ภาคผนวก K

การทดลองที่ 11 การเชื่อมต่ออินพุต-เอาต์พุตกับสัญญาณอินเทอร์รัปต์

การทดลองนี้คาดว่าผู้อ่านเคยเรียนการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C และแอสเซมบลีจากการทดลองก่อนหน้า ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

- เพื่อพัฒนาการทำงานของอินเทอร์รัปต์ร่วมโปรแกรมภาษา C และแอสเซมบลี ตามเนื้อหาในหัวข้อที่ 6.12
- เพื่อศึกษาการทำงานของอินเทอร์รัปต์ร่วมกับขา GPIO ตามเนื้อหาในหัวข้อที่ 6.11

K.1 การจัดการอินเทอร์รัปต์ของ WiringPi

ไลบรารี wiringPi รองรับการทำอินเทอร์รัปต์ของ GPIO ได้ ทำให้โปรแกรมหลักสามารถทำงาน หลักได้ตามปกติ เมื่อเกิดสัญญาณอินเทอร์รัปต์ขึ้น ไม่ว่าจะเป็นสัญญาณจากการกดปุ่มสวิตช์ ทำให้เกิดขอบขาขึ้นหรือขอบขาลงหรือทั้งสองขอบ โดยการเรียกใช้คำสั่ง

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

โดย pin หมายถึง เลขขาที่ wiringPi กำหนด edgeType กำหนดจากค่าคงที่ 4 ค่านี้

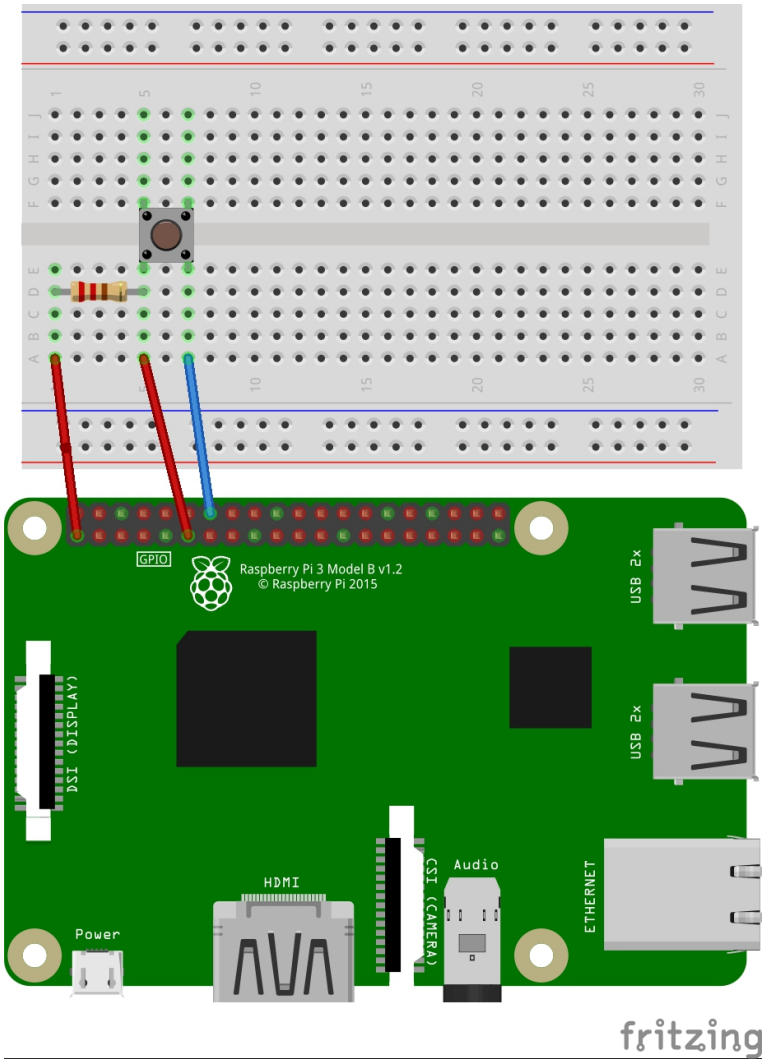
- INT_EDGE_FALLING,
- INT_EDGE_RISING,
- INT_EDGE_BOTH
- INT_EDGE_SETUP.

