第一部分

* 1. 我malloc()要了40MB的記憶體。實驗是利用開htop觀察的，可以明顯看到系統的動作。
  2. code:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define N (40 \* 1024 \* 1024) // 40 MB

int main()

{

srand(time(NULL));

int n, \*data = malloc(N); // 要40 MB，倒數第三的參數是MEM%記憶體配給量，目前是0.0，代表沒有用之前系統並不會給我們記憶體。



scanf("%d", &n);

for (int i = 0; i < N / 4; i++)

data[i] = rand(); // 開始用了，系統就給我們記憶體了（0.0 -> 2.1）



int idx = rand() % N;

printf("data[%d] = %d\n", idx, data[idx]);

scanf("%d", &n);

free(data);

return 0;

}

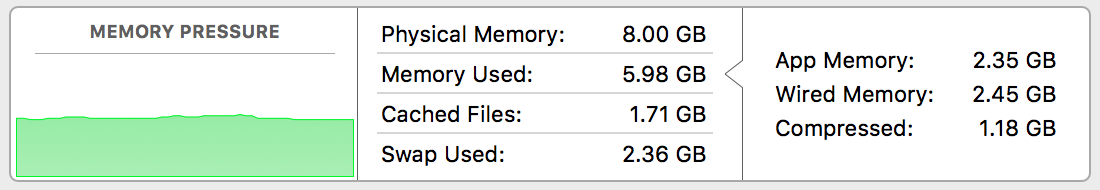
第二部分

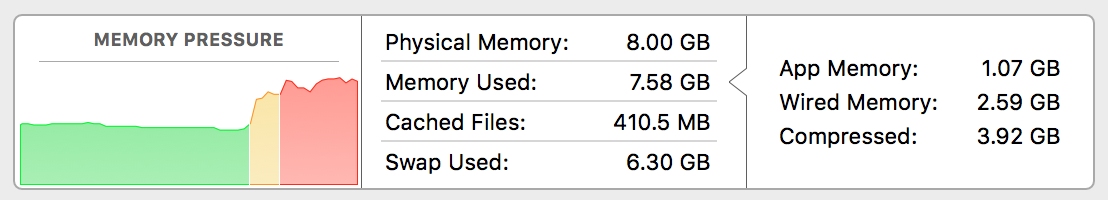
我的做法是我跟電腦malloc() 20次的380 MB，並在每次要完後對裡面進行讀寫。基本上程式運行中，會發現main memory一直被吃掉，swap space用得越來越多。運行完後，main memory會比程式運行前多出一些記憶體。

實驗結果如下：

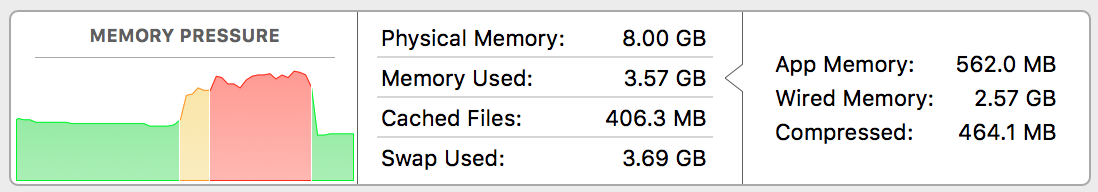
我在自己的電腦上跑，整體執行起來大約花30秒鐘。

第一張圖說明原本OS正用掉5.98GB的main memory。



第二張圖說明程式運行中會造成main memory幾乎被吃光（8GB用了7.58GB）

第三張圖說明，程式結束後，main memory會比原本還多出一些空間來（5.98GB -> 3.57GB，多出了2.41GB，然後swap 的使用量有上升一些）。



#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define N (int)1e8

#define M 20

typedef long long ll;

int main()

{

srand(time(NULL));

int \*ptr[M];

for(int i = 0; i < M; i++) {

printf("Now on %d\n", i);

ptr[i] = malloc(N \* sizeof(int));

if (ptr[i] == NULL) {

break;

}

for(int j = 0; j < N; j++)

ptr[i][j] = rand() % N;

}

for(int i = 0; i < M; i++) {

free(ptr[i]);

}

return 0;

}