

404530030 資管三 林鎰鋒

I. Part.1 – Hw3_1 (my_malloc)

程式執行時，先使用 `malloc()` 配置一段大小為 40MB 的記憶體，再透將這段記憶體每個 Byte 設為 0 的方式去使用這段記憶體，觀察兩者記憶體實際在 OS 下的配置情形。

1. 一開始先使用 `getpid()`取得 process 的 pid，並利用 `htop` 來進行觀察。

```

leon1757@ubuntu: ~$ ./my_malloc
pid = 4348

Tasks: 81, 114 thr: 1 running
Load average: 0.00 0.00 0.00
Uptime: 6 days, 15:19:42

Mem: 175M/7.66G
Swap: 297M/7.63G

PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
4348 leon1757t  20   0   4356    784    708  S   0.0   0.0   0:00.00 ./my_malloc
  
```

2. 先使用 malloc()取得一段 40MB 記憶體

```
leon1757@ubuntu: ~$ ./my_malloc
pid = 4348

malloc memory

1 [ 0.0% Tasks: 81, 114 thr: 1 running
2 [ 0.0% Load average: 0.00 0.00 0.00
3 [ 0.0% Uptime: 6 days, 15:20:45
Mem[ 1.3%
 175M/7.66G
Sup[ 297M/7.63G

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
4348 leon1757t 20 0 45320 784 708 S 0.0 0.0 0:00.00 ./my_malloc
```

可以發現 virtual address(VIRT)增加了 40MB 但 physical address(RES)並沒有增加

3. 再將這段記憶體每個 Byte 寫入 0 的方式去使用這段記憶體後

```
leon1757w@ubuntu:~$  
leon1757w@ubuntu:~/OS_Class/hw3$ ./my_malloc  
pid = 4348  
  
malloc memory  
  
access memory  
  
1 [ 0.7% Tasks: 81, 114 thr; 1 running  
2 [ 0.0% Load average: 0.00 0.00 0.00  
3 [ 0.0% Uptime: 6 days, 15:25:34  
4 [ 1.3%  
Mem[ 215M/7.66G  
Sup[ 297M/7.63G  
  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
4348 leon1757t 20 0 45320 42208 1220 S 0.0 0.5 0:00.20 ./my_malloc  
  
F2help F2Setup F2Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit  
F11Up F12Wall
```

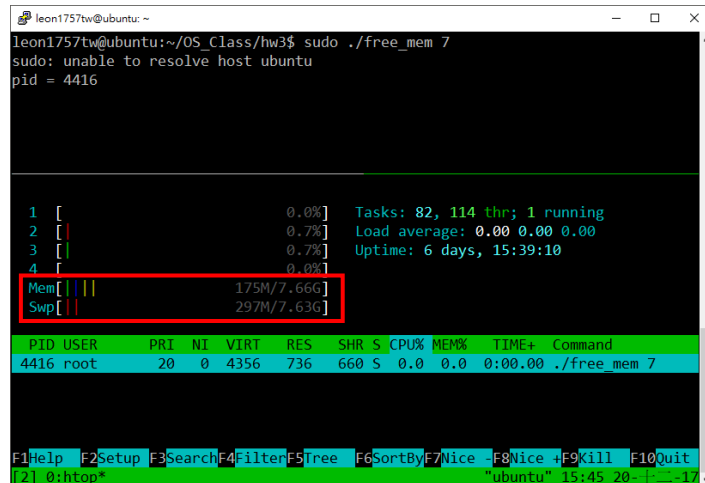
可以發現這時的 physical address(RES)也增加了 40MB

從上述實驗結果可以發現當程式跟 OS 要求配置某個數量的記憶體，實際上 OS 並不會馬上給配置給你，除非真的使用該段記憶體，像上方實驗對記憶體內每一個 byte 進行寫入 0 的方式，OS 才真的將記憶體給程式。

