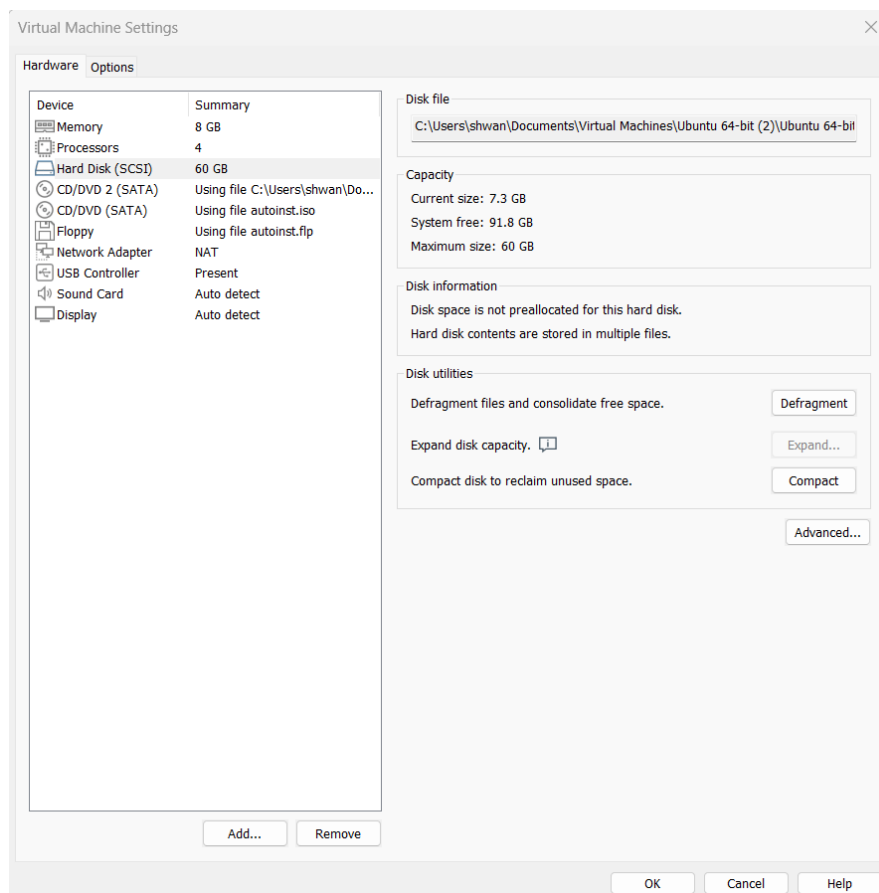
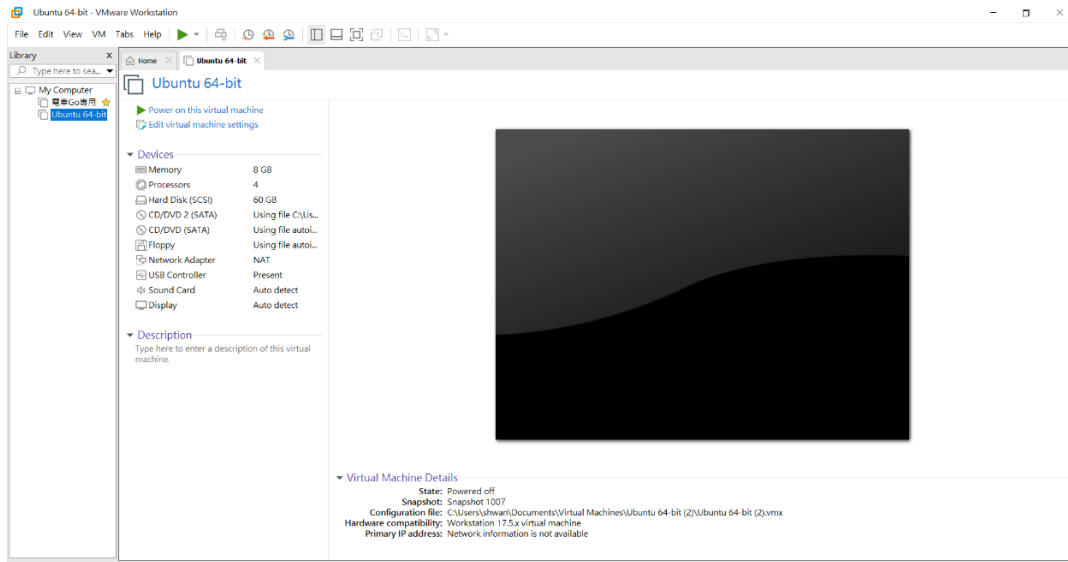


