OS 作業三

404410030 資工三 鄭光宇

環境：

（我的 Ubuntu 主機壞掉了…所以我用 Docker+Ubuntu 來完成作業）

編譯：

make

執行：

./my_malloc

./free_mem GB

（要幾 GB 的記憶體）

第一個實驗：

- 實驗的依據：
  - 程式會先呼叫 malloc() 函式請求記憶體，之後 getchar() 等待使用者輸入，此時還未讀寫請求的記憶體，系統還不會分配 40MB 的記憶體給程式，這點可以利用 ps -aux 指令查看 RSS 欄位驗證，會發現 VSZ（虛擬記憶體）很大，但實際要到的 RSS（實際分配的記憶體）很小。
  - 按下 Enter 後，記憶體開始被寫入，RSS 應該會增加到跟 VSZ 接近的大小，也就是系統真的給程式分配記憶體了。
- 程式是否有特別之處：
  - 沒有，就只是先用 getchar() 等待使用者輸入，方便觀察記憶體變化而已。
- 觀察哪些檔案或資訊，以驗證我的論述：
  - 觀察 ps -aux 對我那隻程式回報的 RSS（實際分配的記憶體）變化，來驗證 malloc() 40MB 時，實體記憶體要真正有讀寫，系統才會真的去分配。
- 結果：
  - 呼叫 malloc() 後，寫入記憶體前：

```
ubuntu@0b073a31b87f:~$ ps -aux -q 186
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ubuntu     186  0.0  0.0  45320   632 pts/1    S+   10:22   0:00 ./my_malloc
```

  - 可以看出，RSS 還很小，系統沒有真的分配 40MB
  - 寫入實際寫入記憶體 40MB 後：

```
ubuntu@0b073a31b87f:~$ ps -aux -q 186
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ubuntu     186  1.3  2.0  45320 42192 pts/1    S+   10:22   0:00 ./my_malloc
```

- 可以看出，RSS 變成 40MB 左右了

第二個實驗：

我的 Docker 的記憶體共有 2 GB，SWAP 空間有 3 GB。

- 程式花了多少時間？釋放了多少記憶體？

```
ubuntu@0b073a31b87f:~/OS/hw3$ ./free_mem 4
PID: 86
Press ENTER to start.
Allocated 1 GB of memory!
Allocated 2 GB of memory!
Allocated 3 GB of memory!
Allocated 4 GB of memory!
Allocated 4 GB of memory in 44.63 seconds.
Press ENTER to release memory.
```

- 程式花了 44.63 秒，分配了 4 GB 的記憶體，因為要分配 4 GB 的記憶體給這隻程式，而實際記憶體大小為 2 GB，2GB < 4GB，不夠放，所以原來在記憶體中的 Process 會被擠到 SWAP 裡，還有把 buffer/cache 從記憶體擠出去。
- 從 free -h 指令觀察，前後共釋放了約 1.9 GB 的記憶體。過程如下圖：

```
ubuntu@0b073a31b87f:~/OS/hw3$ free -h
              total        used        free      shared  buff/cache   available
Mem:           2.0G         78M         72M        162M        1.8G        1.5G
Swap:          3.0G        1.9M        3.0G
```

（剛開始的樣子）

```
ubuntu@0b073a31b87f:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           2.0G        1.8G         75M        3.7M         36M         10M
Swap:          3.0G        2.4G        612M
```

（程式吃光了所有記憶體，把其他程序擠到 SWAP 裡、把 buffer/cache 擠出去）

```
ubuntu@0b073a31b87f:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           2.0G         37M        1.9G        4.4M         43M        1.8G
Swap:          3.0G        203M        2.8G
```

（可以看到可用記憶體從 72M 變成 1.9G，我們得到了約 1.9G 的 free memory）

- 說明程式對 swap space 的影響，如何觀察？
  - 利用 free -h，觀察 swap space 的變化。
  - 從上面三張圖可以看出，swap 原來只有使用 1.9 MB，執行./free_mem 時，因為記憶體不夠用，其他 process 被擠到 swap，swap 使用了 2.4

GB（./free_mem+其他 process），程式結束後，swap 使用了 203 MB。