石育瑋 108062633

CS 342300 Operating System

**OS HW2: Linsting Task**

**Implementation**

1. linear 版本

int hw\_init(void)

{

struct task\_struct \*p;

char p\_buf\_tempalte[] = KERN\_INFO "pid: %6d|pname: %20s|state: %6d\n";

for\_each\_process(p) {

printk(p\_buf\_tempalte, p->pid, p->comm, p->state);

}

return 0;

}

linear版本非常簡單，只需要call <linux/sched/signal.h> 裡面提供的for\_each\_process()就能traverse所有的process，再把資訊print出來就完成了！

1. dfs版本

void dfs(struct task\_struct \*t\_current)

{

struct task\_struct \*t\_children;

struct list\_head \*l\_children;

list\_for\_each(l\_children, &t\_current->children) {

t\_children = list\_entry(l\_children, struct task\_struct, sibling);

printk(p\_buf\_template, t\_children->pid, t\_children->comm, t\_children->state);

dfs(t\_children);

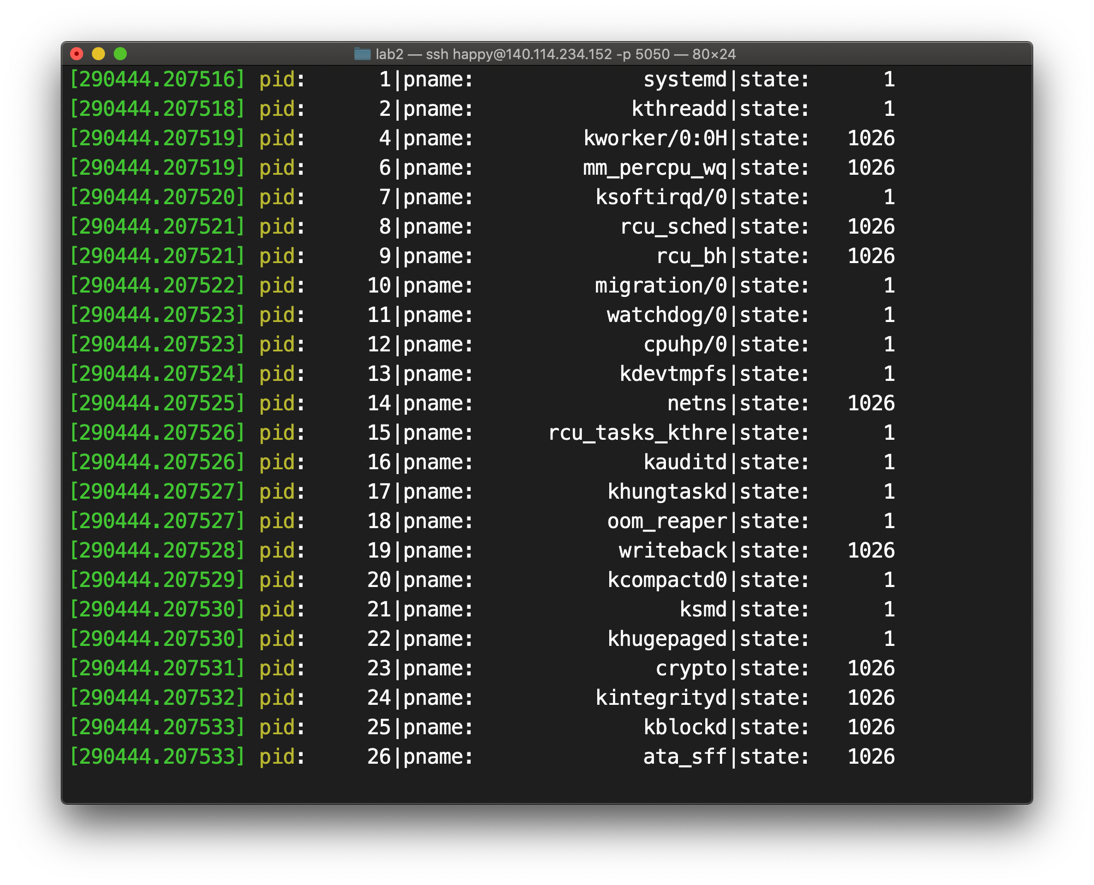
}

}

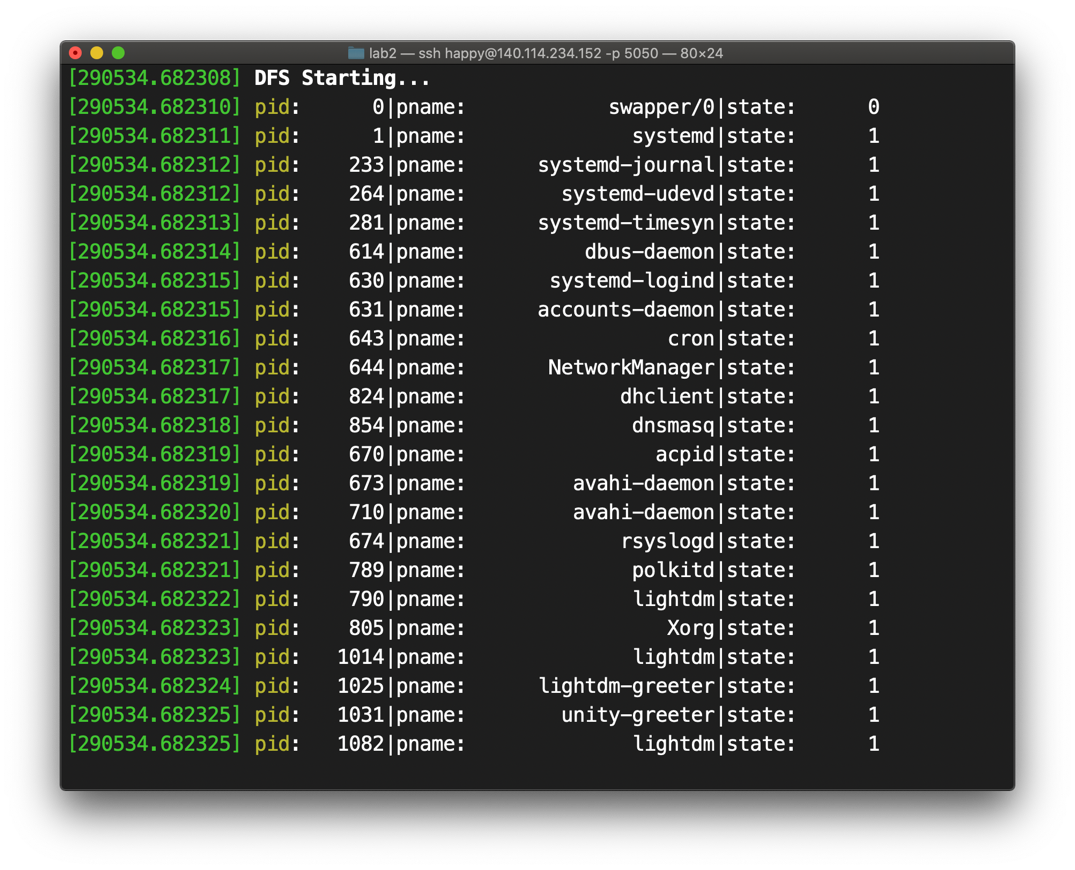
Dfs版本實現的核心就是上面寫的dfs function，在函數裡面對process的children traverse，且print出process的資訊，在執行下一個loop前，會在對當前task\_struct的children執行dfs function，由此達到dfs的目的．

**Result**

1. Linear版本

****

1. Dfs版本

·

**Reference**

[1] The Linux Kernel API (<https://www.kernel.org/doc/htmldocs/kernel-api/Appendix>)

**Appendix**

**#hw\_linear.c**

#include <linux/sched/signal.h>

#include <linux/string.h>

#include <linux/slab.h>

#include <linux/init.h>

#include <linux/module.h>

#include <linux/list.h>

