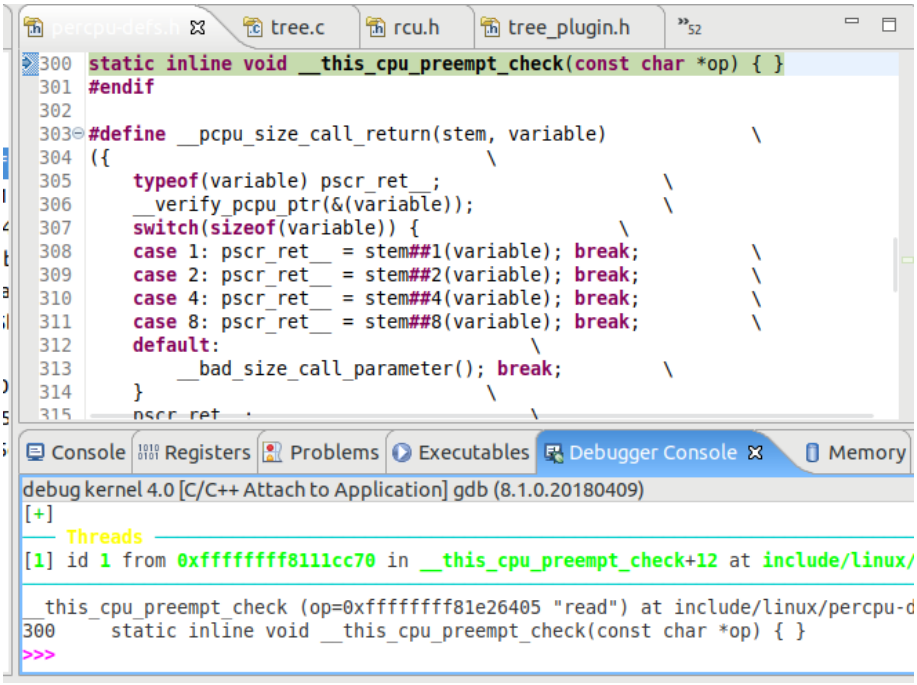

atomic.h
core.c
tree.c
jump_label.h
20 TRACE_EVENT(rcu_utilization,
21
22 TP_PROTO(const char *s),
23
24 TP_ARGS(s),
25
26 TP_STRUCT__entry(
27 __field(const char *, s)
28),
29
30 TP_fast_assign(
31 __entry->s = s;
32),
33
34 TP_printk("%s", __entry->s)
35);
Console
Registers
Problems
Executables
Debugger Console
Memo
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d3a3 in trace_rcu_utilization+57 at include/trace/e
trace_rcu_utilization (s=0xffffffff81e26aa2 "Start scheduler-tick") at include/
20 TRACE_EVENT(rcu_utilization,
>>>

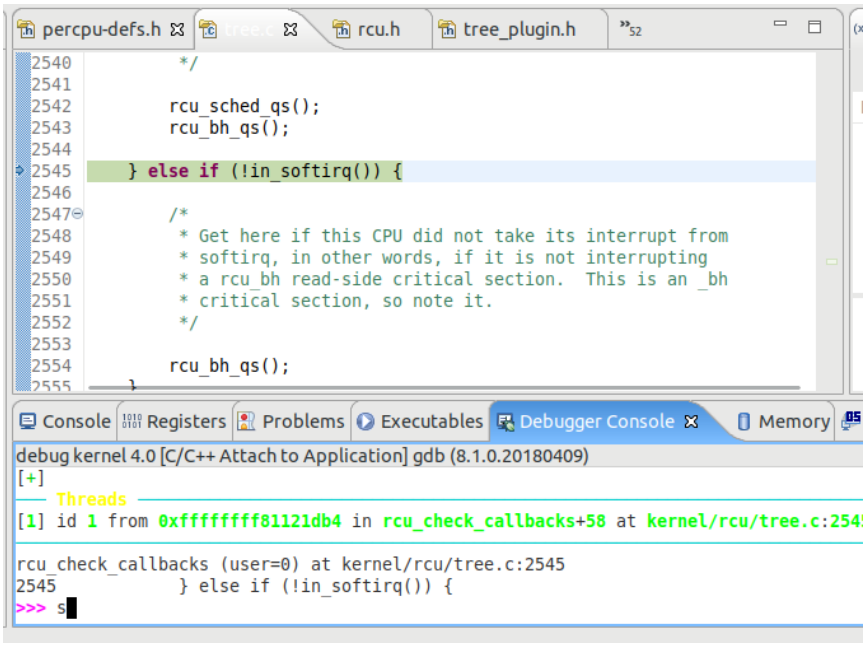

atomic.h
core.c
rcu.h
jump_label.h
2522 * false, there is no point in invoking rcu_check_callbacks().
2523 */
2524 void rcu_check_callbacks(int user)
2525 {
2526 trace_rcu_utilization(TPS("Start scheduler-tick"));
2527 increment_cpu_stall_ticks();
2528 if (user || rcu_is_cpu_rrupt_from_idle()) {
2529
2530 /*
2531 * Get here if this CPU took its interrupt from user
2532 * mode or from the idle loop, and if this is not a
2533 * nested interrupt. In this case, the CPU is in
2534 * a quiescent state, so note it.
2535 *
2536 * No memory barrier is required here because both
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121d94 in rcu_check_callbacks+26 at kernel/rcu/tree.c:25
rcu_check_callbacks (user=0) at kernel/rcu/tree.c:2527
2527 increment_cpu_stall_ticks();
>>>

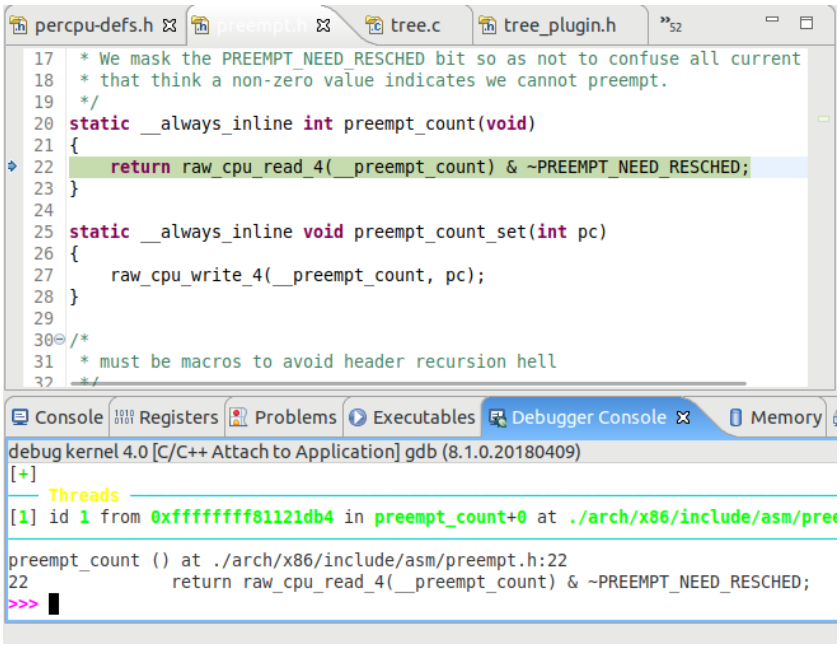

tree.c
rcu.h
jump_label.h
tree_plugin.h
1789 for_each_rcu_flavor(rsp)
1790 raw_cpu_inc(rsp->rda->ticks_this_gp);
1791 }
1792
1793 #else /* #ifdef CONFIG_RCU_CPU_STALL_INFO */
1794
1795 static void print_cpu_stall_info_begin(void)
1796 {
1797 pr_cont(" {");
1798 }
1799
1800 static void print_cpu_stall_info(struct rcu_state *rsp, int cpu)
1801 {
1802 pr_cont(" %d", cpu);
1803 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811244e1 in increment_cpu_stall_ticks+8 at kernel/rcu/tree
increment_cpu_stall_ticks () at kernel/rcu/tree_plugin.h:1789
1789 for_each_rcu_flavor(rsp)
>>>

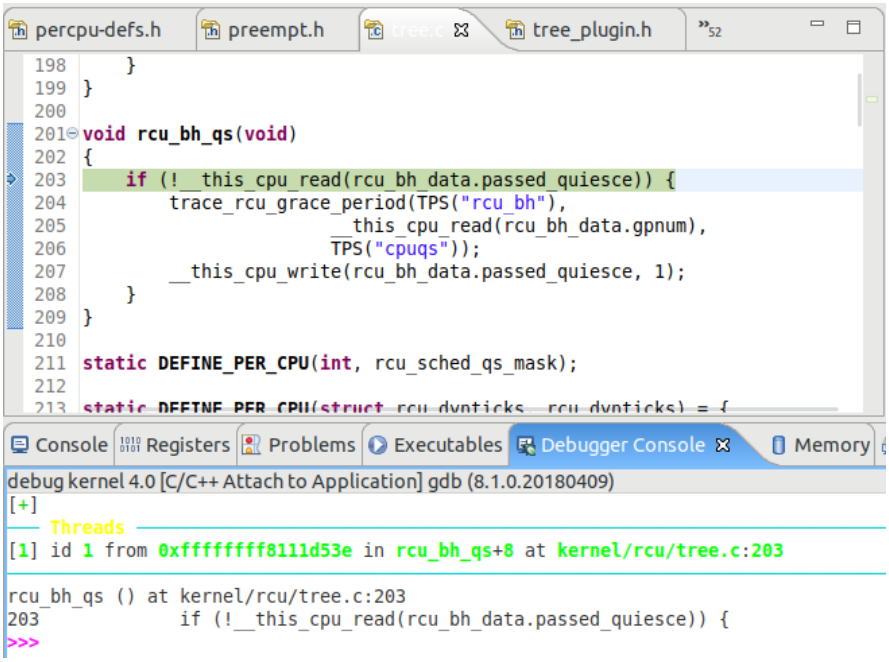

rcu.h
jump_label.h
tree_plugin.h
2522 * false, there is no point in invoking rcu_check_callbacks().
2523 */
2524 void rcu_check_callbacks(int user)
2525 {
2526 trace_rcu_utilization(TPS("Start scheduler-tick"));
2527 increment_cpu_stall_ticks();
2528 if (user || rcu_is_cpu_rrupt_from_idle()) {
2529
2530 /*
2531 * Get here if this CPU took its interrupt from user
2532 * mode or from the idle loop, and if this is not a
2533 * nested interrupt. In this case, the CPU is in
2534 * a quiescent state, so note it.
2535 *
2536 * No memory barrier is required here because both
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121d99 in rcu_check_callbacks+31 at kernel/rcu/tree.c:2
rcu_check_callbacks (user=0) at kernel/rcu/tree.c:2528
2528 if (user || rcu_is_cpu_rrupt_from_idle()) {
>>>

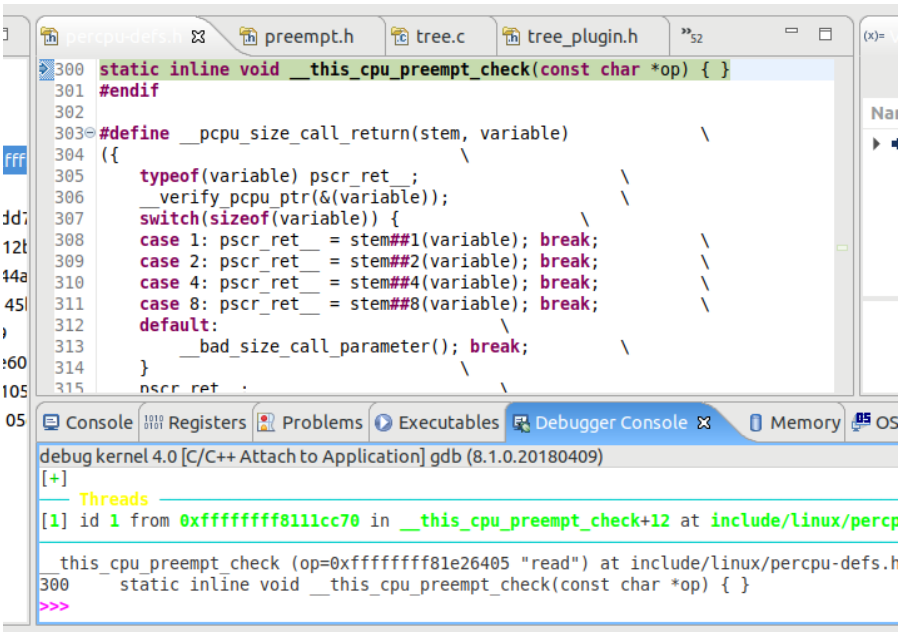

rcu.h
jump_label.h
tree_plugin.h
970 * interrupt from idle, return true. The caller must have at least
971 * disabled preemption.
972 */
973 static int rcu_is_cpu_rrupt_from_idle(void)
974 {
975 return __this_cpu_read(rcu_dynticks.dynticks_nesting) <= 1;
976 }
977
978 /*
979 * Snapshot the specified CPU's dynticks counter so that we can later
980 * credit them with an implicit quiescent state. Return 1 if this CPU
981 * is in dynticks idle mode, which is an extended quiescent state.
982 */
983 static int dyntick_save_progress_counter(struct rcu_data *rdp,
984 bool *isidle, unsigned long *maxj)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111e846 in rcu_is_cpu_rrupt_from_idle+8 at kernel/rcu/tree.
rcu_is_cpu_rrupt_from_idle () at kernel/rcu/tree.c:975
975 return __this_cpu_read(rcu_dynticks.dynticks_nesting) <= 1;
>>>

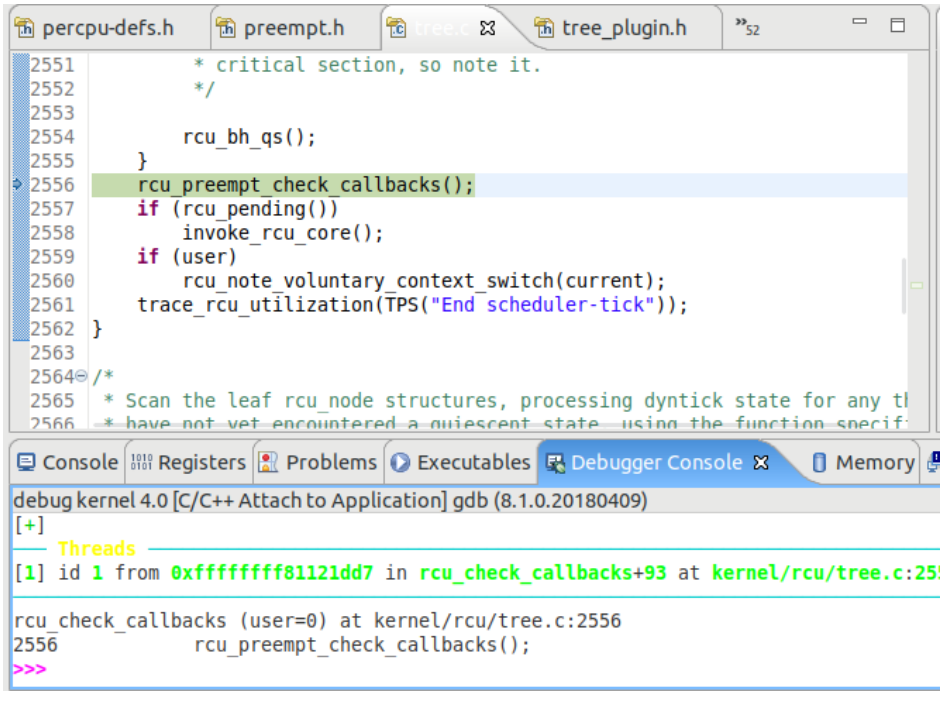

percpu-defs.h
tree.c
rcu.h
tree_plugin.h
300 static inline void __this_cpu_preempt_check(const char *op) { }
301 #endif
302
303 #define __pcpu_size_call_return(stem, variable) \
304 ({ \
305 typeof(variable) pscr_ret__; \
306 __verify_pcpu_ptr(&(variable)); \
307 switch(sizeof(variable)) { \
308 case 1: pscr_ret__ = stem##1(variable); break; \
309 case 2: pscr_ret__ = stem##2(variable); break; \
310 case 4: pscr_ret__ = stem##4(variable); break; \
311 case 8: pscr_ret__ = stem##8(variable); break; \
312 default: \
313 __bad_size_call_parameter(); break; \
314 } \
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111cc70 in __this_cpu_preempt_check+12 at include/linux/
__this_cpu_preempt_check (op=0xffffffff81e26405 "read") at include/linux/percpu-d
300 static inline void __this_cpu_preempt_check(const char *op) { }
>>>

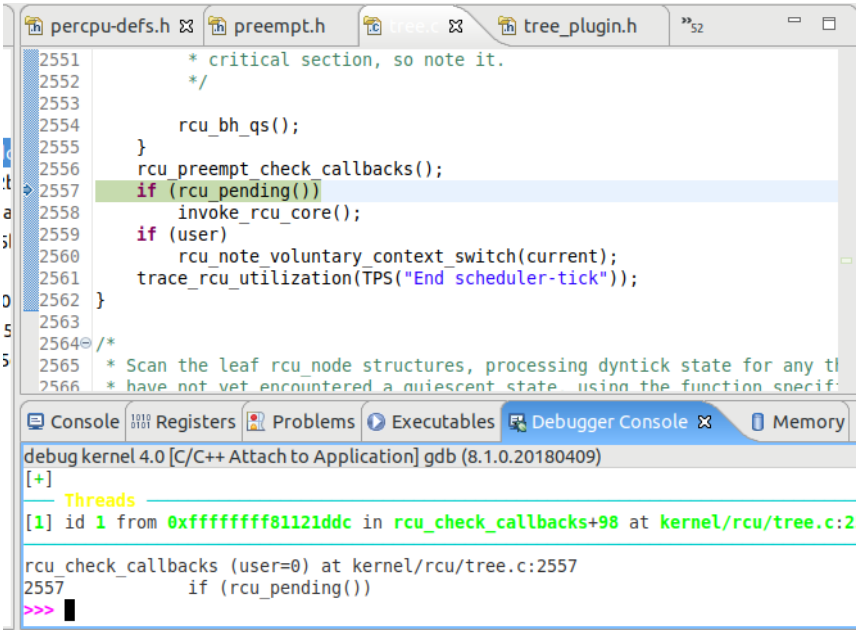

percpu-defs.h
rcu.h
tree_plugin.h
2540 */
2541
2542 rcu_sched_qs();
2543 rcu_bh_qs();
2544
2545 } else if (!in_softirq()) {
2546
2547 /*
2548 * Get here if this CPU did not take its interrupt from
2549 * softirq, in other words, if it is not interrupting
2550 * a rcu_bh read-side critical section. This is an _bh
2551 * critical section, so note it.
2552 */
2553
2554 rcu_bh_qs();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121db4 in rcu_check_callbacks+58 at kernel/rcu/tree.c:254
rcu_check_callbacks (user=0) at kernel/rcu/tree.c:2545
2545 } else if (!in_softirq()) {
>>> s

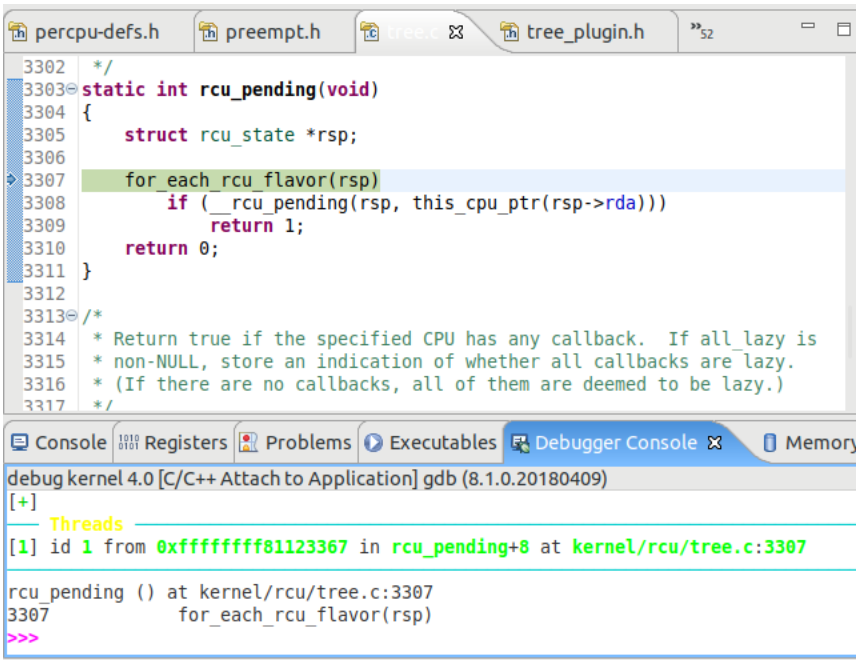

percpu-defs.h
tree.c
tree_plugin.h
17 * We mask the PREEMPT_NEED_RESCHED bit so as not to confuse all current
18 * that think a non-zero value indicates we cannot preempt.
19 */
20 static __always_inline int preempt_count(void)
21 {
22 return raw_cpu_read_4(__preempt_count) & ~PREEMPT_NEED_RESCHED;
23 }
24
25 static __always_inline void preempt_count_set(int pc)
26 {
27 raw_cpu_write_4(__preempt_count, pc);
28 }
29
30 /*
31 * must be macros to avoid header recursion hell
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121db4 in preempt_count+0 at ./arch/x86/include/asm/pree
preempt_count () at ./arch/x86/include/asm/preempt.h:22
22 return raw_cpu_read_4(__preempt_count) & ~PREEMPT_NEED_RESCHED;
>>>

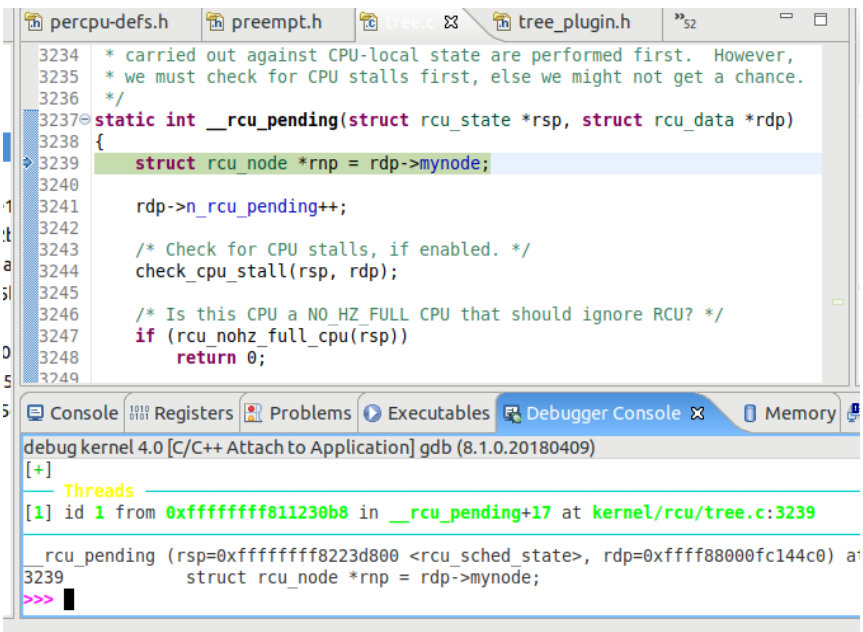

percpu-defs.h
preempt.h
tree_plugin.h
198 }
199 }
200
201 void rcu_bh_qs(void)
202 {
203 if (!__this_cpu_read(rcu_bh_data.passed_quiesce)) {
204 trace_rcu_grace_period(TPS("rcu_bh"),
205 __this_cpu_read(rcu_bh_data.gpnum),
206 TPS("cpuqs"));
207 __this_cpu_write(rcu_bh_data.passed_quiesce, 1);
208 }
209 }
210
211 static DEFINE_PER_CPU(int, rcu_sched_qs_mask);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d53e in rcu_bh_qs+8 at kernel/rcu/tree.c:203
rcu_bh_qs () at kernel/rcu/tree.c:203
203 if (!__this_cpu_read(rcu_bh_data.passed_quiesce)) {
>>>

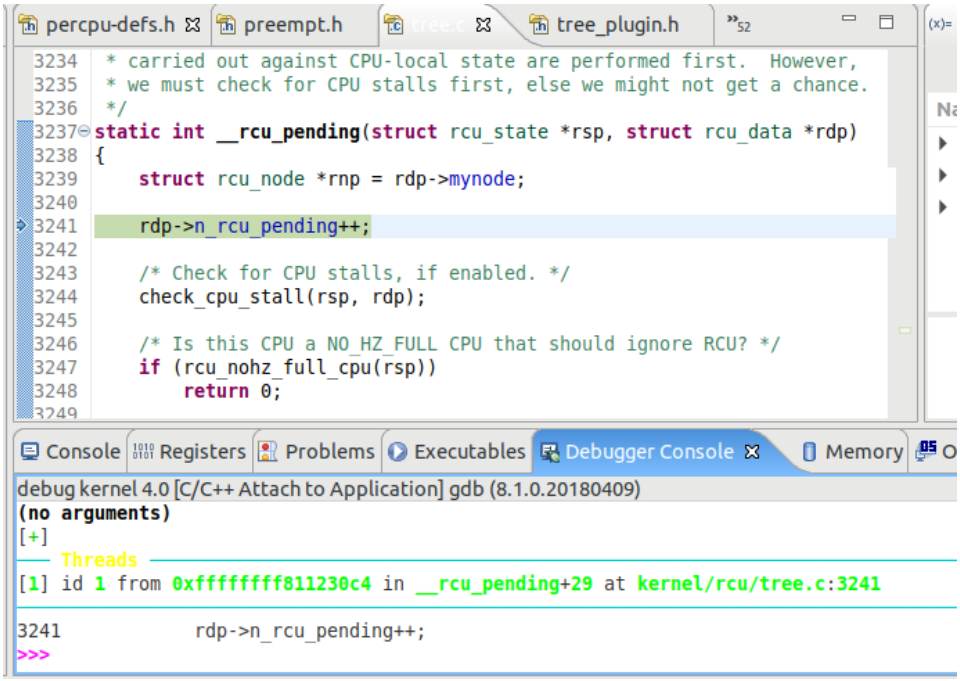

preempt.h
tree.c
tree_plugin.h
300 static inline void __this_cpu_preempt_check(const char *op) { }
301 #endif
302
303 #define __pcpu_size_call_return(stem, variable) \
304 ({ \
305 typeof(variable) pscr_ret__; \
306 __verify_pcpu_ptr(&(variable)); \
307 switch(sizeof(variable)) { \
308 case 1: pscr_ret__ = stem##1(variable); break; \
309 case 2: pscr_ret__ = stem##2(variable); break; \
310 case 4: pscr_ret__ = stem##4(variable); break; \
311 case 8: pscr_ret__ = stem##8(variable); break; \
312 default: \
313 __bad_size_call_parameter(); break; \
314 } \
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111cc70 in __this_cpu_preempt_check+12 at include/linux/percp
__this_cpu_preempt_check (op=0xffffffff81e26405 "read") at include/linux/percpu-defs.h
300 static inline void __this_cpu_preempt_check(const char *op) { }
>>>

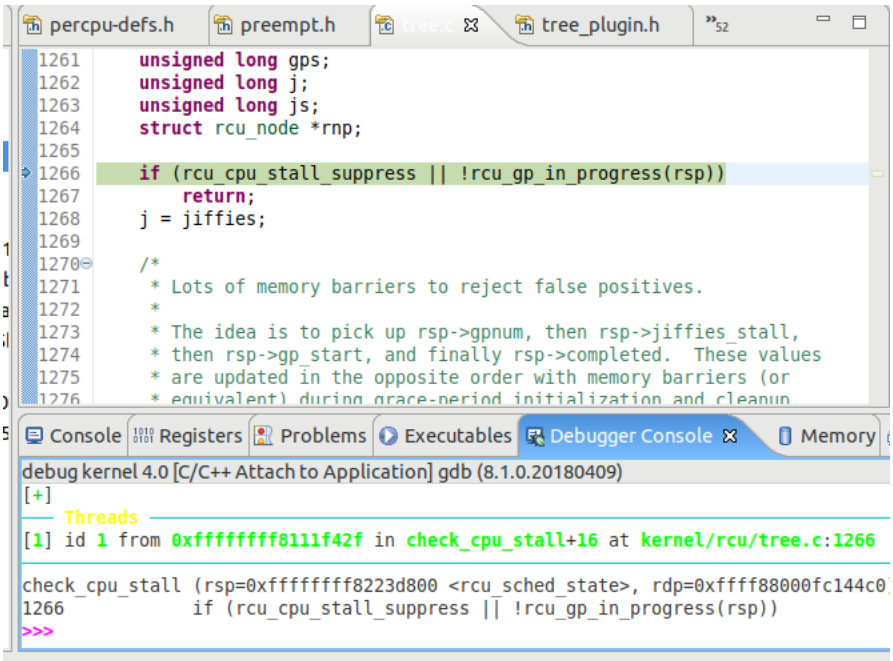

percpu-defs.h
preempt.h
tree_plugin.h
2551 * critical section, so note it.
2552 */
2553
2554 rcu_bh_qs();
2555 }
2556 rcu_preempt_check_callbacks();
2557 if (rcu_pending())
2558 invoke_rcu_core();
2559 if (user)
2560 rcu_note_voluntary_context_switch(current);
2561 trace_rcu_utilization(TPS("End scheduler-tick"));
2562 }
2563
2564 /*
2565 * Scan the leaf rcu_node structures, processing dyntick state for any th
2566 * have not yet encountered a quiescent state, using the function specif
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121dd7 in rcu_check_callbacks+93 at kernel/rcu/tree.c:25
rcu_check_callbacks (user=0) at kernel/rcu/tree.c:2556
2556 rcu_preempt_check_callbacks();
>>>

