

Report 1-B

0710012 電資 11 何權祐

Video link: <https://youtu.be/HIbxGNPOhXs>

Questions:

1. What is a kernel function? What is a system call?

Kernel function: function that is called in the kernel operation, including access computer resource, memory management, and so on.

System call: syscall, the interface between process and OS, when the user processes need the service from OS, it will use syscall to send request to the OS kernel, call kernel function, each kernel function has its syscall number

2. What is KASLR? What is it for?

kernel address space layout randomization, to randomly place the address of the kernel, having a little offset from the link address, to make sure the outside won't have direct access to the kernel

3. What are GDB's non-stop and all-stop modes?

Non-stop mode: when the thread stops to report a debugging event, only the thread is stopped

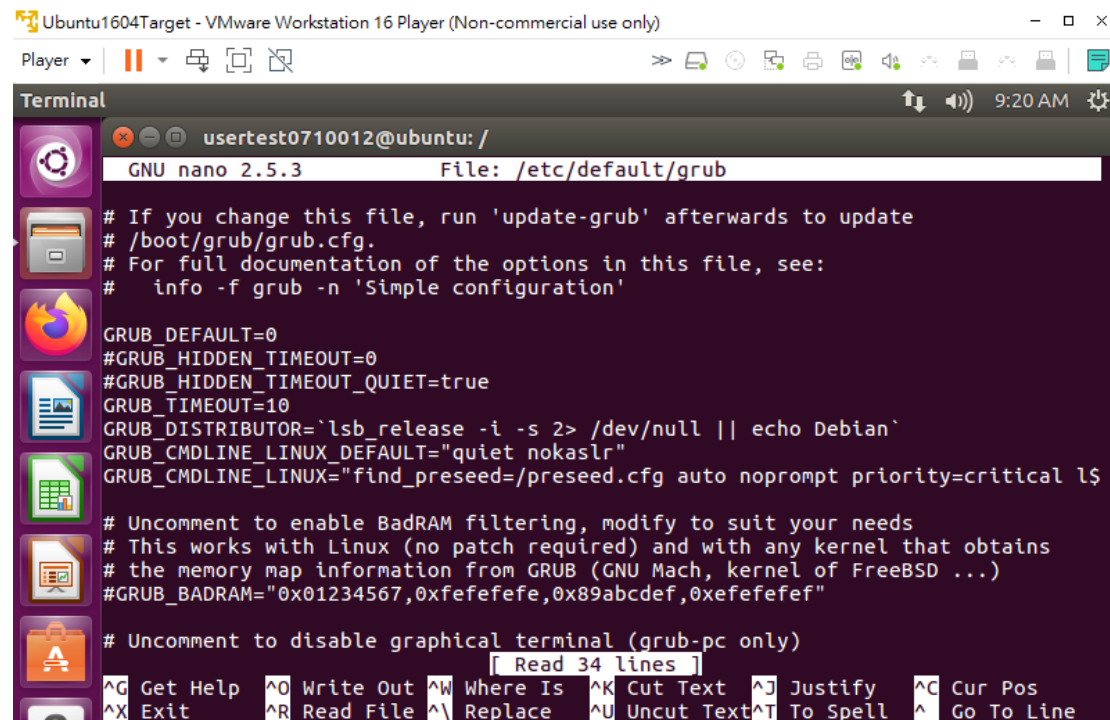
All-stop mode: when the program stops under GDB for any reason, all threads of execution stop

4. Explain what the command **echo g > /proc/sysrq-trigger** does.

The command gives back the control of the vm back to the machine which is debugging the current machine with gdb

Screenshot 1:

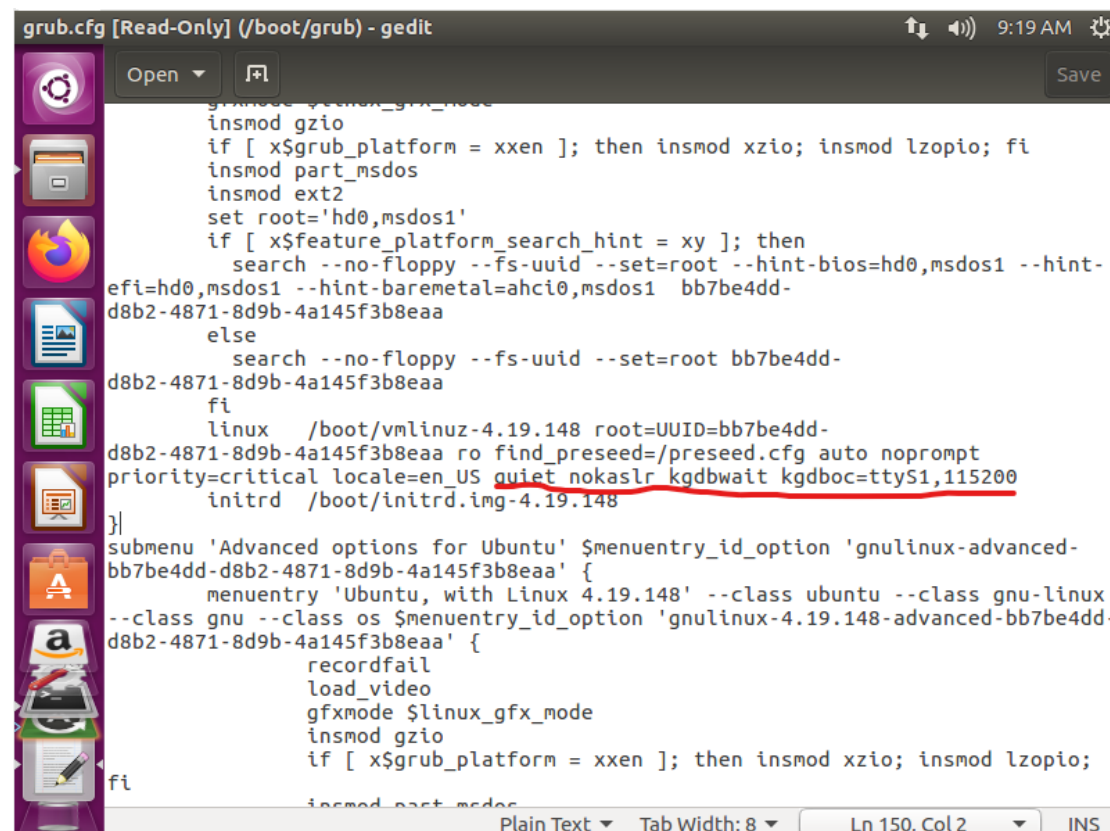
Add "nokaslr" in the comment to make sure that the host machine can directly debug the kernel in target machine



```
user@ubuntu: /  
GNU nano 2.5.3 File: /etc/default/grub  
  
# If you change this file, run 'update-grub' afterwards to update  
# /boot/grub/grub.cfg.  
# For full documentation of the options in this file, see:  
# info -f grub -n 'Simple configuration'  
  
GRUB_DEFAULT=0  
#GRUB_HIDDEN_TIMEOUT=0  
#GRUB_HIDDEN_TIMEOUT_QUIET=true  
GRUB_TIMEOUT=10  
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`  
GRUB_CMDLINE_LINUX_DEFAULT="quiet nokaslr"  
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical ls"  
  
# Uncomment to enable BadRAM filtering, modify to suit your needs  
# This works with Linux (no patch required) and with any kernel that obtains  
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)  
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"  
  
# Uncomment to disable graphical terminal (grub-pc only)  
#GRUB_DISABLE_GRAPHICAL_TERMINAL=true  
  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Screenshot 2:

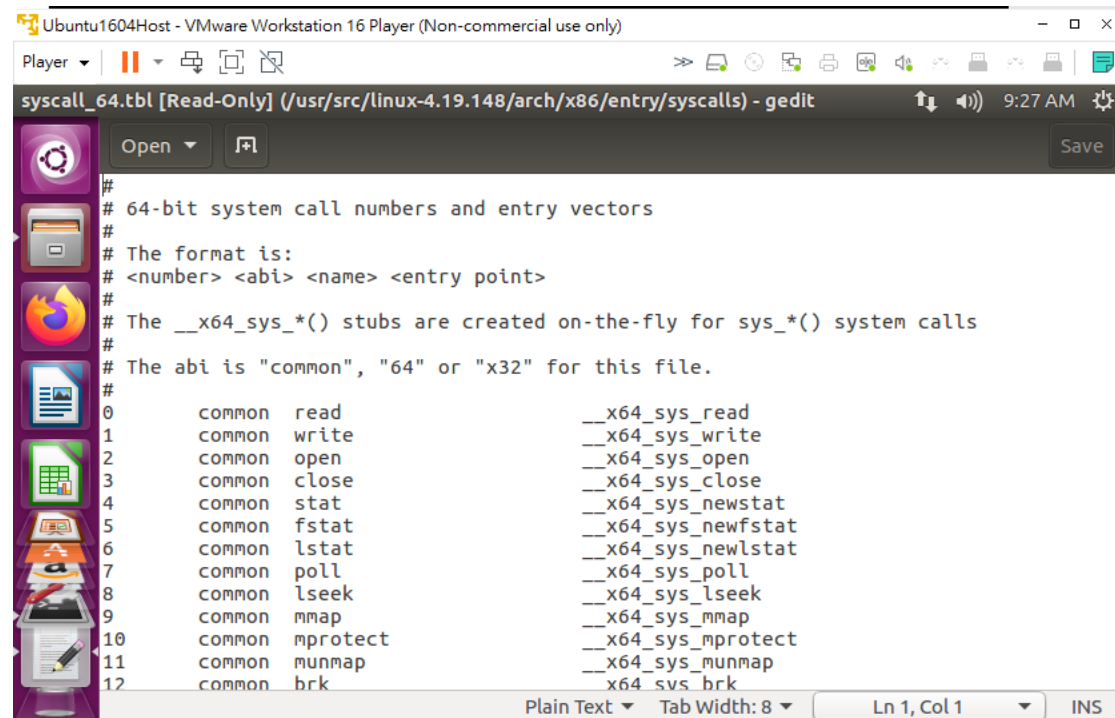
Same as screenshot 1, to make sure that the host can get direct access to the kernel in target



