



Assignment2 Report

Task

Question 1: 6 marks, about 1 to 2 hours Answer the following questions:

- a. Ftrace uses memory to keep the trace record. Before starting a new trace, the old traces are still kept in the memory by ftrace has to be emptied. How can this be done? (1 mark)
- b. What is the command to change the maximum size of the trace file of ftrace? (1 mark)
- c. Assuming you are changing the kernel code to insert a printk-like code to output hello world in ftrace and the message given by the print code is the only thing you want to trace, show your code (with comments) and also the tracer that should use in ftrace. You should provide a shell script that will run the tracing. (2 marks)
- d. Use the function tracer to trace the kernel for about a few seconds. Give the screenshots of the first 20 lines of the trace file and analyse the basic structure of it. (2 marks)

Question2: 6 marks, about 30 minutes to 1 hour

Use the function_graph tracer in ftrace to trace the function call stack of three kernel functions, vfs_read, lookup_dcache and task_fork_fair and set the max function call depth to record as 10. Give the complete commands you use and analyse the trace result. Note: the three functions should be traced at the same time.

Question 3: 8 marks, about 2 to 3 hours

- a. Finish the steps for adding a kernel function into kernel in the tutorial. Supply the two C files. (2 marks)
- b. What is the full path of the system call table? Show the line you added in the system call table. (2 marks)
- c. Show the screenshot of the “dmesg | tail” containing the print message of your new kernel function. (2 marks)
- d. Give the first 20 lines of the trace file of ftrace and analyse it. (2 marks)

Additional exercises (5 marks each)

1. Using an example, show why deletion is not supported in the bit-vector form of the Bloom filter. Can you suggest a modification that would allow it?
2. Consider the following simple C program compiled for a 32 bit x86 Linux machine (64 bit just mean longer addresses with more zeroes to write down):
 - (i). Show the resultant binary of the function “foo” after the first THREE (3) relocation is applied, assuming that the linker decides to allocate globvar to the address 0x4fac0 and foo to address 0x8082ab0.
 - (ii). What do you think the 4-th relocation record is for?
 - (iii). Show what the GOT and PLT may look like at runtime (before the program begins execution, assuming lazy binding). Write down your assumptions about where things (such as the function printf) are allocated.

Task

Question 1: 6 marks, about 1 to 2 hours Answer the following questions:

a. Ftrace uses memory to keep the trace record. Before starting a new trace, the old traces are still kept in the memory by ftrace has to be emptied. How can this be done? (1 mark)

solution:

- All the following commands are execute in the ftrace directory (`cd /sys/kernel/tracing`)
- As we can seen, before starting a new trace, the old traces are still kept in the memory.

```

root@dingfan: /sys/kernel/debug/tracing# cat trace | head -20
# tracer: function_graph
#
# CPU    DURATION    FUNCTION CALLS
#  |      |          |      |
0)  0.592 us  |      |      |      |
0)  0.500 us  |      |      |      |
0)  0.125 us  |      |      |      |
0)  0.112 us  |      |      |      |
0)          |      |      |      |
0)          |      |      |      |
0)  0.105 us  |      |      |      |
0)  0.321 us  |      |      |      |
0)  0.537 us  |      |      |      |
0)  7.776 us  |      |      |      |
0)          |      |      |      |
0)  0.113 us  |      |      |      |
0)  0.116 us  |      |      |      |
0)          |      |      |      |
0)          |      |      |      |
0)          |      |      |      |
0)          |      |      |      |
0)          |      |      |      |
0)          |      |      |      |

```

- In order to clear the old traces, it need 2 commands:
 - `echo 0 > tracing_on` : use this command to disable the tracer
 - `echo 0 > trace` : use this command to clear the old traces
- by command `cat trace` we can see that the trace file is empty

```

root@dingfan: /sys/kernel/debug/tracing
0)      |      |      |      |      |      |      |      |      |      |
0) 0.105 us |      |      |      |      |      |      |      |      |      |
0) 0.321 us |      |      |      |      |      |      |      |      |      |
0) 0.537 us |      |      |      |      |      |      |      |      |      |
0) 7.776 us |      |      |      |      |      |      |      |      |      |
0)      |      |      |      |      |      |      |      |      |      |
0) 0.113 us |      |      |      |      |      |      |      |      |      |
0) 0.116 us |      |      |      |      |      |      |      |      |      |
0)      |      |      |      |      |      |      |      |      |      |
0) 0.128 us |      |      |      |      |      |      |      |      |      |
0) 0.110 us |      |      |      |      |      |      |      |      |      |
root@dingfan:/sys/kernel/debug/tracing# echo 0 > tracing_on
root@dingfan:/sys/kernel/debug/tracing# echo 0 > trace
root@dingfan:/sys/kernel/debug/tracing# cat trace
# tracer: function_graph
#
# CPU  DURATION      FUNCTION CALLS
# |  |  |      |  |  |  |
root@dingfan:/sys/kernel/debug/tracing#

```

b. What is the command to change the maximum size of the trace file of ftrace? (1 mark)

solution:

- The file `buffer_size_kb` is used to modify the size of the internal trace buffers.
- `cat buffer_size_kb` to read the current value
- If we want to change the the maximum size of the trace file of ftrace, use command `echo a number` to the file like `echo 1024 > buffer_size_kb`
- After execute the command `echo 1024 > buffer_size_kb`, we can use the command `cat buffer_size_kb` to check the maximum size of the trace file; it is 1024 now

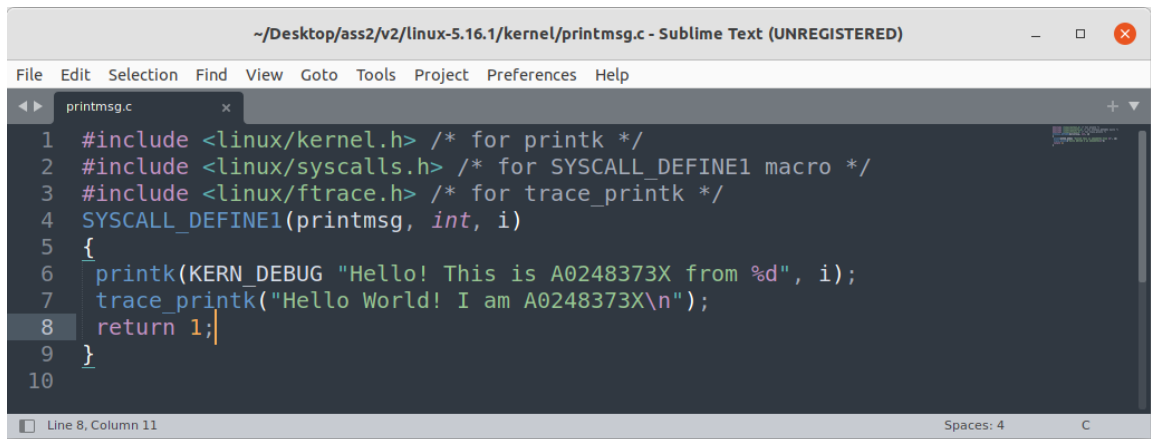
```
root@dingfan: /sys/kernel/debug/tracing
0)      |      rebalance_domains() {
0) 0.113 us |      __rcu_read_lock();
0) 0.116 us |      __msecs_to_jiffies();
0)      |      load_balance() {
0)      |      idle_cpu();
0) 0.128 us |      group_balance_cpu();
0) 0.110 us |
root@dingfan:/sys/kernel/debug/tracing# echo 0 > tracing_on
root@dingfan:/sys/kernel/debug/tracing# echo 0 > trace
root@dingfan:/sys/kernel/debug/tracing# cat trace
# tracer: function_graph
#
# CPU DURATION FUNCTION CALLS
# | | | | |
root@dingfan:/sys/kernel/debug/tracing# cat buffer_size_kb
1408
root@dingfan:/sys/kernel/debug/tracing# echo 1024 > buffer_size_kb
root@dingfan:/sys/kernel/debug/tracing# cat buffer_size_kb
1024
root@dingfan:/sys/kernel/debug/tracing#
```

c. Assuming you are changing the kernel code to insert a printk-like code to output hello world in ftrace and the message given by the print code is the only thing you want to trace, show your code (with comments) and also the tracer that should use in ftrace. You should provide a shell script that will run the tracing. (2 marks)

solution:

- In this question we use the new kernel function add to kernel in the tutorial as the changed kernel function
- Following is the new kernel function(which has been added to the kernel; the new kernel has been recompiled and booted as the linux system kernel)

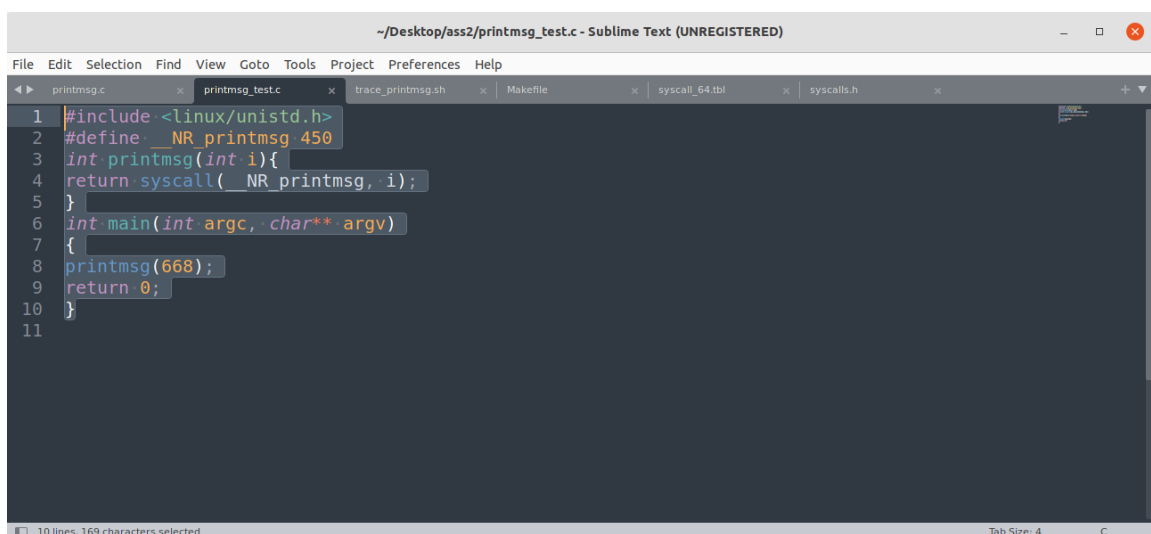
```
#include <linux/kernel.h> /* for printk */
#include <linux/syscalls.h> /* for SYSCALL_DEFINE1 macro */
#include <linux/ftrace.h> /* for trace_printk */
SYSCALL_DEFINE1(printmsg, int, i)
{
    printk(KERN_DEBUG "Hello! This is A0248373X from %d", i);
    trace_printk("Hello World! I am A0248373X\n");
    return 1;
}
```



```
~/Desktop/ass2/v2/linux-5.16.1/kernel/printmsg.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
printmsg.c
1 #include <linux/kernel.h> /* for printk */
2 #include <linux/syscalls.h> /* for SYSCALL_DEFINE1 macro */
3 #include <linux/ftrace.h> /* for trace_printk */
4 SYSCALL_DEFINE1(printmsg, int, i)
5 {
6     printk(KERN_DEBUG "Hello! This is A0248373X from %d", i);
7     trace_printk("Hello World! I am A0248373X\n");
8     return 1;
9 }
10
Line 8, Column 11 Spaces: 4 C
```

