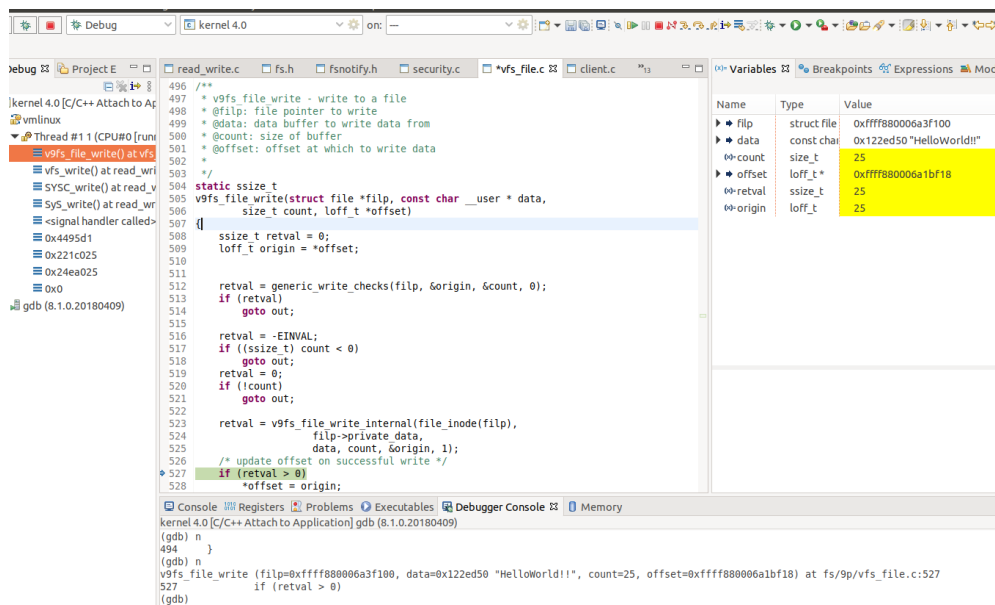


作業系統

407510046 經濟三 黃郁雯

甲.

```
/mnt/sharedFolder/hw09 # mount
rootfs on / type rootfs (rw,size=115272k,nr_inodes=28818)
proc on /proc type proc (rw,relatime)
tmpfs on /tmp type tmpfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
tmpfs on /dev type tmpfs (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
sharedFolder on /mnt/sharedFolder type 9p (rw,sync,dirsync,relatime,trans=virtio,version=9p2000.L,nobootwait)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/mnt/sharedFolder/hw09 # ls
makefile write_w_newline write_w_newline.c
/mnt/sharedFolder/hw09 # ./write_w newline
```



乙.

fdget_pos：透過__to_fd 回傳 struct fd 物件,取得 unsigned long, return __to_fd(__fdget_pos(fd))

