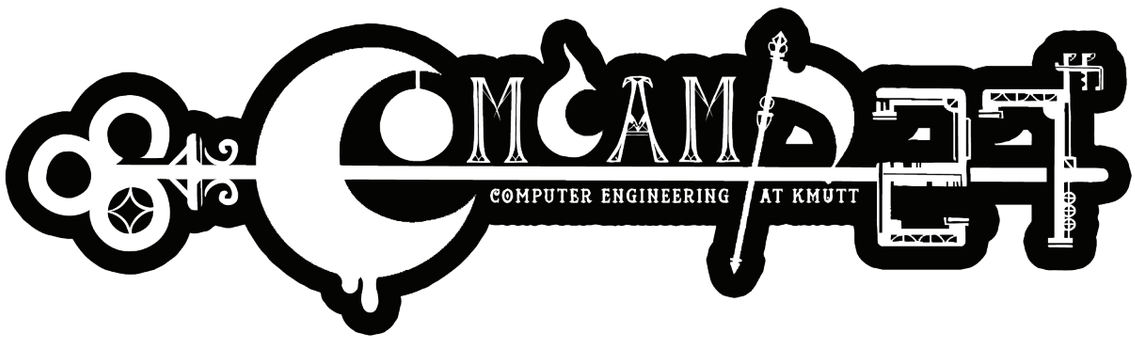
เอกสารประกอบการเรียน
โครงการฝึกอบรมเชิงปฏิบัติการคอมพิวเตอร์เบื้องต้น ครั้งที่ 27

C PROGRAMMING | ROBOT | WEB DEVELOPMENT | BASIC LINUX

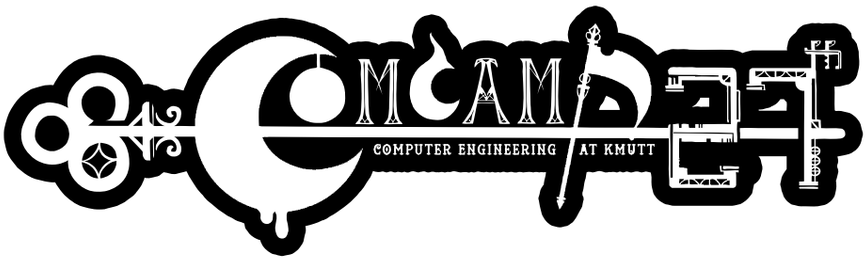
ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี



เอกสารประกอบการเรียน
โครงการฝึกอบรมเชิงปฏิบัติการคอมพิวเตอร์เบื้องต้น ครั้งที่ 27

C PROGRAMMING | ROBOT | WEB DEVELOPMENT | BASIC LINUX

ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี



เอกสารประกอบการเรียน โครงการฝึกอบรม เชิงปฏิบัติการคอมพิวเตอร์เบื้องต้น ครั้งที่ 27

ประธานโครงการ	นางสาวญานิศา	เหมประชิตชัย
ประธานฝ่ายวิชาการ	นายณัฐชนน	นินยวี
ผู้เขียน		
C Programming	นายงามพล	กลั่นบุศย์
Robot&Arduino	นายปวิณ	เตโชโยธิน และนายปุณยวัทน์ วุฒิศาสตร์
Basic Linux	นายสิริวัทก์	ลิ้มวัฒนา และนายเอกพล ล่ำสวย
Web Development	นายพฤกษ์	ภคเมธาวี
ผู้ตรวจทาน	นายวิโรจน์	แซ่เฮ้ง
	นายวรัตม์	กวีพรพจน์
	นางสาวรติมา	ดำคำ
	นายอินทัช	สินพูนภักดิ์
ศิลปกรรม	นางสาวญานิศา	เหมประชิตชัย
	นายฤทธิไชย	ศรีอุบลมาศ
	นายวรัทยา	โรจน์รัชนีกร
	นางสาวภัทรรัฐ	ปิจิธรรม
	นายตันติกร	ภูประเสริฐ
	นายปวเรศ	มงคลกิจงาม
	นายฤทธิเกียรติ	ทองภูธรณ์
คอมพิวเตอร์และจัดรูปเล่ม	นายราชศักดิ์	รักษ์กำเนิด

พิมพ์ครั้งที่ 1 พ.ศ. 2558 ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

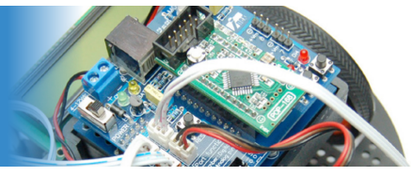
สารบัญ

C Programming

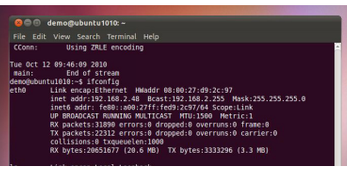
```
int main()

    char ch1,ch2,ch3,ch4,ch5,ch6,ch7,ch
    printf("Enter 1st character : ");
    scanf("%c",&ch1);
    fflush(stdin);
    printf("Enter 2nd character : ");
```


Robot

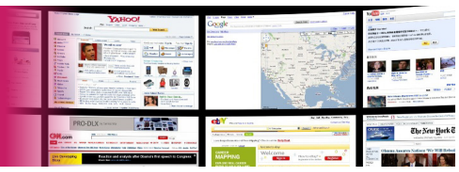
Arduino



Basic Linux



Web development



C programming

การเขียนโปรแกรม

การออกแบบโปรแกรมด้วย Pseudo-Code

ประเภทของตัวแปร

การดำเนินการและการเปรียบเทียบ

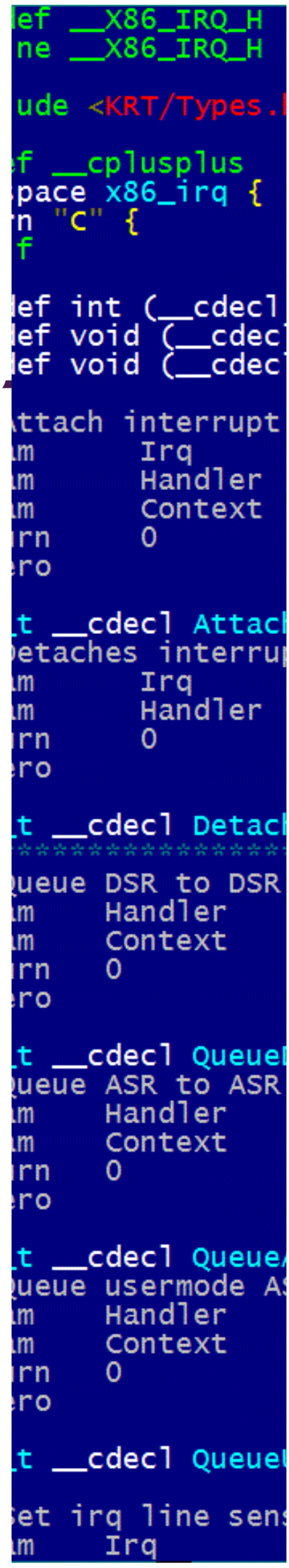
Library ในภาษาซี

Input / Output อย่างง่าย

การสร้างเงื่อนไขให้โปรแกรม

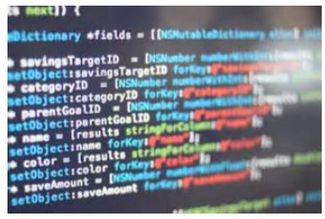
โจทย์ปัญหา

เฉลยโจทย์ปัญหา



การเขียนโปรแกรม

ทุกวันนี้คงปฏิเสธกันไม่ได้ว่าทุก ๆ วันนั้น พวกเราได้ใช้ชีวิตอยู่กับอุปกรณ์อิเล็กโทรนิกส์ดิจิตอลกันทุกคน และเกือบตลอดเวลา แต่ทราบหรือไม่ว่าสิ่งที่ขับเคลื่อนการทำงานของอุปกรณ์เหล่านั้นคือ Software ซึ่ง Software เหล่านี้ก็เกิดมาจากการนำ Function ต่างๆ มารวมกันจนเกิดเป็น Software และในแต่ละ Function ยังเกิดจากการเขียนโปรแกรมออกแบบการทำงานของ Function ต่างๆ อีกด้วย นี่จึงเป็นเหตุผลว่าทำไมการเขียนโปรแกรมถึงสำคัญ เนื่องจากในชีวิตประจำวันทุกคนต่างได้สัมผัสใกล้ชิดกับโปรแกรมตลอดเวลาตั้งแต่คอมพิวเตอร์ไปจนถึงโทรศัพท์มือถือ หรือเครื่องฟังเพลงอย่าง iPod เป็นต้น การพัฒนา Software นั้นพื้นฐานก็เริ่มจากการออกแบบ และลงมือเขียนโปรแกรม Function ต่างๆ ซึ่งการเขียนโปรแกรมก็จะมีการออกแบบส่วนต่างๆ ย่อยลงไปอีก ซึ่งหากได้เข้าศึกษาต่อในระดับที่สูงขึ้น ก็จะได้พบกับรายละเอียดอีกมากมาย



รูปที่ 1 รูปการเขียนโปรแกรมในภาษา objective C ใช้ในการเขียน iOS App

สำหรับสิ่งที่เราจะกล่าวถึงในทีนี้ก็จะเป็นการเขียนโปรแกรมด้วยภาษา C เบื้องต้นซึ่งจะมีบทเรียนดังนี้

- การออกแบบโปรแกรม
- ประเภทของตัวแปรสำหรับการเขียนโปรแกรม
- การดำเนินการและการเปรียบเทียบ
- Library ในภาษา C
- Input/Output อย่างง่าย
- การสร้างเงื่อนไขให้โปรแกรม
- การสร้างเงื่อนไขแบบวนรอบ

การออกแบบโปรแกรมด้วย Psuedo-Code

สำหรับการออกแบบโปรแกรมนั้นมีเครื่องมือเป็นตัวช่วยสองอย่างด้วยกันนั้นคือการเขียน “Flow Chart” และการเขียน “Psuedo-Code” ในที่นี้จะกล่าวถึงเพียงแค่ pseudo-code เท่านั้น สำหรับ Flow Chart สามารถศึกษาได้ตามหนังสือเขียนโปรแกรมทั่วไป

Pseudo-code (โค้ดเทียม) คือการเขียนโค้ดของโปรแกรมขึ้นมาเป็นภาษาที่ใครก็สามารถอ่านเข้าใจวิธีการของโปรแกรมได้ เนื่องจากการเขียนโปรแกรมของแต่ละคนนั้น จะต่างกันไปในแต่ละบุคคล โปรแกรมเดียวกันแต่เขียนด้วยคนละคนกันหน้าตาและการแก้ปัญหาของโปรแกรมก็จะต่างกันไปด้วย ซึ่งการเขียน pseudo-code จะทำให้คนที่อ่านโค้ดเราไม่รู้เรื่อง คนที่เขียนโปรแกรมไม่เป็น และคนที่จำเป็นต้องนำโปรแกรมเราไปใช้ สามารถเข้าใจหลักการโปรแกรมเราได้ ที่สำคัญ Psuedo-Code ยังช่วยในการออกแบบโปรแกรมของเราก่อนจะลงมือเขียนจริงเนื่องจากยังเป็นเพียง Psuedo-Code (โค้ดเทียม) ที่สามารถเปลี่ยนแปลงได้ง่ายต่างจากการลงมือเขียนโค้ดที่หากแก้ไขครั้งหนึ่งอาจจะแก้ไขหมดทั้งโปรแกรม หรือเขียนไปเกิด Error (ข้อผิดพลาด) หรือไม่รู้ว่าควรจะทำอย่างไรต่อ จะทำให้เสียเวลายิ่งกว่าการออกแบบแล้วเขียนอย่างมาก ดังนั้น การออกแบบ โปรแกรมก่อนจึงเป็นสิ่งที่สำคัญมาก ตัวอย่าง Pseudo-code เช่นโปรแกรมบวกเลขสองตัวด้านล่าง

```
รับค่าตัวเลข x และ y จากผู้ใช้
คำตอบ = x + y
แสดงผลค่า คำตอบ ออกมา
```

สำหรับโปรแกรมง่ายๆ อาจจะเขียนได้โดยไม่ต้องออกแบบ แต่สำหรับโปรแกรมที่ยากและใหญ่แล้วการออกแบบก่อนเป็นสิ่งที่ต้องทำเสมอ เมื่อเราสามารถออกแบบโปรแกรมได้แล้ว การเขียนโค้ดจะไม่ใช่ปัญหาอีกต่อไป

ประเภทของตัวแปร

ในการเขียนโปรแกรมนั้นเราจำเป็นจะต้องมีการใช้ตัวแปรเพื่อจำค่าต่าง ๆ และนำไปใช้ในกระบวนการต่าง ๆ สำหรับโปรแกรมของเรา ซึ่งประเภทของตัวแปรมีดังตารางด้านล่าง

ตัวแปรแต่ละชนิดเมื่อเราต้องการจะนำเข้า หรือแสดงผลออก จำเป็นต้องใช้ข้อความนำเข้าและนำออก ให้ถูกต้องตามชนิดตัวแปร มิฉะนั้นจะเกิด Error ขึ้นได้ และการใช้ตัวแปรให้ถูกต้องตามจุดประสงค์และใช้อย่างประหยัด

ประเภทตัวแปร	ลักษณะ	การประกาศใช้ในภาษา C	การนำเข้าและนำออก
Float	จำนวนจริง	float x;	%f
Double	จำนวนจริง(เยอะกว่า float)	double x;	%lf
Integer	จำนวนเต็ม	int x;	%d
Char	ตัวอักษร	char x;	%c
String	ชุดตัวอักษร	char x[20];	%s

ตารางที่ 1 แสดงประเภทของตัวแปร

จะทำให้โปรแกรมเราทำงานเร็วและประหยัด memory รวมถึงทำให้โปรแกรมมีขนาดเล็กลงด้วย

การดำเนินการและการเปรียบเทียบ

ในการเขียนโปรแกรมเราสามารถใช้การดำเนินการพื้นฐานทางคณิตศาสตร์เช่น บวก ลบ คูณ หาร ในการเปลี่ยนแปลงค่าให้กับตัวแปรได้ตามตัวอย่างในตารางด้านล่าง

ตัวดำเนินการ	ชื่อเรียก	ตัวอย่างวิธีการใช้	ผลลัพธ์
+	บวก	x = x + 5;	x มีค่าเท่ากับ 15
-	ลบ	x = x-5;	x มีค่าเท่ากับ 5
*	คูณ	x = x*5;	x มีค่าเท่ากับ 50
/	หาร	x = x/5;	x มีค่าเท่ากับ 2
%	มอดูโล(หารเพื่อหาเศษ)	x = x%3;	x มีค่าเท่ากับ 1(เศษจาก x/3)

ตารางที่ 2 ตัวดำเนินการต่างๆ และผลลัพธ์เมื่อ x มีค่าเท่ากับ 10

นอกจากการดำเนินการทางคณิตศาสตร์แล้วการเขียนโปรแกรมยังจำเป็นต้องพึ่งการเปรียบเทียบทางคณิตศาสตร์โดยหลักการไม่ต่างจากการเปรียบเทียบธรรมดาเช่น 3 มากกว่า 5 , 10 มากกว่าหรือเท่ากับ 5 เป็นต้น

เครื่องหมายเปรียบเทียบ	ชื่อเรียก	ตัวอย่างวิธีการใช้	ความหมาย
==	เท่ากับ	If(x==5)	ถ้า x เท่ากับ 5
!=	ไม่เท่ากับ	If(x!=5)	ถ้า x ไม่เท่ากับ 5
>	มากกว่า	If(x>5)	ถ้า x มากกว่า 5
>=	มากกว่าหรือเท่ากับ	If(x>=5)	ถ้า x มากกว่าหรือเท่ากับ 5
<	น้อยกว่า	If(x<5)	ถ้า x น้อยกว่า 5
<=	น้อยกว่าหรือเท่ากับ	If(x<=5)	ถ้า x น้อยกว่าหรือเท่ากับ 5
&&	และ	If(x==5&&x!=10)	ถ้า x เท่ากับ 5 และ x ไม่เท่ากับ 10
||	หรือ	If(x==5||x<10)	ถ้า x เท่ากับ 5 และ x น้อยกว่า 10

