

ภาคผนวก K

การทดลองที่ 11 การเชื่อมต่ออินพุต-เอาต์พุตกับ สัญญาณอินเทอร์รัปต์

การทดลองนี้ คาดว่าผู้อ่านเคยเรียนการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C และ แอสเซมบลี จาก
การทดลองก่อนหน้านี้ ดังนั้น การทดลองนี้มีวัตถุประสงค์เหล่านี้

- เพื่อพัฒนาการทำงานของอินเทอร์รัปต์ร่วมโปรแกรมภาษา C และ แอสเซมบลี ตามเนื้อหาในหัวข้อ
ที่ [6.12](#)
- เพื่อศึกษาการทำงานของอินเทอร์รัปต์ร่วมกับขา GPIO ตามเนื้อหาในหัวข้อที่ [6.11](#)

K.1 การจัดการอินเทอร์รัปต์ของ WiringPi

ไลบรารี wiringPi รองรับการทำอินเทอร์รัปต์ของ GPIO ได้ ทำให้โปรแกรมหลักสามารถทำงานหลัก
ได้ตามปกติ เมื่อเกิดสัญญาณอินเทอร์รัปต์ขึ้น ไม่ว่าจะเป็นสัญญาณจากการกดปุ่มสวิตช์ ทำให้เกิดขอบขา
ขึ้นหรือขอบขาลงหรือทั้งสองขอบ โดยการเรียกใช้คำสั่ง

```
wiringPiISR(pin, edgeType, &callback)
```

โดย pin หมายถึง เลขขาที่ wiringPi กำหนด edgeType กำหนดจากค่าคงที่ 4 ค่านี้

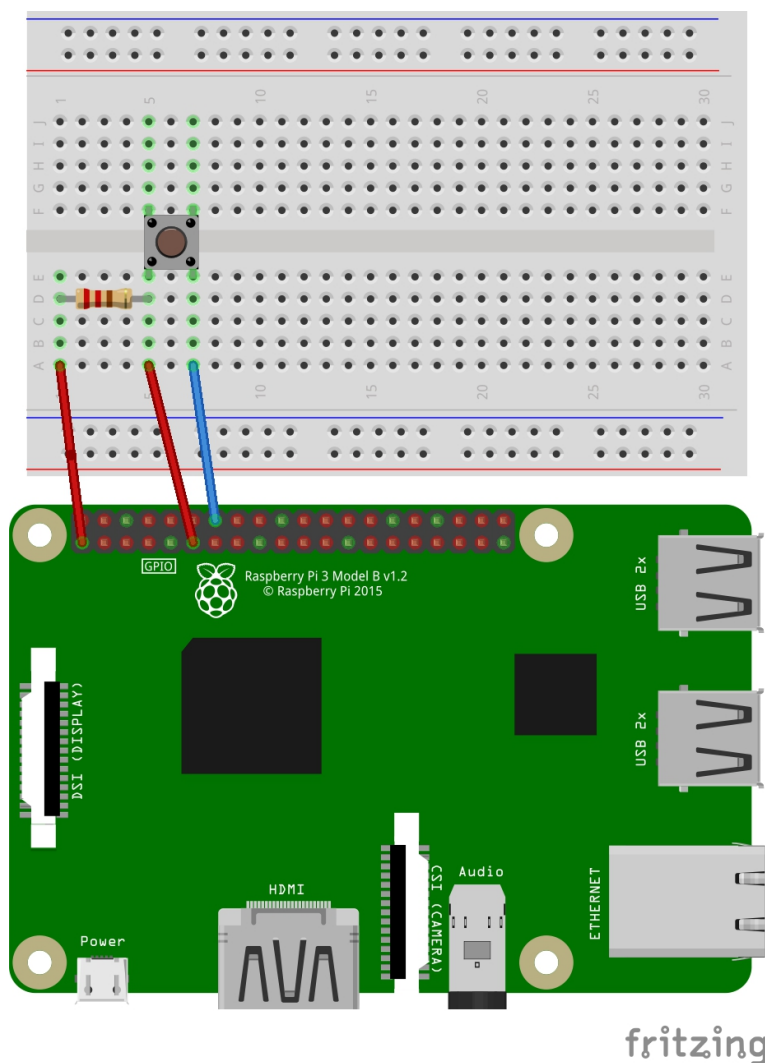
- INT_EDGE_FALLING,
- INT_EDGE_RISING,
- INT_EDGE_BOTH
- INT_EDGE_SETUP.

