Operating Systems Project Report

Project Number (01 / 02 / 03):	01
Name:	陳宥安
Student ID:	109550073
YouTube link (Format youtube.com/watch?v=[key]):	https://youtu.be/mb4Y6aeS1bs
Date (YYYY-MM-DD):	2021/10/28
Names of the files uploaded to E3:	OS_Project01_109550073.pdf
Physical Machine Total RAM (Example: 8.0 GB):	16.0GB
Physical Machine CPU (Example: Intel i7-2600K):	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz

Checklist	
Yes/No	**Item**
Y	**The report name follows the format “OS_ProjectXX_StudentID.pdf”.**
	The report was uploaded to E3 before the deadline.
Y	**The YouTube video is public, and anyone with the link can watch it.**
Y	**The audio of the video has a good volume.**
Y	**The pictures in your report and video have a good quality.**
Y	**All the questions and exercises were answered inside the report.**
Y	**I understand that late submission is late submission, regardless of the time uploaded.**
Y	**I understand that any cheating in my report / video / code will not be tolerated.**

1. Questions to be answered in the report

1.1. What is a Kernel? What are the differences between mainline, stable and longterm? What is a Kernel panic?

kernel is a computer program at the core of a computer's operating system and has complete control over everything in the system.

Mainline, stable and longterm are different categories of kernel released. Mainline is the version with the latest features and development. After each mainline kernel is released, it is regarded as stable version. Nevertheless, there are several longterm maintenance kernel for supporting bugfixes for older kernel trees.

Kernel panic is a safety measure taken by an operating system's kernel upon detecting an internal fatal error in which either it is unable to safely recover or continuing to run the system would have a higher risk of major data loss.

1.2. What are the differences between building, debugging and profiling?

Building means starting with source files produced by developers and ending with things like installation packages that are ready for deployment. **Debugging is the process of looking for bugs and their cause in applications. Profiling is the process of measuring an application or system by running an analysis tool called a profiler.**

1.3. What are GCC, GDB, and KGDB, and what they are used for?

GCC is an compiler produced by the GNU Project supporting various programming languages, hardware architectures and operating systems.

GDB is a debugger that runs on many Unix-like systems and works for many programming languages.

KGDB is a debugger for the Linux kernel and the kernels of NetBSD and FreeBSD

1.4. What are the /usr/, /boot/, /home/, /boot/grub folders for?

/Usr/ is the one of the most important directory that usually contains the largest share of data on system.

/boot/ stores some kernel files for starting the computer.

/home/ is a directory for a particular user of system.

/boot/grub is a boot loader package from GNU project

1.5. What are the general steps to debug a Linux Kernel? (Add a figure)

We use KGDB , a Linux kernel debugger, as a example:

1.6. For this project, why do we need two virtual machines?

Because we have kernel debugging part in our project, which requires a target machine, Ubuntu Server, runs the kernel we built and a host machine, Ubuntu Desktop, for debugging our kernel.

1.7. In Section 3.2, what are the differences between make, make modules_install and make install?

Make compiles and links the kernel image.

Make modules_install installs the kernel modules to /lib/modules/<version>

Make install installs the built kernel to /vmlinuz

1.8. In Section 3.3, what are the commands kgdbwait and kgdboc=ttyS1,115200 for?

Kgdbwait for causing execution to pause and be passed to the kdb debugger during bootup.

Kgdboc for declaring serial service connected to the host and 115200 for the baud rate used.

1.9. What is grub? What is grub.cfg?

Grub is a bootloader package which provides a user the choice to boot one of multiple OS installed on the computer or select a specific kernel configuration.

Grub.cfg is the grub configuration file.

1.10. List at least 10 commands you can use with GDB.

b main, info break, r, c ,f, s, n, d, u, q

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

Kernel function is the function inside kernel, while system call is the request for the kernel to access a resource.

1.12. What is KASLR? What is it for?

KASLR (kernel address space layout randomization) is a memory-protection process for OS that guards against buffer-overflow attacks by randomizing the location where system executables are loaded into memory.

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

For non-stop mode, execution commands only apply to the current thread by default.

For all-stop mode, all threads of execution stop whenever your program stops under GDB under any reason.

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

G means connect to kgdb, with sysrq allowing us to send specific instructions directly to the kernel.

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

clone: system call used to create a new thread of execution.

mmap: system call for mapping files or devices into memory.

write: basic routines for writing data from a buffer to a given device.