ตารางที่ 3 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการและเปรียบเทียบทั้งหมดที่ได้กล่าวมานั้นมีความสำคัญอย่างมากการใช้การเปรียบเทียบมากเกินไปจะทำให้โปรแกรมทำงานช้าและด้อยประสิทธิภาพดังนั้นเมื่อเขียนโปรแกรมเสร็จแล้วควรลอง Optimize โปรแกรมด้วยการลดจำนวนการเปรียบเทียบลง จะทำให้โปรแกรมทำงานได้เร็วและมีเสถียรภาพมากขึ้น

Library ในภาษาซี

Library เป็นสิ่งที่รวมคำสั่งต่างๆ เอาไว้ด้วยกัน อาจเรียกได้ว่าชุดคำสั่ง ซึ่งเราจำเป็นจะต้องบอก Compiler ก่อนเสมอว่าเราจะใช้คำสั่งใน Library ไหนบ้างเพราะถ้าเราไม่บอกไว้ เราจะไม่สามารถใช้คำสั่งต่างๆได้ แม้แต่คำสั่งพื้นฐานอย่าง `printf();` และ `scanf();` เพราะ Compiler จะไม่รู้จักคำสั่งพวกนี้เลยถ้าเราไม่เรียกชุดคำสั่งที่มีคำสั่งที่ต้องการใช้ เข้ามาในโปรแกรมของเราเสียก่อน โดย Library พื้นฐานที่ควรทราบในภาษา C มีดังนี้

Library	ลักษณะการใช้
stdio.h	เขียนโปรแกรมธรรมดาทั่วไป
string.h	ใช้สำหรับการดำเนินการต่างๆ ใน string
math.h	มีคำสั่งที่เป็นฟังก์ชั่นและค่าคงที่ทางคณิตศาสตร์
time.h	คำสั่งที่จำเป็นต้องใช้เรื่องของเวลาเช่นการสุ่มตัวเลข

ตารางที่ 4 ตารางแสดง library ที่สำคัญในการเขียนภาษา C

สำหรับการนำ Library ต่างๆ มาใช้ในโปรแกรมเราจำเป็นจะต้องระบุไว้เหนือสุดของการเขียนโปรแกรม โดยในภาษา C เราจะประกาศว่า `#include<Library>` เพื่อนำ Library ดังกล่าวมาใช้

Input/Output อย่างง่าย

การ**นำเข้า ประมวลผล** และ**นำออก** เป็นหัวใจหลักพื้นฐานของโปรแกรมต่างๆ และคอมพิวเตอร์ โดยเราจะมาทำความรู้จักกับคำสั่งที่ใช้นำเข้า และ นำออกกันโดยเริ่มจากการนำออกก่อน

สำหรับการนำออกนั้นเราจะใช้คำสั่งที่ชื่อว่า `printf();` ซึ่งใช้งานง่าย มีไว้แสดงผลข้อความและค่าของตัวแปรต่างๆ สำหรับวิธีใช้นั้นสามารถเขียนได้ดังนี้

ลักษณะการใช้	วิธีการเขียนคำสั่ง	การแสดงผลที่หน้าต่างโปรแกรม
แสดงข้อความทั่วไป	printf(“Hello World”);	Hello World
แสดงค่าตัวแปร(กำหนดให้ x=5)	printf(“x = %f”,x);	x = 5.000000
การกำหนดจำนวนหลักทศนิยม	printf(“x = %.2f”,x);	x = 5.00
การขึ้นบรรทัดใหม่ด้วยข้อความ \n	printf(“Hello\nWorld”);	Hello World

ตารางที่ 5 คำสั่ง `printf();`

```c
#include<stdio.h>

int main()
{
    int x = 5;
    printf("Hello World\n")
    printf("x = %d",x);
}
```

ผลลัพธ์ที่โปรแกรม

```
Hello World
x = 5
```

รูปที่ 2 การใช้งานคำสั่ง `printf();`

ต่อมาจะเป็นการนำเข้าค่าต่างๆ มาในโปรแกรมด้วยคำสั่ง scanf(); และนำมาเก็บไว้ในตัวแปรต่างๆ โดยวิธีการใช้นั้นสามารถเขียนได้ดังนี้

```c
#include<stdio.h>

int main()
{
    int ans,x,y;
    printf("Enter value of x : ");
    scanf("%d",&x);
    printf("Enter value of y : ");
    scanf("%d",&y);
    ans = x+y;
    printf("Result is %d",ans);
}
```

```
"E:\C Programming\Comca
Enter value of x : 2
Enter value of y : 3
Result is 5
```

รูปที่ 3 การใช้งานคำสั่ง `scanf();`

จากรูปที่ 3 เมื่อโปรแกรมทำงานจบคำสั่ง `printf();` และขึ้นคำสั่ง `scanf();` โปรแกรมจะทำการรอรับค่าจากผู้ใช้จึงจะทำงานบรรทัดต่อๆ ไป

การสร้างเงื่อนไขให้โปรแกรม

จากนี้เราจะทำการเพิ่มความสามารถในการตัดสินใจให้กับโปรแกรมด้วยคำสั่ง if, if-else, If-else if โดยคำสั่งดังกล่าวต้องการตัวดำเนินการเปรียบเทียบที่ได้กล่าวไปแล้วเบื้องต้น และยิ่งมีการเปรียบเทียบมากเท่าใด จะทำให้โปรแกรมทำงานช้าลงตามจำนวนการเปรียบเทียบ(รวมถึงขนาดของข้อมูลด้วย) สำหรับค่าที่จะใส่ในคำสั่ง if นั้นจะเป็นการเปรียบเทียบเงื่อนไขว่าเป็นจริงหรือไม่ ถ้าหากเป็นจริงจะทำคำสั่ง ที่อยู่ใน If ทั้งหมด (else และ else if ก็เช่นกัน ถ้าเป็นจริงจะทำคำสั่งที่อยู่ภายในทั้งหมดเพียงเงื่อนไขเดียว)

```
double number;
printf("Hello World.\nEnter Number : ");
scanf("%lf",&number);
if(number>0)
{
    printf("%lf is positive number.",number);
}
else if(number==0)
{
    printf("%lf is zero",number);
}
else
{
    printf("%lf is negative number.",number);
}
```

```
Hello World.
Enter Number : 0
0.000000 is zero

Hello World.
Enter Number : 10
10.000000 is positive number.

Hello World.
Enter Number : -2
-2.000000 is negative number.
```

รูปที่ 4 (ซ้าย) รหัสต้นฉบับ (Source code) ของ โปรแกรมตรวจสอบจำนวนนับ และ (ขวา) ผลลัพธ์ที่ได้

การสร้างเงื่อนไขแบบวนรอบ

เราได้ทำความรู้จักคำสั่งไปทั้งหมดแล้ว 4 คำสั่ง ต่อมาเราจะทำการสร้างเงื่อนไขให้โปรแกรมทำงานซ้ำด้วยคำสั่ง while, for, do-while หลักการของการวนซ้ำสามารถดูจาก Pseudo-code ด้านล่างนี้

```
X = 0
    ขณะที่ X < 3
    {
        printf("%d\n",x);
        X = X+1 //ทำให้ค่า X เพิ่มขึ้น 1
    }
```

จากโค้ดดังกล่าวโปรแกรมจะทำงานไปเรื่อยๆ วนซ้ำจนกว่า x จะมีค่าเท่ากับ 3 (x ไม่น้อยกว่า 3 แล้ว) ซึ่งเป็นการใช้เงื่อนไขชนิดหนึ่งเหมือนกัน เพียงแต่การวนรอบไม่ได้ใช้เงื่อนไขเพื่อสร้างทางเลือกแบบ if-else ที่ใช้เงื่อนไขเพื่อหาทางออกจากทางเลือกหลายๆ ทาง

สำหรับคำสั่ง for, while นั้นอาจถือได้ว่าเป็นคำสั่งเดียวกัน(วิธีการทำงานเหมือนกัน) แต่หลักการเขียน

ต่างกันเล็กน้อย เราสามารถใช้คำสั่งใดก็ได้ตามความถนัดของแต่ละบุคคล เราจะไปทำความรู้จักกับคำสั่ง while กันก่อนจากโปรแกรมด้านล่าง

```
int main()
{
    int x=0;
    while(x<3)
    {
        printf("x is %d\n",x);
        x++;
    }
}
```

```
"E:\C Programming\C
x is 0
x is 1
x is 2
```

รูปที่ 5
คำสั่ง while

จากรูปที่ 5 เราจะเห็นได้ว่าเมื่อ x มีค่าเท่ากับ 3 โปรแกรมจะหยุดทำงานในส่วนของการวนรอบทันที ซึ่งจากที่กล่าวไว้เบื้องต้นว่าคำสั่ง while กับ for นั้นมีความคล้ายคลึงกันมาก จากโปรแกรมด้านบนเราสามารถเขียนด้วยคำสั่ง for ดังนี้

```
        for( x = 0 ; x<3 ; x++) /* สำหรับ X เริ่มต้นที่ 0 , X < 3
ให้ทำชุดคำสั่งด้านล่างให้เสร็จแล้วเพิ่มค่า X ขึ้น 1 */
        {
            printf("%d\n",x);
        }
```

ดังนั้นเราจะเห็นได้ว่าหลักการเขียนเงื่อนไขวนรอบด้วยคำสั่ง for, while นั้นมีความต่างการเล็กน้อย แต่หลักการของคำสั่งจะเหมือนกัน และสามารถใช้แทนกันได้ขึ้นกับความถนัดของผู้เขียนโปรแกรม

นอกจากคำสั่ง for, while แล้วเรายังสามารถสร้างเงื่อนไขแบบวนรอบได้ด้วยคำสั่ง do-while ซึ่งหลักการของคำสั่งนี้จะต่างจาก for, while เล็กน้อย โดย for, while นั้นจะทำการเปรียบเทียบเงื่อนไขก่อนดำเนินการตามชุดคำสั่งภายใน แต่คำสั่ง do-while จะดำเนินการชุดคำสั่งภายในก่อน 1 รอบ จึงค่อยตรวจสอบเงื่อนไข เราสามารถเห็นความแตกต่างของคำสั่ง while , do-while ได้จากตัวอย่างโปรแกรมด้านล่าง

```
int main()
{
    int x=3;
    while(x<3)
    {
        printf("x is %d\n",x);
        x++;
    }
    printf("Out of loop.");
}
```

```
"E:\C Programmi
Out of loop.
```

```
int main()
{
    int x=3;
    do
    {
        printf("x is %d\n",x);
        x++;
    }while(x<3);
    printf("Out of loop.");
}
```

```
"E:\C Programming
x is 3
Out of loop.
```

รูปที่ 6 (ซ้าย) คำสั่ง while และ (ขวา) คำสั่ง do-while

จากรูปที่ 6 เห็นได้ชัดว่าเมื่อ x = 3 จะทำให้ไม่ตรงเงื่อนไขของการวนรอบ โปรแกรมจึงจะไม่ทำงานในส่วนของการวนรอบ และทำงานในส่วนถัดมาทันที แต่ในคำสั่ง do-while จะทำงานตามชุดคำสั่งภายในการวนรอบ 1 รอบ ก่อนจึงค่อยตรวจสอบเงื่อนไข ซึ่งเป็นเท็จแล้ว(x=4) โปรแกรมจึงหลุดออกมาจากการวนรอบและทำงานคำสั่งถัดไป

โจทย์ปัญหา

เราได้ศึกษาคำสั่งพื้นฐานต่างๆ ไปแล้ว ต่อไปเราจะทำการสร้างโปรแกรมที่มีคำสั่งหลายคำสั่งกันไว้ในโปรแกรมเดียว โดยเฉลยโปรแกรมจะอยู่ถัดไปจากโจทย์ปัญหา(ควรลงมือทำด้วยตนเองก่อน) โจทย์ปัญหาต่อไปนี้ไม่กำหนดเงื่อนไข สามารถเขียนโดยวิธีใดก็ได้

1 ให้เขียนโปรแกรมรับตัวอักษร 10 ตัวโดยรับทีละตัว
จากนั้นให้โปรแกรมพิมพ์ตัวอักษรทั้งหมดออกมาพร้อมกัน

Tip

ต้องใส่คำสั่ง fflush(stdin); ก่อนหน้าที่จะรับคำตัวแปร char ตัวต่อไป (ยกเว้นตัวแรก) **เสมอ**

2 ให้เขียนโปรแกรมทายเลขโดยให้กำหนดค่าที่น้องต้องการขึ้นมา
ในโปรแกรมเอง(ผู้เล่นไม่ทราบ) ให้โปรแกรมรับค่าจากผู้เล่นมา
จากนั้นให้โปรแกรมบอกว่าค่าที่รับมา ถูกมากกว่า หรือน้อยกว่าค่าที่กำหนดไว้
(เลขที่จะทายต้องอยู่ระหว่าง 1-100 และเป็นจำนวนจริงเท่านั้น)

Tip

ควรใช้คำสั่ง for, while ให้โปรแกรมรับค่าเรื่อยๆ จนกว่าจะทายตัวเลขถูก

3 ให้เขียนโปรแกรมค้นหา หรม. และ ครน.
ของจำนวนเต็มที่ผู้ใช้พิมพ์เข้ามา 2 ค่า แล้วแสดงผลออกมา

เฉลยโจทย์ปัญหา

สำหรับข้อแรกนั้นจำเป็นต้องประกาศตัวแปรทั้งหมด 10 แล้ว print ออกมาในทีเดียวตามเฉลยด้านล่าง

```
int main()
{
    char ch1,ch2,ch3,ch4,ch5,ch6,ch7,ch8,ch9,ch10;
    printf("Enter 1st character : ");
    scanf("%c",&ch1);
    fflush(stdin);
    printf("Enter 2nd character : ");
    scanf("%c",&ch2);
    fflush(stdin);
    printf("Enter 3rd character : ");
    scanf("%c",&ch3);
    fflush(stdin);
    printf("Enter 4th character : ");
    scanf("%c",&ch4);
    fflush(stdin);
```

```
Enter 6th character : N
Enter 7th character : A
Enter 8th character : R
Enter 9th character : A
Enter 10th character : K
Result : NINJANARAK
```

ทำให้ครบ 10 ตัวจากนั้นปริ้นออกมา

```
printf("Result : %c%c%c%c%c%c%c%c%c%c",ch1,ch2,ch3,ch4,ch5,ch6,ch7,ch8,ch9,ch10);
```

รูปที่ 8 เฉลยโจทย์ปัญหาข้อที่ 1

ต่อมาในข้อสองนั้นถ้าไม่ต้องการให้โปรแกรมจบการทำงานก่อนจะทายถูก สามารถใช้คำสั่ง while หรือ for ดังตัวอย่างเฉลยด้านล่างโดยในเฉลยได้เพิ่มส่วนที่นับจำนวนครั้งที่ใช้ในการทายจนกว่าจะทายถูก

```
int main ()

{
    int the_number, guess=0, count=0;
    the_number = 63;
    while(guess!=the_number)
    {
        printf("Please enter the value here [between 1-100] : ");
        scanf("%d",&guess);
        count++;
        if(guess<1||guess>100) printf("Number %d is invalid redo.\n",guess);
        else if(guess>the_number) printf("My number is a lower value.\n");
        else if(guess<the_number) printf("My number is a higher value.\n");
        else if(guess==the_number) printf("Correct! you took %d times to answer.",count);
    }
}
```

```
Please enter the value here [between 1-100] : 10
My number is a higher value.
Please enter the value here [between 1-100] : 20
My number is a higher value.
Please enter the value here [between 1-100] : 90
My number is a lower value.
Please enter the value here [between 1-100] : 80
My number is a lower value.
Please enter the value here [between 1-100] : 70
My number is a lower value.
Please enter the value here [between 1-100] : 63
Correct! you took 6 times to answer.
```