การกำหนดชนิดขอบขาเป็น 3 ชนิดแรก ไบรารีจะตั้งค่าเริ่มต้น (Initialization) ให้โดยอัตโนมัติ หากกำหนดชนิดขอบขาเป็น INT_EDGE_SETUP ไบรารีจะไม่ตั้งค่าเริ่มต้น (Initialization) ให้ เนื่องจากโปรแกรมเมอร์จะต้องกำหนดเอง

พารามิเตอร์ **callback** คือ ชื่อฟังก์ชันที่จะทำหน้าที่เป็น ISR สัญลักษณ์ & หมายถึง แอดเรสของฟังก์ชัน callback ฟังก์ชัน callback นี้จะเริ่มต้นทำงานโดยแจ้งต่อวงจร Dispatcher ในหัวข้อที่ 6.12 ก่อนจะเริ่มต้นทำงาน โดยฟังก์ชัน callback จะสามารถอ่าน หรือเขียนค่าของตัวแปรโกลบอลในโปรแกรมได้ ซึ่งตัวอย่างการทำงานจะได้กล่าวในหัวข้อถัดไป

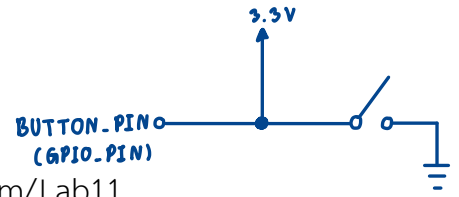
K.2 วงจรสวิตช์ปุ่มกดเชื่อมผ่านขา GPIO

1. ชัตดาวน์และตัดไฟเลี้ยงออกจากบอร์ด Pi เพื่อความปลอดภัยในการต่อวงจร
2. ต่อวงจรตามรูปที่ K.1



รูปที่ K.1: วงจร สวิตช์ ปุ่ม กด สำหรับ ทดลอง การ เขียนโปรแกรม อินเทอร์รัปต์ ในการ ทดลองที่ 11 ที่มา: fritzing.org

3. จงวาดวงจรที่ต่อในรูปที่ K.1 ประกอบด้วย สวิตช์ปุ่มกด ตัวต้านทาน ไฟเลี้ยง 3.3 โวลต์ ขา BUT-TON_PIN และกราวด์ (0 โวลต์)
4. ตรวจสอบความถูกต้อง โดยให้ผู้ควบคุมการทดลองตรวจสอบ
5. สร้าง project ใหม่ชื่อ Lab11 ภายใต้ไดเรกทอรี /home/pi/asm/Lab11



K.3 โปรแกรมภาษา C สำหรับทดสอบวงจรอินเทอร์รัปต์

ผู้อ่านต้องทำความเข้าใจกับตัวโปรแกรม ก่อนคอมไพล์หรือรันโปรแกรม เพื่อความเข้าใจสูงสุด โดยเฉพาะชื่อตัวแปร ชนิดของตัวแปร evenCounter การติดตั้งฟังก์ชัน wiringPiISR เพื่อเชื่อมโยงกับขา GPIO ชนิดของการตรวจจับ และชื่อฟังก์ชัน myInterrupt ซึ่งทำหน้าที่เป็น ISR หรือ ฟังก์ชัน callback

```
#include <stdio.h>
#include <errno.h>
#include <wiringPi.h>
#define BUTTON_PIN 0
// Use GPIO Pin 17 = Pin 0 of wiringPi library
volatile int eventCount = 0;
void myInterrupt(void) { // called every time an event occurs
    eventCount++; // the event counter
}
int main(void) {
    if (wiringPiSetup() < 0) // check the existence of wiringPi library
    {
        printf ("Cannot setup wiringPi: %s\n", strerror (errno));
        return 1; // error code = 1
    }
    // set wiringPi Pin 0 to generate an interrupt from 1-0 transition
    // myInterrupt() = my Interrupt Service Routine
    if (wiringPiISR (BUTTON_PIN, INT_EDGE_FALLING, &myInterrupt) < 0) {
        printf ("Cannot setup ISR: %s\n", strerror (errno));
        return 2; // error code = 2
    }
    // display counter value every second
    while(1) {
        printf("%d\n", eventCount);
```

```

    eventCount = 0;
    delay(1000); // wait 1000 milliseconds = 1 second
}
return 0; // error code = 0 (No Error)
}

```

1. ป้อนโปรแกรมด้านบนใน main.c โดยใช้โปรแกรม Text Editor ทั่วไป
2. สร้าง makefile สำหรับ คอมไพเลอร์ และ ลิงก์โปรแกรม จากการทดลองก่อนหน้านี้ จนไม่เกิดข้อผิดพลาด
3. รันโปรแกรม ทดสอบการทำงานด้วยการกดปุ่มสวิตช์ที่ต่อไว้ สังเกตผลลัพธ์ทางหน้าจอ Terminal ที่รัน

```
$ sudo ./Lab11
```

