*摘要*

也許你很少遇到這種情況，但是當你遇到的時候就會知道哪裡出錯了。

結果很簡單，linux的內存用完了，無法申請緩衝區，內核會挑選進程將其殺死，一般情況下，殺死正在申請內存的程序。頻繁的進行磁盤swap操作，經常會出現這類問題，或是並發處理時啟動的進程數過多。

出現內存耗盡的原因很簡單，你申請的內存大小，超過了可用的虛擬內存的大小，注意時虛擬內存（內存並不是唯一的，交換分區也可以提供內存）

## 探究pom（out of memory）

首先運行下面的程序，不斷申請大量內存：

```
#include <stdio.h>
#include <stdlib.h>

#define MEGABYTE 1024*1024

int main(int argc, char *argv[])
{
        void *myblock = NULL;
        int count = 0;

        while (1)
        {
                myblock = (void *) malloc(MEGABYTE);
                if (!myblock) break;
                printf("Currently allocating %d MB\n", ++count);
        }

        exit(0);
}
```

上面的程序運行一會就會出現oom，現在運行另外一個程序，不斷申請內存，並且將其

填充1.

```
#include <stdio.h>
#include <stdlib.h>

#define MEGABYTE 1024*1024

int main(int argc, char *argv[])
{
        void *myblock = NULL;
        int count = 0;

        while(1)
        {
                myblock = (void *) malloc(MEGABYTE);
                if (!myblock) break;
                memset(myblock,1, MEGABYTE);
                printf("Currently allocating %d MB\n",++count);
        }
        exit(0);

}
```

有發現不同嗎，事實上程序1可以比程序2申請更多的內存。兩個程序退出的原因都是因為空間不夠了，然而程序1的退出時因為malloc的失敗，而程序2的退出則是因為內核所謂的oom killer 將其殺死了。

程序2退出的時候：

Currently allocatinn 1589 MB

程序1退出的時候：

Currently allocating 274520 MB(64位系統）

為什麼程序1相較程序2可以多分配如此多的內存，這是因為linux採用了延遲的頁面分配。也就是說內存只有在真正用的時候才進行分配，這種技術被稱為*optimistic memory*

*allocation*。

查看／proc／pid／status文件就可以知道這個情況。

首先是程序1:

```
VmPeak: 277725508 kB
VmSize: 277725508 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:    541980 kB
VmRSS:    284004 kB
VmData: 277721404 kB
VmStk:       136 kB
VmExe:         4 kB
VmLib:      1876 kB
VmPTE:    542452 kB
VmSwap:   799996 kB
```

然後是程序2:

```
VmPeak:  1620168 kB
VmSize:  1620168 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:    823432 kB
VmRSS:    814608 kB
VmData:  1616064 kB
VmStk:       136 kB
VmExe:         4 kB
VmLib:      1876 kB
VmPTE:      3180 kB
VmSwap:   801500 kB
```

當我們請求一個內存區時，c庫會判斷當前預分配的內存塊是否足夠大，如果不夠，程序會通過擴展堆空間的方式來獲取內存。

查看文件／proc/pid／maps可以看到堆裡的內存塊。