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！

2. dfs 版本

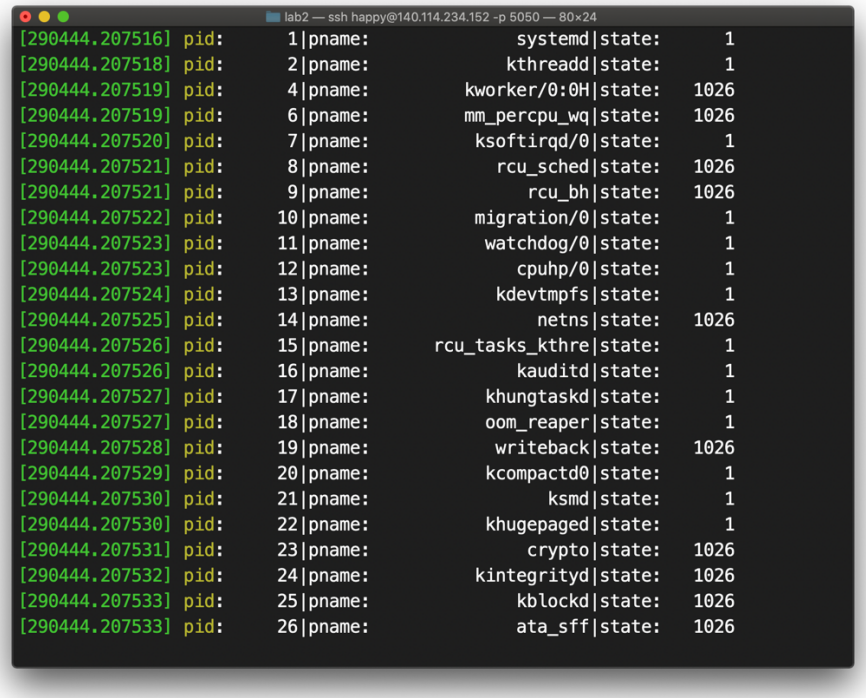
```
void dfs(struct task_struct *t_current)
{
	struct task_struct *t_children;
	struct list_head *l_children;

	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);
	}
}
```

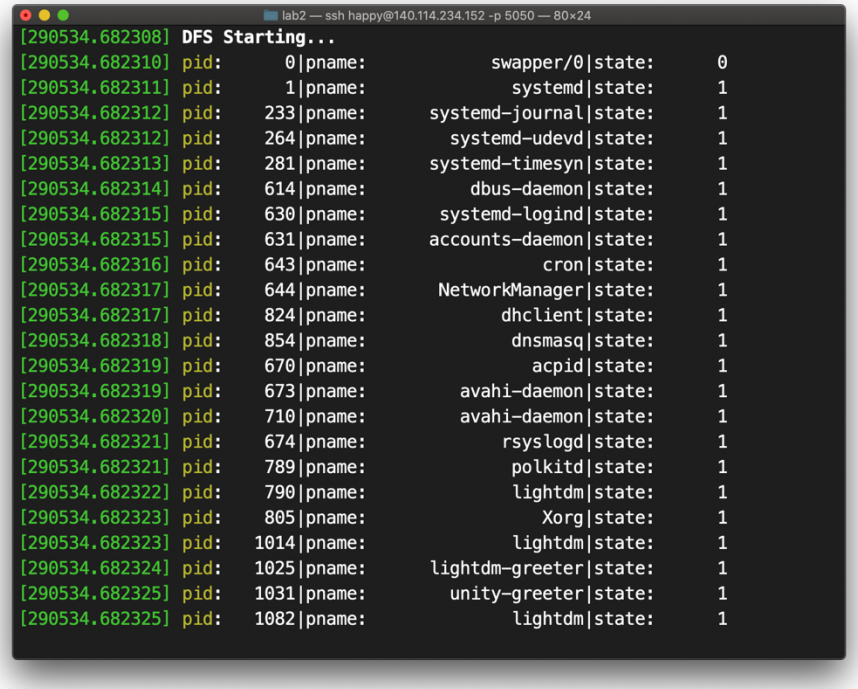
Dfs 版本實現的核心就是上面寫的 dfs function，在函數裡面對 process 的 children traverse，且 print 出 process 的資訊，在執行下一個 loop 前，會在對當前 task_struct 的 children 執行 dfs function，由此達到 dfs 的目的.