K.4 กิจกรรมท้ายการทดลอง

1. จงวาดสัญญาณที่ขา BUTTON_PIN ก่อนกดปุ่มสวิตช์ ระหว่างกดปุ่มสวิตช์ และปล่อยมือจากสวิตช์ปุ่มกด โดยให้แกนนอนเป็นแกนเวลา แกนตั้งเป็นค่าโวลเตจ หรือ ค่าลอจิกของขาสัญญาณ BUTTON_PIN
2. จงบอกความหมายและการประยุกต์ใช้งานตัวแปรชนิด volatile
3. ปรับแก้ volatile ออกเหลือแค่ int eventCount = 0;
make แล้วจึงรันโปรแกรมทดสอบการทำงานด้วยการกดปุ่มสวิตช์ที่ต่อไว้ สังเกตผลลัพธ์ทางหน้าจอ Terminal ที่รัน เปรียบเทียบการทำงานของโปรแกรมก่อนและหลังการปรับแก้ และหาเหตุผล
4. จงปรับแก้โปรแกรมที่ทดลองตามประโยคต่อไปนี้

```

if (wiringPiISR (BUTTON_PIN, INT_EDGE_RISING, &myInterrupt) < 0) {
    ...
}

```

ทำ make ใหม่และทดลองกดปุ่มสวิตช์ สังเกตการเปลี่ยนแปลงและอธิบาย

5. จงตอบคำถามจากประโยคต่อไปนี้

```

if (wiringPiISR (BUTTON_PIN, INT_EDGE_FALLING, &myInterrupt) < 0) {
    ...
}

```

- ฟังก์ชัน wiringPiISR ทำหน้าที่อะไร เหตุใดอยู่ในประโยคเงื่อนไข if
- ตัวแปร &myInterrupt คืออะไร เหตุใดจึงมีสัญลักษณ์ & นำหน้า
- ฟังก์ชันนี้เชื่อมโยงกับตารางที่ 6.6 อย่างไร

6. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 10 นับขึ้นจาก 0-7-0 โดยเพิ่มสวิตช์ปุ่มกดในการทดลองนี้ และเพิ่มฟังก์ชันการอินเทอร์รัปต์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะให้ความเร็วในการนับเพิ่มขึ้น หรือ Delay สั้นลงครึ่งหนึ่ง เมื่อกดครั้งที่ 2 จะสั้นลงอีกครั้งหนึ่ง เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น
7. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 10 แต่นับลงจาก 7-0-7 โดยเพิ่มสวิตช์ปุ่มกดในการทดลองนี้ และเพิ่มฟังก์ชันการอินเทอร์รัปต์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะให้ความเร็วในการนับลดลง หรือ Delay เพิ่มขึ้นเท่าตัว เมื่อกดครั้งที่ 2 Delay เพิ่มขึ้นอีกเท่าตัว เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น

5

- รับค่าใน pin ที่กำหนดใน function ถ้ามีการ interrupt ก็จะทำ return function ตามที่กำหนดไว้ และต่อเงื่อนไข if เพื่อ check interrupt และ error
- &myinterrupt คือชื่อ ค่าที่อยู่ของ callback function และมี & เพราะไป get address ของ function
- ตรวจสอบ falling edge บน GPIO PIN ของ BUTTON_INPUT (เชื่อมโยงกับ register GPFEN ในตาราง 6.6

ภาคผนวก L

การทดลองที่ 12 การศึกษาอุปกรณ์เก็บรักษา ข้อมูลและระบบไฟล์

การทดลองนี้อธิบายและเชื่อมโยงเนื้อหาความรู้ของทุกบทเข้าด้วยกัน แต่จะเน้นบทที่ 6 และบทที่ 7 เพื่อให้ผู้อ่านมองเห็นอุปกรณ์อินพุตและเอาต์พุตเหมือนไฟล์แต่ละไฟล์ โดยมีวัตถุประสงค์ดังนี้

- เพื่อให้เข้าใจการวัดขนาดของไฟล์และไดเรกทอรีในระบบไฟล์
- เพื่อให้รู้จักโครงสร้างและระบบไฟล์ของการดหน่วยความจำไมโคร SD ที่ใช้งานในปัจจุบัน
- เพื่อให้เข้าใจระบบไฟล์ (File System) ชนิดต่าง ๆ บนบอร์ด Pi
- เพื่อให้สามารถเชื่อมโยงอุปกรณ์อินพุต/เอาต์พุตชนิดต่าง ๆ กับระบบไฟล์

L.1 ขนาดของไฟล์และไดเรกทอรี

ผู้อ่านสามารถตรวจสอบขนาดของไฟล์ใด ๆ ชื่อ filename ที่แท้จริง หน่วยเป็นไบต์ ด้วยคำสั่ง `du` (Disk Usage) โดยทำตามขั้นตอนต่อไปนี้

