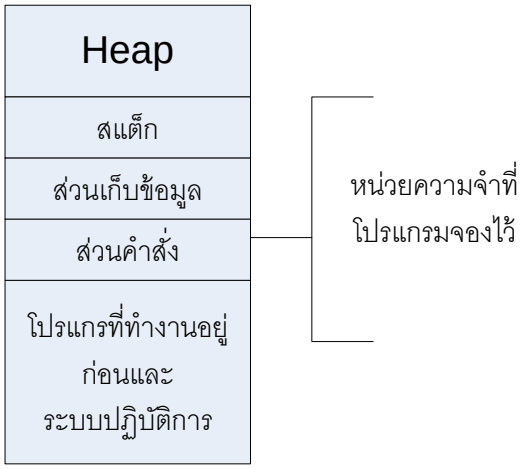
Lab 1 การจัดการหน่วยความจำ

เมื่อผู้เขียนโปรแกรมทั่วไปสั่งให้โปรแกรมทำงาน นั่นหมายถึง จะมีพื้นที่ส่วนหนึ่งในหน่วยความจำถูกจองเพิ่มขึ้น หน่วยความจำส่วนนี้จะถูกคืนให้ระบบเมื่อโปรแกรมทำงานเสร็จสิ้น โครงสร้างของหน่วยความจำส่วนที่จองให้โปรแกรมนี้อาจแบ่งออกได้ดังรูป



เมื่อมีการเรียกใช้โปรแกรม ระบบปฏิบัติการจะจองพื้นที่สำหรับตัวโปรแกรม อ่านโปรแกรมขึ้นมาจากหน่วยความจำสำรอง และจัดการปรับโปรแกรมให้สามารถทำงานได้ ที่สุดก็จะโอนการทำงานไปยังโปรแกรม เมื่อโปรแกรมทำงานเสร็จสิ้น ก็จะคืนพื้นที่ที่ใช้งานให้แก่ระบบ และโอนการทำงานให้แก่ระบบปฏิบัติการ

การจัดการหน่วยความจำแบบสแตติก (Static Memory Management) คือการจัดการพื้นที่หน่วยความจำสำหรับใช้งานในโปรแกรมที่จะมีการจองพื้นที่สำหรับการเก็บข้อมูลหรือตัวคำสั่ง ตั้งแต่โปรแกรมเริ่มทำงาน และจะคืนพื้นที่ส่วนดังกล่าวเมื่อโปรแกรมทำงานเสร็จสิ้นแล้ว

โครงสร้างข้อมูลสแตติก (Static Data Structure) คือตัวแปรเดี่ยว หรือตัวแปรกลุ่มที่มีการจองพื้นที่สำหรับใช้เก็บค่าตั้งแต่โปรแกรมเริ่มทำงาน และจะถูกส่งคืนให้แก่ระบบเมื่อโปรแกรมทำงานเสร็จสิ้น

การจัดการหน่วยความจำแบบไดนามิก (Dynamic Memory Management) คือการจัดการพื้นที่หน่วยความจำสำหรับใช้งานในโปรแกรมเพื่อการเก็บข้อมูลหรือคำสั่ง โดยจะจองพื้นที่ที่ต้องการเมื่อมีความต้องการจะใช้พื้นที่ในการเก็บข้อมูล และเมื่อไม่ต้องการใช้พื้นที่นั้นแล้ว ก็จะคืนพื้นที่นั้น ๆ ให้แก่ระบบได้ทันที (หรืออาจจะคืนในภายหลังก็ได้) นั่นหมายถึง พื้นที่ที่ใช้เก็บข้อมูลไม่จำเป็นจะต้องจองเมื่อโปรแกรมเริ่มทำงาน และไม่จำเป็นต้องคืนในขณะที่โปรแกรมทำงานเสร็จสิ้น