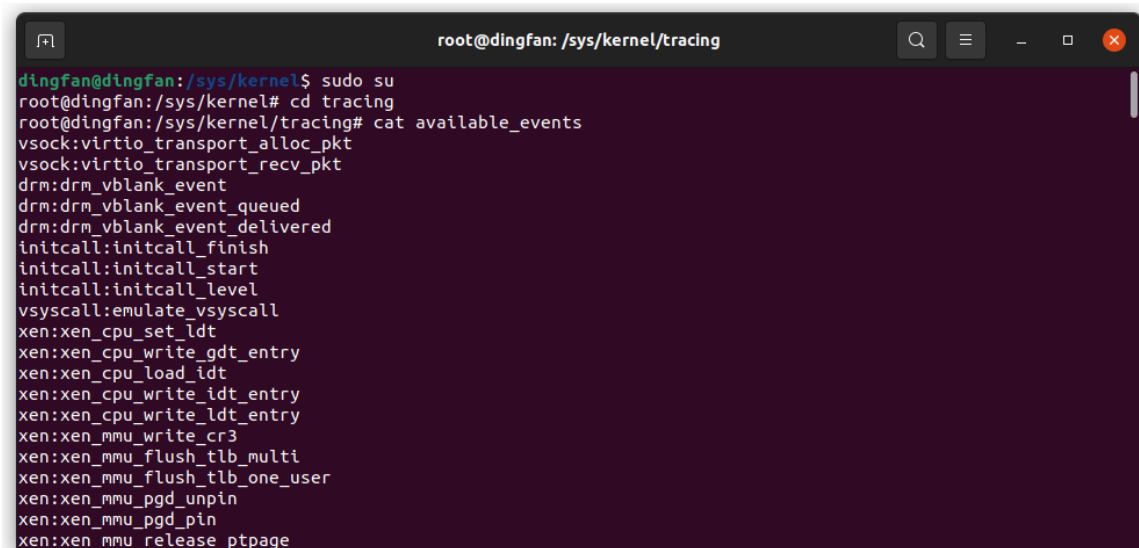
- We need a user mode program to call the new special kernel function
 - the user mode test program is as followed:
 - use command `gcc printmsg_test.c -o printmsg_test.o` compile the test program into the directory `"~/Desktop/ass2/"`

```
#include <linux/unistd.h>
#define __NR_printmsg 450
int printmsg(int i){
    return syscall(__NR_printmsg, i);
}
int main(int argc, char** argv)
{
    printmsg(668);
    return 0;
}
```



```
~/Desktop/ass2/printmsg_test.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
printmsg.c printmsg_test.c trace_printmsg.sh Makefile syscall_64.tbl syscalls.h
1 #include <linux/unistd.h>
2 #define __NR_printmsg 450
3 int printmsg(int i){
4     return syscall( __NR_printmsg, i);
5 }
6
7 int main(int argc, char** argv)
8 {
9     printmsg(668);
10    return 0;
11 }
10 lines, 169 characters selected Tab Size: 4 C
```

- Because the message given by the print code in function `printmsg` is the only thing I want to trace, we need to use the `set_event` and `set_ftrace_filter` in ftrace
- By use command `cat available_events`, I lookup the available events can be used in tracing

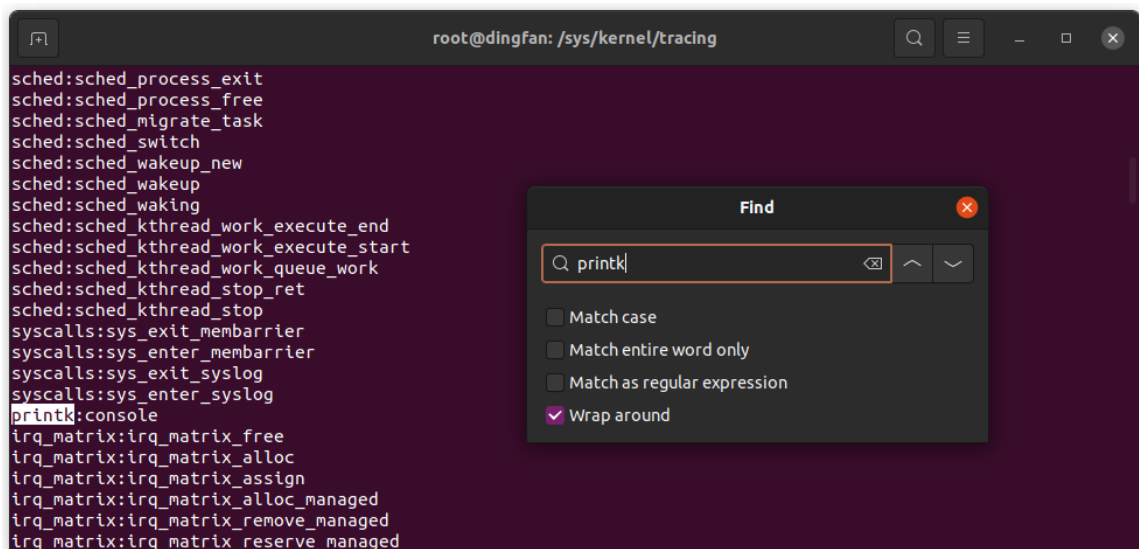


```

root@dingfan: /sys/kernel/tracing
dingfan@dingfan:/sys/kernel$ sudo su
root@dingfan:/sys/kernel# cd tracing
root@dingfan:/sys/kernel/tracing# cat available_events
vsock:virtio_transport_alloc_pkt
vsock:virtio_transport_recv_pkt
drm:drm_vblank_event
drm:drm_vblank_event_queued
drm:drm_vblank_event_delivered
initcall:initcall_finish
initcall:initcall_start
initcall:initcall_level
vsyscall:emulate_vsyscall
xen:xen_cpu_set_ldt
xen:xen_cpu_write_gdt_entry
xen:xen_cpu_load_idt
xen:xen_cpu_write_idt_entry
xen:xen_cpu_write_ldt_entry
xen:xen_mmu_write_cr3
xen:xen_mmu_flush_tlb_multi
xen:xen_mmu_flush_tlb_one_user
xen:xen_mmu_pgdn_unpin
xen:xen_mmu_pgdn_pin
xen:xen_mmu_release_ptpage

```

- I need to use the event `printk`

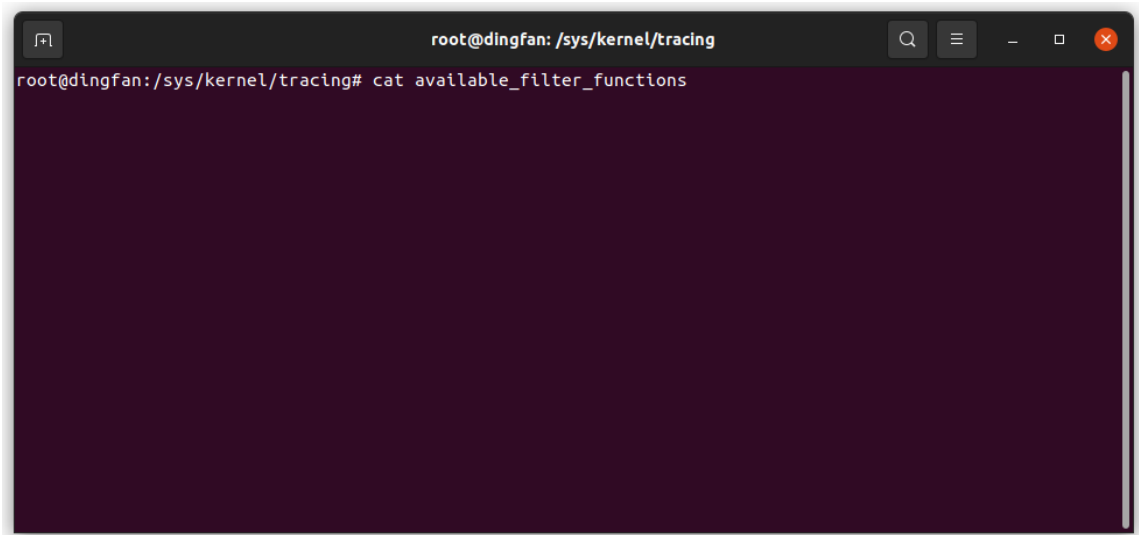


```

root@dingfan: /sys/kernel/tracing
sched:sched_process_exit
sched:sched_process_free
sched:sched_migrate_task
sched:sched_switch
sched:sched_wakeup_new
sched:sched_wakeup
sched:sched_waking
sched:sched_kthread_work_execute_end
sched:sched_kthread_work_execute_start
sched:sched_kthread_work_queue_work
sched:sched_kthread_stop_ret
sched:sched_kthread_stop
syscalls:sys_exit_membarrier
syscalls:sys_enter_membarrier
syscalls:sys_exit_syslog
syscalls:sys_enter_syslog
printk:console
irq_matrix:irq_matrix_free
irq_matrix:irq_matrix_alloc
irq_matrix:irq_matrix_assign
irq_matrix:irq_matrix_alloc_managed
irq_matrix:irq_matrix_remove_managed
irq_matrix:irq_matrix_reserve_managed

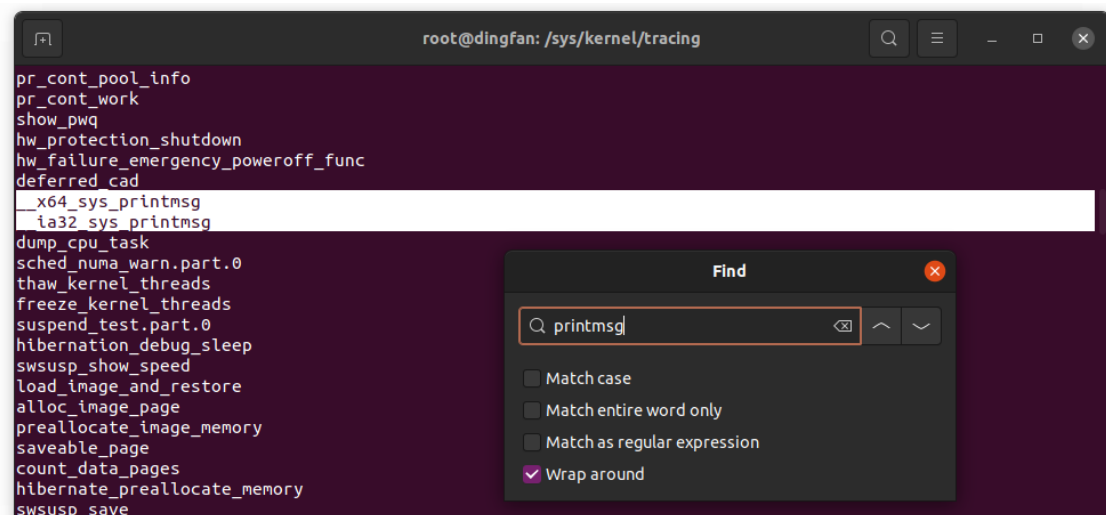
```

- By use command `cat available_filter_functions`, I lookup the available function filter in ftrace



```
root@dingfan: /sys/kernel/tracing
root@dingfan:/sys/kernel/tracing# cat available_filter_functions
```

- I need to use the function filter `__x64_sys_printmsg`



```
pr_cont_pool_info
pr_cont_work
show_pwq
hw_protection_shutdown
hw_failure_emergency_poweroff_func
deferred_cad
__x64_sys_printmsg
ia32_sys_printmsg
dump_cpu_task
sched_numa_warn.part.0
thaw_kernel_threads
freeze_kernel_threads
suspend_test.part.0
hibernation_debug_sleep
swsusp_show_speed
load_image_and_restore
alloc_image_page
preallocate_image_memory
saveable_page
count_data_pages
hibernate_preallocate_memory
swsusp_save
```

Find

printmsg

☐ Match case
☐ Match entire word only
☐ Match as regular expression
☒ Wrap around