__to_fd：return {(struct file*)(v&~3),v&3}

file_pos_read(f.file)：如果 fd f 有符合 file 結構體，取得當前檔案位置 pos

vfs_write：參數 buf，為指向用戶空間的內存地址

loff_t *pos=>pos 表示從何處開始寫入，其值需要初始化

EX: loff_t pos = fp->f_pos

__vfs_write：內部包含了 file_start_write 和 file_end_write，利用 file->f_op->write 寫入檔案

v9fs_file_write：將資料寫入 fs/9p/位置

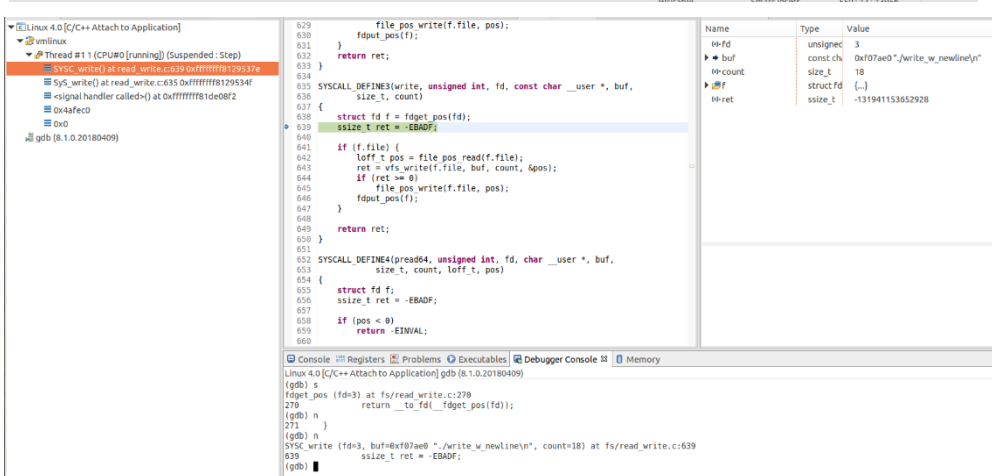
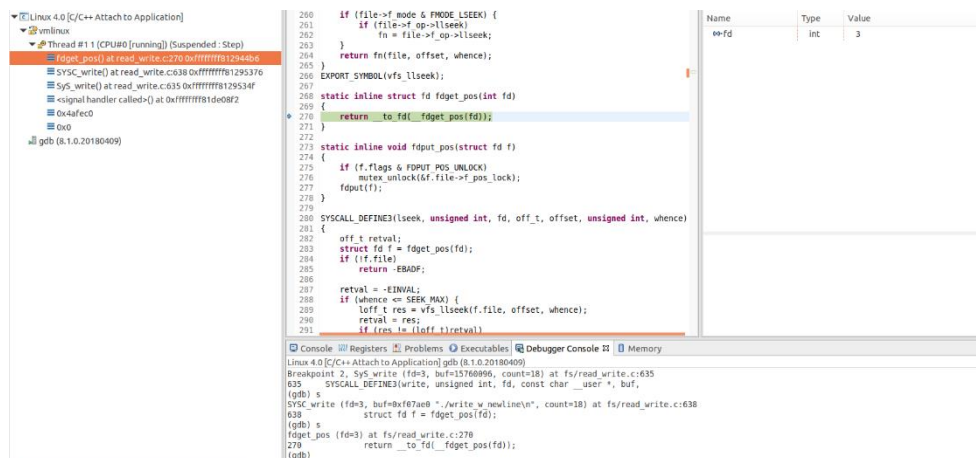
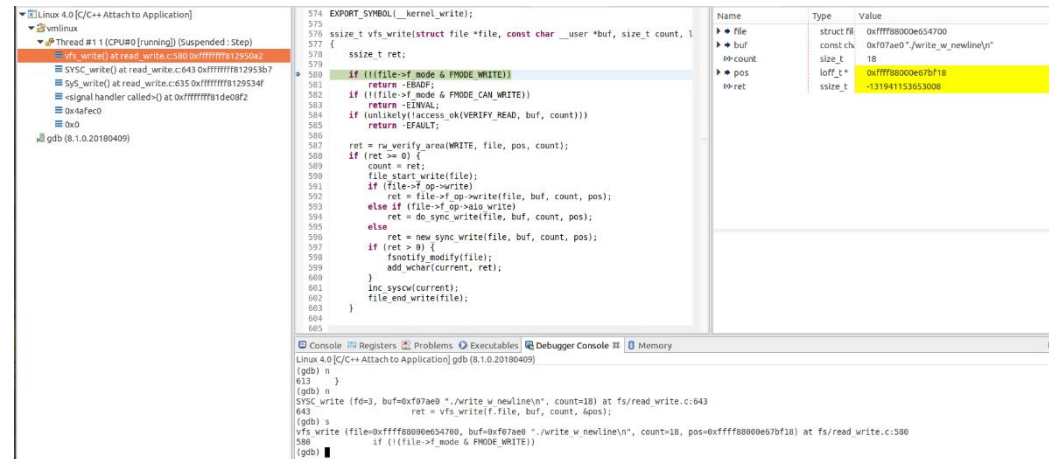
*file_inode：打開檔案

new_sync_write：初始化 kiocb,並調用 call_write_iter

__always_inline：inline 是行內函式，是指將此 function 的內容整合至呼叫此函式位置

file_end_write：停止檔案書寫

丙.



