

(針對公告的各項分別做說明)

### Lab 3 Memory Management

#### <Use bitmap>

我的 bitmap 設計想法是，由於使用 BIOS 中斷服務，最少是讀取一個 sector，也就是 512bytes，所以我的 bitmap 每一格是對應到 512bytes，而 512bytes 在十六進位就是 200，放在 segment 就是每次增加 20，而我給 bitmap 管的記憶體起點是 0x0100，所以管理的範圍是 0x0100:0x0000 ~ 0x02E0:0x0000，畫一個表格來示意

Bitmap	Memory
0	0x0100:0x0000
1	0x0120:0x0000
2	0x0140:0x0000
3	0x0160:0x0000
4	0x0180:0x0000
5	0x01A0:0x0000
6	0x01C0:0x0000
7	0x01E0:0x0000
8	0x0200:0x0000
9	0x0220:0x0000
10	0x0240:0x0000
11	0x0260:0x0000
12	0x0280:0x0000
13	0x02A0:0x0000
14	0x02C0:0x0000
15	0x02E0:0x0000

#### <Memory Allocation/De-allocation>

有 Dump 指令可以顯示目前 memory 的使用狀態，也可以使用 alloc 指令來佔用 bucket，也可以使用 dealloc 來取消佔用，如下圖

```
Bochs for Windows - Display
[bootloader] Load scheduler on the sector (0,0,2~3) to 0600:0000
[bootloader] Load done.
[bootloader] Hook INT13C
[bootloader] Run scheduler...

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
Yoshi:>alloc 0 5
Mark 0 to 4 as used

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| * | * | * | * | * |   |   |   |   |   |   |   |   |   |   |   |
Yoshi:>dealloc 2 2
Mark 2 to 3 as unused

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| * | * |   |   | * |   |   |   |   |   |   |   |   |   |   |   |
Yoshi:>
```

Shell 也可以 load 程式，自己決定要分配多少的 bucket 給 application。這個部份我有做 Bonus，做法是寫一些 header information 在 application 裡，而 shell 會先去 header 看檔案大小，然後再以 First-Fit 的方式去幫它配置空間，若無足夠的連續空間(External Fragmentation)，或是已經沒有空間可以用了，就會出現 No enough contiguous space 的錯誤訊息，如下

```
Bochs for Windows - Display

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
Yoshi:>alloc 1 14
Mark 1 to 14 as used

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|   | * | * | * | * | * | * | * | * | * | * | * | * | * | * |   |
Yoshi:>run 12
filesize is 256 bytes, need 1 bucket(s)
Allocate 0 to 0 for application
Load to 0x100:0
[HELLO1] HELLO WORLD 1
Release memory from 0 to 0 for application

Yoshi:>run 13
filesize is 768 bytes, need 2 bucket(s)
No enough free contiguous space.

Yoshi:>
```

Shell 若執行的是前景程式，則會在 application 執行完後自動 release 使用的記憶體空間，而如果跑的是背景程式，則不會自動釋放，這個地方原本我打算在 header information 加入，讓 application 可以告訴 shell 自己是前景還是背景，但是後來想想這樣 header information 就要越來越多，所以不使用這個做法，而是在 shell 提供二種執行程式的指令，執行前景程式時，就以 run [sector]來執行，則 shell 會在執行完後自動釋放，而要執行背景程式就使用 runb [sector]來執行，則 shell 就不會自動釋放，如下

```

Bochs for Windows - Display
Options: apmbios pcibios eltorito

Booting from Floppy...
[bootloader] Load shell on the sector (0,0,4~9) to 0800:0000
[bootloader] Load scheduler on the sector (0,0,2~3) to 0600:0000
[bootloader] Load done.
[bootloader] Hook INT13C
[bootloader] Run scheduler...

Yoshi:>runb 10
filesize is 1040 bytes, need 3 bucket(s)
Allocate 0 to 2 for application
Load to 0x100:0
Background program, does not release memory automatically.

Yoshi:>run 12
filesize is 256 bytes, need 1 bucket(s)
Allocate 3 to 3 for application
Load to 0x160:0
[HELLO1] HELLO WORLD 1
Release memory from 3 to 3 for application

Yoshi:>
CTRL + 3rd button enables mouse
A. NUM CAPS SCRL

```

前面的說明是介紹我的程式表面，證明我有做到 requirement，接下來我要介紹我的程式架構與做法。

## <磁碟使用分佈>

首先是我的磁碟分佈

C = 0, H = 0,

Sector	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		Bootloader	Scheduler	Shell							5566	7788	Hello1(B)	Hello2(B)

(B) = 背景程式

(F) = 前景程式

5566's file size in header = 1040bytes

7788's file size in header = 512bytes

Hello1's file size in header = 256bytes

Hello2's file size in header = 768bytes

(註:上述這四個程式在 header 有註明自己的檔案大小，但事實上檔案實際可能並沒有那麼大，所以還是只要放在一個 sector 就可以了)