open: system call that opens the file with the specified pathname

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

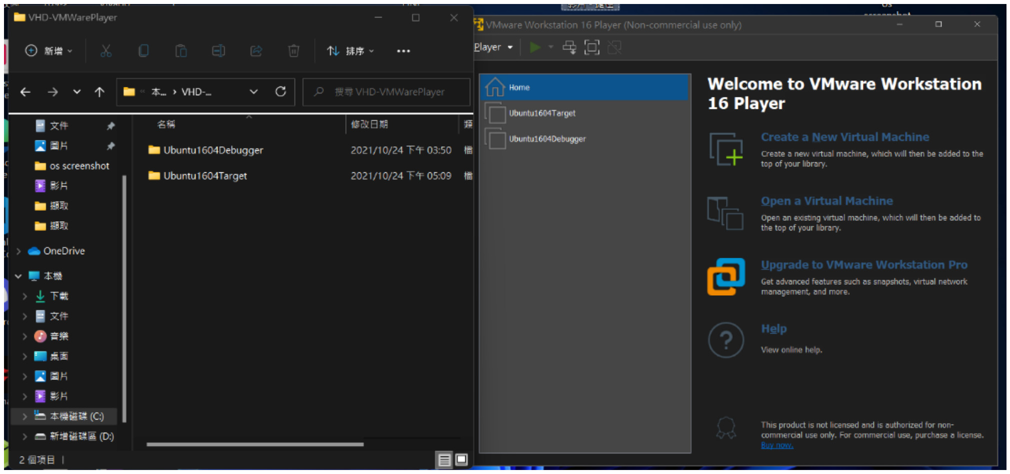
Because a 'fork' is the most natural way for one process to transform itself into another process while continuing to run by itself.

Clone, as fork, creates a new process. Unlike fork, these calls allow the child process to share parts of its execution context with the calling process, such as the memory space, the table of file descriptors, and the table of signal handlers.

------------------The rest questions in section 5 will be answered in the last few screenshots---------

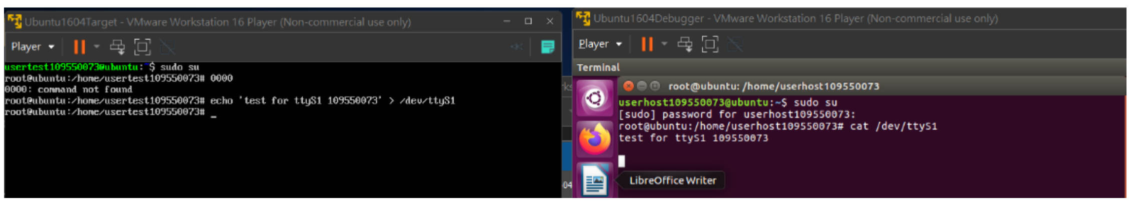
2. Screenshots and explanation

2.1. Screenshot 1



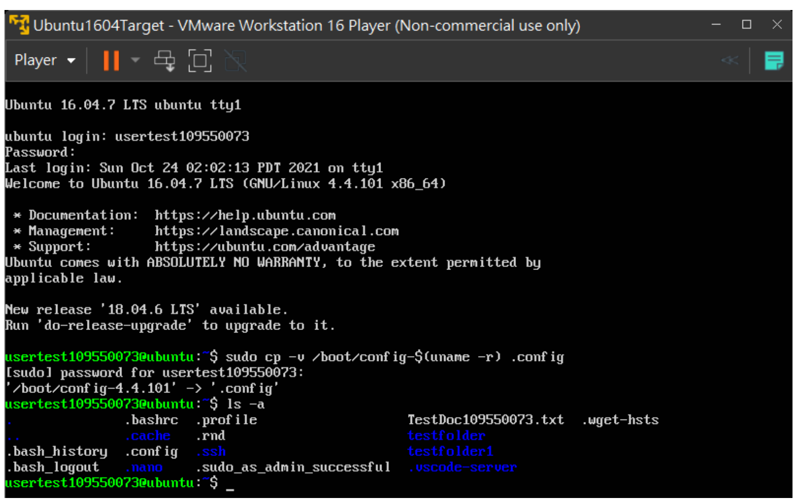
The screenshot shows both virtual machine in VMWare.

2.2. Screenshot 2



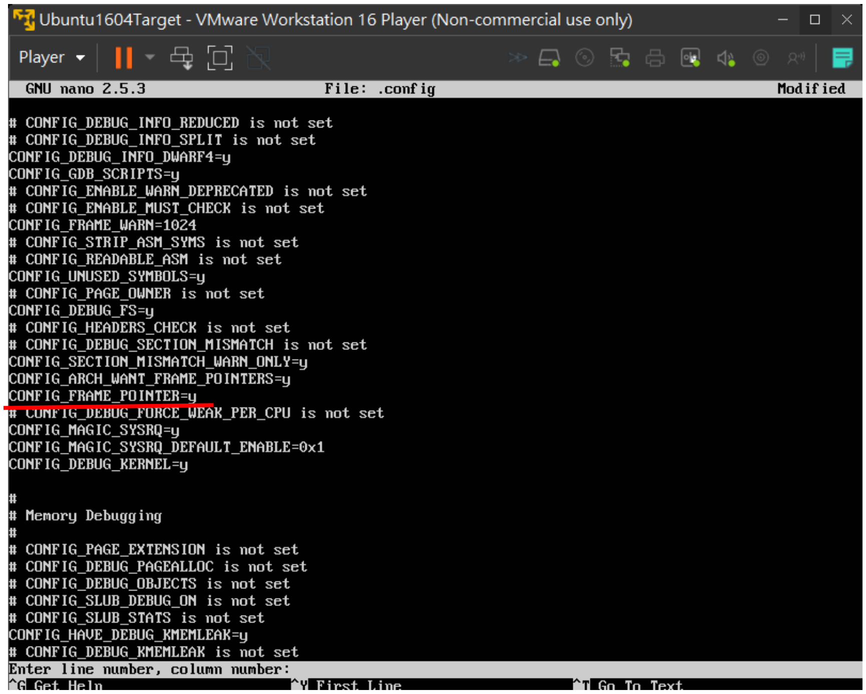
The screenshot implies that the Target machine and the Host machine communicate through serial port (ttyS1).