- ย้ายไดเรกทอรีปัจจุบันไปที่ `/home/pi` ซึ่งเป็นไดเรกทอรีหลักของผู้ใช้ชื่อ pi

```
$ cd /home/pi
```

- สร้างไฟล์ข้อความ `test.txt` ด้วยโปรแกรม `nano` ด้วยคำสั่งต่อไปนี้

```
$ nano test.txt
```

พิมพ์ข้อความ `fdd` ลงในไฟล์ ทำการ Write โดยกดปุ่ม `Ctrl` แห่ตามด้วยปุ่ม `o` ออกจากโปรแกรม โดยกดปุ่ม `Ctrl` แห่ตามด้วยปุ่ม `x`

- คำสั่ง 'du -b filename' จะแสดงผลขนาดเป็นจำนวนไบต์นำหน้าชื่อไฟล์นั้น

```
$ du -b test.txt
```

```
4 test.txt
```

ตัวเลข 4 หมายถึง เลขจำนวนไบต์ที่คำสั่ง du แสดงผลมาตามพารามิเตอร์ b ที่ส่งไป เพื่อบอกค่าขนาดของไฟล์ test.txt เป็นจำนวน 4 ไบต์

- คำสั่ง 'du -B1 filename' ผู้อ่านสามารถตรวจสอบขนาดของไฟล์ใด ๆ ชื่อ filename ที่จัดเก็บเป็นจำนวนเท่าของ 2 ไบต์ ในอุปกรณ์เก็บรักษาข้อมูล SD ด้วยคำสั่งต่อไปนี้

```
$ du -B1 test.txt
```

```
4096 test.txt
```

ตัวเลข 4096 หมายถึง เลขจำนวนไบต์ที่คำสั่ง du แสดงผลมาตามพารามิเตอร์ B1 ที่ส่งไป โดยผู้อ่านจะสังเกตเห็นความแตกต่าง ถึงแม้ไฟล์มีข้อมูลจำนวนน้อยเพียงไม่กี่ไบต์ แต่การจองพื้นที่ในอุปกรณ์สำรองจะมีขนาดเป็นจำนวนเท่าของ 2 ไบต์เสมอ เช่น 8192, 16384 เป็นต้น

- คำสั่ง 'du -h' จะแสดงผลขนาดหรือจำนวนไบต์โดยใช้หน่วยเช่น K (Kibi: 1024) M (Mebi: 1048576) G (Gibi: 1073741824) นำหน้าชื่อไดเรกทอรีหรือโฟลเดอร์ที่อยู่ใต้ไดเรกทอรีปัจจุบัน และจัดบันทึก 5 รายการแรกในตาราง

```
$ du -h
```

Size	Folder Name
84 k	<i>./asm/Lab 7</i>
56 k	<i>./asm/Lab 8</i>
8.0k	<i>./asm/Lab10/Lab10/obj/Debug</i>
12k	<i>./asm/Lab10/Lab10/obj</i>
16k	<i>./asm/Lab10/Lab10/bin/Debug</i>

L.2 ระบบไฟล์

ผู้ใช้หรือผู้ดูแลระบบลินุกซ์ สามารถตรวจสอบการใช้งานอุปกรณ์เก็บรักษาข้อมูล เช่น ฮาร์ดดิสก์ไดรฟ์ โซลิดสเตทไดรฟ์ การ์ดหน่วยความจำ SD ได้โดยคำสั่ง

- คำสั่ง **df** (Disk File System) สามารถแสดงรายละเอียดของอุปกรณ์เก็บรักษาข้อมูลในเครื่อง
- คำสั่ง '**df -h**' จะแสดงรายการ ดังต่อไปนี้ จดบันทึก 5 รายการแรกลงในตารางเพื่อเปรียบเทียบกับตารางที่แล้ว

```
$ df -h
```

Filesystem	Size	Used	Available	Use%	Mounted on
/dev/root	15G	3.7G	11G	26 %	/
devtmpfs	333M	0	333M	0%	/dev
tmpfs	462M	0	462M	0%	/dev/shm
tmpfs	185M	1.2M	184M	1%	/run
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock

โดย Size จะแสดงผลขนาดหรือจำนวนไบต์โดยใช้ตัวคูณที่แตกต่างกัน เช่น K (Kibi: 1024) M (Mebi: 1048576) G (Gibi: 1073741824)

- คำสั่ง '**df -T**' จะเพิ่มคอลัมน์ชนิด (Type) ของแต่ละรายการในการแสดงผล และขนาดเป็นจำนวนเท่าของ 1 KiB (KibiByte) (1K) แทน จดบันทึก 5 รายการที่ตรงกับตารางที่แล้ว

```
$ df -T
```