- The shell script code (with comment) that used to run the the tracing is as followed:
 - the tracer I use here is `function`

```
# cd to the ftrace directory
cd /sys/kernel/tracing/;

# clear the old traces before start new tracing
echo 0 > tracing_on;
echo 0 > trace;

# choose the "function" as the current tracer
echo function > current_tracer;

# enabled the printk event in tracing
echo printk > set_event;
```

```

# set the new kernel function printmsg as the function filter in tracing
echo __x64_sys_printmsg > set_ftrace_filter;

# turn on the tracing
echo 1 > tracing_on;

# run printmsg_test.o to call the new kernel function printmsg 3 times
/home/dingfan/Desktop/ass2/printmsg_test.o;
/home/dingfan/Desktop/ass2/printmsg_test.o;
/home/dingfan/Desktop/ass2/printmsg_test.o;

# turn off the tracing
echo 0 > tracing_on;

# display the output of the ftrace
cat trace;

```

```

~/Desktop/ass2/trace_printmsg.sh - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
mytestprogram.c printmsg.c trace_printmsg.sh
1 # cd to the ftrace directory
2 cd /sys/kernel/tracing/;
3
4 # clear the old traces before start new tracing
5 echo 0 > tracing_on;
6 echo 0 > trace;
7
8 # choose the "function" as the current tracer
9 echo function > current_tracer;
10
11 # enabled the printk event in tracing
12 echo printk > set_event;
13
14 # set the new kernel function printmsg as the function filter in tracing
15 echo __x64_sys_printmsg > set_ftrace_filter;
16
17 # turn on the tracing
18 echo 1 > tracing_on;
19
20 # run printmsg_test.o to call the new kernel function printmsg 3 times
21 /home/dingfan/Desktop/ass2/printmsg_test.o;
22 /home/dingfan/Desktop/ass2/printmsg_test.o;
23 /home/dingfan/Desktop/ass2/printmsg_test.o;
24
25 # turn off the tracing
26 echo 0 > tracing_on;
27
28 # display the output of the ftrace
29 cat trace;

```

- use command `sudo sh trace_printmsg.sh` to run the shell script for tracing
 - As I can seen, the message given by the print code in function printmsg is the only thing I want to trace


```

dingfan@dingfan: ~/Desktop/ass2
dingfan@dingfan:~/Desktop/ass2$ sudo sh ./trace_printmsg.sh
[sudo] password for dingfan:
# tracer: function
#
# entries-in-buffer/entries-written: 6/6   #P:8
#
#          _-----= irqs-off
#          / _-----= need-resched
#          | / _-----= hardirq/softirq
#          || / _--= preempt-depth
#          ||| / _--= migrate-disable
#          |||| / _--= delay
#
# TASK-PID   CPU#  | TIMESTAMP | FUNCTION
# | |       | |   | |        | |
printmsg_test.o-2989 [001] ...1. 4344.959310: __x64_sys_printmsg <-do_syscall_64
printmsg_test.o-2989 [001] .... 4344.959314: __x64_sys_printmsg: Hello World! I am A0248373X
printmsg_test.o-2990 [006] ...1. 4344.960041: __x64_sys_printmsg <-do_syscall_64
printmsg_test.o-2990 [006] .... 4344.960044: __x64_sys_printmsg: Hello World! I am A0248373X
printmsg_test.o-2991 [006] ...1. 4344.960699: __x64_sys_printmsg <-do_syscall_64
printmsg_test.o-2991 [006] .... 4344.960702: __x64_sys_printmsg: Hello World! I am A0248373X
dingfan@dingfan:~/Desktop/ass2$

```

d. Use the function tracer to trace the kernel for about a few seconds. Give the screenshots of the first 20 lines of the trace file and analyse the basic structure of it. (2 marks)

solution:

- use the function tracer to trace the kernel(all functions) for about few seconds. All the commands and the screenshots of the first 20 lines of the trace file is as followed:
 - the command `cat trace | head 20` is used to show the first 20 lines of the trace file

```

root@dingfan: /sys/kernel/tracing
dingfan@dingfan:/sys/kernel$ sudo su
root@dingfan:/sys/kernel# cd tracing
root@dingfan:/sys/kernel/tracing# echo 0 > tracing_on
root@dingfan:/sys/kernel/tracing# echo 0 > trace
root@dingfan:/sys/kernel/tracing# echo function > current_tracer
root@dingfan:/sys/kernel/tracing# echo 1 > tracing_on
root@dingfan:/sys/kernel/tracing# echo 0 > tracing_on
root@dingfan:/sys/kernel/tracing# cat trace | head -20
# tracer: function
#
# entries-in-buffer/entries-written: 384605/1038410   #P:8
#
#          _-----= irqs-off
#          / _-----= need-resched
#          | / _-----= hardirq/softirq
#          || / _--= preempt-depth
#          ||| / _--= migrate-disable
#          |||| / _--= delay
#
# TASK-PID   CPU#  | TIMESTAMP | FUNCTION
# | |       | |   | |        | |
<idle>-0 [001] d..2. 15901.207229: irq_enter_rcu <-sysvec_call_function_single
<idle>-0 [001] d..2. 15901.207230: preempt_count_add <-irq_enter_rcu
<idle>-0 [001] d.h2. 15901.207230: tick_irq_enter <-irq_enter_rcu
<idle>-0 [001] d.h2. 15901.207230: tick_check_oneshot_broadcast_this_cpu <-tick_irq_enter
<idle>-0 [001] d.h2. 15901.207230: ktime_get <-tick_irq_enter
<idle>-0 [001] d.h2. 15901.207230: nr_iowait_cpu <-tick_irq_enter
<idle>-0 [001] d.h2. 15901.207231: tick_do_update_jiffies64 <-tick_irq_enter
<idle>-0 [001] d.h2. 15901.207231: __sysvec_call_function_single <-sysvec_call_function_single
root@dingfan:/sys/kernel/tracing#

```

- analyse the basic structure: of the first 20 lines of the trace file

1. `tracer: function` :indicate the current tracer is the `function`, it depends on what kind of tracer I use when tracing the kernel, it can also set as other tracer like `function_graph`
2. `entries-in-buffer/entries-written:3846005/1038410 #p:8` : show the total number of events in the buffer(`entries-in-buffer`) which is 384605 here; it also show the total number of entries that were written in buffer (`entries-written:3846005`) which is 1038410 here; what's more The difference (between `entries-in-buffer` and `entries-written`) is the number of entries that were lost due to the buffer filling up ($384605 - 1038410 = 110200$ events lost).
3. `TASK-PID` : is part of the header (The header explains the content of the events.), Task indicate the tracing name of the task process; PID indicate its Process Identifier(PID) of the process.
4. `CPU` : indicate the CPU which the process was running on.
5. the latency format:
 - a. `irqs-off` : it indicate whether the interruption is disabled when tracing. 'd' interrupts are disabled. '.' otherwise.
 - b. `need-resched` : it indicate the information of schedule.'N' both `TIF_NEED_RESCHED` and `PREEMPT_NEED_RESCHED` is set, 'n' only `TIF_NEED_RESCHED` is set, 'p' only `PREEMPT_NEED_RESCHED` is set, '.' otherwise.
 - c. `hardirq/softirq` : it indicate the information of interrupt mechanism: 'Z' - NMI occurred inside a hardirq 'z' - NMI is running 'H' - hard irq occurred inside a softirq. 'h' - hard irq is running 's' - soft irq is running '.' - normal context.
 - d. `preempt-depth` : it is the The level of `preempt_disabled`
6. `Timestamp` : is the time at which the function was entered. the timestamp is in <secs>
7. `Function` : indicate the function name which is traced; It also indicate which function call this function. For example: `irq_enter_cpu <- sysvec_call_function_single` means the function `irq_enter_cpu` is called by `sysvec_call_function_single`

Question2: 6 marks, about 30 minutes to 1 hour

Use the `function_graph` tracer in `ftrace` to trace the function call stack of three kernel functions, `vfs_read`, `lookup_dcache` and `task_fork_fair`

and set the max function call depth to record as 10. Give the complete commands you use and analyse the trace result. Note: the three functions should be traced at the same time.

solution:

- step0: `cd sys/kernel/tracing` cd to the ftrace directory
- step1:
 - `echo 0 > tracing_on` to turn off the old tracing
 - `echo 0 > trace` to clean the old traces
- step2: `echo function_graph > current_tracer` set function_graph as the current tracer
- step3: set set_graph_function to only trace vfs_read, lookup_dcache, task_fork_fair 3 functions and the functions they call
 - `echo vfs_read > set_graph_function` add the vfs_read to set_graph_function
 - `echo lookup_dcache >> set_graph_function` append the lookup_dcache to set_graph_function
 - `echo task_fork_fair >> set_graph_function` append the task_fork_fair to set_graph_function
 - can use the command `cat set_graph_function` to check the setting of set_graph_function
- step4: set the max function call depth to record as 10
 - `echo 10 > max_graph_depth`
- step5: turn on the tracing
 - `echo 1 > tracing_on`
- step6: turn off the tracing after tracing for several seconds
 - `echo 0 > tracing_on`

```

root@dingfan: /sys/kernel/tracing
root@dingfan:/sys/kernel/tracing# echo 0 > tracing_on
root@dingfan:/sys/kernel/tracing# echo 0 > trace
root@dingfan:/sys/kernel/tracing# echo function_graph > current_tracer
root@dingfan:/sys/kernel/tracing# echo vfs_read > set_graph_function
root@dingfan:/sys/kernel/tracing# echo lookup_dcache >> set_graph_function
root@dingfan:/sys/kernel/tracing# echo task_fork_fair >> set_graph_function
root@dingfan:/sys/kernel/tracing# cat set_graph_function
task_fork_fair
lookup_dcache
vfs_read
root@dingfan:/sys/kernel/tracing# echo 10 > max_graph_depth
root@dingfan:/sys/kernel/tracing# cat max_graph_depth
10
root@dingfan:/sys/kernel/tracing# echo 1 > tracing_on
root@dingfan:/sys/kernel/tracing# echo 0 > tracing_on
root@dingfan:/sys/kernel/tracing#

```

- step7: display the trace result
 - `cat trace` or `gedit trace`
- the trace result
 - As I can seen in the following screenshot: the three kernel functions `vfs_read`, `lookup_dcache` and `task_fork_fair` we want to trace in ftrace are in the trace file

```

1 # tracer: function_graph
2 #
3 # CPU DURATION FUNCTION CALLS
4 #
5 0) 0.323 us | } /* make kuid */
6 0) 0.111 us | make kuid() {
7 0) 0.336 us | map_id_range_down();
8 0) 0.336 us | }
9 0) 1.149 us | } /* atime needs update */
10 0) 1.360 us | } /* touch atime */
11 0) 3.490 us | } /* filemap_read */
12 0) 3.695 us | } /* generic_file_read_iter */
13 0) 3.936 us | } /* new_sync_read */
14 0) 0.111 us | fsnotify_parent();
15 0) 6.220 us | } /* vfs_read */
16 0)
17 0) rw_verify_area() {
18 0) security_file_permission() {
19 0) apparmor_file_permission() {
20 0) aa_file_perm();
21 0) 0.103 us | rcu_read_lock();
22 0) 0.108 us | aa_label_is_subset();
23 0) 0.105 us | rcu_read_unlock();
24 0) 0.759 us | }
25 0) 0.973 us | }
26 0) 0.166 us | fsnotify_parent();
27 0) 1.490 us | }
28 0) 1.711 us | }
29 0) new_sync_read() {
30 0) generic_file_read_iter() {
31 0) filemap_read() {
32 0) 0.101 us | cond_resched();
33 0) filemap_get_pages() {
34 0) filemap_get_read_batch() {
35 0) 0.104 us | rcu_read_lock();
36 0) 0.105 us | rcu_read_unlock();
37 0) 0.579 us | }
38 0) 0.804 us | cond_resched();
39 0) 0.100 us | }
40 0) touch_atime() {
41 0) atime_needs_update() {
42 0) make kuid() {
43 0) 0.108 us | map_id_range_down();
44 0) 0.340 us | }
45 0) make kuid() {
46 0) 0.985 us | map_id_range_down();
47 0) 0.108 us | }
48 0) 0.221 us | }
49 0) 1.200 us | }
50 0) 2.901 us | }
51 0) 3.334 us | }
52 0) 0.111 us | fsnotify_parent();
53 0) 5.618 us | }
54 0)