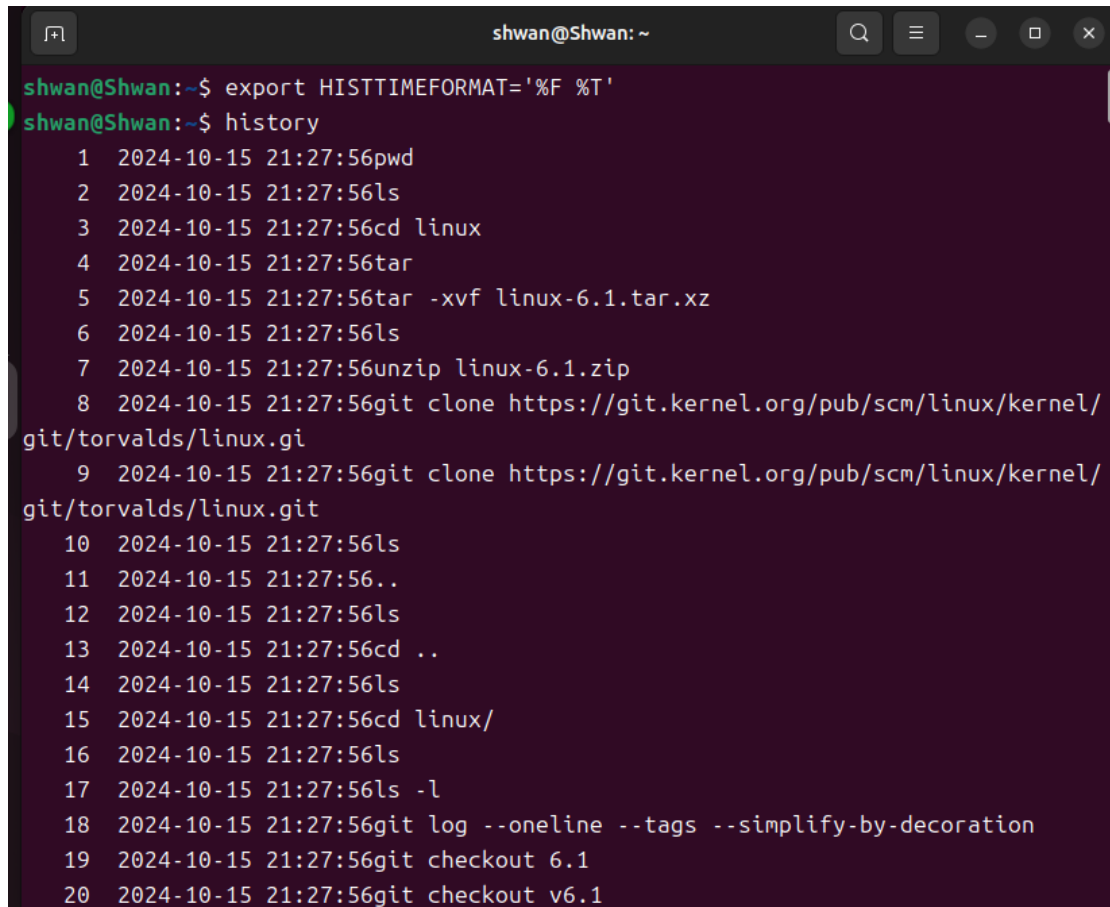
# Installing a Linux Distribution and Compiling the Linux Kernel

1. 個人使用 VMware Workstation Pro 來建立一台 VM，設定 Storage 60GB，Memory 8GB，以及 4 個 core 作為硬體規格來安裝 Ubuntu 24.02。



## 2. Download the Linux kernel source and change local version

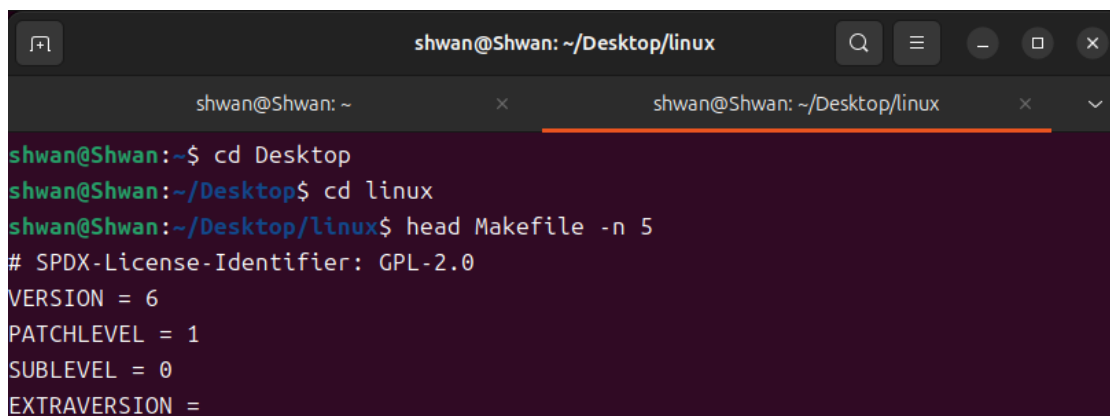
```
$ sudo apt update
$ sudo apt install git
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```



```
shwan@Shwan:~$ export HISTTIMEFORMAT='%F %T'
shwan@Shwan:~$ history
 1  2024-10-15 21:27:56pwd
 2  2024-10-15 21:27:56ls
 3  2024-10-15 21:27:56cd linux
 4  2024-10-15 21:27:56tar
 5  2024-10-15 21:27:56tar -xvf linux-6.1.tar.xz
 6  2024-10-15 21:27:56ls
 7  2024-10-15 21:27:56unzip linux-6.1.zip
 8  2024-10-15 21:27:56git clone https://git.kernel.org/pub/scm/linux/kernel/
git/torvalds/linux.gi
 9  2024-10-15 21:27:56git clone https://git.kernel.org/pub/scm/linux/kernel/
git/torvalds/linux.git
10  2024-10-15 21:27:56ls
11  2024-10-15 21:27:56..
12  2024-10-15 21:27:56ls
13  2024-10-15 21:27:56cd ..
14  2024-10-15 21:27:56ls
15  2024-10-15 21:27:56cd linux/
16  2024-10-15 21:27:56ls
17  2024-10-15 21:27:56ls -l
18  2024-10-15 21:27:56git log --oneline --tags --simplify-by-decoration
19  2024-10-15 21:27:56git checkout 6.1
20  2024-10-15 21:27:56git checkout v6.1
```

如上圖，我下載了整個 linux kernel 安裝包，並確定有存在 v6.1，執行

\$ head Makefile -n 5 指令得到以下結果，確認為 6.1.0 kernel