Filesystem	Type	1K-blocks Used	Available	Use%	Mounted on
/dev/root	ext4	14978128	10643664	26%	/
devtmpfs	devtmpfs	340548	340548	0%	/dev
tmpfs	tmpfs	472132	472132	0%	/dev/shm
tmpfs	tmpfs	188856	187668	1%	/run
tmpfs	tmpfs	5120	5116	1%	/run/lock

- คำสั่ง 'df -i' จะแสดงรายการต่าง ๆ ดังนี้ จดบันทึก 5 รายการที่ตรงกับตารางที่แล้ว

```
$ df -i
```

Filesystem	Inodes	IUsed	IFree	IUse%	Mounted on
/dev/root	950976	123838	827138	14%	/
devtmpfs	85137	474	84663	1%	/dev
tmpfs	118033	1	118032	1%	/dev/shm
tmpfs	819200	762	818438	1%	/run
tmpfs	118033	3	118030	1%	/run/lock

โดยคอลัมน์ที่ 2 จากทางซ้ายจะแสดงผลเป็นจำนวน **ไอโนด** แทน รายละเอียดเรื่องไอโนด ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้ในบทที่ 7 และทาง [wikipedia](https://en.wikipedia.org/wiki/Inode)

- คำสั่ง **stat** แสดงรายละเอียดของไฟล์หรือไดเรกทอรี การทดลองนี้จะใช้ไดเรกทอรี `asm` ที่มีอยู่และเติมตัวเลขในช่องว่าง

```
$ cd /home/pi
```

```
$ stat asm
```

```
File: asm
Size: 4 0 9 6      Blocks: 8      IO Block: 4 0 9 6  directory
Device: b 3 0 2 h/4 5 8 2 6 d Inode: 2 6 0 7 1 8      Links: 6
Access: ( 0 7 5 5/      drwxr-xr-x)  Uid: ( 1 0 0 0/
  _pi _)  Gid: ( 1 0 0 0/_pi _)
Access: ... 2023-01-30 17:27:01.074682918 +0700
Modify: ... 2023-04-05 19:45:13.709999916 +0700
Change: ... 2023-04-05 19:45:13.709999916 +0700
Birth: - 2023-01-30 17:27:01.074682918 +0700
```

ผู้อ่านจะต้องกรอกผลลัพธ์ในช่องว่าง ดังต่อไปนี้

- ชื่อ `asm`
- ขนาด 4 0 9 6 ไบต์ ใช้พื้นที่จำนวน 8 Blocks ซึ่งหมายถึง 8 เซ็กเตอร์ ๆ ละ 512 ไบต์ คิดเป็น 4 0 9 6 ไบต์ เป็น directory

- มีหมายเลข Device = b302_h/45826d หรือเท่ากับ b302₁₆/45826₁₀
- มีหมายเลข Inode = 262098₁₀ จำนวน 6 Links
- สิทธิ์เข้าถึง (Access Permission) ด้วยรหัส 0755₁₆ หรือ 111₂:101₂:101₂ โดยผู้ใช้หมายเลข Uid (User ID)= 1000 ชื่อผู้ใช้ (Username)= pi ในกรุปหมายเลข Groupid= 1000 ชื่อกรุป pi
- เข้าถึง (Access) ... 2023-01-30 17:27:01.07482918 +0700
- เปลี่ยนแปลง (Modify) ... 2023-04-05 19:45:13.709999916 +0700
- เวลาที่ Inode เปลี่ยนแปลง (Change) ... 2023-01-30 17:27:01.074682918 +0700

เบื้องต้นผู้เขียนขอให้ผู้อ่านสร้างไฟล์ผลลัพธ์จากคำสั่ง stat ไปเก็บในไฟล์ เพื่อมาใช้ประกอบการทดลองต่อไป โดย

```
$ stat asm > stat_asm.txt
```

หลังจากนั้น เราสามารถตรวจสอบสถานะของไฟล์ stat_asm.txt ได้ดังนี้

```
$ stat stat_asm.txt
```

```
File: stat_asm.txt
Size: 375      Blocks: 8      IO Block: 4096      regular file
Device: b302_h/45826d      Inode: 262098      Links: 1
Access: (0755/-rw-r--r--)  Uid: (1000/pi)      Gid: (1000/pi)
Access: ... 2023-04-05 22:49:39.7462085671 +0700
Modify: ... 2023-04-05 23:51:41.3698725912 +0700
Change: ... 2023-04-05 23:51:41.3698725912 +0700
Birth: - 2023-04-05 22:49:39.7462085671 +0700
```

ผู้อ่านจะต้องกรอกผลลัพธ์ในช่องว่าง ดังต่อไปนี้เป็นรายบรรทัด

- ชื่อ stat_asm.txt
- ขนาด 375 ไบต์ ใช้พื้นที่จำนวน 8 Blocks ซึ่งหมายถึง 8 เซ็กเตอร์ ะ 512 ไบต์ คิดเป็น 4096 ไบต์ เป็น regular file
- มีหมายเลข Device = b302_h/45826d หรือเท่ากับ b302₁₆/45826₁₀