Linux 4.0 [C/C++ Attach to Application]

vmlinux

Thread #11 (CPU#0) (running) (Suspended: Step)

file_start_write() at fs.h:2381 0xffffffff8129361c

vfs_write() at read_write.c:590 0xffffffff81295149

SYSC_write() at read_write.c:643 0xffffffff812953b7

SYs_write() at read_write.c:635 0xffffffff8129534f

<signal handler called> at 0xffffffff81de08f2

0x4afec0

0x0

gdb (8.1.0.20180409)

2381

if (!S_ISREG(file_inode(file)->i_mode))

return;

__sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, true);

2384

2385

static inline bool file_start_write_trylock(struct file *file)

{

if (!S_ISREG(file_inode(file)->i_mode))

return true;

return __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, false);

2391

2392

static inline void file_end_write(struct file *file)

{

if (!S_ISREG(file_inode(file)->i_mode))

return;

__sb_end_write(file_inode(file)->i_sb, SB_FREEZE_WRITE);

2396

2397

2398

2399

2400 /*

2401 * get_write_access() gets write permission for a file.

2402 * put_write_access() releases this write permission.

2403 * This is used for regular files.

2404 * We cannot support write (and maybe mmap read-write shared) accesses and

2405 * MAP_DENYWRITE mappings simultaneously. The i_writecount field of an inode

2406 * can have the following values:

2407 * 0: no writers, no VM_DENYWRITE mappings

2408 * < 0: (-i_writecount) VM areas structs with VM_DENYWRITE set exist

2409 * > 0: (i_writecount) users are writing to the file.

2410 *

2411 * Normally we operate on that counter with atomic (inc,dec) and it's safe

2412 * except for the cases where we don't hold i_writecount yet. Then we need to

Console

Registers

Problems

Executables

Debugger Console

Memory

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

588

if (ret >= 0) {

(gdb) n

589

count = ret;

(gdb) n

590

file_start_write(file);

(gdb) s

file_start_write (file=0xfffff8000e654700) at include/linux/fs.h:2381

2381

if (!S_ISREG(file_inode(file)->i_mode))

(gdb) █

2381

if (!S_ISREG(file_inode(file)->i_mode))

return;

__sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, true);

2384

2385

static inline bool file_start_write_trylock(struct file *file)

{

if (!S_ISREG(file_inode(file)->i_mode))

return true;

return __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, false);

2391

2392

static inline void file_end_write(struct file *file)

{

if (!S_ISREG(file_inode(file)->i_mode))

return;

__sb_end_write(file_inode(file)->i_sb, SB_FREEZE_WRITE);

2396

2397

2398

2399

2400 /*

2401 * get_write_access() gets write permission for a file.

2402 * put_write_access() releases this write permission.

2403 * This is used for regular files.

2404 * We cannot support write (and maybe mmap read-write shared) accesses and

2405 * MAP_DENYWRITE mappings simultaneously. The i_writecount field of an inode

2406 * can have the following values:

2407 * 0: no writers, no VM_DENYWRITE mappings

2408 * < 0: (-i_writecount) VM areas structs with VM_DENYWRITE set exist

2409 * > 0: (i_writecount) users are writing to the file.