ตัวเลขที่จะทายต้องอยู่ระหว่าง 1<= guess <= 100 นะครับ

รูปที่ 9 เฉลยโจทย์ปัญหาข้อที่ 2

สำหรับข้อสุดท้าย โดยตัวโจทย์นั้นจะค่อนข้างยากด้วยคณิตศาสตร์ อาจจะต้องใช้เวลานานในการออกแบบ

```c
int main()
{
    int a,b,x,y,i,gcd,lcm;
    printf("Please Enter First Value : ");
    scanf("%d",&x);
    printf("Please Enter Second Value : ");
    scanf("%d",&y);
    a = x;
    b = y;
    while (b!=0)
        {
            i=b;
            b=a%b;
            a=i;
        }
    gcd = a;
    lcm = (x*y)/gcd;

  printf("GCD = %d\n",gcd);
  printf("LCM = %d\n",lcm);
```

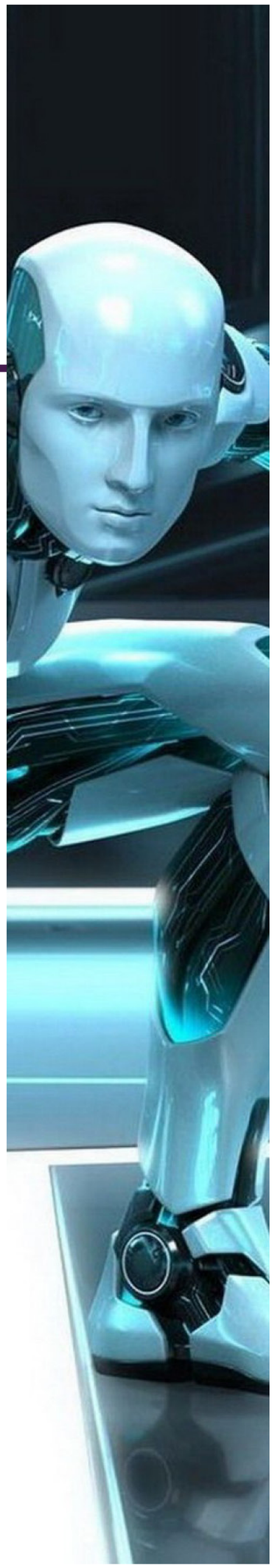
```
Please Enter First Value : 5
Please Enter Second Value : 10
GCD = 5
LCM = 10
```

หรม. = 5

ครน. = 10

รูปที่ 10 เฉลยโจทย์ปัญหาข้อที่ 3

Robot



หุ่นยนต์ คือ อะไร

หุ่นยนต์ประกอบด้วยอะไรบ้าง

กลไก (Mechanics)

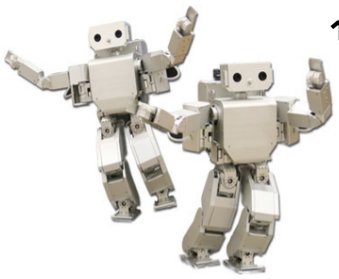
วงจรไฟฟ้า (Electronics)

ชุดคำสั่ง (Controller)

ประโยชน์ของหุ่นยนต์ในด้านต่าง ๆ

อนาคตของหุ่นยนต์

Introduction

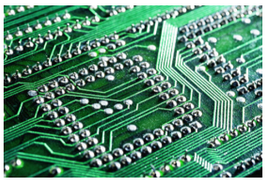


1.1 หุ่นยนต์ คือ อะไร

เป็นเครื่องจักรที่ทำงานแทนสิ่งมีชีวิตต่างๆ ซึ่งส่วนใหญ่ใช้งานแทนมนุษย์ สามารถนำใช้งานได้ตามที่ออกแบบลำดับขั้นตอนในการทำงานต่างๆ ไว้ในเครื่องนั้นๆ เช่น Pick&Place Robot, Arduino Robot เป็นต้น

1.2 หุ่นยนต์ ประกอบด้วยอะไรบ้าง

1.2.1 กลไก (Mechanics) คือ สิ่งที่ทำให้หุ่นยนต์สามารถเคลื่อนไหวหรือเคลื่อนที่ได้ รวมถึงโครงสร้างของตัวหุ่นยนต์ด้วย ตัวอย่าง เช่น มอเตอร์ เฟือง สายพาน โซ่ ลูกเบี้ยว เป็นต้น



1.2.2 วงจรไฟฟ้า (Electronics) ทำหน้าคล้ายระบบประสาทคอยเชื่อมระหว่างชุดคำสั่งกับส่วนกลไกรวมถึงระบบการรับรู้ของหุ่นยนต์ด้วย ตัวอย่าง เช่น Touch Sensor (เซนเซอร์ตรวจจับการสัมผัส), Magnetic Sensor (เซนเซอร์ตรวจจับสนามแม่เหล็ก), IR Sensor (เซนเซอร์ที่ใช้ตรวจจับวัตถุในระยะหนึ่งได้โดยใช้รังสี อินฟาเรด), Ultrasonic Sensor (เซนเซอร์ตรวจจับวัตถุในระยะโดยใช้คลื่นเสียง)

1.2.3 ชุดคำสั่ง (Controller) คือ สิ่งที่กำหนดการทำงานของหุ่นยนต์ คล้ายกับสมองที่ทำหน้าที่สั่งงานส่วนต่าง ๆ ของร่างกาย ตัวอย่าง เช่น Microcontroller.

1.3 ประโยชน์ของหุ่นยนต์ในด้านต่าง ๆ

- การใช้งานอุตสาหกรรม เช่น แขนกลที่สามารถเชื่อมโลหะ ยกของ หรือประกอบยานยนต์
- การใช้งานในชีวิตประจำวัน เช่น เครื่องดูดฝุ่นอัตโนมัติb
- การแพทย์ ใช้ในการผ่าตัด

1.4 อนาคตของหุ่นยนต์

- จินตนาการเกี่ยวกับหุ่นยนต์จากภาพยนตร์ หุ่นยนต์จะสามารถคิดเองได้ หุ่นยนต์จะสามารถผลิตตัวเองได้และพัฒนาตัวเองได้ หุ่นยนต์มีความคิดสร้างสรรค์
- การใช้ชีวิตกับหุ่นยนต์ งานหลายงาน เช่น งานเก็บขยะ งานทำอาหาร งานรักษาความปลอดภัย จะทำโดยหุ่นยนต์ทั้งหมดอาจมีมนุษย์ควบคุมบ้างเล็กน้อย

1.5 Robotics (วิทยาการหุ่นยนต์) เป็นศาสตร์ในการศึกษาหุ่นยนต์ ทั้งการออกแบบ การสร้าง การดำเนินการ และการนำหุ่นยนต์ไปใช้ ตลอดถึงระบบคอมพิวเตอร์ที่มีการควบคุม การตอบสนอง และการประมวลผลข้อมูล

1.5.1 การเข้าใจ หมายถึง ความสามารถในการประมวลผลสิ่งที่รับรู้ให้มีความหมาย เช่น การที่หุ่นยนต์มองเห็นขวดน้ำสามขวด หุ่นยนต์จะต้องเข้าใจว่ามีขวดน้ำอยู่สามขวดตั้งอยู่ด้านหน้าตัวเอง

1.5.2. การรับรู้ หมายถึง การมองเห็น การได้ยิน การสัมผัส หรือรับข้อมูลจากสิ่งแวดล้อมโดยใช้เซนเซอร์ต่าง ๆ

1.5.3. การควบคุม หมายถึง กลไกการควบคุมส่วนต่าง ๆ ของหุ่นยนต์ในส่วนนี้จะพูดระบบต่าง ๆ ที่จะผสานการทำงานระหว่าง ชุดคำสั่ง วงจรไฟฟ้า และ กลไก

Arduino

Microcontroller

โครงสร้างทั่วไปของ Microcontroller

Arduino(POP-168)

การใช้งาน Arduino

โปรแกรม Arduino

ขั้นตอนในการทดสอบ Arduino POP-168

โครงสร้างของ Programming สำหรับตัว Arduino POP-168

การอ่านค่า รับค่า แสดงผล

ตัวแปร (Variable)

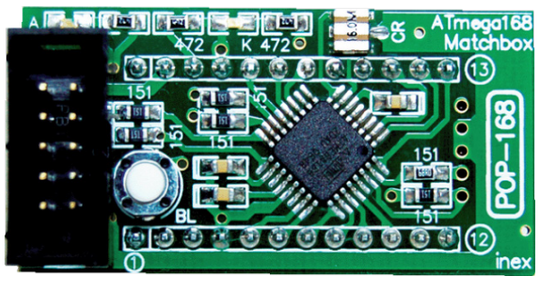
อุปกรณ์ต่าง ๆ ที่จะได้ใช้ในการเรียน

Microcontroller

คือ อุปกรณ์ควบคุมขนาดเล็กที่มาช่วยในการบันทึกคำสั่งที่มนุษย์ได้สร้างขึ้น และทำหน้าที่สั่งการไปยังอุปกรณ์ต่างๆ ซึ่งตัว Microcontroller นั้นบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ โดยในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำ และพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน

โครงสร้างทั่วไป

1. หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)
2. หน่วยความจำ (Memory)
3. ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port)
4. ช่องทางเดินของสัญญาณ หรือบัส (BUS)
5. วงจรกำเนิดสัญญาณนาฬิกา



Arduino (POP-168)

เป็น โปรเจคตัวหนึ่งที่สามารถใช้เป็นเครื่องมือในการควบคุมอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ตามที่เราต้องการ

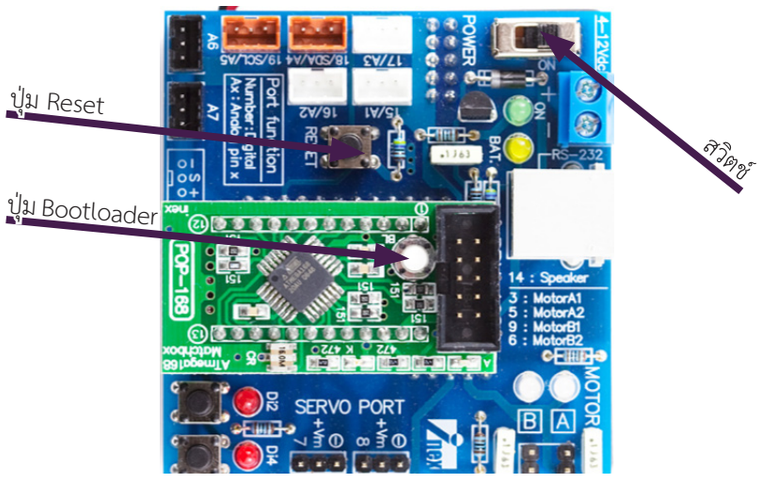
Arduino เป็นการพัฒนาในรูปแบบ Open-Source สามารถเรียกใช้ Library ตามความเหมาะสมในชิ้นงานที่เราต้องการ สามารถใช้เป็นเครื่องมือในการควบคุมอุปกรณ์อิเล็กทรอนิกส์ต่างๆ เช่น เซนเซอร์ LED motor เป็นต้น

Arduino POP-168 Board เป็น Module Microcontroller ตระกูล AVR ซึ่งได้เลือกใช้ Hard-

Tip

Open source หมายถึง สามารถใช้และพัฒนาโดยบุคคลทั่วไป
Library หมายถึง แหล่งรวบรวมคำสั่งที่ผู้พัฒนาได้เขียนไว้ให้แล้ว

ware ที่อยู่ในโครงการ Open-Source Microcontroller จึงทำให้สามารถนำชุดพัฒนาของ Arduino มาใช้งานได้ ซึ่งในนั้นจะมี คำสั่งในภาษาซีสำหรับติดต่อกับ Hardware จำนวนมากไว้ให้ ทำให้สามารถเขียนโปรแกรมสั่งงานต่างๆ ได้ง่าย โดยไม่จำเป็นต้องศึกษาในรายละเอียดในตัว Microcontroller มากนัก



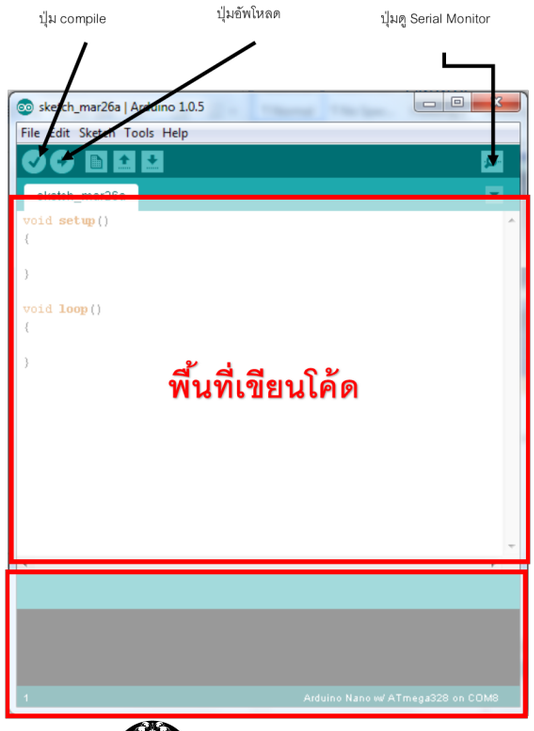
ภาพด้านบนของ POP-168 ที่น้องจะได้ใช้ในการเรียน

Tip

Module Microcontroller คือ ตัวควบคุมหุ่นยนต์ซึ่งสามารถถอดแล้วติดตามแผงวงจรได้
AVR เป็นตระกูลของ microcontroller ที่โปรแกรมได้โดยการ Flash Memory