```
shwan@Shwan:~$ cd Desktop
shwan@Shwan:~/Desktop$ cd linux
shwan@Shwan:~/Desktop/linux$ head Makefile -n 5
# SPDX-License-Identifier: GPL-2.0
VERSION = 6
PATCHLEVEL = 1
SUBLEVEL = 0
EXTRAVERSION =
```

接下來進行安裝：

(1)sudo apt install libncurses-dev gawk flex bison openssl libssl-dev dkms

libelf-dev libudev-dev libpci-dev libiberty-dev autoconf llvm

安裝所需套件

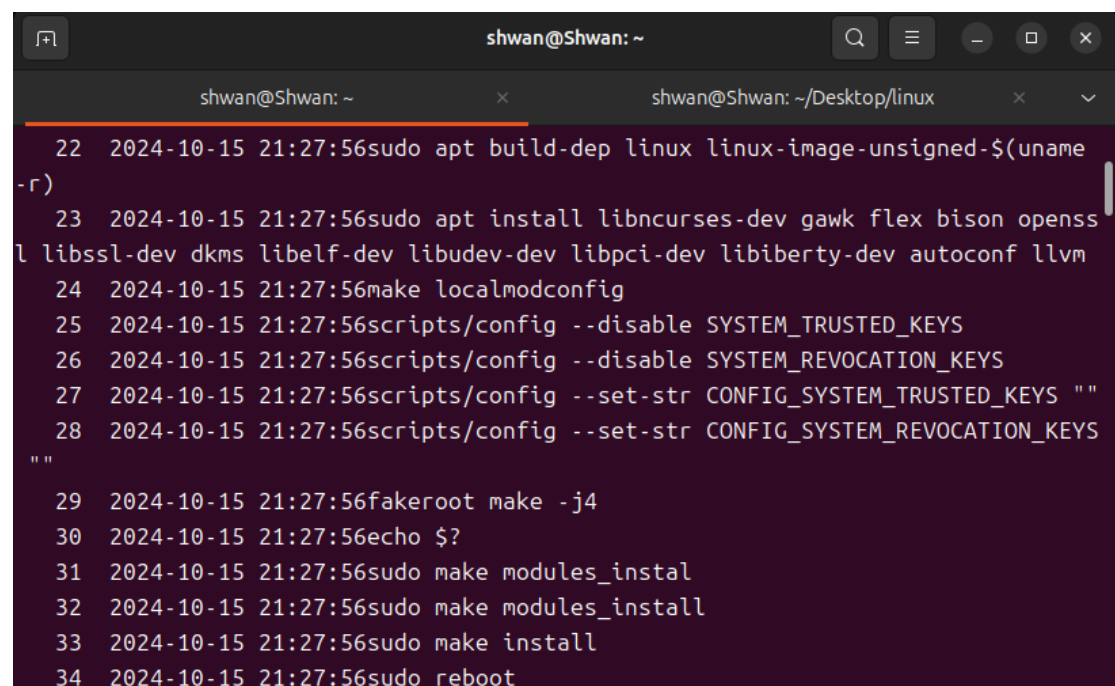
(2)生成 local config file(此處先不動任何設定)用於等等的 kernel compile

(3)將 system\_trusted\_keys 等等安全設置先解除，以便待會開始編譯 kernel

(4)make -j4 進行 kernel compile，make\_modules\_install 和 make install 分別

代表安裝 kernel modules，以及安裝整個 kernel image file，先執行前者再執

行後者確保 modules 和 kernel 都正確被安裝。