```

```

38587 3) 0.132 us  | eventfd_read() {
38588 3) 0.403 us  |   _raw_spin_lock_irq() {
38589 3) 0.132 us  |     preempt_count_add();
38590 3) 0.403 us  |   }
38591 3) 0.132 us  |   _raw_spin_unlock_irq() {
38592 3) 0.398 us  |     preempt_count_sub();
38593 3) 1.258 us  |   }
38594 3) 1.598 us  | }
38595 3) 3.852 us  | }
38596 2) VizComp-2562 => Chrome-2493
38597
38598
38599
38600
38601 2) lookup_dcache() {
38602 2)   d_lookup() {
38603 2)     d_lookup() {
38604 2)       _rcu_read_lock();
38605 2)       _raw_spin_lock();
38606 2)       preempt_count_add();
38607 2)       _raw_spin_unlock();
38608 2)       preempt_count_sub();
38609 2)       _rcu_read_unlock();
38610 2)     }
38611 2)     0.151 us
38612 2)     2.153 us
38613 2)     2.584 us
38614 2)     4.047 us
38615 }
38616 3) gnome-s-1681 => Composit-2864
38617
38618
38619 3)   vfs_read() {
38620 3)     rw_verify_area() {
38621 3)       security_file_permission() {
38622 3)         apparmor_file_permission() {
38623 3)           aa_file_perm() {
38624 3)             _rcu_read_lock();
38625 3)             _rcu_read_unlock();
38626 3)           }
38627 3)         }
38628 3)       fnotify_parent() {
38629 3)         dget_parent() {
38630 3)           _rcu_read_lock();
38631 3)           _rcu_read_unlock();
38632 3)         }
38633 3)       fnotify();
38634 3)       dput() {
38635 3)         cond_resched();
38636 3)         _rcu_read_lock();
38637 3)         _rcu_read_unlock();
38638 3)       }
38639 3)     }
38640 3)   }
38641 3)   7.459 us
38642 3)   7.853 us

```

```

123130 0) 2.869 us  | }
123131 0)   place_entity() {
123132 0)     sched_slice() {
123133 0)       _calc_delta();
123134 0)       _calc_delta();
123135 0)     }
123136 0)     0.929 us
123137 0)   }
123138 0)   raw_spin_rq_unlock() {
123139 0)     _raw_spin_unlock();
123140 0)     preempt_count_sub();
123141 0)   }
123142 0)   0.329 us
123143 0)   6.689 us
123144 0) }
123145 0)
123146 0)
123147 0)
123148 0)
123149 0)
123150 0)
123151 0)
123152 0)
123153 0)
123154 0)
123155 0)
123156 0)
123157 0)
123158 0)
123159 0)
123160 0)
123161 0)
123162 0)
123163 0)
123164 0)
123165 0)
123166 0)
123167 0)
123168 0)
123169 0)
123170 0)
123171 0)
123172 0)
123173 0)
123174 0)
123175 0)
123176 0)
123177 0)
123178 0)
123179 0)
123180 0)
123181 0)
123182 0)
123183 0)
123184 0)
123185 0)
123186 0)
123187 0)
123188 0)
123189 0)
123190 0)
123191 0)
123192 0)
123193 0)
123194 0)
123195 0)
123196 0)
123197 0)
123198 0)
123199 4) <...>-6985 => S-6988
123200
123201
123202 4)
123203 4)
123204 4)
123205 4)
123206 4)
123207 4)
123208 4)
123209 4)
123210 4)
123211 4)
123212 4)
123213 4)
123214 4)
123215 4)
123216 4)
123217 4)
123218 4)
123219 4)
123220 4)
123221 4)
123222 4)
123223 4)
123224 4)
123225 4)
123226 4)
123227 4)
123228 4)
123229 4)
123230 4)
123231 4)
123232 4)
123233 4)
123234 4)
123235 4)
123236 4)
123237 4)
123238 4)
123239 4)
123240 4)
123241 4)
123242 4)
123243 4)
123244 4)
123245 4)
123246 4)
123247 4)
123248 4)
123249 4)
123250 4)
123251 4)
123252 4)
123253 4)
123254 4)
123255 4)
123256 4)
123257 4)
123258 4)
123259 4)
123260 4)
123261 4)
123262 4)
123263 4)
123264 4)
123265 4)
123266 4)
123267 4)
123268 4)
123269 4)
123270 4)
123271 4)
123272 4)
123273 4)
123274 4)
123275 4)
123276 4)
123277 4)
123278 4)
123279 4)
123280 4)
123281 4)
123282 4)
123283 4)
123284 4)
123285 4)
123286 4)
123287 4)
123288 4)
123289 4)
123290 4)
123291 4)
123292 4)
123293 4)
123294 4)
123295 4)
123296 4)
123297 4)
123298 4)
123299 4)
123300 4)
123301 4)
123302 4)
123303 4)
123304 4)
123305 4)
123306 4)
123307 4)
123308 4)
123309 4)
123310 4)
123311 4)
123312 4)
123313 4)
123314 4)
123315 4)
123316 4)
123317 4)
123318 4)
123319 4)
123320 4)
123321 4)
123322 4)
123323 4)
123324 4)
123325 4)
123326 4)
123327 4)
123328 4)
123329 4)
123330 4)
123331 4)
123332 4)
123333 4)
123334 4)
123335 4)
123336 4)
123337 4)
123338 4)
123339 4)
123340 4)
123341 4)
123342 4)
123343 4)
123344 4)
123345 4)
123346 4)
123347 4)
123348 4)
123349 4)
123350 4)
123351 4)
123352 4)
123353 4)
123354 4)
123355 4)
123356 4)
123357 4)
123358 4)
123359 4)
123360 4)
123361 4)
123362 4)
123363 4)
123364 4)
123365 4)
123366 4)
123367 4)
123368 4)
123369 4)
123370 4)
123371 4)
123372 4)
123373 4)
123374 4)
123375 4)
123376 4)
123377 4)
123378 4)
123379 4)
123380 4)
123381 4)
123382 4)
123383 4)
123384 4)
123385 4)
123386 4)
123387 4)
123388 4)
123389 4)
123390 4)
123391 4)
123392 4)
123393 4)
123394 4)
123395 4)
123396 4)
123397 4)
123398 4)
123399 4)
123400 4)
123401 4)
123402 4)
123403 4)
123404 4)
123405 4)
123406 4)
123407 4)
123408 4)
123409 4)
123410 4)
123411 4)
123412 4)
123413 4)
123414 4)
123415 4)
123416 4)
123417 4)
123418 4)
123419 4)
123420 4)
123421 4)
123422 4)
123423 4)
123424 4)
123425 4)
123426 4)
123427 4)
123428 4)
123429 4)
123430 4)
123431 4)
123432 4)
123433 4)
123434 4)
123435 4)
123436 4)
123437 4)
123438 4)
123439 4)
123440 4)
123441 4)
123442 4)
123443 4)
123444 4)
123445 4)
123446 4)
123447 4)
123448 4)
123449 4)
123450 4)
123451 4)
123452 4)
123453 4)
123454 4)
123455 4)
123456 4)
123457 4)
123458 4)
123459 4)
123460 4)
123461 4)
123462 4)
123463 4)
123464 4)
123465 4)
123466 4)
123467 4)
123468 4)
123469 4)
123470 4)
123471 4)
123472 4)
123473 4)
123474 4)
123475 4)
123476 4)
123477 4)
123478 4)
123479 4)
123480 4)
123481 4)
123482 4)
123483 4)
123484 4)
123485 4)
123486 4)
123487 4)
123488 4)
123489 4)
123490 4)
123491 4)
123492 4)
123493 4)
123494 4)
123495 4)
123496 4)
123497 4)
123498 4)
123499 4)
123500 4)
123501 4)
123502 4)
123503 4)
123504 4)
123505 4)
123506 4)
123507 4)
123508 4)
123509 4)
123510 4)
123511 4)
123512 4)
123513 4)
123514 4)
123515 4)
123516 4)
123517 4)
123518 4)
123519 4)
123520 4)
123521 4)
123522 4)
123523 4)
123524 4)
123525 4)
123526 4)
123527 4)
123528 4)
123529 4)
123530 4)
123531 4)
123532 4)
123533 4)
123534 4)
123535 4)
123536 4)
123537 4)
123538 4)
123539 4)
123540 4)
123541 4)
123542 4)
123543 4)
123544 4)
123545 4)
123546 4)
123547 4)
123548 4)
123549 4)
123550 4)
123551 4)
123552 4)
123553 4)
123554 4)
123555 4)
123556 4)
123557 4)
123558 4)
123559 4)
123560 4)
123561 4)
123562 4)
123563 4)
123564 4)
123565 4)
123566 4)
123567 4)
123568 4)
123569 4)
123570 4)
123571 4)
123572 4)
123573 4)
123574 4)
123575 4)
123576 4)
123577 4)
123578 4)
123579 4)
123580 4)
123581 4)
123582 4)
123583 4)
123584 4)
123585 4)
123586 4)
123587 4)
123588 4)
123589 4)
123590 4)
123591 4)
123592 4)
123593 4)
123594 4)
123595 4)
123596 4)
123597 4)
123598 4)
123599 4)
123600 4)
123601 4)
123602 4)
123603 4)
123604 4)
123605 4)
123606 4)
123607 4)
123608 4)
123609 4)
123610 4)
123611 4)
123612 4)
123613 4)
123614 4)
123615 4)
123616 4)
123617 4)
123618 4)
123619 4)
123620 4)
123621 4)
123622 4)
123623 4)
123624 4)
123625 4)
123626 4)
123627 4)
123628 4)
123629 4)
123630 4)
123631 4)
123632 4)
123633 4)
123634 4)
123635 4)
123636 4)
123637 4)
123638 4)
123639 4)
123640 4)
123641 4)
123642 4)
123643 4)
123644 4)
123645 4)
123646 4)
123647 4)
123648 4)
123649 4)
123650 4)
123651 4)
123652 4)
123653 4)
123654 4)
123655 4)
123656 4)
123657 4)
123658 4)
123659 4)
123660 4)
123661 4)
123662 4)
123663 4)
123664 4)
123665 4)
123666 4)
123667 4)
123668 4)
123669 4)
123670 4)
123671 4)
123672 4)
123673 4)
123674 4)
123675 4)
123676 4)
123677 4)
123678 4)
123679 4)
123680 4)
123681 4)
123682 4)
123683 4)
123684 4)
123685 4)
123686 4)
123687 4)
123688 4)
123689 4)
123690 4)
123691 4)
123692 4)
123693 4)
123694 4)
123695 4)
123696 4)
123697 4)
123698 4)
123699 4)
123700 4)
123701 4)
123702 4)
123703 4)
123704 4)
123705 4)
123706 4)
123707 4)
123708 4)
123709 4)
123710 4)
123711 4)
123712 4)
123713 4)
123714 4)
123715 4)
123716 4)
123717 4)
123718 4)
123719 4)
123720 4)
123721 4)
123722 4)
123723 4)
123724 4)
123725 4)
123726 4)
123727 4)
123728 4)
123729 4)
123730 4)
123731 4)
123732 4)
123733 4)
123734 4)
123735 4)
123736 4)
123737 4)
123738 4)
123739 4)
123740 4)
123741 4)
123742 4)
123743 4)
123744 4)
123745 4)
123746 4)
123747 4)
123748 4)
123749 4)
123750 4)
123751 4)
123752 4)
123753 4)
123754 4)
123755 4)
123756 4)
123757 4)
123758 4)
123759 4)
123760 4)
123761 4)
123762 4)
123763 4)
123764 4)
123765 4)
123766 4)
123767 4)
123768 4)
123769 4)
123770 4)
123771 4)
123772 4)
123773 4)
123774 4)
123775 4)
123776 4)
123777 4)
123778 4)
123779 4)
123780 4)
123781 4)
123782 4)
123783 4)
123784 4)
123785 4)
123786 4)
123787 4)
123788 4)
123789 4)
123790 4)
123791 4)
123792 4)
123793 4)
123794 4)
123795 4)
123796 4)
123797 4)
123798 4)
123799 4)
123800 4)
123801 4)
123802 4)
123803 4)
123804 4)
123805 4)
123806 4)
123807 4)
123808 4)
123809 4)
123810 4)
123811 4)
123812 4)
123813 4)
123814 4)
123815 4)
123816 4)
123817 4)
123818 4)
123819 4)
123820 4)
123821 4)
123822 4)
123823 4)
123824 4)
123825 4)
123826 4)
123827 4)
123828 4)
123829 4)
123830 4)
123831 4)
123832 4)
123833 4)
123834 4)
123835 4)
123836 4)
123837 4)
123838 4)
123839 4)
123840 4)
123841 4)
123842 4)
123843 4)
123844 4)
123845 4)
123846 4)
123847 4)
123848 4)
123849 4)
123850 4)
123851 4)
123852 4)
123853 4)
123854 4)
123855 4)
123856 4)
123857 4)
123858 4)
123859 4)
123860 4)
123861 4)
123862 4)
123863 4)
123864 4)
123865 4)
123866 4)
123867 4)
123868 4)
123869 4)
123870 4)
123871 4)
123872 4)
123873 4)
123874 4)
123875 4)
123876 4)
123877 4)
123878 4)
123879 4)
123880 4)
123881 4)
123882 4)
123883 4)
123884 4)
123885 4)
123886 4)
123887 4)
123888 4)
123889 4)
123890 4)
123891 4)
123892 4)
123893 4)
123894 4)
123895 4)
123896 4)
123897 4)
123898 4)
123899 4)
123900 4)
123901 4)
123902 4)
123903 4)
123904 4)
123905 4)
123906 4)
123907 4)
123908 4)
123909 4)
123910 4)
123911 4)
123912 4)
123913 4)
123914 4)
123915 4)
123916 4)
123917 4)
123918 4)
123919 4)
123920 4)
123921 4)
123922 4)
123923 4)
123924 4)
123925 4)
123926 4)
123927 4)
123928 4)
123929 4)
123930 4)
123931 4)
123932 4)
123933 4)
123934 4)
123935 4)
123936 4)
123937 4)
123938 4)
123939 4)
123940 4)
123941 4)
123942 4)
123943 4)
123944 4)
123945 4)
123946 4)
123947 4)
123948 4)
123949 4)
123950 4)
123951 4)
123952 4)
123953 4)
123954 4)
123955 4)
123956 4)
123957 4)
123958 4)
123959 4)
123960 4)
123961 4)
123962 4)
123963 4)
123964 4)
123965 4)
123966 4)
123967 4)
123968 4)
123969 4)
123970 4)
123971 4)
123972 4)
123973 4)
123974 4)
123975 4)
123976 4)
123977 4)
123978 4)
123979 4)
123980 4)
123981 4)
123982 4)
123983 4)
123984 4)
123985 4)
123986 4)
123987 4)
123988 4)
123989 4)
123990 4)
123991 4)
123992 4)
123993 4)
123994 4)
123995 4)
123996 4)
123997 4)
123998 4)
123999 4)
124000 4)

