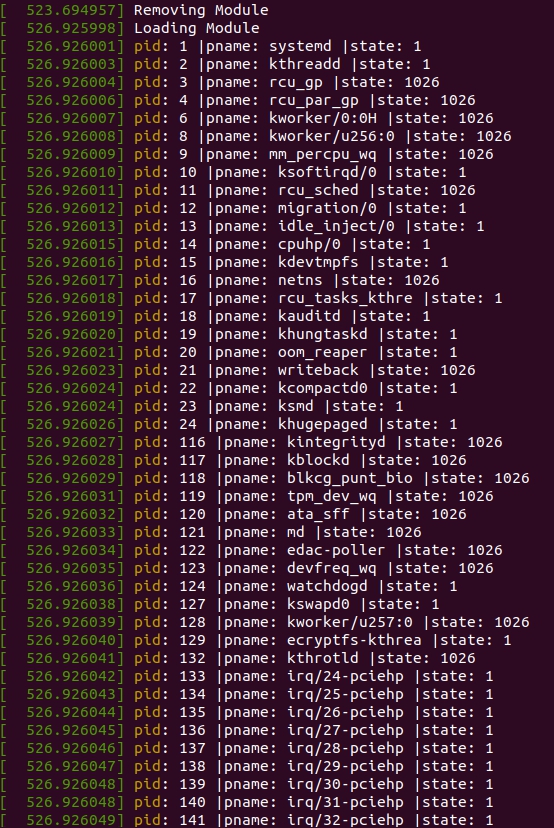
HW2 Listing Tasks

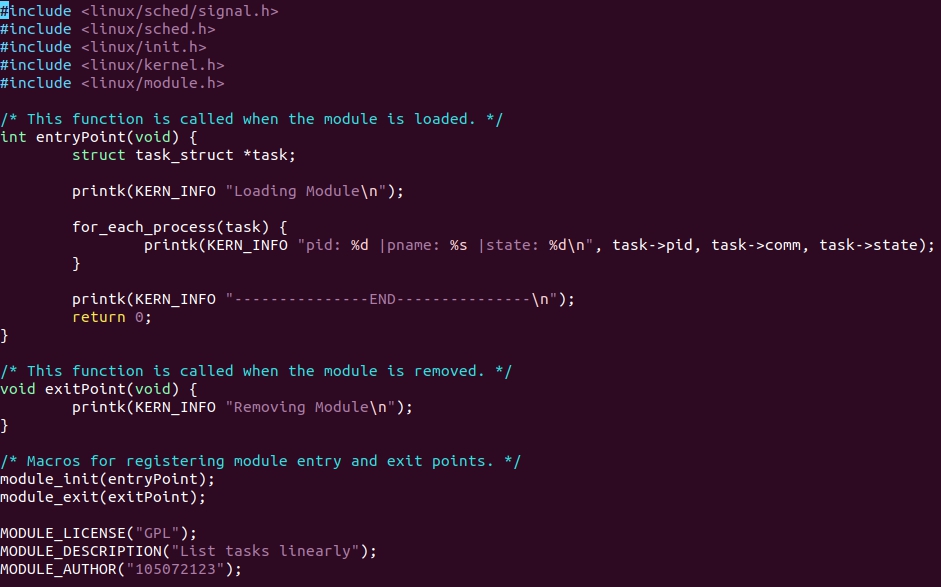
105072123黃海茵

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

* Output



* Code



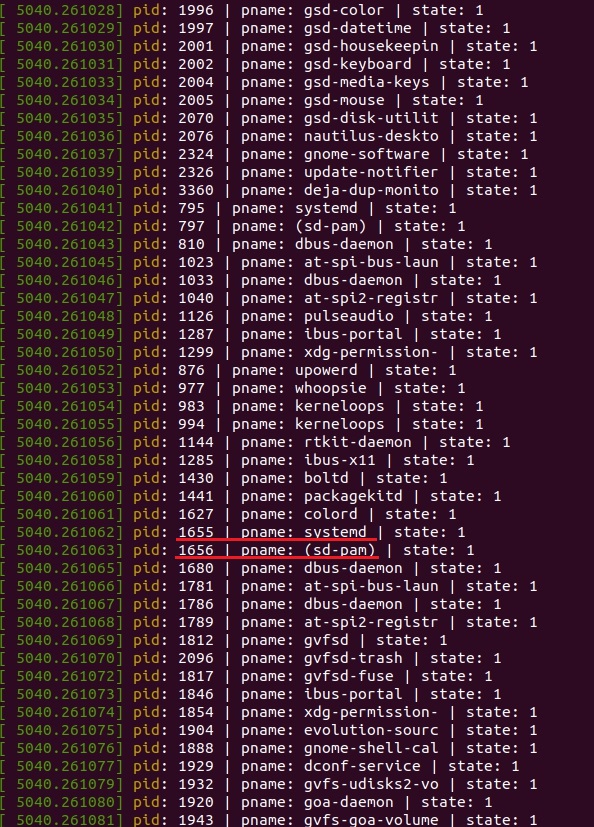
首先include所有會用到的header檔，然後建立一個<linux/sched.h>內建的

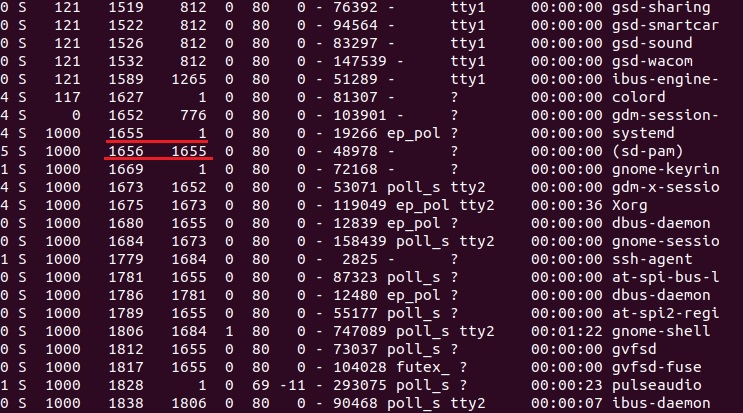
structure task\_struct \*task，作為要被印出info的task的pointer，包含prev和next。