```
shwan@Shwan: ~  
22 2024-10-15 21:27:56sudo apt build-dep linux linux-image-unsigned-$(uname  
-r)  
23 2024-10-15 21:27:56sudo apt install libncurses-dev gawk flex bison openss  
l libssl-dev dkms libelf-dev libudev-dev libpci-dev libiberty-dev autoconf llvm  
24 2024-10-15 21:27:56make localmodconfig  
25 2024-10-15 21:27:56scripts/config --disable SYSTEM_TRUSTED_KEYS  
26 2024-10-15 21:27:56scripts/config --disable SYSTEM_REVOCATION_KEYS  
27 2024-10-15 21:27:56scripts/config --set-str CONFIG_SYSTEM_TRUSTED_KEYS ""  
28 2024-10-15 21:27:56scripts/config --set-str CONFIG_SYSTEM_REVOCATION_KEYS  
""  
29 2024-10-15 21:27:56fakeroot make -j4  
30 2024-10-15 21:27:56echo $?  
31 2024-10-15 21:27:56sudo make modules_instal  
32 2024-10-15 21:27:56sudo make modules_install  
33 2024-10-15 21:27:56sudo make install  
34 2024-10-15 21:27:56sudo reboot
```

make localmodconfig

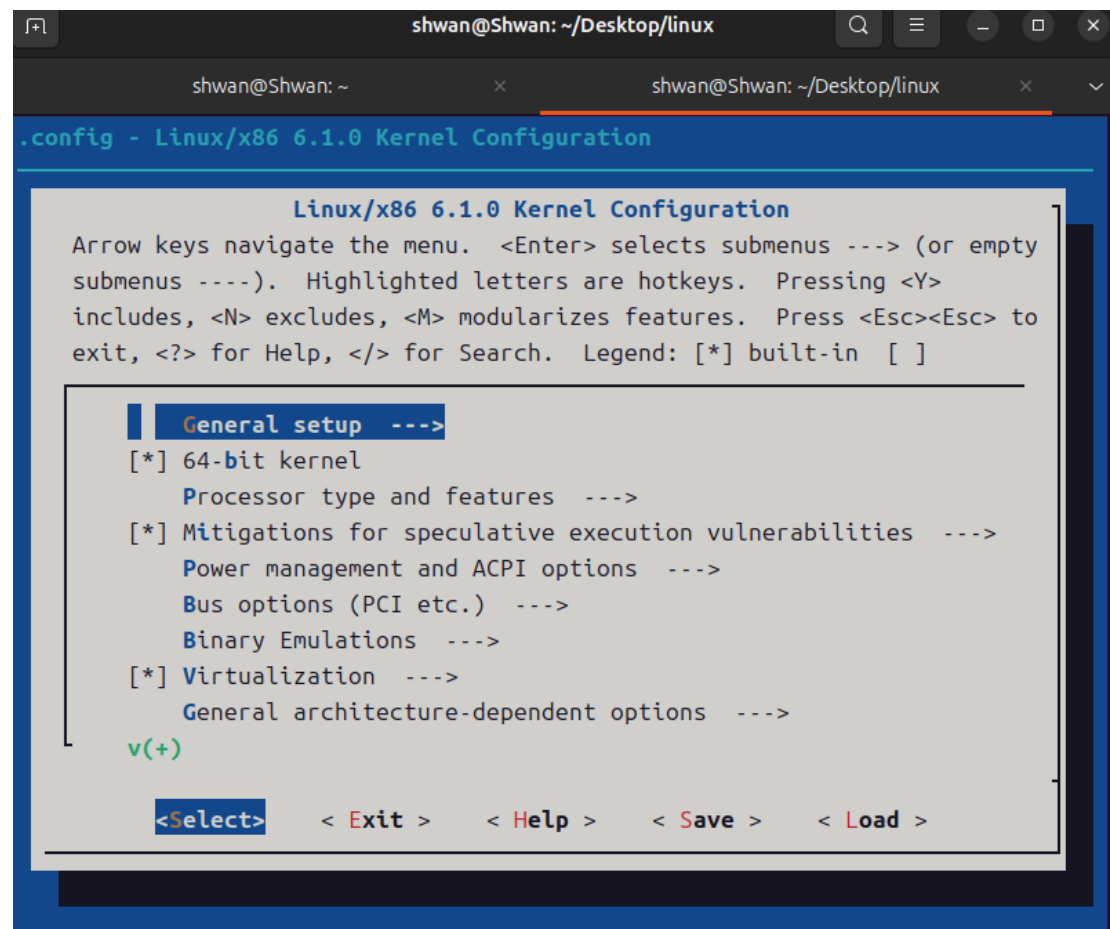
```
$ scripts/config --disable SYSTEM_TRUSTED_KEYS