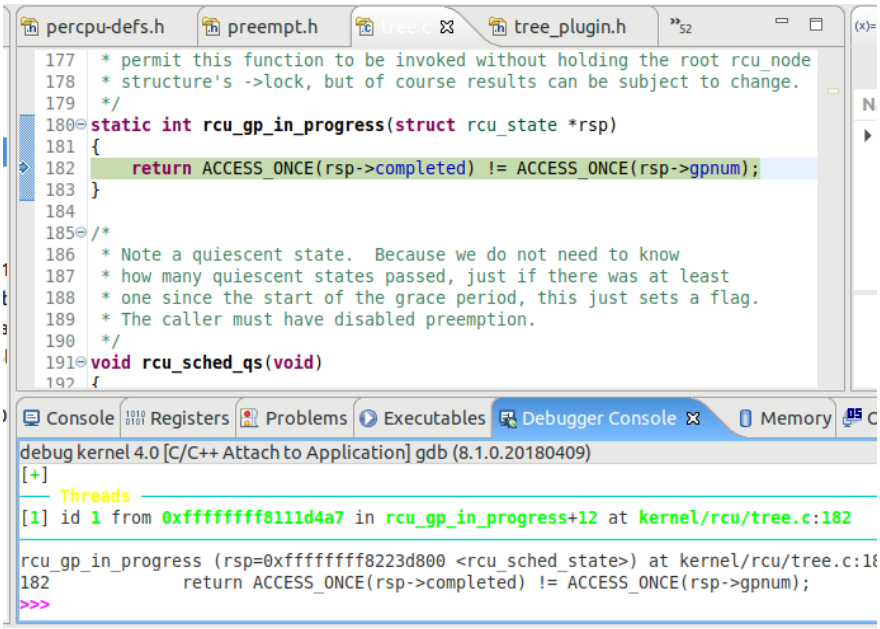

percpu-defs.h
preempt.h
tree_plugin.h
2551 * critical section, so note it.
2552 */
2553
2554 rcu_bh_qs();
2555 }
2556 rcu_preempt_check_callbacks();
2557 if (rcu_pending())
2558 invoke_rcu_core();
2559 if (user)
2560 rcu_note_voluntary_context_switch(current);
2561 trace_rcu_utilization(TPS("End scheduler-tick"));
2562 }
2563
2564 /*
2565 * Scan the leaf rcu_node structures, processing dyntick state for any th
2566 * have not yet encountered a quiescent state, using the function specif
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121ddc in rcu_check_callbacks+98 at kernel/rcu/tree.c:2
rcu_check_callbacks (user=0) at kernel/rcu/tree.c:2557
2557 if (rcu_pending())
>>>

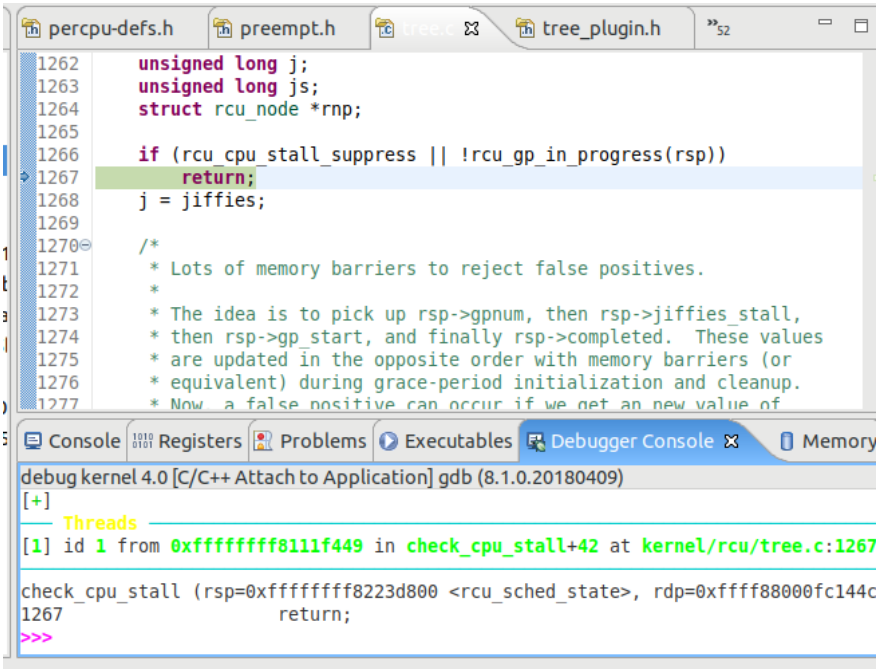

percpu-defs.h
preempt.h
tree_plugin.h
3302 */
3303 static int rcu_pending(void)
3304 {
3305 struct rcu_state *rsp;
3306
3307 for_each_rcu_flavor(rsp)
3308 if (__rcu_pending(rsp, this_cpu_ptr(rsp->rda)))
3309 return 1;
3310 return 0;
3311 }
3312
3313 /*
3314 * Return true if the specified CPU has any callback. If all_lazy is
3315 * non-NULL, store an indication of whether all callbacks are lazy.
3316 * (If there are no callbacks, all of them are deemed to be lazy.)
3317 */
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81123367 in rcu_pending+8 at kernel/rcu/tree.c:3307
rcu_pending () at kernel/rcu/tree.c:3307
3307 for_each_rcu_flavor(rsp)
>>>

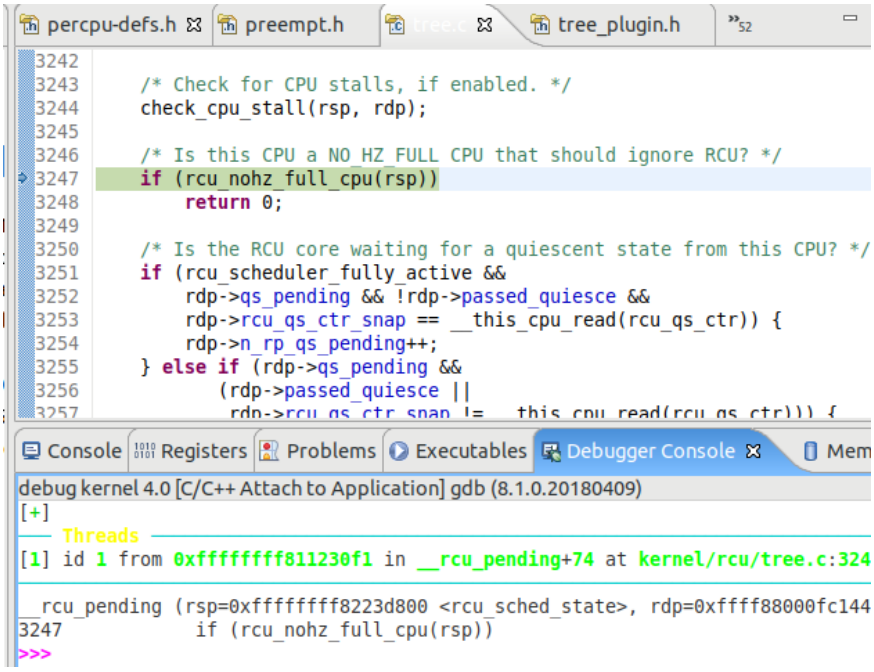

percpu-defs.h
preempt.h
tree_plugin.h
3234 * carried out against CPU-local state are performed first. However,
3235 * we must check for CPU stalls first, else we might not get a chance.
3236 */
3237 static int __rcu_pending(struct rcu_state *rsp, struct rcu_data *rdp)
3238 {
3239 struct rcu_node *rnp = rdp->mynode;
3240
3241 rdp->n_rcu_pending++;
3242
3243 /* Check for CPU stalls, if enabled. */
3244 check_cpu_stall(rsp, rdp);
3245
3246 /* Is this CPU a NO_HZ_FULL CPU that should ignore RCU? */
3247 if (rcu_nohz_full_cpu(rsp))
3248 return 0;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811230b8 in __rcu_pending+17 at kernel/rcu/tree.c:3239
__rcu_pending (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144c0) at
3239 struct rcu_node *rnp = rdp->mynode;
>>>

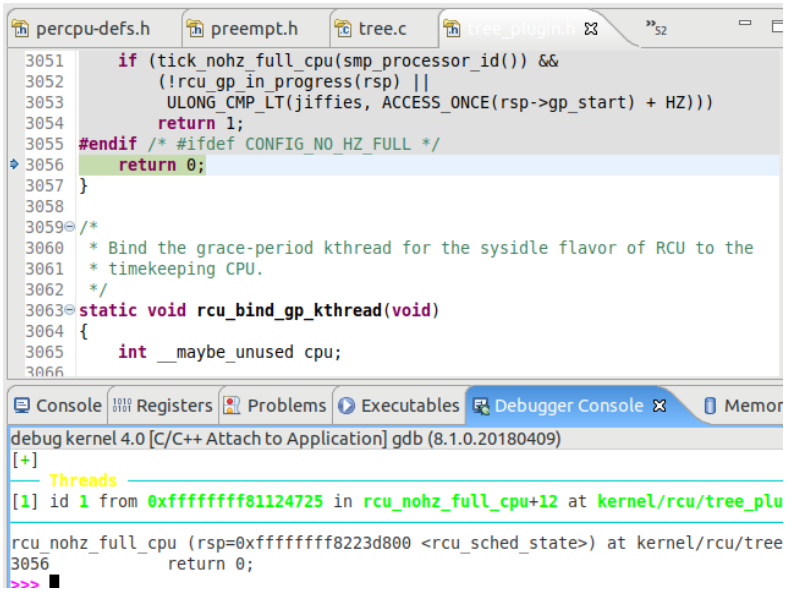

percpu-defs.h
preempt.h
tree_plugin.h
3234 * carried out against CPU-local state are performed first. However,
3235 * we must check for CPU stalls first, else we might not get a chance.
3236 */
3237 static int __rcu_pending(struct rcu_state *rsp, struct rcu_data *rdp)
3238 {
3239 struct rcu_node *rnp = rdp->mynode;
3240
3241 rdp->n_rcu_pending++;
3242
3243 /* Check for CPU stalls, if enabled. */
3244 check_cpu_stall(rsp, rdp);
3245
3246 /* Is this CPU a NO_HZ_FULL CPU that should ignore RCU? */
3247 if (rcu_nohz_full_cpu(rsp))
3248 return 0;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
(no arguments)
[+]
Threads
[1] id 1 from 0xffffffff811230c4 in __rcu_pending+29 at kernel/rcu/tree.c:3241
3241 rdp->n_rcu_pending++;
>>>

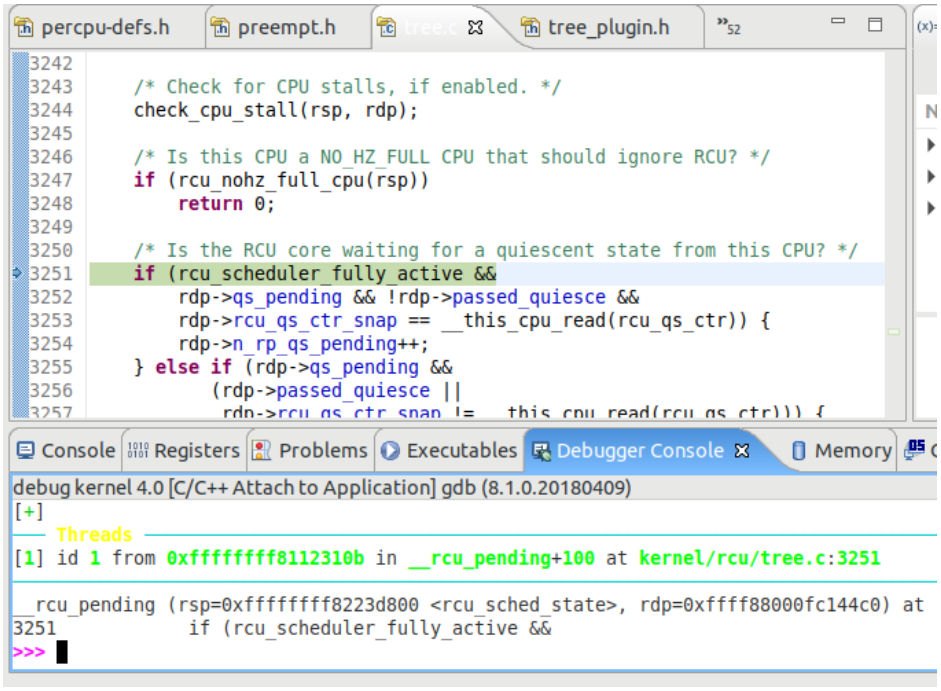

percpu-defs.h
preempt.h
tree_plugin.h
1261 unsigned long gps;
1262 unsigned long j;
1263 unsigned long js;
1264 struct rcu_node *rnp;
1265
1266 if (rcu_cpu_stall_suppress || !rcu_gp_in_progress(rsp))
1267 return;
1268 j = jiffies;
1269
1270 /*
1271 * Lots of memory barriers to reject false positives.
1272 *
1273 * The idea is to pick up rsp->gpnum, then rsp->jiffies_stall,
1274 * then rsp->gp_start, and finally rsp->completed. These values
1275 * are updated in the opposite order with memory barriers (or
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111f42f in check_cpu_stall+16 at kernel/rcu/tree.c:1266
check_cpu_stall (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144c0
1266 if (rcu_cpu_stall_suppress || !rcu_gp_in_progress(rsp))
>>>

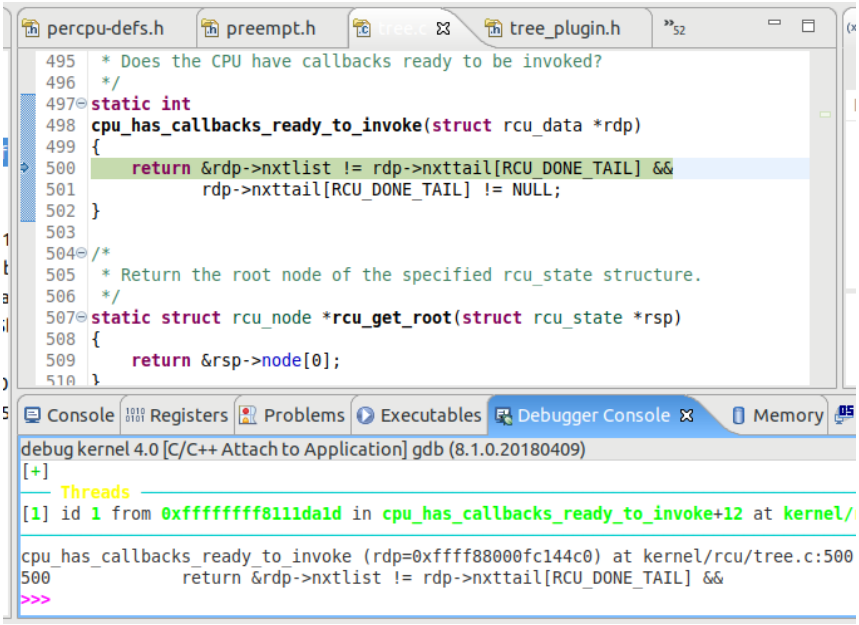

percpu-defs.h
preempt.h
tree_plugin.h
177 * permit this function to be invoked without holding the root rcu_node
178 * structure's ->lock, but of course results can be subject to change.
179 */
180 static int rcu_gp_in_progress(struct rcu_state *rsp)
181 {
182 return ACCESS_ONCE(rsp->completed) != ACCESS_ONCE(rsp->gpnum);
183 }
184
185 /*
186 * Note a quiescent state. Because we do not need to know
187 * how many quiescent states passed, just if there was at least
188 * one since the start of the grace period, this just sets a flag.
189 * The caller must have disabled preemption.
190 */
191 void rcu_sched_qs(void)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d4a7 in rcu_gp_in_progress+12 at kernel/rcu/tree.c:182
rcu_gp_in_progress (rsp=0xffffffff8223d800 <rcu_sched_state>) at kernel/rcu/tree.c:18
182 return ACCESS_ONCE(rsp->completed) != ACCESS_ONCE(rsp->gpnum);
>>>

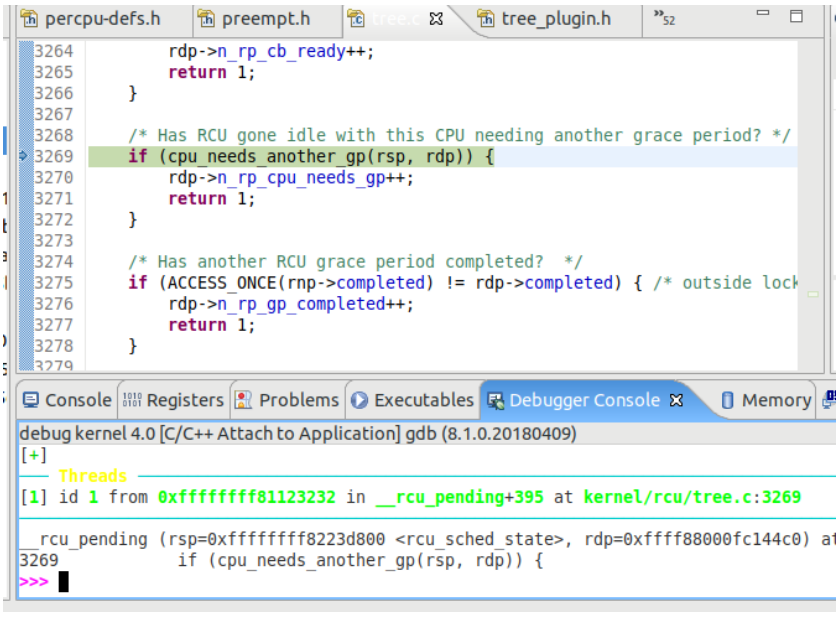

percpu-defs.h
preempt.h
tree_plugin.h
1262 unsigned long j;
1263 unsigned long js;
1264 struct rcu_node *rnp;
1265
1266 if (rcu_cpu_stall_suppress || !rcu_gp_in_progress(rsp))
1267 return;
1268 j = jiffies;
1269
1270 /*
1271 * Lots of memory barriers to reject false positives.
1272 *
1273 * The idea is to pick up rsp->gpnum, then rsp->jiffies_stall,
1274 * then rsp->gp_start, and finally rsp->completed. These values
1275 * are updated in the opposite order with memory barriers (or
1276 * equivalent) during grace-period initialization and cleanup.
1277 * Now a false positive can occur if we get an new value of
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111f449 in check_cpu_stall+42 at kernel/rcu/tree.c:1267
check_cpu_stall (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144c
1267 return;
>>>

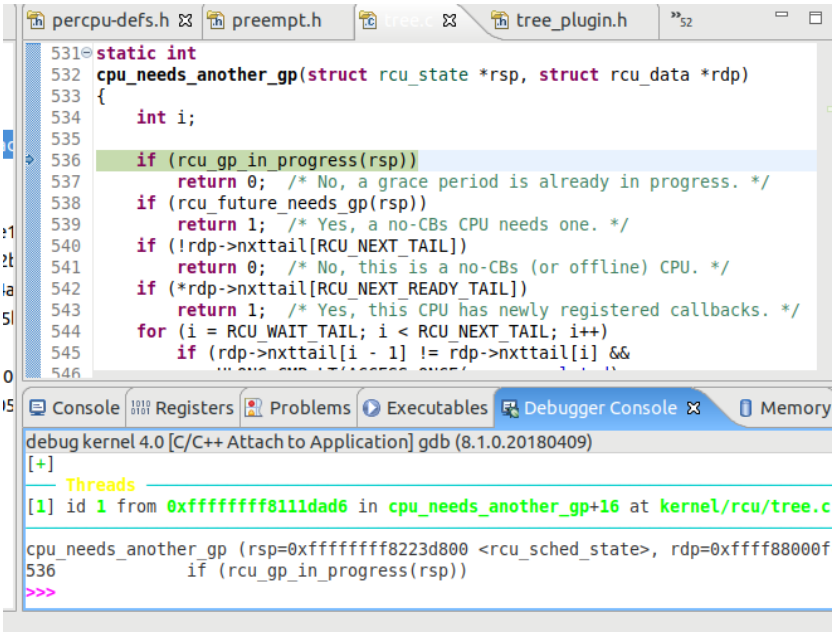

percpu-defs.h
preempt.h
tree_plugin.h
3242
3243 /* Check for CPU stalls, if enabled. */
3244 check_cpu_stall(rsp, rdp);
3245
3246 /* Is this CPU a NO_HZ_FULL CPU that should ignore RCU? */
3247 if (rcu_nohz_full_cpu(rsp))
3248 return 0;
3249
3250 /* Is the RCU core waiting for a quiescent state from this CPU? */
3251 if (rcu_scheduler_fully_active &&
3252 rdp->qs_pending && !rdp->passed_quiesce &&
3253 rdp->rcu_qs_ctr_snap == __this_cpu_read(rcu_qs_ctr)) {
3254 rdp->n_rp_qs_pending++;
3255 } else if (rdp->qs_pending &&
3256 (rdp->passed_quiesce ||
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811230f1 in __rcu_pending+74 at kernel/rcu/tree.c:324
__rcu_pending (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144
3247 if (rcu_nohz_full_cpu(rsp))
>>>

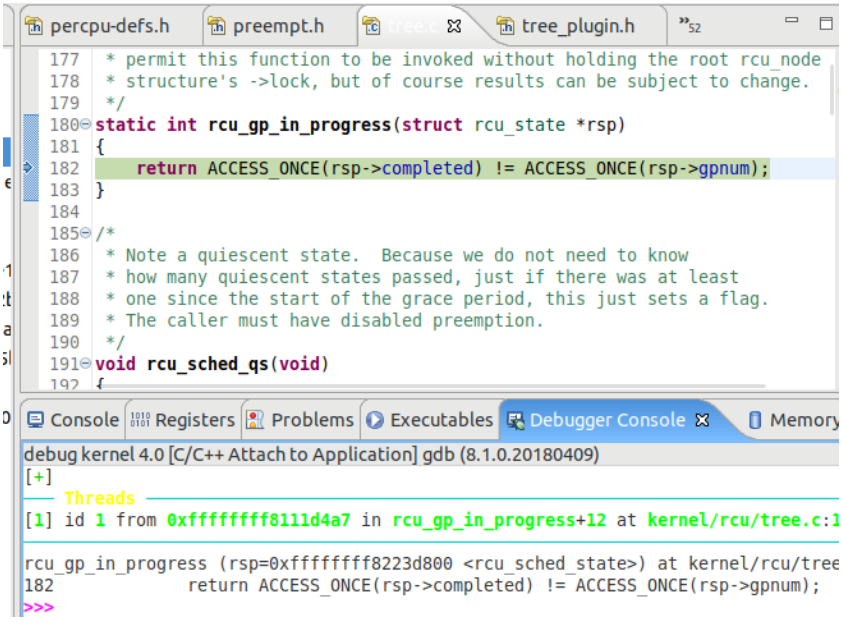

percpu-defs.h
preempt.h
tree.c
3051 if (tick_nohz_full_cpu(smp_processor_id()) &&
3052 (!rcu_gp_in_progress(rsp) ||
3053 ULONG_CMP_LT(jiffies, ACCESS_ONCE(rsp->gp_start) + HZ)))
3054 return 1;
3055 #endif /* #ifdef CONFIG_NO_HZ_FULL */
3056 return 0;
3057 }
3058
3059 /*
3060 * Bind the grace-period kthread for the sysidle flavor of RCU to the
3061 * timekeeping CPU.
3062 */
3063 static void rcu_bind_gp_kthread(void)
3064 {
3065 int __maybe_unused cpu;
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81124725 in rcu_nohz_full_cpu+12 at kernel/rcu/tree_plu
rcu_nohz_full_cpu (rsp=0xffffffff8223d800 <rcu_sched_state>) at kernel/rcu/tree
3056 return 0;
>>>

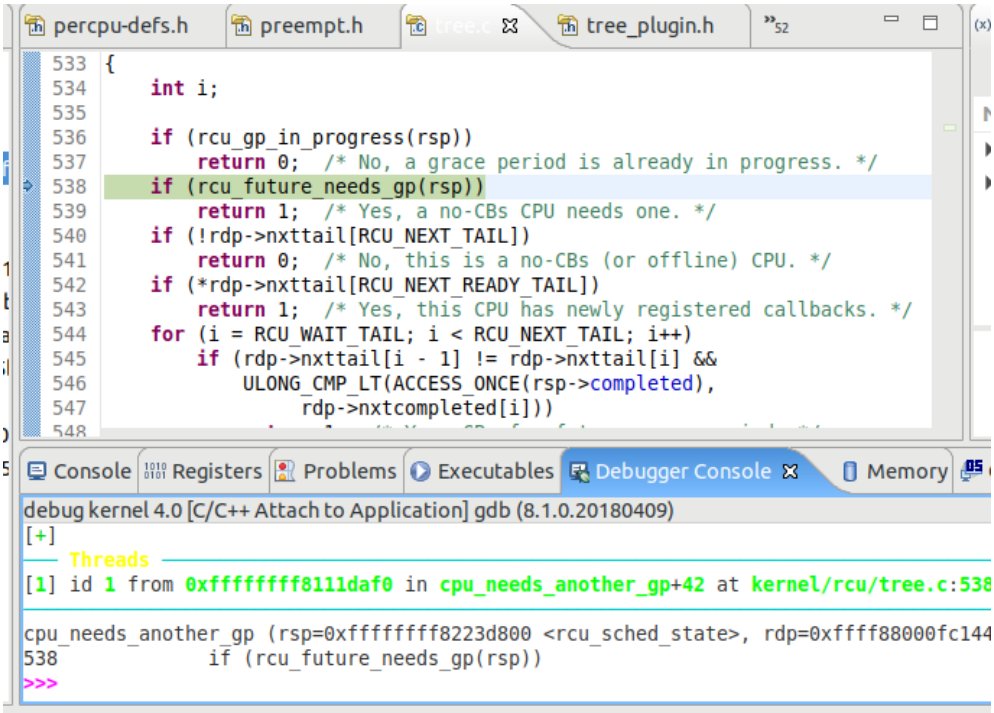

percpu-defs.h
preempt.h
tree_plugin.h
3242
3243 /* Check for CPU stalls, if enabled. */
3244 check_cpu_stall(rsp, rdp);
3245
3246 /* Is this CPU a NO_HZ_FULL CPU that should ignore RCU? */
3247 if (rcu_nohz_full_cpu(rsp))
3248 return 0;
3249
3250 /* Is the RCU core waiting for a quiescent state from this CPU? */
3251 if (rcu_scheduler_fully_active &&
3252 rdp->qs_pending && !rdp->passed_quiesce &&
3253 rdp->rcu_qs_ctr_snap == __this_cpu_read(rcu_qs_ctr)) {
3254 rdp->n_rp_qs_pending++;
3255 } else if (rdp->qs_pending &&
3256 (rdp->passed_quiesce ||
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8112310b in __rcu_pending+100 at kernel/rcu/tree.c:3251
__rcu_pending (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144c0) at
3251 if (rcu_scheduler_fully_active &&
>>>