2.3. Screenshot 3



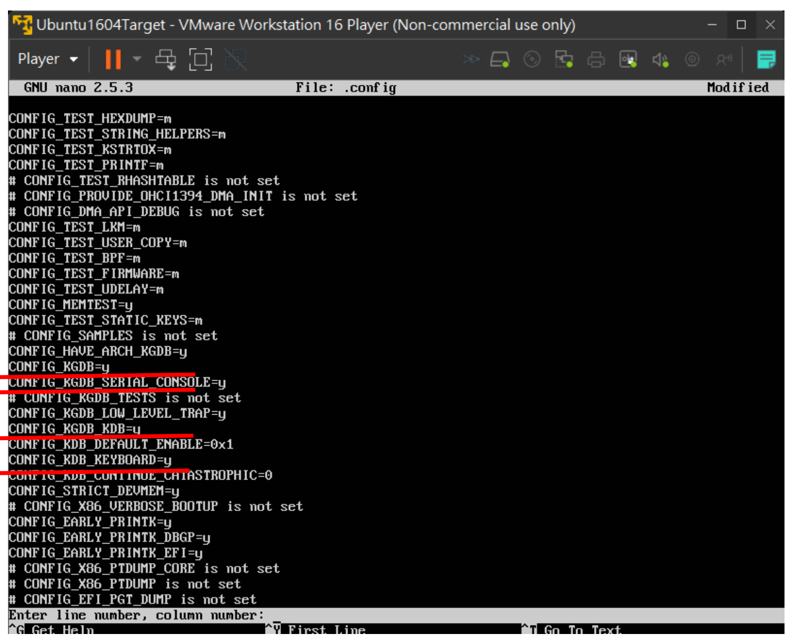
The screenshot tells that the /boot/config-4.4.101 is copied to .config with cp command.

2.4. Screenshot 4



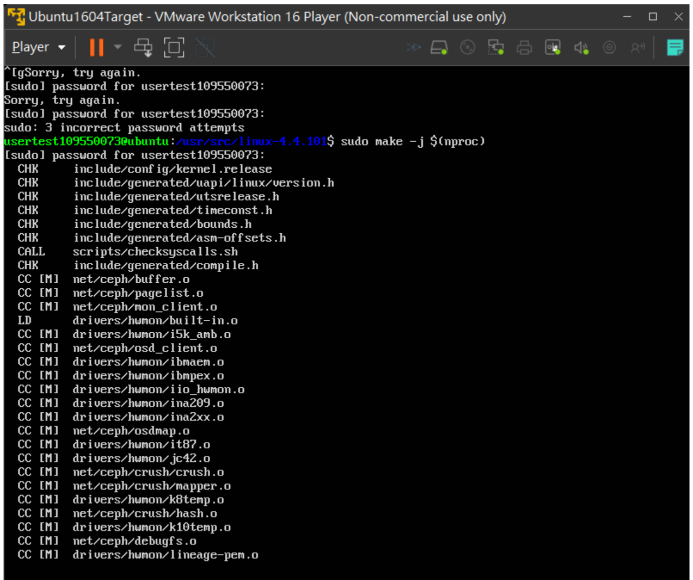
The screenshot shows the changes in .config file, the change I made here is CONFIG_FRAME_POINTER.