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。

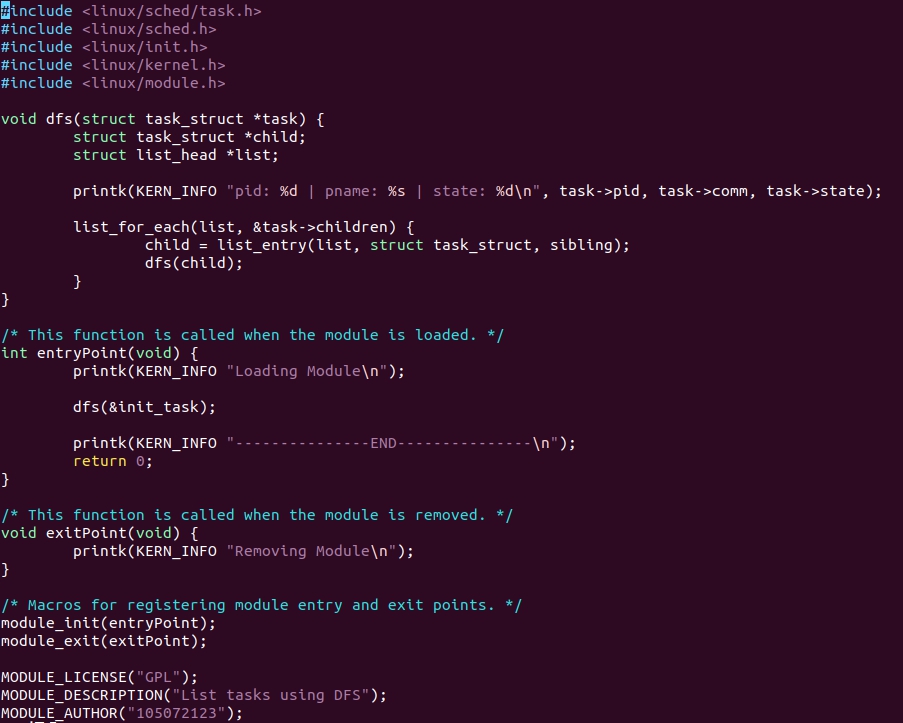
----------------------------------------------------- DFS -----------------------------------------------------

* Output





* Code



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系统中第一個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。