การใช้งาน Arduino

โปรแกรม Arduino

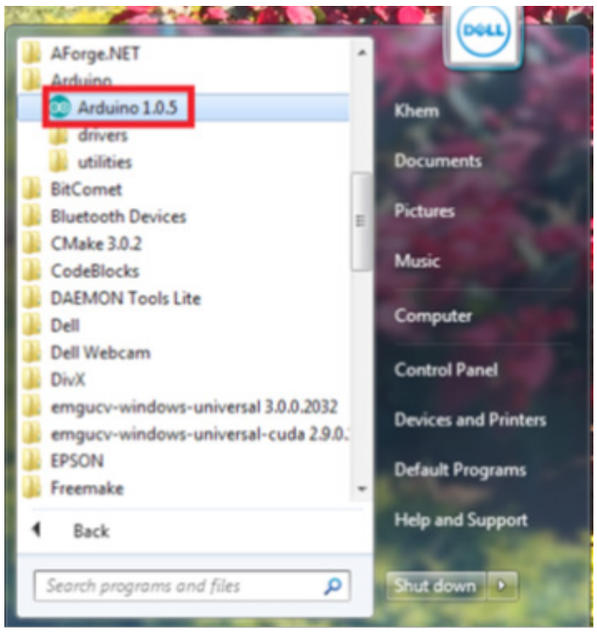


ขั้นตอนในการทดสอบ Arduino POP-168

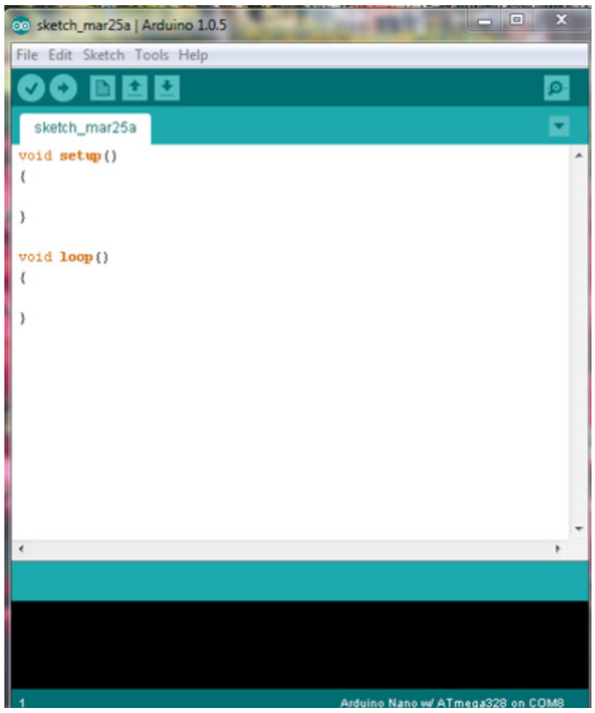
มีด้วยกัน 9 ขั้นตอนดังนี้

1. ต่อสายเชื่อมระหว่าง POP-168 กับตัว JX-POP168
2. เปิด โปรแกรม Arduino

Start -> All Programs -> Arduino

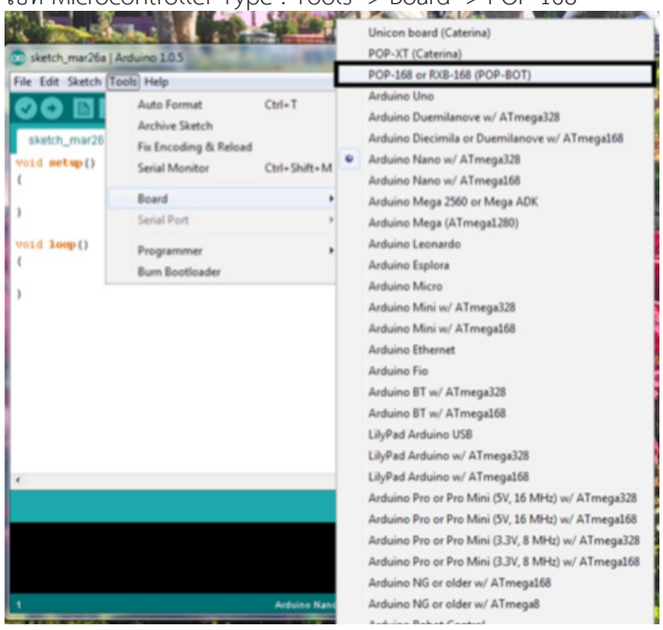


3. จะมีหน้าต่างของ Arduino ปรากฏขึ้นมา

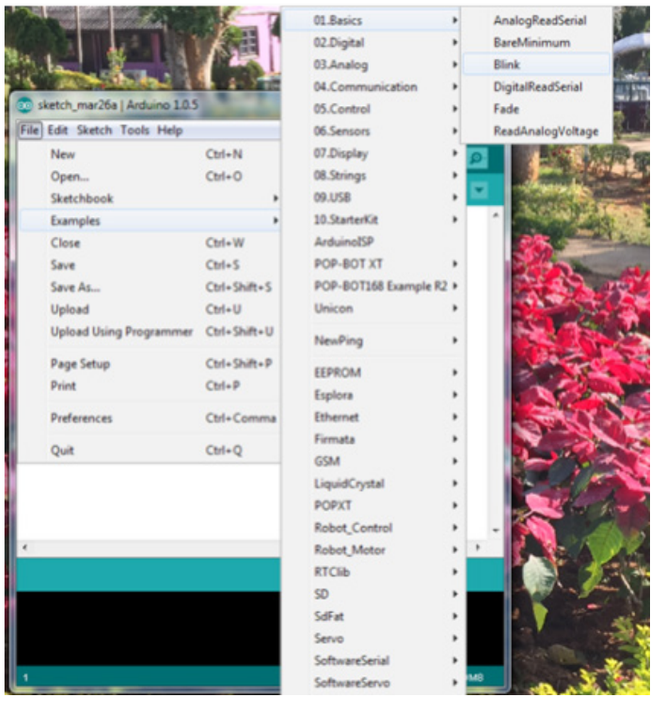


การตั้งค่าโปรแกรม Arduino ให้ทำงานร่วมกับ POP-168

ไปที่ Microcontroller Type : Tools -> Board -> POP-168



Serial Port : Tools -> Serial Port -> เลือกกด Port ที่เป็นตัว Arduino ของเรา



6. เตรียม POP-168 run boot loader mode มีขั้นตอนดังนี้

Boot loader mode มีไว้เพื่อให้ POP-168 สามารถรับคำสั่งจากโปรแกรม Arduino ได้

1.) ปิด POP-168

2.) กดปุ่ม boot loader ค้างไว้ จากนั้นให้เปิด POP-168 ซึ่งต้องให้หลอด LED สีฟ้าบน board นั้นติดและไม่กระพริบ

3.) ให้ปล่อยปุ่ม boot loader

7. กดปุ่ม Upload บนแถบหน้าต่าง Arduino

8. ให้รอจนกว่าจะมีข้อความแจ้งเตือนว่า “Done uploading” ที่หน้า console

9. ให้รีเซ็ตตัว board อีกครั้ง จากนั้นตรวจสอบว่าโค้ดที่ upload นั้นทำงานอย่างไร ถ้าเป็นไปดังที่ต้องการ ถือเป็นอันเสร็จสมบูรณ์ จากนั้นก็ลองสร้างผลงานของตัวเองได้เลย

โครงสร้างของ Programming สำหรับตัว Arduino POP-168

โครงสร้างของ Arduino Language จะมีความคล้ายกับโรงสร้างภาษาซี ในตัวโปรแกรมนั้นจะมีโครงสร้างหลักอยู่ 2 ส่วน นั่นคือ

1. `void setup()` – เป็นส่วนที่ทำงานอย่างแรกเพียงรอบเดียวเมื่อทำการเปิดสวิตช์ของ POP-168 โดยส่วนมากจะใช้ช่วงฟังก์ชั่นนี้ในการตั้งโหมดของ pin ต่าง ๆ ที่ต้องการใช้หรือแม้กระทั่งตั้งค่าช่องสัญญาณที่จะใช้ติดต่อกับคอมพิวเตอร์ผ่านทาง serial port

2. `void loop()` – เป็นส่วนที่หลังจาก `void setup()` ทำงานเสร็จ และจำทำซ้ำไปเรื่อย ๆ ไม่มีสิ้นสุดจนกว่าจะปิดสวิตช์ โดยส่วนมากใน `void loop()` จะใช้เขียนโปรแกรมหรือคำสั่งที่จะให้หุ่นยนต์ทำภารกิจที่ต้องการจนเสร็จสิ้นภารกิจ

การอ่านค่า การรับค่า การแสดงผล

`analogRead(pin)` - เป็นการอ่านค่าจาก pin นั้นๆ ซึ่ง pin คือตัวเลขของตัว analog input pin ที่อยู่ใน boardค่าที่ได้จากการอ่านจะมีตั้งแต่ 0 ถึง 1023 ซึ่งเกิดจากการเปลี่ยนเป็นตัวเลขที่อ่านค่ามาได้เป็นเลขที่มี10บิท

`digitalRead(pin)` - เป็นการอ่านค่าจาก pin นั้นๆ ซึ่ง pin คือตัวเลขของตัว digital pin ที่อยู่ใน board ซึ่งค่าที่อ่านได้จะมีเพียง2แบบเท่านั้น คือ HIGH กับ LOW

`analogWrite(pin,value)` – เป็นการทำให้มีการแสดงผลใน pin นั้นๆ ด้วยค่าๆหนึ่ง (value) ซึ่งมีค่าได้ตั้งแต่ 0 ถึง 255

`digitalWrite(pin,value)` - เป็นการทำให้มีการแสดงผลใน pin นั้นๆ ด้วยค่าๆหนึ่ง (value) ซึ่งมีค่าเพียง HIGH กับ LOW เท่านั้น

`pinMode(pin,mode)` - เป็นการตั้งว่า จะมีการติดต่อกับ pin นั้น เพื่อที่จะทำใน mode ใด ซึ่ง mode มี 2 อย่าง คือ INPUT , OUTPUT

`Serial.println(gp2)` - เป็นการแสดงผลของค่า gp2 บนจอแสดงผลบน Serial Monitor

`delay(t)` – เป็นคำสั่งที่ใช้ในคงสถานะของโปรแกรมทั้งหมดไว้เป็นเวลา t มิลลิวินาที

ส่วนการควบคุมนั้นจะมีคำสั่งซึ่งเหมือนกับภาษาซี ประกอบด้วย

1. `if , if...else`
2. `switch() case`
3. `for , while , do...while`
4. `return`
5. `break`
6. `continue`

ตัวแปร (Variable)

มีไว้เก็บค่าต่างๆ เช่น การอ่านค่าจากเซนเซอร์ที่อ่านจาก analog pin, การกำหนดค่าที่จะเปลี่ยนแปลงเมื่อเจอสถานการณ์ใด ๆ เป็นต้น

ชนิดของตัวแปรก็คล้ายกับภาษาซีนั่นคือ มี จำนวนเต็ม ทศนิยม ตัวอักขระ และที่เพิ่มมาก็ คือ ตัวแปรชนิด string หรือตัวแปรชนิดเป็นคำ

```
void setup()
{
  pinMode(2,OUTPUT);
  digitalWrite(2,HIGH);
}

void loop()
{

}
```

ตัวอย่างโค้ดที่ทำให้หลอด LED ติดไฟ

จากโค้ดข้างต้นเป็นการสั่งให้หลอด LED ที่ต่ออยู่กับช่อง pin(D2) (digital) นั้นติดอยู่ตลอดเวลา หลังจากที่ setup() หลอด LED ก็ยังคงติดเหมือนเดิม หากไม่ทำการปิด POP-168

อุปกรณ์ต่าง ๆ ที่จะได้ใช้ในการเรียน

1 Motor

เป็นสิ่งที่ทำให้หุ่นยนต์สามารถเคลื่อนไหวหรือเคลื่อนที่ได้ โดยหลักการที่ใช้คือการจ่ายไฟฟ้าเข้าไปที่มอเตอร์ จะทำให้มอเตอร์หมุน ขนาดความต่างศักย์ของขั้วทั้งสองของมอเตอร์จะบ่งบอกถึงความเร็วที่มอเตอร์จะหมุน และมอเตอร์นี้สามารถควบคุมทิศทางการหมุนได้ด้วยการเปลี่ยนทิศทางที่กระแสไฟฟ้าไหล

```
/***
มาทำความเข้าใจกันก่อนนะ
หุ่นยนต์รถ Arduino จะมีมอเตอร์ 2 ตัว ซึ่งเพื่อจะต่อมอเตอร์กับบอร์ดไว้ดังนี้ :
1. มอเตอร์ตัวที่ 1 (ล้อซ้าย) ขั้วบวก  ต่อช่อง D3
2. มอเตอร์ตัวที่ 1 (ล้อซ้าย) ขั้วลบ ต่อช่อง D5
3. มอเตอร์ตัวที่ 2 (ล้อขวา) ขั้วบวก ต่อช่อง D6
4. มอเตอร์ตัวที่ 2 (ล้อขวา) ขั้วลบ ต่อช่อง D9
***/
void setup(){ //ฟังก์ชันตั้งค่าเริ่มต้นของหุ่นยนต์
    pinMode(3, OUTPUT);   //ตั้งให้พอร์ต D3 (มอเตอร์ตัวที่ 1 (ล้อซ้าย) ขั้วบวก) เป็นค่าสั่งขาออก
    pinMode(5, OUTPUT);   //ตั้งให้พอร์ต D5 (มอเตอร์ตัวที่ 1 (ล้อซ้าย) ขั้วลบ) เป็นค่าสั่งขาออก
    pinMode(6, OUTPUT);   //ตั้งให้พอร์ต D6 (มอเตอร์ตัวที่ 2 (ล้อขวา) ขั้วบวก) เป็นค่าสั่งขาออก
    pinMode(9, OUTPUT);   //ตั้งให้พอร์ต D9 (มอเตอร์ตัวที่ 2 (ล้อขวา) ขั้วลบ) เป็นค่าสั่งขาออก
    Serial.begin(115200); //เรียกชุดคำสั่งของของอนุกรม (ช่องที่เราต่อกับคอมพิวเตอร์เพื่อไสโค้ด)
                          //ให้เริ่มทำงานที่ความถี่ 115200 บิตต่อวินาที เอาไว้ใช้คู่กับคำสั่ง
                          //Serial.print("bla bla bla") เพื่อดูค่าหรือผลลัพธ์ของ sensor
}
```

<TechnicalData>

analogWrite กับ DigitalWrite ต่างกันอย่างไร?

analogWrite จะเป็นการสั่งจ่ายกระแสไฟฟ้าแบบ PWM (Pulse Width Modulator — การจ่ายกระแสไฟฟ้าเป็นช่วงแบบกำหนดความถี่ให้คงที่ แต่ความกว้างของลูกคลื่น Pulse ไม่คงที่ ผลที่ได้คืออุปกรณ์ที่รับกระแสจะเปิดปิดไฟฟ้ารัวๆ ตามความกว้างของลูกคลื่นที่จ่ายมาให้) ทำให้อุปกรณ์ที่รับกระแสไฟฟ้าทำงานเสมือนจ่ายกระแสไฟฟ้าแบบค่อยลง เช่น มอเตอร์ลดความเร็ว (จริงๆ คือมอเตอร์หมุนแล้วหยุดเร็วมากๆ จนเหมือนลดความเร็วลง) ไฟ LED หรี่แสงลง (ลองใช้คำสั่งตัวอย่างจาก File->Sketchbook->Examples->Analog)ซึ่งค่าที่ใช้ควบคุมได้ มีได้ 256 ระดับ คือ 0-255

ส่วน digitalWrite จะเป็นการจ่ายกระแสไฟฟ้าแบบเปิดปิดทั่วไปที่เราใช้ทั่วไป คือเปิดก็จ่ายไฟตลอด ปิดก็คือปิดไปเลย ซึ่งสั่งได้แค่ 2 แบบ คือ HIGH (จ่ายไฟให้) และ LOW (ไม่จ่ายให้)

Reference :

http://arduino.cc/en/Reference/analogWrite, http://arduino.cc/en/Reference/digitalWrite

http://www.thaicpf.com/webboard/index.php?topic=68.0

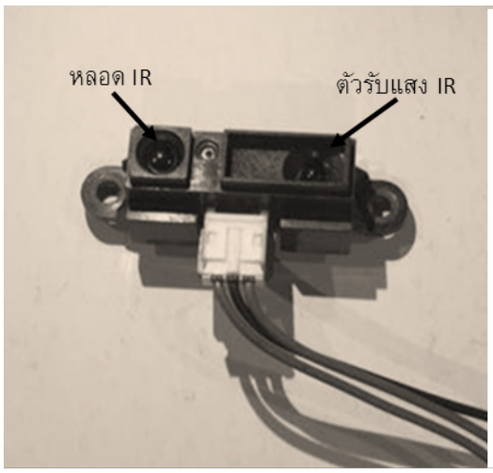
```
void Forward(int l, int r){
    /*สั่งให้รถเคลื่อนที่ไปข้างหน้าตามความเร็ว l สำหรับล้อซ้าย และ r สำหรับล้อขวา
    กำหนดความเร็วได้ตั้งแต่ 0 (ล้อไม่วิ่งเลย) ถึง 255 (ล้อวิ่งด้วยความเร็วสูงสุด)    */
    analogWrite(3, l); //สั่งมอเตอร์ตัวที่ 1 (ล้อซ้าย) ขั้วบวกให้มีศักย์ไฟฟ้าเป็น l
    digitalWrite(5, LOW); //สั่งให้มอเตอร์ตัวที่ 1 (ล้อซ้าย) ขั้วลบให้มีศักย์ไฟฟ้าเป็น 0
    /*การที่ขั้วบวกมีศักย์ไฟฟ้าเป็น l(มากกว่า 0) จะทำให้กระแสไฟฟ้าไหลจากขั้วบวกไปขั้วลบทำให้มอเตอร์
      ตัวที่ 1(ล้อซ้าย) หมุนไปข้างหน้าด้วยความเร็ว l
    */
    analogWrite(6, l); //สั่งมอเตอร์ตัวที่ 2 (ล้อขวา) ขั้วบวกให้มีศักย์ไฟฟ้าเป็น r
    digitalWrite(9, LOW); //สั่งให้มอเตอร์ตัวที่ 2 (ล้อขวา) ขั้วลบให้มีศักย์ไฟฟ้าเป็น 0
    /*การที่ขั้วบวกมีศักย์ไฟฟ้าเป็น r(มากกว่า 0) จะทำให้กระแสไฟฟ้าไหลจากขั้วบวกไปขั้วลบทำให้มอเตอร์
    ตัวที่ 2(ล้อขวา) หมุนไปข้างหน้าด้วยความเร็ว r
    */
```