โครงสร้างข้อมูลไดนามิก (Dynamic Data Structure) คือตัวแปรเดี่ยว หรือตัวแปรกลุ่มที่มีการจองพื้นที่สำหรับใช้เก็บค่าเมื่อมีความต้องการตัวแปรหรือตัวแปรกลุ่ม และจะคืนพื้นที่ให้แก่ระบบเมื่อไม่ต้องการตัวแปร หรือตัวแปรกลุ่มนั้น ๆ แล้ว

การจัดการหน่วยความจำแบบไดนามิก

ในการจัดการโครงสร้างข้อมูลไดนามิกนั้น พื้นที่ที่จะใช้เก็บข้อมูลจะยังไม่ถูกจองเมื่อโปรแกรมเริ่มทำงานเหมือนกับโครงสร้างข้อมูลสแตติก ดังนั้นจึงเป็นหน้าที่ของผู้เขียนโปรแกรม ที่จะต้องใส่ฟังก์ชันสำหรับจองหน่วยความจำก่อนที่จะเขียนโปรแกรมติดต่อกับโครงสร้างข้อมูลไดนามิก และเมื่อใช้งานเสร็จสิ้นก็จะเป็นหน้าที่ของผู้เขียนโปรแกรม ที่จะต้องคืนหน่วยความจำเหล่านั้นให้ระบบด้วย ลำดับการจัดการหน่วยความจำแบบไดนามิก จึงอาจเขียนเป็นขั้นตอนได้ดังนี้

1. **จองหน่วยความจำ** เราจะกำหนดจำนวนพื้นที่ที่ต้องการสำหรับข้อมูลหนึ่งหน่วย แล้วจองพื้นที่จากระบบตามจำนวนที่ต้องการ ค่าที่ได้จากระบบก็คือค่าหน่วยความจำเริ่มต้นสำหรับการใช้งาน ค่าดังกล่าวเราจะเก็บไว้ในตัวชี้ข้อมูลชนิดที่สอดคล้องกับโครงสร้างข้อมูลไดนามิกที่จองนั้น สำหรับใช้อ้างตำแหน่งเริ่มต้นเพื่อใช้งานต่อไป

2. **ใช้หน่วยความจำ** เราจะติดต่อกับพื้นที่ที่ต้องการโดยอาศัยตัวชี้ไปยังจุดเริ่มต้นหน่วยข้อมูลนี้ นอกจากนั้น เราอาจกำหนดตัวชี้ขึ้นใหม่ เพื่อใช้ในการทำงานอีกก็ได้

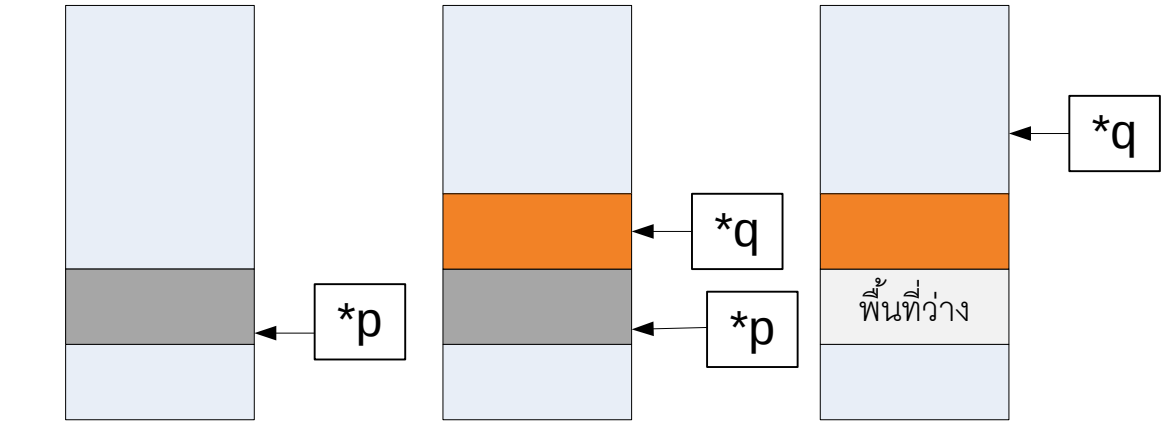
3. **คืนหน่วยความจำ** หลังจากใช้งานเสร็จสิ้นแล้ว เราจะคืนหน่วยความจำที่ได้จองไว้แก่ระบบ โดยอาศัยค่าการชี้ที่ได้จากการจองหน่วยความจำนั่นเอง