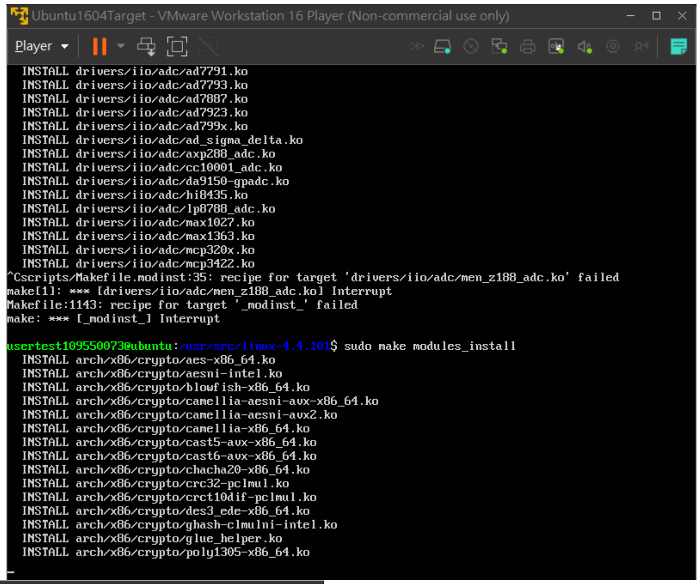
2.5. Screenshot 5

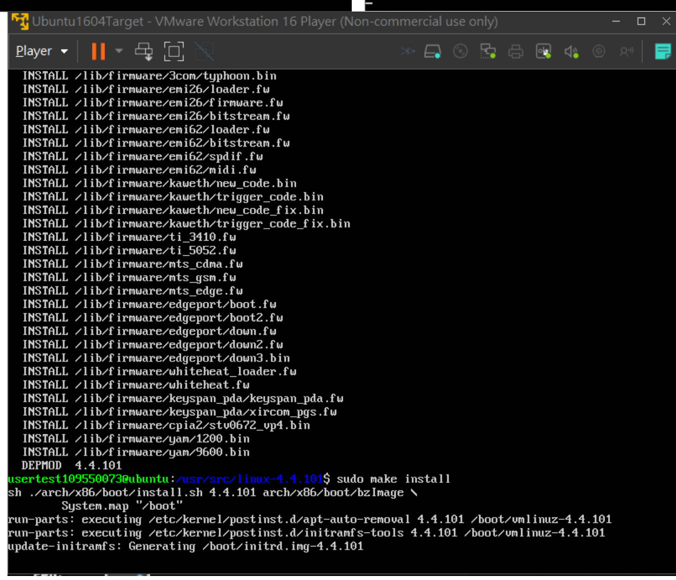


The screenshot shows the changes in .config file, the changes I made here are CONFIG_KGDB, CONFIG_KGDB_SERIAL_CONSOLE, CONFIG_KGDB_KDB, CONFIG_KDB_KEYBOARD.

2.6. Screenshot 6 & 7 & 8



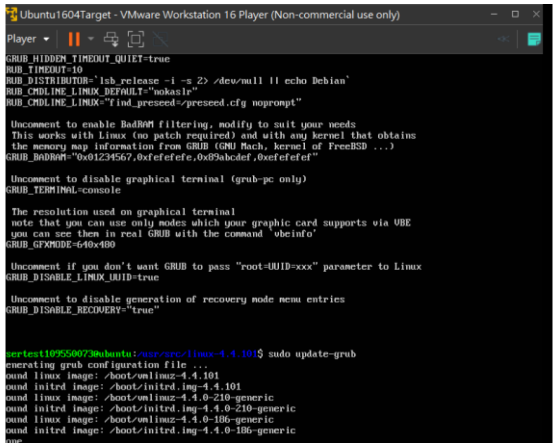


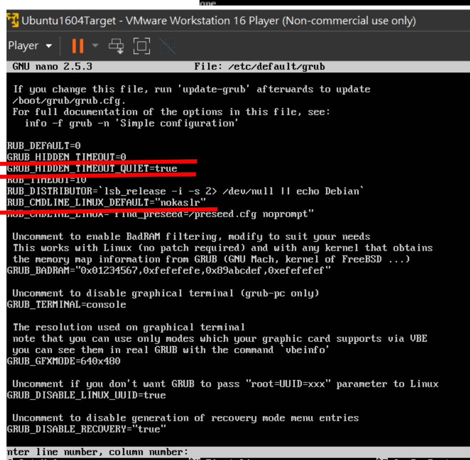


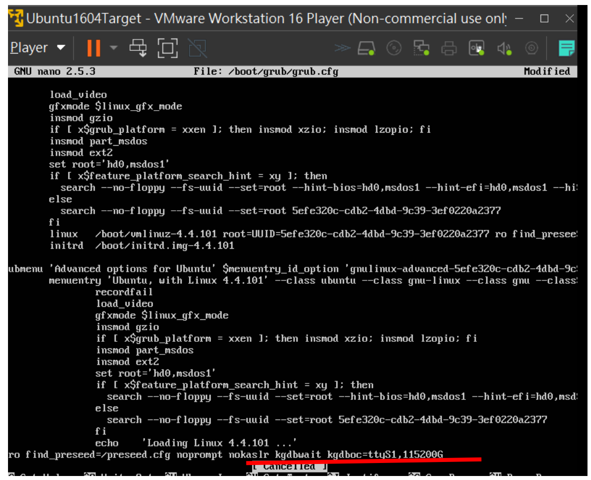
The screenshots we made here represent the execution of three following commands, which are “sudo make -j $(nproc)”, “sudo make modules_install”, “sudo make install”.

The difference between these three commands is answered in Section 1.7

2.7. Screenshot 9 & 10-1 & 10-2

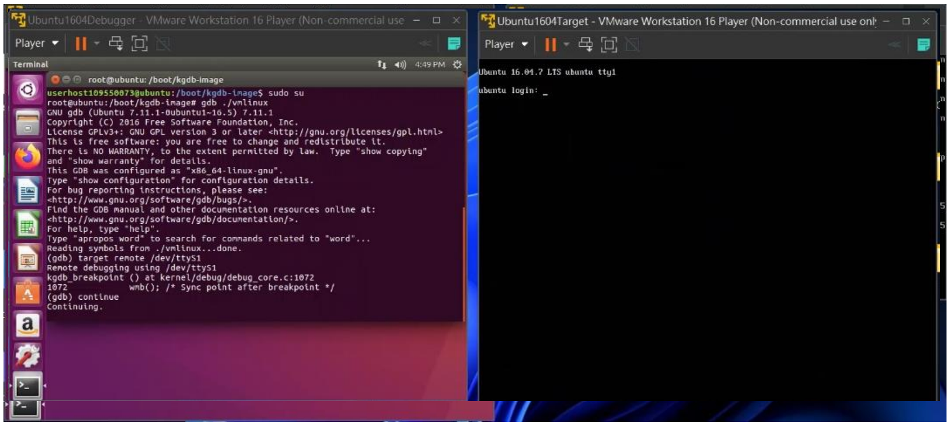






The screenshots represent the grub update in my VM. In screenshot 9, we first run update-grub command for generating configuration files, while we change some parameters in grub's configuration file, which are GRUB_HIDDEN_TIMEOUT, GRUB_HIDDEN_TIMEOUT_QUIET, GRUB_CMDLINE_LINUX_DEFAULT in screenshot 10-1 and kgdbwait kgdboc=ttyS1, 115200 in screenshot 10-2.