$ scripts/config --disable SYSTEM_REVOCATION_KEYS
$ scripts/config --set-str CONFIG_SYSTEM_TRUSTED_KEYS ""
$ scripts/config --set-str CONFIG_SYSTEM_REVOCATION_KEYS ""
```

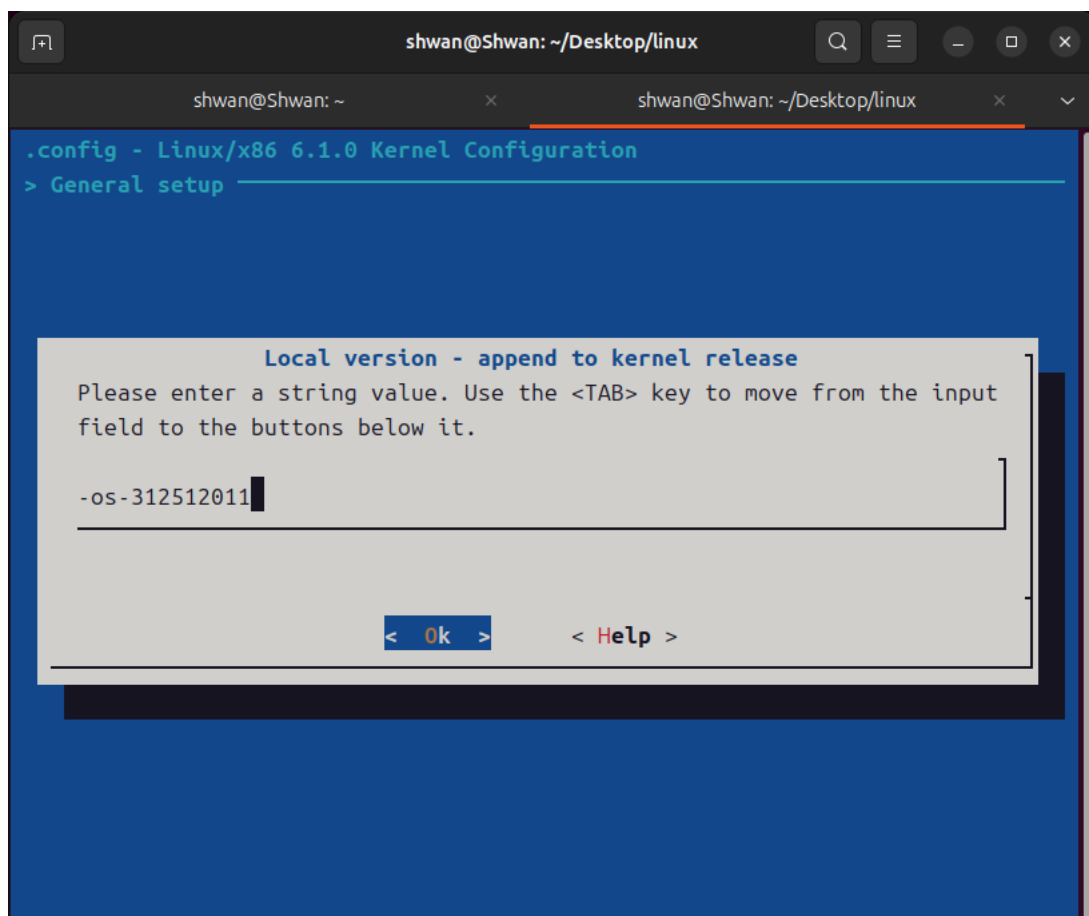
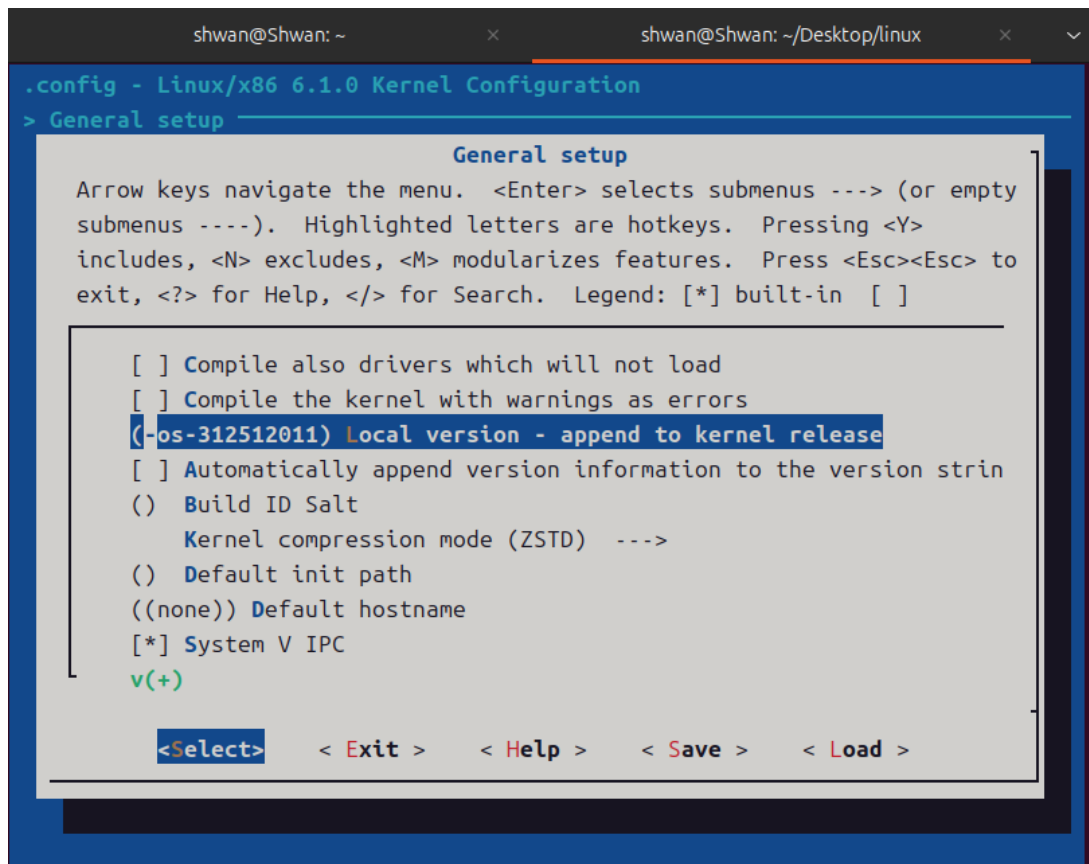
(上面兩個資料來源：<https://davidaugustat.com/linux/how-to-compile-linux-kernel-on-ubuntu>)

Change kernel local version:

General setup → Local Version – append to kernel release →

Enter string " -os- 312512011"



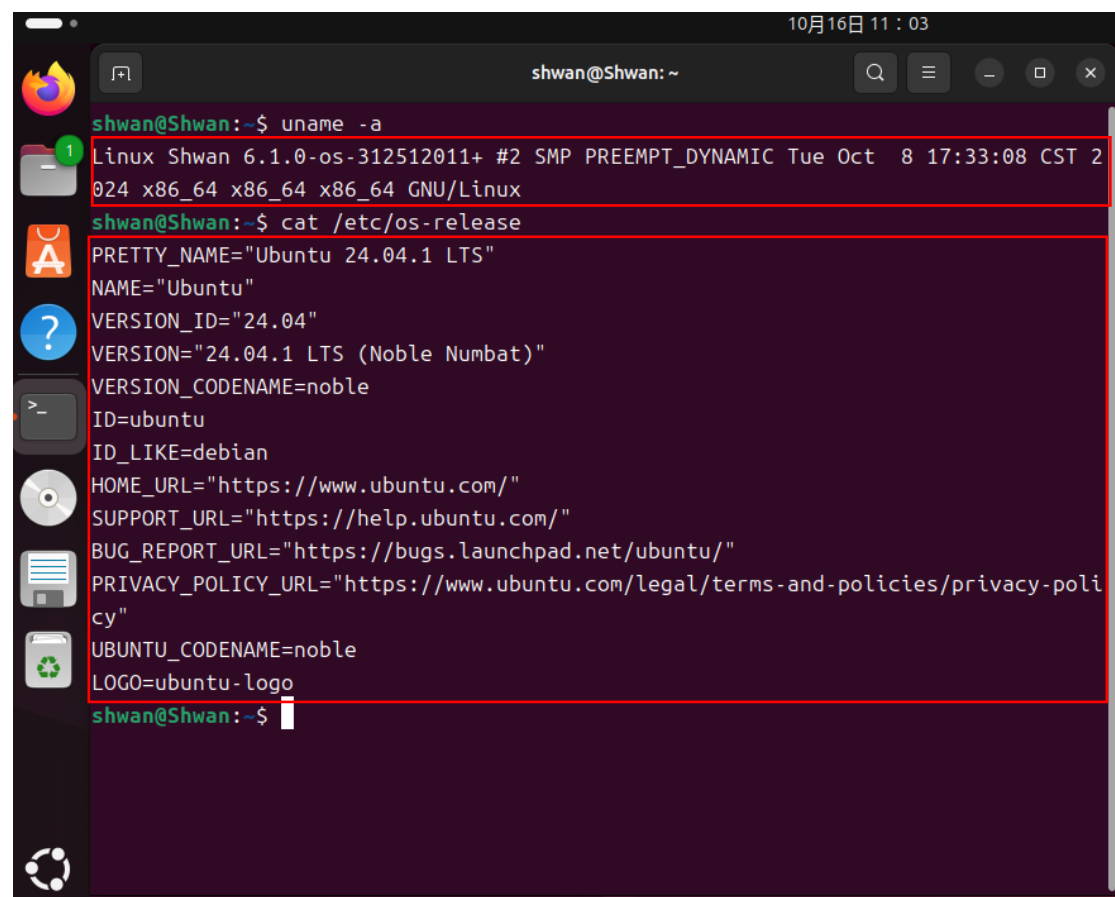


設定完成後 OK→Exit→Save

之後 `make -j4`、`make_modules_install` 和 `make install` 來安裝修改過 local

version 的 kernel(此三個指令都需要 sudo 權限來執行，-j4 表示以 4 個 core 運來編譯和運行目前的 Ubuntu 24.04)

- 執行 `uname -a` and `cat /etc/os-release` commands 的結果如下：(如下方兩個紅色框框內表示)



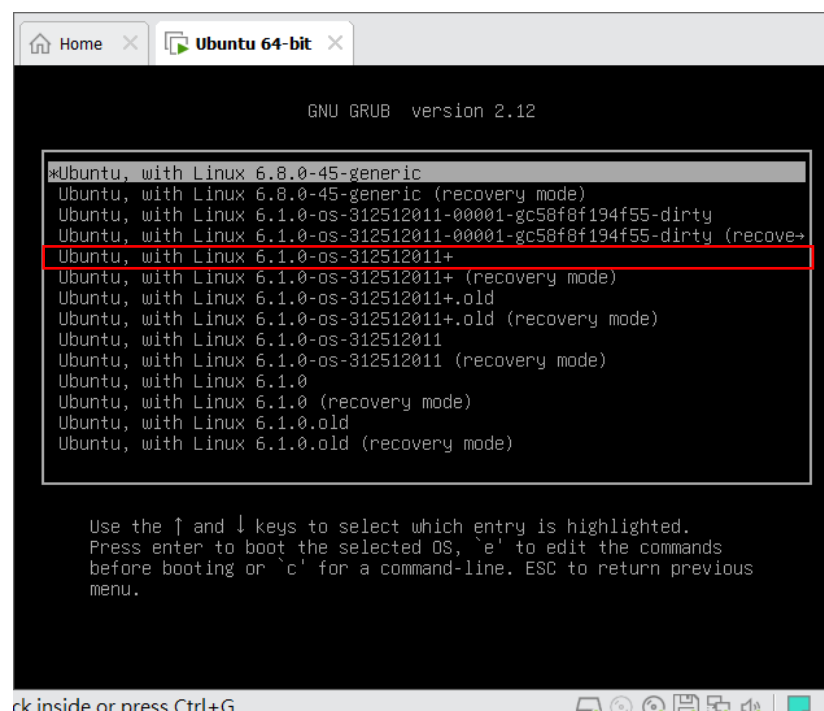
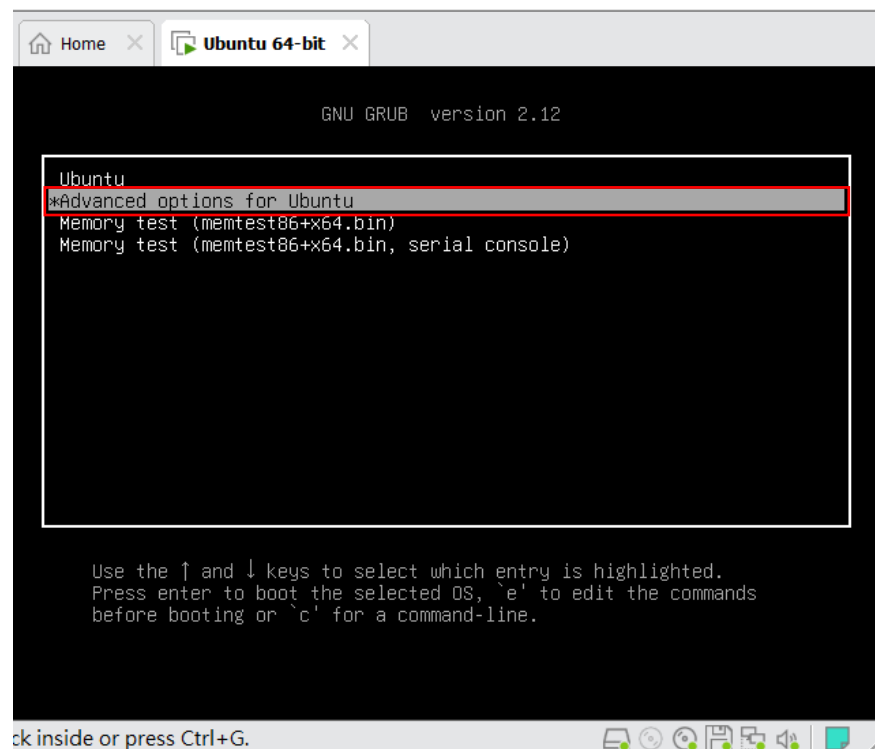
```
shwan@Shwan:~$ uname -a
Linux Shwan 6.1.0-os-312512011+ #2 SMP PREEMPT_DYNAMIC Tue Oct  8 17:33:08 CST 2024 x86_64 x86_64 x86_64 GNU/Linux