```
grub.cfg [Read-Only] (/boot/grub) - gedit  
Open Save  
  
insmod gzio  
if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi  
insmod part_msdos  
insmod ext2  
set root='hd0,msdos1'  
if [ x$feature_platform_search_hint = xy ]; then  
search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-  
efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa  
else  
search --no-floppy --fs-uuid --set=root bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa  
fi  
linux /boot/vmlinuz-4.19.148 root=UUID=bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa ro find_preseed=/preseed.cfg auto noprompt  
priority=critical locale=en_US quiet nokaslr kgdbwait kgdboc=ttyS1,115200  
initrd /boot/initrd.img-4.19.148  
}  
submenu 'Advanced options for Ubuntu' $menuentry_id_option 'gnulinux-advanced-  
bb7be4dd-d8b2-4871-8d9b-4a145f3b8eaa' {  
menuentry 'Ubuntu, with Linux 4.19.148' --class ubuntu --class gnu-linux  
--class gnu --class os $menuentry_id_option 'gnulinux-4.19.148-advanced-bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa' {  
recordfail  
load_video  
gfxmode $linux_gfx_mode  
insmod gzio  
if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio;  
fi  
insmod part_msdos  
insmod ext2  
set root='hd0,msdos1'  
if [ x$feature_platform_search_hint = xy ]; then  
search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-  
efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa  
else  
search --no-floppy --fs-uuid --set=root bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa  
fi  
linux /boot/vmlinuz-4.19.148 root=UUID=bb7be4dd-  
d8b2-4871-8d9b-4a145f3b8eaa ro find_preseed=/preseed.cfg auto noprompt  
priority=critical locale=en_US quiet nokaslr kgdbwait kgdboc=ttyS1,115200  
initrd /boot/initrd.img-4.19.148  
}  
}  
fi
```

Screenshot 3:

Directly check the kernel functions in the kernel package



```
#
# 64-bit system call numbers and entry vectors
#
# The format is:
# <number> <abi> <name> <entry point>
#
# The __x64_sys_*() stubs are created on-the-fly for sys_*() system calls
#
# The abi is "common", "64" or "x32" for this file.
#
0      common  read      __x64_sys_read
1      common  write     __x64_sys_write
2      common  open      __x64_sys_open
3      common  close     __x64_sys_close
4      common  stat      __x64_sys_newstat
5      common  fstat     __x64_sys_newfstat
6      common  lstat     __x64_sys_newlstat
7      common  poll      __x64_sys_poll
8      common  lseek     __x64_sys_lseek
9      common  mmap      __x64_sys_mmap
10     common  mprotect  __x64_sys_mprotect
11     common  munmap    __x64_sys_munmap
12     common  brk       __x64_sys_brk
```

Screenshot 4:

Make sure the syscall tables find online are coherent to one another

79	getcwd	man/ cs/	0x4f	char *buf	unsigned long size
80	chdir	man/ cs/	0x50	const char *filename	-
81	fchdir	man/ cs/	0x51	unsigned int fd	-
82	rename	man/ cs/	0x52	const char *oldname	const char *newname
83	mkdir	man/ cs/	0x53	const char *pathname	umode_t mode
84	rmdir	man/ cs/	0x54	const char *pathname	-
85	creat	man/ cs/	0x55	const char *pathname	umode_t mode

Screenshot 5:

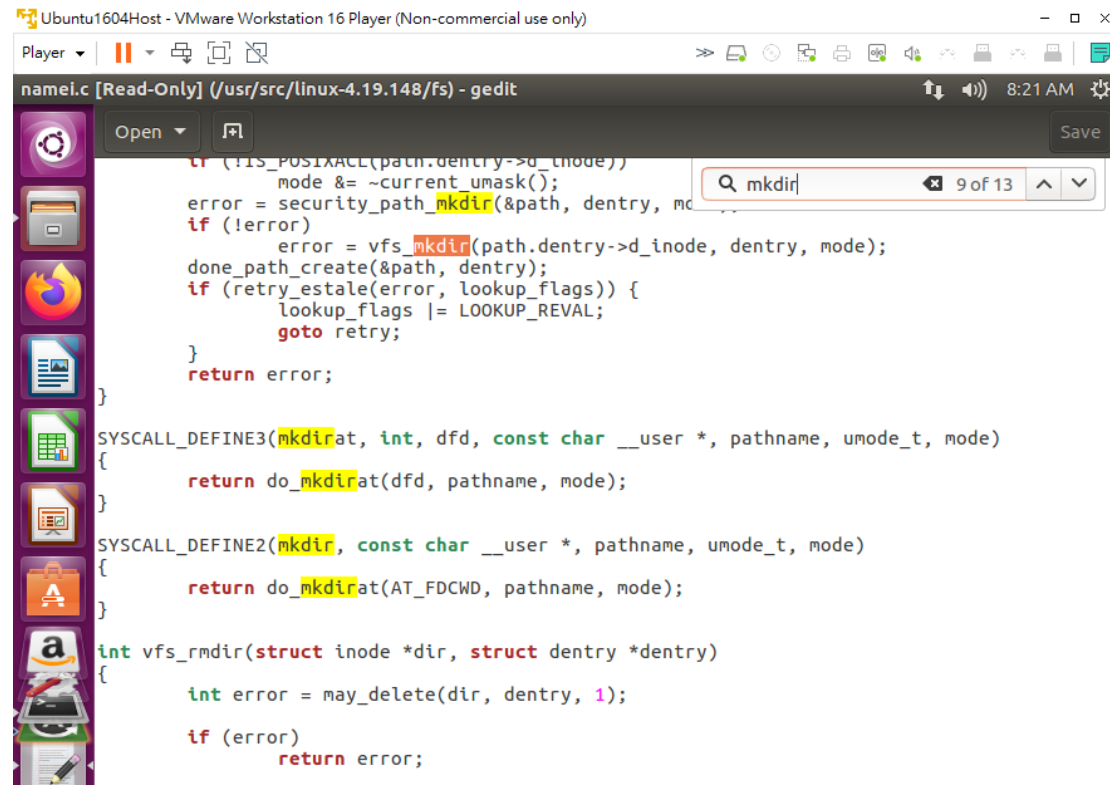
76	sys_truncate	const char *path	long length	
77	sys_ftruncate	unsigned int fd	unsigned long length	
78	sys_getdents	unsigned int fd	struct linux_dirent *dirent	unsigned int count
79	sys_getcwd	char *buf	unsigned long size	
80	sys_chdir	const char *filename		
81	sys_fchdir	unsigned int fd		
82	sys_rename	const char *oldname	const char *newname	
83	sys_mkdir	const char *pathname	int mode	
84	sys_rmdir	const char *pathname		
85	sys_creat	const char *pathname	int mode	
86	sys_link	const char *oldname	const char *newname	
87	sys_unlink	const char *pathname		

Screenshot 6:

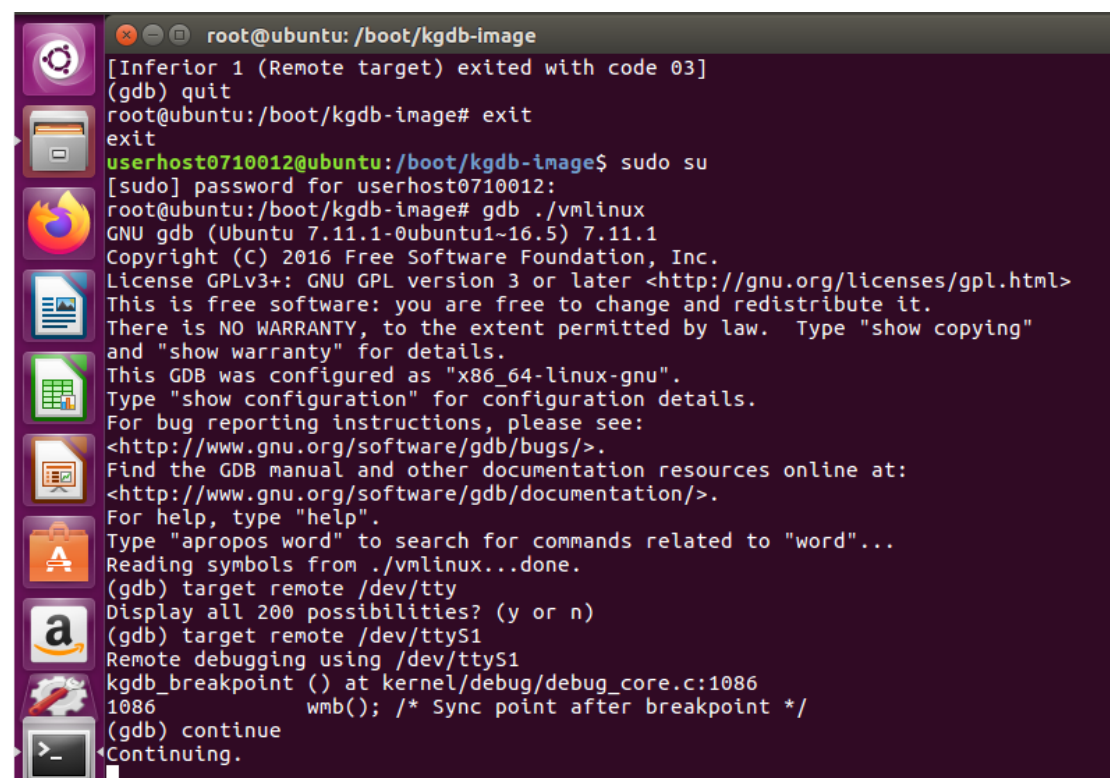
76	truncate	sys_truncate	fs/open.c
77	ftruncate	sys_ftruncate	fs/open.c
78	getdents	sys_getdents	fs/readdir.c
79	getcwd	sys_getcwd	fs/dcache.c
80	chdir	sys_chdir	fs/open.c
81	fchdir	sys_fchdir	fs/open.c
82	rename	sys_rename	fs/namei.c
83	mkdir	sys_mkdir	fs/namei.c
84	rmdir	sys_rmdir	fs/namei.c
85	creat	sys_creat	fs/open.c
86	link	sys_link	fs/namei.c
87	unlink	sys_unlink	fs/namei.c

Screenshot 7:

As we are making a break point in mkdir, we have to find the actual function that the syscall call, which is do_mkdirat, so we use command "break do_mkdirat" to set up the break point