2410 *

2411 * Normally we operate on that counter with atomic (inc,dec) and it's safe

2412 * except for the cases where we don't hold i_writecount yet. Then we need to

Console

Registers

Problems

Executables

Debugger Console

Memory

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

588

if (ret >= 0) {

(gdb) n

589

count = ret;

(gdb) n

590

file_start_write(file);

(gdb) s

file_start_write (file=0xfffff8000e654700) at include/linux/fs.h:2381

2381

if (!S_ISREG(file_inode(file)->i_mode))

(gdb) █

Name	Type	Value
file	struct file	0xfffff8000e654700

kernel 4.0 [C/C++ Attach to Application]

vmlinux

Thread #11 (CPU#0) (running) (Suspended: Step)

v9fs_file_write() at v9fs

vfs_write() at read_write

SYSC_write() at read_write

SYs_write() at read_write

<signal handler called>

0x4495d1

0x221c025

0x24ea025

0x0

gdb (8.1.0.20180409)

497

* v9fs file write - write to a file

498

* @filp: file pointer to write

499

* @data: data buffer to write data from

500

* @count: size of buffer

501

* @offset: offset at which to write data

502

503

static ssize_t

v9fs_file_write(struct file *filp, const char __user * data,

size_t count, loff_t *offset)

{

508

ssize_t retval = 0;

loff_t origin = *offset;

510

511

512

retval = generic_write_checks(filp, &origin, &count, 0);

513

if (retval)

goto out;

515

516

retval = -EINVAL;

517

if ((ssize_t) count < 0)

goto out;

518

519

retval = 0;

520

if (!count)

goto out;

521

522

523

524

525

526

/* update offset on successful write */

527

if (retval > 0)

*offset = origin;

528

Console

Registers

Problems

Executables

Debugger Console

Memory

kernel 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

(gdb) n

494

}

(gdb) n

v9fs_file_write (filp=0xfffff8000e6a3f100, data=0x122ed50 "HelloWorld!!", count=25, offset=0xfffff8000e6a1bf18) at fs/9p/vfs_file.c:527

527

if (retval > 0)

(gdb) █

Name	Type	Value
filp	struct file	0xfffff8000e6a3f100
data	const char *	0x122ed50 "HelloWorld!!"
count	size_t	25
offset	loff_t *	0xfffff8000e6a1bf18
retval	ssize_t	25
origin	loff_t	25

Linux 4.0 [C/C++ Attach to Application]

vmlinux

Thread #11 (CPU#0) (running) (Suspended: Step)

fdp_pos_read() at read_write.c:632 0xffffffff81295333

SYSC_write() at read_write.c:642 0xffffffff812953b6

SYs_write() at read_write.c:635 0xffffffff8129534f

<signal handler called> at 0xffffffff81de08f2

0x4afec0

0x0

gdb (8.1.0.20180409)

603

file_end_write(file);

604

return ret;

605

606

EXPORT_SYMBOL(vfs_write);

607

608

static inline loff_t file_pos_read(struct file *file)

{

611

return file->f_pos;

612

613

614

static inline void file_pos_write(struct file *file, loff_t pos)

{

616

file->f_pos = pos;

617

618

619

620

SYSCALL_DEFINE3(read, unsigned int, fd, char __user *, buf, size_t, count)

{

621

struct fd f = fdget_pos(fd);

622

ssize_t ret = -EBADF;

623

624

if (!f.file) {

625

loff_t pos = file_pos_read(f.file);

626

ret = vfs_read(f.file, buf, count, &pos);

627

if (ret >= 0)

file_pos_write(f.file, pos);

628

fdput_pos(f);

629

return ret;

630

631

632

633

Console

Registers

Problems

Executables

Debugger Console

Memory

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

639

ssize_t ret = -EBADF;

(gdb) n

641

if (!f.file) {

(gdb) n

642

loff_t pos = file_pos_read(f.file);

(gdb) s

file_pos_read (file=0xfffff8000e654700) at fs/read_write.c:612

612

return file->f_pos;

(gdb) █

Name	Type	Value
file	struct file	0xfffff8000e654700

Linux 4.0 [C/C++ Attach to Application]