การจองหน่วยความจำ เราจะใช้ฟังก์ชั่น malloc() ดังตัวอย่าง

```
char *p;
.....
p = (char *)malloc(16);
```

คำสั่งแรกเป็น คำสั่งจองพื้นที่หน่วยความจำขนาด 16 ไบต์ และให้ตัวชี้ p เป็นผู้ชี้ตำแหน่งเริ่มต้นของหน่วยความจำที่จองมาได้ สังเกตว่าเราใช้ (char *) เพื่อเปลี่ยนชนิดของค่าการชี้จาก void ให้เป็นค่าการชี้ข้อมูลชนิด char

การจองพื้นที่หน่วยความจำ จะได้พื้นที่มาจากส่วน Heap ซึ่งเป็นพื้นที่ที่เหลือจากการใช้งานของโปรแกรม ในการทำงานจริง พื้นที่ส่วน Heap นี้อาจไม่ได้อยู่ติดกันเป็นส่วนเดียว แต่อาจจะกระจายแทรกอยู่ระหว่างพื้นที่ที่ใช้งานก็ได้ ซึ่งเกิดจากการคืนหน่วยความจำบางส่วนให้ระบบไปก่อน ดังรูป



คำสั่ง coreleft() ใช้รายงาน Heap ที่เหลือที่โปรแกรมสามารถขอใช้ได้ มีรูปแบบ ดังนี้

```
printf("System memory left %lu bytes\n",coreleft();
```

ส่วนคำสั่ง farcorleft() ใช้รายงานหน่วยความจำของระบบที่เหลือทั้งหมด มีรูปแบบ ดังนี้

```
printf("System memory left %lu bytes\n",farcoreleft();
```

คำสั่ง farfree() และ free() คือฟังก์ชันสำหรับคืนหน่วยความจำให้แก่ระบบ มีรูปแบบดังนี้

```
free((void *)p);
farfree((void far *)q);
```

ตัวอย่าง 1.1 เป็นการทดลองจองพื้นที่ขนาด 80 ไบต์ ให้กับตัวชี้ char ซึ่งสามารถใช้ได้ในรูปของอาร์เรย์ของตัวอักษรหรือตัวแปรสตริง และเมื่อใช้งานเสร็จสิ้นแล้ว ก็จะคืนพื้นที่ที่จองให้แก่ระบบ

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

void main()
{
        /*Declare pointer*/
        char *p;

        /*Report heap left*/
        printf("Memory left %lu\n",coreleft());

        /*Allocate memory for 80 bytes */
        p = (char *)malloc(80);

        /*Report heap after allocate memory */
        printf("Memory left %lu\n",coreleft());

        printf("Please input string : ");
        /*Use as static data structure*/
        gets(p);
        printf("String is :%s\n",p);

        /*Free memory */
        free((void *)p);

        /*Report heap after use*/
        printf("Memory left %lu\n",coreleft());
}
```

จงเขียนโปรแกรมโดยดัดแปลงจากตัวอย่างที่ 1.1 เก็บข้อมูลในอาร์เรย์ขนาด 5 คูณ 5 โดยรับค่าเป็น char ทางคีย์บอร์ดเพื่อใส่ค่าในอาร์เรย์ แสดงข้อมูลหน่วยความจำโดยใช้คำสั่ง coreleft() และ farcorleft() ทั้งก่อนและหลังใส่ข้อมูล