```
namei.c [Read-Only] (/usr/src/linux-4.19.148/fs) - gedit
Open Save
if (IS_POSIXACL(path.dentry->d_inode))
    mode &= ~current_umask();
error = security_path_mkdir(&path, dentry, mode);
if (!error)
    error = vfs_mkdir(path.dentry->d_inode, dentry, mode);
done_path_create(&path, dentry);
if (retry_estale(error, lookup_flags)) {
    lookup_flags |= LOOKUP_REVAL;
    goto retry;
}
return error;
}
SYSCALL_DEFINE3(mkdirat, int, dfd, const char __user *, pathname, umode_t, mode)
{
    return do_mkdirat(dfd, pathname, mode);
}
SYSCALL_DEFINE2(mkdir, const char __user *, pathname, umode_t, mode)
{
    return do_mkdirat(AT_FDCWD, pathname, mode);
}
int vfs_rmdir(struct inode *dir, struct dentry *dentry)
{
    int error = may_delete(dir, dentry, 1);
    if (error)
        return error;
}
```



```
root@ubuntu: /boot/kgdb-image
[Inferior 1 (Remote target) exited with code 03]
(gdb) quit
root@ubuntu: /boot/kgdb-image# exit
exit
userhost0710012@ubuntu: /boot/kgdb-image$ sudo su
[sudo] password for userhost0710012:
root@ubuntu: /boot/kgdb-image# gdb ./vmlinux
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./vmlinux...done.
(gdb) target remote /dev/tty
Display all 200 possibilities? (y or n)
(gdb) target remote /dev/ttyS1
Remote debugging using /dev/ttyS1
kgdb_breakpoint () at kernel/debug/debug_core.c:1086
1086          wmb(); /* Sync point after breakpoint */
(gdb) continue
Continuing.
```

Screenshot 8:

When the break point is hit, the path will show where it is hit, the example is that we are trying to call mkdir in desktop by creating a new folder

Ubuntu1604Host - VMware Workstation 16 Player (Non-commercial use only)

Player ▾ | [Icons: Play, Full Screen, Toggle Full Screen, Exit Full Screen, Power, Snapshot, Undo, Redo, Find, Print, Copy, Paste, Close, Help, Settings, About]

Terminal [Icons: Up Arrow, Speaker] 9:07 AM [Settings Icon]

```

root@ubuntu: /boot/kgdb-image

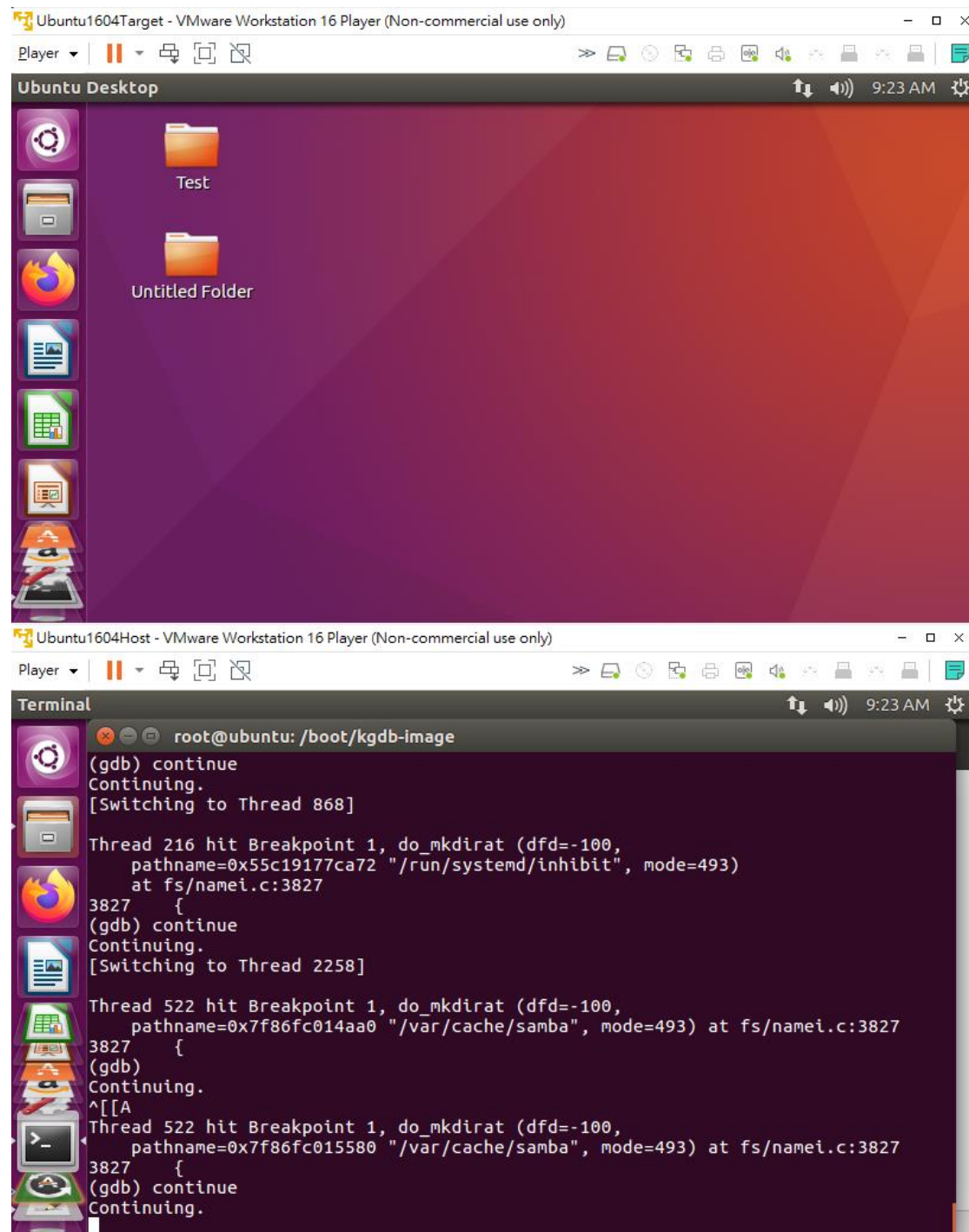
Thread 191 hit Breakpoint 1, do_mkdirat (dfd=-100,
    pathname=0x14b27a0 "/tmp/vmware-root", mode=448) at fs/namei.c:3827
3827 {
(gdb) continue
Continuing.
[New Thread 2216]
[New Thread 2217]

Thread 191 hit Breakpoint 1, do_mkdirat (dfd=-100,
    pathname=0x14aedf0 "/tmp/vmware-root", mode=448) at fs/namei.c:3827
3827 {
(gdb) continue
Continuing.
[New Thread 2220]
[New Thread 2218]
[Switching to Thread 2220]