การกำหนดชนิดขอบขาเป็น 3 ชนิดแรก ไลบรารีจะตั้งค่าเริ่มต้น (Innitialization) ให้โดยอัตโนมัติ หากกำหนดชนิดขอบเป็น INT_EDGE_SETUP ไลบรารีจะไม่ตั้งค่าเริ่มต้น (Innitialization) ให้ เนื่องจากโปรแกรมเมอร์จะต้องกำหนดเอง

พารามิเตอร์ **callback** คือ ชื่อฟังก์ชันที่จะทำหน้าที่เป็น ISR สัญลักษณ์ & หมายถึง แอดเดรสของฟังก์ชัน 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: วงจรสวิตช์ปุ่มกดสำหรับทดลองการเขียนโปรแกรมอินเทอร์รัพต์ในการทดลองที่ 11 ที่มา: fritzing.org

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

K.3 โปรแกรมภาษา C สำหรับทดสอบวงจรอินเทอร์รัปต์

ผู้อ่านต้องทำความเข้าใจกับตัวโปรแกรมก่อนคอมไพล์หรือรันโปรแกรม 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 สำหรับคอมไพล์และลิงก์โปรแกรมจากการทดลองก่อนหน้านี้ 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 เมื่อกดปุ่มแต่ละครั้งจะทำให้ความเร็วในการนับเพิ่มขึ้น หรือ Delay สั้นลงครึ่งหนึ่ง เมื่อกดครั้งที่ 2 จะสั้นลงอีกครึ่งหนึ่ง เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น

7. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 10 แต่นับลงจาก 7-0-7 โดยเพิ่มสวิตช์ปุ่มกดในการทดลองนี้ และเพิ่มฟังก์ชันการอินเทอร์รัปต์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะทำให้ความเร็วในการนับลดลง หรือ Delay เพิ่มขึ้นเท่าตัว เมื่อกดครั้งที่ 2 Delay เพิ่มขึ้นอีกเท่าตัว เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น

5.

-มีหน้าที่รับค่า interrupt ใน pin ที่กำหนดว้ในฟังก์ชั้น ถ้ามีการ interrupt จะทำ return function ตามที่กำหนดไว้ และเพื่อเช็ค interrupt และ error จึงต้องเอา if ไว้ด้านใน

-คือค่าที่อยู่ของ call back function มี & เพื่อชี้ไปที่ address ของ function เวลา callback เพราะ & คือ address ของ function

-เป็นการตรวจจับ flling edge บน GPIO PIN ของ BUTTON_INPUT

ภาคผนวก L

การทดลองที่ 12 การศึกษาอุปกรณ์เก็บรักษาข้อมูลและระบบไฟล์

การทดลองนี้อธิบายและเชื่อมโยงเนื้อหาความรู้ของทุกบทเข้าด้วยกัน แต่จะเน้นบทที่ 6 และบทที่ 7 เพื่อให้ผู้อ่านมองเห็นอุปกรณ์อินพุตและเอาต์พุตเหมือนไฟล์แต่ละไฟล์ โดยมีวัตถุประสงค์ดังนี้

- เพื่อให้เข้าใจการวัดขนาดของไฟล์และไดเรกทอรีในระบบไฟล์
- เพื่อให้รู้จักโครงสร้างและระบบไฟล์ของการ์ดหน่วยความจำไมโคร 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 แช่ตามด้วยปุ่ม o ออกจากโปรแกรมโดยกดปุ่ม Ctrl แช่ตามด้วยปุ่ม 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 filenameที่จัดเก็บเป็นจำนวนเท่าของ1024ไบต์ ในอุปกรณ์เก็บรักษาข้อมูล SD ด้วยคำสั่งต่อไปนี้

```
$ du -B1 test.txt
4096 test.txt
```