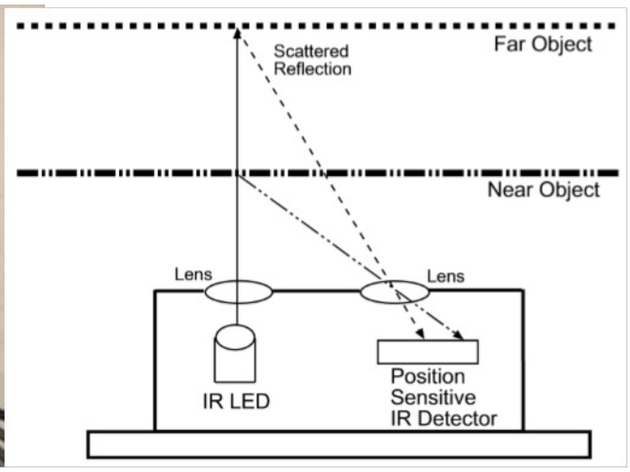
Source code ตัวอย่างในการสั่งให้หุ่นยนต์เดินหน้าตามความเร็วที่กำหนด

2 Sensor

เครื่องมือในการตรวจจับค่าต่างๆทางกายภาพ เช่น แสง เสียง ความร้อน รวมไปถึงการสัมผัส แบ่งออกเป็นหัวข้อหลัก ๆ ดังนี้

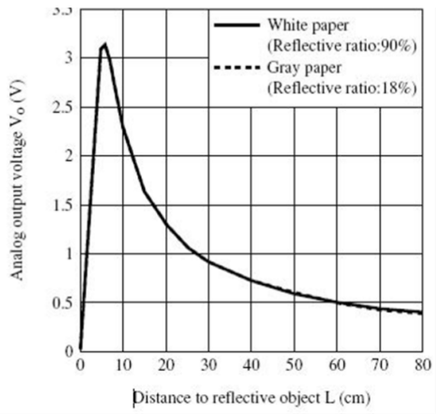
2.1. Infrared Sensor เป็นอุปกรณ์ที่ตรวจวัดระยะห่างของวัตถุจากตัวเซนเซอร์





หลักการทำงานของ Infrared sensor (IR sensor)

ทำงานโดยหลักการ triangulation นั่นคือ ตัวเซนเซอร์จะปล่อยแสงอินฟราเรดออกมาจากตัวหลอดไฟ เป็นจังหวะ จากนั้นแสงก็จะเดินทางไปยังวัตถุข้างหน้า แล้วสะท้อนกลับมายังตัวรับของเซนเซอร์ โดยตำแหน่งที่สะท้อนกลับมานั้นขึ้นอยู่กับระยะห่างระหว่างวัตถุ จากนั้นตัวเซนเซอร์จะเอาตำแหน่งที่รับแสงอินฟราเรดได้ไปคำนวณต่อว่าได้ระยะทางเท่าไหร่แล้วแปลงมาให้เป็นค่าสัญญาณ โดยแนวโน้มค่าสัญญาณที่ได้จาก IR sensor จะมีค่าตามกราฟนี้

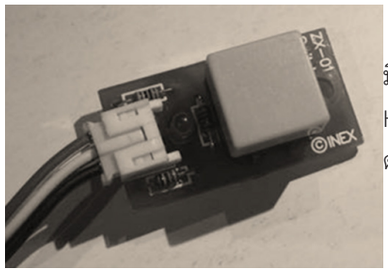


นั่นคือ ค่าที่รับได้จากเซนเซอร์จะมีค่ามากเมื่อวัตถุอยู่ใกล้ และค่าที่รับได้จะมีค่าน้อยเมื่อวัตถุอยู่ไกลออกไป ในการรับค่าจาก IR sensor นั้นจะรับค่าเป็นแบบ analog

```
/***
มาทำความเข้าใจกันก่อนนะ
หุ่นยนต์ Arduino ในโปรแกรมนี้จะมี IR sensor หนึ่งตัว
ซึ่งต่อกับ พอร์ต A2
***/
int gp2;                    // ตัวแปรสำหรับเก็บค่าที่รับได้จาก IR sensor
void setup()                //ฟังก์ชันตั้งค่าเริ่มต้นของหุ่นยนต์
{
    pinMode(A2, INPUT);     //ตั้งให้พอร์ต A2 เป็นค่าสังขาเข้า
    Serial.begin(115200); //เรียกชุดคำสั่งของช่องอนุกรม (ช่องที่เราต่อกับคอมพิวเตอร์เพื่อไล่โค้ด)
                          //ให้เริ่มทำงานที่ความถี่ 115200 บิตต่อวินาที เอาไว้ใช้คู่กับคำสั่ง
                          //Serial.print("A") เพื่อดูค่าหรือผลลัพธ์ของ sensor
}

/**** <TechnicianData>
***analogRead กับ digitalRead ต่างกันอย่างไร?
analogRead เป็นอ่านอ่านค่าจากพอร์ตที่เป็น analog ซึ่งค่าที่อ่านได้จะเกิดจากการเปรียบเทียบ สเกล 0 โวลต์ ถึง 5 โวลต์ ให้กลายเป็น 0 ถึง 1023
เช่นคือ ค่าที่อ่านได้ 1 หน่วยมีค่าเท่ากับ ศักย์ไฟฟ้า 5/1024 หรือ 0.0049 โวลต์

ส่วน digitalRead จะเป็นค่าแบบ HIGH(1) หรือ LOW(0) เท่านั้น
จะอ่านได้ค่า LOW ก็ต่อเมื่อค่าที่รับได้เป็น 0 และ
จะได้ค่า HIGH เมื่อค่าที่รับได้เป็นอะไรก็ได้เช่น 1,2 ซึ่งไม่ใช่ 0
Reference :
http://arduino.cc/en/Reference/DigitalRead
http://arduino.cc/en/Reference/AnalogRead
</TechnicianData>****/

void loop()                 // ฟังก์ชันทำงานวนรอบของหุ่นยนต์
{
    gp2 = analogRead(A2); // เป็นคำสั่งเพื่อเก็บค่าที่รับได้จาก IR sensor ไว้ในตัวแปร gp2
    Serial.println(gp2);
    delay(500);
}
```

ในโค้ดนี้จะมีการกำหนดตัวแปรชื่อ gp2 ไว้ซึ่งจะเป็นตัวแปรที่จะเก็บค่าที่รับได้จากเซนเซอร์ และใช้คำสั่ง analogRead(2) เพื่อรับค่าจากเซนเซอร์ซึ่งต่ออยู่ใน pin analog 2 ข้อดีอย่างหนึ่งของการใช้ IR sensor คือ สามารถตรวจจับวัตถุที่ระยะต่าง ๆ ได้ระยะหนึ่งซึ่ง ไม่ไกลมากแต่ไม่จำเป็นที่วัตถุจะต้องมาแตะกับตัวเซนเซอร์ ก็สามารถตรวจพบวัตถุได้ ดังนั้นจึงนิยมใช้ การรับค่าแบบ analog เมื่อใช้ IR sensor

2.2. Bumper sensor เป็นอุปกรณ์ที่ไว้ตรวจจับเจอวัตถุเมื่อมีการสัมผัสกับวัตถุต่างๆ

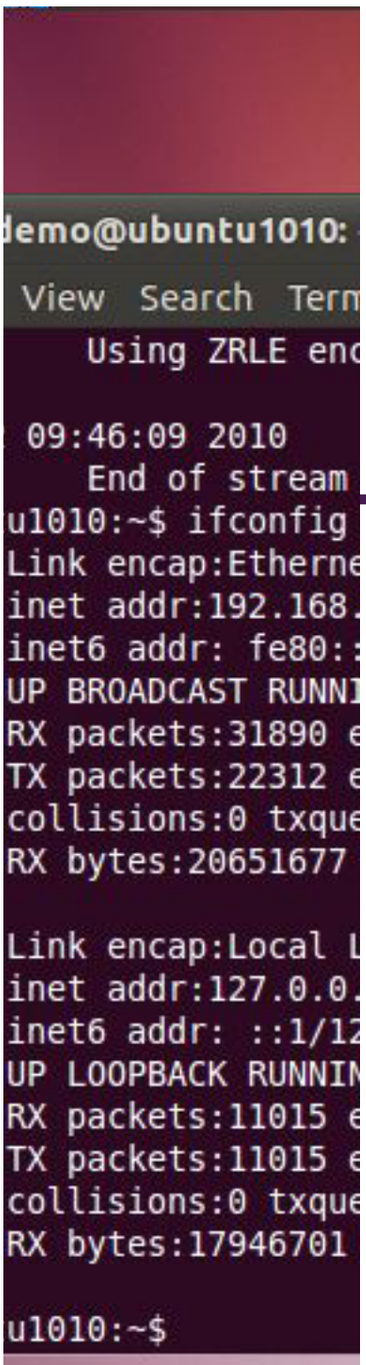


หลักการทำงานของ bumper sensor ดังที่กล่าวข้างต้น คือ คล้ายสวิตซ์ เมื่อพบวัตถุนั่นคือมีการกดปุ่มค่าสัญญาณที่ได้คือ 0 หรือ LOW และในสถานะปกติที่ไม่พบวัตถุค่าสัญญาณที่ได้ คือ 1 หรือ HIGH ดังนั้นการรับค่าจาก bumper sensor จึงใช้การรับค่าแบบ digital คือ พบวัตถุ กับไม่พบวัตถุ โดยด้วยแค่ bumper sensor นั้น ไม่สามารถบ่งบอกระยะที่วัตถุอยู่ห่างจากตัวเซนเซอร์หรือตัวหุ่นยนต์ได้

```
oid setup()


oid loop()

 int l=digitalRead(A2),r=digitalRead(A3);
 if(r==0)
 {

 }
 else if(l==0)
 {

 }
 else
 {

 }
```

ตัวอย่างโค้ดการใช้ bumper sensor

Note



Basic Linux

OS (ระบบปฏิบัติการ)

Linux

ระบบปฏิบัติการสำเร็จรูป (Linux Distribution)

ระบบไฟล์ในลินุกซ์ (Linux File System)

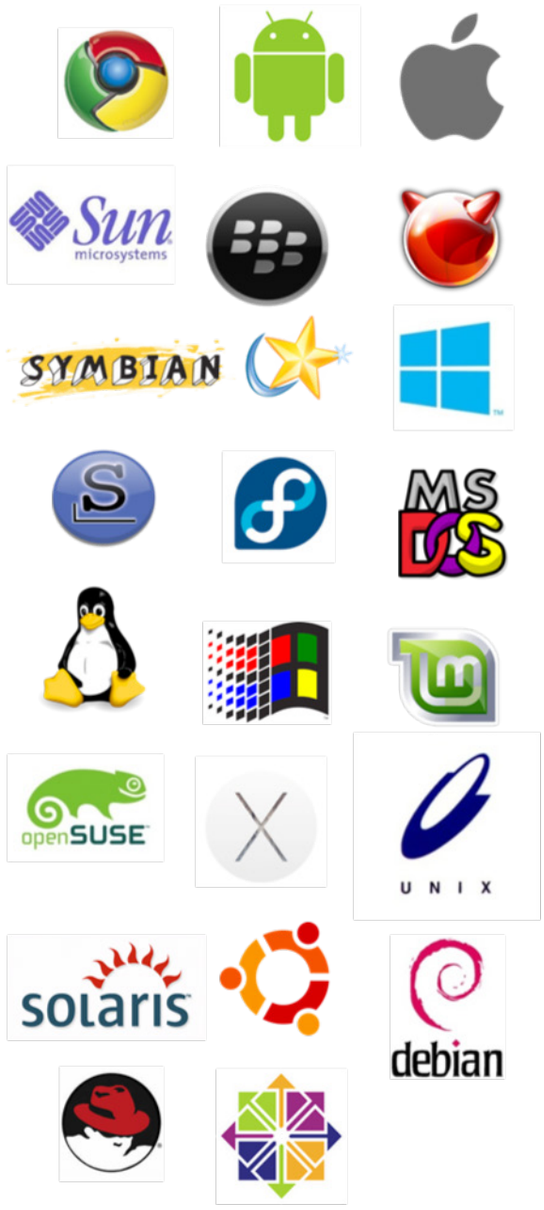
โครงสร้างการเก็บข้อมูลในฮาร์ดดิสก์

ช่องทางการเชื่อมต่อ (Mount Point)

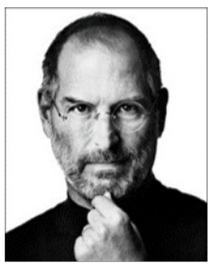
คำสั่งเบื้องต้นที่ควรทราบ

OS

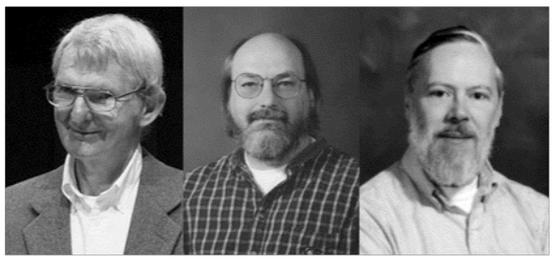
ย่อมาจาก Operating System หรือที่เรารู้จักว่า ระบบปฏิบัติการ ทำหน้าที่เป็นตัวกลางที่ทำหน้าที่ควบคุมการทำงาน ติดต่อระหว่างผู้ใช้ และฮาร์ดแวร์ ทั้งนี้ยังช่วยในเรื่องการจัดการทรัพยากรของเครื่อง โดยระบบปฏิบัติการ ในโลกใบนี้มีมากมาย โดยแต่ละระบบก็มีจุดน่าสนใจ และมีความสำคัญแตกต่างกันไป เช่น iOS ที่มีการใช้งานที่เรียบง่าย และมีความนิยมต่อเนื่องในปัจจุบัน windows ระบบปฏิบัติการยอดนิยมที่มีผู้ใช้งานอย่างแพร่หลาย แอนดรอยด์ ระบบปฏิบัติการบนสมาร์ทโฟน ยูนิกส์ แม้ไม่ค่อยมีใครรู้จัก แต่เป็นระบบปฏิบัติการที่มีนำมาพัฒนากันอย่างแพร่หลายในรูปแบบต่างๆ เช่น ลินุกส์ โซลาร์ลิส แม้กระทั่ง Mac OS



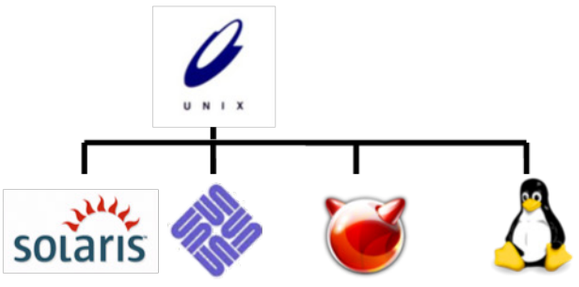
Bill Gates



Steve Jobs



Douglas Mcllroy | Ken Thompson | Dennis Ritchie



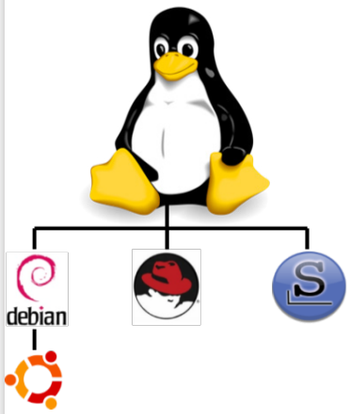
Linux

รูปที่ 1 นาย Linus Torvalds ผู้ริเริ่มพัฒนา Linux

ลินุกซ์ (Linux) เป็นระบบปฏิบัติการคอมพิวเตอร์ระบบหนึ่งซึ่งเลียนแบบมาจากยูนิกส์ (Unix-Like) โดยนาย Linus Torvalds จากมหาวิทยาลัยเฮลซิงกิ ประเทศฟินแลนด์ ในปี ค.ศ. 1980

