404410016 資工三 宋慶澔

作業系統作業三

題目一

第一部分

- 設計一個實驗，了解作業系統配置記憶體的方法
- 經由這個實驗回答：malloc 40MB 時，作業系統是否會立刻配置 40MB 的記憶體給該程式
- 了解作業系統是需求分頁，當你跟作業系統要求配置某個數量的記憶體，實際上作業系統不會馬上給你，除非真的使用該快記憶體，例如對記憶體內每一個 byte 進行讀寫

- 第一個實驗
  - 說明你實驗的依據，
  - 說明你所設計的程式是否有特別之處
  - 你觀察 Linux 中那些檔案或者資訊，以驗證你的論述

答案一

1. 程式一開始會呼叫 **malloc()** 要求配置一塊 40 MB 大小的記憶體，再使用這塊記憶體之前先*透過 **free** 觀察該 process 的記憶體使用情況。*

```
int *p = (int *) malloc(sizeof(int) * 1000000);
```

```
system("free");
```

2. 程式會呼叫 **memset()** 將這塊記憶體填入初始值 1，再*使用 **free** 觀測該 process 的記憶體使用狀況*

```
memset(p, 1, sizeof(int) * 1000000);
```

```
system("free");
```

3. 程式呼叫 **free()** 把動態產生的記憶體釋放，結束程式。

```
free(p);
```

結論：

在真正使用宣告的記憶體之前，系統並不會真的配置一塊 40 MB 的記憶體給該 process。透過 **free** 觀測行程之記憶體使用狀況如下圖所示：

```
howard@howard-PE70-2QE: ~/SystemProgramming/aboutMemory
howard@howard-PE70-2QE:~/SystemProgramming/aboutMemory$ make
gcc my_malloc.c -o my_malloc
howard@howard-PE70-2QE:~/SystemProgramming/aboutMemory$ ./my_malloc
Before assignments (in KBs)
              total        used        free      shared  buff/cache   available
Mem:        8084280     2063188     4336528      489736     1684564     5219764
Swap:       7905276           0     7905276
After assignments (in KBs)
              total        used        free      shared  buff/cache   available
Mem:        8084280     2102628     4297088      489736     1684564     5180324
Swap:       7905276           0     7905276
howard@howard-PE70-2QE:~/SystemProgramming/aboutMemory$
```

可以清楚看到程式在assignments之後才把記憶體給了該process，used memory 從原來的 2063.18MB 變成了 2102.62 MB，相差了 40 MB。

題目二

第二部分

- 撰寫一支程式可要求大量的 memory，迫使 system 做 swap，之後使用 free()使 used memory 變成 free memory，由於曾經 swap 過因此最後的 free memory 會比之前多。
- 了解作業系統是以全域的方式配置記憶體，當跟系統要很多記憶體，就會將其他 process 的記憶體『擠出去』到 swap space
- 可輸入一個參數 int，表示要求多少 GB 的 memory。例如：
  ./free_mem 2 ，表示 2*1024*1024*1024 byte。
- 必須提供觀察方式，但不可使用 GUI、系統監控（相當於 Windows 的系統管理員）。

- 第二個實驗
  - 說明你的程式花費了多少時間，釋放了多少記憶體
  - 說明你的 free_mem 對於 swap space 的影響（即：是否額外造成 page-out 及 page-in），如何觀察？

答案二

1. 使用者從 argv 給 x，x 為要 malloc 產生的記憶體大小(單位：GB)
   /* ps：建議 range 在 1 ≦ X ≦ 5，超過可能會死當 */
2. 程式一開始會先用透過 free 指令來觀測當前記憶體使用狀況。
3. 接著會 1 GB 1 GB 的透過 malloc 宣告記憶體，並且對這塊記憶體做寫入。直到達到 x GB為止。
4. 程式接著會再一次透過 free 指令來觀測當前記憶體使用狀況
5. 接著程式會將剛剛動態產生的 memory 通通釋放。
6. 再次呼叫 free 來觀測記憶體使用狀況，結束程式。

```
howard@howard-PE70-2QE: ~/SystemProgramming/aboutMemory
howard@howard-PE70-2QE:~/SystemProgramming/aboutMemory$ time ./free_mem 5
Before malloc:
               total        used        free      shared  buff/cache   available
Mem:            7.7G        1.9G        5.0G        299M        745M        5.2G
Swap:           7.5G        359M        7.2G

After malloc:
               total        used        free      shared  buff/cache   available
Mem:            7.7G        6.9G        174M        296M        664M        252M
Swap:           7.5G        446M        7.1G

After free:
               total        used        free      shared  buff/cache   available
Mem:            7.7G        1.9G        5.2G        296M        663M        5.3G
Swap:           7.5G        446M        7.1G


real    0m0.985s
user    0m0.192s
sys     0m0.772s
howard@howard-PE70-2QE:~/SystemProgramming/aboutMemory$
```

結論：

上圖中，程式總共透過 malloc() 要求配置了 5GB的空間，在0.985秒內完成所有動作，可以看見在程式結束完成記憶體釋放之後，free memory 從原來的 5.0 GB 變成 5.2 GB 。swap 也從原來的 359 MB 變大成 446 MB。*結論是，free_mem 會對 swap 造成額外的 page in / page out。*