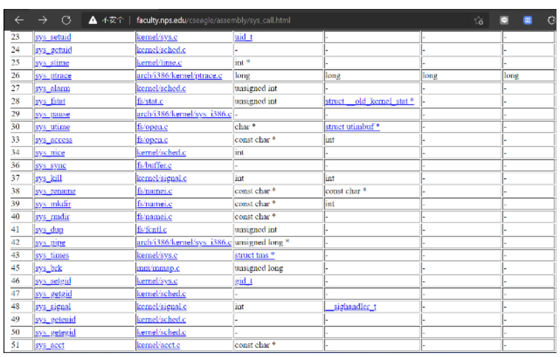
2.8. Screenshot 11 & 12

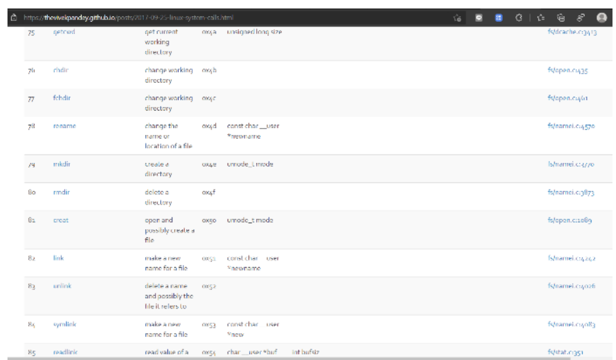


In Screenshot 11, the Target machine was freeze while it is booting with the Host machine open GDB for debugging the Target machine's kernel using serial port.

In Screenshot 12, with the continue command send by GDB in the Host machine, the Target machine start the rest booting procedure and start successfully.

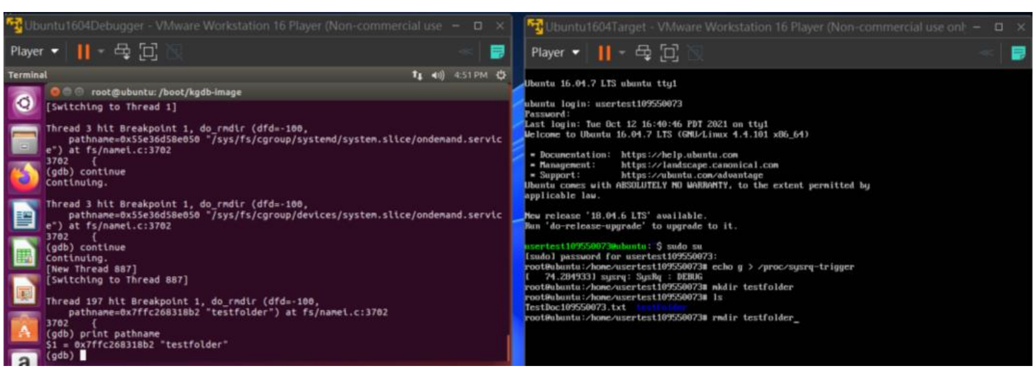
2.9. Screenshot 13-1 & 13-2





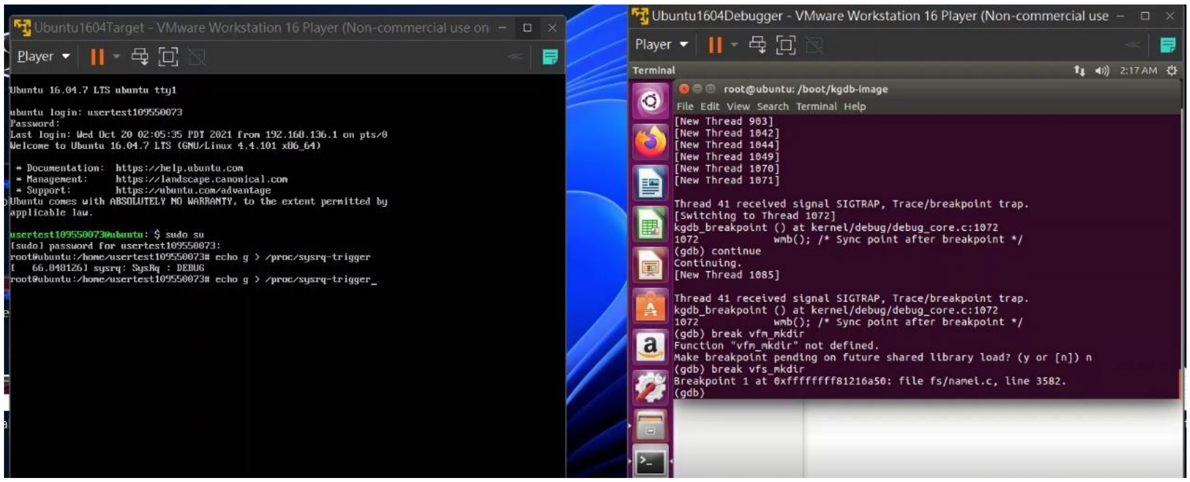
These two screenshots show the Linux system call table I found on the Internet in different websites.