จุดเด่นของ Linux

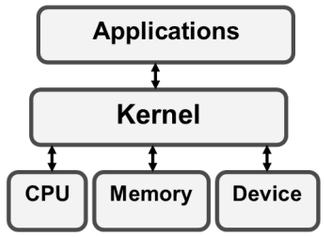
1. ฟรี และเป็น Open Source
2. เข้ากันได้ (Compatible) กับ Unix
3. ทำงานร่วมกับ DOS และ Windows ได้
4. สามารถใช้แฟ้มร่วมระบบปฏิบัติการอื่นได้
5. มีความสามารถในเรื่องของเน็ตเวิร์ค
6. มีประสิทธิภาพสูงในการใช้ฮาร์ดแวร์
7. เคอร์เนล (Kernel) มีประสิทธิภาพสูง
8. มีการช่วยเหลือและคำแนะนำเมื่อเกิดปัญหา



รูปที่ 2 (บน) “Tux” Mascot ของกลุ่มระบบปฏิบัติการ Linux
(ล่าง) Distribution หลักของ Linux ได้แก่ Debian, Ubuntu (ต่อยอดจาก Debian), RedHat, และ Slackware

	Linux	Windows
ค่าใช้จ่าย	ฟรี (บาง Distribution)	3,000 – 7,000 บาท
ความเสถียร	สูงกว่า	สูง
ความต้องการระบบ	ใช้ทรัพยากรน้อย	ใช้ทรัพยากรมาก
ผู้พัฒนา	กลุ่มผู้ใช้ทั่วโลก	ไมโครซอฟท์
ระบบไฟล์	ใต้ root	จำลอง Partition
การใช้งาน	ต้องมีความรู้พื้นฐาน	ไม่จำเป็นต้องมี
ไดร์เวอร์	นอกเหนือจากเคอร์เนล	จำเป็นต้องลง

เคอร์เนล (Kernel) คือ แก่นซึ่งเป็นส่วนที่สำคัญของระบบปฏิบัติการ ซึ่งคอยดูแลบริหารทรัพยากรของระบบ และติดต่อกับฮาร์ดแวร์และ ซอฟต์แวร์ เนื่องจากว่าเป็นส่วนประกอบพื้นฐานของระบบปฏิบัติการ เคอร์เนล นั้นเป็นฐานล่างสุดในการติดต่อกับทรัพยากรต่างๆ เช่น หน่วยความจำ หน่วยประมวลผลกลาง และ อุปกรณ์อินพุตและเอาต์พุต ดังนั้นเคอร์เนลจึงเปรียบเสมือนหัวใจหลักของระบบปฏิบัติการ



รูปที่ 3 แสดงถึงความสัมพันธ์ของการทำงานของแต่ละส่วน

รูปที่ 5 แสดงถึง Distribution ต่างๆ ของระบบปฏิบัติการลินุกซ์

ระบบปฏิบัติการสำเร็จรูป (Linux Distribution)

ลินุกซ์ (Linux) นั้นโดยความเข้าใจของคนทั่วไปคือ ระบบปฏิบัติการ (Operating System) แต่ในความเป็นจริงแล้วมันเป็นเพียงแค่ เคอร์เนล หรือเป็นเพียงองค์ประกอบหลักองค์ประกอบหนึ่งของระบบปฏิบัติการ (OS) เท่านั้น เรายังต้องการองค์ประกอบอื่นๆอีกเพื่อให้ครบถ้วนเพียงพอต่อการใช้งาน ซึ่งองค์ประกอบอื่นๆที่ว่าก็คือ ซอฟต์แวร์แพ็คเกจต่างๆ เช่น Window Manager, Text Editor, Terminal, File Manager เป็นต้น การจะที่เราจะดาวน์โหลดองค์ประกอบต่างๆ มาติดตั้งเองทั้งหมดคงไม่ใช่เรื่องง่ายแน่ ดังนั้น ในการใช้งานจริงเราจึงเลือกที่จะดาวน์โหลดและติดตั้ง Linux Distribution หรือที่เรียกสั้นๆว่า Distro แทน อาทิเช่น Red Hat, Ubuntu, Arch เป็นต้น Linux Distro ก็คือ Linux สำเร็จรูปพร้อมใช้ที่มีกลุ่มคนหรือองค์กรรวบรวมองค์ประกอบต่างๆที่จำเป็นต่อการเป็นระบบปฏิบัติการให้แล้ว

อย่างไรก็ตาม Distro นั้นมีอยู่มากมาย และแต่ละ Distro ก็มีความหลากหลายทั้งจุดมุ่งหมาย,แนวทางและรายละเอียด บ้างก็เน้นไปที่การใช้งานเป็น Desktop บ้างก็เน้นไปที่การใช้งานเป็นเซิร์ฟเวอร์ (Server) และแต่ละตัวก็ล้วนแล้วแต่มีข้อดีน่าใช้ทั้งสิ้นปัจจุบันมี Linux Distro มากกว่า 250 Distribution (ข้อมูลจาก : http://distrowatch.com)

ระบบไฟล์ในลินุกซ์ (Linux file system)

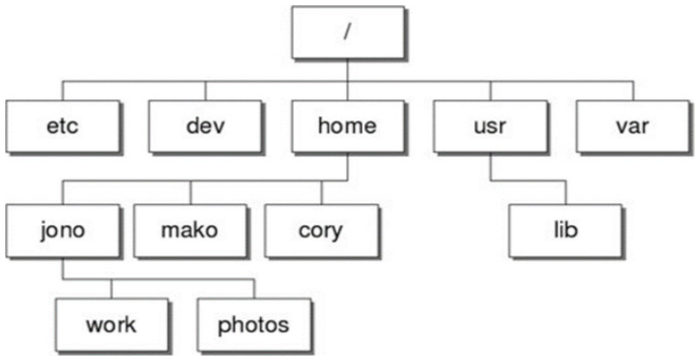
การจะใช้งานลินุกซ์ ให้มีประสิทธิภาพนั้น หากเรามีความรู้เกี่ยวกับไฟล์ ไดเรกทอรี และสิทธิ์สำหรับการใช้งานจะส่งผลให้สามารถเข้าใจการทำงานของ ลินุกซ์ โดยคร่าวๆ ซึ่ง ลินุกซ์ เป็นระบบปฏิบัติการ เสมือน Unix (ระบบปฏิบัติการทำงานจึงเหมือนกับ Unix แทบจะทั้งหมด) ลินุกซ์ จึงมีข้อมูลทุกอย่างเป็นไฟล์ด้วยเช่นกัน ไฟล์ใน ลินุกซ์ จะจัดแบ่งและเรียงลำดับกันเป็นโครงสร้างต้นไม้ (Tree Structure)

ไดเรกทอรีและไฟล์ในระบบยูนิกซ์

- **ไฟล์ (File)** คือสิ่งที่ใช้แทนตำแหน่งของข้อมูลที่อยู่ในหน่วยความจำ โดยทั่วไปการตั้งชื่อไฟล์นิยมใช้เครื่องหมาย . ในการแยกชื่อกับส่วนขยายของไฟล์ (File Extension) เพื่อระบุว่าไฟล์นี้เป็นประเภทอะไร
- **ไดเรกทอรี (Directory)** คือพื้นที่เสมือนในคอมพิวเตอร์ มีไว้สำหรับจัดเก็บไฟล์หรือไดเรกทอรีอื่นๆให้อยู่ในพื้นที่เดียวกัน

โครงสร้างของไดเรกทอรีในระบบยูนิกซ์

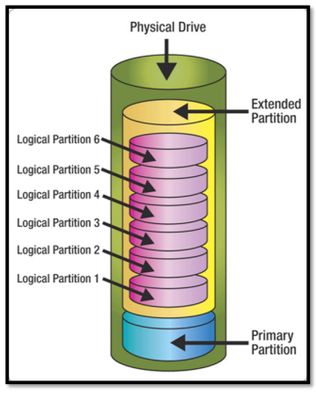
ไดเรกทอรีในระบบยูนิกซ์จะอ้างอิงโครงสร้างข้อมูลแบบต้นไม้หัวกลับ จากรากของต้นไม้ที่แตกรากสาขาจากด้านบนลงสู่ด้านล่าง โดยมีส่วนที่อยู่ด้านบนสุดเรียกว่า root ซึ่งแทนด้วยเครื่องหมาย / ดังรูป



ไดเรกทอรีหลักต่างๆ ในลินุกซ์

ไดเรกทอรี	หน้าที่
/	root
/bin	เก็บโปรแกรมที่เป็นคำสั่งของระบบ
/boot	เก็บ Kernel (โปรแกรมส่วนที่เป็นการทำงานหลักของระบบปฏิบัติการ) และไฟล์ที่เกี่ยวข้องกับการเริ่มต้นระบบ
/dev	เก็บไฟล์ที่เป็นส่วนติดต่อของอุปกรณ์ต่างๆ เช่น ดิสก์ เครื่องพิมพ์ ฯลฯ
/etc	เก็บไฟล์ที่ใช้สำหรับกำหนดค่าการทำงานและไฟล์ที่เกี่ยวข้องกับการเริ่มต้นระบบ
/home	ไดเรกทอรี home ของผู้ใช้ ตั้งชื่อตามชื่อบัญชีผู้ใช้
/lib	เก็บไฟล์ไลบรารีของระบบ
/media	mount point ของอุปกรณ์เก็บข้อมูลแบบถอดได้ (มีในระบบปฏิบัติการสมัยใหม่)
/mnt	mount point ของอุปกรณ์เก็บข้อมูลแบบถอดได้
/opt	เก็บโปรแกรมที่ผู้ใช้ติดตั้งเพิ่มเติมซึ่งไม่เกี่ยวข้องกับระบบหลัก
/proc	ไฟล์ข้อมูลของโปรเซสทั้งหมดที่อยู่ในหน่วยความจำ
/root	ไดเรกทอรี home ของ root (บางระบบปฏิบัติการจะอยู่ใน /home)
/sbin	เก็บโปรแกรมที่เป็นคำสั่งของระบบที่ต้องใช้สิทธิ์ของ root ในการรัน
/tmp	เก็บไฟล์ชั่วคราว (Tempolary file) จะถูกลบทิ้งเมื่อเปิดเครื่องขึ้นมาใหม่
/usr	เก็บข้อมูลคำสั่งอื่นๆเพิ่มเติม โครงสร้างภายในคล้ายโครงสร้างของ /
/var	เก็บไฟล์ที่เกี่ยวข้องกับการตั้งค่าและการทำงานของระบบ

โครงสร้างการเก็บข้อมูลในฮาร์ดดิสก์



รูปที่ 7 ผังสรุปโครงสร้างการเก็บข้อมูลใน harddisk

ฮาร์ดดิสก์ (Hard disk) ส่วนของพื้นที่เก็บข้อมูล เรียกว่า พาร์ติชั่น (Partition) ซึ่งสามารถมีได้มากกว่า 1 พาร์ติชั่น และสูงสุดไม่เกิน 4 พาร์ติชั่นรวมเรียกว่า พาร์ติชั่นหลัก(Primary Partition)

หากเราต้องการสร้างมากกว่า 4 พาร์ติชั่น เราต้องต้องให้พาร์ติชั่นใดพาร์ติชั่นหนึ่งมาทำเป็นพาร์ติชั่นเสริม (Extended Partition) จึงจะสามารถสร้างพาร์ติชั่นใหม่เพิ่มเติมในพาร์ติชั่นเสริม ซึ่งพาร์ติชั่นที่สร้างใหม่นี้จะเรียกว่า พาร์ติชั่นย่อย (Logical Partition)

Primary Partition คือ Partition ของระบบเดิม ซึ่งมีได้มากถึง 4 Partition (1,2,3,4 ตามลำดับ)

Extended Partition คือ Partition ที่เกิดขึ้นมาภายหลัง เพื่อแก้ปัญหาความต้องการของผู้ใช้ ที่จะแบ่ง Partition ให้มากกว่าของเดิม (4 Partition) โดยการแบ่งพาร์ติชั่นย่อยภายในพาร์ติชั่นเสริม (ความจริงแล้ว Extended Partition ก็เป็น Primary partition)

Logical Partition คือ Partition ที่สร้างขึ้นภายในพาร์ติชั่นเสริม โดยจะเริ่มนับจาก 5 ไปเรื่อยๆ (5,6,7, ...)

การตั้งชื่อพาร์ติชั่น

ลินุกซ์จะมองเห็นอุปกรณ์ทุกอย่างเป็นไฟล์ เรียกว่า device file อยู่ภายใต้ไดเรกทอรี ชื่อ /dev ฮาร์ดดิสก์จะถูกมองเป็น device file เช่นกัน วิธีการตั้งชื่อ device file ของฮาร์ดดิสก์ โดยจะมีการตั้งชื่อแยกเป็น 3 ส่วน คือ ประเภทของอุปกรณ์ ,ลำดับของอุปกรณ์ และลำดับ Partition ของ Hard disk

ส่วนแรก แทนประเภทของฮาร์ดดิสก์

- ถ้าเป็นแบบ ide จะแทนด้วยอักษร hd (ปัจจุบันแทบจะหาไม่ได้แล้ว เลิกใช้นานแล้ว)
- ถ้าเป็น SATA จะแทนด้วยตัวอักษร sd

ส่วนที่สอง แทนลำดับของอุปกรณ์ โดยจะเรียงจากตามตัวอักษรภาษาอังกฤษเริ่มจาก a – z แล้วจึงเริ่ม Aa–Az เช่น

sda หมายถึง Hard disk ตัวที่ 1

sdb หมายถึง Hard disk ตัวที่ 2

sdc หมายถึง Hard disk ตัวที่ 3

ส่วนสุดท้าย แทนหมายเลข Partition ของฮาร์ดดิสก์ที่กล่าวไว้ข้างต้น เช่น

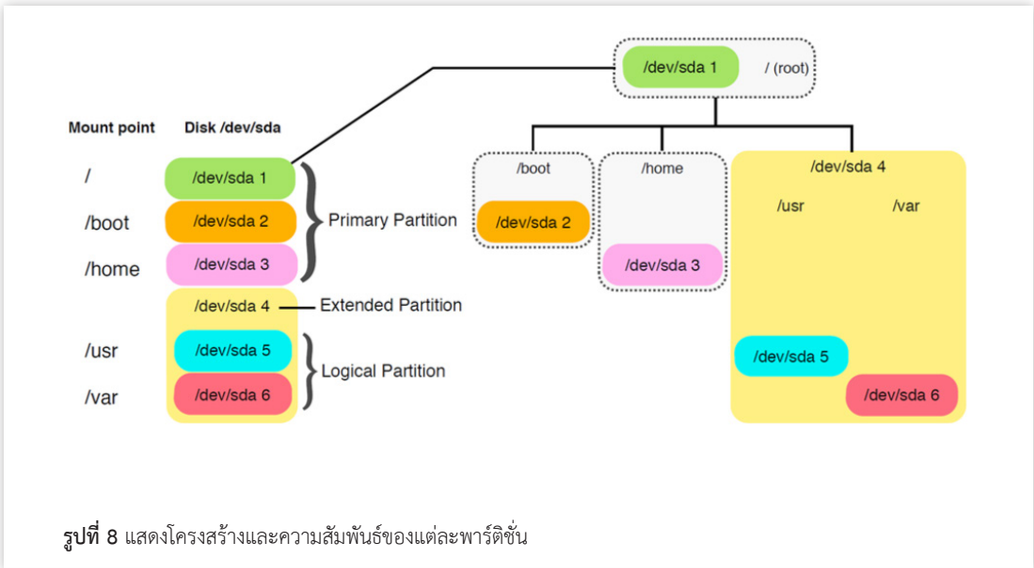
» /dev/sda1 หมายถึง Partition ที่ 1ของ Hard disk ลูกที่ 1