shwan@Shwan:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.1 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
shwan@Shwan:~$
```

可以看到版號 312512011(個人學號)以及 ID ID\_LIKE 等等資訊，顯示變更成功。

附註：每次執行編譯完新的 kernel，需要在開機時選到編譯完的 kernel，否則

系統預設為 kernel 6.8.0，我們這次要實作的是 kernel 6.1.0.(下方附上選擇畫面

Advanced options for Ubuntu→Ubuntu with linux 6.1.0-os-312512011+)



# Implementing a new System Calls

我的方法：

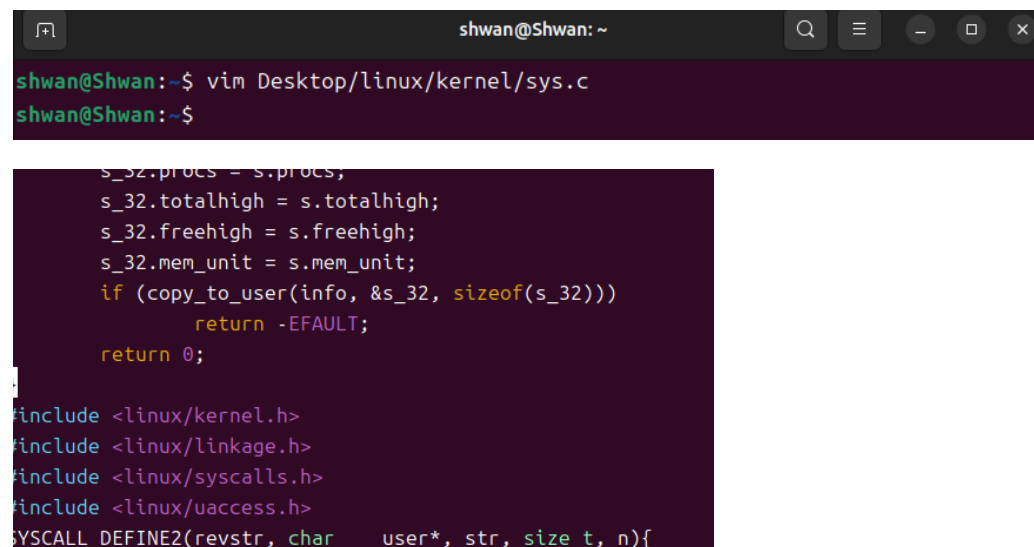
- (1) 直接對 `Desktop/linux/kernel` 的 `sys.c` 檔案進行修改，把我自定義的 `system call(sys_revstr)` 加在這個檔案的底部，
- (2) 在 `include/linux/syscall.h` 的 headfile 以 `asmlinkage long sys_revstr` 宣告新的 system call
- (3) 在 `arch/x86/entry/syscalls/syscall_64.tbl` 的 system call table 中註冊自定義的 `system call(No.451)`
- (4) 在 `linux/kernel` 內的 `Makefile` 檔案需要加入 `obj-y += sys.o` 這一行，才能把 `sys.c` 這個檔案編譯成為 kernel 的一部分，也就是在編譯時把 `sys.c` 編譯成對應的 `sys.o` 物件檔，並被打包到整個 kernel image file 當中。



以下會附上各個檔案修改過後的 screenshot，來證明我所做的每一個步驟：

(註：過程中我所使用的皆為 vim 文字編輯器)

(1) sys.c 尾端添加自定義的 system call(第二張圖證明前方有其他 system call)