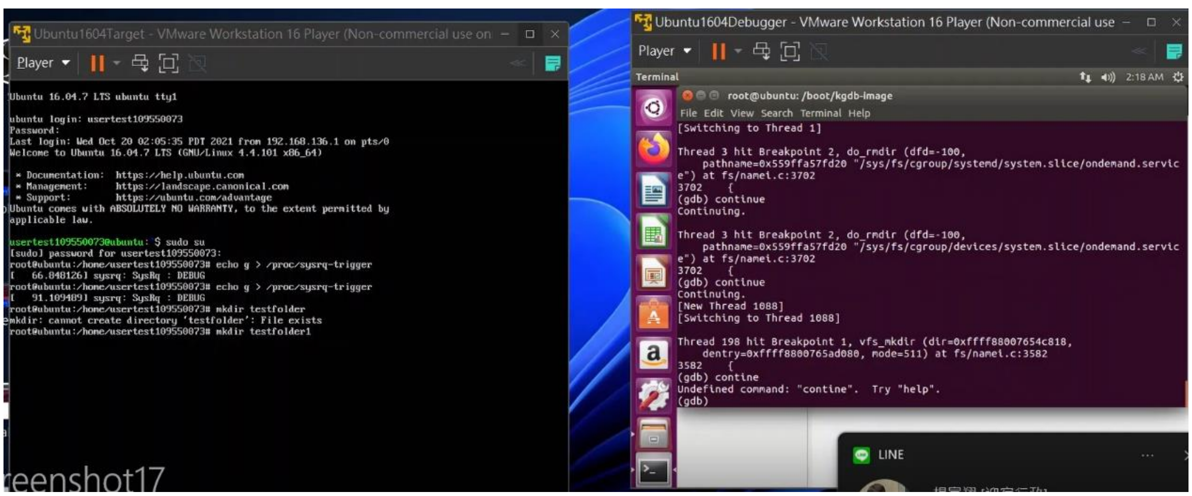
2.10. Screenshot 14



The screenshot implies the procedure that we put the breakpoint at function do_mkdir. After the breakpoint is set, the Target machine do syscall mkdir then rmdir, which trigger the GDB and freeze the Target machine.