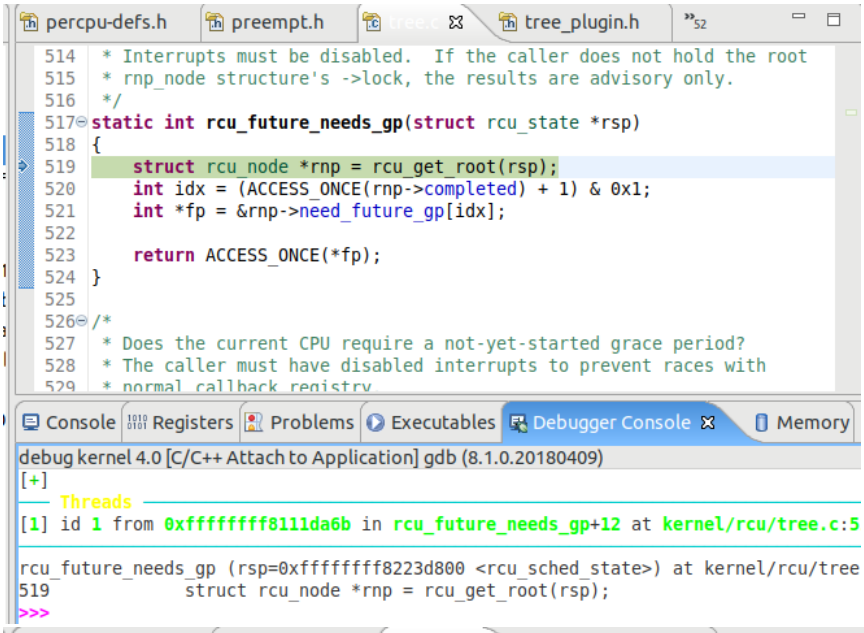

percpu-defs.h
preempt.h
tree_plugin.h
495 * Does the CPU have callbacks ready to be invoked?
496 */
497 static int
498 cpu_has_callbacks_ready_to_invoke(struct rcu_data *rdp)
499 {
500 return &rdp->nxtlist != rdp->nxttail[RCU_DONE_TAIL] &&
501 rdp->nxttail[RCU_DONE_TAIL] != NULL;
502 }
503
504 /*
505 * Return the root node of the specified rcu_state structure.
506 */
507 static struct rcu_node *rcu_get_root(struct rcu_state *rsp)
508 {
509 return &rsp->node[0];
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111da1d in cpu_has_callbacks_ready_to_invoke+12 at kernel/
cpu_has_callbacks_ready_to_invoke (rdp=0xffff88000fc144c0) at kernel/rcu/tree.c:500
500 return &rdp->nxtlist != rdp->nxttail[RCU_DONE_TAIL] &&
>>>

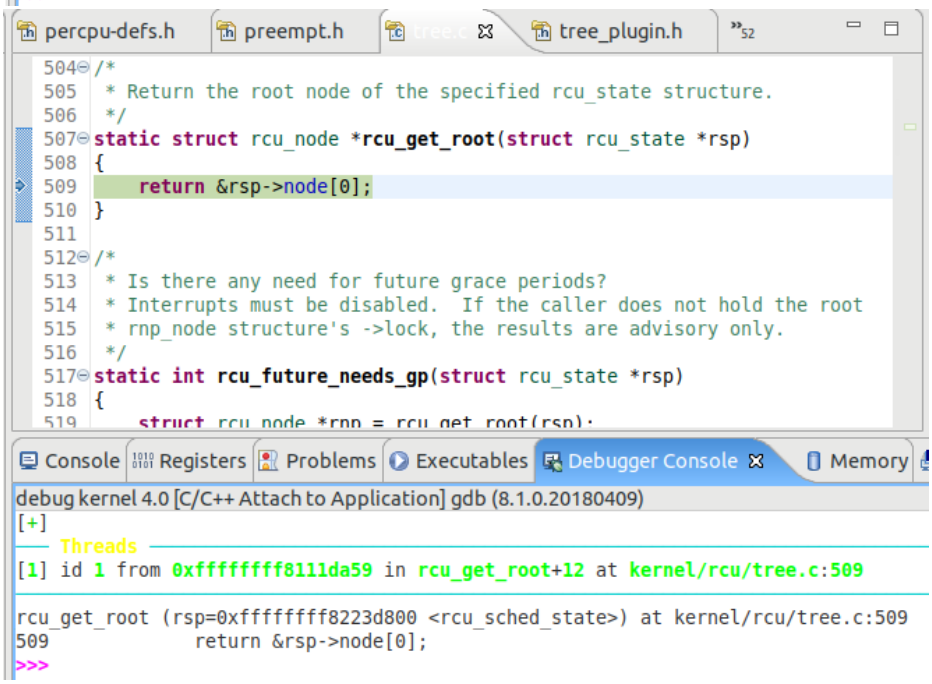

percpu-defs.h
preempt.h
tree_plugin.h
3264 rdp->n_rp_cb_ready++;
3265 return 1;
3266 }
3267
3268 /* Has RCU gone idle with this CPU needing another grace period? */
3269 if (cpu_needs_another_gp(rsp, rdp)) {
3270 rdp->n_rp_cpu_needs_gp++;
3271 return 1;
3272 }
3273
3274 /* Has another RCU grace period completed? */
3275 if (ACCESS_ONCE(rnp->completed) != rdp->completed) { /* outside lock
3276 rdp->n_rp_gp_completed++;
3277 return 1;
3278 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81123232 in __rcu_pending+395 at kernel/rcu/tree.c:3269
__rcu_pending (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144c0) at
3269 if (cpu_needs_another_gp(rsp, rdp)) {
>>>

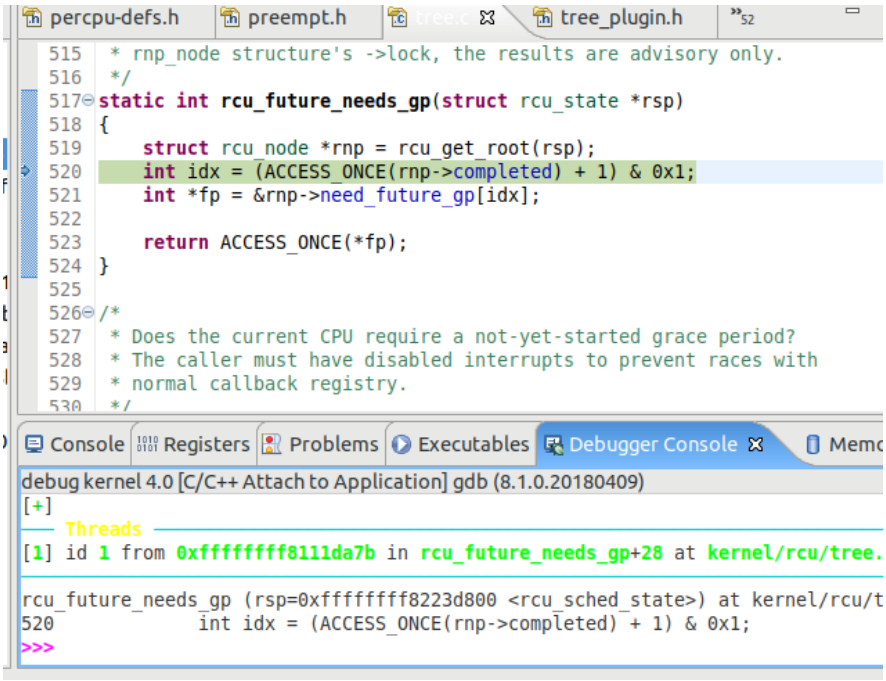

percpu-defs.h
preempt.h
tree_plugin.h
531 static int
532 cpu_needs_another_gp(struct rcu_state *rsp, struct rcu_data *rdp)
533 {
534 int i;
535
536 if (rcu_gp_in_progress(rsp))
537 return 0; /* No, a grace period is already in progress. */
538 if (rcu_future_needs_gp(rsp))
539 return 1; /* Yes, a no-CBs CPU needs one. */
540 if (!rdp->nxttail[RCU_NEXT_TAIL])
541 return 0; /* No, this is a no-CBs (or offline) CPU. */
542 if (*rdp->nxttail[RCU_NEXT_READY_TAIL])
543 return 1; /* Yes, this CPU has newly registered callbacks. */
544 for (i = RCU_WAIT_TAIL; i < RCU_NEXT_TAIL; i++)
545 if (rdp->nxttail[i - 1] != rdp->nxttail[i] &&
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111dad6 in cpu_needs_another_gp+16 at kernel/rcu/tree.c
cpu_needs_another_gp (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000f
536 if (rcu_gp_in_progress(rsp))
>>>

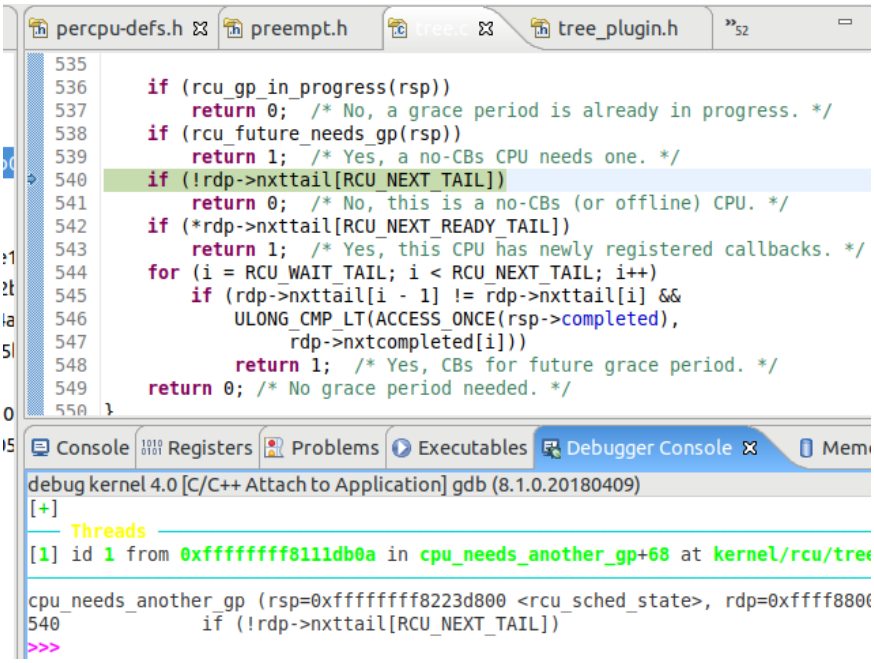

percpu-defs.h
preempt.h
tree_plugin.h
177 * permit this function to be invoked without holding the root rcu_node
178 * structure's ->lock, but of course results can be subject to change.
179 */
180 static int rcu_gp_in_progress(struct rcu_state *rsp)
181 {
182 return ACCESS_ONCE(rsp->completed) != ACCESS_ONCE(rsp->gpnum);
183 }
184
185 /*
186 * Note a quiescent state. Because we do not need to know
187 * how many quiescent states passed, just if there was at least
188 * one since the start of the grace period, this just sets a flag.
189 * The caller must have disabled preemption.
190 */
191 void rcu_sched_qs(void)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d4a7 in rcu_gp_in_progress+12 at kernel/rcu/tree.c:1
rcu_gp_in_progress (rsp=0xffffffff8223d800 <rcu_sched_state>) at kernel/rcu/tree
182 return ACCESS_ONCE(rsp->completed) != ACCESS_ONCE(rsp->gpnum);
>>>

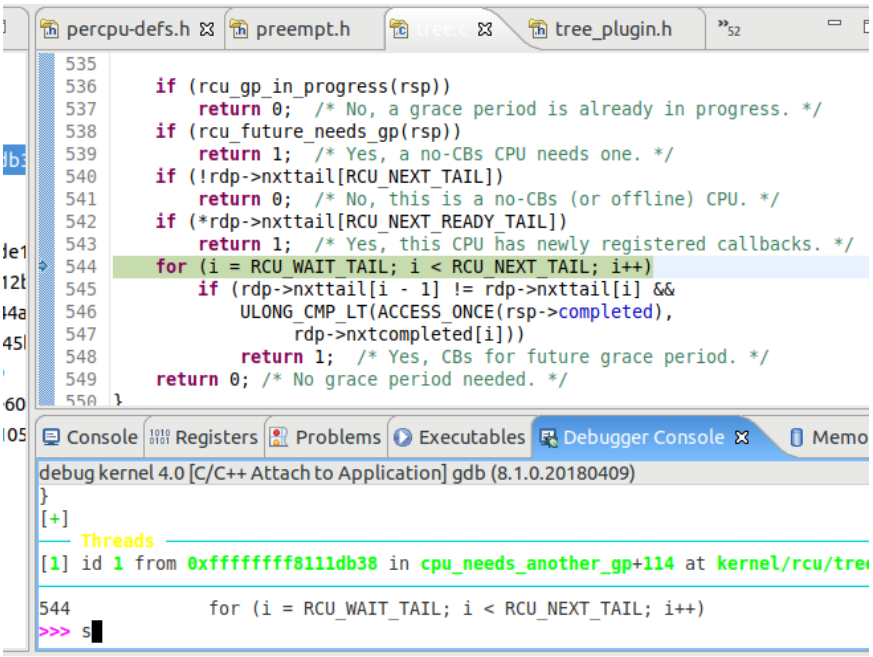

percpu-defs.h
preempt.h
tree_plugin.h
533 {
534 int i;
535
536 if (rcu_gp_in_progress(rsp))
537 return 0; /* No, a grace period is already in progress. */
538 if (rcu_future_needs_gp(rsp))
539 return 1; /* Yes, a no-CBs CPU needs one. */
540 if (!rdp->nxttail[RCU_NEXT_TAIL])
541 return 0; /* No, this is a no-CBs (or offline) CPU. */
542 if (*rdp->nxttail[RCU_NEXT_READY_TAIL])
543 return 1; /* Yes, this CPU has newly registered callbacks. */
544 for (i = RCU_WAIT_TAIL; i < RCU_NEXT_TAIL; i++)
545 if (rdp->nxttail[i - 1] != rdp->nxttail[i] &&
546 ULONG_CMP_LT(ACCESS_ONCE(rsp->completed),
547 rdp->nxtcompleted[i]))
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111daf0 in cpu_needs_another_gp+42 at kernel/rcu/tree.c:538
cpu_needs_another_gp (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144
538 if (rcu_future_needs_gp(rsp))
>>>

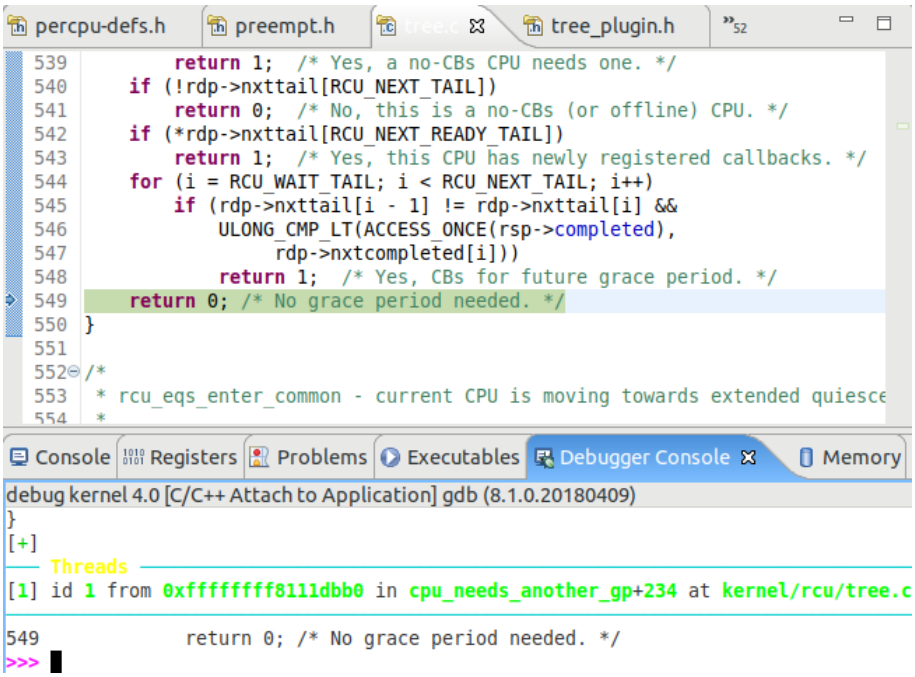

percpu-defs.h
preempt.h
tree_plugin.h
514 * Interrupts must be disabled. If the caller does not hold the root
515 * rnp_node structure's ->lock, the results are advisory only.
516 */
517 static int rcu_future_needs_gp(struct rcu_state *rsp)
518 {
519 struct rcu_node *rnp = rcu_get_root(rsp);
520 int idx = (ACCESS_ONCE(rnp->completed) + 1) & 0x1;
521 int *fp = &rnp->need_future_gp[idx];
522
523 return ACCESS_ONCE(*fp);
524 }
525
526 /*
527 * Does the current CPU require a not-yet-started grace period?
528 * The caller must have disabled interrupts to prevent races with
529 * normal callback registry
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111da6b in rcu_future_needs_gp+12 at kernel/rcu/tree.c:5
rcu_future_needs_gp (rsp=0xffffffff8223d800 <rcu_sched_state>) at kernel/rcu/tree
519 struct rcu_node *rnp = rcu_get_root(rsp);
>>>

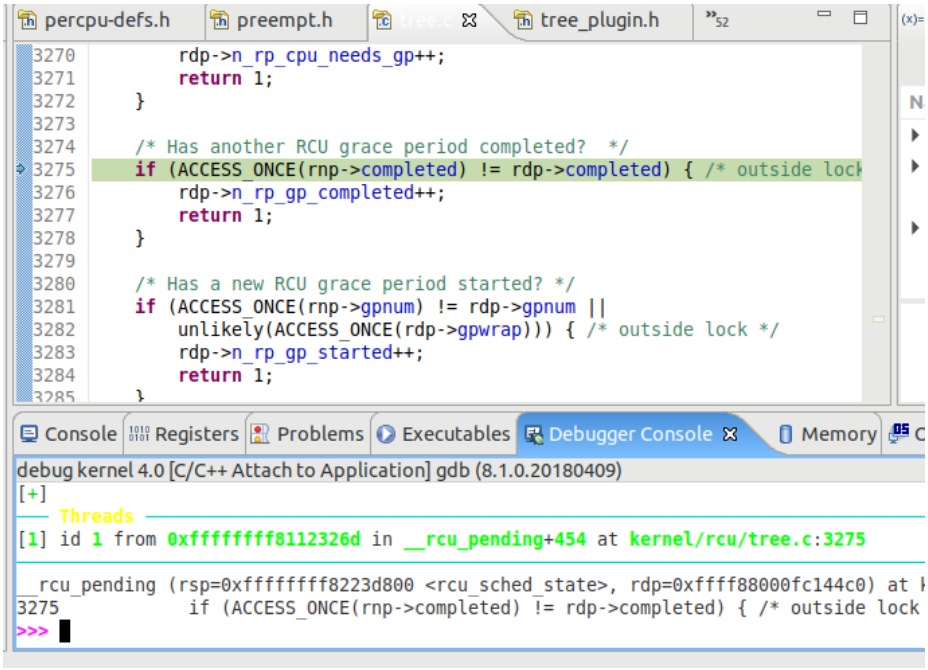

percpu-defs.h
preempt.h
tree_plugin.h
504 /*
505 * Return the root node of the specified rcu_state structure.
506 */
507 static struct rcu_node *rcu_get_root(struct rcu_state *rsp)
508 {
509 return &rsp->node[0];
510 }
511
512 /*
513 * Is there any need for future grace periods?
514 * Interrupts must be disabled. If the caller does not hold the root
515 * rnp_node structure's ->lock, the results are advisory only.
516 */
517 static int rcu_future_needs_gp(struct rcu_state *rsp)
518 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111da59 in rcu_get_root+12 at kernel/rcu/tree.c:509
rcu_get_root (rsp=0xffffffff8223d800 <rcu_sched_state>) at kernel/rcu/tree.c:509
509 return &rsp->node[0];
>>>

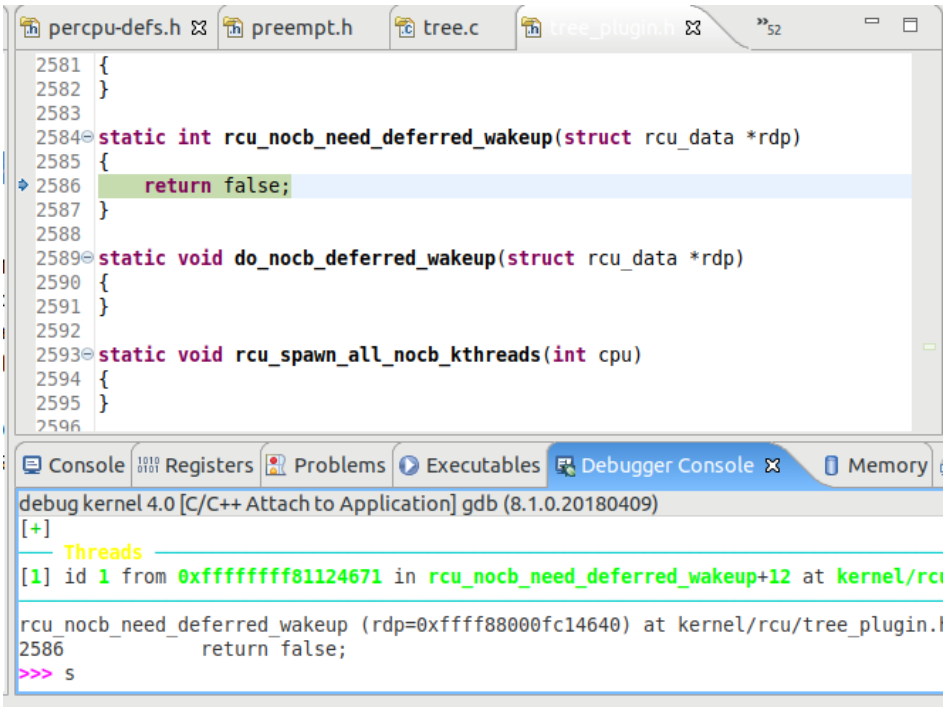

percpu-defs.h
preempt.h
tree_plugin.h
515 * rnp_node structure's ->lock, the results are advisory only.
516 */
517 static int rcu_future_needs_gp(struct rcu_state *rsp)
518 {
519 struct rcu_node *rnp = rcu_get_root(rsp);
520 int idx = (ACCESS_ONCE(rnp->completed) + 1) & 0x1;
521 int *fp = &rnp->need_future_gp[idx];
522
523 return ACCESS_ONCE(*fp);
524 }
525
526 /*
527 * Does the current CPU require a not-yet-started grace period?
528 * The caller must have disabled interrupts to prevent races with
529 * normal callback registry.
530 */
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111da7b in rcu_future_needs_gp+28 at kernel/rcu/tree.
rcu_future_needs_gp (rsp=0xffffffff8223d800 <rcu_sched_state>) at kernel/rcu/t
520 int idx = (ACCESS_ONCE(rnp->completed) + 1) & 0x1;
>>>

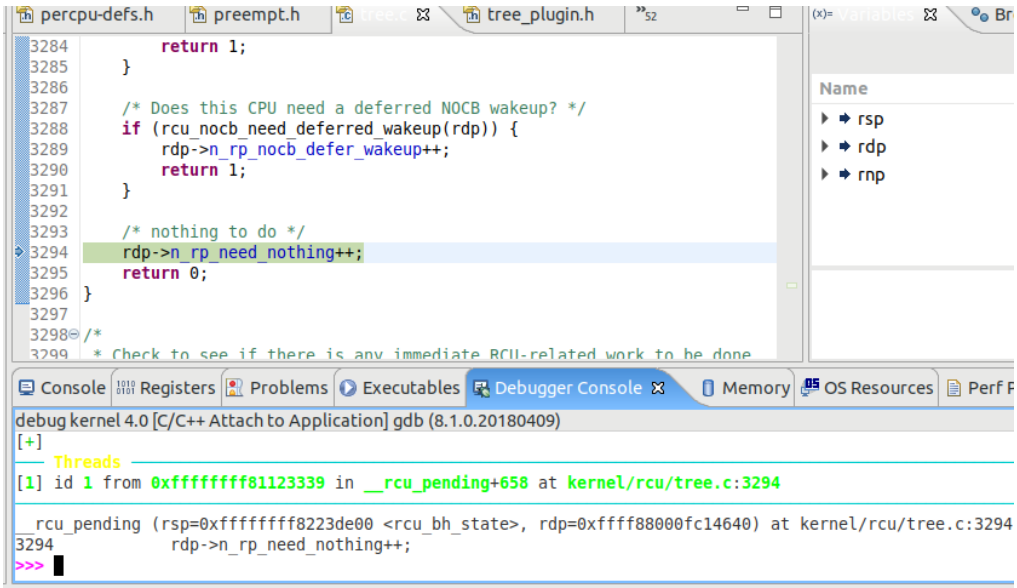

percpu-defs.h
preempt.h
tree_plugin.h
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff8111db0a in cpu_needs_another_gp+68 at kernel/rcu/tree
cpu_needs_another_gp (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff8800
540 if (!rdp->nxttail[RCU_NEXT_TAIL])
>>>

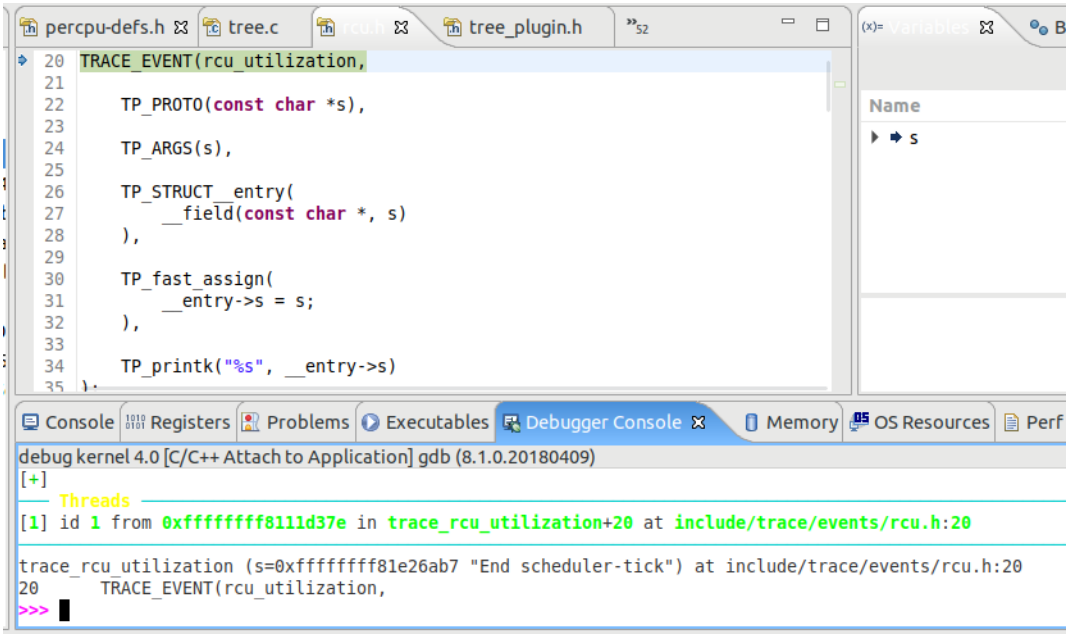

percpu-defs.h
preempt.h
tree_plugin.h
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff8111db38 in cpu_needs_another_gp+114 at kernel/rcu/tree
544 for (i = RCU_WAIT_TAIL; i < RCU_NEXT_TAIL; i++)
>>> s

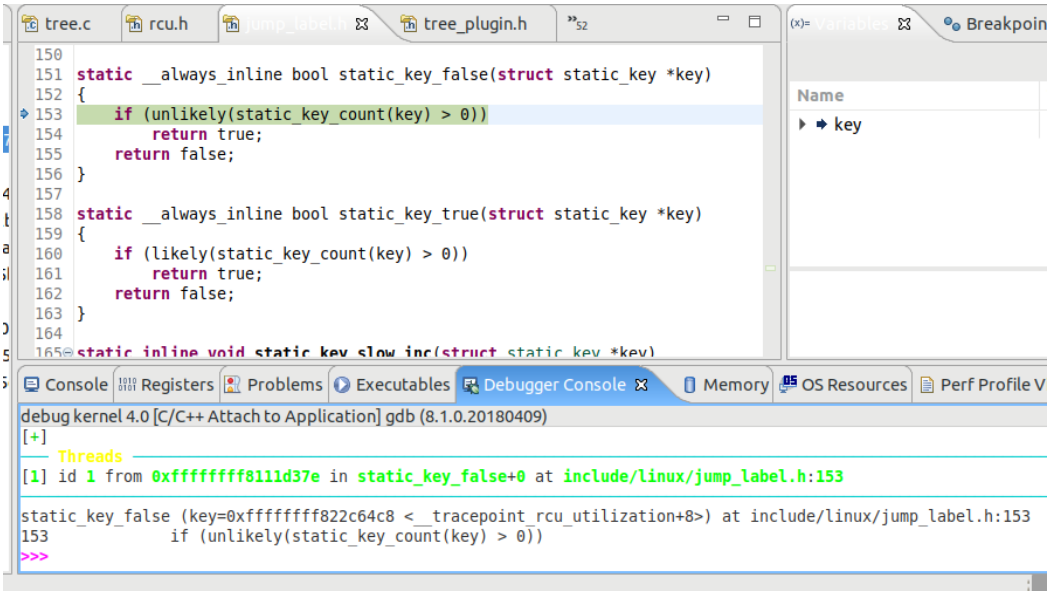

percpu-defs.h
preempt.h
tree_plugin.h
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff8111dbb0 in cpu_needs_another_gp+234 at kernel/rcu/tree.c
549 return 0; /* No grace period needed. */
>>>


percpu-defs.h
preempt.h
tree_plugin.h
3270 rdp->n_rp_cpu_needs_gp++;
3271 return 1;
3272 }
3273
3274 /* Has another RCU grace period completed? */
3275 if (ACCESS_ONCE(rnp->completed) != rdp->completed) { /* outside lock
3276 rdp->n_rp_gp_completed++;
3277 return 1;
3278 }
3279
3280 /* Has a new RCU grace period started? */
3281 if (ACCESS_ONCE(rnp->gpnum) != rdp->gpnum ||
3282 unlikely(ACCESS_ONCE(rdp->gpwrap))) { /* outside lock */
3283 rdp->n_rp_gp_started++;
3284 return 1;
3285 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8112326d in __rcu_pending+454 at kernel/rcu/tree.c:3275
__rcu_pending (rsp=0xffffffff8223d800 <rcu_sched_state>, rdp=0xffff88000fc144c0) at
3275 if (ACCESS_ONCE(rnp->completed) != rdp->completed) { /* outside lock
>>>

