Linux OS

Project 2

106522102 陳品豪

106522066 賴孟昇

106522033 陳韋佑

I. 前言：

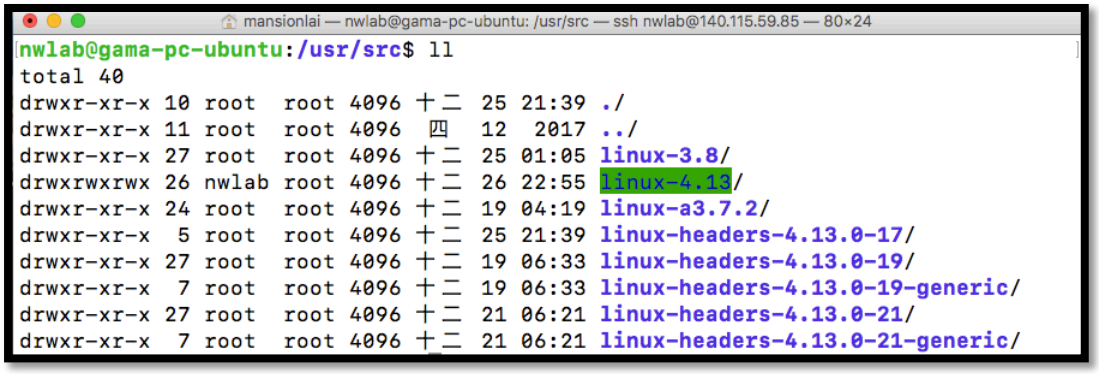
- 一開始考慮要在 Fedora 還是 Ubuntu 上實作，由於 Fedora 我們只有虛擬機，而 Ubuntu 確有一台現成的實體機可用，而且是 Intel Core i7 處理器，CPU 有八個核心在編譯速度上快上不少，所以採用 Ubuntu 作為本次專題的發行版。
- 建置環境

  Ubuntu 17.10 x86_64 with Linux 4.13

II. Project 2-1：

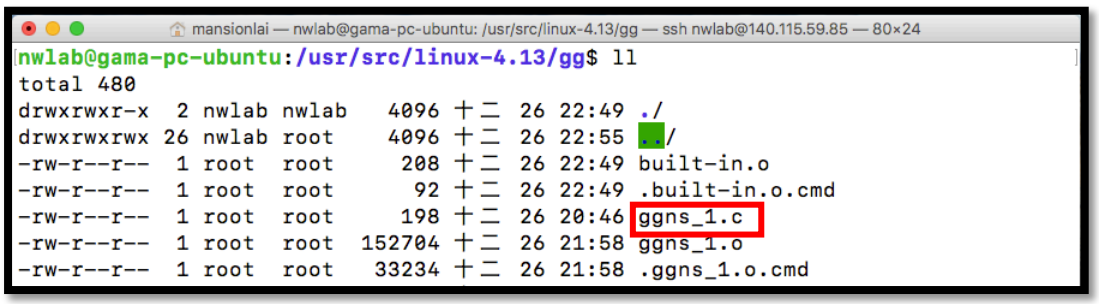
Step1:

下載 Linux Kernel 4.13 source code，並放置於 /usr/src 目錄下。



Step2:

在 linux-4.13 目錄中，建立新的資料夾，名為 gg，用來存放新加入的程式碼，cd 到 gg 內，並建立 Project 2-1 的 System call (ggns_1.c)。



Step3:

編輯 ggns_1.c，程式碼如下圖所示。

```
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$ cat ggns_1.c
#include <linux/kernel.h>
#include <linux/linkage.h>
#include <linux/sched.h>
#include <linux/pid_namespace.h>

asmlinkage int get_ns_level(void) {
        return current->pids[PIDTYPE_PID].pid->level;
}
```

Step4:

開啟 linux-4.13/include/linux/syscalls.h，加入新增的 System call function

- asmlinkage int get_ns_level(void)

```
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/include/linux$ ll | grep syscalls
-rw-r--r--  1 root root  40135 十二  26 21:56 syscalls.h
```

```
904 asmlinkage long sys_pkey_alloc(unsigned long flags, unsigned long init_val);
905 asmlinkage long sys_pkey_free(int pkey);
906 asmlinkage long sys_statx(int dfd, const char __user *path, unsigned flags,
907                           unsigned mask, struct statx __user *buffer);
908 asmlinkage int get_ns_level(void);
909 struct pdps_with_pid_at_ancestor_ns;
910 asmlinkage int get_pdps_of_pid_at_ancestor_ns(struct pdps_with_pid_at_ancest
    or_ns * results, long pid);
911 #endif
                                                          908,34        Bot
```

Step5:

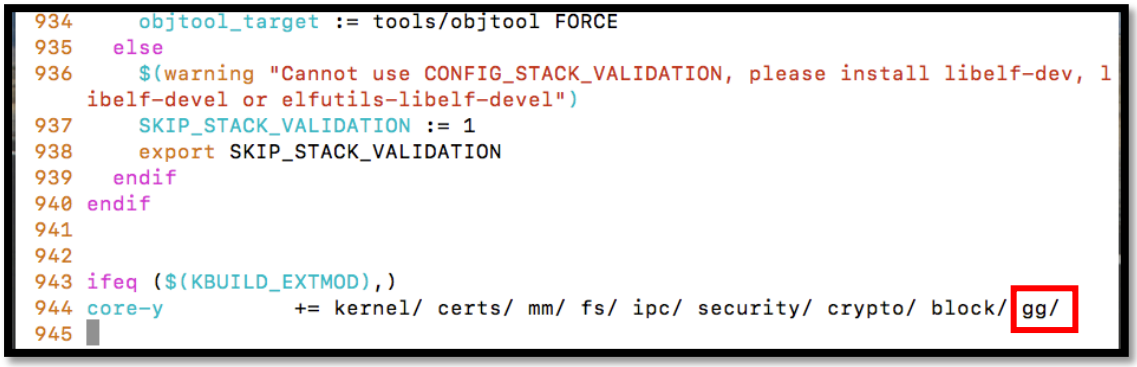
切換到先前新增的 linux-4.13/gg 資料夾中，並且新增 Makefile，並且在 Makefile 中加入"obj-y:=ggns_1.o"，告訴 make 要告訴 make 要編譯該資料夾中的程式。

```
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$ ll | grep Makefile
-rw-r--r--  1 root  root      25 十二  26 20:47 Makefile
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$ cat Makefile
obj-y:=ggns_1.o ggns_2.o
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$
```

Step6:

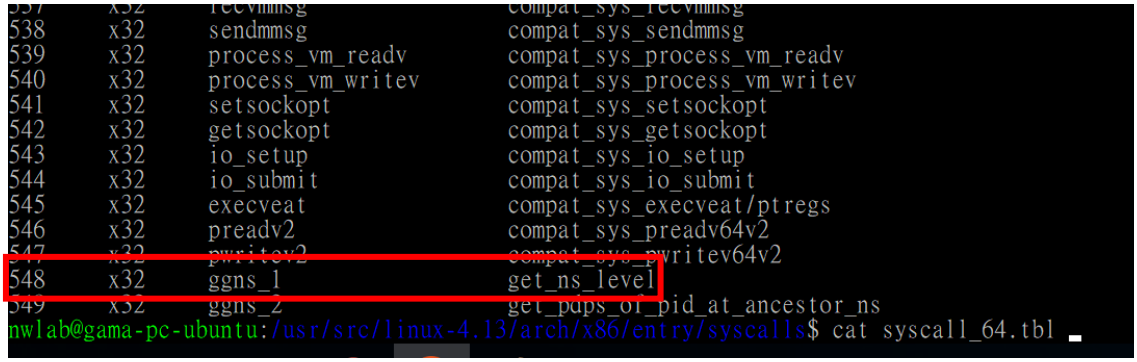
在 linux-4.13 目錄下的 Makefile 修改一行指令，確保我們新增的資料夾 "gg" 會加入編譯 Kernel 的清單中。

如下圖 944 行，在最後面加上 gg/



Step7:

在 liunx-4.13/arch/x86/entry/syscalls/syscall_64.tbl 新增我們的 System call，並給他一個編號（548）。



Step8:

接下來開始編譯我們的 Kernel

- cd 到/usr/src/linux-4.13
- 下指令 *sudo make menuconfig // 選擇要編譯的功能，並且產生 .config 檔*
- 下指令 *sudo make oldconfig // 告知編譯過程使用的設定檔為現有的 .config*
- 下指令 *sudo make -j8 // 因為主機是 8 核心，使用 8 個執行緒編譯可加快速度*
- 下指令 *sudo make module -j8 // 編譯 Driver*

- 下指令 *sudo make module_install // 將 Driver 安裝至該 kernel*
- 下指令 *sudo make install // 將編譯好的 kernel image 安裝至 /boot 之下*

Step9:

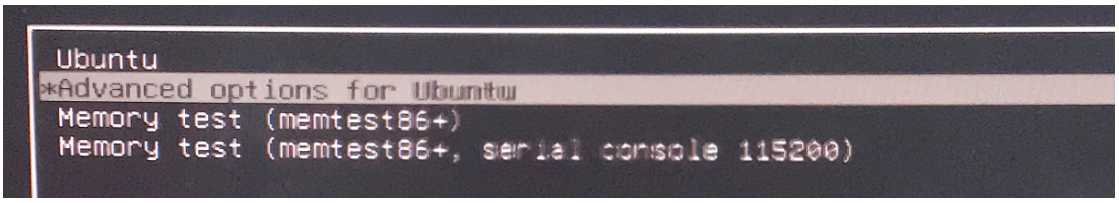
在開機選單中新增我們編譯完成的 Kernel

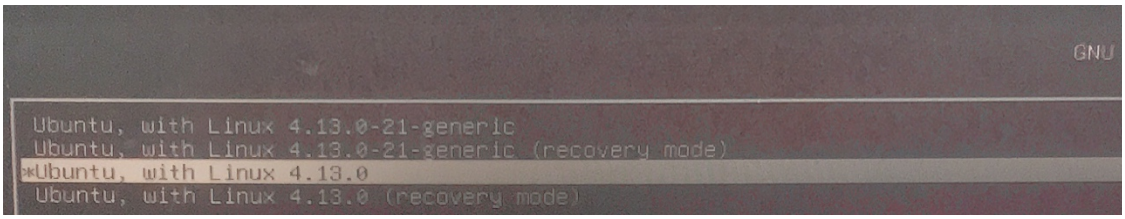
- 下指令 *sudo update-grub2*

註：該指令為 Ubuntu 中內建，會自動掃描 /boot 下的目錄中有哪些 kernel image，並且自動產生開機的選單。

Step10:

最後就重開機選擇我們編譯好的 Kernel 開機囉！

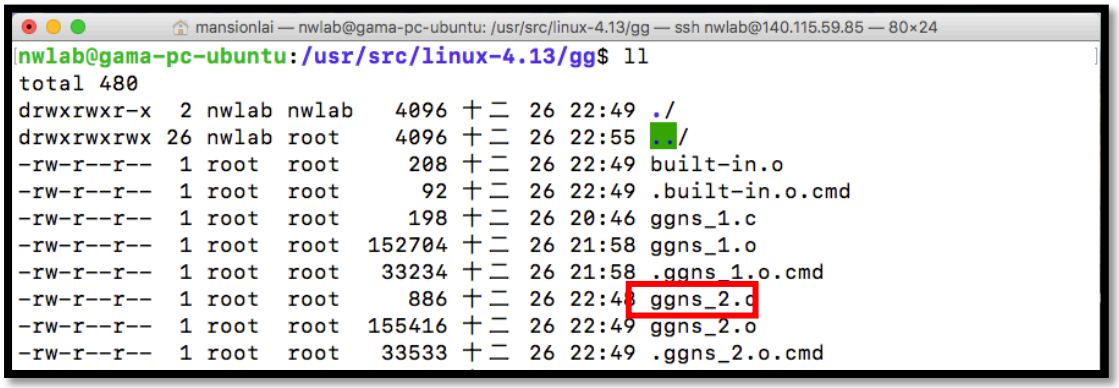




III. Project 2-2：

步驟與 project2-1 相同，僅有程式名稱及 system call 名稱不同。

- 新增 ggns_2.c，加入程式碼



```
mansionlai — nwlab@gama-pc-ubuntu: /usr/src/linux-4.13/gg — ssh nwlab@140.115.59.85 — 97×30
 1 #include <linux/kernel.h>
 2 #include <linux/linkage.h>
 3 #include <linux/pid_namespace.h>
 4 #include <linux/sched.h>
 5 #include <linux/uaccess.h>
 6
 7 struct pdps_with_pid_at_ancestor_ns {
 8         long process_descriptor_pointer;
 9 };
10
11 asmlinkage int get_pdps_of_pid_at_ancestor_ns(struct pdps_with_pid_at_ancestor_ns * results,
   long pid) {
12         struct pdps_with_pid_at_ancestor_ns _results[20];
13         struct pid * _pid = current->pids[PIDTYPE_PID].pid;
14         int max_level = _pid->level, temp_level;
15         struct pid_namespace * _ns = _pid->numbers[max_level].ns;
16
17         if (max_level == 0 || max_level > 19) return -1;
18
19         for (temp_level = max_level;temp_level >= 0; temp_level--) {
20                 _pid = find_pid_ns(pid, _ns);
21                 _results[temp_level].process_descriptor_pointer = (long) pid_task(_pid, PIDTY
   PE_PID);
22                 _ns = _ns->parent;
23         }
24         copy_to_user(results, _results, 20*sizeof(struct pdps_with_pid_at_ancestor_ns));
25         return max_level;
26 }
```