```

- analyse the trace result:
 - the first line: **tracer: function** :indicate the current tracer is the **function_graph**
 - CPU** : indicate the CPU which the process was running on.
 - Duration** : indicate the function's time of execution
 - Function calls** : indicate both entry and exit of the functions that we want to trace: this column also includes the comments print by `trace_printk`

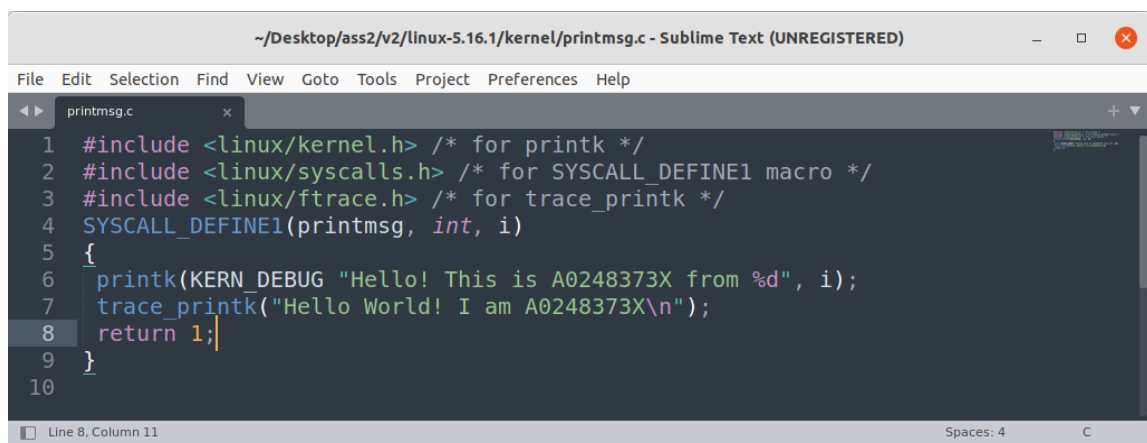
Question 3: 8 marks, about 2 to 3 hours

a. Finish the steps for adding a kernel function into kernel in the tutorial. Supply the two C files. (2 marks)

solution:

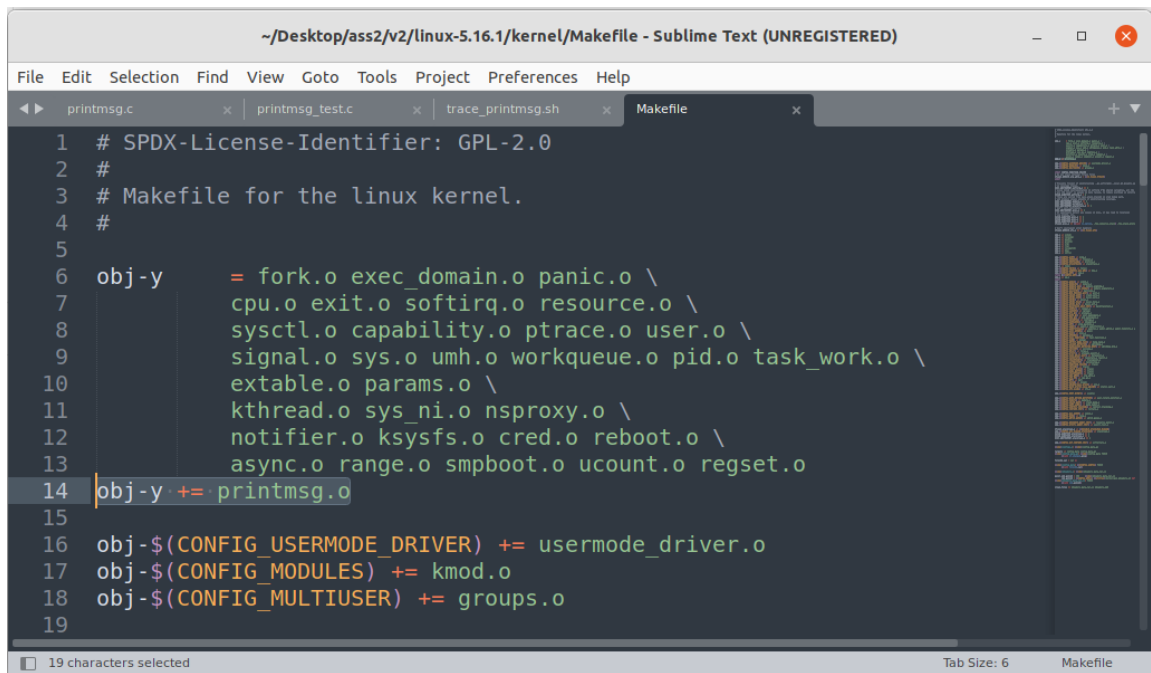
- step1: Add a file in the kernel/ directory under the linux kernel source file
 - the name of the new kernel function is `printmsg`
 - the message include my student ID `A0248373X`

```
#include <linux/kernel.h> /* for printk */
#include <linux/syscalls.h> /* for SYSCALL_DEFINE1 macro */
#include <linux/ftrace.h> /* for trace_printk */
SYSCALL_DEFINE1(printmsg, int, i)
{
    printk(KERN_DEBUG "Hello! This is A0248373X from %d", i);
    trace_printk("Hello World! I am A0248373X\n");
    return 1;
}
```



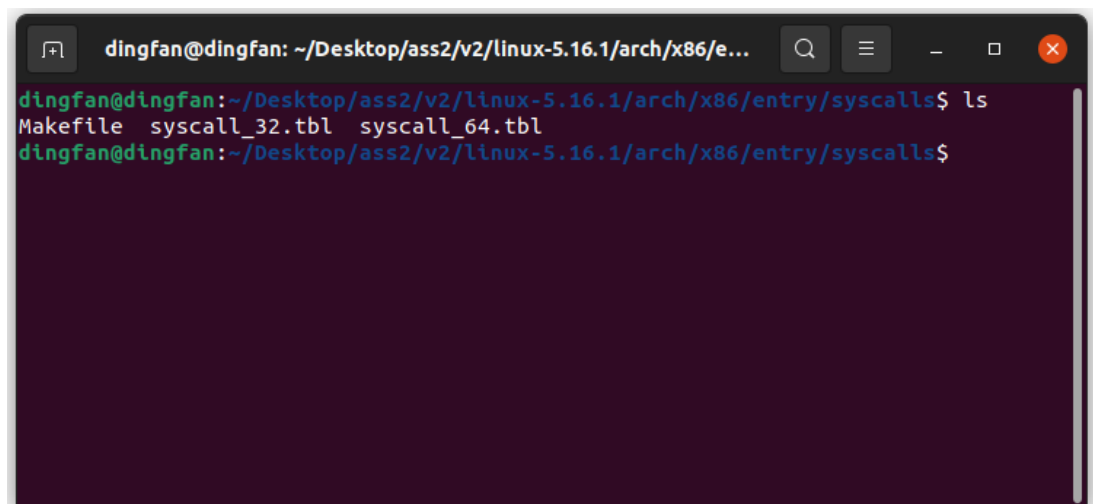
```
~/Desktop/ass2/v2/linux-5.16.1/kernel/printmsg.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
printmsg.c
1 #include <linux/kernel.h> /* for printk */
2 #include <linux/syscalls.h> /* for SYSCALL_DEFINE1 macro */
3 #include <linux/ftrace.h> /* for trace_printk */
4 SYSCALL_DEFINE1(printmsg, int, i)
5 {
6     printk(KERN_DEBUG "Hello! This is A0248373X from %d", i);
7     trace_printk("Hello World! I am A0248373X\n");
8     return 1;
9 }
10
Line 8, Column 11 Spaces: 4 C
```

- step2: Add a line `obj-y += printmsg.o` in the Makefile under the kernel/ directory so that the kernel will add my new program into kernel image.



```
1 # SPDX-License-Identifier: GPL-2.0
2 #
3 # Makefile for the linux kernel.
4 #
5
6 obj-y      = fork.o exec_domain.o panic.o \
7             cpu.o exit.o softirq.o resource.o \
8             sysctl.o capability.o ptrace.o user.o \
9             signal.o sys.o umh.o workqueue.o pid.o task_work.o \
10            extable.o params.o \
11            kthread.o sys_ni.o nsproxy.o \
12            notifier.o ksysfs.o cred.o reboot.o \
13            async.o range.o smpboot.o ucount.o regset.o
14 obj-y += printmsg.o
15
16 obj-$(CONFIG_USERMODE_DRIVER) += usermode_driver.o
17 obj-$(CONFIG_MODULES) += kmod.o
18 obj-$(CONFIG_MULTIVER) += groups.o
19
```

