SPINLOCK [Report]

Group Members:
Sohaib Ashraf **(20k-0488)**
Khawaja Abdullah **(20k-0385)**
M.Shahzaib **(20k-1067)**

Teacher: **Sir Ghufran**
Lab**: Sir Amin Sadiq**

Date: 24th May 2022

Introduction:

Spinlock is a lock that causes a thread trying to acquire it to simply wait in a loop ("spin") while repeatedly checking whether the lock is available. Since the thread remains active but is not performing a useful task, the use of such a lock is a kind of busy waiting.

In our project, we are using a type of spinlock called **Read-Write** spinlock. For e.g we have 4 reader threads (thread-2 to thread-5). Now all four threads want to read the data from the variable at the same time. What will happen in this situation if we use a simple spinlock? Let’s assume thread-2 got the lock while other reading threads are trying hard for the lock. Those other reading threads are simply wasting time to take lock since the variable won’t change. That means performance will be reduced.

Hence with **read-write** spinlock, let's assume if thread-2 will take the read lock and read the data. And other reading threads also will take the read lock without spinning (blocking) and read the data. Because no one is writing. So what about the performance now? There is no waiting between reading operations. Thus, in this case, a read-write spinlock is useful.
So, If multiple threads require read access to the same data, there is no reason why they should not be able to execute simultaneously. Spinlocks don’t differentiate between read and read/write access.

In Read Write spinlock multiple readers are permitted at the same time but only one writer. (i.e) If a writer has the lock, no reader is allowed to enter the critical section. If only a reader has the lock, then multiple readers are permitted in the critical section.

Where to use Read Write Spinlock?

- If you are only reading the data then you take read lock
- If you are writing then go for a write lock

How we implemented:

We made a system call to show implementation of spin lock. We used two kernel threads. In the first thread, Writer is locked and the global variable increments to 1. No reader is allowed to enter. In thread2, readers keep the lock and the writer can not enter.

STEPS:

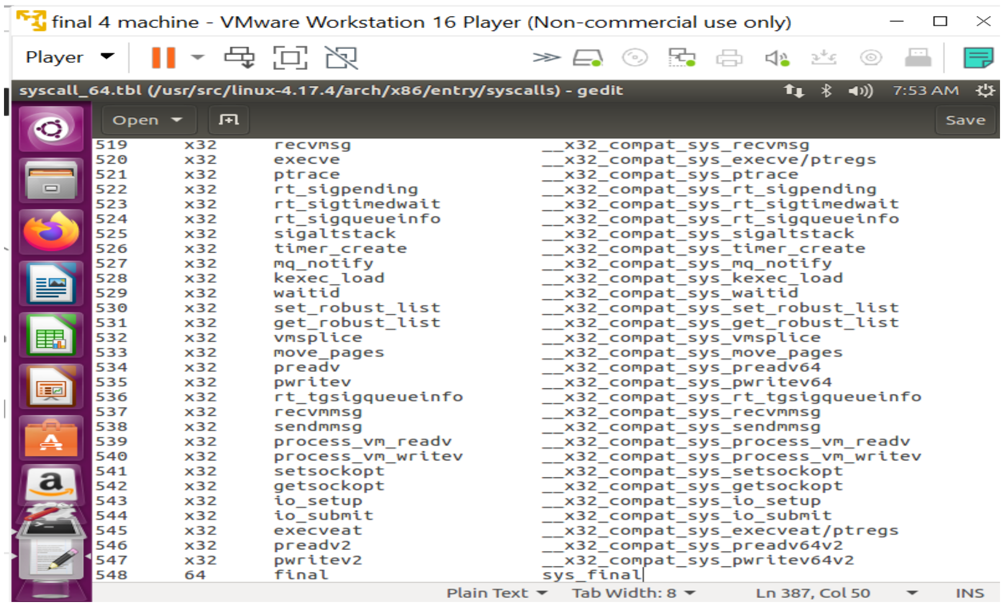
1- Create a directory named final/ and made a file final.c

2- We wrote our **read-write spinlock code** in the final.c file.

3- Create a “Makefile” in the final directory and add the following line to it:
obj-y := final.o
This is to ensure that the hello.c file is compiled and included in the kernel source code.

