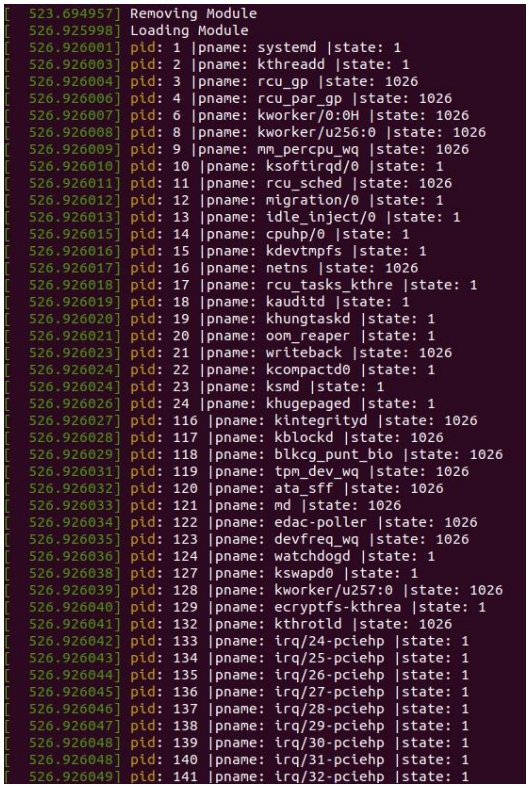
HW2 Listing Tasks

105072123 黃海茵

---------- LINEAR ----------

- Output



- Code

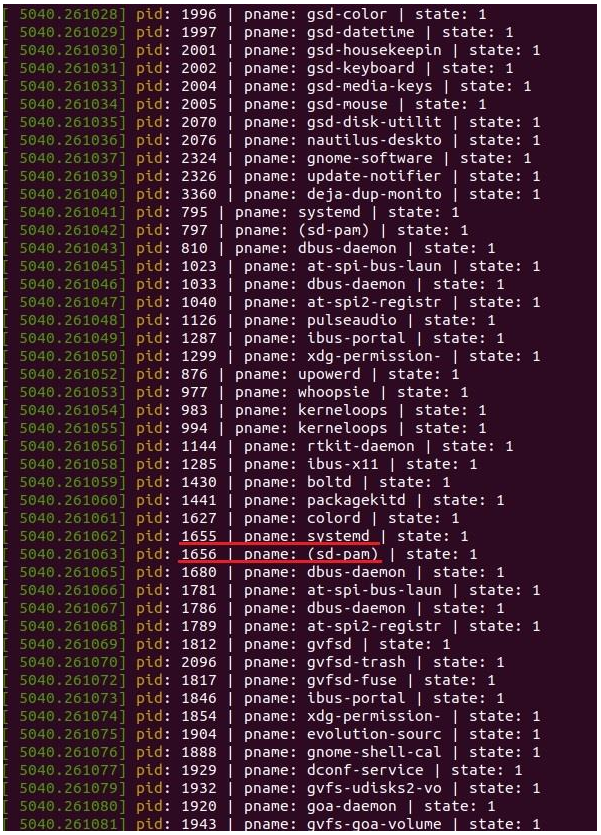
```c
#include <linux/sched/signal.h>
#include <linux/sched.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>

/* This function is called when the module is loaded. */
int entryPoint(void) {
        struct task_struct *task;

        printk(KERN_INFO "Loading Module\n");

        for_each_process(task) {
                printk(KERN_INFO "pid: %d |pname: %s |state: %d\n", task->pid, task->comm, task->state);
        }

        printk(KERN_INFO "---------------END--------------\n");
        return 0;
}

/* This function is called when the module is removed. */
void exitPoint(void) {
        printk(KERN_INFO "Removing Module\n");
}

/* Macros for registering module entry and exit points. */
module_init(entryPoint);
module_exit(exitPoint);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("List tasks linearly");
MODULE_AUTHOR("105072123");
```

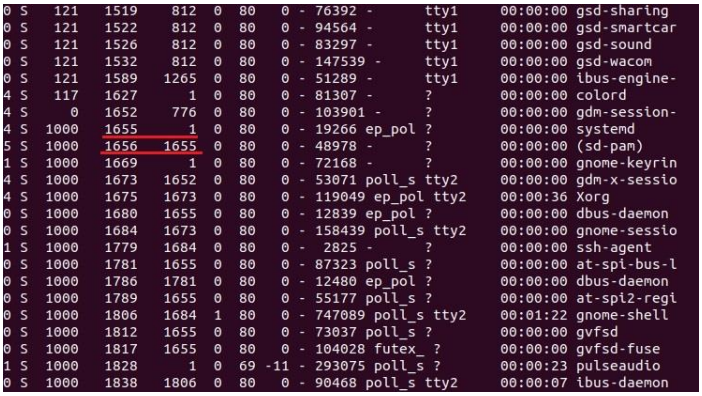
首先 include 所有會用到的 header 檔，然後建立一個<linux/sched.h>內建的 structure task_struct *task，作為要被印出 info 的 task 的 pointer，包含 prev 和 next。

再用<linux/sched/signal.h>內建的 for_each_process 這個 macro 依序拜訪所有 process。task->pid 是 task 的 process ID，task->comm 是 task 的 name，task->state 是 task 的 state，而-1 表示 unrunnable，0 表示 runnable，>0 表示 stopped。

---------- DFS ----------

- Output





- Code

```c
#include <linux/sched/task.h>
#include <linux/sched.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>

void dfs(struct task_struct *task) {
        struct task_struct *child;
        struct list_head *list;

        printk(KERN_INFO "pid: %d | pname: %s | state: %d\n", task->pid, task->comm, task->state);

        list_for_each(list, &task->children) {
                child = list_entry(list, struct task_struct, sibling);
                dfs(child);
        }
}

/* This function is called when the module is loaded. */
int entryPoint(void) {
        printk(KERN_INFO "Loading Module\n");

        dfs(&init_task);

        printk(KERN_INFO "---------------END---------------\n");
        return 0;
}

/* This function is called when the module is removed. */
void exitPoint(void) {
        printk(KERN_INFO "Removing Module\n");
}

/* Macros for registering module entry and exit points. */
module_init(entryPoint);
module_exit(exitPoint);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("List tasks using DFS");
MODULE_AUTHOR("105072123");
```

首先一樣先 include 所有會用到的 header 檔，過程中我發現 init_task 從 linux4.11 起，是在<linux/sched/task.h>中聲明，而非<linux/sched.h>中。所以如果只有 include<linux/sched.h>的話，會產生”init_task undeclared”的 error。

再呼叫 function dfs，然後跟第一題同樣建立一個 linux 內建的 structure task_struct *task，傳入 init_task。init_task 就是 kernel 中 process 0 使用的 process 描述符，也是 Linux 系統中第一個 process 描述符。

在 fuction dfs 中，首先建立一個 struct task_struct *child，作為 next child 的 pointer。再建立一個 struct list_head *list，list_head 在上一次的作業中就有使用過了，是用來儲存 prev 和 next 的 list。然後 printk 的部分就跟第一題是一模一樣的。

再用 list_for_each 來 traverse task 的 list，list 為一個 pointer 並指向第一個節點”task 的 children”，以此為起點一路 traverse 下去，終止條件是當 pointer 指向 head 時，代表已經 traverse 完一圈了。然後在這個 loop 中使用 list_entry 找到 task 的 children 的 sibling，將它們設為 child，最後傳入 dfs 中再次呼叫的 dfs fuction，不斷 iterate。