```
shwan@Shwan: ~  
shwan@Shwan:~$ vim Desktop/linux/kernel/sys.c  
shwan@Shwan:~$  
  
    s_32.procs = s.procs;  
    s_32.totalhigh = s.totalhigh;  
    s_32.freehigh = s.freehigh;  
    s_32.mem_unit = s.mem_unit;  
    if (copy_to_user(info, &s_32, sizeof(s_32)))  
        return -EFAULT;  
    return 0;  
}  
  
#include <linux/kernel.h>  
#include <linux/linkage.h>  
#include <linux/syscalls.h>  
#include <linux/uaccess.h>  
SYSCALL_DEFINE2(revstr, char __user*, str, size_t, n){
```

上一頁的圖證明我的程式所在位置，下一頁的圖是程式的內容。



```
#include <linux/linkage.h>  
#include <linux/syscalls.h>  
#include <linux/uaccess.h>  
SYSCALL_DEFINE2(revstr, char __user*, str, size_t, n){  
    char kstr[256];  
    char revs[256];  
  
    if (copy_from_user(kstr, str, n)){  
        return -EFAULT;  
    }  
    kstr[n] = '\0';  
    printk("Ori: %s\n", kstr);  
    for( int i=0; i<n; i++){  
        revs[i] = kstr[n-1-i];  
    }  
    revs[n] = '\0';  
    printk("Rev: %s\n", revs);  
    if (copy_to_user(str, revs, n)){  
        return -EFAULT;  
    }  
    return 0;  
}  
#endif /* CONFIG_COMPAT */
```

2813,2-9 底端

結構說明：先定義 kernel space 內的一個字串 kstr，長度預設 256(個人認為這

應該是足夠應付平常的字串大小)，接著使用 `copy_from_user(kstr, str, n)` 從

userspace 抓取目標 string(str)以及長度(n)，接著 `kstr[n] = '\0'` 使最末端

的字串為空白，表示這個 string 的結束。接著反轉原本的字串接著

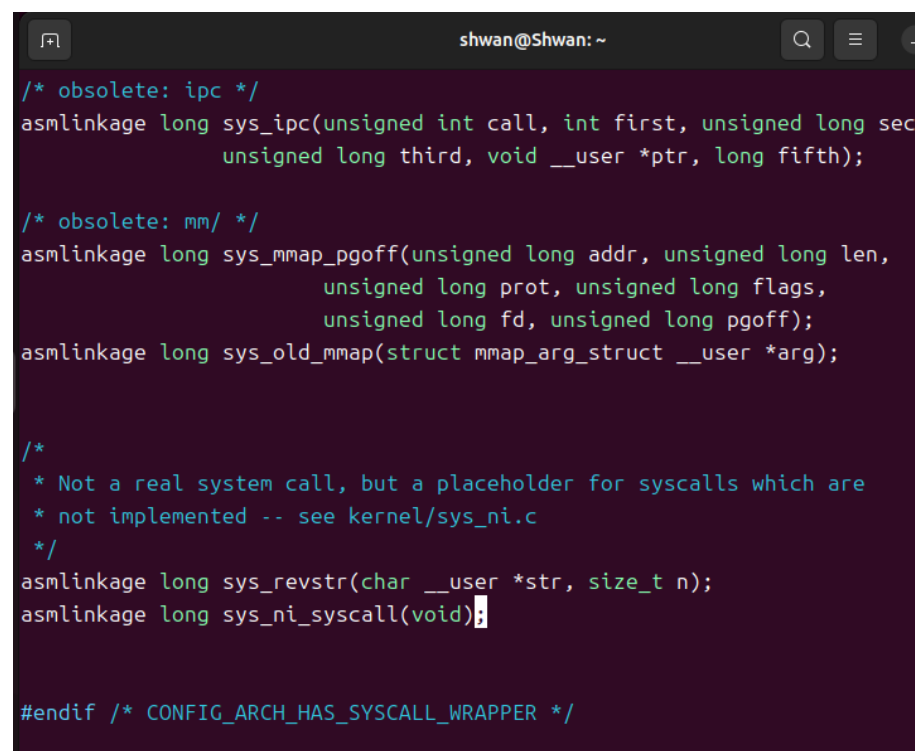
`copy_to_user(str, revs, n)` 返回 userspace，並 return 0 表示這個部分的 code

成功執行。並且在 `copy_from_user(kstr, str, n)` 和 `copy_to_user(str, revs, n)`

失敗的時候回傳錯誤作為防呆機制。

(2) 執行 `vim cd Desktop/linux/include/linux/syscall.h`，並加入一行(游標所

在的位置 `asmlinkage long sys_revstr(char __user *str, size_t n);`