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

- คำสั่ง ‘du -h’ จะแสดงผลขนาดหรือจำนวนไบต์โดยใช้หน่วย เช่น K (Kibi: 1024) M (Mebi: 1048576) G (Gibi: 1073741824) นำหน้าชื่อไดเรกทอรีหรือโฟลเดอร์ที่อยู่ใต้ไดเรกทอรีปัจจุบัน และจดบันทึก 5 รายการแรกในตาราง

```
$ du -h
```

Size	Folder Name
84K	./asm/Lab7
12K	./asm/Lab8/Lab8_4/obj/Debug
16K	./asm/Lab8/Lab8_4/obj
16K	./asm/Lab8/Lab8_4/bin/Debug
16K	./asm/Lab8/Lab8_4/bin

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	27%	/
devtmpfs	333M	0	333M	0%.	/dev
tmpfs	462M	24M	438M	6%.	/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 3810468.	10502136	27%	/
devtmpfs	devtmpfs	340548 0	340548	0%.	/dev
tmpfs.	tmpfs	472132 24576	447556	6%	/dev/shm
tmpfs.	tmpfs	188856 1136	187720	1%.	/run
tmpfs	tmpfs	5120 4	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	128350	822626	14%	/
devtmpfs	85137	454	84683	1%.	/dev
tmpfs	118033	83.	117950	1%.	/dev/shm
tmpfs	819200	720.	818480	1%.	/run
tmpfs	118033	3	118030	1%.	/run/lock

โดยคอลัมน์ที่ 2 จากทางซ้ายจะแสดงผลเป็นจำนวน **ไอโหนด** แทน รายละเอียดเรื่องไอโหนด ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้ในบทที่ 7 และทาง wikipedia

- คำสั่ง **stat** แสดงรายละเอียดของไฟล์หรือไดเรกทอรี การทดลองนี้จะใช้ไดเรกทอรี asm ที่มีอยู่และเติมตัวเลขในช่องว่าง

```
$ cd /home/pi
$ stat asm
```

```
  File: asm
  Size: _4096_        Blocks: 8          IO Block: _4096_   Directory
Device: _b302.h/45826_d Inode: _26174._ _      Links:9_
Access: ( _0755_/                drwxr-xr-x)  Uid: ( _1000_/
mayd)   Gid: ( _1000._/mayd)
Access: 2023-01-30 17:11:38.285481752 +0700
Modify: 2023-04-03 16:26:03.039999947 +0700
Change: 2023-04-03 16:26:03.039999947 +0700
 Birth: 2023-01-30 17:11:38.285481752 +0700
```

ผู้อ่านจะต้องกรอกผลลัพธ์ในช่องว่าง ดังต่อไปนี้

- ชื่อ asm
- ขนาด 4096 ไบต์ ใช้พื้นที่จำนวน 8 Blocks ซึ่งหมายถึง 8 **เซ็กเตอร์** ๆ ละ 512 ไบต์ คิดเป็น 4096 ไบต์ เป็น Directory