4- Go back to the parent dir (linux-4.17.4) and open “Makefile”
Add final/’ to the end of this line:
core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ final/
Edit the extraversion to our roll name

5- Now go to arch/x86/entry/syscalls/ and open **syscall_64.tbl**
Go to the last of the document and add a new line like so:
548 64 final sys_final


final 4 machine - VMware Workstation 16 Player (Non-commercial use only)

syscall_64.tbl (/usr/src/linux-4.17.4/arch/x86/entry/syscalls) - gedit

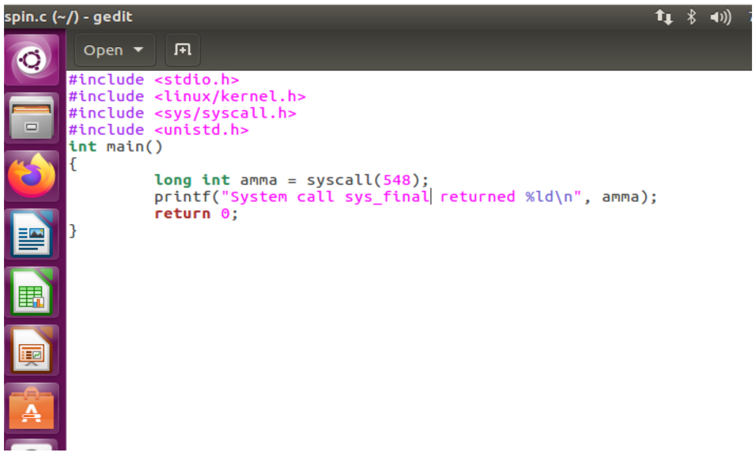
```
519     x32     recvmsg                 __x32_compat_sys_recvmsg
520     x32     execve                  __x32_compat_sys_execve/ptregs
521     x32     ptrace                  __x32_compat_sys_ptrace
522     x32     rt_sigpending           __x32_compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         __x32_compat_sys_rt_sigtimedwait
524     x32     rt_sigqueueinfo         __x32_compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             __x32_compat_sys_sigaltstack
526     x32     timer_create            __x32_compat_sys_timer_create
527     x32     mq_notify               __x32_compat_sys_mq_notify
528     x32     kexec_load              __x32_compat_sys_kexec_load
529     x32     waitid                  __x32_compat_sys_waitid
530     x32     set_robust_list         __x32_compat_sys_set_robust_list
531     x32     get_robust_list         __x32_compat_sys_get_robust_list
532     x32     vmsplice                __x32_compat_sys_vmsplice
533     x32     move_pages              __x32_compat_sys_move_pages
534     x32     preadv                  __x32_compat_sys_preadv64
535     x32     pwritev                 __x32_compat_sys_pwritev64
536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                __x32_compat_sys_recvmmsg
538     x32     sendmmsg                __x32_compat_sys_sendmmsg
539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
541     x32     setsockopt              __x32_compat_sys_setsockopt
542     x32     getsockopt              __x32_compat_sys_getsockopt
543     x32     io_setup                __x32_compat_sys_io_setup
544     x32     io_submit               __x32_compat_sys_io_submit
545     x32     execveat                __x32_compat_sys_execveat/ptregs
546     x32     preadv2                 __x32_compat_sys_preadv64v2
547     x32     pwritev2                __x32_compat_sys_pwritev64v2
548     64      final                   sys_final
```

6- Go to the linux-4.17.4/ directory and type the following commands:
cd include/linux/
gedit syscalls.h
Add the following line to the end of the document before the #endif statement:
asmlinkage long sys_final(void);

7- Now we compile the kernel using **sudo make.**

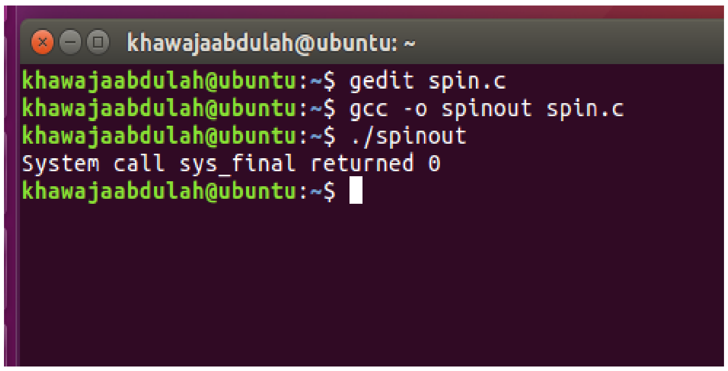
8- Run the following command in your terminal:
sudo make modules_install install

9- Now we restart our system. Go to your home(~) directory using the following commands and create a spin.c file.
We will call our system call from there.