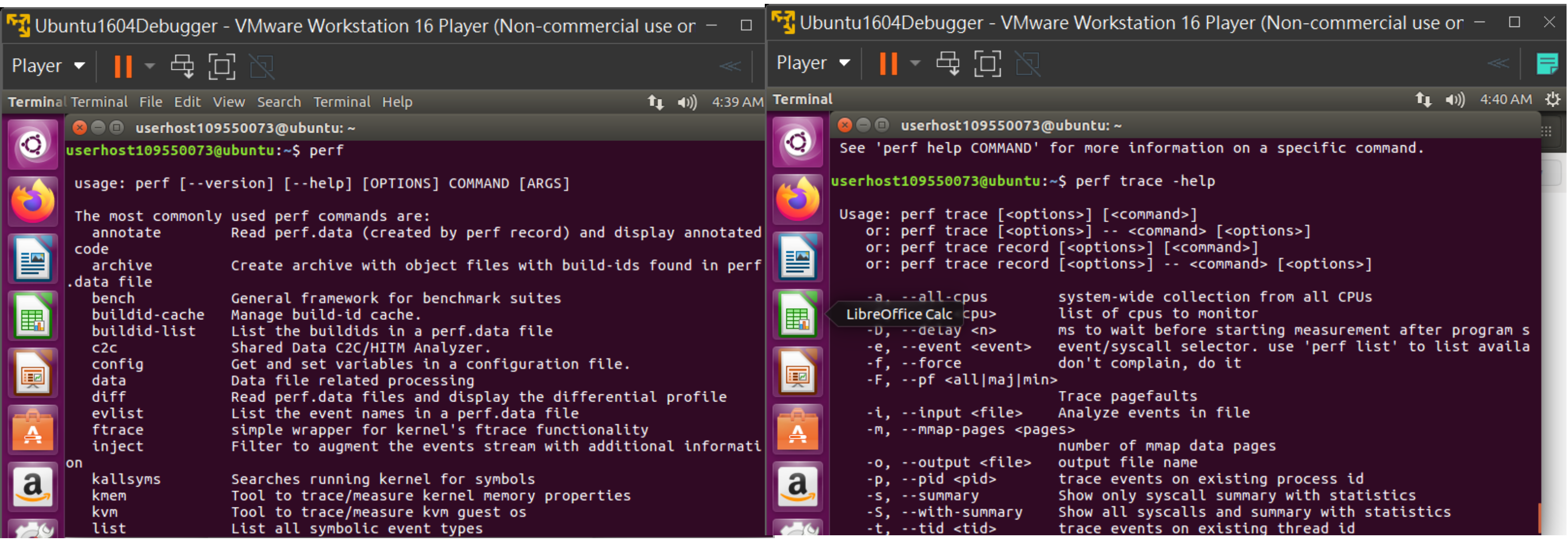
2.11. Screenshot 15 & 16 & 17





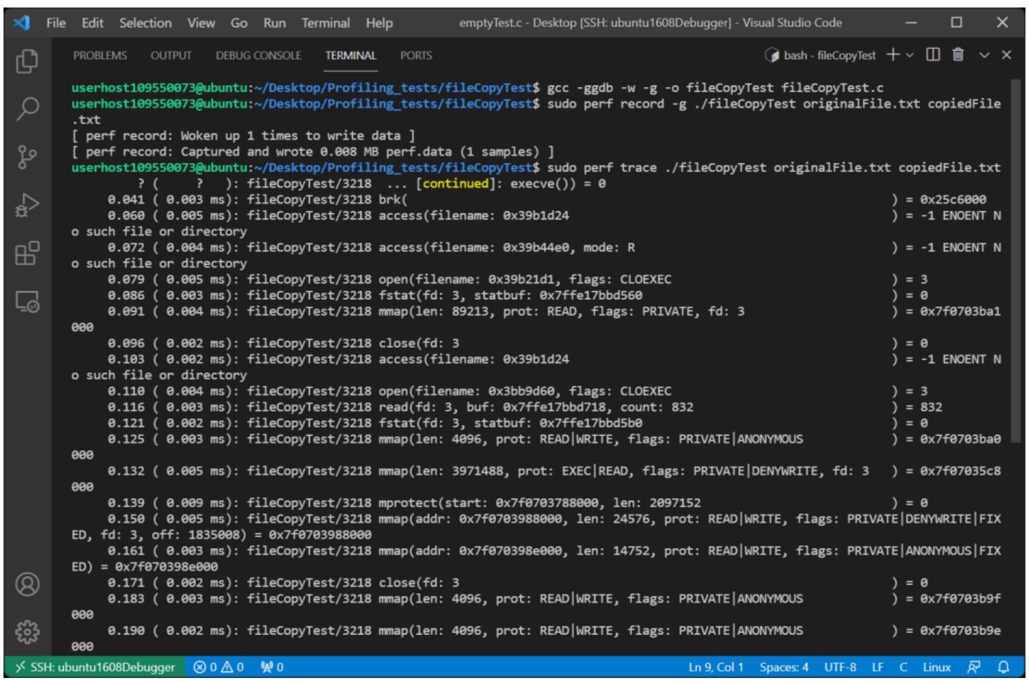
The screenshots show that I set a breakpoint at vfs_mkdir, which is the function I found in namei.c, and the debugger continue. The debugger is triggered again while I make a new directory—testfolder1.

2.12. Screenshot 18



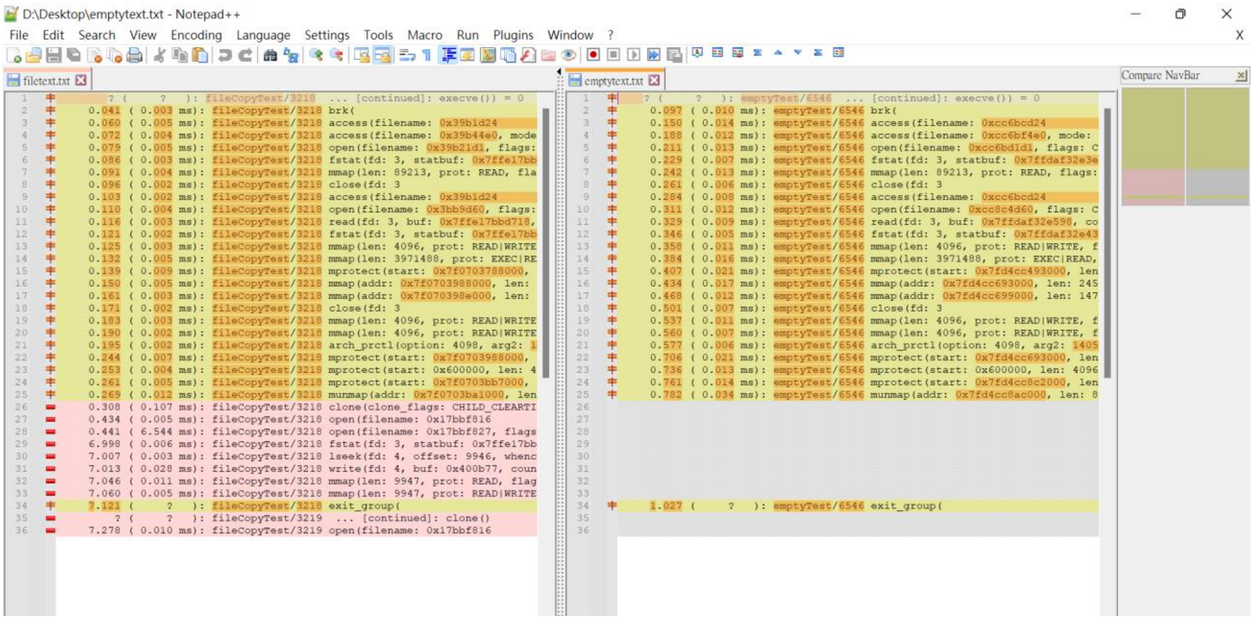
The screenshot shows the function of perf and perf trace.