Thread 498 hit Breakpoint 1, do_mkdirat (dfd=-100,
    pathname=0x7f1afc004800 "/home/user/test0710012/Desktop/Untitled Folder",
    mode=511) at fs/namei.c:3827
3827 {
(gdb) print pathname
$3 = 0x7f1afc004800 "/home/user/test0710012/Desktop/Untitled Folder"
(gdb)
  
```

Screenshot 9:

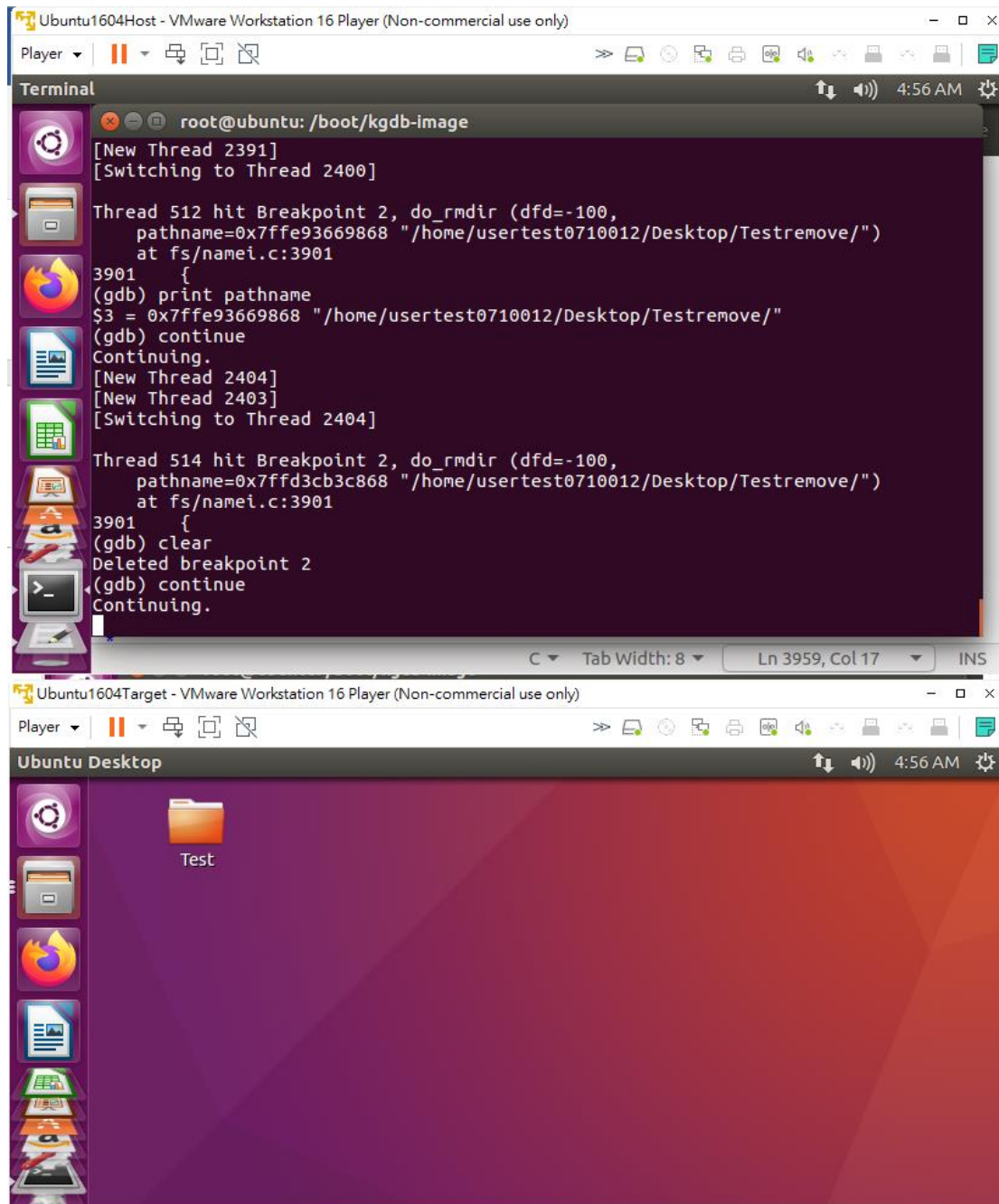
Actually we get through many break point before we reach the status that the target can back to normal mode, as the mkdir is called in many process that we don't know, however, at last, the folder is create successfully and the target resume to work mode



Do it yourself:

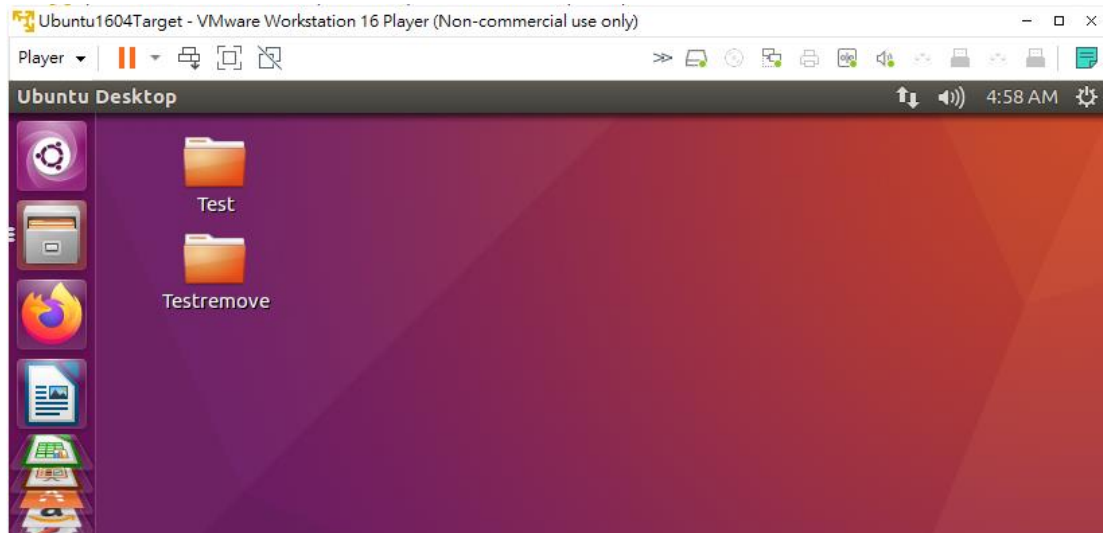
Screenshot 10:

The machine is responsive



Screenshot 11:

I choose the “rmdir” syscall function as the “do it yourself” example, and I plan to display the example by deleting a folder from the desktop, therefore, there is a folder called “Testremove” to help me explain the idea, and enter the rmdir syscall to hit the breakpoint



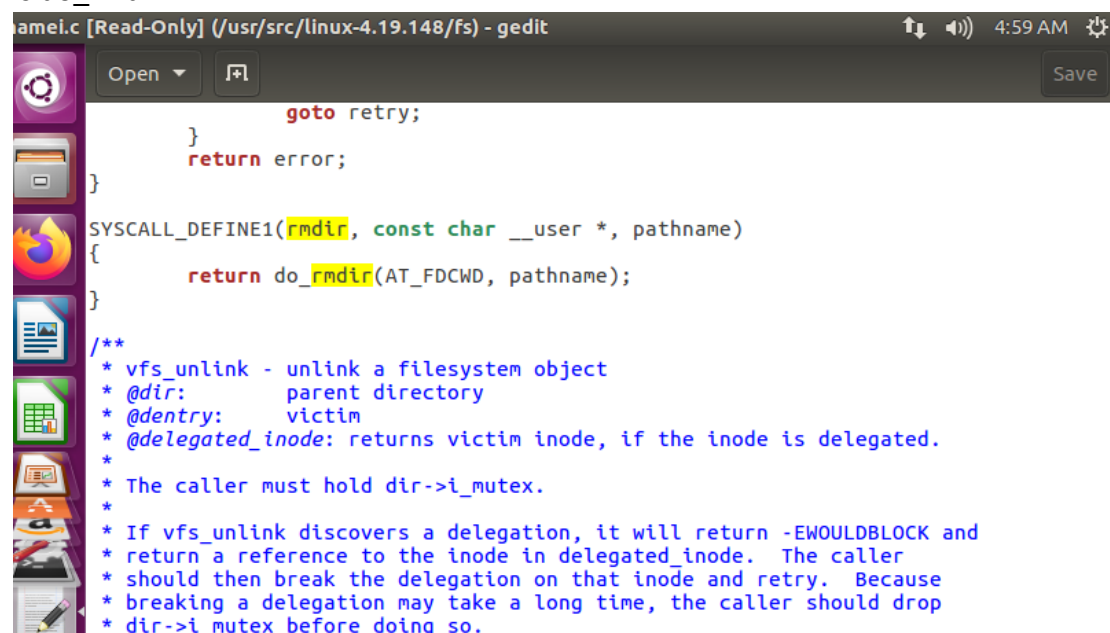
```

root@ubuntu: /
root@ubuntu:/home/userstest0710012/bin# cd ../
root@ubuntu:/home/userstest0710012# cd desktop
bash: cd: desktop: No such file or directory
root@ubuntu:/home/userstest0710012# cd Desktop
root@ubuntu:/home/userstest0710012/Desktop# ls
Test
root@ubuntu:/home/userstest0710012/Desktop# mkdir Testremove
root@ubuntu:/home/userstest0710012/Desktop# cd ../
root@ubuntu:/home/userstest0710012# cd ../
root@ubuntu:/home# cd ../
root@ubuntu:/# ls
bin    dev    initrd.img    lib64    mnt    root    snap    tmp    vmlinuz
boot  etc    initrd.img.old  lost+found  opt    run    srv    usr    vmlinuz.old
cdrom  home  lib          media    proc  sbin   sys    var
root@ubuntu:/# rmdir /home/userstest0710012/Desktop/Testremove/
root@ubuntu:/# rmdir /home/userstest0710012/Desktop/Testremove/
rmdir: failed to remove '/home/userstest0710012/Desktop/Testremove/': No such file or directory
root@ubuntu:/# rmdir /home/userstest0710012/Desktop/Testremove/

```

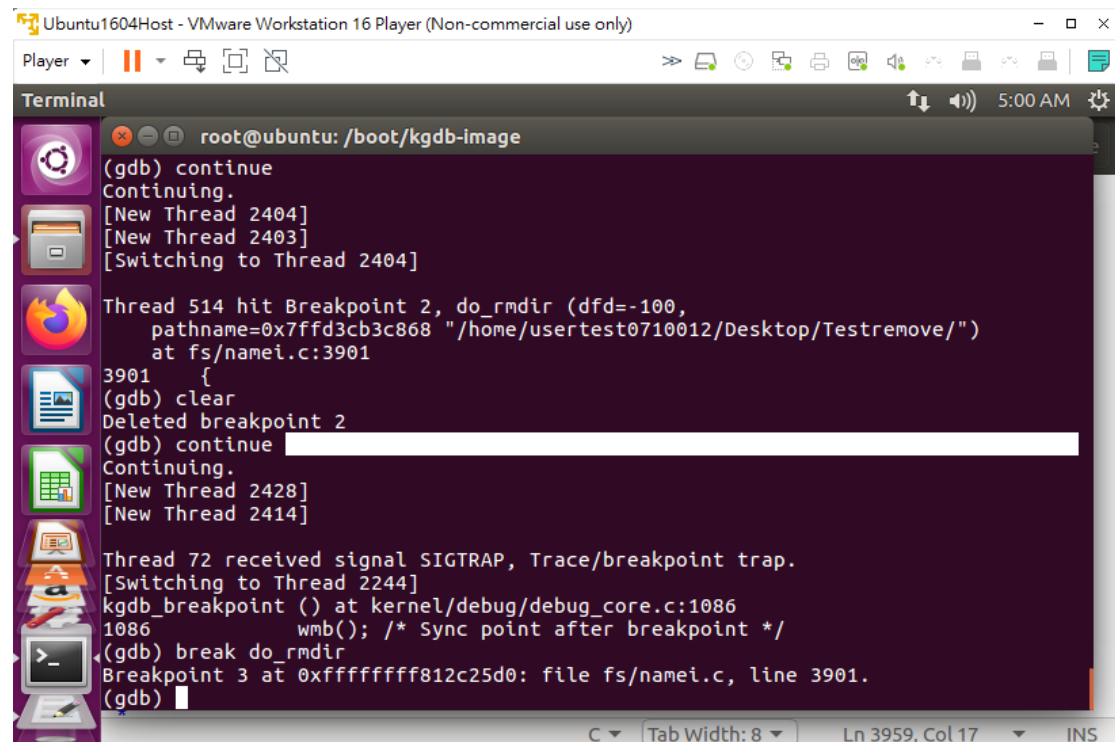
Screenshot 12:

Find the rmdir function in the syscall file and find out that the actual function it calls is do_rmdir



Screenshot 13:

Create a break point at do_rmdir



The screenshot shows a GDB terminal window titled 'Terminal' with the prompt 'root@ubuntu: /boot/kgdb-image'. The terminal output includes the following commands and messages:

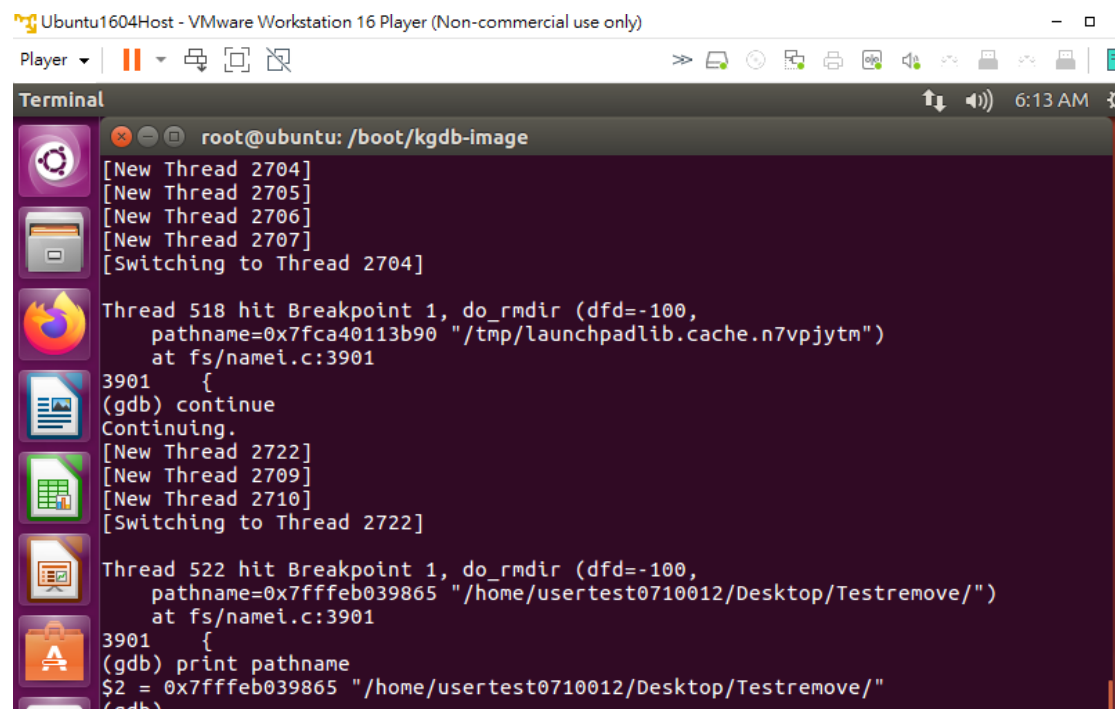
```
(gdb) continue
Continuing.
[New Thread 2404]
[New Thread 2403]
[Switching to Thread 2404]