- step3:find the location of the table of function entries and add the new function at the end of the table as a new entry
 - the location of the table of function entries is `linux-5.16.1/arch/x86/entry/syscalls/syscall_64.tbl` in the linux kernel source code



```
dingfan@dingfan: ~/Desktop/ass2/v2/linux-5.16.1/arch/x86/e...
dingfan@dingfan:~/Desktop/ass2/v2/linux-5.16.1/arch/x86/entry/syscalls$ ls
Makefile  syscall_32.tbl  syscall_64.tbl
dingfan@dingfan:~/Desktop/ass2/v2/linux-5.16.1/arch/x86/entry/syscalls$
```

- add a new line in as the screenshot shows: `450 common printmsg sys_printmsg`

```

362 438 common piada_getra sys_piada_getra
363 439 common faccessat2 sys_faccessat2
364 440 common process_madvise sys_process_madvise
365 441 common epoll_pwait2 sys_epoll_pwait2
366 442 common mount_setattr sys_mount_setattr
367 443 common quotactl_fd sys_quotactl_fd
368 444 common landlock_create_ruleset sys_landlock_create_ruleset
369 445 common landlock_add_rule sys_landlock_add_rule
370 446 common landlock_restrict_self sys_landlock_restrict_self
371 447 common memfd_secret sys_memfd_secret
372 448 common process_mrelease sys_process_mrelease
373 449 common futex_waitv sys_futex_waitv
374 450 common printmsg sys_printmsg
375
376 #
377 # Due to a historical design error, certain syscalls are numbered
378 # differently
379 # in x32 as compared to native x86_64. These syscalls have numbers
380 # 512-547.
381 # Do not add new syscalls to this range. Numbers 548 and above are

```

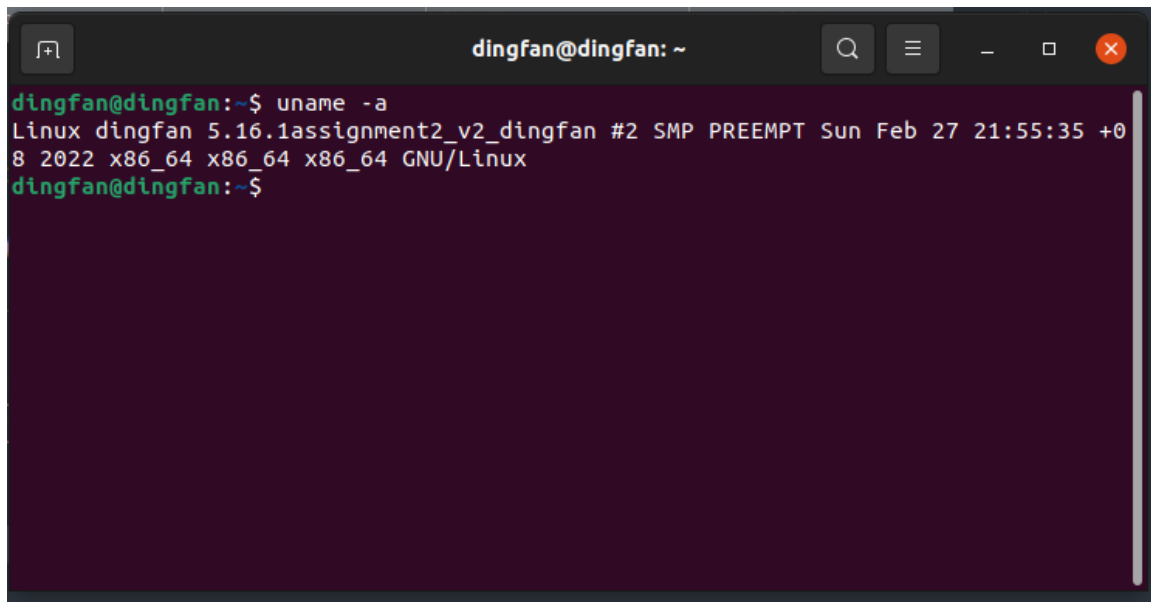
- step4: add `asmlinkage long sys_printmsg(int i);` in file `linux-5.16.1/include/linux/syscall.h`

```

1260 asmlinkage long sys_old_getrlimit(unsigned int resource, struct rlimit __user *rlim);
1261 #endif
1262
1263 /* obsolete: ipc */
1264 asmlinkage long sys_ipc(unsigned int call, int first, unsigned long second,
1265                          unsigned long third, void __user *ptr, long fifth);
1266
1267 /* obsolete: mm/ */
1268 asmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned long len,
1269                                unsigned long prot, unsigned long flags,
1270                                unsigned long fd, unsigned long pgoff);
1271 asmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);
1272
1273 asmlinkage long sys_printmsg(int i);
1274
1275 /*
1276  * Not a real system call, but a placeholder for syscalls which are
1277  * not implemented -- see kernel/sys ni.c
1278  */

```

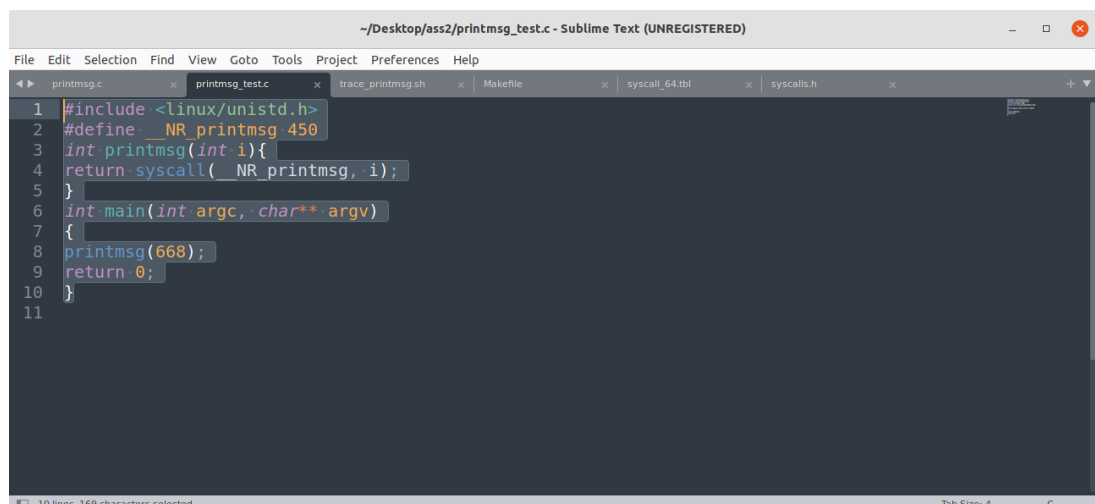
- step5: Recompile the kernel and boot in the new kernel as Assignment 1.



```
dingfan@dingfan: ~  
dingfan@dingfan:~$ uname -a  
Linux dingfan 5.16.1assignment2_v2_dingfan #2 SMP PREEMPT Sun Feb 27 21:55:35 +08 2022 x86_64 x86_64 x86_64 GNU/Linux  
dingfan@dingfan:~$
```

- step6: Build a user mode program to call the new kernel function `printmsg`. Compile and run the program.
 - the user mode program is `printmsg_test`, the index is `450`

```
#include <linux/unistd.h>  
#define __NR_printmsg 450  
int printmsg(int i){  
    return syscall(__NR_printmsg, i);  
}  
int main(int argc, char** argv)  
{  
    printmsg(668);  
    return 0;  
}
```



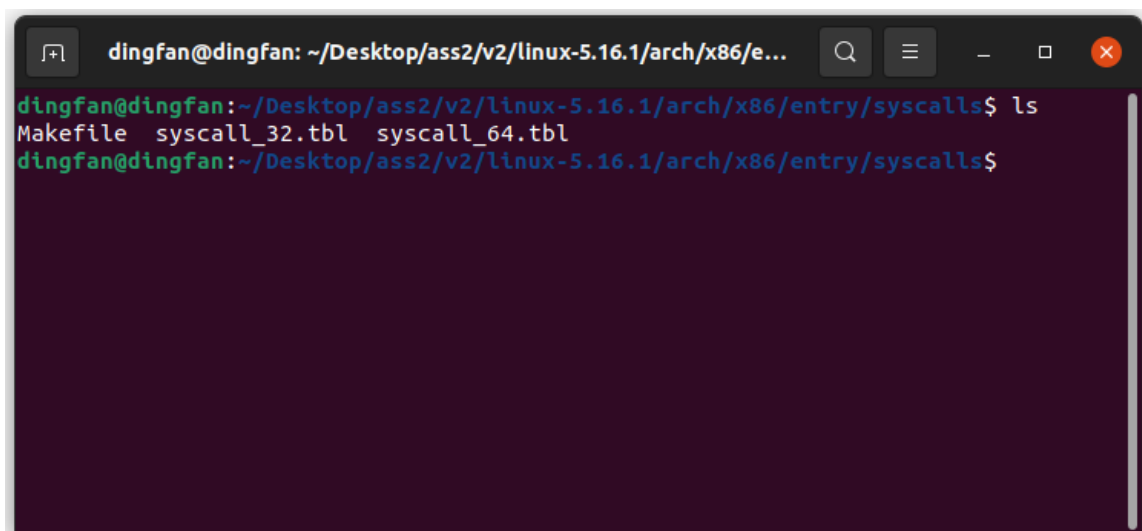
```
~/Desktop/ass2/printmsg_test.c - Sublime Text (UNREGISTERED)  
File Edit Selection Find View Goto Tools Project Preferences Help  
printmsg.c x printmsg_test.c x trace_printmsg.sh x Makefile x syscall_64.tbl x syscalls.h x  
1 #include <linux/unistd.h>  
2 #define __NR_printmsg 450  
3 int printmsg(int i){  
4     return syscall(__NR_printmsg, i);  
5 }  
6 int main(int argc, char** argv)  
7 {  
8     printmsg(668);  
9     return 0;  
10 }  
11  
10 lines, 169 characters selected  
Tab Size: 4 C
```

- use command `gcc printmsg_test.c -o printmsg_test.o` compile the test program into the directory `"~/Desktop/ass2/"`

b. What is the full path of the system call table? Show the line you added in the system call table. (2 marks)

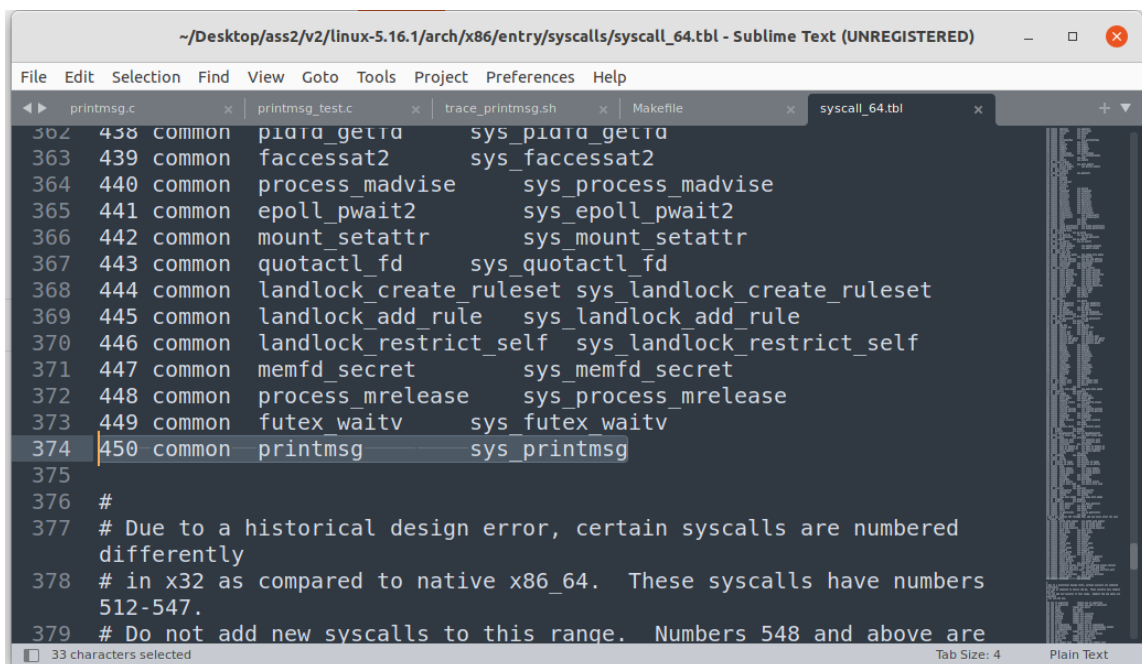
solution:

- the full path of the system call table is `linux-5.16.1/arch/x86/entry/syscalls/syscall_64.tbl`



```
dingfan@dingfan: ~/Desktop/ass2/v2/linux-5.16.1/arch/x86/e...
dingfan@dingfan:~/Desktop/ass2/v2/linux-5.16.1/arch/x86/entry/syscalls$ ls
Makefile  syscall_32.tbl  syscall_64.tbl
dingfan@dingfan:~/Desktop/ass2/v2/linux-5.16.1/arch/x86/entry/syscalls$
```

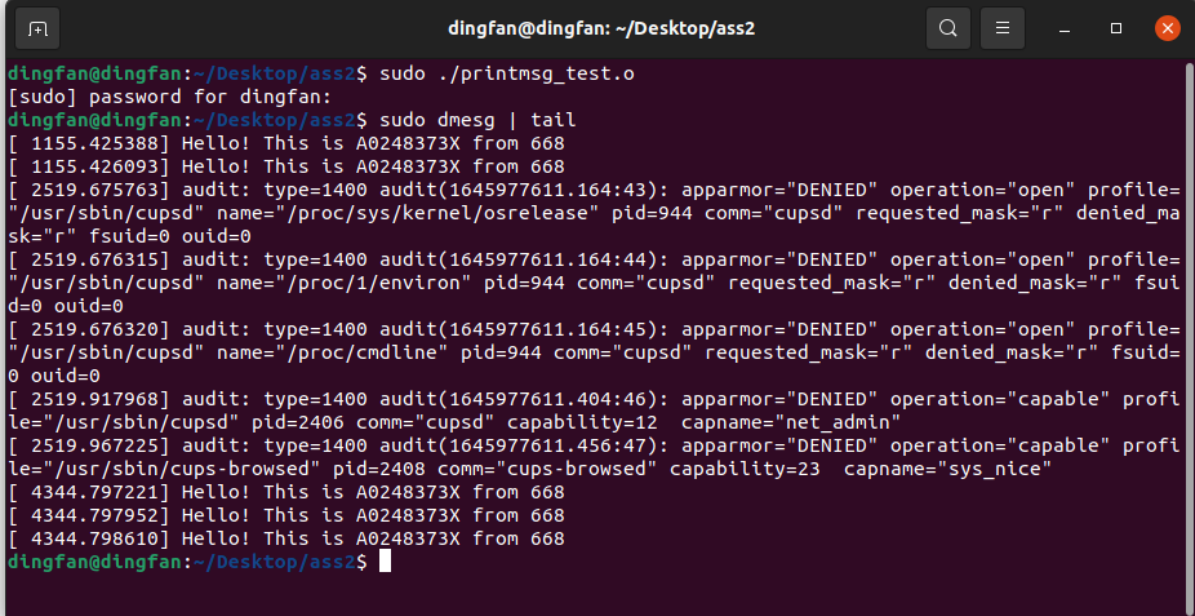
- the new line added in the system call table: `450 common printmsg sys_printmsg`



```
~/Desktop/ass2/v2/linux-5.16.1/arch/x86/entry/syscalls/syscall_64.tbl - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
printmsg.c x printmsg_test.c x trace_printmsg.sh x Makefile x syscall_64.tbl x
362 438 common piada_gettd sys_piada_gettd
363 439 common faccessat2 sys_faccessat2
364 440 common process_madvise sys_process_madvise
365 441 common epoll_pwait2 sys_epoll_pwait2
366 442 common mount_setattr sys_mount_setattr
367 443 common quotactl_fd sys_quotactl_fd
368 444 common landlock_create_ruleset sys_landlock_create_ruleset
369 445 common landlock_add_rule sys_landlock_add_rule
370 446 common landlock_restrict_self sys_landlock_restrict_self
371 447 common memfd_secret sys_memfd_secret
372 448 common process_mrelease sys_process_mrelease
373 449 common futex_waitv sys_futex_waitv
374 450 common printmsg sys_printmsg
375
376 #
377 # Due to a historical design error, certain syscalls are numbered
378 # differently
379 # in x32 as compared to native x86_64. These syscalls have numbers
380 # 512-547.
381 # Do not add new syscalls to this range. Numbers 548 and above are
```

c. Show the screenshot of the “dmesg | tail” containing the print message of your new kernel function. (2 marks)

solution:

A terminal window titled 'dingfan@dingfan: ~/Desktop/ass2' with a dark purple background. The user has executed 'sudo ./printmsg_test.o' and 'sudo dmesg | tail'. The output shows several lines of kernel messages, including 'Hello! This is A0248373X from 668' and several audit messages from the 'cupsd' process. The terminal window has standard Linux window controls at the top right.

```
dingfan@dingfan:~/Desktop/ass2$ sudo ./printmsg_test.o
[sudo] password for dingfan:
dingfan@dingfan:~/Desktop/ass2$ sudo dmesg | tail
[ 1155.425388] Hello! This is A0248373X from 668
[ 1155.426093] Hello! This is A0248373X from 668
[ 2519.675763] audit: type=1400 audit(1645977611.164:43): apparmor="DENIED" operation="open" profile=
"/usr/sbin/cupsd" name="/proc/sys/kernel/osrelease" pid=944 comm="cupsd" requested_mask="r" denied_ma
sk="r" fsuid=0 ouid=0
[ 2519.676315] audit: type=1400 audit(1645977611.164:44): apparmor="DENIED" operation="open" profile=
"/usr/sbin/cupsd" name="/proc/1/environ" pid=944 comm="cupsd" requested_mask="r" denied_mask="r" fsui
d=0 ouid=0
[ 2519.676320] audit: type=1400 audit(1645977611.164:45): apparmor="DENIED" operation="open" profile=
"/usr/sbin/cupsd" name="/proc/cmdline" pid=944 comm="cupsd" requested_mask="r" denied_mask="r" fsuid=
0 ouid=0
[ 2519.917968] audit: type=1400 audit(1645977611.404:46): apparmor="DENIED" operation="capable" profi
le="/usr/sbin/cupsd" pid=2406 comm="cupsd" capability=12 capname="net_admin"
[ 2519.967225] audit: type=1400 audit(1645977611.456:47): apparmor="DENIED" operation="capable" profi
le="/usr/sbin/cups-browsed" pid=2408 comm="cups-browsed" capability=23 capname="sys_nice"
[ 4344.797221] Hello! This is A0248373X from 668
[ 4344.797952] Hello! This is A0248373X from 668
[ 4344.798610] Hello! This is A0248373X from 668
dingfan@dingfan:~/Desktop/ass2$
```

d. Give the first 20 lines of the trace file of ftrace and analyse it. (2 marks)

solution:

- This question means use ftrace to trace the function that I inserted and called above:
- Because I have implemented a shell script in [Question 1c](#) (which includes all the related commands). That shell script is used for tracing the function(`printmsg`) which I inserted into the new kernel
- Here we just run the shell script to see the first 20 lines of the trace files and analyse it:

```

~/Desktop/ass2/trace_printmsg.sh - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
mytestprogram.c printmsg.c trace_printmsg.sh
1 # cd to the ftrace directory
2 cd /sys/kernel/tracing/;
3
4 # clear the old traces before start new tracing
5 echo 0 > tracing_on;
6 echo 0 > trace;
7
8 # choose the "function" as the current tracer
9 echo function > current_tracer;
10
11 # enabled the printk event in tracing
12 echo printk > set_event;
13
14 # set the new kernel function printmsg as the function filter in tracing
15 echo __x64_sys_printmsg > set_ftrace_filter;
16
17 # turn on the tracing
18 echo 1 > tracing_on;
19
20 # run printmsg test.o to call the new kernel function printmsg 3 times
21 /home/dingfan/Desktop/ass2/printmsg_test.o;
22 /home/dingfan/Desktop/ass2/printmsg_test.o;
23 /home/dingfan/Desktop/ass2/printmsg_test.o;
24
25 # turn off the tracing
26 echo 0 > tracing_on;
27
28 # display the output of the ftrace
29 cat trace;

```

```

dingfan@dingfan: ~/Desktop/ass2
dingfan@dingfan:~/Desktop/ass2$ sudo sh ./trace_printmsg.sh
[sudo] password for dingfan:
# tracer: function
#
# entries-in-buffer/entries-written: 6/6 #P:8
#
# -----> irqsoff
# -----> need-resched
# / -----> hardirq/softirq
# || / -----> preempt-depth
# ||| / -----> migrate-disable
# |||| / -----> delay
#
TASK-PID CPU# ||||| TIMESTAMP FUNCTION
#
printmsg_test.o-2989 [001] ...1. 4344.959310: __x64_sys_printmsg <-do_syscall_64
printmsg_test.o-2989 [001] .... 4344.959314: __x64_sys_printmsg: Hello World! I am A0248373X
printmsg_test.o-2990 [006] ...1. 4344.960041: __x64_sys_printmsg <-do_syscall_64
printmsg_test.o-2990 [006] .... 4344.960044: __x64_sys_printmsg: Hello World! I am A0248373X
printmsg_test.o-2991 [006] ...1. 4344.960699: __x64_sys_printmsg <-do_syscall_64
printmsg_test.o-2991 [006] .... 4344.960702: __x64_sys_printmsg: Hello World! I am A0248373X
dingfan@dingfan:~/Desktop/ass2$

```

- analyse the result:
 - As I can see, the trace only has the information related to the new kernel function `printmsg` that I added to the new kernel
 - `tracer: function` : indicate the tracer I used when trace the new kernel function is `function`
 - `entries-in-buffer/entries-written:6/6` : show the total number of events in the buffer(`entries-in-buffer`) which is 6 here; it also show the total number of entries

that were written in buffer (`entries-written:3846005`) which is 6 here;

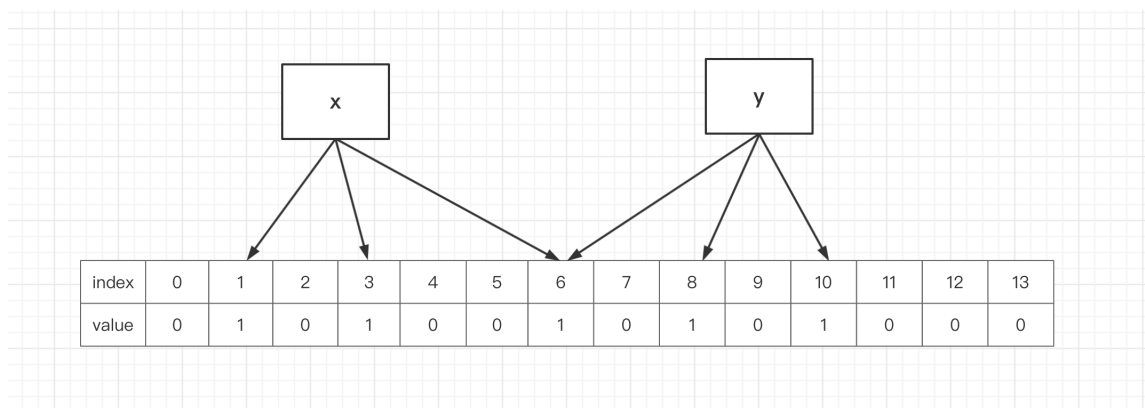
- **TASK-PID** : As the screenshot show, all the 6 trace entries belongs to the same TASK `printmsg_test.o` (which is the user mode program I write to test the new kernel function I mentioned above). The PID of that processes is 2989 2990 and 2991, because I call the `printmsg_test.o` program 3 times to test the new kernel function in my shell script as I mentioned above.
- **Function** : `__x64_sys_printmsg <- do_syscall_64` means the function `__x64_sys_printmsg` is called by `do_syscall_64`; `__x64_sys_printmsg: Hello World! I am A0248373X` this entry is print by the `trace_printk()` in my new kernel function as I mentioned above.