Result

1. Linear 版本


lab2 — ssh happy@140.114.234.152 -p 5050 — 80×24
[290444.207516] pid: 1|pname: systemd|state: 1
[290444.207518] pid: 2|pname: kthreadd|state: 1
[290444.207519] pid: 4|pname: kworker/0:0H|state: 1026
[290444.207519] pid: 6|pname: mm_percpu_wq|state: 1026
[290444.207520] pid: 7|pname: ksoftirqd/0|state: 1
[290444.207521] pid: 8|pname: rcu_sched|state: 1026
[290444.207521] pid: 9|pname: rcu_bh|state: 1026
[290444.207522] pid: 10|pname: migration/0|state: 1
[290444.207523] pid: 11|pname: watchdog/0|state: 1
[290444.207523] pid: 12|pname: cpuhp/0|state: 1
[290444.207524] pid: 13|pname: kdevtmpfs|state: 1
[290444.207525] pid: 14|pname: netns|state: 1026
[290444.207526] pid: 15|pname: rcu_tasks_kthre|state: 1
[290444.207526] pid: 16|pname: kauditd|state: 1
[290444.207527] pid: 17|pname: khungtaskd|state: 1
[290444.207527] pid: 18|pname: oom_reaper|state: 1
[290444.207528] pid: 19|pname: writeback|state: 1026
[290444.207529] pid: 20|pname: kcompactd0|state: 1
[290444.207530] pid: 21|pname: ksmd|state: 1
[290444.207530] pid: 22|pname: khugepaged|state: 1
[290444.207531] pid: 23|pname: crypto|state: 1026
[290444.207532] pid: 24|pname: kintegrityd|state: 1026
[290444.207533] pid: 25|pname: kblockd|state: 1026
[290444.207533] pid: 26|pname: ata_sff|state: 1026

2. Dfs 版本


lab2 — ssh happy@140.114.234.152 -p 5050 — 80×24
[290534.682308] DFS Starting...
[290534.682310] pid: 0|pname: swapper/0|state: 0
[290534.682311] pid: 1|pname: systemd|state: 1
[290534.682312] pid: 233|pname: systemd-journal|state: 1
[290534.682312] pid: 264|pname: systemd-udevd|state: 1
[290534.682313] pid: 281|pname: systemd-timesyn|state: 1
[290534.682314] pid: 614|pname: dbus-daemon|state: 1
[290534.682315] pid: 630|pname: systemd-logind|state: 1
[290534.682315] pid: 631|pname: accounts-daemon|state: 1
[290534.682316] pid: 643|pname: cron|state: 1
[290534.682317] pid: 644|pname: NetworkManager|state: 1
[290534.682317] pid: 824|pname: dhclient|state: 1
[290534.682318] pid: 854|pname: dnsmasq|state: 1
[290534.682319] pid: 670|pname: acpid|state: 1
[290534.682319] pid: 673|pname: avahi-daemon|state: 1
[290534.682320] pid: 710|pname: avahi-daemon|state: 1
[290534.682321] pid: 674|pname: rsyslogd|state: 1
[290534.682321] pid: 789|pname: polkitd|state: 1
[290534.682322] pid: 790|pname: lightdm|state: 1
[290534.682323] pid: 805|pname: Xorg|state: 1
[290534.682323] pid: 1014|pname: lightdm|state: 1
[290534.682324] pid: 1025|pname: lightdm-greeter|state: 1
[290534.682325] pid: 1031|pname: unity-greeter|state: 1
[290534.682325] pid: 1082|pname: lightdm|state: 1

-

Reference

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

Appendix

#hw_linear.c

```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

```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);
```