Thread 514 hit Breakpoint 2, do_rmdir (dfd=-100,
    pathname=0x7ffd3cb3c868 "/home/userstest0710012/Desktop/Testremove/")
    at fs/namei.c:3901
3901 {
(gdb) clear
Deleted breakpoint 2
(gdb) continue
Continuing.
[New Thread 2428]
[New Thread 2414]

Thread 72 received signal SIGTRAP, Trace/breakpoint trap.
[Switching to Thread 2244]
kgdb_breakpoint () at kernel/debug/debug_core.c:1086
1086      wmb(); /* Sync point after breakpoint */
(gdb) break do_rmdir
Breakpoint 3 at 0xffffffff812c25d0: file fs/namei.c, line 3901.
(gdb)
```

Screenshot 14:

The path name showed that the breakpoint in rmdir function is hit in
/home/userstest0710012/Desktop/Testremove



The screenshot shows a GDB terminal window titled 'Terminal' with the prompt 'root@ubuntu: /boot/kgdb-image'. The terminal output includes the following commands and messages:

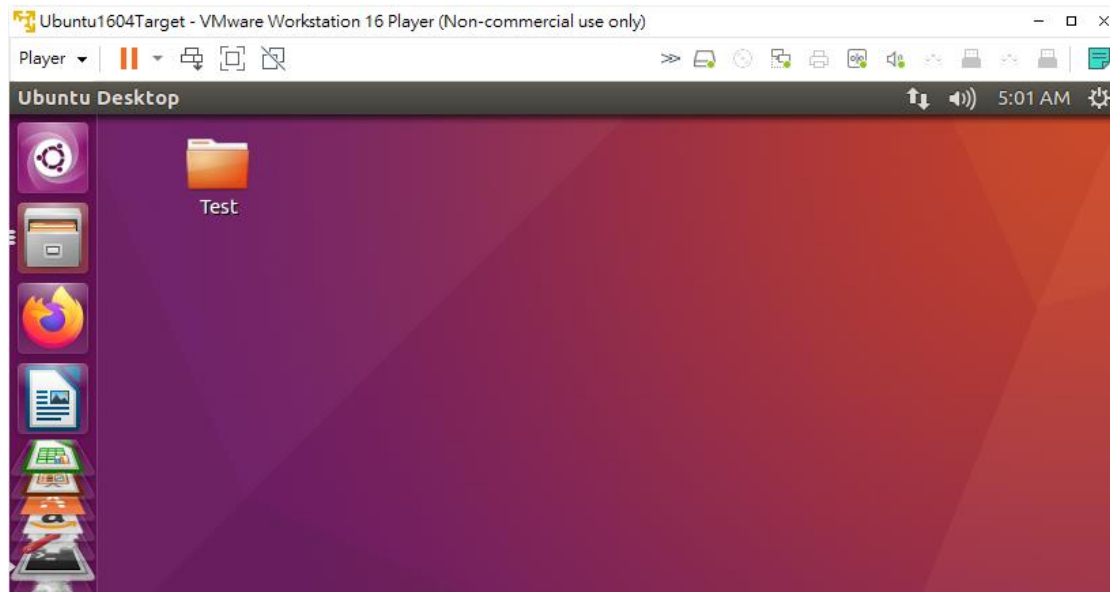
```
[New Thread 2704]
[New Thread 2705]
[New Thread 2706]
[New Thread 2707]
[Switching to Thread 2704]

Thread 518 hit Breakpoint 1, do_rmdir (dfd=-100,
    pathname=0x7fca40113b90 "/tmp/launchpadlib.cache.n7vpjytm")
    at fs/namei.c:3901
3901 {
(gdb) continue
Continuing.
[New Thread 2722]
[New Thread 2709]
[New Thread 2710]
[Switching to Thread 2722]