Additional exercises (5 marks each)

1. Using an example, show why deletion is not supported in the bit-vector form of the Bloom filter. Can you suggest a modification that would allow it?

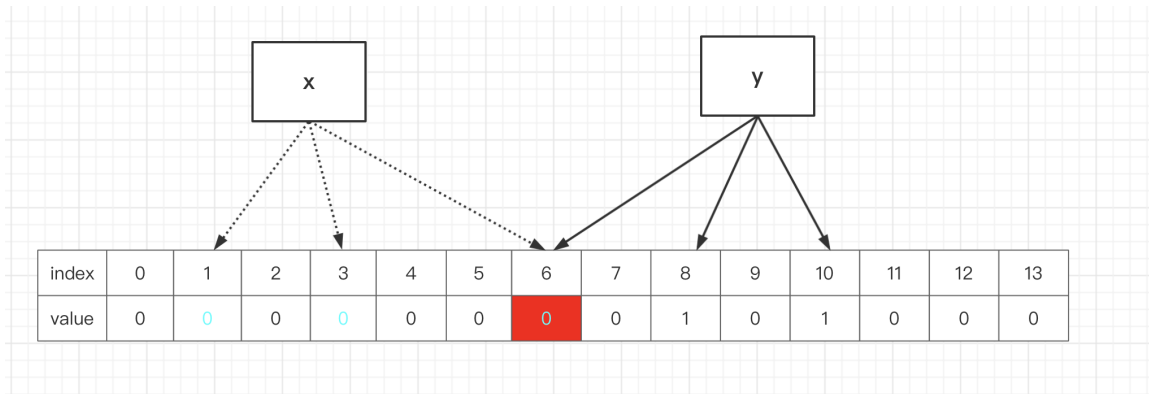
solution:

- using the following example to explain why deletion is not supported in bit vector form of the Bloom filter.

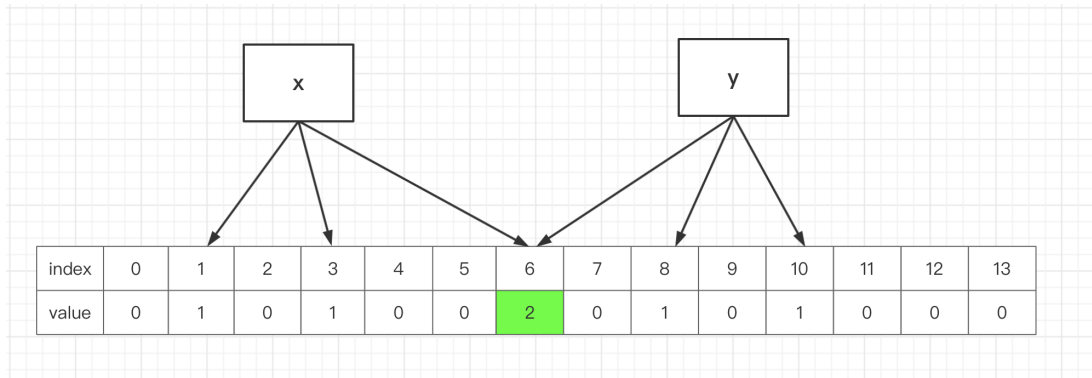


- Assume the element `x` is already in the set; and the hash value of `x` set the index: 1 ,3, 6 of the bit-vector to `1`
- Assume the element `y` is also in the set; and the hash value of `y` set the index:6, 8, 10 of the bit-vector to `1`

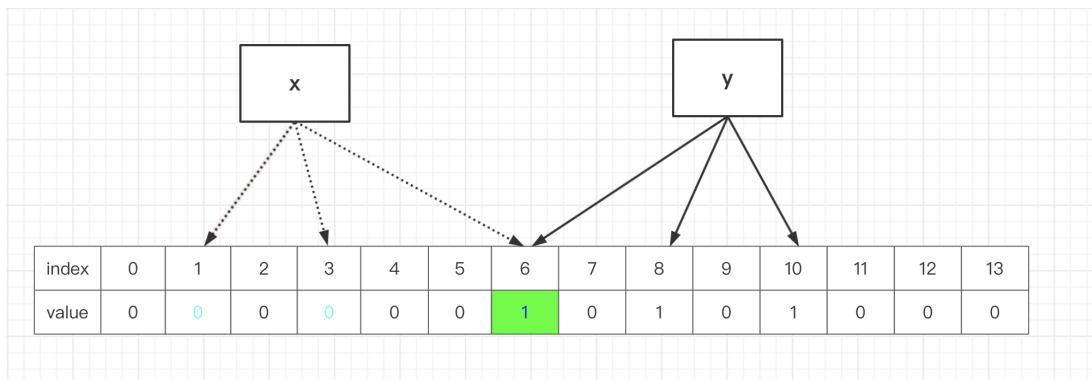
- Assume this set currently only have the two element x and y, all other index remain value 0, only the value of index 1,3,6,8,10 is 1
- In this condition if we delete element x, because the 3 hash index related to x is 1, 3, 6, we need set these three index to 0. Than we get the following graph:



- After deleting the element x, it may arise a problem when we loop up using the bloom filter the element y whether in the set.
- The reason is one of the hash value of y has a collision with one of the hash value of x: the index 6, the value of index 6 was set to 0 due to the deletion of element x
- When we using the bloom filter to check whether the element is in the set, the bloom filter will return: “element is not in the set” because the value of index 6 is 0
- According to the example we can see that deletion is not supported in bit vector form of the Bloom filter.
- Suggest a possible modification that would allow it:
 - Using the above example and make some modifications
 - Assume firstly add the value x to the set; set the value of index of 1,3,6 to value = value + 1, the value of the index 1,3,6 is set to 1 from 0
 - Then add the value y to the set; set the value of index 6,8,10 to value = value + 1, the value of the index 8, 10 is set to 1 from 0
 - I find one of the hash value of element and x has a collision with one of the element y in the index 6, so I set the value of index 6 to 2 from 1
 - In this condition if we want to check whether an element is in the set using bloom filter, we need to check whether all the 3 hash values of that element is larger than 0 (instead of equal to 1 as previous)



- After modification, when we delete the element **x**, we set the 3 hash values of the element **x** : `value = value - 1`



- and lookup whether the element **y** is in the set now
- the bloom filter will return “the element **y** is in the set”.
- now deletion is supported in the bit-vector form of the Bloom filter.

2. Consider the following simple C program compiled for a 32 bit x86 Linux machine (64 bit just mean longer addresses with more zeroes to write down):

```
#include <stdio.h>

int globvar = 42;

int foo(int arg)
{
    return globvar + arg;
}

main()
{
    foo(10);
    printf("%d\n", globvar);
}
```

And the relocation section of the “.o” file is:


```
[wongwf@asura ~]$ objdump -d PIC-example.o

PIC-example.o:      file format elf32-i386


Disassembly of section .text:

00000000 <foo>:
  0:  55                push    %ebp
  1:  89 e5             mov     %esp,%ebp
  3:  8b 15 00 00 00 00 mov     0x0,%edx
  9:  8b 45 08           mov     0x8(%ebp),%eax
 c:  01 d0             add     %edx,%eax
 e:  5d               pop     %ebp
 f:  c3               ret

00000010 <main>:
 10:  55                push    %ebp
 11:  89 e5             mov     %esp,%ebp
 13:  83 e4 f0           and     $0xffffffff0,%esp
 16:  83 ec 10           sub     $0x10,%esp
 19:  c7 04 24 0a 00 00 00 movl    $0xa,(%esp)
 20:  e8 fc ff ff ff     call    21 <main+0x11>
 25:  a1 00 00 00 00     mov     0x0,%eax
 2a:  89 44 24 04         mov     %eax,0x4(%esp)
 2e:  c7 04 24 00 00 00 00 movl    $0x0,(%esp)
 35:  e8 fc ff ff ff     call    36 <main+0x26>
 3a:  c9               leave
 3b:  c3               ret
```

And the relocation section of the “.o” file is:

```
Relocation section '.rel.text' at offset 0x1f8 contains 5 entries:
 Offset      Info      Type           Sym.Value    Sym. Name
00000005     00000901 R_386_32        00000000     globvar
00000021     00000a02 R_386_PC32      00000000     foo
00000026     00000901 R_386_32        00000000     globvar
00000031     00000501 R_386_32        00000000     .rodata
00000036     00000c02 R_386_PC32      00000000     printf
```

(i). Show the resultant binary of the function “foo” after the first THREE (3) relocation is applied, assuming that the linker decides to allocate globvar to the address 0x4fac0 and foo to address 0x8082ab0.

solution:

- After the first three relocation is applied, the resultant binary of the function “foo” is as followed:
- According to second entries in the relocation section, we need patch the value at offset 0x5 with the address of symbol foo
- Because the address is 0x8082ab0 (big endian), we need to transform into little endian in 32 bit. It is 0x **c0 fa 04 00**.

```
00000000 <foo>:
0: 55
1: 89 e5
3: 8b 15 c0 fa 04 00
9: 8b 45 08
c: 01 d0
e: 5d
f: c3
```

(ii). What do you think the 4-th relocation record is for?

solution:

- An executable file usually has a few segments, one of the segment is the read-only one for read-only data
- The name of the 4-th relocation is `.rodata`, it is for the read only data “%d\n”

(iii). Show what the GOT and PLT may look like at runtime (before the program begins execution, assuming lazy binding). Write down your assumptions about where things (such as the function printf) are allocated.

solution:

- step 1: when the program begin execution initially, before main call the print function:
 - GOT look like as followed:
 - GOT[0]: address of the program’s `.dynamic segment` ;
assume the address of the program’s `.dynamic segment` is 0x00010000
 - GOT[1]: pointer to a linked list of nodes corresponding to the symbol tables for each shared library linked with the program
`_dl_runtime_resolve()`
assume the address of `_dl_runtime_resolve()` is 0x00020000

- GOT[2]: address of the symbol resolution function within the dynamic linker.

assume the address of this is 0x00030000

- **assume the index of printf in GOT is 50**

index	value
0	0x 00 01 00 00
1	0x 00 02 00 00
2	0x 00 03 00 00
...	...
50	address of “pushl 50 of PLT[1]”

- PLT look like as followed:

- **assume the index of printf in PLT is 1**

index	value
0	pushl GOT[1] jmp *(GOT[2])
1	jmp *(GOT[50]) pushl 50 jmp PLT[0]

- step 2: when main first call the print function

assume the allocated address of printf is 0x00040000

- GOT look like as followed:

- the `_dl_runtime_resolve()` will find the final allocated address of printf , and using the relocation information patch the GOT[50] as followed:

index	value
0	0x 00 01 00 00
1	0x 00 02 00 00
2	0x 00 03 00 00
...	...
50	0x 00 04 00 00

- PLT look like as followed:

index	value
0	pushl GOT[1] jmp *(GOT[2])
1	jmp *(GOT[50]) pushl 50 jmp PLT[0]