- vmlinux
 - Thread #11 (CPU#0) (running) (Suspended: Step)
 - file_inode() at fs.h:1.963 0xffffffff1293ba9
 - file_start_write() at fs.h:2.383 0xffffffff1293c46
 - vfs_write() at read_write.c:590 0xffffffff1295149
 - SVSC_write() at read_write.c:643 0xffffffff12953b7
 - Sys_write() at read_write.c:635 0xffffffff129534f
 - <signal handler called>() at 0xffffffff1de08f2
 - 0x4afcd
 - 0x0

```

1953 extern bool fs_fully_visible(struct file_system_type *);
1954
1955 extern int current_umask(void);
1956
1957 extern void ihold(struct inode * inode);
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
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_t,
1977                                /*
1978     * Candidates for mandatory locking have the setgid bit set
1979     * but no group execute bit - an otherwise meaningless combination.
1980     */
1981                                static inline int __mandatory_lock(struct inode *ino)
1982                                {
1983
1984

```

Name	Type	Value
f	const struct file	0xffffffff1293ba9

Console
Registers
Problems
Executables
Debugger Console
Memory

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

```

(gdb) s
file_start_write (file=0xfffff8000e54700) at include/linux/fs.h:2381
2381     if (!ISREG(file_inode(file)->i_mode))
(gdb) n
__sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, true);
2383
(gdb) s
file_inode (f=0xfffff8000e54700) at include/linux/fs.h:1963
1963     return f->f_inode;
(gdb)

```

Linux 4.0 [C/C++ Attach to Application]

- vmlinux
 - Thread #11 (CPU#0) (running) (Suspended: Step)
 - new_sync_write() at read_write.c:526 0xffffffff1294e00
 - vfs_write() at read_write.c:592 0xffffffff1295178
 - SVSC_write() at read_write.c:643 0xffffffff12953b7
 - Sys_write() at read_write.c:635 0xffffffff129534f
 - <signal handler called>() at 0xffffffff1de08f2
 - 0x4afcd
 - 0x0

```

516 if (!EIOCBQUEUED == ret)
517     ret = wait_on_sync_kiocb(&kiocb);
518     *ppos = kiocb.ki_pos;
519     return ret;
520
521 EXPORT_SYMBOL(do_sync_write);
522
523 ssize_t new_sync_write(struct file *filp, const char __user *buf, size_t len)
524 {
525     struct iov_iter iter;
526     struct kiocb kiocb;
527     struct iov_iter iter;
528     ssize_t ret;
529
530     init_sync_kiocb(&kiocb, filp);
531     kiocb.ki_pos = *ppos;
532     kiocb.ki_nbytes = len;
533     iov_iter_init(&iter, WRITE, &iov, 1, len);
534
535     ret = filp->op->write_iter(&kiocb, &iter);
536     if (!EIOCBQUEUED == ret)
537         ret = wait_on_sync_kiocb(&kiocb);
538     *ppos = kiocb.ki_pos;
539     return ret;
540
541 EXPORT_SYMBOL(new_sync_write);
542
543 ssize_t __kernel_write(struct file *file, const char __user *buf, size_t count, lof
544 {
545     mm_segment_t old_fs;
546
547

```

Name	Type	Value
filp	struct file	0xfffff8000e54700
buf	const char *	0xf07ae0 "/write_w_newline\n"
len	size_t	18
ppos	loff_t *	0xfffff8000e67bf18
kiocb	struct kiocb	[...]
iter	struct iov_iter	[...]
ret	ssize_t	-131941153814794

Console
Registers
Problems
Executables
Debugger Console
Memory

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

```

(gdb) n
vfs_write (file=0xfffff8000e54700, buf=0xf07ae0 "/write_w_newline\n", count=18, pos=0xfffff8000e67bf18) at fs/read_write.c:591
591     if (file->op->write)
(gdb) n
ret = file->op->write(file, buf, count, pos);
(gdb) s
new_sync_write (filp=0xfffff8000e54700, buf=0xf07ae0 "/write_w_newline\n", len=18, ppos=0xfffff8000e67bf18) at fs/read_write.c:526
526     struct iov_iter iter = { .iov_base = (void __user *)buf, .iov_len = len };
(gdb)