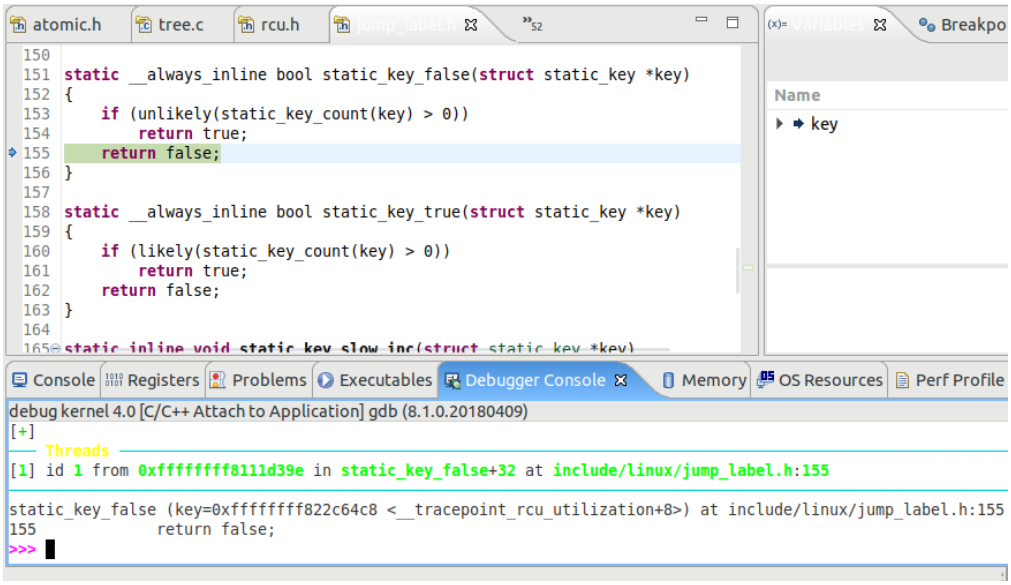

percpu-defs.h
preempt.h
tree.c
2581 {
2582 }
2583
2584 static int rcu_nocb_need_deferred_wakeup(struct rcu_data *rdp)
2585 {
2586 return false;
2587 }
2588
2589 static void do_nocb_deferred_wakeup(struct rcu_data *rdp)
2590 {
2591 }
2592
2593 static void rcu_spawn_all_nocb_kthreads(int cpu)
2594 {
2595 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81124671 in rcu_nocb_need_deferred_wakeup+12 at kernel/rcu
rcu_nocb_need_deferred_wakeup (rdp=0xffff88000fc14640) at kernel/rcu/tree_plugin.h
2586 return false;
>>> s


percpu-defs.h
preempt.h
tree_plugin.h
3284 return 1;
3285 }
3286
3287 /* Does this CPU need a deferred NOCB wakeup? */
3288 if (rcu_nocb_need_deferred_wakeup(rdp)) {
3289 rdp->n_rp_nocb_defer_wakeup++;
3290 return 1;
3291 }
3292
3293 /* nothing to do */
3294 rdp->n_rp_need_nothing++;
3295 return 0;
3296 }
3297
3298 /*
Name
rsp
rdp
rnp
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81123339 in __rcu_pending+658 at kernel/rcu/tree.c:3294
__rcu_pending (rsp=0xffffffff8223de00 <rcu_bh_state>, rdp=0xffff88000fc14640) at kernel/rcu/tree.c:3294
3294 rdp->n_rp_need_nothing++;
>>>

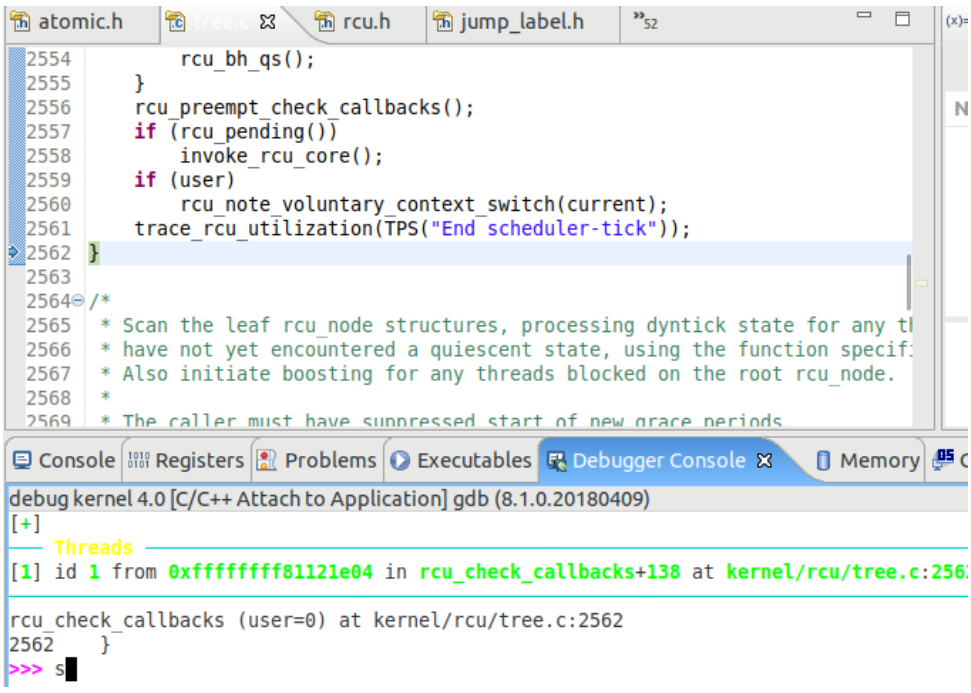

percpu-defs.h
tree.c
tree_plugin.h
20 TRACE_EVENT(rcu_utilization,
22 TP_PROTO(const char *s),
24 TP_ARGS(s),
26 TP_STRUCT__entry(
27 __field(const char *, s)
28),
30 TP_fast_assign(
31 __entry->s = s;
32),
34 TP_printk("%s", __entry->s)
Name
s
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d37e in trace_rcu_utilization+20 at include/trace/events/rcu.h:20
trace_rcu_utilization (s=0xffffffff81e26ab7 "End scheduler-tick") at include/trace/events/rcu.h:20
20 TRACE_EVENT(rcu_utilization,
>>>

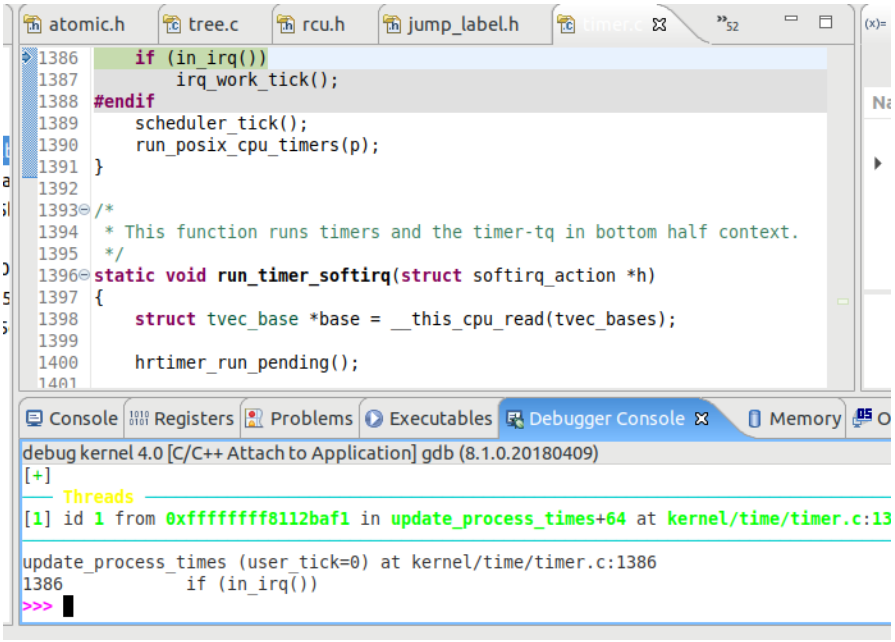

tree.c
rcu.h
tree_plugin.h
150
151 static __always_inline bool static_key_false(struct static_key *key)
152 {
153 if (unlikely(static_key_count(key) > 0))
154 return true;
155 return false;
156 }
157
158 static __always_inline bool static_key_true(struct static_key *key)
159 {
160 if (likely(static_key_count(key) > 0))
161 return true;
162 return false;
163 }
164
Name
key
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf Profile V
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d37e in static_key_false+0 at include/linux/jump_label.h:153
static_key_false (key=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at include/linux/jump_label.h:153
153 if (unlikely(static_key_count(key) > 0))
>>>

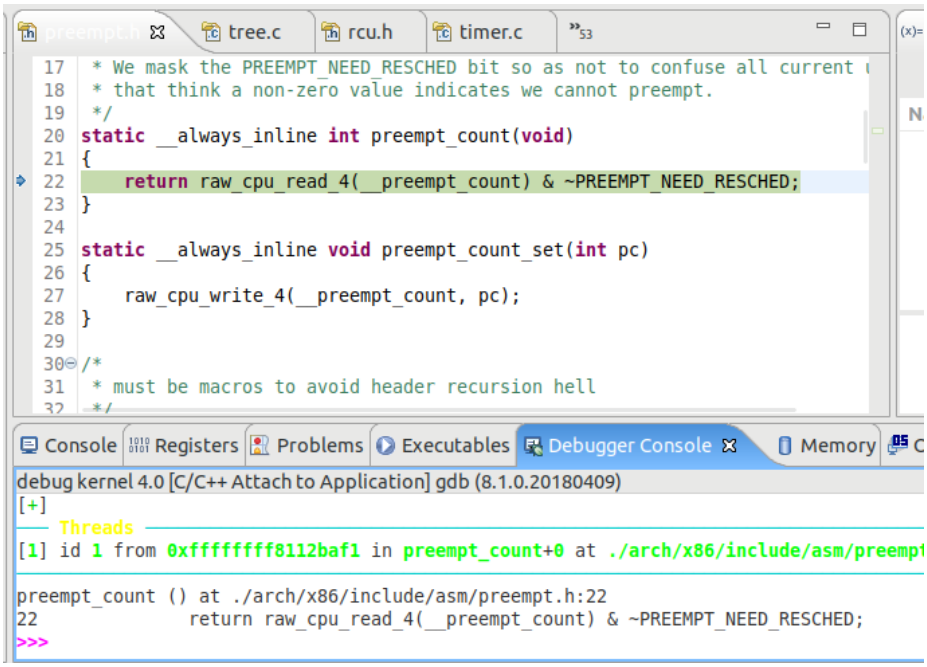

tree.c
rcu.h
jump_label.h
27 return ACCESS_ONCE((v)->counter);
28 }
29
30 /**
31 * atomic_set - set atomic variable
32 * @v: pointer of type atomic_t
33 * @i: required value
34 *
35 * Atomically sets the value of @v to @i.
36 */
37 static inline void atomic_set(atomic_t *v, int i)
38 {
39 v->counter = i;
40 }
41
Name
v
__var
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf Profi
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111cf4e in atomic_read+12 at ./arch/x86/include/asm/atomic.h:27
atomic_read (v=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at ./arch/x86/include/asm/atomic.h:27
27 return ACCESS_ONCE((v)->counter);
>>>

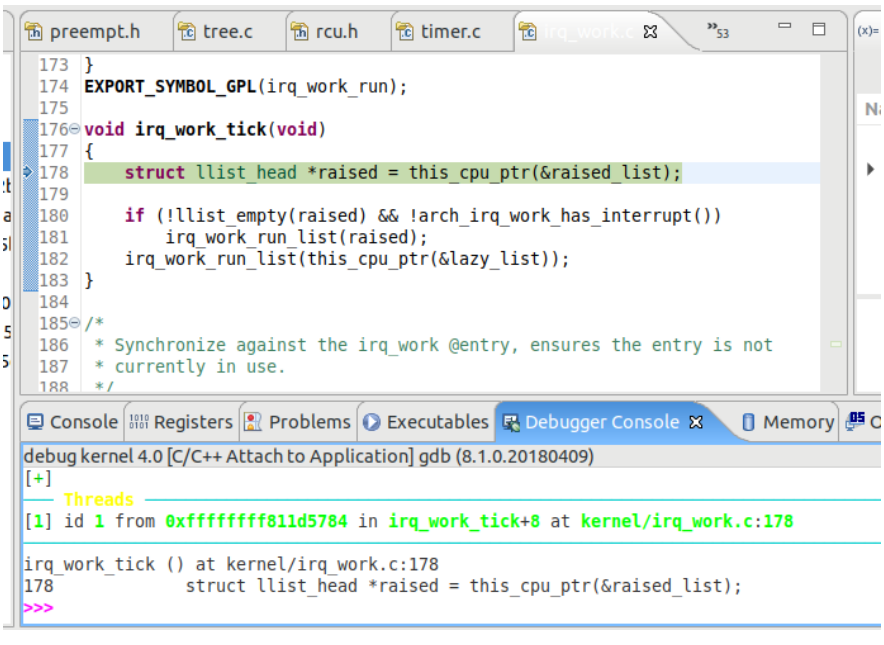

atomic.h
tree.c
rcu.h
jump_label.h
Variables
Breakpo
150
151 static __always_inline bool static_key_false(struct static_key *key)
152 {
153 if (unlikely(static_key_count(key) > 0))
154 return true;
155 return false;
156 }
157
158 static __always_inline bool static_key_true(struct static_key *key)
159 {
160 if (likely(static_key_count(key) > 0))
161 return true;
162 return false;
163 }
164
Name
key
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf Profile
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d39e in static_key_false+32 at include/linux/jump_label.h:155
static_key_false (key=0xffffffff822c64c8 <__tracepoint_rcu_utilization+8>) at include/linux/jump_label.h:155
155 return false;
>>>

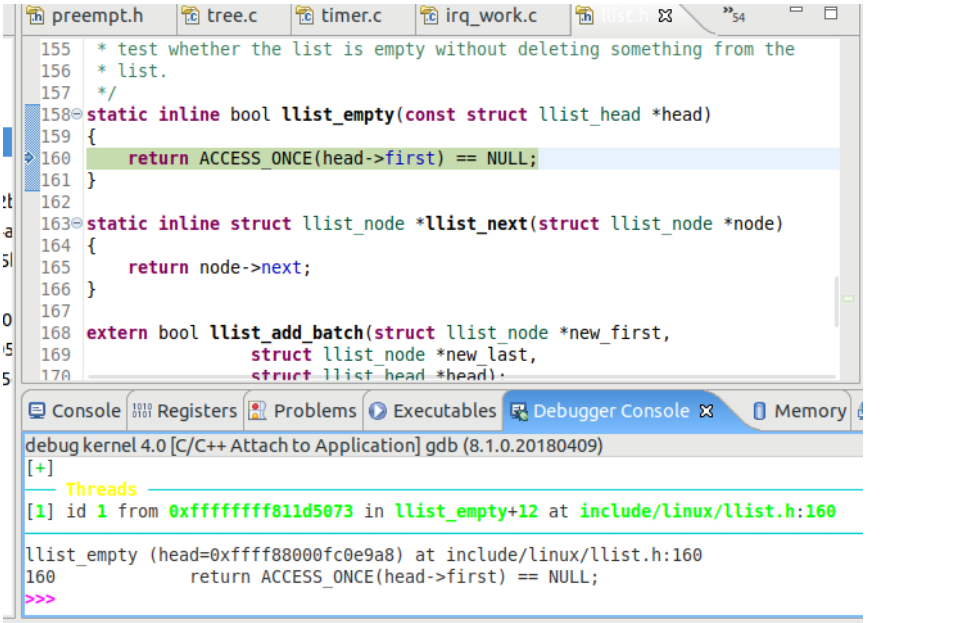

atomic.h
tree.c
jump_label.h
Variables
Br
20 TRACE_EVENT(rcu_utilization,
21
22 TP_PROTO(const char *s),
23
24 TP_ARGS(s),
25
26 TP_STRUCT__entry(
27 __field(const char *, s)
28),
29
30 TP_fast_assign(
31 __entry->s = s;
32),
33
34 TP_printk("%s", __entry->s)
Name
s
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8111d3a3 in trace_rcu_utilization+57 at include/trace/events/rcu.h:20
trace_rcu_utilization (s=0xffffffff81e26ab7 "End scheduler-tick") at include/trace/events/rcu.h:20
20 TRACE_EVENT(rcu_utilization,
>>>

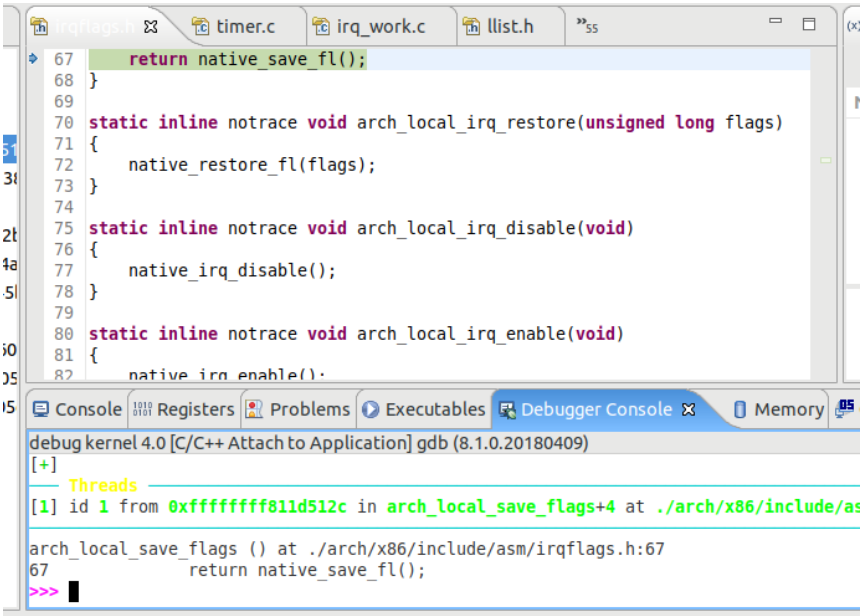

atomic.h
rcu.h
jump_label.h
2554 rcu_bh_qs();
2555 }
2556 rcu_preempt_check_callbacks();
2557 if (rcu_pending())
2558 invoke_rcu_core();
2559 if (user)
2560 rcu_note_voluntary_context_switch(current);
2561 trace_rcu_utilization(TPS("End scheduler-tick"));
2562 }
2563
2564 /*
2565 * Scan the leaf rcu_node structures, processing dyntick state for any tl
2566 * have not yet encountered a quiescent state, using the function specif
2567 * Also initiate boosting for any threads blocked on the root rcu_node.
2568 *
2569 * The caller must have suppressed start of new grace periods
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81121e04 in rcu_check_callbacks+138 at kernel/rcu/tree.c:256
rcu_check_callbacks (user=0) at kernel/rcu/tree.c:2562
2562 }
>>> s

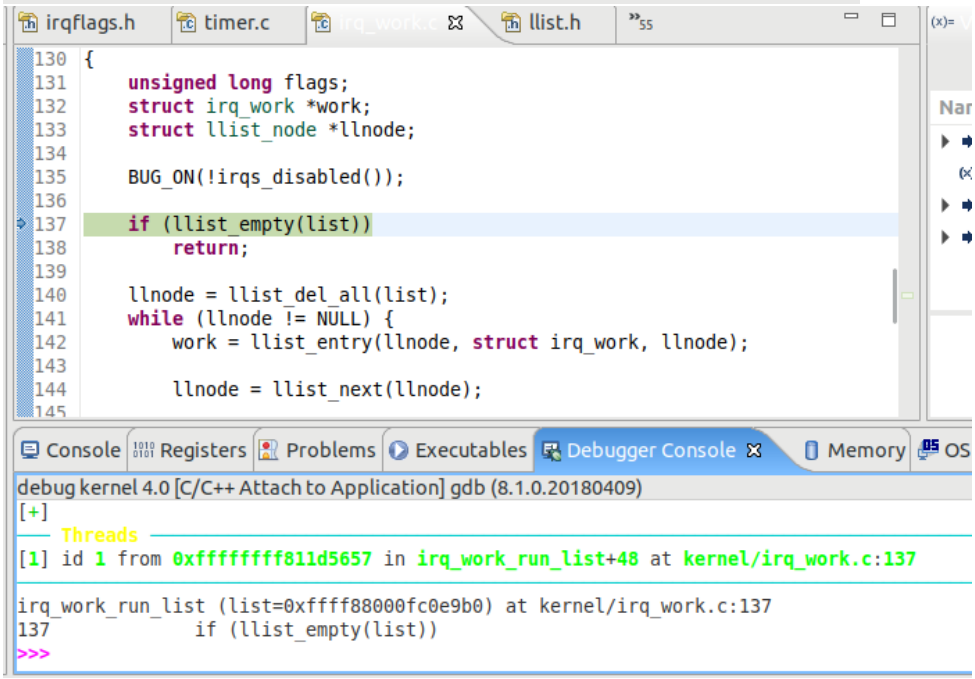

atomic.h
tree.c
rcu.h
jump_label.h
1386 if (in_irq())
1387 irq_work_tick();
1388 #endif
1389 scheduler_tick();
1390 run_posix_cpu_timers(p);
1391 }
1392
1393 /*
1394 * This function runs timers and the timer-tq in bottom half context.
1395 */
1396 static void run_timer_softirq(struct softirq_action *h)
1397 {
1398 struct tvec_base *base = __this_cpu_read(tvec_bases);
1399
1400 hrtimer_run_pending();
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8112baf1 in update_process_times+64 at kernel/time/timer.c:13
update_process_times (user_tick=0) at kernel/time/timer.c:1386
1386 if (in_irq())
>>>

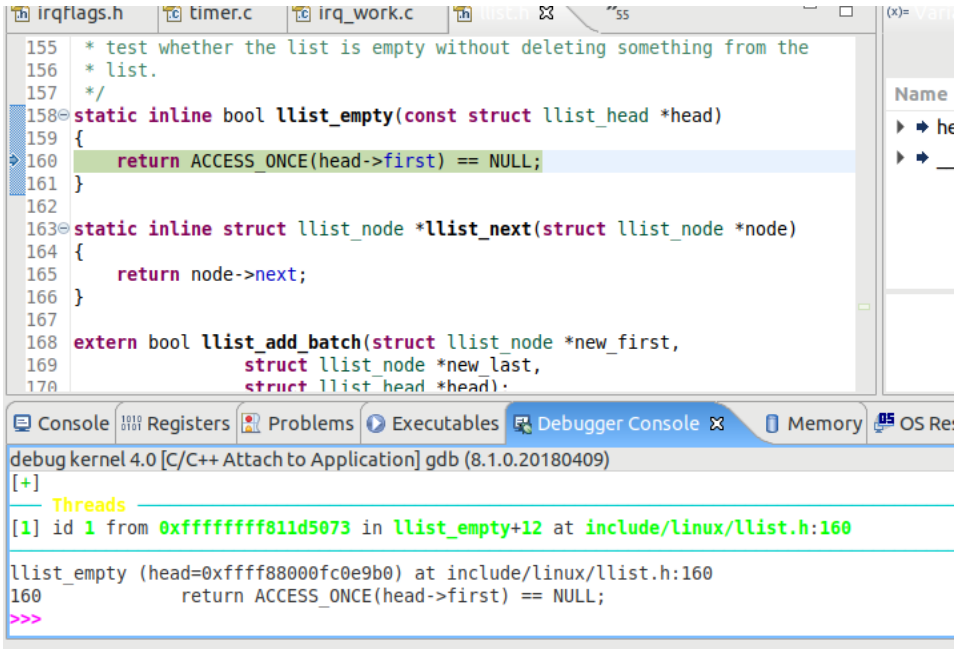

tree.c
rcu.h
timer.c
17 * We mask the PREEMPT_NEED_RESCHED bit so as not to confuse all current
18 * that think a non-zero value indicates we cannot preempt.
19 */
20 static __always_inline int preempt_count(void)
21 {
22 return raw_cpu_read_4(__preempt_count) & ~PREEMPT_NEED_RESCHED;
23 }
24
25 static __always_inline void preempt_count_set(int pc)
26 {
27 raw_cpu_write_4(__preempt_count, pc);
28 }
29
30 /*
31 * must be macros to avoid header recursion hell
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8112baf1 in preempt_count+0 at ./arch/x86/include/asm/preempt
preempt_count () at ./arch/x86/include/asm/preempt.h:22
22 return raw_cpu_read_4(__preempt_count) & ~PREEMPT_NEED_RESCHED;
>>>

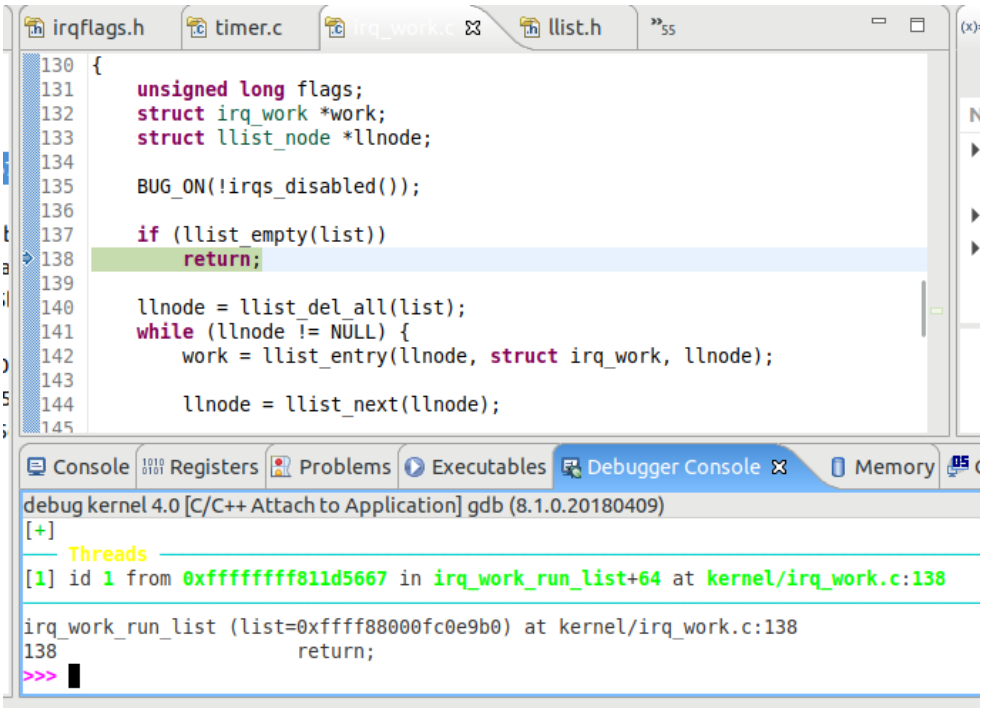

preempt.h
tree.c
rcu.h
timer.c
173 }
174 EXPORT_SYMBOL_GPL(irq_work_run);
175
176 void irq_work_tick(void)
177 {
178 struct llist_head *raised = this_cpu_ptr(&raised_list);
179
180 if (!llist_empty(raised) && !arch_irq_work_has_interrupt())
181 irq_work_run_list(raised);
182 irq_work_run_list(this_cpu_ptr(&lazy_list));
183 }
184
185 /*
186 * Synchronize against the irq_work @entry, ensures the entry is not
187 * currently in use.
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811d5784 in irq_work_tick+8 at kernel/irq_work.c:178
irq_work_tick () at kernel/irq_work.c:178
178 struct llist_head *raised = this_cpu_ptr(&raised_list);
>>>