- มีหมายเลข Inode = 744₁₀ จำนวน 1 Links
- สิทธิ์เข้าถึง (Access Permission) ด้วยรหัส 0644₁₆ หรือ 110₂:100₂:100₂ โดยผู้ใช้หมายเลข Uid (User ID)= 1000 ชื่อผู้ใช้ (Username)= pi ในกรุปหมายเลข Groupid= 1000 ชื่อกรุป pi
- เข้าถึง (Access) ... 2023-04-05 22.49.39.7462085671 +0700
- เปลี่ยนแปลง (Modify) ... 2023-04-05 23.51.41.3498725912 +0700
- เวลาที่ Inode เปลี่ยนแปลง (Change) ... 2023-04-05 22.49.39.7462085671 +0700
- หมายเลข Inode ของ asm กับ หมายเลข Inode ของ stat_asm.txt ตรงกันหรือไม่ เพราะเหตุใด
ไม่ตรงกัน :: inode เป็นหมายเลขประจำตัว /filevo directory นั้นๆ
- asm เป็น ไดรেকทอรี ในขณะที่ stat_asm.txt เป็น regular file
- สิทธิ์เข้าถึง (Access Permission) รหัส 0765₁₆ มีความหมายดังต่อไปนี้
 - 111₂: เป็นของใคร เจ้าของ
 - 110₂: เป็นของใคร กลุ่ม
 - 101₂: เป็นของใคร คนอื่น

L.3 อุปกรณ์อินพุต/เอาต์พุตในระบบไฟล์

การทดลองในหัวข้อนี้จะเชื่อมต่อกับเนื้อหาในบทที่ 3 และ การทดลองที่ 4 ภาคผนวก D หลักการของระบบปฏิบัติการยูนิกซ์ คือ การ**เมาท์** (Mount) อุปกรณ์กับไดเรกทอรีด้วยระบบไฟล์ (File System) ที่แตกต่างกัน โดยใช้ชื่อไดเรกทอรีที่แตกต่างกัน โดยมีไดเรกทอรีรูท (Root Directory) หรือโพลเดอร์รูท เป็นตำแหน่งเริ่มต้น ผู้อ่านสามารถพิมพ์คำสั่งใน Terminal

```
$ mount
```

คำสั่งนี้จะแสดงรายชื่อการเมาท์ หรือ ผูกยึด อุปกรณ์อินพุต/เอาต์พุต เข้ากับ ระบบไฟล์ที่เหมาะสมกับอุปกรณ์นั้น ๆ ด้วยชื่อไดเรกทอรีหรือชื่อไฟล์ของระบบปฏิบัติการ ผู้อ่านจะต้องกรอกผลลัพธ์ที่สำคัญในช่องว่าง และศึกษาคำอธิบายต่อไปนี้

- /dev/mmcblk0p2 on / type ext4 (rw,noatime) เป็นระบบไฟล์ **ext4** ซึ่งเป็นระบบไฟล์หลักของลินุกซ์ ย่อมาจากคำว่า Fourth Extended File System เป็นเวอร์ชันที่ 4 พัฒนาจากชนิด ext3 ซึ่งพัฒนาจากระบบยูนิกซ์ตามรายละเอียดในหัวข้อที่ 7.1 และ [wikipedia](https://en.wikipedia.org/wiki/Ext4)

- devtmpfs on /dev type devtmpfs (rw, reltime, size=3834564k, nr_inodes=958641, mode=755)
- proc on /proc type proc (rw, reltime) เป็นระบบไฟล์เสมือน (Virtual File System) สำหรับระบบสำคัญต่าง ๆ เช่น CPU, โดยจะสร้างขึ้นเมื่อบูตเครื่อง และลบทิ้งเมื่อชัตดาวน์ระบบ [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,reltime) เป็นระบบไฟล์เสมือน (Virtual File System) [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- securityfs on /sys/kernel/security type securityfs (rw, nosuid, nodev, noexec, reltime)
- tmpfs on /dev/shm type tmpfs (rw, nosuid, nodev) ย่อมาจากคำว่า Temporary File System [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- devpts on /dev/pts type devpts (rw, nosuid, noexec, reltime, gid=5, mode=620, ptmxmode=000) เป็นระบบไฟล์เสมือน (Virtual File System) สำหรับอุปกรณ์อินพุตเอาต์พุตต่าง ๆ [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- ...
- /dev/mmcblk0p_1 on /boot type vfat ระบบไฟล์ **vfat** เป็นส่วนต่อขยายของระบบไฟล์ FAT ซึ่งย่อมาจากคำว่า File Allocation Table เพื่อรองรับชื่อไฟล์ที่ยาวกว่า FAT ที่มา: [wikipedia](#)
- ...