Thread 522 hit Breakpoint 1, do_rmdir (dfd=-100,
    pathname=0x7fffeb039865 "/home/userstest0710012/Desktop/Testremove/")
    at fs/namei.c:3901
3901 {
(gdb) print pathname
$2 = 0x7fffeb039865 "/home/userstest0710012/Desktop/Testremove/"
(gdb)
```

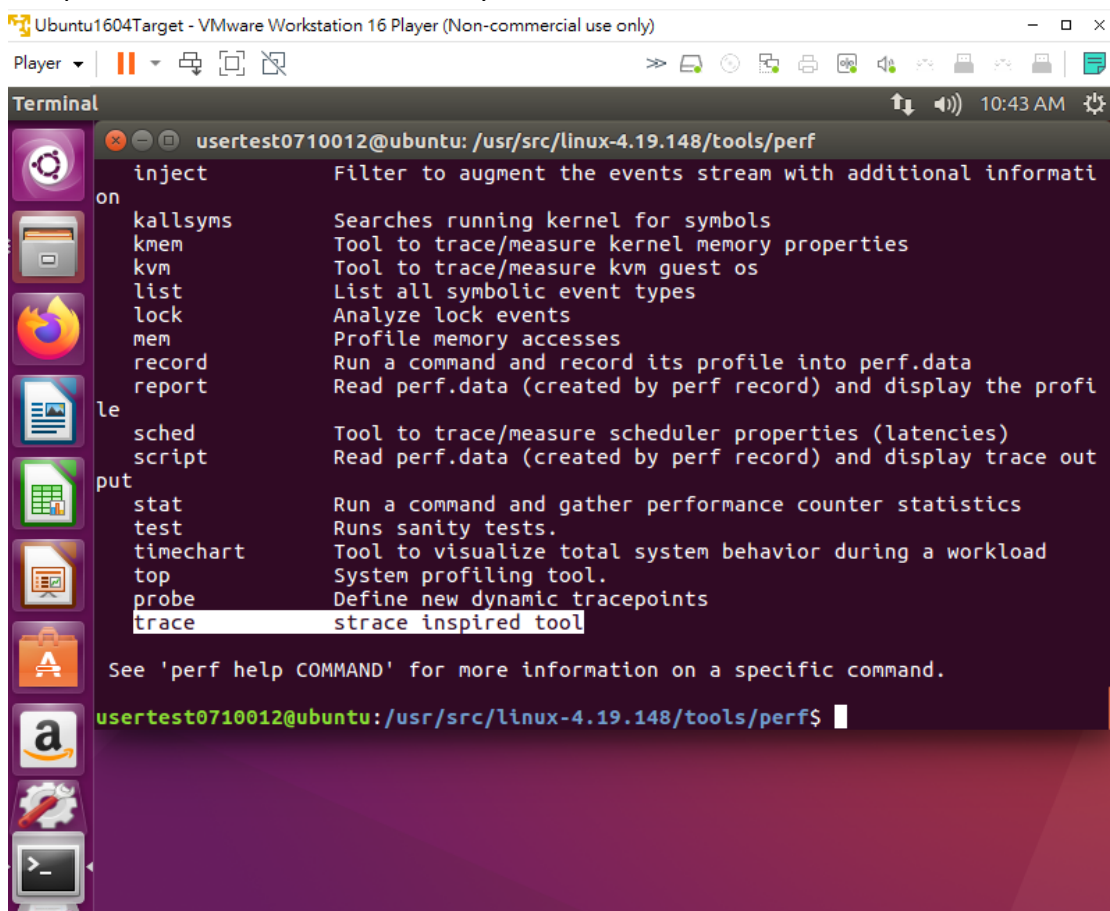
Screenshot 15:

The "Testremove" folder is successfully removed



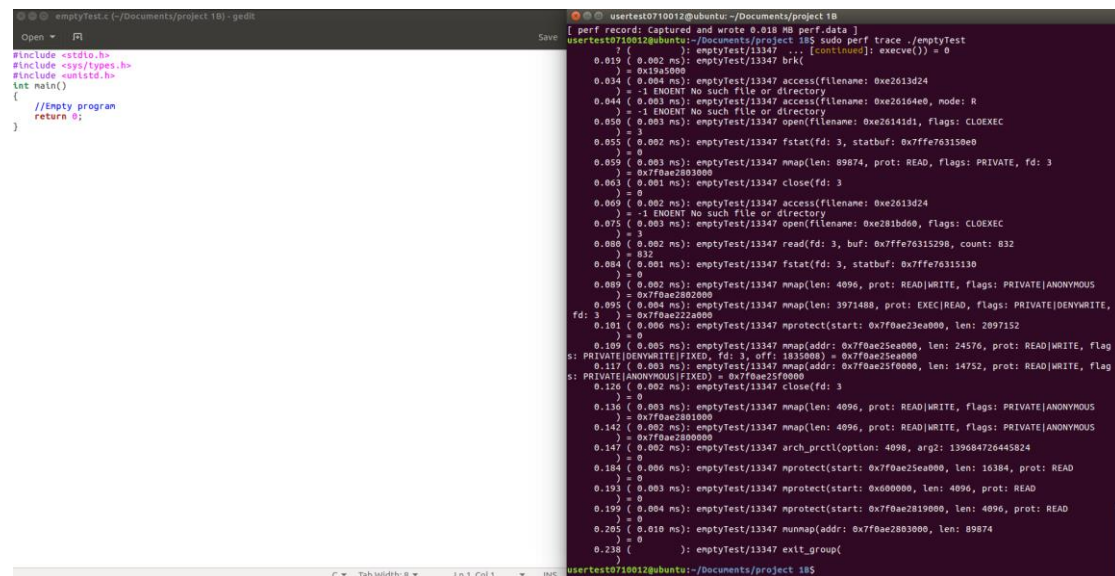
Screenshot 16:

The perf trace function is successfully installed



Screenshot 17:

The program code and the trace result of emptyTest.c

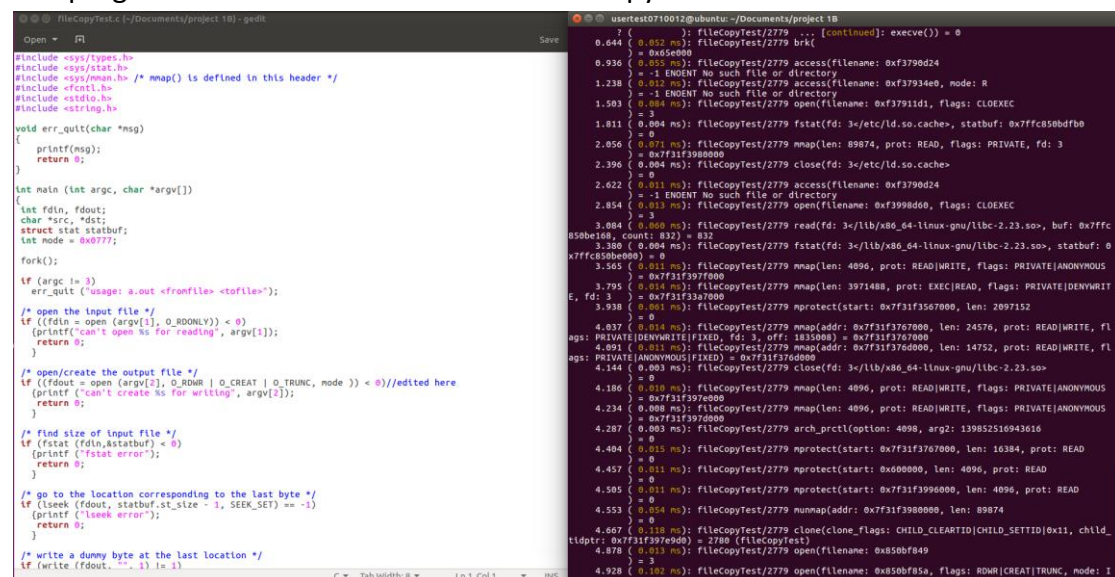


```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    //Empty program
    return 0;
}
```

```
[ perf record: Captured and wrote 0.018 MB perf.data ]
userstest0710012@ubuntu: ~/Documents/project 18$ sudo perf trace ./emptyTest
0.019 ( 0.002 ms): emptyTest/13347 ... [continued]: execve(0) = 0
0.019 ( 0.002 ms): emptyTest/13347 brk(
0.034 ( 0.004 ms): emptyTest/13347 access(filename: 0xe2613d24
0.044 ( 0.003 ms): emptyTest/13347 access(filename: 0xe26164e0, mode: R
0.050 ( 0.003 ms): emptyTest/13347 open(filename: 0xe26141d1, flags: CLOEXEC
0.055 ( 0.002 ms): emptyTest/13347 fstat(fd: 3, statbuf: 0x7ffe763150e0
0.059 ( 0.003 ms): emptyTest/13347 mmap(len: 89874, prot: READ, flags: PRIVATE, fd: 3
0.063 ( 0.001 ms): emptyTest/13347 close(fd: 3
0.069 ( 0.002 ms): emptyTest/13347 access(filename: 0xe2613d24
0.075 ( 0.003 ms): emptyTest/13347 open(filename: 0xe281bd60, flags: CLOEXEC
0.080 ( 0.002 ms): emptyTest/13347 read(fd: 3, buf: 0x7ffe76315298, count: 832
0.084 ( 0.001 ms): emptyTest/13347 fstat(fd: 3, statbuf: 0x7ffe76315130
0.089 ( 0.002 ms): emptyTest/13347 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS
0.095 ( 0.004 ms): emptyTest/13347 mmap(len: 3971488, prot: EXEC|READ, flags: PRIVATE|DENYWRITE,
fd: 3
0.101 ( 0.006 ms): emptyTest/13347 mprotect(start: 0x7f0ae23ea000, len: 2097152
0.109 ( 0.005 ms): emptyTest/13347 mmap(addr: 0x7f0ae23ea000, len: 24576, prot: READ|WRITE, flag
s: PRIVATE|DENYWRITE|FIXED, fd: 3, off: 1835008) = 0x7f0ae23ea000
0.117 ( 0.003 ms): emptyTest/13347 mmap(addr: 0x7f0ae25f0000, len: 14752, prot: READ|WRITE, flag
s: PRIVATE|ANONYMOUS|FIXED) = 0x7f0ae25f0000
0.120 ( 0.002 ms): emptyTest/13347 close(fd: 3
0.130 ( 0.003 ms): emptyTest/13347 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS
0.142 ( 0.002 ms): emptyTest/13347 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS
0.147 ( 0.002 ms): emptyTest/13347 arch_prctl(option: 4098, arg2: 139684726445824
0.184 ( 0.006 ms): emptyTest/13347 mprotect(start: 0x7f0ae23ea000, len: 16384, prot: READ
0.190 ( 0.003 ms): emptyTest/13347 mprotect(start: 0x00000000, len: 4096, prot: READ
0.199 ( 0.004 ms): emptyTest/13347 mprotect(start: 0x7f0ae2819000, len: 4096, prot: READ
0.205 ( 0.010 ms): emptyTest/13347 munmap(addr: 0x7f0ae2803000, len: 89874
0.238 ( 0
): emptyTest/13347 exit_group(
```

Screenshot 18

The program code and the trace result of fileCopyTest.c



```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h> /* mmap() is defined in this header */
#include <fcntl.h>
#include <stdio.h>
#include <string.h>

void err_quit(char *msg)
{
    printf(msg);
    return 0;
}

int main(int argc, char *argv[])
{
    int fdin, fdout;
    char *src, *dst;
    struct stat statbuf;
    int mode = 0x0777;

    fork();

    if (argc != 3)
        err_quit("usage: a.out <fromfile> <tofile>");
    if ((fdin = open(argv[1], O_RDONLY)) < 0)
        err_quit("can't open %s for reading", argv[1]);
    /* open/create the output file */
    if ((fdout = open(argv[2], O_RDWR | O_CREAT | O_TRUNC, mode)) < 0) //edited here
        err_quit("can't create %s for writing", argv[2]);
    /* find size of input file */
    if (fstat(fdin, statbuf) < 0)
        err_quit("fstat error");
    /* go to the location corresponding to the last byte */
    if (lseek(fdout, statbuf.st_size - 1, SEEK_SET) == -1)
        err_quit("lseek error");
    /* write a dummy byte at the last location */
    if (write(fdout, "\n", 1) != 1)
```