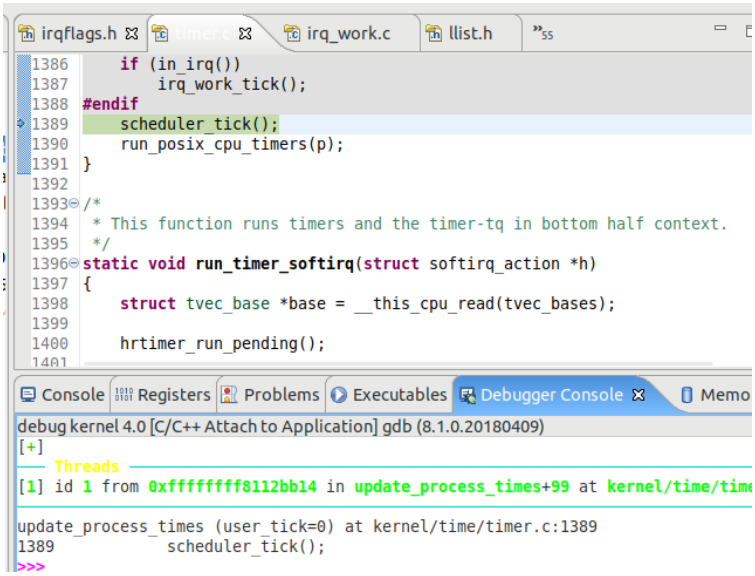

preempt.h
tree.c
timer.c
irq_work.c
155 * test whether the list is empty without deleting something from the
156 * list.
157 */
158 static inline bool llist_empty(const struct llist_head *head)
159 {
160 return ACCESS_ONCE(head->first) == NULL;
161 }
162
163 static inline struct llist_node *llist_next(struct llist_node *node)
164 {
165 return node->next;
166 }
167
168 extern bool llist_add_batch(struct llist_node *new_first,
169 struct llist_node *new_last,
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811d5073 in llist_empty+12 at include/linux/llist.h:160
llist_empty (head=0xffff88000fc0e9a8) at include/linux/llist.h:160
160 return ACCESS_ONCE(head->first) == NULL;
>>>

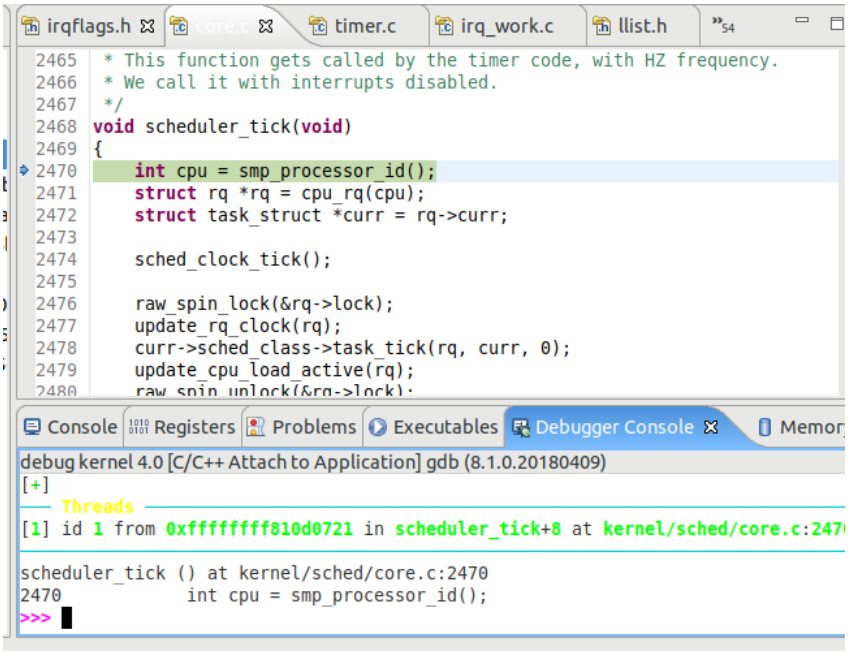

timer.c
irq_work.c
llist.h
67 return native_save_fl();
68 }
69
70 static inline notrace void arch_local_irq_restore(unsigned long flags)
71 {
72 native_restore_fl(flags);
73 }
74
75 static inline notrace void arch_local_irq_disable(void)
76 {
77 native_irq_disable();
78 }
79
80 static inline notrace void arch_local_irq_enable(void)
81 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811d512c in arch_local_save_flags+4 at ./arch/x86/include/as
arch_local_save_flags () at ./arch/x86/include/asm/irqflags.h:67
67 return native_save_fl();
>>>

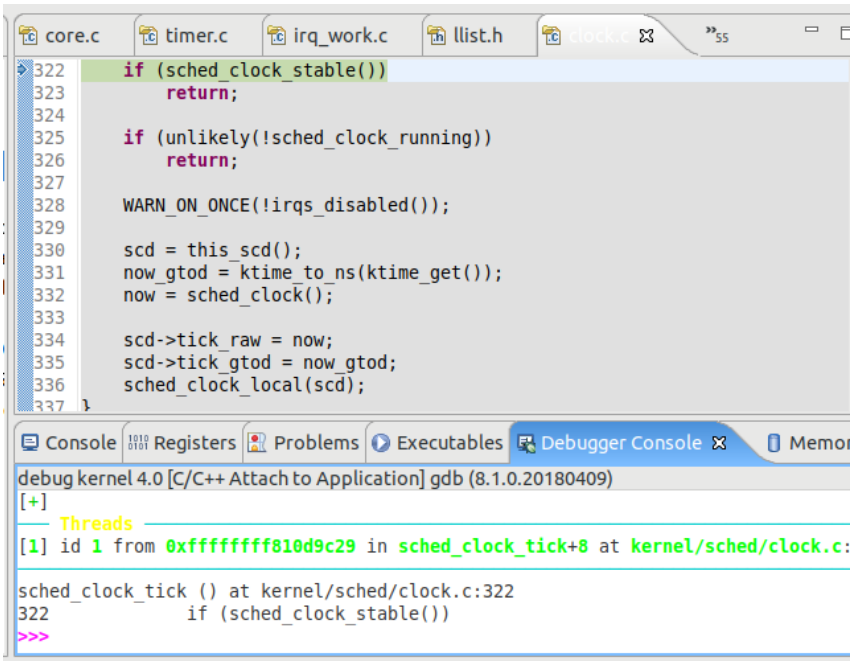

irqflags.h
timer.c
llist.h
130 {
131 unsigned long flags;
132 struct irq_work *work;
133 struct llist_node *llnode;
134
135 BUG_ON(!irqs_disabled());
136
137 if (llist_empty(list))
138 return;
139
140 llnode = llist_del_all(list);
141 while (llnode != NULL) {
142 work = llist_entry(llnode, struct irq_work, llnode);
143
144 llnode = llist_next(llnode);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811d5657 in irq_work_run_list+48 at kernel/irq_work.c:137
irq_work_run_list (list=0xffff88000fc0e9b0) at kernel/irq_work.c:137
137 if (llist_empty(list))
>>>

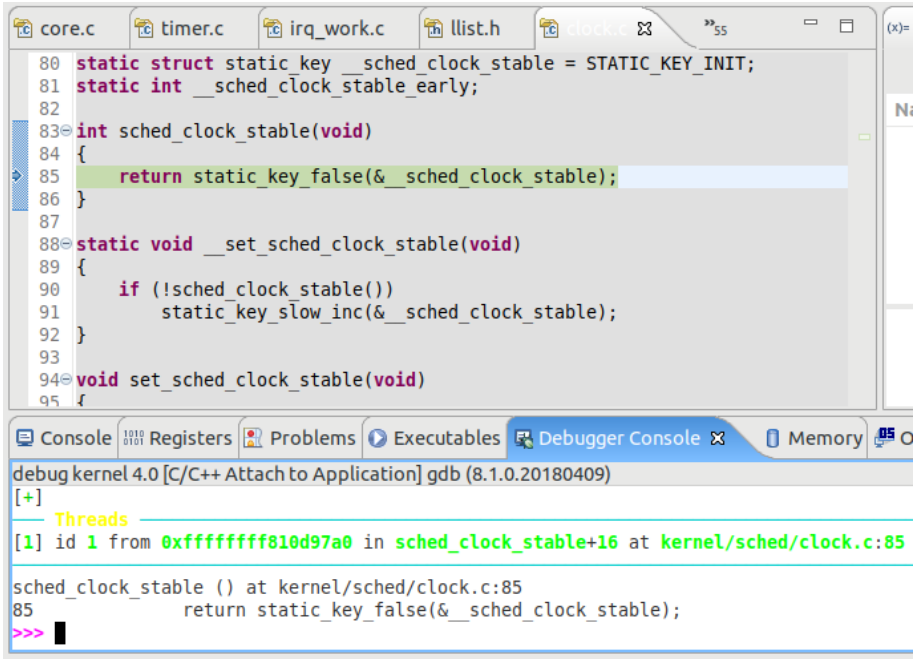

irqflags.h
timer.c
irq_work.c
155 * test whether the list is empty without deleting something from the
156 * list.
157 */
158 static inline bool llist_empty(const struct llist_head *head)
159 {
160 return ACCESS_ONCE(head->first) == NULL;
161 }
162
163 static inline struct llist_node *llist_next(struct llist_node *node)
164 {
165 return node->next;
166 }
167
168 extern bool llist_add_batch(struct llist_node *new_first,
169 struct llist_node *new_last,
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811d5073 in llist_empty+12 at include/linux/llist.h:160
llist_empty (head=0xffff88000fc0e9b0) at include/linux/llist.h:160
160 return ACCESS_ONCE(head->first) == NULL;
>>>

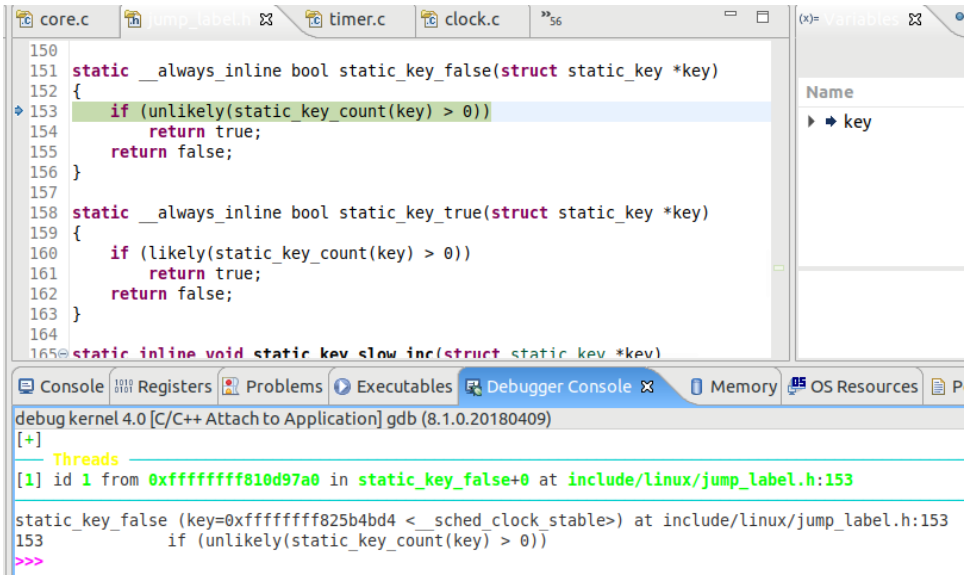

irqflags.h
timer.c
llist.h
130 {
131 unsigned long flags;
132 struct irq_work *work;
133 struct llist_node *llnode;
134
135 BUG_ON(!irqs_disabled());
136
137 if (llist_empty(list))
138 return;
139
140 llnode = llist_del_all(list);
141 while (llnode != NULL) {
142 work = llist_entry(llnode, struct irq_work, llnode);
143
144 llnode = llist_next(llnode);
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811d5667 in irq_work_run_list+64 at kernel/irq_work.c:138
irq_work_run_list (list=0xffff88000fc0e9b0) at kernel/irq_work.c:138
138 return;
>>>

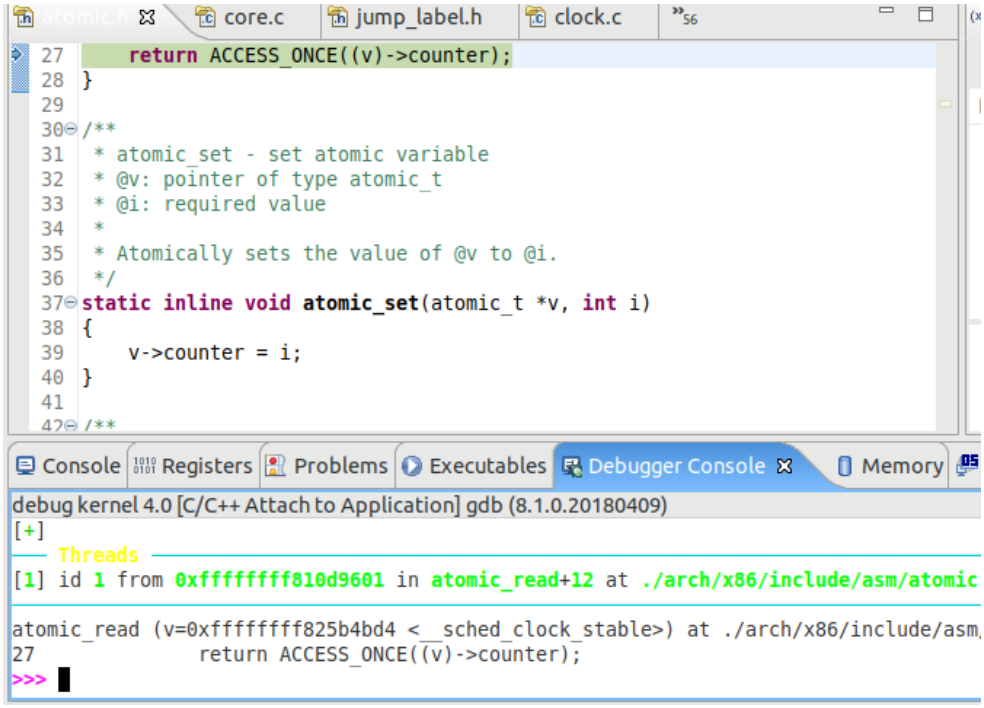

irqflags.h
irq_work.c
llist.h
1386 if (in_irq())
1387 irq_work_tick();
1388 #endif
1389 scheduler_tick();
1390 run_posix_cpu_timers(p);
1391 }
1392
1393 /*
1394 * This function runs timers and the timer-tq in bottom half context.
1395 */
1396 static void run_timer_softirq(struct softirq_action *h)
1397 {
1398 struct tvec_base *base = __this_cpu_read(tvec_bases);
1399
1400 hrtimer_run_pending();
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8112bb14 in update_process_times+99 at kernel/time/time
update_process_times (user_tick=0) at kernel/time/timer.c:1389
1389 scheduler_tick();
>>>

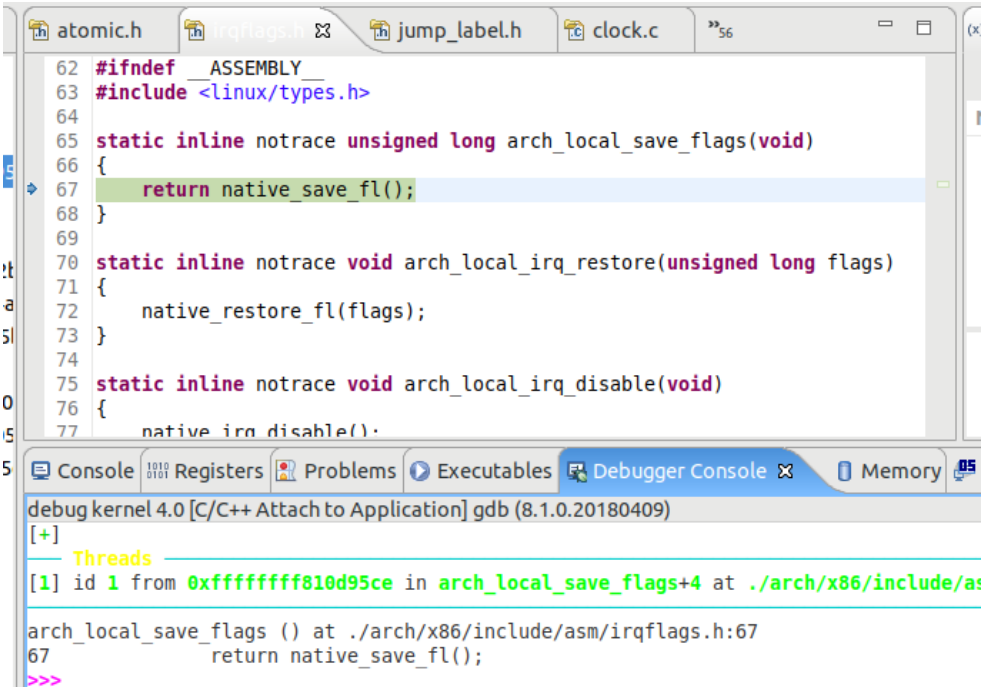

irqflags.h
timer.c
irq_work.c
llist.h
» 54
2465 * This function gets called by the timer code, with HZ frequency.
2466 * We call it with interrupts disabled.
2467 */
2468 void scheduler_tick(void)
2469 {
2470 int cpu = smp_processor_id();
2471 struct rq *rq = cpu_rq(cpu);
2472 struct task_struct *curr = rq->curr;
2473
2474 sched_clock_tick();
2475
2476 raw_spin_lock(&rq->lock);
2477 update_rq_clock(rq);
2478 curr->sched_class->task_tick(rq, curr, 0);
2479 update_cpu_load_active(rq);
Console
Registers
Problems
Executables
Debugger Console
Memor
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d0721 in scheduler_tick+8 at kernel/sched/core.c:247
scheduler_tick () at kernel/sched/core.c:2470
2470 int cpu = smp_processor_id();
>>>

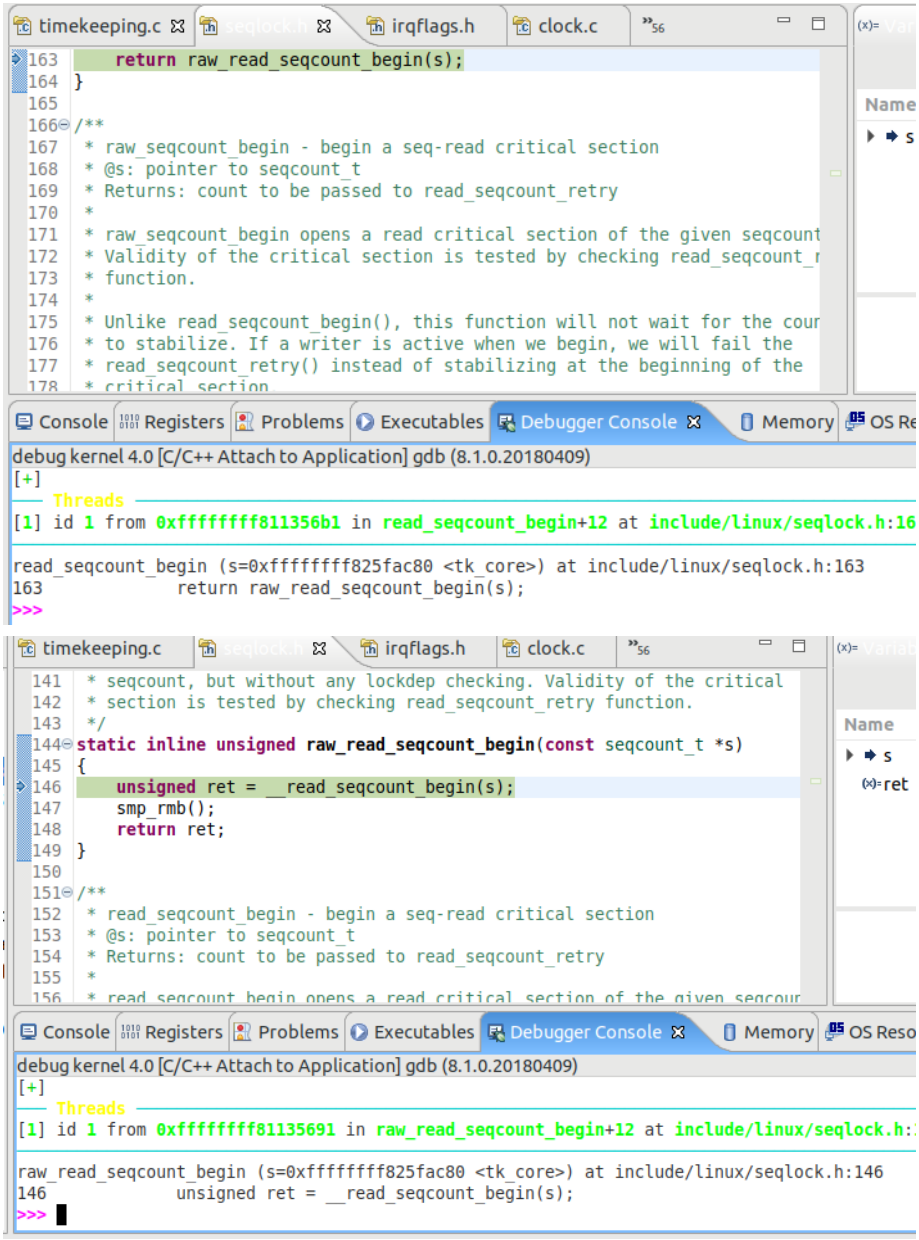

core.c
timer.c
irq_work.c
llist.h
» 55
322 if (sched_clock_stable())
323 return;
324
325 if (unlikely(!sched_clock_running))
326 return;
327
328 WARN_ON_ONCE(!irqs_disabled());
329
330 scd = this_scd();
331 now_gtod = ktime_to_ns(ktime_get());
332 now = sched_clock();
333
334 scd->tick_raw = now;
335 scd->tick_gtod = now_gtod;
336 sched_clock_local(scd);
337 }
Console
Registers
Problems
Executables
Debugger Console
Memor
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d9c29 in sched_clock_tick+8 at kernel/sched/clock.c:
sched_clock_tick () at kernel/sched/clock.c:322
322 if (sched_clock_stable())
>>>


core.c
timer.c
irq_work.c
llist.h
» 55
80 static struct static_key __sched_clock_stable = STATIC_KEY_INIT;
81 static int __sched_clock_stable_early;
82
83 int sched_clock_stable(void)
84 {
85 return static_key_false(&__sched_clock_stable);
86 }
87
88 static void __set_sched_clock_stable(void)
89 {
90 if (!sched_clock_stable())
91 static_key_slow_inc(&__sched_clock_stable);
92 }
93
94 void set_sched_clock_stable(void)
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d97a0 in sched_clock_stable+16 at kernel/sched/clock.c:85
sched_clock_stable () at kernel/sched/clock.c:85
85 return static_key_false(&__sched_clock_stable);
>>>

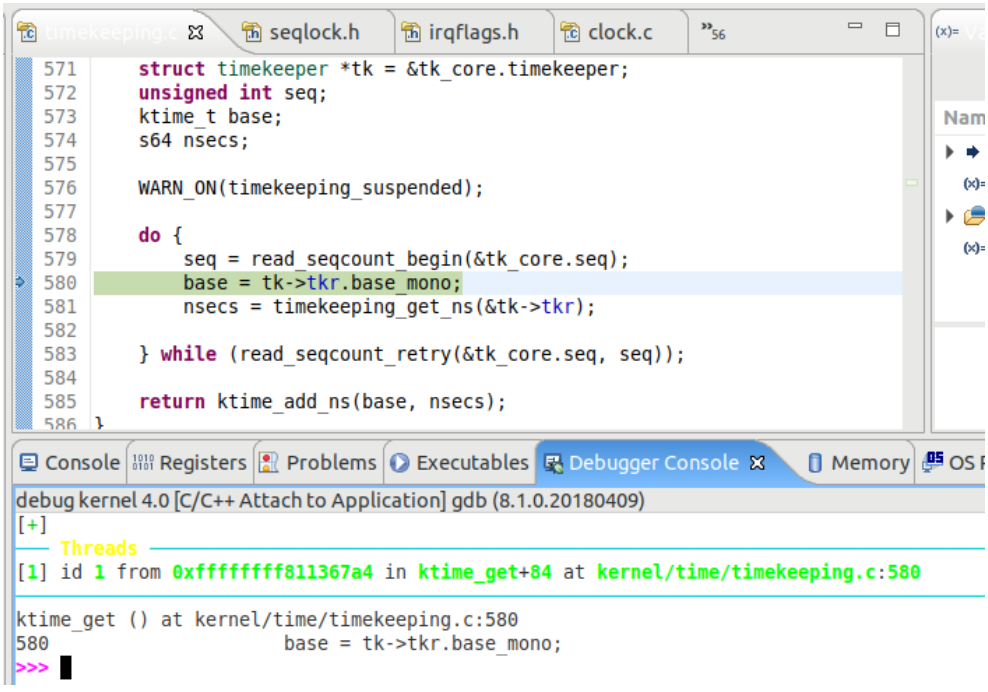

core.c
timer.c
clock.c
Variables
Name
key
150
151 static __always_inline bool static_key_false(struct static_key *key)
152 {
153 if (unlikely(static_key_count(key) > 0))
154 return true;
155 return false;
156 }
157
158 static __always_inline bool static_key_true(struct static_key *key)
159 {
160 if (likely(static_key_count(key) > 0))
161 return true;
162 return false;
163 }
164
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d97a0 in static_key_false+0 at include/linux/jump_label.h:153
static_key_false (key=0xffffffff825b4bd4 <__sched_clock_stable>) at include/linux/jump_label.h:153
153 if (unlikely(static_key_count(key) > 0))
>>>

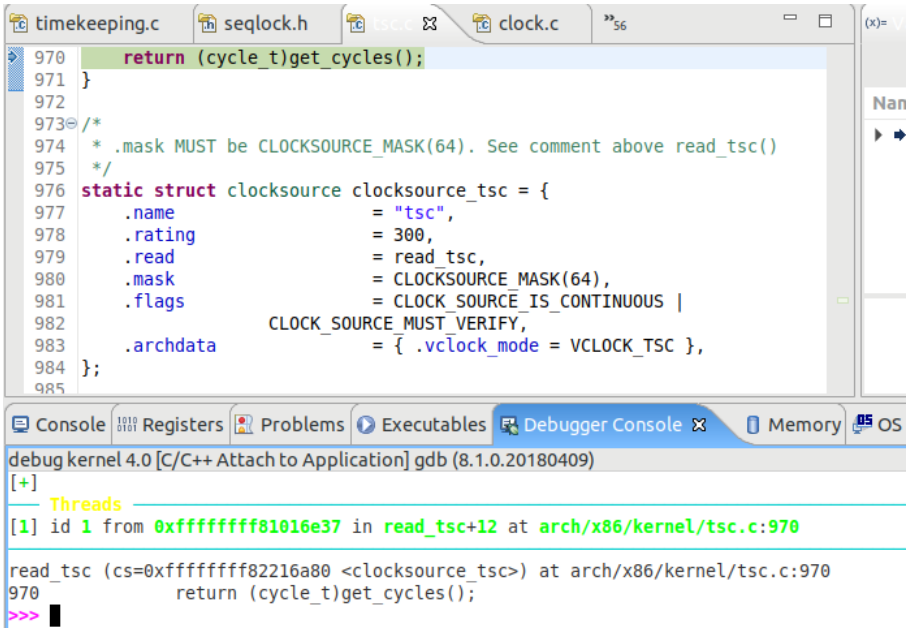

core.c
jump_label.h
clock.c
27 return ACCESS_ONCE((v)->counter);
28 }
29
30 /**
31 * atomic_set - set atomic variable
32 * @v: pointer of type atomic_t
33 * @i: required value
34 *
35 * Atomically sets the value of @v to @i.
36 */
37 static inline void atomic_set(atomic_t *v, int i)
38 {
39 v->counter = i;
40 }
41
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d9601 in atomic_read+12 at ./arch/x86/include/asm/atomic
atomic_read (v=0xffffffff825b4bd4 <__sched_clock_stable>) at ./arch/x86/include/asm
27 return ACCESS_ONCE((v)->counter);
>>>