```

Linux 4.0 [C/C++ Attach to Application]

- vmlinux
 - Thread #11 (CPU#0) (running) (Suspended: Step)
 - fsnotify_modify() at fsnotify.h:214 0xffffffff1293cfe
 - vfs_write() at read_write.c:590 0xffffffff1295149
 - SVSC_write() at read_write.c:643 0xffffffff12953b7
 - Sys_write() at read_write.c:635 0xffffffff129534f
 - <signal handler called>() at 0xffffffff1de08f2
 - 0x4afcd
 - 0x0

```

214 struct path *path = &file->f_path;
215 struct inode *inode = file_inode(file);
216 __u32 mask = FS_MODIFY;
217
218 if (S_ISDIR(inode->i_mode))
219     mask |= FS_ISDIR;
220
221 if (!file->f_mode & FMODE_WRITE) {
222     fsnotify_parent(path, NULL, mask);
223     fsnotify(inode, mask, path, FSNOTIFY_EVENT_PATH, NULL, 0);
224 }
225
226 /*
227 * fsnotify_open - file was opened
228 */
229 static inline void fsnotify_open(struct file *file)
230 {
231     struct path *path = &file->f_path;
232     struct inode *inode = file_inode(file);
233     __u32 mask = FS_OPEN;
234
235     if (S_ISDIR(inode->i_mode))
236         mask |= FS_ISDIR;
237
238     fsnotify_parent(path, NULL, mask);
239     fsnotify(inode, mask, path, FSNOTIFY_EVENT_PATH, NULL, 0);
240 }
241
242 /*
243 * fsnotify_close - file was closed
244 */
245

```

Name	Type	Value
file	struct file	0xfffff8000e54700
path	struct path	0x206c1c_stack_union+518
inode	struct inode	0x10c1c_stack_union+16
mask	__u32	4294936576

Console
Registers
Problems
Executables
Debugger Console
Memory

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

```

(gdb) n
vfs_write (file=0xfffff8000e54700, buf=0xf07ae0 "/write_w_newline\n", count=18, pos=0xfffff8000e67bf18) at fs/read_write.c:597
597     if (ret > 0) {
(gdb) n
fsnotify_modify(file);
(gdb) s
fsnotify_modify (file=0xfffff8000e54700) at include/linux/fsnotify.h:214
214     struct path *path = &file->f_path;
(gdb)

```

Linux 4.0 [C/C++ Attach to Application]

Thread #11 (CPU#0) [running] (Suspended: Step)

get_current() at sched.h:1408

fs_write() at read_write.c:599

SYSC_write() at read_write.c:643

sys_write() at read_write.c:635

signal handler called=() at 0xffffffff1de08f2

0x0

0x4fec0

gdb (8.1.0.20180409)

```
1 #ifndef __ASM_X86_CURRENT_H
2 #define __ASM_X86_CURRENT_H
3
4 #include <linux/compiler.h>
5 #include <asm/percpu.h>
6
7 #ifndef __ASSEMBLY__
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
```

Name Type Value

pfo_ret_ struct ta 0x10-dirq_stack_union+16>

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

(gdb) n

225 }

(gdb) n

fs_write (file=0xfffff8000e654700, buf=0xf07ae0 "/write_w_newline\n", count=18, pos=0xfffff8000e67bf18) at fs/read_write.c:599

599 add_uchar(current, ret);

(gdb) s

get_current () at ./arch/x86/include/asm/current.h:14

14 return this_cpu_read_stable(current_task);

(gdb)

Linux 4.0 [C/C++ Attach to Application]