```
userstest0710012@ubuntu: ~/Documents/project 18$
0.644 ( 0.012 ms): fileCopyTest/2779 ... [continued]: execve(0) = 0
0.936 ( 0.015 ms): fileCopyTest/2779 access(filename: 0xf379bd24
1.238 ( 0.012 ms): fileCopyTest/2779 access(filename: 0xf37934e0, mode: R
1.583 ( 0.004 ms): fileCopyTest/2779 open(filename: 0xf37911d1, flags: CLOEXEC
1.811 ( 0.004 ms): fileCopyTest/2779 fstat(fd: 3=/etc/ld.so.cache, statbuf: 0x7ffc850bdfb0
2.056 ( 0.071 ms): fileCopyTest/2779 mmap(len: 89874, prot: READ, flags: PRIVATE, fd: 3
2.396 ( 0.004 ms): fileCopyTest/2779 close(fd: 3=/etc/ld.so.cache
2.622 ( 0.011 ms): fileCopyTest/2779 access(filename: 0xf379bd24
2.854 ( 0.013 ms): fileCopyTest/2779 open(filename: 0xf3998d60, flags: CLOEXEC
3.084 ( 0.060 ms): fileCopyTest/2779 read(fd: 3=/lib/x86_64-linux-gnu/libc-2.23.so, buf: 0x7ffc850b0000, count: 832) = 832
3.380 ( 0.004 ms): fileCopyTest/2779 fstat(fd: 3=/lib/x86_64-linux-gnu/libc-2.23.so, statbuf: 0
3.565 ( 0.013 ms): fileCopyTest/2779 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS
3.795 ( 0.014 ms): fileCopyTest/2779 mmap(len: 3971488, prot: EXEC|READ, flags: PRIVATE|DENYWRIT
E, fd: 3
3.938 ( 0.003 ms): fileCopyTest/2779 mprotect(start: 0x7f31f3567000, len: 2097152
4.037 ( 0.014 ms): fileCopyTest/2779 mmap(addr: 0x7f31f3760000, len: 24576, prot: READ|WRITE, fl
ags: PRIVATE|DENYWRITE|FIXED, fd: 3, off: 1835008) = 0x7f31f3760000
4.091 ( 0.013 ms): fileCopyTest/2779 mmap(addr: 0x7f31f3760000, len: 14752, prot: READ|WRITE, fl
ags: PRIVATE|ANONYMOUS|FIXED) = 0x7f31f3760000
4.144 ( 0.003 ms): fileCopyTest/2779 close(fd: 3=/lib/x86_64-linux-gnu/libc-2.23.so
4.186 ( 0.010 ms): fileCopyTest/2779 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS
4.234 ( 0.009 ms): fileCopyTest/2779 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS
4.287 ( 0.003 ms): fileCopyTest/2779 arch_prctl(option: 4098, arg2: 139852516943616
4.404 ( 0.015 ms): fileCopyTest/2779 mprotect(start: 0x7f31f3767000, len: 16384, prot: READ
4.457 ( 0.011 ms): fileCopyTest/2779 mprotect(start: 0x00000000, len: 4096, prot: READ
4.585 ( 0.011 ms): fileCopyTest/2779 mprotect(start: 0x7f31f3996000, len: 4096, prot: READ
4.553 ( 0.054 ms): fileCopyTest/2779 munmap(addr: 0x7f31f3980000, len: 89874
4.667 ( 0.118 ms): fileCopyTest/2779 clone(clone_flags: CHILD_CLEARID|CHILD_SETTID|0x11, child_
tidptr: 0x7f31f3970000) = 2780 (fileCopyTest)
4.878 ( 0.013 ms): fileCopyTest/2779 open(filename: 0x850bf849
4.928 ( 0.102 ms): fileCopyTest/2779 open(filename: 0x850bf85a, flags: RDWR|CREAT|TRUNC, mode: 1
```

Screenshot 19:

We can see that the trace result of fileCopyTest.c has a bulk of text more than emptyTest.c, which should be the actual code that is executed in fileCopyTest.c

1. What are these functions: **clone**, **mmap**, **write** and **open**?

Clone: create new(child) process

Mmap: create a new mapping in the virtual address space of the calling process

Write: write the context in the buffer into the file that is referred to

Open: open the file that is referred to

2. Why is there no **fork** system call? What is the difference between **fork** and **clone**?

As it is unnecessary to create a new process that is totally independent to the origin process, clone can save more resources

The child process created by clone allows them to share parts of its execution context with the calling process

3. Will the functions' execution time be longer if the file is bigger?

Yes, as the data the program take in and print out is different. More data should take more time.

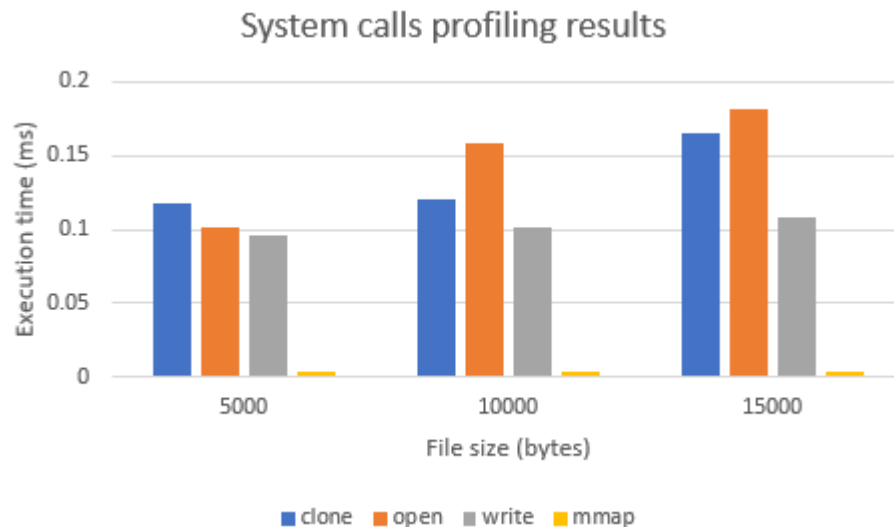
```
emptyTest.txt
1  ? ( ): emptyTest/2604 ... [continued]: execve() = 0
2  0.693 ( 0.098 ms): emptyTest/2604 brk( )
3  1.176 ( 0.141 ms): emptyTest/2604 access(filename: 0x7aa94d24 )
4  1.564 ( 0.013 ms): emptyTest/2604 access(filename: 0x7aa974e0, mode: R )
5  1.799 ( 0.058 ms): emptyTest/2604 open(filename: 0x7aa951d1, flags: CLOEXEC )
6  2.105 ( 0.004 ms): emptyTest/2604 fstat(fd: 3</etc/ld.so.cache>, statbuf: 0x7ffd0cbbfee0 )
7  2.292 ( 0.070 ms): emptyTest/2604 mmap(len: 89874, prot: READ, flags: PRIVATE, fd: 3 )
8  2.400 ( 0.003 ms): emptyTest/2604 close(fd: 3</etc/ld.so.cache> )
9  2.505 ( 0.012 ms): emptyTest/2604 access(filename: 0x7aa94d24 )
10 2.558 ( 0.011 ms): emptyTest/2604 open(filename: 0x7ac9cd60, flags: CLOEXEC )
11 2.605 ( 0.057 ms): emptyTest/2604 read(fd: 3</lib/x86_64-linux-gnu/libc-2.23.so>, buf: 0x7ffd0cbc0098, co
12 2.700 ( 0.003 ms): emptyTest/2604 fstat(fd: 3</lib/x86_64-linux-gnu/libc-2.23.so>, statbuf: 0x7ffd0cbbff3 )
13 2.735 ( 0.010 ms): emptyTest/2604 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS )
14 2.778 ( 0.006 ms): emptyTest/2604 mmap(len: 3971488, prot: EXEC|READ, flags: PRIVATE|DENYWRITE, fd: 3 )
15 2.787 ( 0.007 ms): emptyTest/2604 mprotect(start: 0x7fb07a86b000, len: 2097152 )
16 2.795 ( 0.005 ms): emptyTest/2604 mmap(addr: 0x7fb07aa6b000, len: 24576, prot: READ|WRITE, flags: PRIVATE
17 2.805 ( 0.003 ms): emptyTest/2604 mmap(addr: 0x7fb07aa71000, len: 14752, prot: READ|WRITE, flags: PRIVATE
18 2.815 ( 0.002 ms): emptyTest/2604 close(fd: 3</lib/x86_64-linux-gnu/libc-2.23.so> )
19 2.825 ( 0.003 ms): emptyTest/2604 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS )
20 2.832 ( 0.002 ms): emptyTest/2604 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS )
21 2.837 ( 0.002 ms): emptyTest/2604 arch_prctl(option: 4098, arg2: 140395950909184 )
22 2.875 ( 0.006 ms): emptyTest/2604 mprotect(start: 0x7fb07aa6b000, len: 16384, prot: READ )
23 2.884 ( 0.004 ms): emptyTest/2604 mprotect(start: 0x6000000, len: 4096, prot: READ )
24 2.891 ( 0.005 ms): emptyTest/2604 mprotect(start: 0x7fb07ac9a000, len: 4096, prot: READ )
25 2.898 ( 0.010 ms): emptyTest/2604 munmap(addr: 0x7fb07ac84000, len: 89874 )
26
27
28
29
30
31
32
33
34 2.932 ( ): emptyTest/2604 exit_group( )

