HW1

I. Compiling the Linux Kernel

參考: https://phoenixnap.com/kb/build-linux-kernel

1. 根據助教提供之設定步驟建立環境。
2. building the kernel 前，安裝必要的 package

   Command: **sudo apt-get install git build-essential ncurses-dev libssl-dev bc flex libelf-dev bison**
3. 修改 .config ,然後設定作業中所要求的 kernel local version suffix

   Command: **make menuconfig**

   vim .config

```
CONFIG_CC_CAN_LINK=y
CONFIG_CC_CAN_LINK_STATIC=y
CONFIG_CC_HAS_ASM_GOTO_OUTPUT=y
CONFIG_CC_HAS_ASM_GOTO_TIED_OUTPUT=y
CONFIG_CC_HAS_ASM_INLINE=y
CONFIG_CC_HAS_NO_PROFILE_FN_ATTR=y
CONFIG_PAHOLE_VERSION=0
CONFIG_IRQ_WORK=y
CONFIG_BUILDTIME_TABLE_SORT=y
CONFIG_THREAD_INFO_IN_TASK=y

#
# General setup
#
CONFIG_INIT_ENV_ARG_LIMIT=32
# CONFIG_COMPILE_TEST is not set
# CONFIG_WERROR is not set
CONFIG_LOCALVERSION="-os-312512032"
# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_BUILD_SALT=""
CONFIG_HAVE_KERNEL_GZIP=y
CONFIG_HAVE_KERNEL_BZIP2=y
CONFIG_HAVE_KERNEL_LZMA=y
".config" 11833L, 272557B                                    31,34         0%
```

4. building the kernel 、安裝 modules

   Command: **make**

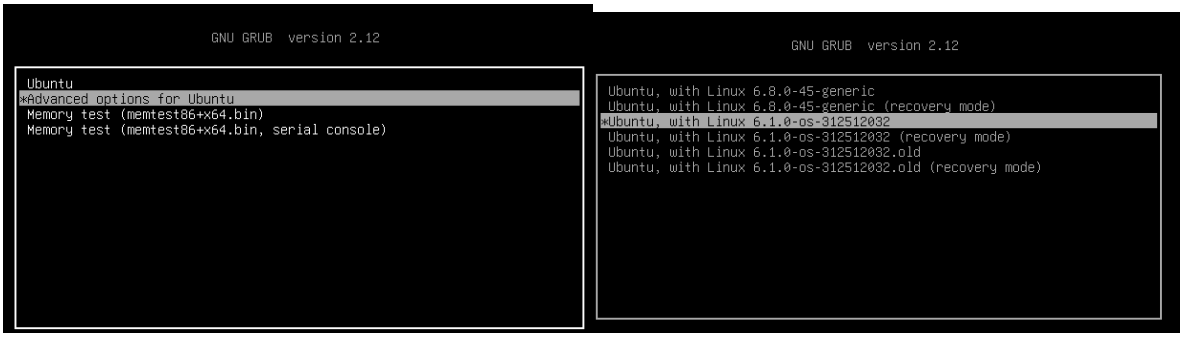
   sudo make modules_install
5. 確認要安裝的 kernel 版本

   Command: **make kernelrelease**

```
benson-312512032@benson-312512032-VirtualBox:~/linux$ make kernelrelease
6.1.0-os-312512032
```

6. 安裝 kernel.

   Command: **sudo make install**
7. 設定開機時所要選的 kernel.(開機時按 Esc 進入 GNU GRUB 確認你的 kernel 在選單的哪裡 Example: Advanced options for Ubuntu > Ubuntu, with Linux 6.1.0-os-312512032, 更新 /etc/default /grub 的 GRUB_DEFAULT)




```
# 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="1>2"
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`( . /etc/os-release; echo ${NAME:-Ubuntu} ) 2>/dev/null || ech
o Ubuntu`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# If your computer has multiple operating systems installed, then you
# probably want to run os-prober. However, if your computer is a host
# for guest OSes installed via LVM or raw disk devices, running
# os-prober can cause damage to those guest OSes as it mounts
# filesystems to look for things.
GRUB_DISABLE_OS_PROBER=false

# 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 ...)
"/etc/default/grub" 40L, 1553B                                1,1           Top
```

8. 更新 GRUB bootloader

   Command: **sudo update-grub**

```
benson-312512032@benson-312512032-VirtualBox:~/linux$ sudo update-grub
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.8.0-45-generic
Found initrd image: /boot/initrd.img-6.8.0-45-generic
Found linux image: /boot/vmlinuz-6.1.0-os-312512032
Found initrd image: /boot/initrd.img-6.1.0-os-312512032
Found linux image: /boot/vmlinuz-6.1.0-os-312512032.old
Found initrd image: /boot/initrd.img-6.1.0-os-312512032
Found memtest86+x64 image: /boot/memtest86+x64.bin
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

9. 重新開機

   Command: **sudo reboot**

10. 確認當前之 kernel 版本

    Command: **uname -a**

    cat /etc/os-release