```
/* obsolete: ipc */
asmlinkage long sys_ipc(unsigned int call, int first, unsigned long sec
                        unsigned long third, void __user *ptr, long fifth);

/* obsolete: mm/ */
asmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned long len,
                                unsigned long prot, unsigned long flags,
                                unsigned long fd, unsigned long pgoff);
asmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);

/*
 * Not a real system call, but a placeholder for syscalls which are
 * not implemented -- see kernel/sys_ni.c
 */
asmlinkage long sys_revstr(char __user *str, size_t n);
asmlinkage long sys_ni_syscall(void);

#endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */
```

(3)修改 `arch/x86/entry/syscalls/syscall_64.tbl` 這個 system call table，加入

451 common revstr sys\_revstr(圖中紅線框框處)註冊新的 system call

```
shwan@Shwan: ~/Desktop/linux/arch/x86/entry/syscalls
435 common clone3 sys_clone3
436 common close_range sys_close_range
437 common openat2 sys_openat2
438 common pidfd_getfd sys_pidfd_getfd
439 common faccessat2 sys_faccessat2
440 common process_madvise sys_process_madvise
441 common epoll_pwait2 sys_epoll_pwait2
442 common mount_setattr sys_mount_setattr
443 common quotactl_fd sys_quotactl_fd
444 common landlock_create_ruleset sys_landlock_create_ruleset
445 common landlock_add_rule sys_landlock_add_rule
446 common landlock_restrict_self sys_landlock_restrict_self
447 common memfd_secret sys_memfd_secret
448 common process_mrelease sys_process_mrelease
449 common futex_waitv sys_futex_waitv
450 common set_mempolicy_home_node sys_set_mempolicy_home_node
451 common revstr sys_revstr
#
# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64. These syscalls have numbers 512-547.
# Do not add new syscalls to this range. Numbers 548 and above are available
# for non-x32 use.
#
```

(4)修改 kernel 內的 Makefile 檔，使新增的 sys.c 在編譯時產生的物件檔 sys.o

會被視為 kernel image file 的一部分，使新功能加入 kernel。

```
shwan@Shwan: ~/Desktop/linux/kernel
CFLAGS_REMOVE_cfi.o := $(CC_FLAGS_CFI)

obj-y += sched/
obj-y += locking/
obj-y += power/
obj-y += printk/
obj-y += irq/
obj-y += rcu/
obj-y += livepatch/
obj-y += dma/
obj-y += entry/
obj-y += sys.o
obj-$(CONFIG_MODULES) += module/

obj-$(CONFIG_KCMP) += kcmp.o
obj-$(CONFIG_FREEZER) += freezer.o
```

以上修改完成後，和加入 local version 的流程相同，以 sudo 權限執行 `make -j4`、`make_modules_install` 和 `make install` 三個指令：

我寫了一個 `test_revstr.c` 來測試 system call 的功能，先印出當前輸入字串，之後進入 system call 將其反轉，再來以是否 return 0 告訴使用者新的 system call 是否成功執行，最後印出執行後的結果(反轉的字串)。

```
shwan@Shwan: ~  
shwan@Shwan: ~/Desktop  
#include <unistd.h>  
#include <string.h>  
#include <stdio.h>  
#include <assert.h>  
#include <sys/syscall.h>  
#include <linux/kernel.h>  
#define __NR_revstr 451  
int main(int argc, char *argv[]){  
    char str1[16] = "hello";  
    printf("Original string is: %s\n", str1);  
    int rev1 = syscall(__NR_revstr, str1, strlen(str1));  
    printf("System call sys_revstr returned %d\n ", rev1);  
    printf("Reversed string is: %s\n", str1);  
    assert(rev1 == 0);  
    char str2[20] = "Operating System";  
    printf("Original string is: %s\n", str2);  
    int rev2 = syscall(__NR_revstr, str2, strlen(str2));  
    printf("System call sys_revstr returned %d\n ", rev2);  
    printf("Reversed string is: %s\n", str2);  
  
    return 0;  
}
```

此程式碼執行結果如下，可以看到 hello 字串被反轉成 olleh，並且有回傳 0 表示此一 system call 被觸發並成功地執行。

```
shwan@Shwan:~/Desktop$ gcc -o test_revstr test_revstr.c  
shwan@Shwan:~/Desktop$ ./test_revstr  
Original string is: hello  
System call sys_revstr returned 0  
Reversed string is: olleh  
Original string is: Operating System  
System call sys_revstr returned 0  
Reversed string is: metsyS gnitarep0
```

附錄：Patch 生成過程

git init：初始化一個 git repository，開始追蹤所有變更

git add .：把所有變更的檔案加入到暫存區，為下一步的提交做準備。

git commit -m 'New'：提交變更並附上提交訊息 "New"

git format-patch master：生成 patch

```
496 cd Desktop
497 ./test_revstr
498 git init
499 git add .
500 git add.
501 git add
502 git add .
503 git reset --merge
```

```
527 git format-patch origin/main
528 git format-patch origin/New
529 git format-patch origin--New
530 git format-patch master
531 vim 0001-Try.patch
532 dpkg --get-selections>packages_list<312512011>.txr
533 dpkg --get-selections>packages_list<312512011>.txt
534 dpkg --get-selections > packages_list<312512011>.txt
535 dpkg --get-selections > packages_list_<3125121011>.txt
536 dpkg --get-selections > packages_list_312512011.txt
537 cd /path/to/your/files
```

按照 assignment 指示列出用到的 packages list，複製到指定資料夾內壓縮成

zip 檔案上傳，即完成此次作業的內容。

我的書面報告到此結束，謝謝助教耐心批閱