Thread #11 (CPU#0) [running] (Suspended: Step)

inc_syscow() at sched.h:3039

fs_write() at read_write.c:601

SYSC_write() at read_write.c:643

sys_write() at read_write.c:635

signal handler called=() at 0xffffffff1de08f2

0x0

0x4fec0

gdb (8.1.0.20180409)

```
3836 static inline void inc_syscow(struct task_struct *tsk)
3837 {
3838     tsk->syscow++;
3839 }
3840
3841 #else
3842 static inline void add_rchar(struct task_struct *tsk, ssize_t amt)
3843 {
3844 }
3845
3846 static inline void add_wchar(struct task_struct *tsk, ssize_t amt)
3847 {
3848 }
3849
3850 static inline void inc_syscr(struct task_struct *tsk)
3851 {
3852 }
3853
3854 static inline void inc_syscw(struct task_struct *tsk)
3855 {
3856 }
3857
3858 #endif
3859
3860 #ifndef TASK_SIZE_OF
3861 #define TASK_SIZE_OF(tsk) TASK_SIZE
3862 #endif
3863
3864 #ifndef CONFIG_MEMCG
3865 extern void mm_update_next_owner(struct mm_struct *mm);
3866 #else
3867 static inline void mm_update_next_owner(struct mm_struct *mm)
3868 {
3869 }
3870
```

Name Type Value

tsk struct ta 0xfffff8000e641b00

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

get_current () at ./arch/x86/include/asm/current.h:14

14 return this_cpu_read_stable(current_task);

(gdb) n

fs_write (file=0xfffff8000e654700, buf=0xf07ae0 "/write_w_newline\n", count=18, pos=0xfffff8000e67bf18) at fs/read_write.c:601

601 inc_syscow(current);

(gdb) s

inc_syscow (tsk=0xfffff8000e641b00) at include/linux/sched.h:3839

3839 tsk->syscow++;

(gdb)

Linux 4.0 [C/C++ Attach to Application]

Thread #11 (CPU#0) [running] (Suspended: Step)

file_end_write() at fs.h:395

fs_write() at read_write.c:602

SYSC_write() at read_write.c:643

sys_write() at read_write.c:635

signal handler called=() at 0xffffffff1de08f2

0x0

0x4fec0

gdb (8.1.0.20180409)

```
2374 static inline bool execute_ok(struct inode *inode)
2375 {
2376     return (inode->i_mode & S_IWUGO) || S_ISDIR(inode->i_mode);
2377 }
2378
2379 static inline void file_start_write(struct file *file)
2380 {
2381     if (IS_ISSREG(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 (IS_ISSREG(file_inode(file)->i_mode))
2389         return true;
2390     return __sb_start_write(file_inode(file)->i_sb, SB_FREEZE_WRITE, false);
2391 }
2392
2393 static inline void file_end_write(struct file *file)
2394 {
2395     if (IS_ISSREG(file_inode(file)->i_mode))
2396         return;
2397     __sb_end_write(file_inode(file)->i_sb, SB_FREEZE_WRITE);
2398 }
2399
2400 /*
2401 * get write access() gets write permission for a file.
2402 * put write access() releases this write permission.
2403 * This is used for regular files.
2404 * We cannot support write (and maybe mmap read-write shared) accesses and
2405 * MAP_DENYWRITE mappings simultaneously. The i_writcount field of an inode
2406
```

Name Type Value

File struct fil 0xfffff8000e654700

Linux 4.0 [C/C++ Attach to Application] gdb (8.1.0.20180409)

(gdb) n

3840 }

(gdb) n

fs_write (file=0xfffff8000e654700, buf=0xf07ae0 "/write_w_newline\n", count=18, pos=0xfffff8000e67bf18) at fs/read_write.c:602

602 file_end_write(file);

(gdb) s

file_end_write (file=0xfffff8000e654700) at include/linux/fs.h:2395

2395 if (IS_ISSREG(file_inode(file)->i_mode))

(gdb)

丁.

目前追蹤到現在，還是沒有找到。