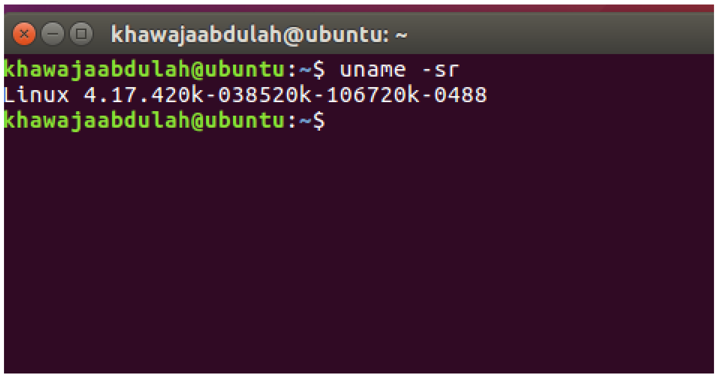
```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main()
{
        long int amma = syscall(548);
        printf("System call sys_final returned %ld\n", amma);
        return 0;
}
```

10- After compile our run, spin.c programs returns 0 which means our spinlock system call is working



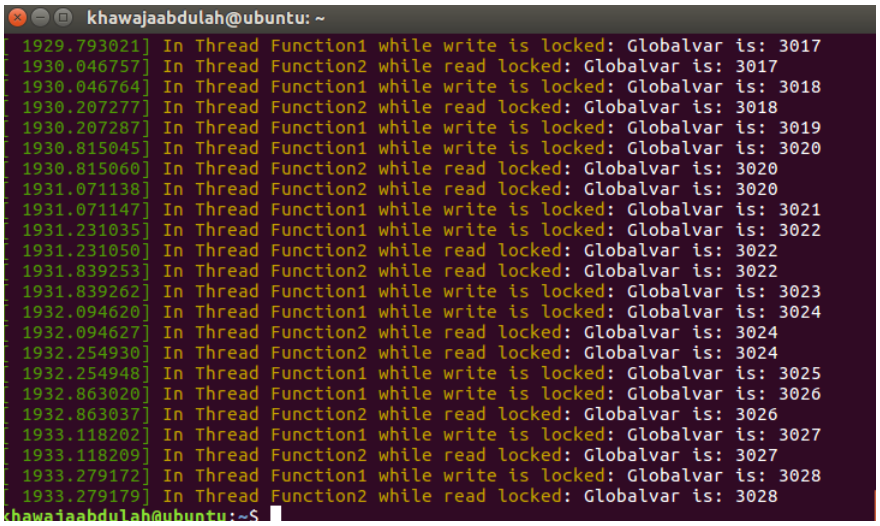
```
khawajaabdulah@ubuntu:~$ gedit spin.c
khawajaabdulah@ubuntu:~$ gcc -o spinout spin.c
khawajaabdulah@ubuntu:~$ ./spinout
System call sys_final returned 0
khawajaabdulah@ubuntu:~$
```

11- Uname -sr command shows our roll numbers



```
khawajaabdulah@ubuntu:~$ uname -sr
Linux 4.17.420k-038520k-106720k-0488
khawajaabdulah@ubuntu:~$
```

12- Write dmesg command. It will return our results. When read is locked, the value of the shared variable is read. When write is locked, value of shared variable is increased by 1, preventing any race conditions.



```
[ 1929.793021] In Thread Function1 while write is locked: Globalvar is: 3017
[ 1930.046757] In Thread Function2 while read locked: Globalvar is: 3017
[ 1930.046764] In Thread Function1 while write is locked: Globalvar is: 3018
[ 1930.207277] In Thread Function2 while read locked: Globalvar is: 3018
[ 1930.207287] In Thread Function1 while write is locked: Globalvar is: 3019
[ 1930.815045] In Thread Function1 while write is locked: Globalvar is: 3020
[ 1930.815060] In Thread Function2 while read locked: Globalvar is: 3020
[ 1931.071138] In Thread Function2 while read locked: Globalvar is: 3020
[ 1931.071147] In Thread Function1 while write is locked: Globalvar is: 3021
[ 1931.231035] In Thread Function1 while write is locked: Globalvar is: 3022
[ 1931.231050] In Thread Function2 while read locked: Globalvar is: 3022
[ 1931.839253] In Thread Function2 while read locked: Globalvar is: 3022
[ 1931.839262] In Thread Function1 while write is locked: Globalvar is: 3023
[ 1932.094620] In Thread Function1 while write is locked: Globalvar is: 3024
[ 1932.094627] In Thread Function2 while read locked: Globalvar is: 3024
[ 1932.254930] In Thread Function2 while read locked: Globalvar is: 3024
[ 1932.254948] In Thread Function1 while write is locked: Globalvar is: 3025
[ 1932.863020] In Thread Function1 while write is locked: Globalvar is: 3026
[ 1932.863037] In Thread Function2 while read locked: Globalvar is: 3026
[ 1933.118202] In Thread Function1 while write is locked: Globalvar is: 3027
[ 1933.118209] In Thread Function2 while read locked: Globalvar is: 3027
[ 1933.279172] In Thread Function1 while write is locked: Globalvar is: 3028
[ 1933.279179] In Thread Function2 while read locked: Globalvar is: 3028
khawajaabdulah@ubuntu:~$
```