HW4 Linux Semaphore

> 如何編譯：使用 `makefile` 進行編譯
> 主要執行結果

如何編譯

- 由於題目需要編譯多個 c++ 檔案並且進行執行，方便進行編譯是使用

```
TARGETS = p1 p2 p3

all: $(TARGETS) # 執行 p1, p2, p3 target
    ./p1 & ./p2 & ./p3& # 執行命令

p1: p1.c
    gcc -o p1 p1.c sem.c # 執行 p1

p2: p2.c
    gcc -o p2 p2.c sem.c # 執行 p2

p3: p3.c
    gcc -o p3 p3.c sem.c # 執行 p3

clean:
    rm -f $(TARGETS)
```

執行

修改過程

PROF

- 由於必須讓 p1, p2, p3 這三份檔案執行 print 且要依照順序，因此開了 3 個 semaphore 分別是 p1,p2,p3
- 其中必須先讓 `p1.c` 先啟動，所以要先設 p1 semaphore 為 1

```
// *** Please insert proper semaphore initialization here
int semid = create_sem(".", 'A', 2);
int semid_nxt = create_sem(".", 'B', 0);
int semid_nxt2 = create_sem(".", 'C', 0);
int i = 0;
```

- 其中由於 p3.c 需要 print 2 次,同時會啟動兩次 V(p1), 因此卡住 p1.c 需要用 P(p1) 兩次

```
do
{
    // *** this is where you should place semaphore
    P(semid);
    P(semid);
    printf("P1111111111 is here\n");
    i++;

    // *** this is where you should place semaphore
    V(semid_nxt);
} while (i < 100);
```

- 因為 p3.c 要 print 2 次，因此需要對 p3 semaphore 去做 V(p3) 兩次, p2.c

```
int semid_nxt = get_sem(".", 'C');
do
{
    // *** this is where you should place semaphore
    P(semid);
    printf("P222222222 is here\n");
    i++;

    // *** this is where you should place semaphore
    V(semid_nxt);
    V(semid_nxt);
} while (i < 100);
```

- 那 p3.c 因為他要 print 2 次，但在 p2.c 有啟動兩次 V(p3) 因此可以 print 2 次

```
do
{
    // *** this is where you should place semaphore
    P(semid);
    printf("P3333333 is here\n");
    i++;

    // *** this is where you should place semaphore
    V(semid_nxt);
} while (i < 200);
```

- 最後要將 semaphore destory

```
destroy_sem(semid);
destroy_sem(semid_nxt);
destroy_sem(semid_nxt2);
```