- มีหมายเลข Device = b302h/45826d หรือเท่ากับ $b302_{16}/45826_{10}$
- มีหมายเลข Inode = 261074_{10} จำนวน 9 Links
- สิทธิ์เข้าถึง (Access Permission) ด้วยรหัส 0755_{16} หรือ $111_2:101_2:101_2$ โดยผู้ใช้หมายเลข Uid (User ID)=1000 ชื่อผู้ใช้ (Username)=mayd ในกรุ๊ปหมายเลข Groupid=1000 ชื่อกรุ๊ป
- เข้าถึง (Access) ...2023-01-30 17:11:38.285481752 +0700
- เปลี่ยนแปลง (Modify) ... 2023-04-03 16:26:03.039999947 +0700
- เวลาที่ Inode เปลี่ยนแปลง (Change) ... 2023-04-03 16:26:03.039999947 +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: b302h/45826d    Inode: 264850      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    mayd)   Gid: ( 1000/
mayd)
Access: 2023-04-03 17:46:04.957502929 +0700
Modify: 2023-04-03 17:46:04.967502921 +0700
Change: 2023-04-03 17:46:04.967502921 +0700
 Birth: 2023-04-03 17:46:04.957502929 +0700
```

ผู้อ่านจะต้องกรอกผลลัพธ์ในช่องว่าง ดังต่อไปนี้เป็นรายบรรทัด

- ชื่อ stat_asm.txt
- ขนาด 375 ไบต์ ใช้พื้นที่จำนวน 8 Blocks ซึ่งหมายถึง 8 **เซ็กเตอร์** ๆ ละ 512 ไบต์ คิดเป็น 4096 ไบต์ เป็น Regular file
- มีหมายเลข Device = b302h/45826d หรือเท่ากับ $b302_{16}/45826_{10}$

- มีหมายเลข Inode = 264850 $_{10}$ จำนวน 1 Links
- สิทธิ์เข้าถึง (Access Permission) ด้วยรหัส 0644$_{16}$ หรือ 110$_2$:100$_2$:100 $_2$ โดยผู้ใช้หมายเลข Uid (User ID)= 1000 ชื่อผู้ใช้ (Username)= mayd ในกรุปหมายเลข Groupid= 1000 ชื่อกรุป mayd
- เข้าถึง (Access) ... 2023-04-03 17:46:04.957502929 +0700
- เปลี่ยนแปลง (Modify) ... 2023-04-03 17:46:04.967502921 +0700
- เวลาที่ Inode เปลี่ยนแปลง (Change) ... 2023-04-03 17:46:04.967502921 +0700
- หมายเลข Inode ของ asm กับ หมายเลข Inode ของ stat_asm.txt ตรงกันหรือไม่ เพราะเหตุใด
  ไม่ตรงกัน เนื่องจาก หมายเลข Inode เป็นหมายเลขประจำตัวของไฟล์หรือ directory นั้นๆ
- asm เป็น **ไดเรกทอรี** ในขณะที่ stat_asm.txt เป็น File
- สิทธิ์เข้าถึง (Access Permission) รหัส 0765$_{16}$ มีความหมายดังต่อไปนี้
  - 111$_2$: เป็นของใคร User
  - 110$_2$: เป็นของใคร Group
  - 101$_2$: เป็นของใคร Other

L.3 อุปกรณ์อินพุต/เอาต์พุตในระบบไฟล์

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

```
$ mount
```

คำสั่งนี้จะแสดงรายชื่อการเมาท์ หรือ ผูกยึด อุปกรณ์อินพุต/เอาต์พุต เข้ากับระบบไฟล์ที่เหมาะกับอุปกรณ์นั้น ๆ ด้วยชื่อไดเรกทอรีหรือชื่อไฟล์ของระบบปฏิบัติการ ผู้อ่านจะต้องกรอกผลลัพธ์ที่สำคัญในช่องว่าง และศึกษาคำอธิบายต่อไปนี้

- **/dev/mmcblk0p__** on / type ext4 (rw,noatime) เป็นระบบไฟล์ **ext4** ซึ่งเป็นระบบไฟล์หลักของลินุกซ์ ย่อมาจากคำว่า Fourth Extended File System เป็นเวอร์ชันที่ 4 พัฒนาจากชนิด ext3 ซึ่งพัฒนาจากระบบยูนิกซ์ตามรายละเอียดในหัวข้อที่ 7.1 และ wikipedia

- devtmpfs on /dev type devtmpfs (rw, relatime, size=3834564k, nr_inodes=958641, mode=755)
- proc on /proc type proc (rw, relatime) เป็นระบบไฟล์เสมือน (Virtual File System) สำหรับระบบสำคัญต่าง ๆ เช่น CPU, โดยจะสร้างขึ้นเมื่อบูตเครื่อง และลบทิ้งเมื่อชัตดาวน์ระบบ รายละเอียดเพิ่มเติมที่ wikipedia
- sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime) เป็นระบบไฟล์เสมือน (Virtual File System) รายละเอียดเพิ่มเติมที่ wikipedia
- securityfs on /sys/kernel/security type securityfs (rw, nosuid, nodev, noexec, relatime)
- tmpfs on /dev/shm type tmpfs (rw, nosuid, nodev) ย่อมาจากคำว่า Temporary File System รายละเอียดเพิ่มเติมที่ wikipedia
- devpts on /dev/pts type devpts (rw, nosuid, noexec, relatime, gid=5, mode=620, ptmxmode=000) เป็นระบบไฟล์เสมือน (Virtual File System) สำหรับอุปกรณ์อินพุตเอาต์พุตต่าง ๆ รายละเอียดเพิ่มเติมที่ wikipedia
- ...
- **/dev/mmcblk0p__** on /boot type vfat ระบบไฟล์ **vfat** เป็นส่วนต่อขยายของระบบไฟล์ FAT ซึ่งย่อมาจากคำว่า File Allocation Table เพื่อรองรับชื่อไฟล์ที่ยาวกว่า FAT ที่มา: wikipedia
- ...

รายชื่อต่อไปนี้ คือ ตัวเลือกคุณสมบัติ (Atttribute) ที่สำคัญของระบบไฟล์ เช่น

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

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

ผู้อ่านสามารถแสดงรายชื่อไฟล์หรือไดเรกทอรีหรือชื่ออุปกรณ์ภายใต้ไดเรกทอรี /dev โดยพิมพ์คำสั่งบนโปรแกรม Terminal