#include <linux/sched.h>

// init function

int hw\_init(void)

{

struct task\_struct \*p;

char p\_buf\_tempalte[] = KERN\_INFO "pid: %6d|pname: %20s|state: %6d\n";

for\_each\_process(p) {

printk(p\_buf\_tempalte, p->pid, p->comm, p->state);

}

return 0;

}

void hw\_exit(void)

{

printk(KERN\_INFO "remove module\n");

}

module\_init(hw\_init);

module\_exit(hw\_exit);

**#hw\_dfs.c**

#include <linux/sched/signal.h>

#include <linux/string.h>

#include <linux/slab.h>

#include <linux/init.h>

#include <linux/module.h>

#include <linux/list.h>

#include <linux/sched.h>

char p\_buf\_template[] = KERN\_INFO "pid: %6d|pname: %20s|state: %6d\n";

void dfs(struct task\_struct \*t\_current)

{

struct task\_struct \*t\_children;

struct list\_head \*l\_children;

list\_for\_each(l\_children, &t\_current->children) {

t\_children = list\_entry(l\_children, struct task\_struct, sibling);

printk(p\_buf\_template, t\_children->pid, t\_children->comm, t\_children->state);

dfs(t\_children);

}

}

// init function

int hw\_init(void)

{

printk("DFS Starting...");

struct task\_struct \*p\_children;

printk(p\_buf\_template, init\_task.pid, init\_task.comm, init\_task.state);

dfs(&init\_task);

return 0;

}

void hw\_exit(void)

{

printk(KERN\_INFO "remove module\n");

}

module\_init(hw\_init);

module\_exit(hw\_exit);