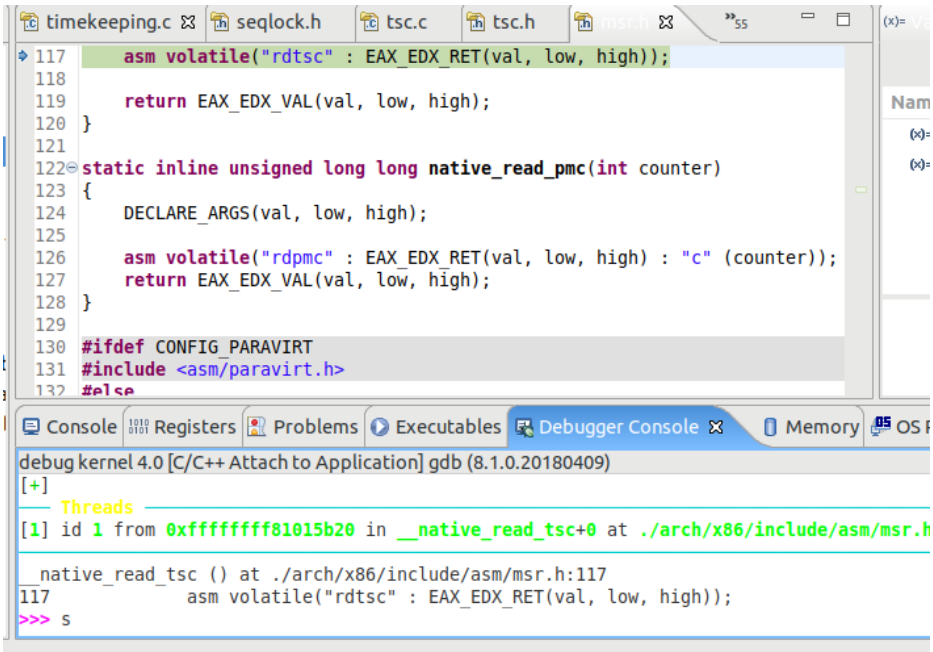

atomic.h
jump_label.h
clock.c
62 #ifndef __ASSEMBLY__
63 #include <linux/types.h>
64
65 static inline notrace unsigned long arch_local_save_flags(void)
66 {
67 return native_save_fl();
68 }
69
70 static inline notrace void arch_local_irq_restore(unsigned long flags)
71 {
72 native_restore_fl(flags);
73 }
74
75 static inline notrace void arch_local_irq_disable(void)
76 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d95ce in arch_local_save_flags+4 at ./arch/x86/include/as
arch_local_save_flags () at ./arch/x86/include/asm/irqflags.h:67
67 return native_save_fl();
>>>

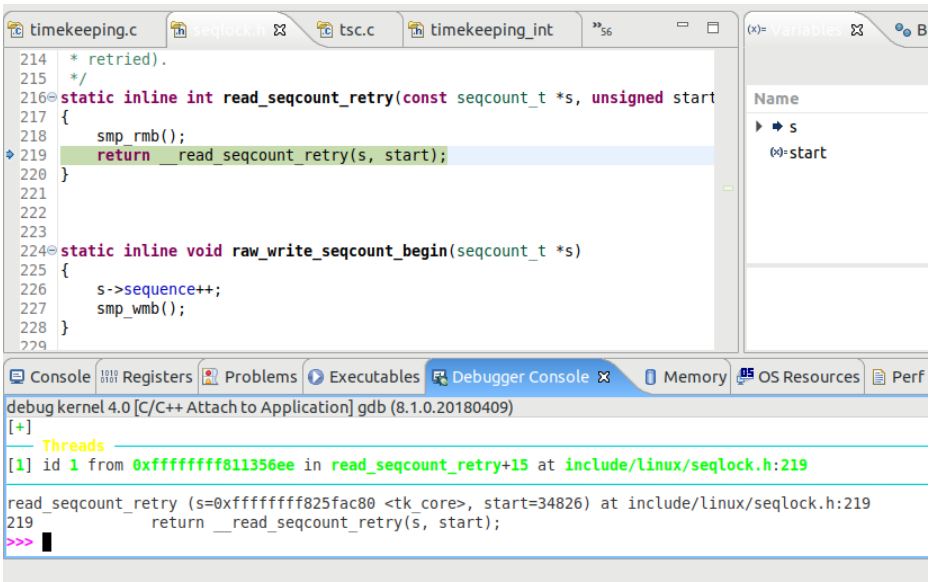

timekeeping.c
irqflags.h
clock.c
163 return raw_read_seqcount_begin(s);
164 }
166 /**
167 * raw_seqcount_begin - begin a seq-read critical section
168 * @s: pointer to seqcount_t
169 * Returns: count to be passed to read_seqcount_retry
171 * raw_seqcount_begin opens a read critical section of the given seqcount
172 * Validity of the critical section is tested by checking read_seqcount_r
173 * function.
175 * Unlike read_seqcount_begin(), this function will not wait for the cour
176 * to stabilize. If a writer is active when we begin, we will fail the
177 * read_seqcount_retry() instead of stabilizing at the beginning of the
178 * critical section
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811356b1 in read_seqcount_begin+12 at include/linux/seqlock.h:16
read_seqcount_begin (s=0xffffffff825fac80 <tk_core>) at include/linux/seqlock.h:163
163 return raw_read_seqcount_begin(s);
>>>

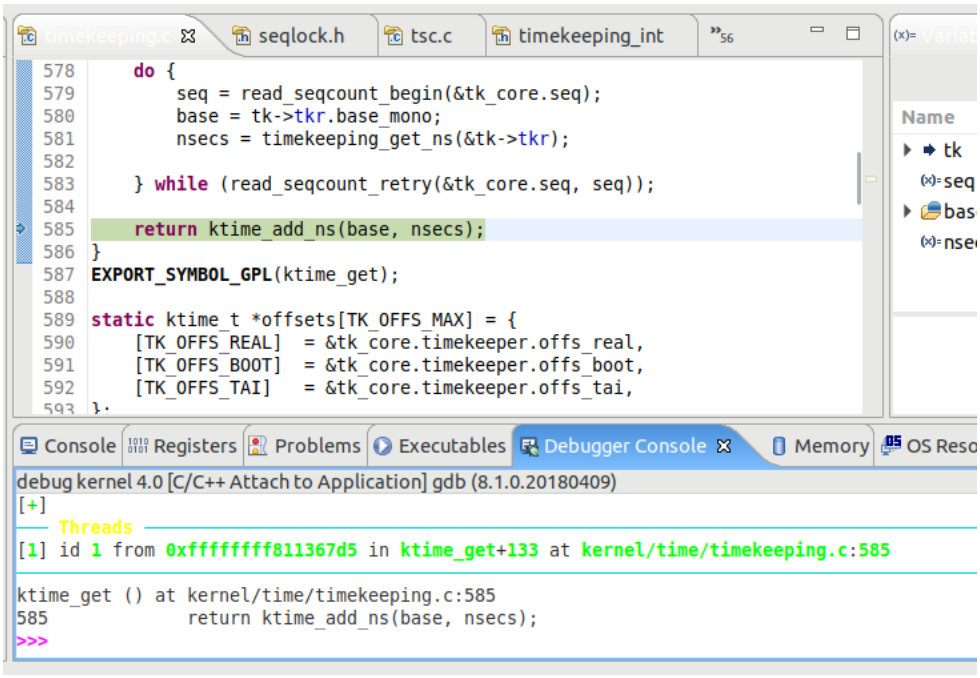
timekeeping.c
irqflags.h
clock.c
141 * seqcount, but without any lockdep checking. Validity of the critical
142 * section is tested by checking read_seqcount_retry function.
143 */
144 static inline unsigned raw_read_seqcount_begin(const seqcount_t *s)
145 {
146 unsigned ret = __read_seqcount_begin(s);
147 smp_rmb();
148 return ret;
149 }
151 /**
152 * read_seqcount_begin - begin a seq-read critical section
153 * @s: pointer to seqcount_t
154 * Returns: count to be passed to read_seqcount_retry
Name
s
ret
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81135691 in raw_read_seqcount_begin+12 at include/linux/seqlock.h:
raw_read_seqcount_begin (s=0xffffffff825fac80 <tk_core>) at include/linux/seqlock.h:146
146 unsigned ret = __read_seqcount_begin(s);
>>>

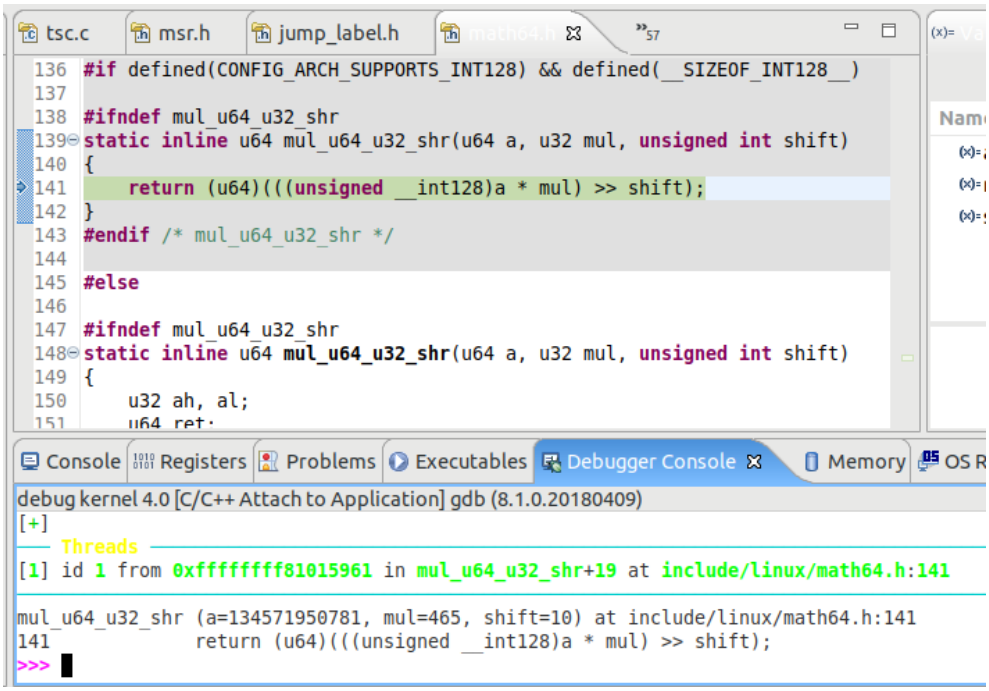

seqlock.h
irqflags.h
clock.c
571 struct timekeeper *tk = &tk_core.timekeeper;
572 unsigned int seq;
573 ktime_t base;
574 s64 nsecs;
576 WARN_ON(timekeeping_suspended);
578 do {
579 seq = read_seqcount_begin(&tk_core.seq);
580 base = tk->tkr.base_mono;
581 nsecs = timekeeping_get_ns(&tk->tkr);
583 } while (read_seqcount_retry(&tk_core.seq, seq));
585 return ktime_add_ns(base, nsecs);
586 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811367a4 in ktime_get+84 at kernel/time/timekeeping.c:580
ktime_get () at kernel/time/timekeeping.c:580
580 base = tk->tkr.base_mono;
>>>

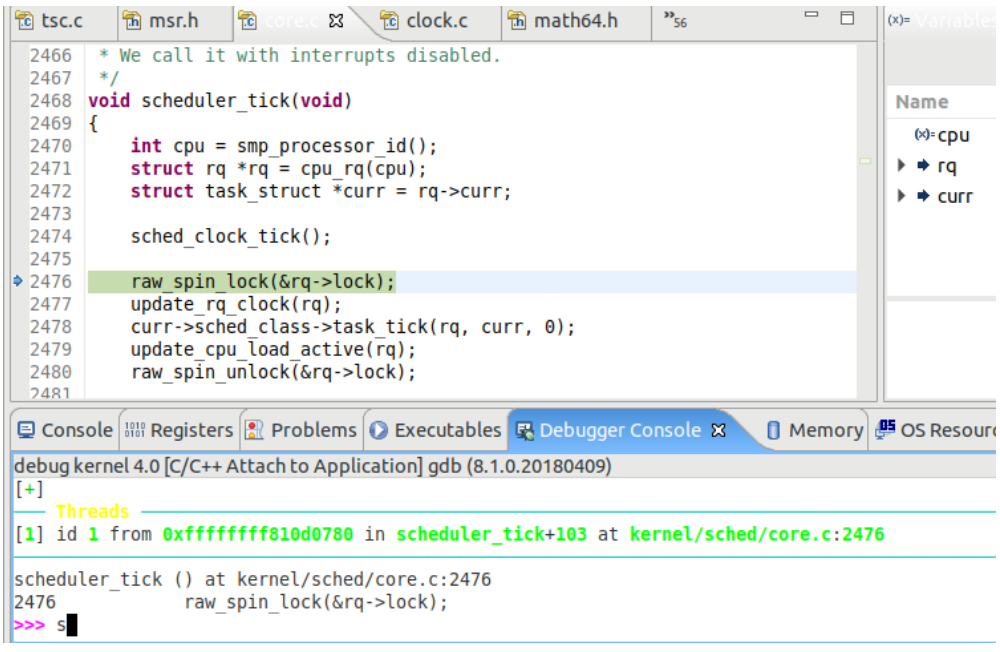

timekeeping.c
seqlock.h
clock.c
970 return (cycle_t)get_cycles();
971 }
973 /*
974 * .mask MUST be CLOCKSOURCE_MASK(64). See comment above read_tsc()
975 */
976 static struct clocksource clocksource_tsc = {
977 .name = "tsc",
978 .rating = 300,
979 .read = read_tsc,
980 .mask = CLOCKSOURCE_MASK(64),
981 .flags = CLOCK_SOURCE_IS_CONTINUOUS |
982 CLOCK_SOURCE_MUST_VERIFY,
983 .archdata = { .vclock_mode = VCLOCK_TSC },
984 };
Console
Registers
Problems
Executables
Debugger Console
Memory
OS
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81016e37 in read_tsc+12 at arch/x86/kernel/tsc.c:970
read_tsc (cs=0xffffffff82216a80 <clocksource_tsc>) at arch/x86/kernel/tsc.c:970
970 return (cycle_t)get_cycles();
>>>

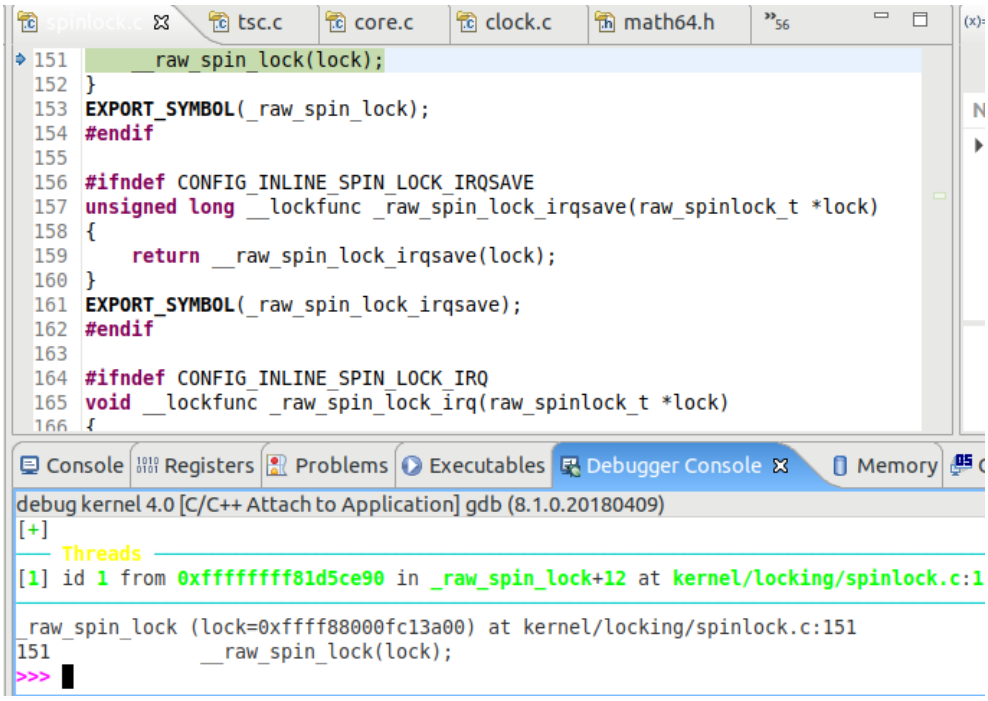

timekeeping.c
seqlock.h
tsc.c
tsc.h
117 asm volatile("rdtsc" : EAX_EDX_RET(val, low, high));
119 return EAX_EDX_VAL(val, low, high);
120 }
122 static inline unsigned long long native_read_pmc(int counter)
123 {
124 DECLARE_ARGS(val, low, high);
126 asm volatile("rdpmc" : EAX_EDX_RET(val, low, high) : "c" (counter));
127 return EAX_EDX_VAL(val, low, high);
128 }
130 #ifdef CONFIG_PARAVIRT
131 #include <asm/paravirt.h>
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81015b20 in __native_read_tsc+0 at ./arch/x86/include/asm/msr.h
__native_read_tsc () at ./arch/x86/include/asm/msr.h:117
117 asm volatile("rdtsc" : EAX_EDX_RET(val, low, high));
>>> s

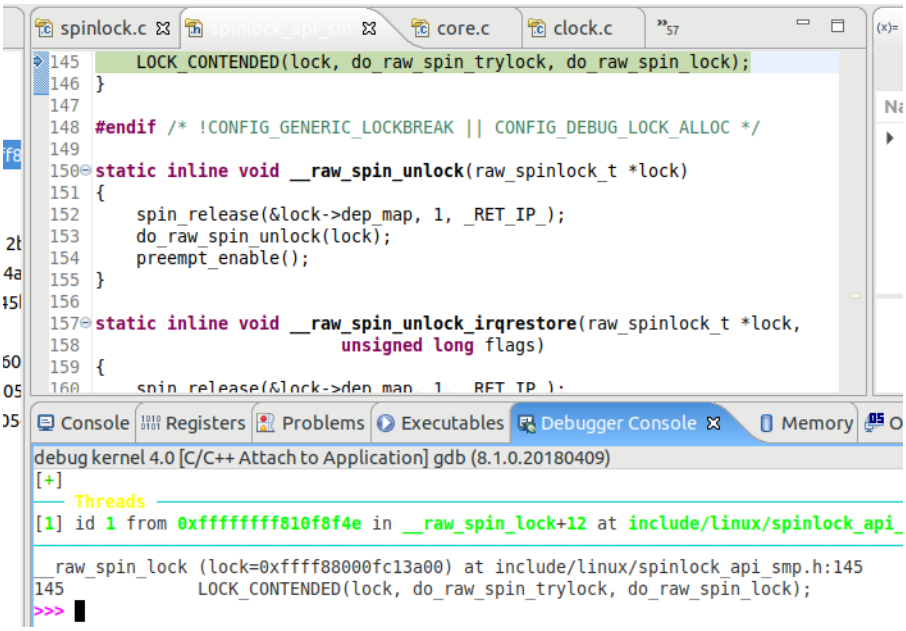

timekeeping.c
tsc.c
timekeeping_int
214 * retried).
215 */
216 static inline int read_seqcount_retry(const seqcount_t *s, unsigned start
217 {
218 smp_rmb();
219 return __read_seqcount_retry(s, start);
220 }
224 static inline void raw_write_seqcount_begin(seqcount_t *s)
225 {
226 s->sequence++;
227 smp_wmb();
228 }
Name
s
start
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811356ee in read_seqcount_retry+15 at include/linux/seqlock.h:219
read_seqcount_retry (s=0xffffffff825fac80 <tk_core>, start=34826) at include/linux/seqlock.h:219
219 return __read_seqcount_retry(s, start);
>>>

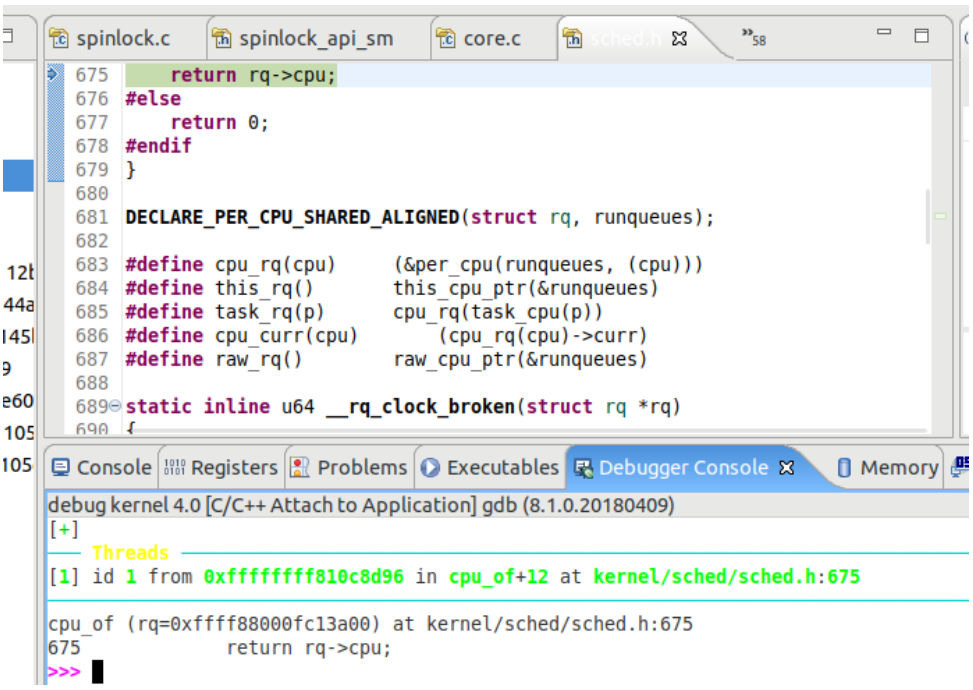

seqlock.h
tsc.c
timekeeping_int
578 do {
579 seq = read_seqcount_begin(&tk_core.seq);
580 base = tk->tkr.base_mono;
581 nsecs = timekeeping_get_ns(&tk->tkr);
582
583 } while (read_seqcount_retry(&tk_core.seq, seq));
584
585 return ktime_add_ns(base, nsecs);
586 }
587 EXPORT_SYMBOL_GPL(ktime_get);
588
589 static ktime_t *offsets[TK_OFFS_MAX] = {
590 [TK_OFFS_REAL] = &tk_core.timekeeper.offs_real,
591 [TK_OFFS_BOOT] = &tk_core.timekeeper.offs_boot,
592 [TK_OFFS_TAI] = &tk_core.timekeeper.offs_tai,
Name
tk
seq
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff811367d5 in ktime_get+133 at kernel/time/timekeeping.c:585
ktime_get () at kernel/time/timekeeping.c:585
585 return ktime_add_ns(base, nsecs);
>>>

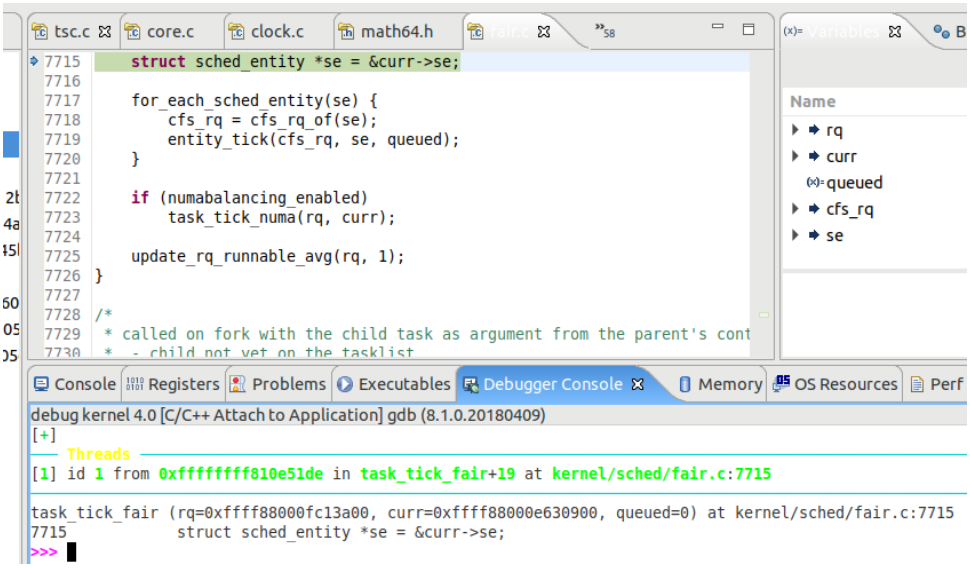

tsc.c
msr.h
jump_label.h
136 #if defined(CONFIG_ARCH_SUPPORTS_INT128) && defined(__SIZEOF_INT128__)
137
138 #ifndef mul_u64_u32_shr
139 static inline u64 mul_u64_u32_shr(u64 a, u32 mul, unsigned int shift)
140 {
141 return (u64)(((unsigned __int128)a * mul) >> shift);
142 }
143 #endif /* mul_u64_u32_shr */
144
145 #else
146
147 #ifndef mul_u64_u32_shr
148 static inline u64 mul_u64_u32_shr(u64 a, u32 mul, unsigned int shift)
149 {
150 u32 ah, al;
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81015961 in mul_u64_u32_shr+19 at include/linux/math64.h:141
mul_u64_u32_shr (a=134571950781, mul=465, shift=10) at include/linux/math64.h:141
141 return (u64)(((unsigned __int128)a * mul) >> shift);
>>>

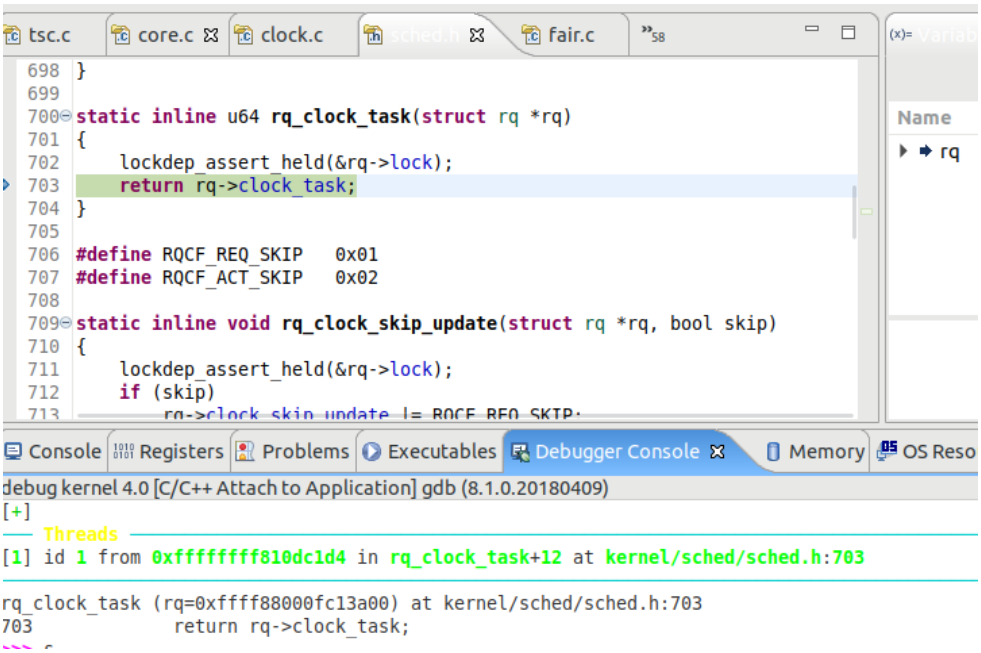

tsc.c
msr.h
clock.c
math64.h
2466 * We call it with interrupts disabled.
2467 */
2468 void scheduler_tick(void)
2469 {
2470 int cpu = smp_processor_id();
2471 struct rq *rq = cpu_rq(cpu);
2472 struct task_struct *curr = rq->curr;
2473
2474 sched_clock_tick();
2475
2476 raw_spin_lock(&rq->lock);
2477 update_rq_clock(rq);
2478 curr->sched_class->task_tick(rq, curr, 0);
2479 update_cpu_load_active(rq);
2480 raw_spin_unlock(&rq->lock);
Name
cpu
rq
curr
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810d0780 in scheduler_tick+103 at kernel/sched/core.c:2476
scheduler_tick () at kernel/sched/core.c:2476
2476 raw_spin_lock(&rq->lock);
>>> s