fileCopyTest.txt
1  ? ( ): fileCopyTest/2779 ... [continued]: execve() = 0
2  0.644 ( 0.052 ms): fileCopyTest/2779 brk( )
3  0.936 ( 0.055 ms): fileCopyTest/2779 access(filename: 0xf3790d24 )
4  1.238 ( 0.012 ms): fileCopyTest/2779 access(filename: 0xf37934e0, mode: R )
5  1.503 ( 0.084 ms): fileCopyTest/2779 open(filename: 0xf37911d1, flags: CLOEXEC )
6  1.811 ( 0.004 ms): fileCopyTest/2779 fstat(fd: 3</etc/ld.so.cache>, statbuf: 0x7ffc850bdfb0 )
7  2.056 ( 0.071 ms): fileCopyTest/2779 mmap(len: 89874, prot: READ, flags: PRIVATE, fd: 3 )
8  2.396 ( 0.004 ms): fileCopyTest/2779 close(fd: 3</etc/ld.so.cache> )
9  2.622 ( 0.011 ms): fileCopyTest/2779 access(filename: 0xf3790d24 )
10 2.854 ( 0.013 ms): fileCopyTest/2779 open(filename: 0xf3998d60, flags: CLOEXEC )
11 3.084 ( 0.060 ms): fileCopyTest/2779 read(fd: 3</lib/x86_64-linux-gnu/libc-2.23.so>, buf: 0x7ffc850bel68,
12 3.380 ( 0.004 ms): fileCopyTest/2779 fstat(fd: 3</lib/x86_64-linux-gnu/libc-2.23.so>, statbuf: 0x7ffc850b )
13 3.565 ( 0.011 ms): fileCopyTest/2779 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS )
14 3.795 ( 0.014 ms): fileCopyTest/2779 mmap(len: 3971488, prot: EXEC|READ, flags: PRIVATE|DENYWRITE, fd: 3 )
15 3.938 ( 0.061 ms): fileCopyTest/2779 mprotect(start: 0x7f31f3567000, len: 2097152 )
16 4.037 ( 0.014 ms): fileCopyTest/2779 mmap(addr: 0x7f31f3767000, len: 24576, prot: READ|WRITE, flags: PRIV
17 4.091 ( 0.011 ms): fileCopyTest/2779 mmap(addr: 0x7f31f376d000, len: 14752, prot: READ|WRITE, flags: PRIV
18 4.144 ( 0.003 ms): fileCopyTest/2779 close(fd: 3</lib/x86_64-linux-gnu/libc-2.23.so> )
19 4.186 ( 0.010 ms): fileCopyTest/2779 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS )
20 4.234 ( 0.008 ms): fileCopyTest/2779 mmap(len: 4096, prot: READ|WRITE, flags: PRIVATE|ANONYMOUS )
21 4.287 ( 0.003 ms): fileCopyTest/2779 arch_prctl(option: 4098, arg2: 139852516943616 )
22 4.404 ( 0.015 ms): fileCopyTest/2779 mprotect(start: 0x7f31f3767000, len: 16384, prot: READ )
23 4.457 ( 0.011 ms): fileCopyTest/2779 mprotect(start: 0x6000000, len: 4096, prot: READ )
24 4.505 ( 0.011 ms): fileCopyTest/2779 mprotect(start: 0x7f31f3996000, len: 4096, prot: READ )
25 4.553 ( 0.054 ms): fileCopyTest/2779 munmap(addr: 0x7f31f3980000, len: 89874 )
26 4.667 ( 0.118 ms): fileCopyTest/2779 clone(clone_flags: CHILD_CLEARID|CHILD_SETTID|0x11, child_tidptr: 0
27 4.878 ( 0.013 ms): fileCopyTest/2779 open(filename: 0x850bf849 )
28 4.928 ( 0.102 ms): fileCopyTest/2779 open(filename: 0x850bf85a, flags: RDWR|CREAT|TRUNC, mode: IRUGO|ISGI
29 5.073 ( 0.003 ms): fileCopyTest/2779 fstat(fd: 3</home/userstest0710012/Documents/project 1B/originalFile.
30 5.118 ( 0.003 ms): fileCopyTest/2779 lseek(fd: 4</home/userstest0710012/Documents/project 1B/copiedFile.tx
31 5.212 ( 0.141 ms): fileCopyTest/2779 write(fd: 4</home/userstest0710012/Documents/project 1B/copiedFile.tx
32 5.397 ( 0.011 ms): fileCopyTest/2779 mmap(len: 9947, prot: READ, flags: SHARED, fd: 3 )
33 5.444 ( 0.005 ms): fileCopyTest/2779 mmap(len: 9947, prot: READ|WRITE, flags: SHARED, fd: 4 )
34 5.483 ( ): fileCopyTest/2779 exit_group( )
```

Screenshot 20:

-from fastest to slowest- :mmap, write, clone, open

Though open takes less time when executing the 5000 bytes file, the average time is longer than clone



Screenshot 21:

a.read the perf.data file and display the context

b.The percentage under the overhead column shows the overall samples collected in the corresponding function and its children, therefore, the fileCppyTest function and its children, which is dynamically linked and have the shared library named “mptbase” contain 100% of the overall samples, while the fileCopyTest function(the one that come from the kernel) and its children contains 0% of the overall samples.

Actually can’y really understand the chart, as it looks totally different with the one I find online, no matter it is the parents function display or the percentage. Moreover, I think my perf.data may have some problem as the percentage is shown in red font, which is not normal, however, no matter how many time I tried to rerun the program or “record” a new perf.data, the result is always the same.


```

userstest0710012@ubuntu: ~/Documents/project 1B
# To display the perf.data header info, please use --header/--header-only opti
#
#
# Total Lost Samples: 0
#
# Samples: 1  of event 'cpu-clock'
# Event count (approx.): 250000
#
# Children      Self  Command      Shared Object      Symbol
# .....      .....  .....
#
# 100.00%    100.00%  fileCopyTest  [mptbase]          [k] mpt_put_msg_frame
#          |
#          ---entry_SYSCALL_64_after_hwframe
#          do_syscall_64
#          0xffffffff8109a6d8
#          do_group_exit
#          do_exit
#          task_work_run
#          ____fput
#          __fput
#          ext4_release_file
#          ext4_alloc_da_blocks
#          filemap_flush
#          __filemap_fdatawrite_range
#          do_writepages
#          ext4_writepages
#          blk_finish_plug
#          blk_flush_plug_list
#          queue_unplugged
#          __blk_run_queue
#          scsi_request_fn
#          scsi_dispatch_cmd
#          mptspi_qcmd
#          mptscsih_qcmd
#          mpt_put_msg_frame
#
# 100.00%    0.00%  fileCopyTest  [kernel.kallsyms]  [k] entry_SYSCALL_64_af
#          |
#          ---entry_SYSCALL_64_after_hwframe
#          do_syscall_64
#          0xffffffff8109a6d8
#          do_group_exit
#          do_exit
#          task_work_run
#          ____fput
#          __fput
#          ext4_release_file
#          ext4_alloc_da_blocks
#          filemap_flush
#          __filemap_fdatawrite_range
#          do_writepages
#          ext4_writepages
userstest0710012@ubuntu:~/Documents/project 1B$

```

The example I found online:

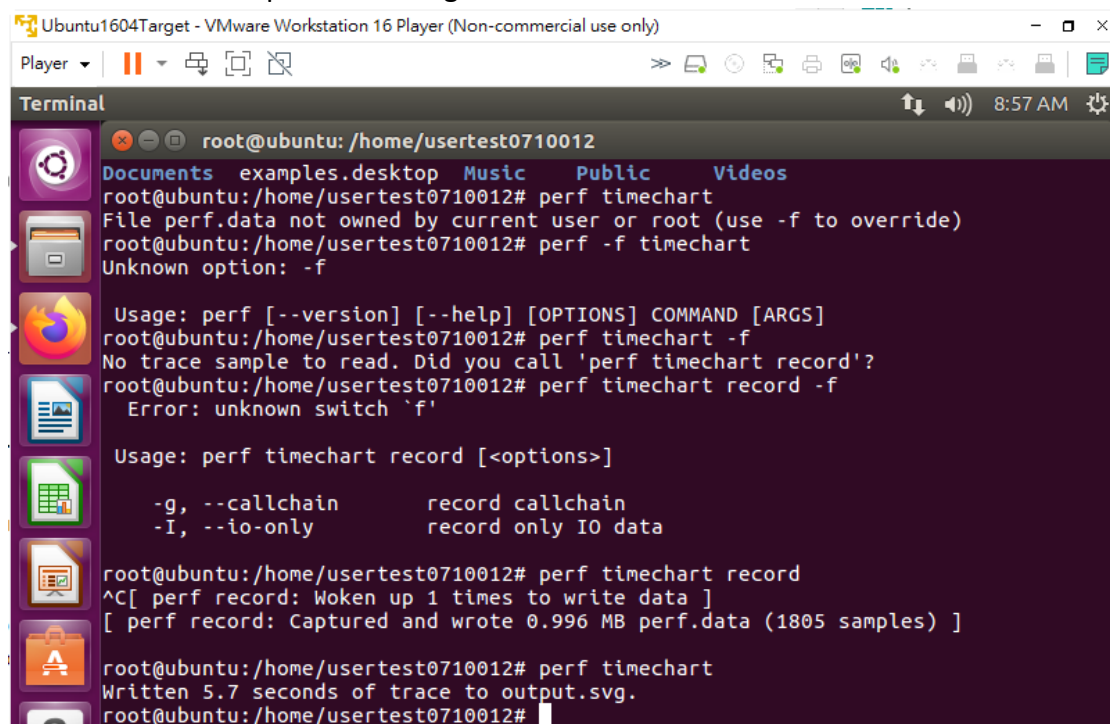
```
Children      Self  Symbol
-----
100.00%      0.00%  __libc_start_main
|
--- __libc_start_main
100.00%      0.00%  main
|
--- main
    __libc_start_main
100.00%     40.00%  bar
|
--- bar
    main
    __libc_start_main
60.00%     60.00%  foo
|
--- foo
    bar
    main
    __libc_start_main
```

Screenshot 22:

I choose “perf timechart” as my example, the function is simply record(perf.data) the time table (system behavior) of the program and print

(Tool to visualize total system behavior during a workload)

The function will output a file called output.svg(graphic file), which is shown in below, we can see the execution of the function in the graph, showing the relative time and feature of parallel among the functions.



```
Ubuntu1604Target - VMware Workstation 16 Player (Non-commercial use only)
Player
Terminal
root@ubuntu: /home/usertest0710012
Documents examples.desktop Music Public Videos
root@ubuntu: /home/usertest0710012# perf timechart
File perf.data not owned by current user or root (use -f to override)
root@ubuntu: /home/usertest0710012# perf -f timechart
Unknown option: -f

Usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]
root@ubuntu: /home/usertest0710012# perf timechart -f
No trace sample to read. Did you call 'perf timechart record'?
root@ubuntu: /home/usertest0710012# perf timechart record -f
Error: unknown switch `f'

Usage: perf timechart record [<options>]

-g, --callchain      record callchain
-I, --io-only        record only IO data

root@ubuntu: /home/usertest0710012# perf timechart record
^C[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.996 MB perf.data (1805 samples) ]

root@ubuntu: /home/usertest0710012# perf timechart
Written 5.7 seconds of trace to output.svg.
root@ubuntu: /home/usertest0710012#
```