### <記憶體配置>

Scheduler → 0x0600:0x0000

Shell → 0x0800:0x0000

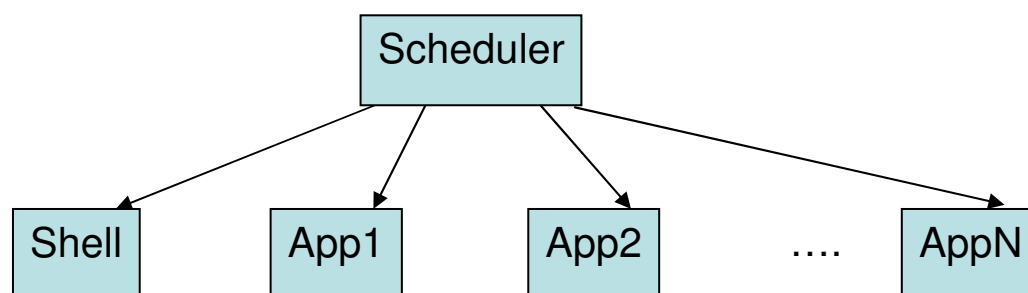
5566/7788 → 不停修改 0x0900:0x0000 / 0x0900:0x0002

0x100:0x0000 ~ 0x2E0:0x0000 → 供 loader 動態配置記憶體

### <Source Code>

- S\_XXXX 代表這是 XXXX.c 的 entry。如 S\_Shell 為 shell.c 的 entry。
- libXXX 代表這是 XXXX.c 使用的組語 library，如 libshell.asm 就是給 shell.c 用的組語 function。
- Sche 即為 Scheduler
- 5566.asm, 7788.asm 為背景程式
- Hello1.asm, Hello2.asm 為前景程式
- 

### <Scheduler>



基本上整個程式最重要的核心就是 Scheduler，Scheduler 以 Round-Robin 的方式為 Process 做排班，而 Scheduler 與 Shell 之間會以 share-memory 方式來做溝通，shell 會告知 scheduler 要加入新的 process 排班或是停止排班一個 process。而記憶體的管理完全是做在 Shell 裡，所以也就是 Scheduler 管排班，Shell 管 UI，Memory Management。

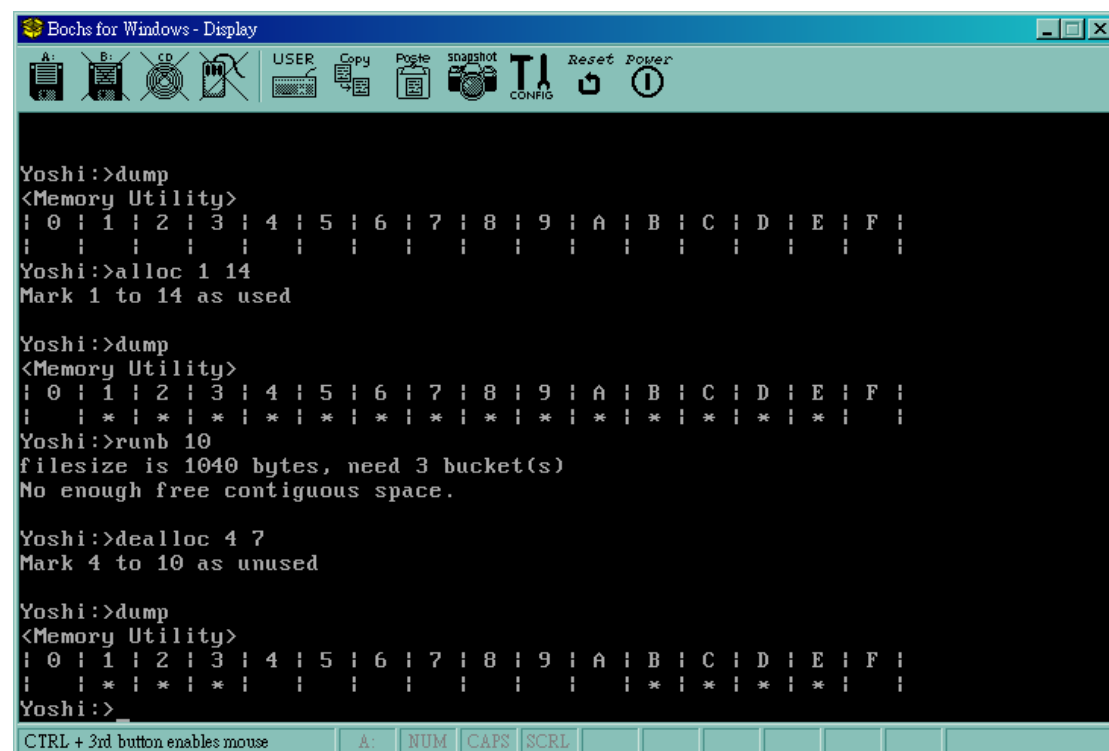
Scheduler 是我註冊在 INT 1Ch 的中斷服務，所以是以 timer-interrupt 的方式驅動，所以在 coding 時就要小心任何時候都有可能發生 context-switch，我覺得這個比使用 yield 的方式要多考慮一些情況，也比較難 coding，但在 multi-tasking 的效果我覺得比較好，所以我還是選擇這個方式。