- 於 linux-4.13/include/linux/syscalls.h 中加入新增的 System call function
    - asmlinkage int get_pdps_of_pid_at_ancestor_ns(struct pdps_with_pid_at_ancestor_ns * results, long pid)

```
904 asmlinkage long sys_pkey_alloc(unsigned long flags, unsigned long init_val);
905 asmlinkage long sys_pkey_free(int pkey);
906 asmlinkage long sys_statx(int dfd, const char __user *path, unsigned flags,
907                           unsigned mask, struct statx __user *buffer);
908 asmlinkage int get_ns_level(void);
909 struct pdps_with_pid_at_ancestor_ns;
910 asmlinkage int get_pdps_of_pid_at_ancestor_ns(struct pdps_with_pid_at_ancest
    or_ns * results, long pid);
911 #endif
```

- 在 gg 目錄中的 Makefile 加入"obj-y:=ggns_2.o"

```
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$ ll | grep Makefile
-rw-r--r--  1 root  root      25 十二  26 20:47 Makefile
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$ cat Makefile
obj-y:=ggns_1.o ggns_2.o
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/gg$
```

- 開啟 liunx-4.13/arch/x86/entry/syscalls/syscall_64.tbl，新增 project2-2 的 System call，並給他一個編號（549）

```
540     x32     process_vm_writev       compat_sys_process_vm_writev
541     x32     setsockopt              compat_sys_setsockopt
542     x32     getsockopt              compat_sys_getsockopt
543     x32     io_setup                compat_sys_io_setup
544     x32     io_submit               compat_sys_io_submit
545     x32     execveat                compat_sys_execveat/ptregs
546     x32     preadv2                 compat_sys_preadv64v2
547     x32     pwritev2                compat_sys_pwritev64v2
548     x32     ggns_1                  get_ns_level
549     x32     ggns_2                  get_pdps_of_pid_at_ancestor_ns
nwlab@gama-pc-ubuntu:/usr/src/linux-4.13/arch/x86/entry/syscalls$ cat syscall_64.tbl
```

IV. 作業成果：

作業成果測試，由助教提供的測試程式判斷我們的 System call 是否能達到作業的要求。

- Project2-1:

```
mansionlai — nwlab@gama-pc-ubuntu: ~/testprogram — ssh nwlab@140.115.59.85 — 84×24
nwlab@gama-pc-ubuntu:~/testprogram$ sudo ./testprogram_64
=========Linux Project 2 verify program.=========
================================================
1. verify syscall: get_ns_level
2. verify syscall: get_pdps_of_pid_at_ancestor_ns
3. Exit.
================================================
Enter your choice: 1
Please input your syscall number of get_ns_level : 548
The level is 12
```

- Project2-2:

```
mansionlai — nwlab@gama-pc-ubuntu: ~/testprogram — ssh nwlab@140.115.59.85 — 84×27
nwlab@gama-pc-ubuntu:~/testprogram$ sudo ./testprogram_64
=========Linux Project 2 verify program.=========
================================================
1. verify syscall: get_ns_level
2. verify syscall: get_pdps_of_pid_at_ancestor_ns
3. Exit.
================================================
Enter your choice: 2
Please input your syscall number of get_pdps_of_pid_at_ancestor_ns : 549
process descriptor of local pid 3
At level 0, process descriptor pointer addr is 1a7b15c0
At level 1, process descriptor pointer addr is 0
At level 2, process descriptor pointer addr is cfa595c0
At level 3, process descriptor pointer addr is 0
At level 4, process descriptor pointer addr is d5332b80
At level 5, process descriptor pointer addr is 0
At level 6, process descriptor pointer addr is cfc30000

process descriptor of local pid 6
At level 0, process descriptor pointer addr is 1a002b80
At level 1, process descriptor pointer addr is cfa5c140
At level 2, process descriptor pointer addr is d2bb0000
At level 3, process descriptor pointer addr is d5332b80
At level 4, process descriptor pointer addr is cfc34140
At level 5, process descriptor pointer addr is cfc30000
At level 6, process descriptor pointer addr is 0
```

V. 結論

我們一開始依照程式架構寫了第一個版本，用來顯示當下 PID Namespace 的 Level。一開始以為 Linux 原先作業系統就已經擁有多層 NS，因為在 Ubuntu 中下指令 *lsns*，在一般使用者以及 root 權限下看到的內容也有差異，故認為 NS 應該是有值並且為大於 0 的 Level。

但是，不管怎麼更改程式，都會統一取得 Level 0 這個值。後來利用 Docker 以及 clone 的方式驗證我們最一開始寫的第一個版本就是正確的，讓我們折騰了非常久一段時間，非常恰巧的測試程式在隔天發布，若更早提供測試程式就不會多走這麼多冤妄路啊！希望下次 Project 題目可以與測試程式一同發布。

因為一直誤以為原本寫的是錯誤的，花了很多時間及研究是否有其他寫法，所以對其架構已經熟悉到不能再熟悉了，編譯也重複做了許多次，雖然多花了許多額外的時間，但也因此有了不少的收穫。