» /dev/sda2 หมายถึง Partition ที่ 2 ของ Hard disk ลูกที่ 1

» /dev/sdb1 หมายถึง Partition ที่ 1 ของ Hard disk ลูกที่ 2

ช่องทางการเชื่อมต่อ (Mount point)

จากข้างต้น เราคงพอทราบว่าระบบปฏิบัติการลินุกซ์ (Linux) นั้นจัดเก็บข้อมูลทั้งหมดภายใต้ Root Directory (/) ดังนั้นการที่จะเข้าถึงข้อมูลที่อยู่ พาร์ติชั่น (Partition) อื่นๆของ ฮาร์ดดิสก์ จึงต้องสร้างช่องทางเชื่อมต่อ โดยเราสามารถสร้างช่องทางเชื่อมต่อของพาร์ติชั่นใดๆ ให้เข้าถึงได้จากที่ใดก็ได้ภายใน Root Directory



รูปที่ 8 แสดงโครงสร้างและความสัมพันธ์ของแต่ละพาร์ติชั่น

จากรูป เราจะทราบได้ว่า Hard disk ที่ 1 มี 5 Partitions โดย

- Partition 1 เป็น Primary Partition ที่มี Mount point คือ Root Directory (/)
- Partition 2 เป็น Primary Partition ที่มี Mount point คือ /boot
- Partition 3 เป็น Primary Partition ที่มี Mount point คือ /home

- Partition 4 เป็น Extended Partition จึงไม่มี Mount point
- Partition 5 เป็น Logical Partition ที่อยู่ใน Partition 4 มี Mount point คือ /usr
- Partition 6 เป็น Logical Partition ที่อยู่ใน Partition 4 มี Mount point คือ /var

คำสั่งเบื้องต้นที่ควรทราบ

หมวดหมู่พื้นฐานไฟล์และไดเรกทอรี

คำสั่ง	หน้าที่	การนำไปใช้
pwd	แสดงที่อยู่แบบเต็มของไดเรกทอรีที่ทำงานอยู่ในขณะนั้น	pwd
ls	ลิสต์ข้อมูลทั้งหมดใน directory ปัจจุบัน	ls
cd	เปลี่ยนจากไดเรกทอรีปัจจุบัน ไปยังไดเรกทอรีอื่น	cd *DIRECTORY_NAME* เช่น cd /home/comcamp/Doucument/Book
cp	คัดลอกข้อมูล (Copy)	cp *SOURCE TARGET* เช่น cp text.txt /home/comcamp/Document/
mv	ย้ายข้อมูล (Move)	mv *SOURCE TARGET* เช่น mv text.txt /home/comcamp/Document/ mv *NAME NEW_NAME* (ความจริงคือ ย้ายไปยังชื่อใหม่)
rm	ลบ (Remove) ลบไฟล์สามารถใช้ rm ได้เลย แต่ถ้าลบ ไดเรกทอรีต้องใช้ rm –r (-r คือ option ของ rm สามารถศึกษาเพิ่มเติมได้)	rm *FILE_NAME* rm –r *DIRECTORY_NAME*
mkdir	สร้างไดเรกทอรี (Make Directory)	mkdir *DIRECTORY_NAME*

Tip

ตำแหน่งปัจจุบัน แทนด้วยเครื่องหมาย . (จุด)
ตำแหน่ง ของไดเรกทอรีที่อยู่ด้านบนขึ้นไป parent directory แทนด้วยเครื่องหมาย ..(2 จุด)
ตัวอย่าง : หากปัจจุบันอยู่ที่ /home/comcamp
cd /home/comcamp/Doucument/Book เท่ากับ cd ./Document/Book
cd /home/comcamp/Doucument/Book เท่ากับ Document/Book (ไม่แนะนำให้ใช้)
cd / home/comcamp/Documentเท่ากับ cd ..
*** เครื่องหมาย . และ .. สามารถนำไปใช้ในการระบุไดเรกทอรี่/ที่อยู่ไฟล์ ได้ทั้งหมด ***

หมวดหมู่เครื่องมือพื้นฐาน

คำสั่ง	หน้าที่	การนำไปใช้
cat	แสดงข้อมูลในไฟล์ Text	cat *FILE_NAME*
nano	แก้ไข/สร้าง ไฟล์ Text	nano *FILE_NAME*
wget	Download ข้อมูลจาก url	wget *URL*
find	ค้นหาไฟล์โดยระบุไดเร็คทอรี	find *DIRECTORY* –name *FILE_NAME*
locate	ค้นหาไฟล์โดยไม่ระบุไดเร็คทอรี	locate *FILE_NAME*
man	คู่มือของคำสั่งต่างๆ	rm *File_NAME* rm –r *DIRECTORY_NAME*

หมวดหมู่การใช้งานทั่วไป

คำสั่ง	หน้าที่	การนำไปใช้
top	แสดงสถานะการทำงานทั้งหมด	top
ifconfig	แสดงข้อมูลการเชื่อมต่อเน็ตเวิร์ค	ifconfig
w	ดูว่ามีใคร login อยู่ และกำลังทำอะไร	w
date	แสดงวัน เวลา ปัจจุบัน	date
cal	แสดงปฏิทิน	cal

Web development

เว็บไซต์หมายถึงอะไร

HTML/CSS/JavaScript/PHP

เว็บไซต์สำเร็จรูป (CMS)

Framework

Server/Domain

Front End/Back End

คำศัพท์ที่ควรรู้

เว็บไซต์หมายถึงอะไร

หมายถึง หน้าเว็บเพจหลายหน้า ซึ่งเชื่อมโยงกันผ่านทางไฮเปอร์ลิงก์ ส่วนใหญ่จัดทำขึ้นเพื่อนำเสนอข้อมูลผ่านคอมพิวเตอร์ โดยถูกจัดเก็บไว้ในเวิลด์ไวด์เว็บ หน้าแรกของเว็บไซต์ที่เก็บไว้ที่ชื่อหลักจะเรียกว่า โฮมเพจ เว็บไซต์โดยทั่วไปจะให้บริการต่อผู้ใช้ฟรี แต่ในขณะเดียวกันบางเว็บไซต์จำเป็นต้องมีการสมัครสมาชิกและเสียค่าบริการเพื่อที่จะดูข้อมูล ในเว็บไซต์นั้น ซึ่งได้แก่ข้อมูลทางวิชาการ ข้อมูลตลาดหลักทรัพย์หรือข้อมูลสื่อต่างๆ ผู้ทำเว็บไซต์มีหลากหลายระดับ ตั้งแต่สร้างเว็บไซต์ส่วนตัว จนถึงระดับเว็บไซต์สำหรับธุรกิจหรือองค์กรต่างๆ การเรียกดูเว็บไซต์โดยทั่วไปนิยมเรียกดูผ่านซอฟต์แวร์ในลักษณะของ เว็บเบราว์เซอร์

หลังจากทราบพื้นฐานของคำว่าเว็บไซต์กันแล้ว จะเห็นได้ว่า เว็บไซต์มีความหมายไม่ได้ลึกอย่างที่คิด แต่ความจริงแล้ว มันประกอบไปด้วยหลายๆส่วนที่ต่างกัน ทำให้เว็บไซต์ต่างๆจึงแตกต่างกัน หรือทำให้เรียกเว็บเหล่านี้ แตกต่างกัน เช่น Web-application Web-blog หรืออีกหลายๆอย่าง

: ต่อมาจะ เป็นการศึกษาคำต่างๆที่นัก เขียน เว็ปควรรู้ติดตัวไว้ :

HTML คืออะไร

HTML คือ ภาษาหลักที่ใช้ในการเขียนเว็บเพจ โดยใช้ Tag ในการกำหนดการแสดงผล HTML ย่อมาจากคำว่า Hypertext Markup Language โดย Hypertext หมายถึง ข้อความที่เชื่อมต่อกันผ่านลิ้ง(Hyperlink) Markup language หมายถึงภาษาที่ใช้ Tag ในการกำหนดการแสดงผลสิ่งต่างๆที่แสดงอยู่บนเว็บเพจ ดังนั้น HTML จึงหมายถึง ภาษาที่ใช้ Tag ในการกำหนดการแสดงผลเว็บเพจที่ต่างก็เชื่อมถึงกันในHyperspace ผ่าน Hyperlink

```
CODE : <html> ,<body>,<br>
ใช้เพื่อ : เขียนพื้นฐานหน้าเว็ป
```

CSS คืออะไร

CSS ย่อมาจาก Cascading Style Sheets เป็นภาษาที่มีรูปแบบการเขียน Syntax ที่เฉพาะ และถูกกำหนดมาตรฐานโดย W3C (World Wide Web Consortium) เช่นเดียวกับ HTML และ XHTML ใช้สำหรับตกแต่งเอกสาร HTML/ XHTML ให้มีหน้าตา สีสัน ตัวอักษร เส้นขอบ พื้นหลัง ระยะห่าง ฯลฯ อย่างที่เราต้องการ ด้วยการกำหนดคุณสมบัติให้กับ Element ต่างๆ ของ HTML เช่น <body>, <p>, <h1>เป็นต้น

```
CODE : p{ color : red;}
ใช้เพื่อ : ตกแต่งหน้าเว็ป
```

Tip

HTML เปรียบเหมือนการแปลงจากภาษาเป็นหน้าตา แต่ยังขาดสีสัน CSS จึงเปรียบเสมือนตัวตกแต่งของ HTML ในการสร้างเว็ป

จาวาสคริปต์ (JavaScript)

มีลักษณะการเขียนแบบโปรโตไทพ (Prototyped-based Programming) ส่วนมากใช้ในหน้าเว็บเพื่อประมวลผลข้อมูลที่ฝั่งของผู้ใช้งาน แต่ก็ยังมีใช้เพื่อเพิ่มเติมความสามารถในการเขียนสคริปต์โดยฝังอยู่ในโปรแกรมอื่นๆ ปัจจุบันมีการใช้จาวาสคริปต์ที่ฝังอยู่ในเว็บเบราว์เซอร์ในหลายรูปแบบ เช่น ใช้เพื่อสร้างเนื้อหาที่เปลี่ยนแปลงเสมอภายในเว็บเพจ, ใช้เพื่อตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้กรอกก่อนนำเข้าระบบ, ใช้เพื่อเข้าถึงข้อมูลที่อยู่ภายใต้โครงสร้างแบบ Document Object Model (DOM)เป็นต้นนอกจากนี้จาวาสคริปต์ยังถูกฝังอยู่ในแอปพลิเคชันต่างๆ นอกเหนือจากเว็บเบราว์เซอร์ได้อีกด้วย เช่น widget ของ ยาฮู! เป็นต้น โดยรวมแล้วจาวาสคริปต์ถูกใช้เพื่อให้นักพัฒนาโปรแกรม สามารถเขียนสคริปต์เพื่อสร้างคุณสมบัติพิเศษต่างๆ เพิ่มเติมจากที่มีอยู่บนแอปพลิเคชันดังเดิม โปรแกรมใดๆ ที่สนับสนุนจาวาสคริปต์จะมีตัวขับเคลื่อนจาวาสคริปต์ (JavaScript Engine) ของตัวเอง เพื่อเรียกใช้งานโครงสร้างเชิงวัตถุของโปรแกรมหรือแอปพลิเคชันนั้นๆ

```
CODE : var a; document.getElementById("test").innerHTML
ใช้เพื่อ : ลูกเล่นบนเว็ป หรือการเขียนสคริปทำงานต่างๆ
```

พีเอชพี (PHP)

ภาษาคอมพิวเตอร์ในลักษณะเซิร์ฟเวอร์-ไซด์ สคริปต์ โดยลิขสิทธิ์อยู่ในลักษณะโอเพนซอร์ส ภาษาพีเอชพี ใช้สำหรับจัดทำเว็บไซต์ และแสดงผลออกมาในรูปแบบ HTML โดยมีรากฐานโครงสร้างคำสั่งมาจากภาษา ภาษาซี ภาษาจาวา และ ภาษาเพิร์ล ซึ่ง ภาษาพีเอชพี นั้นง่ายต่อการเรียนรู้ ซึ่งเป้าหมายหลักของภาษานี้ คือให้นักพัฒนาเว็บไซต์สามารถเขียน เว็บเพจ ที่มีความตอบโต้ได้อย่างรวดเร็ว การแสดงผลของพีเอชพี จะปรากฏในลักษณะHTML ซึ่งจะไม่แสดงคำสั่งที่ผู้ใช้เขียน ซึ่งเป็นลักษณะเด่นที่พีเอชพีแตกต่างจากภาษาในลักษณะไคลเอนต์-ไซด์ สคริปต์ เช่น ภาษาจาวาสคริปต์ ที่ผู้ชมเว็บไซต์สามารถอ่าน ดูและคัดลอกคำสั่งไปใช้เองได้

```
CODE : echo "hello" ;
ใช้เพื่อ : เขียนโค๊ดเบื้องหลัง การทำงานฝั่ง server
```

เว็บไซต์สำเร็จรูป (Content Management System : CMS)

หรือมีชื่อเรียกอีกอย่างว่า “ระบบจัดการเนื้อหา” เป็นเว็บไซต์ที่มีระบบจัดการเนื้อหาที่อยู่บนเว็บไซต์ทั้งหมดผ่านระบบ “Admin” ที่ผู้ดูแลเว็บไซต์ (Webmaster) สร้างขึ้น เพื่ออำนวยความสะดวกในการปรับแต่งข้อความ เนื้อหา Banner หรือส่วนอื่นๆ ตามต้องการ โดยไม่จำเป็นที่จะต้องเปิดไฟล์เว็บไซต์ขึ้นมาเพื่อแก้ไขข้อความและทำการอัพโหลดไฟล์ขึ้นไปบน Server ใหม่อีกครั้ง

เว็บไซต์สำเร็จรูปจึงมีลักษณะพิเศษอยู่ 2 อย่าง คือ

1. มีระบบผู้ดูแลอยู่หลังบ้าน (ระบบ Admin) เพื่อเอาไว้จัดการส่วนต่าง ๆ ของเว็บไซต์
2. ไม่ต้องแก้ไขข้อมูลต่าง ๆ บนเว็บไซต์ด้วยการแก้ไขไฟล์เว็บไซต์

เว็บไซต์สำเร็จรูปมีหลายประเภท ตัวอย่างเช่น:

- CMS ประเภท Blog ได้แก่ Wordpress, Drupal
- CMS ประเภท เว็บบอร์ด ได้แก่ SMF, phpBB
- CMS ประเภท e-Learning เช่น Moodle, Sakai
- CMS ประเภท e-Commerce เช่น Magento, VirtueMart, osCommerce, PhpShop

ประโยชน์ของ CMS

ขึ้นชื่อว่า “เว็บไซต์สำเร็จรูป” ซึ่งข้อดีหรือว่าจุดเด่นของมันเลยก็คือ ความเร็วในการสร้างเว็บไซต์ เพียงแค่ดาวน์โหลด CMS ที่เราต้องการมาอัพขึ้นไปบน Server และทำการติดตั้ง ก็ทำให้เรามีเว็บไซต์ได้ภายในพริบตาโดยที่ไม่ต้องทำอะไรต่อเลย มีระบบจัดการให้ทุก ๆ อย่าง แล้วแต่ CMS ประเภทนั้น ๆ จะเป็นอะไร เช่นถ้า CMS ประเภท Blog ก็จะมีระบบเขียนบทความ พร้อมเครื่องมือช่วย ส่วนเสริมต่าง ๆ ให้คุณได้ใช้งาน เป็นต้น