前面有提到跑前景程式使用 run，而跑背景使用 runb，這其中的差異只是 shell

要不要幫 process 在結束後釋放 memory，其實也可以做一個 EXIT function 來給前景程式 call，原本有想要做，但是後來想想下一個 Lab 可能還要修改到這個部份，所以我這個地方就暫時以這種方式來做。

這個作業我覺得自己的完成度算很高，投影片上的要求應該都有做到，下面就 demo 我的執行畫面。

### <Running Screen>



```
Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
Yoshi:>alloc 1 14
Mark 1 to 14 as used

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|   | * | * | * | * | * | * | * | * | * | * | * | * | * | * |   |
Yoshi:>runb 10
filesize is 1040 bytes, need 3 bucket(s)
No enough free contiguous space.

Yoshi:>dealloc 4 7
Mark 4 to 10 as unused

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|   | * | * | * |   |   |   |   |   |   |   | * | * | * | * |   |
Yoshi:>
```

1. 一開始做 dump，memory 都是 free 的
2. allocate 1 14，所以被佔用的打上了\*字號
3. 執行 sector 10，是一個背景程式，因為這個程式 1040bytes，要三個 bucket，所以顯示沒有足夠的連續空間
4. de-allocate 4~7 於是 4 ~ A 就空出來了

```

Bochs for Windows - Display
A: B: CD USER Copy Paste snapshot T! Reset Power
CONFIG

| | * | * | * | * | | | | | * | * | * | * |
Yoshi:>runb 10
filesize is 1040 bytes, need 3 bucket(s)
Allocate 4 to 6 for application
Load to 0x180:0
Background program, does not release memory automatically.

Yoshi:>dump
<Memory Utility>
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| | * | * | * | * | * | * | | | | | * | * | * | * | |
Yoshi:>runb 11
filesize is 512 bytes, need 1 bucket(s)
Allocate 0 to 0 for application
Load to 0x100:0
Background program, does not release memory automatically.

Yoshi:>run 12
filesize is 256 bytes, need 1 bucket(s)
Allocate 7 to 7 for application
Load to 0x1E0:0
[HELLO1] HELLO WORLD 1
Release memory from 7 to 7 for application

Yoshi:>_
CTRL + 3rd button enables mouse  A: NUM CAPS SCRL

```

1. 現在有足夠空間執行 sector 10 的背景程式了，shell(loader)選擇了 First-Fit 的 4~6 Bucket 供其使用
2. 接著執行 sector 11 的背景程式，佔用一個 bucket
3. 執行 Hello1，前景程式，執行完後自動釋放配置的記憶體(bucket 7)

```

Bochs for Windows - Display
A: B: CD USER Copy Paste snapshot T! Reset Power
CONFIG

Allocate 7 to 7 for application
Load to 0x1E0:0
[HELLO1] HELLO WORLD 1
Release memory from 7 to 7 for application

Yoshi:>run 13
filesize is 768 bytes, need 2 bucket(s)
Allocate 7 to 8 for application
Load to 0x1E0:0
[HELLO2] HELLO WORLD 2
Release memory from 7 to 8 for application

Yoshi:>u 0x900:0x0
0x900:0x0 0x1155

Yoshi:>u 0x900:0x0
0x900:0x0 0xF3D5

Yoshi:>u 0x900:0x2
0x900:0x2 0x2A01

Yoshi:>u 0x900:0x2
0x900:0x2 0x5B1E

Yoshi:>_
CTRL + 3rd button enables mouse  A: NUM CAPS SCRL

```

1. 執行 Hello2，檔案大小 768bytes，佔用二個 bucket，所以選了 7,8 給它用
2. 之前有執行背景程式在執行，背景程式分別會修改 0x900:0x0 和 0x900:0x2

3. 二次 dump memory 同樣的位置，數值都有在改變，證明背景程式在執行

### <心得>

這個程式一開始要先把流程規畫好，如果到後面才分別考量各個 requirement 會有很大的麻煩，而這次我是第一次用組語+C 來寫，所以除了 bootloader 之外，可以說是全部都重寫了，所以花了我很多心力，特別是在 Scheduler 的核心和 Shell 的處理，很多小地方要注意，到後來大致完成後，開始美化我的介面，加入 16 進位轉換，dump memory 的輸出...等，而使用 timer interrupt 讓我常常腦筋打結，整個程式隨時都可能 context-switch 讓我想了很久才開始動手。我覺得這個作業我收穫很多。