2.13. Screenshot 19



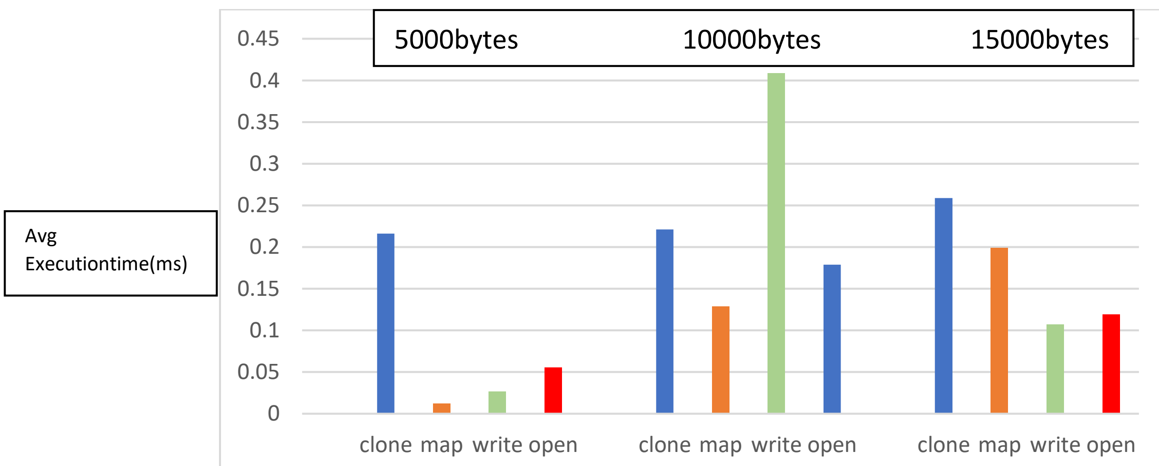
The screenshot shows the performance of the three commands we ran.

2.14. Screenshot 20



The red spot implies the differences.

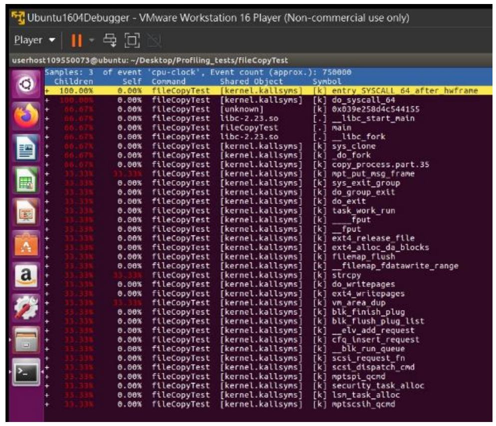
2.15. Screenshot 21



a) No, we can see that in the graph. It is not absolute.

b) mmap < open < write < clone

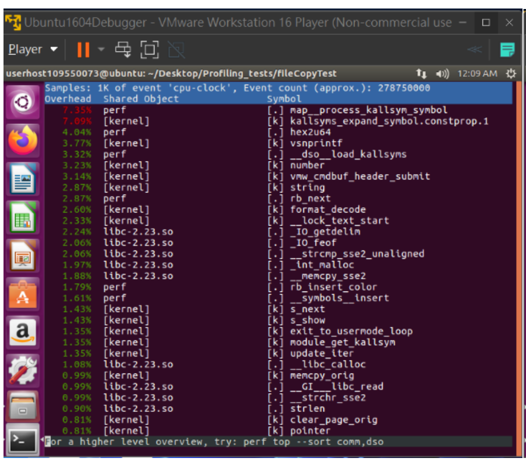
2.16. Screenshot 22

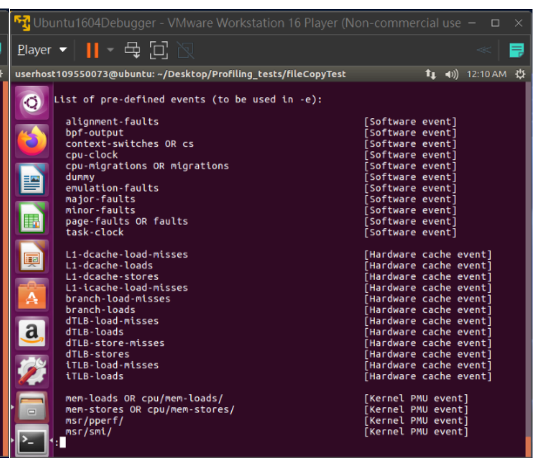


i) It collects perf data for analyzing and optimization.

ii) We can see that mpt_put_mgs_frame, strcpy, vm_area_dup's self is 33.3% while the others are 0% .

2.17. Screenshot 23-1 & 23-2





For screenshot 23-1, the perf top can detect the occupation rate of functions in real time.

For screenshot 23-2, the perf list can list the supported perf events.