ข้อเสียของ CMS

CMS เป็นระบบขนาดใหญ่ที่ใช้ผู้พัฒนาหลายคน ฉะนั้นการเขียนระบบค่อนข้างที่จะซับซ้อน เพราะต้องคำนึงถึงความรวดเร็วในการโหลดหน้าเว็บไซต์ ความปลอดภัยของเว็บไซต์และข้อมูล

framework คืออะไร

ในเชิง HTML framework นั้นตามความเข้าใจจะคือโครงสร้างของการเขียนโปรแกรม มีcode ที่วางไว้อย่างเป็นระบบ มีรูปแบบแผน และลักษณะการเขียน เป็นมาตรฐาน ตามโปรแกรมต่างๆครับ ซึ่งจะเหมาะกับการเขียนโปรแกรมที่มีขนาดใหญ่ มีผู้พัฒนาหลายคน เพราะจะช่วยให้ code ที่เขียนเป็นไปในทิศทางเดียวกัน เปรียบเสมือนมี template เดียวกันตัวอย่าง framework Bootstrap , Foundation , AngularJS

server

คือ เครื่องคอมพิวเตอร์หรือระบบปฏิบัติการหรือโปรแกรมคอมพิวเตอร์ ที่ทำหน้าที่ให้บริการอย่างใดอย่างหนึ่งหรือหลายอย่าง แก่เครื่องคอมพิวเตอร์หรือโปรแกรมคอมพิวเตอร์ที่เป็นลูกข่าย ในระบบเครือข่าย ข้อความแบบนี้อาจจะงงอยู่บ้าง สรุปอีกครั้งนะครับ Server ในทาง computer มี 3 ความหมายคือ

เครื่องคอมพิวเตอร์ที่ทำหน้าที่ให้ บริการอะไรบางอย่างแก่คอมพิวเตอร์หรือโปรแกรมคอมพิวเตอร์อื่น

ระบบ ปฏิบัติการคอมพิวเตอร์ที่ทำหน้าที่ให้บริการอะไรบางอย่างแก่คอมพิวเตอร์หรือ โปรแกรมคอมพิวเตอร์อื่น

Domain คืออะไร

Domain ในความหมายทั่วไป หมายถึง พื้นที่ที่ควบคุม หรือ โลกของความรู้ในอินเตอร์เน็ต domain ประกอบด้วย กลุ่มของตำแหน่งเครือข่าย ชื่อ domain จัดโครงสร้างเป็นระดับ โดยระดับบนสุดเป็นการระบุด้านภูมิศาสตร์หรือจุดมุ่งหมายขององค์กร (เช่น .th หมายถึงประเทศไทย .com หมายถึงหน่วยธุรกิจ) ระดับที่สองเป็นชื่อที่ไม่ซ้ำ (Unique) ภายใน Domain ระดับบนสุด และระดับต่ำที่ต้องนำมาใช้ ถ้ากล่าวอย่างชัดเจนแล้ว domain name system ของอินเตอร์เน็ต domain เป็นชื่อที่สามารถบันทึกเป็นชื่อร่วมกับ subdomain หรือ host เช่น widebase.net สามารถบันทึกชื่อ domain เป็นwww.widebase.net และ www1.widebase.net ใน Windows NT และ Windows 2000 ความหมายของ domain หมายถึงกลุ่มของทรัพยากรใน เครือข่ายสำหรับกลุ่มของผู้ใช้ โดยผู้ใช้ต้อง Login เข้าสู่ระบบเพื่อดึงอุปกรณ์ต่าง ๆ ซึ่งอาจจะอยู่ในเครื่องแม่ข่ายคนละเครื่องในระบบเครือข่าย

Front ends

เรียกกันง่าย ๆ ว่าหน้าบ้าน ซึ่งความหมายก็ตรงตัวจริง ๆ ครับ Front ends คือส่วนติดต่อผู้ใช้ (User interface) การแสดงผลต่าง ๆ ภาพแบนเนอร์ เนื้อหา เรื่องราว หากมอง thatsay.com ก็ประกอบด้วย

- ส่วนติดต่อก็คือลิงค์ต่าง ๆ ที่ผู้ใช้งานคลิก
- หน้าจอสำหรับสร้างเรื่องราว(story) ใหม่ ๆ

ในมุมของการพัฒนาเว็บไซต์ก็จะประกอบด้วยหลาย ๆ ภาษาสคริปต์ หรือ มาร์คอัพ เช่น

HTML, HTML5, CSS, Javascript, JSON data, ไฟล์รูปภาพ(.jpg, .png, .gif) หรือภาษาคอมพิวเตอร์ (ฝั่ง Server) ก็เขียน Front End ได้เช่นกัน ที่นิยมก็เช่น PHP,C#,JAVA,VB.NET ก็สามารถแสดงผลได้เช่นกัน (แต่กลุ่มนี้ก็ควบคุมฝั่ง Back Ends ได้ด้วย)

คนที่ทำงานด้าน Front ends ก็ควรจำเป็นต้องมีทักษะด้านนี้ ตัวอย่างตำแหน่งงานที่ตรงกับด้านนี้

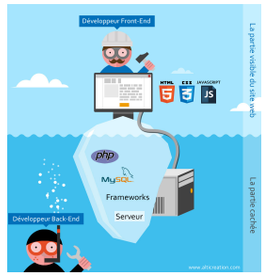
1.Graphic Designer

2.Web Designer

3.Web Developer/Front end Developer (Javascript,CSS,HTML5,JQuery,AngularJS)

Credit : Frontend vs Backend iceburge —http://skillcrush.com/2012/04/17/frontend-vs-backend-3/

Back Ends



เรียกติดปากในหมู่นักพัฒนาระบบ(Developer) นักเขียนโปรแกรม (Programmer)ว่า หลังบ้านคือส่วนจัดการกับข้อมูล

หากมองในมุมการเขียนโปรแกรม ก็คือ ฐานข้อมูล(Database) นั่นเอง ซึ่ง Database ก็ขึ้นอยู่กับนักพัฒนาเองว่า จะเก็บอะไรบ้าง เช่น ข้อมูลสมาชิก(Members) ข้อมูลสินค้า(Products) ข้อมูลการสั่งซื้อ(Order)

นอกจาก Database แล้วยังรวมไปถึงการจัดการไฟล์ข้อมูลต่าง ๆ ไม่ว่าจะเป็น XML,Text File รวมไปถึง Cloud Storage ด้วย ซึ่ง Cloud กำลังเริ่มได้รับความสนใจมากขึ้นเรื่อย ๆ และคำถามคือ แล้ว

Tip

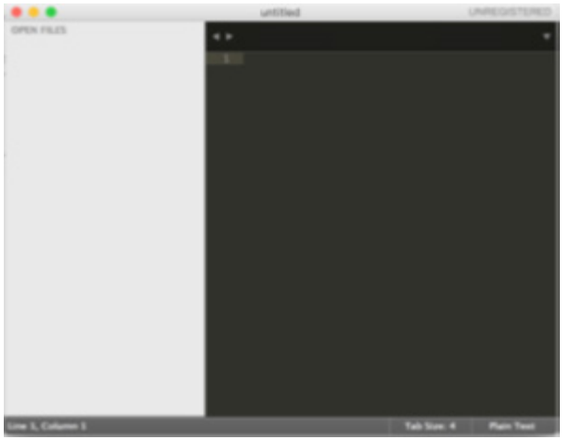
ส่วนไหนสำคัญกว่ากัน ? คำตอบต้องขึ้นอยู่กับตัวระบบ หรือเว็บไซต์ที่พัฒนามากกว่า หากเน้นเก็บข้อมูลไม่เยอะ และ แสดงผลโดยใช้ไฟล์ Html มากกว่า ความสำคัญก็จะตกอยู่กับ Front ends เพราะ Design ให้ออกมาสวยและถูกใจคน เข้าชมแต่หากข้อมูลระดับเยอะ ๆ เช่นมากกว่า 1 แสนรายการ (มองระดับ record ของ Database) ความยากก็จะตกอยู่กับฝั่งหลังบ้าน ว่าจะจัดการข้อมูลเหล่านี้อย่างไร ที่ไม่ทำให้ระบบช้าลงในการแสดงผล

หลังจากเรารู้คำศัพท์ต่างๆที่ใช้บนเว็บไซต์ หรือใช้สร้างเว็บไซต์เหล่านี้ไปแล้ว ต่อมาสิ่งที่ควรรู้เบื้องต้นในการเขียนเว็บคือ ภาษา HTML เพราะเป็นภาษาที่ง่ายที่สุด และเป็นภาษาพื้นฐานของเว็บ หรือพูดง่ายๆ คือ ทุกเว็บต่างอยู่ในการแสดงของ HTML ทั้งสิ้น หลังจากนั้นจึงเป็นส่วนของ CSS ซึ่งใช้ตกแต่งหน้าเว็บให้สวยงาม ต่อมาเมื่อใช้ HTML + CSS ได้แล้ว จะสามารถเขียนเว็บได้จำนวนมาก เพราะหลายๆเว็บต่างอยู่บนพื้นฐาน 2 ภาษานี้ทั้งสิ้น แต่หากต้องการจะพัฒนาฝีมือต่อ จะเริ่มเข้าสู่การเขียนหลังบ้าน หรือ Back Ends ซึ่งใช้ ภาษา PHP หรืออื่นๆ ซึ่งจะสามารถเชื่อมต่อกับฐานข้อมูลได้และหากยังสนใจต่อไป ก็สามารถที่จะดูถึงการทำงานพิเศษๆบนหน้าเว็ป ด้วยภาษา Javascript เพื่อที่จะทำให้เว็ปเรามีลูกเล่นมากขึ้น สุดท้ายเมื่อเก็ยพื้นฐานทุกอย่างครบแล้วจึงสามารถหาความรู้อื่นๆ เช่น

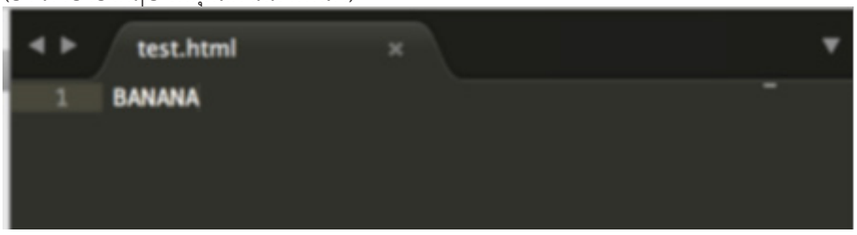
- Framework ต่างๆ
- CMS
- Server admin

Lab 1 เริ่มต้น

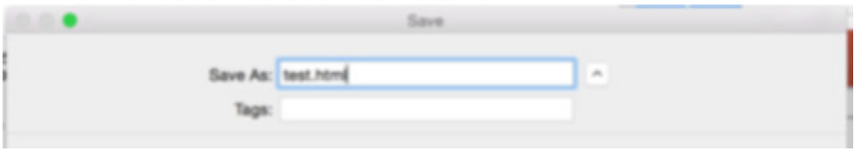
1. เปิด Sublime text หรือ notepad



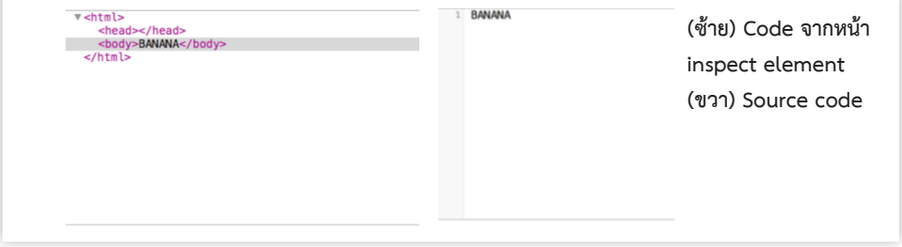
2. พิมชื่อตัวเองภาษาอังกฤษ
(ย้ำ ภาษาอังกฤษ เหตุผลตามมาทีหลัง)



3. กด ctrl + s หรือ ไปกด save as ตั้งชื่อว่า test.html



ผลลัพธ์



(ซ้าย) Code จากหน้า inspect element
(ขวา) Source code

Lab 2 HTML

1. แก้โค๊ดจาก lab 1 เป็นดังนี้

```
<!DOCTYPE html>
<html>
<head>
<title>Test lab2</title>
<meta charset="UTF-8">
</head>
<body>
<h1>หัวข้อ 1</h1>
<h2>หัวข้อ 2</h2>
<p>ทดสอบเนื้อหา <br> ทดสอบเนื้อหา ทดสอบเนื้อหา</p>
</body>
</html>
```

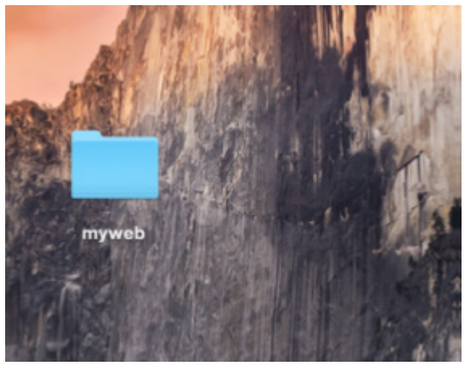
Tip

<meta charset> คือการเปลี่ยนการแสดงผลทางภาษาบนหน้าเว็ป
UTF-8 สากล สามารถรองรับภาษาไทยได้
การเขียนโดยการสร้าง Element ให้ถูกต้องและ แบ่งเป็นส่วนๆให้ดี จะทำให้ บราวเซอร์แสดงผลที่เหมาะสม และหากในระดับสูงๆ จะเหมาะกับการใช้ framework และ SEO*
*การทำให้เว็ปติด Search Engine เช่น Google

(บน) ผลลัพธ์ที่ได้
(ล่าง) Source code

Lab3 เว็ปไซต์แรก ง่ายสุดๆไปเลย

1. สร้าง folder บน Desktop ว่า myweb



2. พิมพ์โค๊ดเหล่านี้ลงในโปรแกรมเขียน

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test lab3</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h1>หัวขข้อ</h1>
        <p>ทดสอบเนื้อื้อหา</p>
        <a href="http://www.google.com"> ลิงค์ไปยังเว็ป google </a>
        <br>
        <img src="pic_1.jpg" height="100" width="100">
        <br>
        <table>
            <th>
                <td>หัวขข้อ 1</td>
                <td>หัวขข้อ 2</td>
            </th>
            <tr>
                <td>ช่อง 1</td>
                <td>ช่อง 2</td>
            </tr>
        </table>
        <div>
            ข้างในนี้คือกล่อง
        </div>
        <ul>
            <li>list 1</li>
            <li>list 2</li>
            <li>list 3</li>
        </ul>
    </body>
</html>

```

3. ตั้งชื่อว่า index.html

Lab4 CSS

1. เขียนโค๊ดตามตัวอย่างข้างล่าง

```html
<html>
    <head>
        <title>Test lab4</title>
        <meta charset="UTF-8">
        <style>
            h1{
                color:blue
            }
        </style>
    </head>
    <body>
        <h1>หัวข้อ</h1>
        <p style="color:red">ทดสอบเนื้อหา</p>
        <a href="index.html"> ไปหน้า 1 </a>
    </body>
</html>
```

2. save ไฟล์ ตั้งชื่อว่า page2.html เก็บไว้ใน folder เดียวกับ html

Lab5 My web

ให้ออกแบบเว็ปของตัวเองเป็นกลุ่ม 4-5 คน โดยดูโค๊ดต่างๆได้จาก folder cod_web ใน desktop
โดยเนื้อหาที่ต้องมีคือ
ชื่อกลุ่ม เรื่องที่ต้องการนำเสนอ สมาชิกทุกคน
ให้เวลา 1 ชม.

Tip

สามารถหาโค๊ดจากเว็บอื่นๆได้
แหล่งข้อมูลเพิ่มเติม - www.w3schools.com

Note

Note

Note

Note