II. Part.2 – Hw3_2 (free_mem)

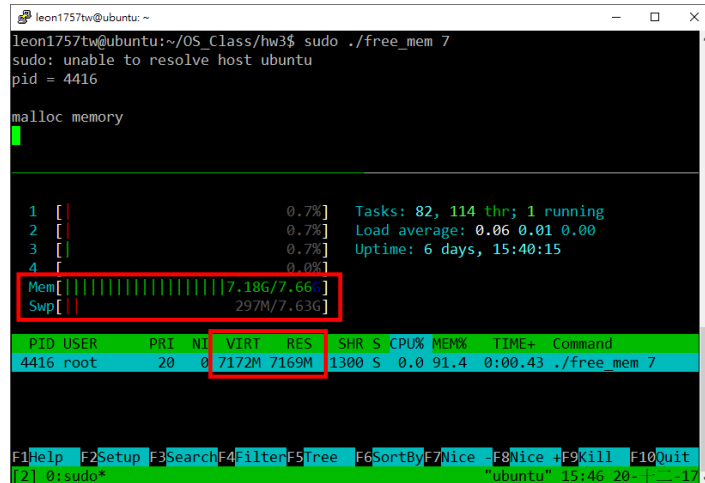
程式使用 `mlockall()` 這個 system call 強制 OS 將其程式使用的記憶體空間鎖在 physical address(RES)中避免被 swap，並根據使用者輸入的值跟 OS 要求大量的記憶體，迫使 system 在記憶體不足時將其他 process swap out 到 swap space，最後再使用 `free()` 使 used memory 變成 free memory，觀察其記憶體與 swap 的使用情形。

- 一開始先使用 `getpid()` 取得 process 的 pid，並利用 `htop` 來進行觀察。



```
leon1757tw@ubuntu: ~  
leon1757tw@ubuntu:~/OS_Class/hw3$ sudo ./free_mem 7  
sudo: unable to resolve host ubuntu  
pid = 4416  
  
1 [ 0.0% Tasks: 82, 114 thr; 1 running  
2 [ 0.7% Load average: 0.00 0.00 0.00  
3 [ 0.7% Uptime: 6 days, 15:39:10  
4 [ 0.0%  
Mem[|||] 175M/7.66G  
Swp[||] 297M/7.63G  
  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
4416 root 20 0 4356 736 660 S 0.0 0.0 0:00.00 ./free_mem 7  
  
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit  
[2] 0:htop* "ubuntu" 15:45 20 - - -17
```

- 直接使用 `malloc()` 根據使用者輸入 `malloc() N(GB)` 的記憶體(圖中為 `malloc 7GB` 的記憶體)



```
leon1757tw@ubuntu: ~  
leon1757tw@ubuntu:~/OS_Class/hw3$ sudo ./free_mem 7  
sudo: unable to resolve host ubuntu  
pid = 4416  
  
malloc memory  
  
1 [ 0.7% Tasks: 82, 114 thr; 1 running  
2 [ 0.7% Load average: 0.06 0.01 0.00  
3 [ 0.7% Uptime: 6 days, 15:40:15  
4 [ 0.0%  
Mem[|||||||||||||] 7.18G/7.66G  
Swp[||] 297M/7.63G  
  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
4416 root 20 0 7172M 7169M 1300 S 0.0 91.4 0:00.43 ./free_mem 7  
  
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit  
[2] 0:sudo* "ubuntu" 15:46 20 - - -17
```

- 可以發現 virtual address(VIRT)與 physical address(RES)同時都增加了約 7GB，與上方實驗不同的是因為使用了 `mlockall()`，所以在 `malloc` 時 OS 就直接配置了 physical address(RES)，所以不用像上方實驗對這段記憶體每個 Byte 寫入 0，執行速度很快。
- 若記憶體時在 `malloc()` 時發生不足，OS 會將其他的 process SWAP OUT，導致 swap space(SWP)的值增加。(此次實驗並沒有發生這種情況)。