รายชื่อต่อไปนี้ คือ ตัวเลือกคุณสมบัติ (Attribute) ที่สำคัญของระบบไฟล์ เช่น

- rw : read/write สามารถอ่านและเขียนได้
- noatime และ atime: No/ Access Time หมายถึง ไม่มี/มีการบันทึกเวลาในการสร้าง อ่านหรือเขียนไฟล์ทุกครั้ง
- reltime หมายถึง มีการบันทึกเวลาในการสร้าง อ่านหรือเขียนไฟล์ เมื่อเกิดการแก้ไขไฟล์ หรือการอ่านหรือเข้าถึงไฟล์มากกว่าเวลาที่บันทึกไว้ก่อนหน้านี้อย่างน้อย 24 ชั่วโมง
- nosuid: No SuperUser ID เป็นการป้องกันไม่ให้ผู้ดูแลระบบ (SuperUser) กระทำการใด ๆ ได้เพื่อความมั่นคงปลอดภัย
- noexec: No Execution เพื่อตั้งค่าไม่ให้รันไฟล์ที่อยู่ในไดเรกทอรีนี้ได้ เช่น ไฟล์ที่เป็นไวรัสหรือมัลแวร์ (Malware) ที่แอบแฝงเข้ามา
- nodev: No Device หมายถึง การไม่อนุญาตให้สร้างหรืออ่านโหนด (Node) ซึ่งเป็นไฟล์ชนิดพิเศษ

- mode หมายถึง สิทธิการเข้าถึงไฟล์หรือไดเรกทอรี ประกอบด้วย บิตควบคุม Read Write Execute 3 ชุด รวมทั้งหมด 9 บิต ซึ่งได้อธิบายแล้วในหัวข้อที่ 7.1.4

ผู้อ่านสามารถแสดงรายชื่อไฟล์หรือไดเรกทอรีหรือชื่ออุปกรณ์ภายใต้ไดเรกทอรี /dev โดยพิมพ์คำสั่งบนโปรแกรม Terminal

```
$ ls /dev
```

ผู้อ่าน ต้อง เปรียบ เทียบ กับ ชื่อ อุปกรณ์ ที่ ผู้เขียน ตัว หนา ไว้ ว่า ตรง กัน หรือ ไม่ อย่างไร เพื่อให้ ผู้อ่าน มอง เห็น ชัด ว่า **mmcblk0p2** มีอยู่จริง และ ระบบ ได้ ทำ การ เมาท์ เข้า กับ ไดเรกทอรี รุท (Root) นั้น คือ ไดเรกทอรี / ด้วยชนิด ext4 ตามที่ได้แสดงในคำสั่งก่อนหน้านี้แล้ว

ทว

ทว

```
ashmem autofs block btrfs-control bus cachefiles cec0 cec1 char console cpu_dma_latency
cuse disk dma_heap dri fb0 fd full fuse gpiochip0 gpiochip1 gpiochip2 gpiomem hidraw0
hidraw1 hidraw2 hidraw3 hwrng i2c-20 i2c-21 initctl input kmsg kvm log loop0 loop1 loop2
loop3 loop4 loop5 loop6 loop7 loop-control mapper media0 media1 mem mmcblk0
mmcblk0p_1 mmcblk0p_2 mqueue net null port ppp ptmx pts ram0 ram1 ram10 ram11
ram12 ram13 ram14 ram15 ram2 ram3 ram4 ram5 ram6 ram7 ram8 ram9 random raw
rkill rpidvid-h264mem rpidvid-hevcmmem rpidvid-initc rpidvid-vp9mem serial1 shm snd stderr
stdin stdout tty tty0 ... ttyAMA0 ttyprintk uhid uinput urandom vchiq vcio vc-mem vcs ...
watchdog watchdog0 zero
```

นอกจากนี้ อุปกรณ์สำคัญอื่น ๆ เช่น stdin (standard input) stdout (standard output) และ stderr (standard error) นั้นเกี่ยวข้องกับโปรแกรม Terminal ซึ่งเชื่อมโยงกับประโยคในภาษา C ในการทดลองที่ 5 ภาคผนวก E

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

L.4 กิจกรรมท้ายการทดลอง

1. จงใช้โปรแกรม File Manager แล้วคลิกขวาบนชื่อไฟล์เพื่อแสดงคุณสมบัติ (Properties) ต่าง ๆ บนแท็บ General และอธิบายโดยเฉพาะหัวข้อ Total size of files และ Size on disk ว่าเหตุใดถึงแตกต่างกัน
2. สร้างไฟล์ (New) ด้วยโปรแกรม nano คีย์ข้อความด้วยตัวอักษรจำนวน 1 ตัวแล้วบันทึก (Save) ใช้คำสั่ง ls -l เพื่อแสดงรายละเอียดของไดเรกทอรีที่บรรจุไฟล์นั้น เพื่อหาขนาดไฟล์ที่แท้จริง
3. โปรดสังเกตว่าใน test.txt ที่สร้างด้วยโปรแกรม nano เราได้พิมพ์ประโยค fdd คิดเป็นจำนวน 3 ตัวอักษร ๆ ละ 1 ไบต์เท่านั้น จงหาว่าไบต์ที่ 4 คือตัวอักษรใดในรูปที่ 2.12