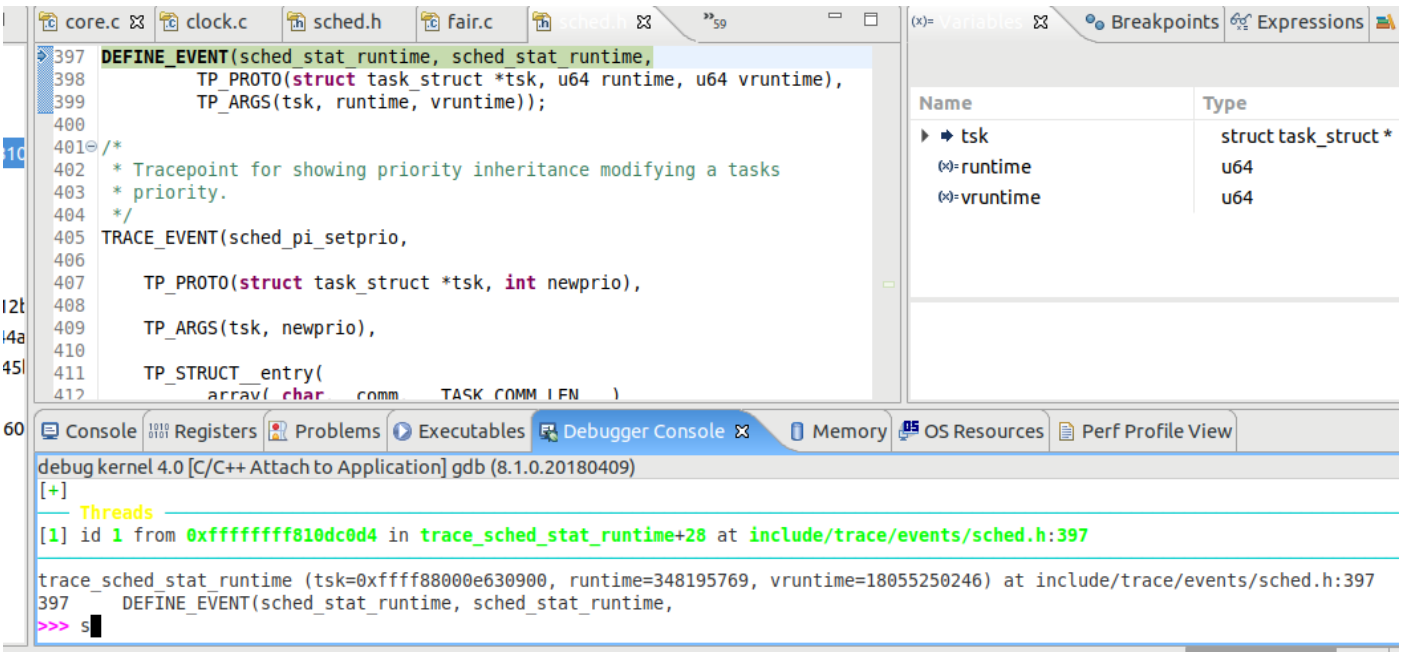

tsc.c
core.c
clock.c
math64.h
151 __raw_spin_lock(lock);
152 }
153 EXPORT_SYMBOL(_raw_spin_lock);
154 #endif
155
156 #ifndef CONFIG_INLINE_SPIN_LOCK_IRQSAVE
157 unsigned long __lockfunc _raw_spin_lock_irqsave(raw_spinlock_t *lock)
158 {
159 return __raw_spin_lock_irqsave(lock);
160 }
161 EXPORT_SYMBOL(_raw_spin_lock_irqsave);
162 #endif
163
164 #ifndef CONFIG_INLINE_SPIN_LOCK_IRQ
165 void __lockfunc _raw_spin_lock_irq(raw_spinlock_t *lock)
166 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81d5ce90 in _raw_spin_lock+12 at kernel/locking/spinlock.c:1
_raw_spin_lock (lock=0xffff88000fc13a00) at kernel/locking/spinlock.c:151
151 __raw_spin_lock(lock);
>>>

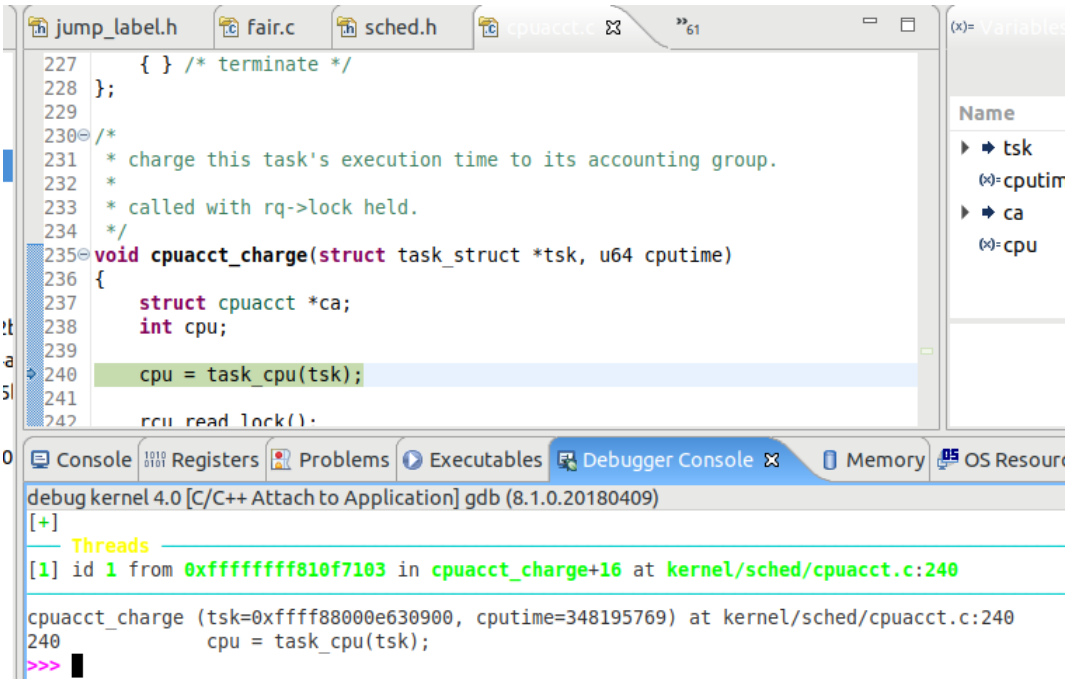

spinlock.c
core.c
clock.c
145 LOCK_CONTENDED(lock, do_raw_spin_trylock, do_raw_spin_lock);
146 }
147
148 #endif /* !CONFIG_GENERIC_LOCKBREAK || CONFIG_DEBUG_LOCK_ALLOC */
149
150 static inline void __raw_spin_unlock(raw_spinlock_t *lock)
151 {
152 spin_release(&lock->dep_map, 1, _RET_IP_);
153 do_raw_spin_unlock(lock);
154 preempt_enable();
155 }
156
157 static inline void __raw_spin_unlock_irqrestore(raw_spinlock_t *lock,
158 unsigned long flags)
159 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810f8f4e in __raw_spin_lock+12 at include/linux/spinlock_api_
__raw_spin_lock (lock=0xffff88000fc13a00) at include/linux/spinlock_api_smp.h:145
145 LOCK_CONTENDED(lock, do_raw_spin_trylock, do_raw_spin_lock);
>>>

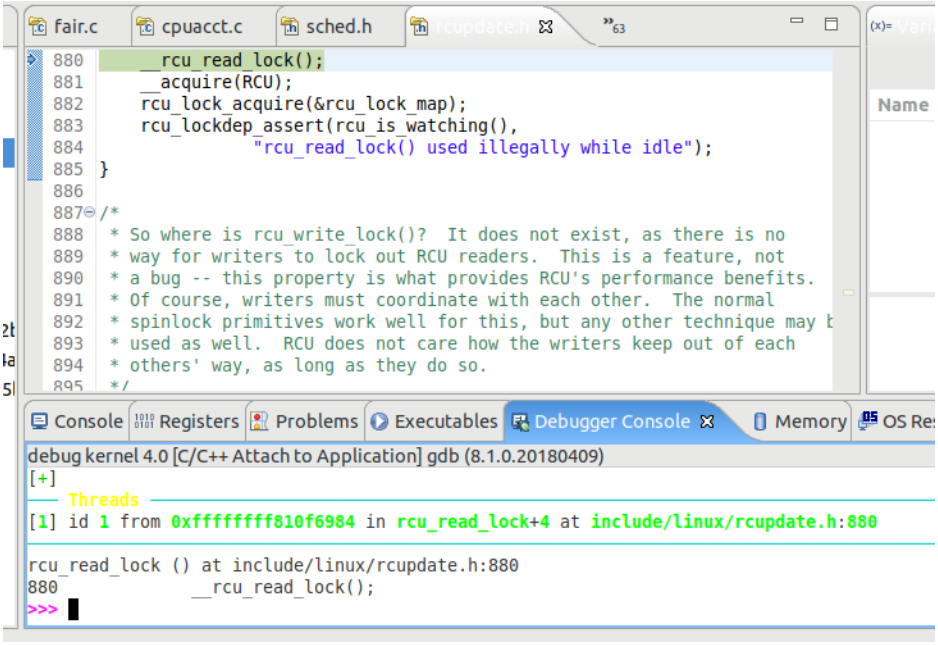

spinlock.c
spinlock_api_sm
core.c
675 return rq->cpu;
676 #else
677 return 0;
678 #endif
679 }
680
681 DECLARE_PER_CPU_SHARED_ALIGNED(struct rq, runqueues);
682
683 #define cpu_rq(cpu) (&per_cpu(runqueues, (cpu)))
684 #define this_rq() this_cpu_ptr(&runqueues)
685 #define task_rq(p) cpu_rq(task_cpu(p))
686 #define cpu_curr(cpu) (cpu_rq(cpu)->curr)
687 #define raw_rq() raw_cpu_ptr(&runqueues)
688
689 static inline u64 __rq_clock_broken(struct rq *rq)
690 {
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810c8d96 in cpu_of+12 at kernel/sched/sched.h:675
cpu_of (rq=0xffff88000fc13a00) at kernel/sched/sched.h:675
675 return rq->cpu;
>>>

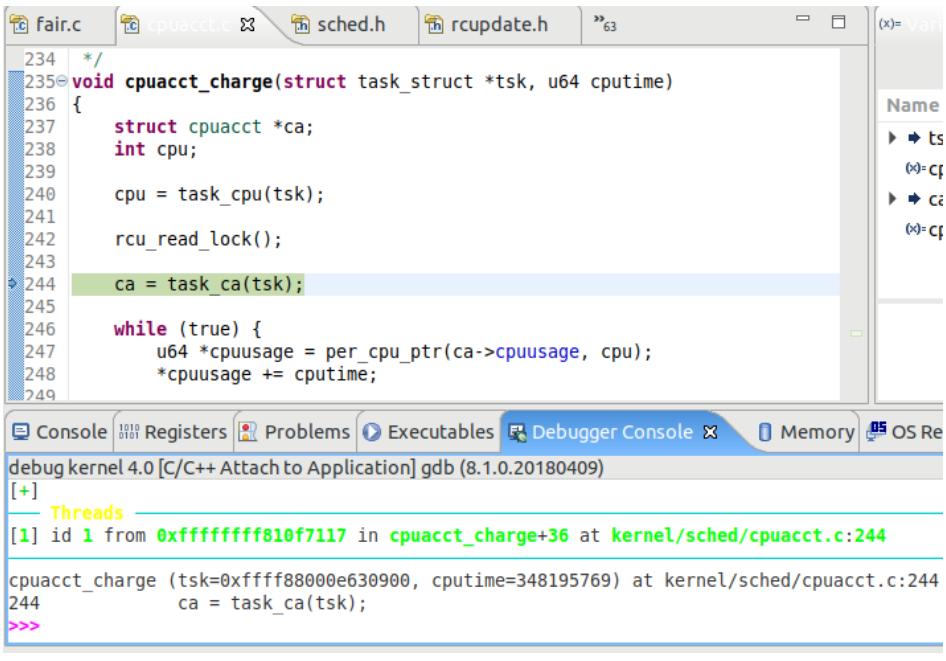

tsc.c
core.c
clock.c
math64.h
fair.c
7715 struct sched_entity *se = &curr->se;
7716
7717 for_each_sched_entity(se) {
7718 cfs_rq = cfs_rq_of(se);
7719 entity_tick(cfs_rq, se, queued);
7720 }
7721
7722 if (numabalancing_enabled)
7723 task_tick_numa(rq, curr);
7724
7725 update_rq_runnable_avg(rq, 1);
7726 }
7727
7728 /*
7729 * called on fork with the child task as argument from the parent's cont
7730 * - child not yet on the tasklist
Name
rq
curr
queued
cfs_rq
se
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810e51de in task_tick_fair+19 at kernel/sched/fair.c:7715
task_tick_fair (rq=0xffff88000fc13a00, curr=0xffff88000e630900, queued=0) at kernel/sched/fair.c:7715
7715 struct sched_entity *se = &curr->se;
>>>

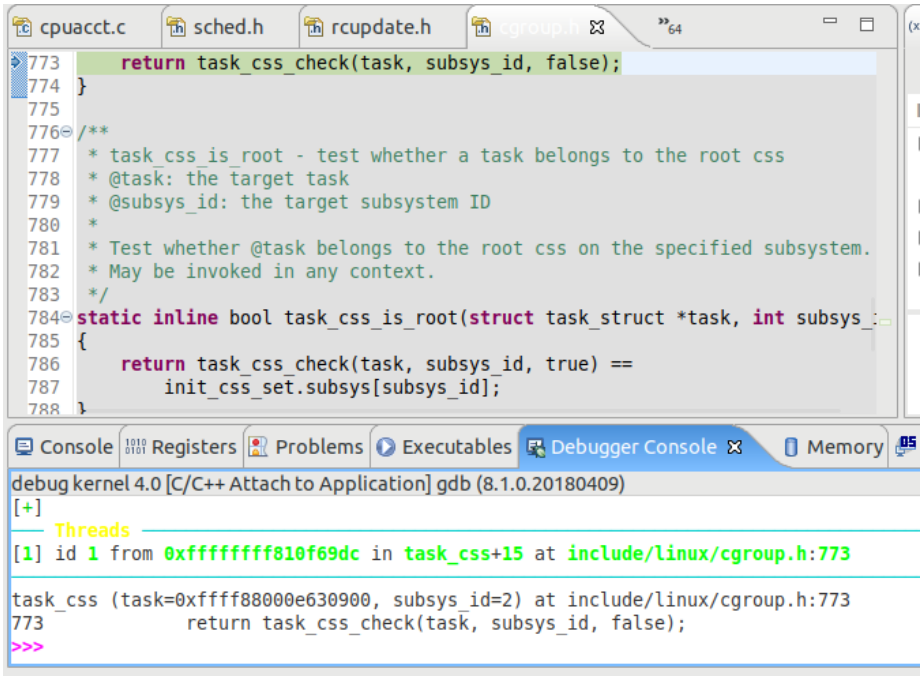

tsc.c
core.c
clock.c
fair.c
698 }
699
700 static inline u64 rq_clock_task(struct rq *rq)
701 {
702 lockdep_assert_held(&rq->lock);
703 return rq->clock_task;
704 }
705
706 #define RQCF_REQ_SKIP 0x01
707 #define RQCF_ACT_SKIP 0x02
708
709 static inline void rq_clock_skip_update(struct rq *rq, bool skip)
710 {
711 lockdep_assert_held(&rq->lock);
712 if (skip)
Name
rq
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810dc1d4 in rq_clock_task+12 at kernel/sched/sched.h:703
rq_clock_task (rq=0xffff88000fc13a00) at kernel/sched/sched.h:703
703 return rq->clock_task;

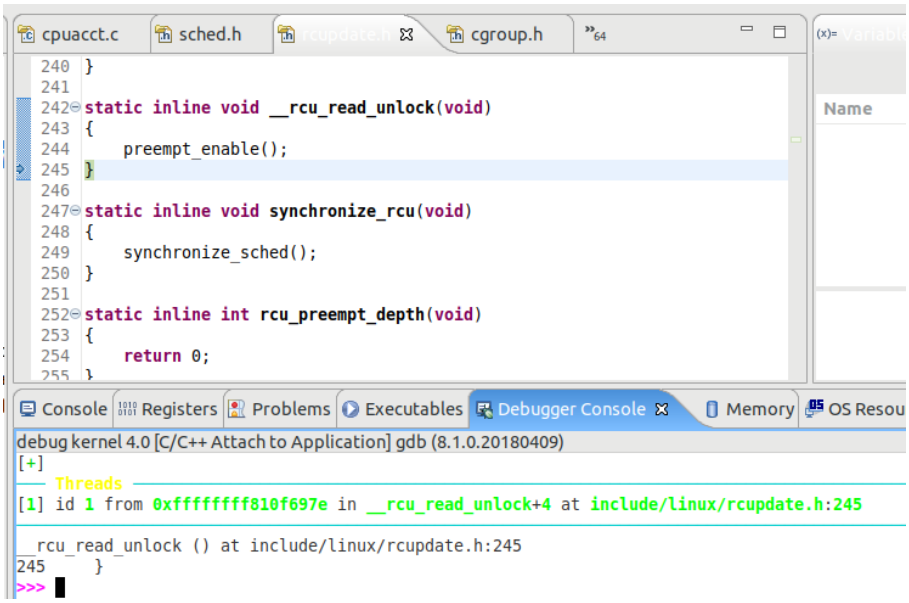

core.c
clock.c
sched.h
fair.c
397 DEFINE_EVENT(sched_stat_runtime, sched_stat_runtime,
398 TP_PROTO(struct task_struct *tsk, u64 runtime, u64 vruntime),
399 TP_ARGS(tsk, runtime, vruntime));
400
401 /*
402 * Tracepoint for showing priority inheritance modifying a tasks
403 * priority.
404 */
405 TRACE_EVENT(sched_pi_setprio,
406
407 TP_PROTO(struct task_struct *tsk, int newprio),
408
409 TP_ARGS(tsk, newprio),
410
411 TP_STRUCT__entry(
Breakpoints
Expressions
Name
Type
tsk
struct task_struct *
runtime
u64
vruntime
u64
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf Profile View
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810dc0d4 in trace_sched_stat_runtime+28 at include/trace/events/sched.h:397
trace_sched_stat_runtime (tsk=0xffff88000e630900, runtime=348195769, vruntime=18055250246) at include/trace/events/sched.h:397
397 DEFINE_EVENT(sched_stat_runtime, sched_stat_runtime,
>>> s

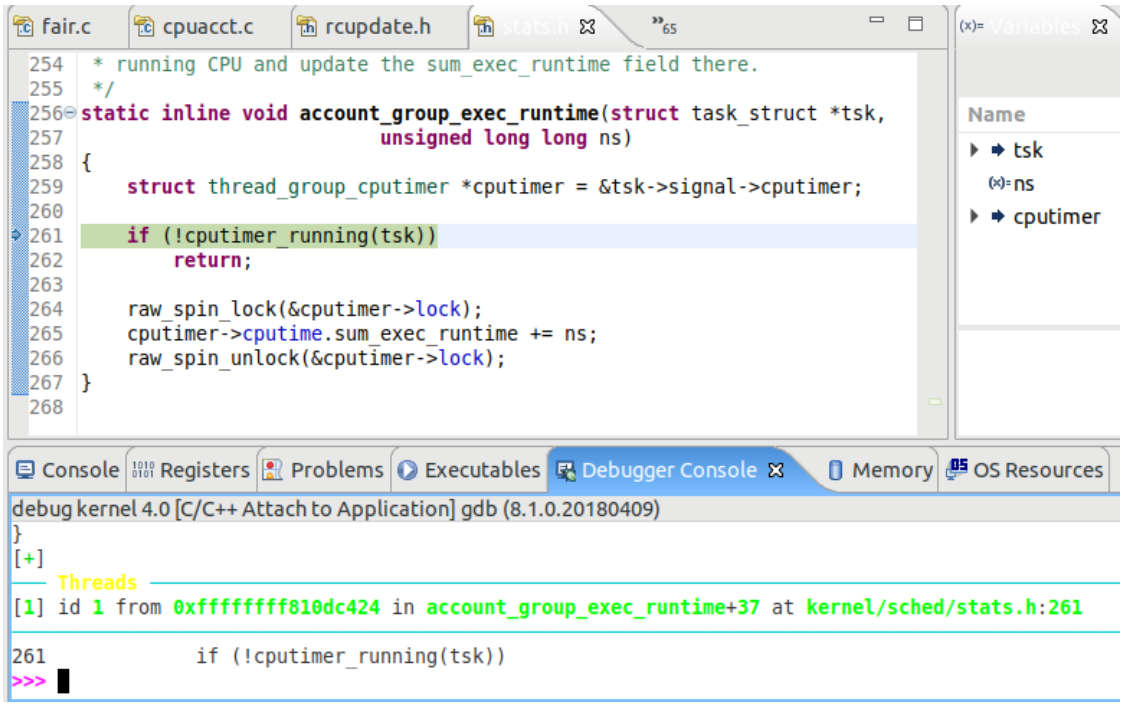

jump_label.h
fair.c
sched.h
227 { } /* terminate */
228 };
229
230 /*
231 * charge this task's execution time to its accounting group.
232 *
233 * called with rq->lock held.
234 */
235 void cpuacct_charge(struct task_struct *tsk, u64 cputime)
236 {
237 struct cpuacct *ca;
238 int cpu;
239
240 cpu = task_cpu(tsk);
241
242 rcu_read_lock();
Name
tsk
cputim
ca
cpu
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resour
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810f7103 in cpuacct_charge+16 at kernel/sched/cpuacct.c:240
cpuacct_charge (tsk=0xffff88000e630900, cputime=348195769) at kernel/sched/cpuacct.c:240
240 cpu = task_cpu(tsk);
>>>

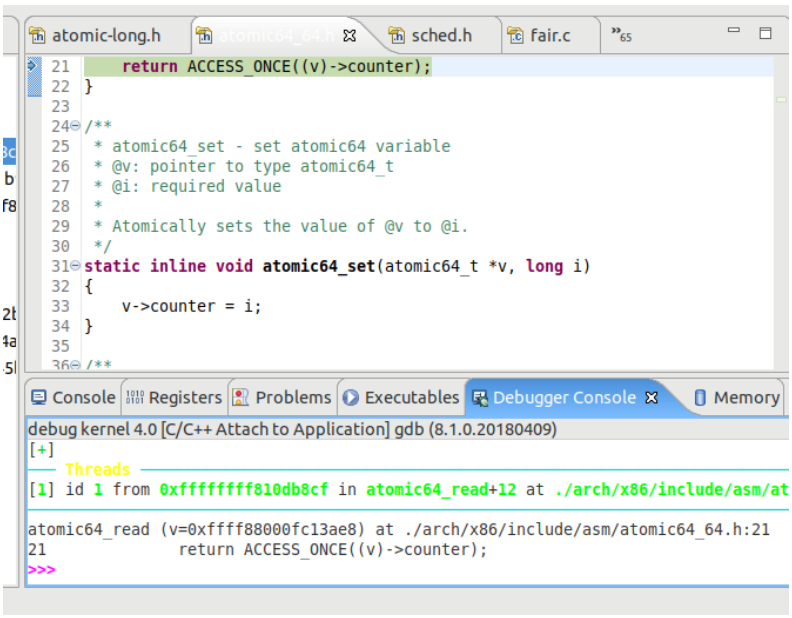

fair.c
cpuacct.c
sched.h
880 __rcu_read_lock();
881 __acquire(RCU);
882 rcu_lock_acquire(&rcu_lock_map);
883 rcu_lockdep_assert(rcu_is_watching(),
884 "rcu_read_lock() used illegally while idle");
885 }
886
887 /*
888 * So where is rcu_write_lock()? It does not exist, as there is no
889 * way for writers to lock out RCU readers. This is a feature, not
890 * a bug -- this property is what provides RCU's performance benefits.
891 * Of course, writers must coordinate with each other. The normal
892 * spinlock primitives work well for this, but any other technique may b
893 * used as well. RCU does not care how the writers keep out of each
894 * others' way, as long as they do so.
895 */
Name
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810f6984 in rcu_read_lock+4 at include/linux/rcupdate.h:880
rcu_read_lock () at include/linux/rcupdate.h:880
880 __rcu_read_lock();
>>>

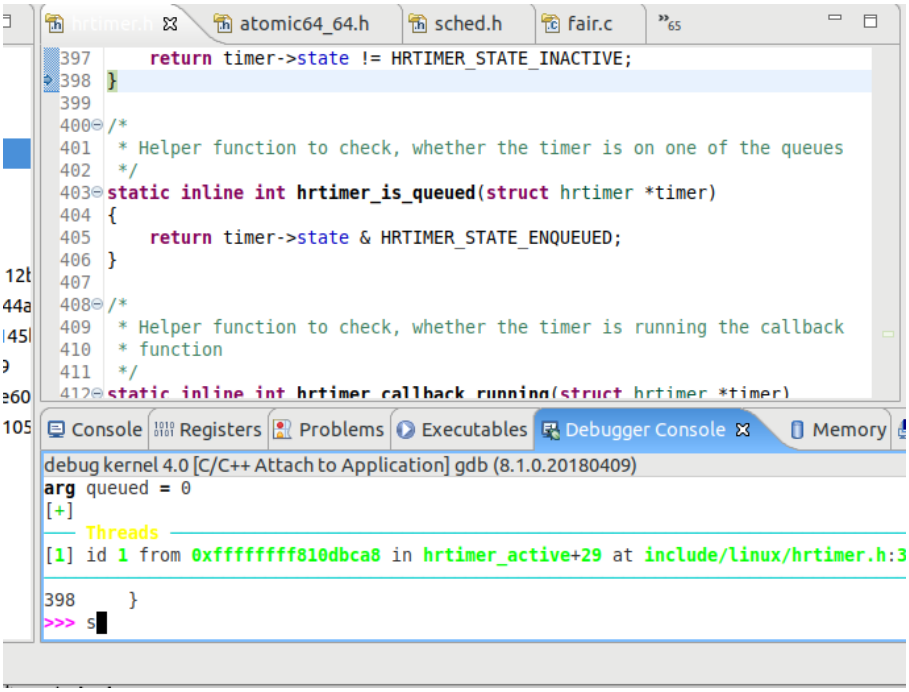

fair.c
sched.h
rcupdate.h
234 */
235 void cpuacct_charge(struct task_struct *tsk, u64 cputime)
236 {
237 struct cpuacct *ca;
238 int cpu;
239
240 cpu = task_cpu(tsk);
241
242 rcu_read_lock();
243
244 ca = task_ca(tsk);
245
246 while (true) {
247 u64 *cpuusage = per_cpu_ptr(ca->cpuusage, cpu);
248 *cpuusage += cputime;
Name
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810f7117 in cpuacct_charge+36 at kernel/sched/cpuacct.c:244
cpuacct_charge (tsk=0xffff88000e630900, cputime=348195769) at kernel/sched/cpuacct.c:244
244 ca = task_ca(tsk);
>>>

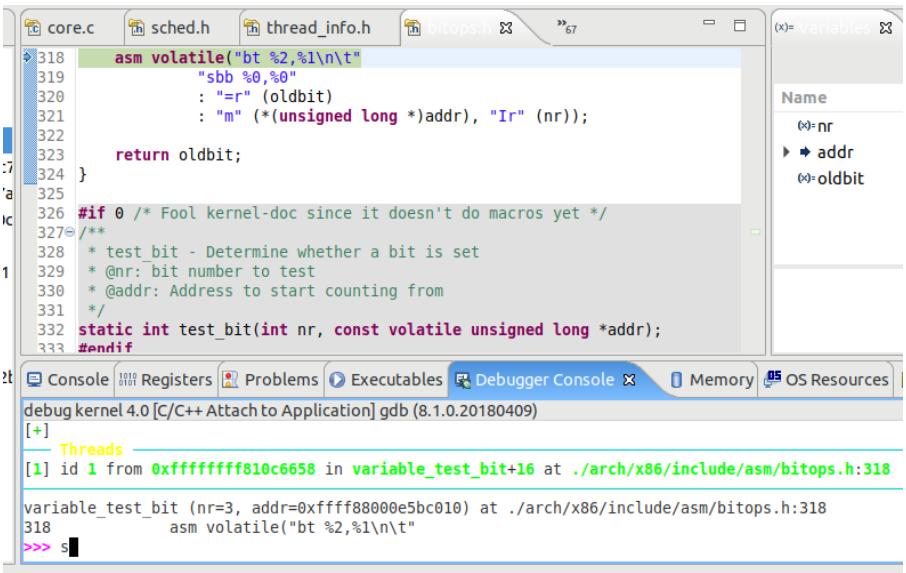

cpuacct.c
sched.h
rcupdate.h
773 return task_css_check(task, subsys_id, false);
774 }
775
776 /**
777 * task_css_is_root - test whether a task belongs to the root css
778 * @task: the target task
779 * @subsys_id: the target subsystem ID
780 *
781 * Test whether @task belongs to the root css on the specified subsystem.
782 * May be invoked in any context.
783 */
784 static inline bool task_css_is_root(struct task_struct *task, int subsys_
785 {
786 return task_css_check(task, subsys_id, true) ==
787 init_css_set.subsys[subsys_id];
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810f69dc in task_css+15 at include/linux/cgroup.h:773
task_css (task=0xffff88000e630900, subsys_id=2) at include/linux/cgroup.h:773
773 return task_css_check(task, subsys_id, false);
>>>