```
benson-312512032@benson-312512032-VirtualBox:~/Desktop$ uname -a
Linux benson-312512032-VirtualBox 6.1.0-os-312512032 #2 SMP PREEMPT_DYNA
MIC Tue Sep 24 15:37:40 CST 2024 x86_64 x86_64 x86_64 GNU/Linux
benson-312512032@benson-312512032-VirtualBox:~/Desktop$ cat /etc/os-rele
ase
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/priv
acy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
benson-312512032@benson-312512032-VirtualBox:~/Desktop$
```

II. Implementing a new System Calls

1. 在 linux 內建立 revstr 資料夾,並撰寫該 system call 的內容於 revstr.c
   此函數有兩個參數 msg、len，因此使用 SYSCALL_DEFINE2.
   由於 user/kernel space 的資料無法直接與 kernel/user space 共用,因此使用 copy_from_user()、copy_to_user,進行資料的傳遞,並以動態記憶體配置的方式給儲存資料(為什麼是 len+1，因為 user space 傳遞過來的長度不包含 '\0',導致%s 在印 System 時會多一串亂碼，因此透過此方式解決),接著由頭尾互相交換直到中間,完成字串反轉,最後釋放動態記憶體。

```c
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <linux/slab.h>
#include <linux/uaccess.h>
// syscall_num =451
SYSCALL_DEFINE2(revstr,char __user * ,msg,unsigned int ,len){
    //printk(KERN_INFO"%d",len);
    char* kernel_msg =  kmalloc((len+1)*sizeof(char),GFP_KERNEL);
    if(! kernel_msg){
        printk(KERN_ERR "memory allocation failed \n");
        return -ENOMEM ;
    }
    if (copy_from_user(kernel_msg,msg,len)){
        printk(KERN_ERR "can not copy form user ,please check your address!\n");
        kfree(kernel_msg);
        return -EFAULT;
    }
/**********revstr function*********/
    kernel_msg[len]='\0';
    printk(KERN_INFO "The origin string: %s\n",kernel_msg);
    for(unsigned int i=0,j=len-1 ;i<j;i++,j--){
        char temp = kernel_msg[i];
        kernel_msg[i]=kernel_msg[j];
        kernel_msg[j]=temp;
    }
    kernel_msg[len]='\0';
    printk(KERN_INFO "The reversed string: %s\n",kernel_msg);
/*********************************/
    if ( copy_to_user(msg,kernel_msg,len)){
        printk(KERN_ERR "can not copy to user!!\n");
        return -EFAULT;
    }
    kfree(kernel_msg);
    return 0;
}
```

2. 撰寫編譯 revstr.c 的 Makefile 於 ~/linux/revstr,使其被編譯並連結至 kernel image 內

```
obj-y := revstr.o
```

3. 將所撰寫的 sys 加入 ~/linux/arch/x86/entry/syscalls/syscall_64.tbl ，加入 64 bit 的最後面，我們的 system call number 是 451，名稱為 revstr,進入點為 sys_revstr

```
# don't use numbers 387 through 423, add new calls after the last
# 'common' entry
424     common  pidfd_send_signal      sys_pidfd_send_signal
425     common  io_uring_setup         sys_io_uring_setup
426     common  io_uring_enter         sys_io_uring_enter
427     common  io_uring_register      sys_io_uring_register
428     common  open_tree              sys_open_tree
429     common  move_mount             sys_move_mount
430     common  fsopen                 sys_fsopen
431     common  fsconfig               sys_fsconfig
432     common  fsmount                sys_fsmount
433     common  fspick                 sys_fspick
434     common  pidfd_open             sys_pidfd_open
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.
```

4. 於 ~/linux/include/linux/syscalls.h 宣告 sys_revstr 的原型，使 Linux Kernel 編譯器就能識別這個新系統呼叫，並在需要的地方呼叫它

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

/* obsolete: mm/ */
asmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned long
                        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 a
 * not implemented -- see kernel/sys_ni.c
 */
asmlinkage long sys_ni_syscall(void);
/* revstr/revstr.h */
asmlinkage long sys_revstr(char __user * msg ,unsigned int len);
```

5. 將資料夾~/linux/revstr 之相對路徑加入~/linux/Makefile 中的 core-y ,使其加入在編譯過程中

```
ifneq ($(filter all modules nsdeps %compile_commands.json clang-%,$(MAKECMDGOALS)),)
  KBUILD_MODULES := 1
endif

ifeq ($(MAKECMDGOALS),)
  KBUILD_MODULES := 1
endif

export KBUILD_MODULES KBUILD_BUILTIN

ifdef need-config
include include/config/auto.conf
endif

ifeq ($(KBUILD_EXTMOD),)
# Objects we will link into vmlinux / subdirs we need to visit
core-y          := revstr/
drivers-y       :=
libs-y          := lib/
endif # KBUILD_EXTMOD
```

6. 重新燒入並重新啟動

   Command: make -j8 && sudo make modules_install && sudo make install && sudo reboot

7. 編寫測試檔案,__NR_revstr 是 system call number ,藉此讓 syscall() function 執行 system call table 中對應號碼的 system call

```
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <assert.h>

#define __NR_revstr 451

int main(int argc ,char* argv[]){

    char str1[20] ="hello";
    printf("Ori: %s\n",str1);
    int ret1 = syscall(__NR_revstr,str1,strlen(str1));
    assert(ret1 == 0);
    printf("Rev: %s\n",str1);

    char str2[20] = "Operating System";
    printf("Ori: %s\n",str2);

    int ret2=syscall(__NR_revstr,str2,strlen(str2));
    assert(ret2 == 0);
    printf("Rev: %s\n",str2);

    return 0;
}
```

8. 執行測試檔案和查看 user space 的結果及 dmesg 之 kernel space 的結果