หน้าถัดไป

4. เพิ่มจำนวนตัวอักษรไปเรื่อย ๆ ใน test.txt จนทำให้ไฟล์มีขนาดมากกว่าเท่ากับ 4096 ไบต์ แล้วใช้คำสั่ง `du -B1 test.txt` ตรวจสอบขนาดไฟล์อีกรอบ บันทึกและอธิบายผลที่ได้โดยเฉพาะจำนวน Blocks ที่ได้จากคำสั่งว่าเท่ากับกี่เซกเตอร์

5. จงเปรียบเทียบผลลัพธ์ของคำสั่ง `stat` ระหว่าง ไดเรกทอรี และ ไฟล์

6. สิทธิ์การเข้าถึง (Permission) ของไดเรกทอรีหรือของไฟล์ประกอบด้วยบิตจำนวน 9 บิต แบ่งเป็น 3 ชุด ๆ ละ 3 บิต จงเรียงลำดับชุดต่าง ๆ ว่าเป็นของสิทธิ์ของใครบ้าง

7. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายชื่อไดเรกทอรีและไฟล์ และ อธิบายผลว่าหมายเลขที่อยู่ด้านซ้ายสุดคืออะไร และเหตุใดจึงมีค่าซ้ำ

```
$ ls -li -l /
```

8. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายละเอียดของชื่อไดเรกทอรีคู่ที่ซ้ำจากข้อที่แล้ว และอธิบายผลว่ามีอะไรที่แตกต่างกัน เพราะเหตุใด

```
$ stat /proc
```

```
$ stat /sys
```

```
$ stat /dev
```

```
$ stat /run
```

9. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายละเอียดของอุปกรณ์ และ อธิบายว่ามีผลลัพธ์ที่แตกต่างกันหรือไม่ เพราะเหตุใด

```
$ stat /dev/mmcblk0p2
```

```
$ stat /
```

10. จงอธิบายว่าเหตุใดไดเรกทอรี `asound` จึงอยู่ใต้ `/proc` ในหัวข้อที่ [I.2.3](#) การทดลองที่ 9 ภาคผนวก [I](#)

11. จงอธิบายความเชื่อมโยงระหว่าง `gpiomem` ที่ได้จากคำสั่ง `ls /dev` กับกิจกรรมท้ายการทดลองที่ 10 ภาคผนวก [J](#)

File: asm

Size: 4 0 9 6 Blocks: 8 IO Block: 4 0 9 6 directory
 Device: b 3 0 2h/4 5 8 26d Inode: 2 6 0 7 1 8 Links: 6
 Access: (0 7 5 5/ drwxr-xr-x) Uid: (1 0 0 0/
pi) Gid: (1 0 0 0/pi)
 Access: ... 2023-01-30 17:27:01.074682918 +0700
 Modify: ... 2023-04-05 19:45:13.709999916 +0700
 Change: ... 2023-04-05 19:45:13.709999916 +0700
 Birth: - 2023-01-30 17:27:01.074682918 +0700

File: stat_asm.txt

Size: 3 7 5 Blocks: 8 IO Block: 4 0 9 6 regular file
 Device: b 3 0 2h/4 5 8 26d Inode: _ 744 _ Links: 1
 Access: (0 6 4 4/-rw-r--r--) Uid: (1 0 0 0/pi) Gid: (1 0 0 0/
pi)
 Access: ... 2023-04-05 22:49:39.7461085671 +0700
 Modify: ... 2023-04-05 23:51:41.3498725912 +0700
 Change: ... 2023-04-05 23:51:41.3498725912 +0700
 Birth: - 2023-04-05 22:49:39.7461085671 +0700

stat directory (asm) แสดงขนาดของ file ใน directory นั้นๆ

stat file (stat_asm.txt) แสดงขนาดในส่วนของ file นั้น/ไดเรกทอรี=

- size ของ directory > file

- จำนวน blocks ที่เก็บ

- ประเภท (directory / file) ที่เก็บ

- device ที่เก็บ

- inode ที่เก็บ

- Link ของ directory ที่เก็บ

- access ที่เก็บ

- stat file หรือ access, modify, change, birth แสดงวันที่

- stat directory หรือ access, modify, change, birth แสดงวันที่