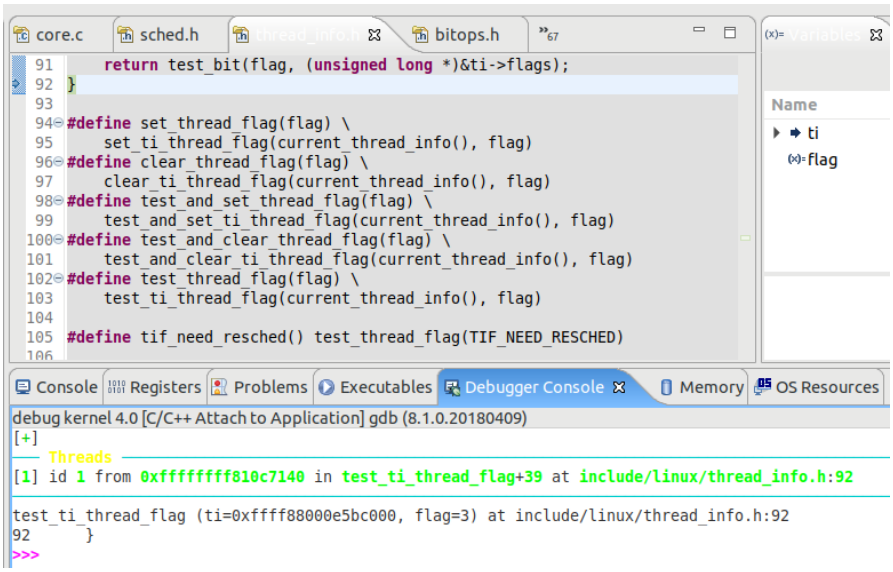

cpuacct.c
sched.h
cgroup.h
240 }
241
242 static inline void __rcu_read_unlock(void)
243 {
244 preempt_enable();
245 }
246
247 static inline void synchronize_rcu(void)
248 {
249 synchronize_sched();
250 }
251
252 static inline int rcu_preempt_depth(void)
253 {
254 return 0;
Name
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resou
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810f697e in __rcu_read_unlock+4 at include/linux/rcupdate.h:245
__rcu_read_unlock () at include/linux/rcupdate.h:245
245 }
>>>

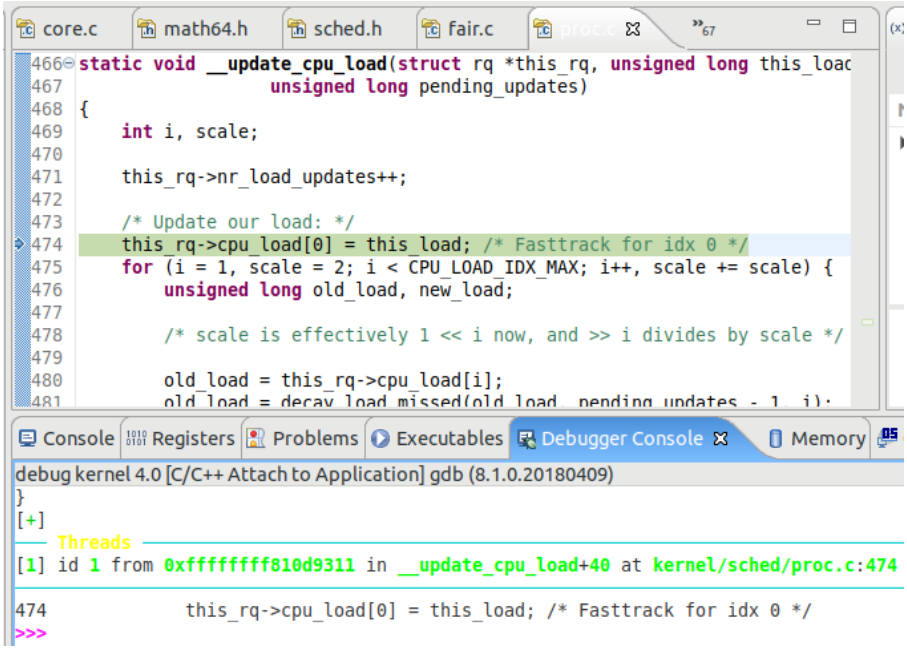

fair.c
cpuacct.c
rcupdate.h
254 * running CPU and update the sum_exec_runtime field there.
255 */
256 static inline void account_group_exec_runtime(struct task_struct *tsk,
257 unsigned long long ns)
258 {
259 struct thread_group_cputimer *cputimer = &tsk->signal->cputimer;
260
261 if (!cputimer_running(tsk))
262 return;
263
264 raw_spin_lock(&cputimer->lock);
265 cputimer->cputime.sum_exec_runtime += ns;
266 raw_spin_unlock(&cputimer->lock);
267 }
268
Name
tsk
ns
cputimer
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
}
[+]
Threads
[1] id 1 from 0xffffffff810dc424 in account_group_exec_runtime+37 at kernel/sched/stats.h:261
261 if (!cputimer_running(tsk))
>>>

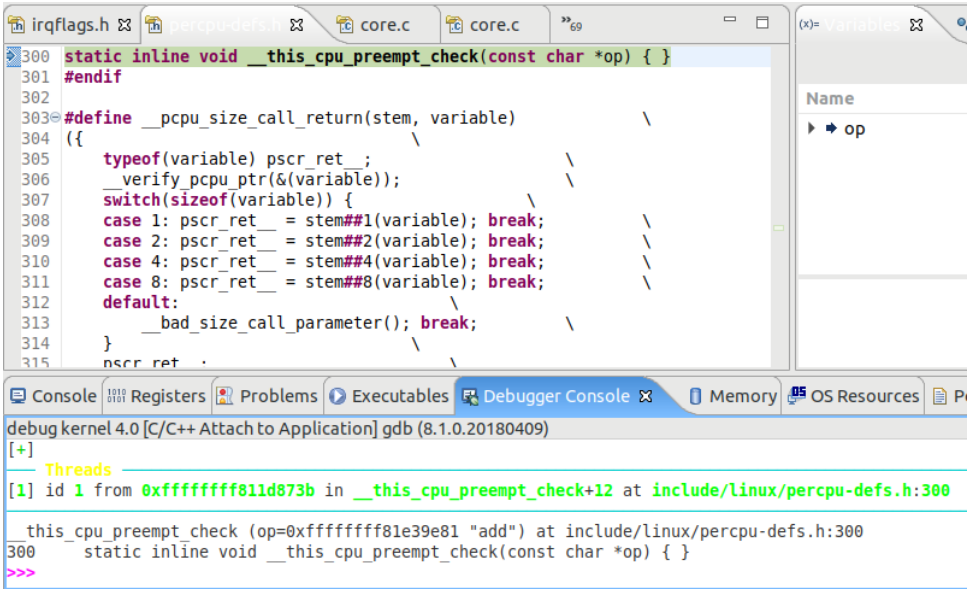

atomic-long.h
sched.h
fair.c
21 return ACCESS_ONCE((v)->counter);
22 }
24 /**
25 * atomic64_set - set atomic64 variable
26 * @v: pointer to type atomic64_t
27 * @i: required value
29 * Atomically sets the value of @v to @i.
31 static inline void atomic64_set(atomic64_t *v, long i)
32 {
33 v->counter = i;
34 }
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810db8cf in atomic64_read+12 at ./arch/x86/include/asm/at
atomic64_read (v=0xffff88000fc13ae8) at ./arch/x86/include/asm/atomic64_64.h:21
21 return ACCESS_ONCE((v)->counter);
>>>

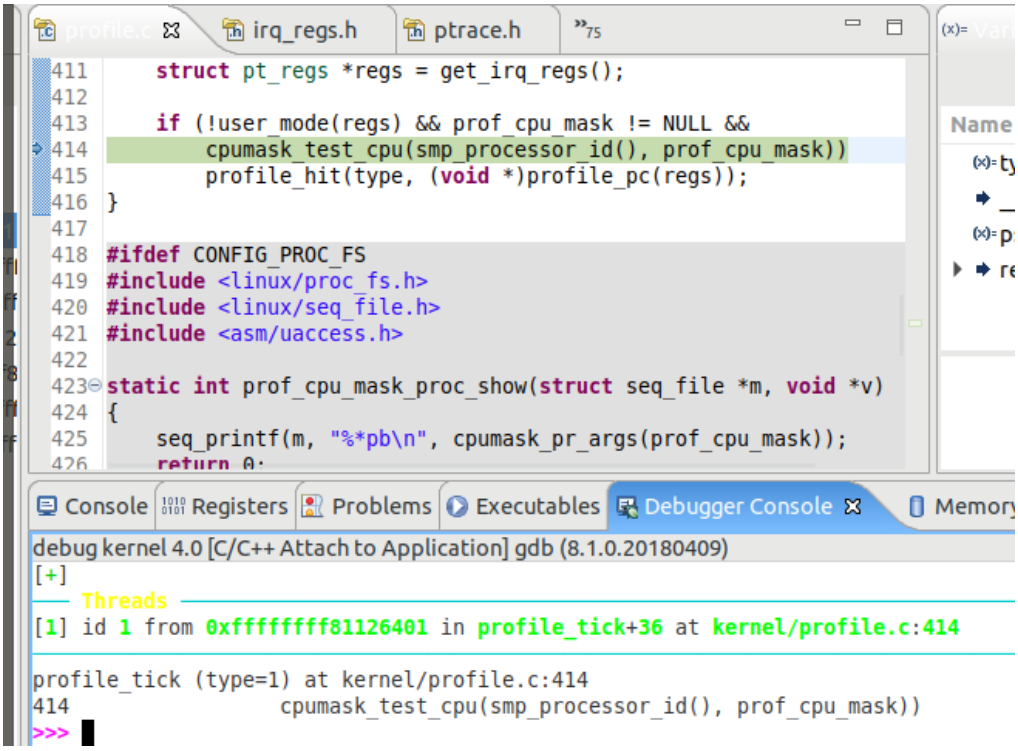

atomic64_64.h
sched.h
fair.c
397 return timer->state != HRTIMER_STATE_INACTIVE;
398 }
400 /*
401 * Helper function to check, whether the timer is on one of the queues
403 static inline int hrtimer_is_queued(struct hrtimer *timer)
404 {
405 return timer->state & HRTIMER_STATE_ENQUEUED;
406 }
408 /*
409 * Helper function to check, whether the timer is running the callback
410 * function
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
arg queued = 0
[+]
Threads
[1] id 1 from 0xffffffff810dbca8 in hrtimer_active+29 at include/linux/hrtimer.h:3
398 }
>>> s

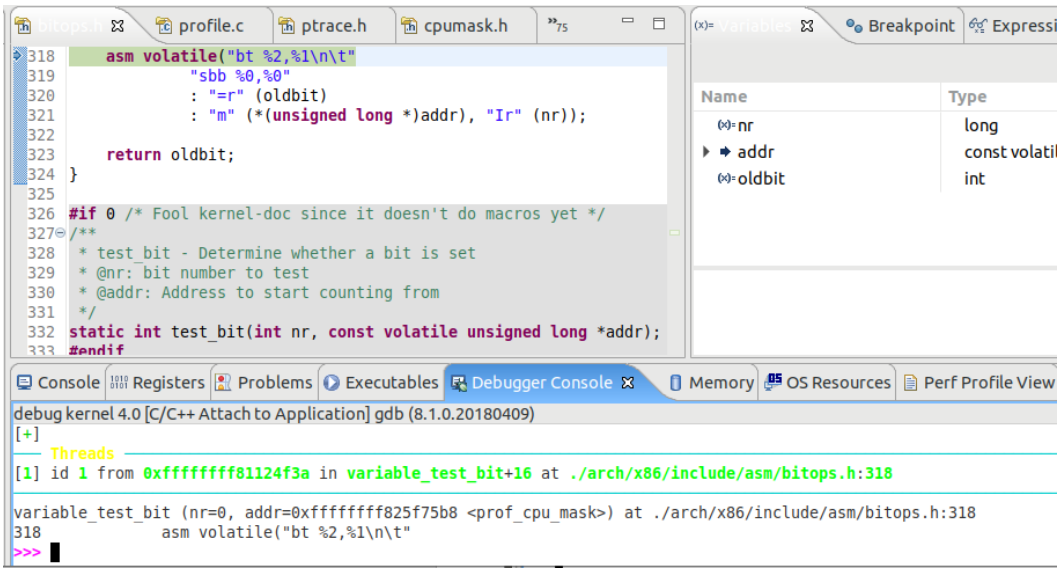

core.c
sched.h
thread_info.h
318 asm volatile("bt %2,%1\n\t"
319 "sbb %0,%0"
320 : "=r" (oldbit)
321 : "m" (*(unsigned long *)addr), "Ir" (nr));
323 return oldbit;
324 }
326 #if 0 /* Fool kernel-doc since it doesn't do macros yet */
327 /**
328 * test_bit - Determine whether a bit is set
329 * @nr: bit number to test
330 * @addr: Address to start counting from
332 static int test_bit(int nr, const volatile unsigned long *addr);
Name
nr
addr
oldbit
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff810c6658 in variable_test_bit+16 at ./arch/x86/include/asm/bitops.h:318
variable_test_bit (nr=3, addr=0xffff88000e5bc010) at ./arch/x86/include/asm/bitops.h:318
318 asm volatile("bt %2,%1\n\t"
>>> s

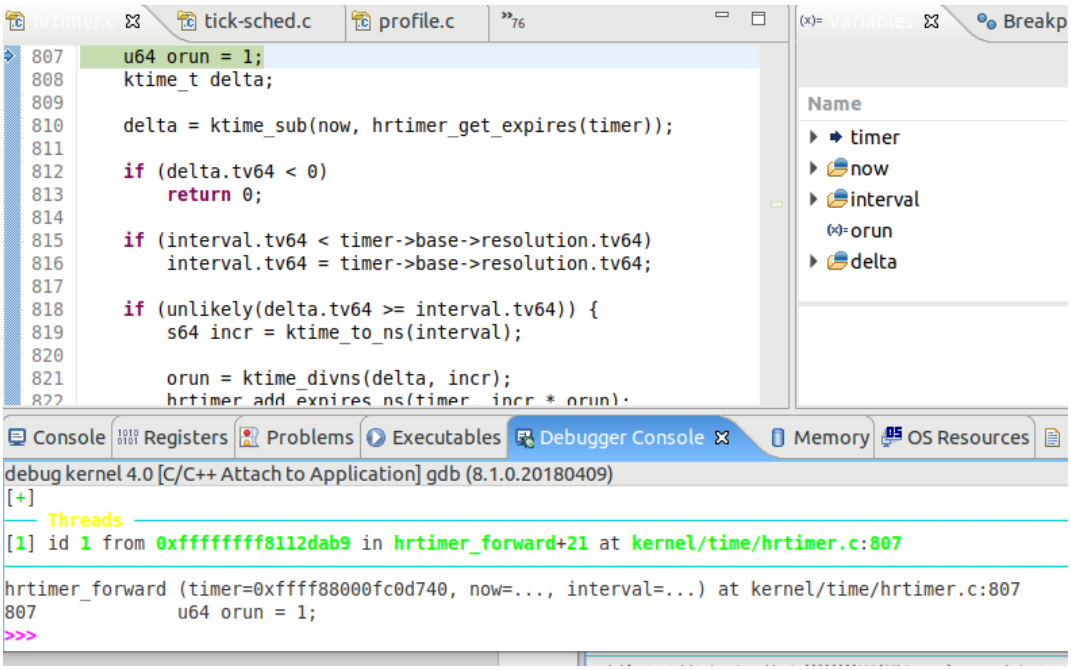

core.c
sched.h
bitops.h
Variables
Name
ti
flag
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff810c7140 in test_ti_thread_flag+39 at include/linux/thread_info.h:92
test_ti_thread_flag (ti=0xffff88000e5bc000, flag=3) at include/linux/thread_info.h:92
92 }
>>>

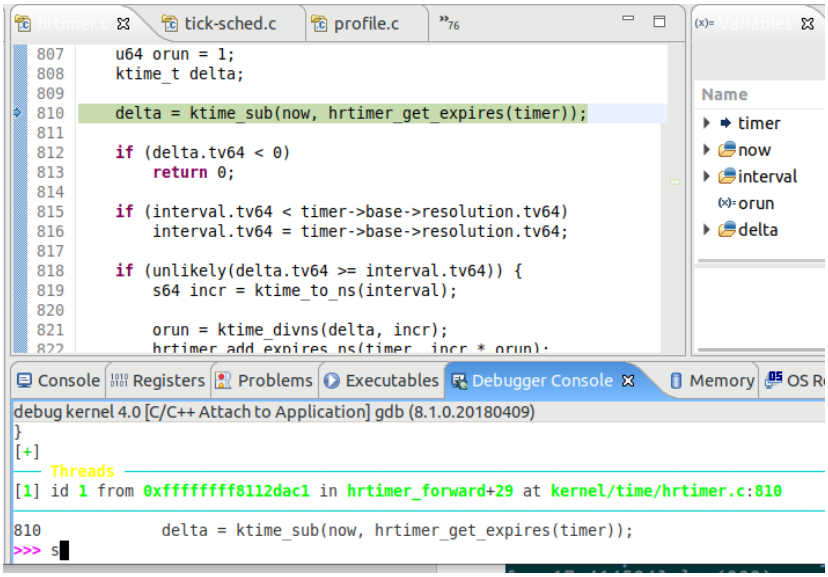

core.c
math64.h
sched.h
fair.c
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff810d9311 in __update_cpu_load+40 at kernel/sched/proc.c:474
474 this_rq->cpu_load[0] = this_load; /* Fasttrack for idx 0 */
>>>

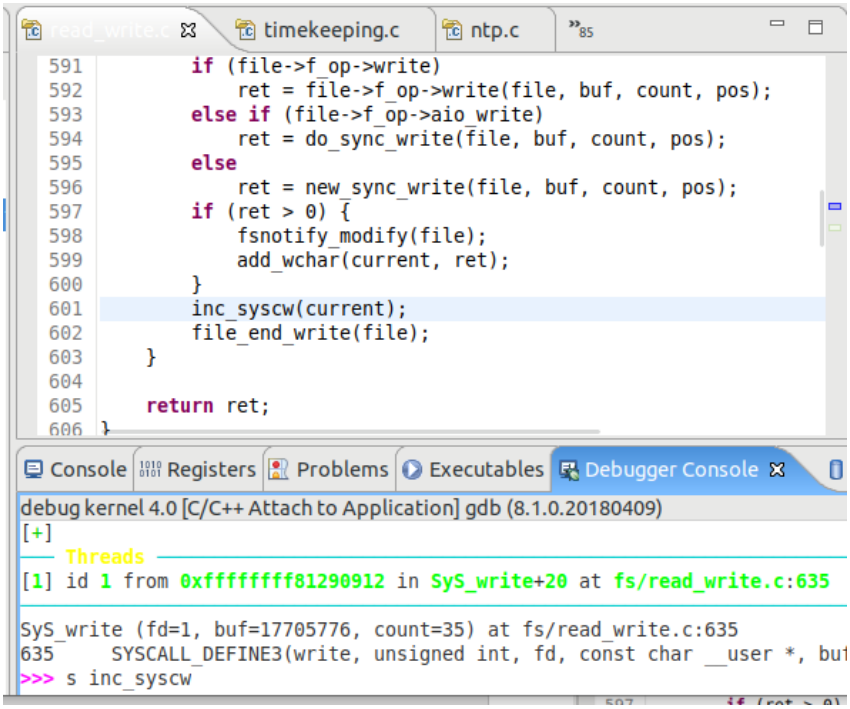

irqflags.h
core.c
core.c
Variables
Name
op
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
Threads
[1] id 1 from 0xffffffff811d873b in __this_cpu_preempt_check+12 at include/linux/percpu-defs.h:300
__this_cpu_preempt_check (op=0xffffffff81e39e81 "add") at include/linux/percpu-defs.h:300
300 static inline void __this_cpu_preempt_check(const char *op) { }
>>>

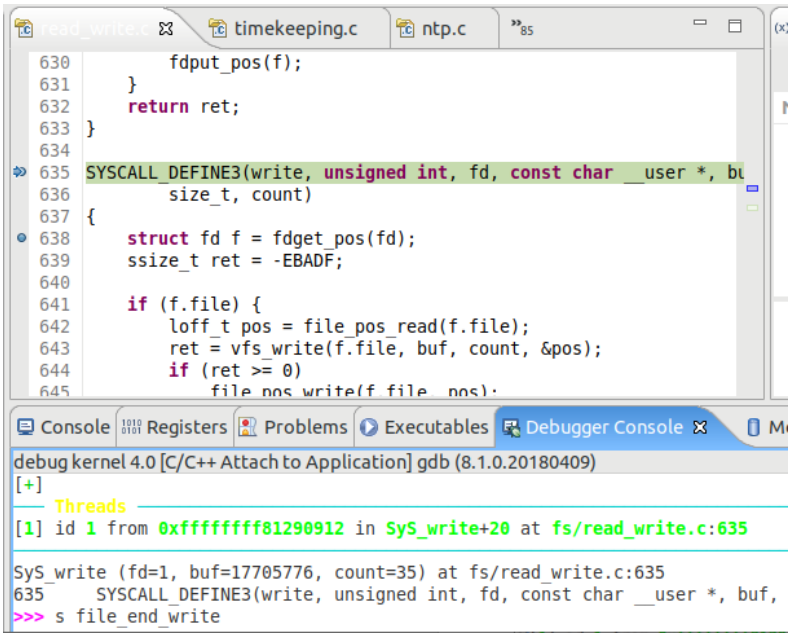

irq_regs.h
ptrace.h
411 struct pt_regs *regs = get_irq_regs();
412
413 if (!user_mode(regs) && prof_cpu_mask != NULL &&
414 cpumask_test_cpu(smp_processor_id(), prof_cpu_mask))
415 profile_hit(type, (void *)profile_pc(regs));
416 }
417
418 #ifdef CONFIG_PROC_FS
419 #include <linux/proc_fs.h>
420 #include <linux/seq_file.h>
421 #include <asm/uaccess.h>
422
423 static int prof_cpu_mask_proc_show(struct seq_file *m, void *v)
424 {
425 seq_printf(m, "%*pb\n", cpumask_pr_args(prof_cpu_mask));
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81126401 in profile_tick+36 at kernel/profile.c:414
profile_tick (type=1) at kernel/profile.c:414
414 cpumask_test_cpu(smp_processor_id(), prof_cpu_mask))
>>>

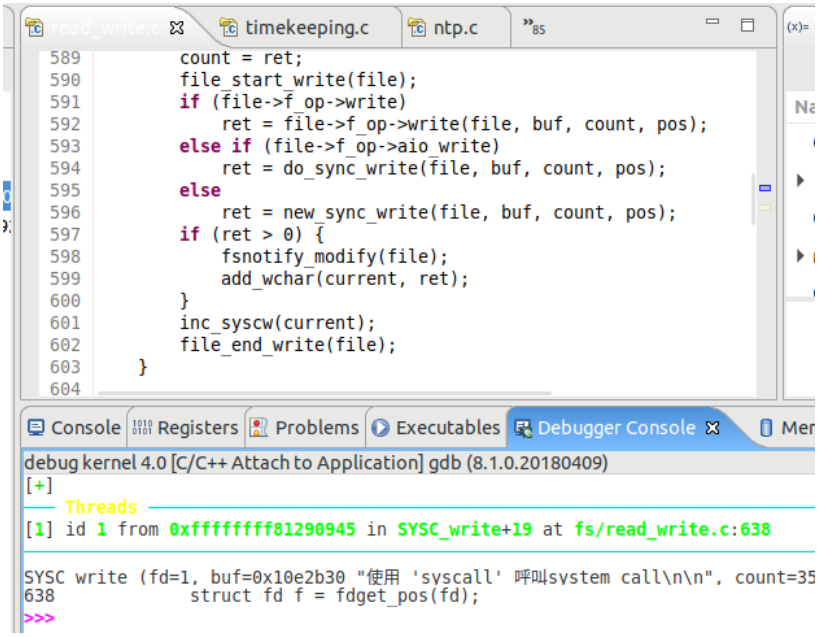

profile.c
ptrace.h
cpumask.h
318 asm volatile("bt %2,%1\n\t"
319 "sbb %0,%0"
320 : "=r" (oldbit)
321 : "m" (*(unsigned long *)addr), "Ir" (nr));
322
323 return oldbit;
324 }
325
326 #if 0 /* Fool kernel-doc since it doesn't do macros yet */
327 /**
328 * test_bit - Determine whether a bit is set
329 * @nr: bit number to test
330 * @addr: Address to start counting from
331 */
332 static int test_bit(int nr, const volatile unsigned long *addr);
333 #endif
Breakpoint
Name
Type
nr
long
addr
const volatil
oldbit
int
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
Perf Profile View
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81124f3a in variable_test_bit+16 at ./arch/x86/include/asm/bitops.h:318
variable_test_bit (nr=0, addr=0xffffffff825f75b8 <prof_cpu_mask>) at ./arch/x86/include/asm/bitops.h:318
318 asm volatile("bt %2,%1\n\t"
>>>


tick-sched.c
profile.c
807 u64 orun = 1;
808 ktime_t delta;
809
810 delta = ktime_sub(now, hrtimer_get_expires(timer));
811
812 if (delta.tv64 < 0)
813 return 0;
814
815 if (interval.tv64 < timer->base->resolution.tv64)
816 interval.tv64 = timer->base->resolution.tv64;
817
818 if (unlikely(delta.tv64 >= interval.tv64)) {
819 s64 incr = ktime_to_ns(interval);
820
821 orun = ktime_divns(delta, incr);
Name
timer
now
interval
orun
delta
Console
Registers
Problems
Executables
Debugger Console
Memory
OS Resources
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff8112dab9 in hrtimer_forward+21 at kernel/time/hrtimer.c:807
hrtimer_forward (timer=0xffff88000fc0d740, now=..., interval=...) at kernel/time/hrtimer.c:807
807 u64 orun = 1;
>>>


tick-sched.c
profile.c
807 u64 orun = 1;
808 ktime_t delta;
809
810 delta = ktime_sub(now, hrtimer_get_expires(timer));
811
812 if (delta.tv64 < 0)
813 return 0;
814
815 if (interval.tv64 < timer->base->resolution.tv64)
816 interval.tv64 = timer->base->resolution.tv64;
817
818 if (unlikely(delta.tv64 >= interval.tv64)) {
819 s64 incr = ktime_to_ns(interval);
820
821 orun = ktime_divns(delta, incr);
Name
timer
now
interval
orun
delta
Console
Registers
Problems
Executables
Debugger Console
Memory
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
}
[+]
Threads
[1] id 1 from 0xffffffff8112dac1 in hrtimer_forward+29 at kernel/time/hrtimer.c:810
810 delta = ktime_sub(now, hrtimer_get_expires(timer));
>>> s


timekeeping.c
ntp.c
591 if (file->f_op->write)
592 ret = file->f_op->write(file, buf, count, pos);
593 else if (file->f_op->aio_write)
594 ret = do_sync_write(file, buf, count, pos);
595 else
596 ret = new_sync_write(file, buf, count, pos);
597 if (ret > 0) {
598 fsnotify_modify(file);
599 add_wchar(current, ret);
600 }
601 inc_syscw(current);
602 file_end_write(file);
603 }
604
605 return ret;
606 }
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290912 in SyS_write+20 at fs/read_write.c:635
SyS_write (fd=1, buf=17705776, count=35) at fs/read_write.c:635
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf
>>> s inc_syscw


timekeeping.c
ntp.c
630 fdput_pos(f);
631 }
632 return ret;
633 }
634
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, bu
636 size_t, count)
637 {
638 struct fd f = fdget_pos(fd);
639 ssize_t ret = -EBADF;
640
641 if (f.file) {
642 loff_t pos = file_pos_read(f.file);
643 ret = vfs_write(f.file, buf, count, &pos);
644 if (ret >= 0)
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290912 in SyS_write+20 at fs/read_write.c:635
SyS_write (fd=1, buf=17705776, count=35) at fs/read_write.c:635
635 SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
>>> s file_end_write


timekeeping.c
ntp.c
count = ret;
file_start_write(file);
if (file->f_op->write)
ret = file->f_op->write(file, buf, count, pos);
else if (file->f_op->aio_write)
ret = do_sync_write(file, buf, count, pos);
else
ret = new_sync_write(file, buf, count, pos);
if (ret > 0) {
fsnotify_modify(file);
add_wchar(current, ret);
}
inc_syscw(current);
file_end_write(file);
}
Console
Registers
Problems
Executables
Debugger Console
debug kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)
[+]
Threads
[1] id 1 from 0xffffffff81290945 in SYSC_write+19 at fs/read_write.c:638
SYSC write (fd=1, buf=0x10e2b30 "使用 'syscall' 呼叫system call\n\n", count=35
638 struct fd f = fdget_pos(fd);
>>>