```
$ ls /dev
```

ผู้อ่านต้องเปรียบเทียบกับชื่ออุปกรณ์ที่ผู้เขียนตัวหนาไว้ว่าตรงกันหรือไม่ อย่างไร เพื่อให้ผู้อ่านมองเห็นชัดว่า **mmcblk0p2** มีอยู่จริงและระบบได้ทำการเมาท์เข้ากับไดเรกทอรีรูท (Root) นั่นคือไดเรกทอรี / ด้วยชนิด ext4 ตามที่ได้แสดงในคำสั่งก่อนหน้าแล้ว

ashem 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__ mmcblk0p__ 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 rfkill rpivid-h264mem rpivid-hevcmem rpivid-initc rpivid-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

4. เพิ่มจำนวนตัวอักษรไปเรื่อย ๆ ใน test.txt จนทำให้ไฟล์มีขนาดมากกว่าเท่ากับ 4096 ไบต์ แล้วใช้คำสั่ง du -B1 test.txt ตรวจสอบขนาดไฟล์อีกรอบ บันทึกและอธิบายผลที่ได้โดยเฉพาะจำนวน Blocks ที่ได้จากคำสั่งว่าเท่ากับกี่**เซ็กเตอร์**

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 และอธิบายผลว่าหมายเลขที่อยู่ด้านซ้ายสุดคืออะไร และเหตุใดจึงมีค่าซ้ำ

```
$ ls -i -l /
```

8. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายละเอียดของชื่อไดเรกทอรีคู่ที่ซ้ำจากข้อที่แล้ว และอธิบายผลว่ามีอะไรที่แตกต่างกัน เพราะเหตุใด

```
$ stat /proc
$ stat /sys

$ stat /dev
$ stat /run
```

9. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายละเอียดของอุปกรณ์ และอธิบายว่ามีผลลัพธ์ที่แตกต่างกันหรือไม่ 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

Stat directory (asm)

```
$ stat stat_asm.txt

  File: stat_asm.txt
  Size: 375       Blocks: 8       IO Block: 4096   Regular file
Device: b302 h/ 45826 d    Inode: 264850      Links: 1
Access: (0644 -/-rw-r--r--)  Uid: ( 1000 /    mayd)   Gid: ( 1000 /
mayd )
Access: 2023-04-03 17:46:04.957502929 +0700
Modify: 2023-04-03 17:46:04.967502921 +0700
Change: 2023-04-03 17:46:04.967502921 +0700
 Birth: 2023-04-03 17:46:04.957502929 +0700
```

Stat file (stat_asm.txt)

```
$ stat asm

  File: asm
  Size: 4096       Blocks: 8          IO Block: 4096   Directory
Device: b302 h/45826 d Inode: 26174      Links:9
Access: ( 0755 /                  drwxr-xr-x)  Uid: ( 1000 /
mayd)   Gid: ( 1000 /mayd)
Access: 2023-01-30 17:11:38.285481752 +0700
Modify: 2023-04-03 16:26:03.039999947 +0700
Change: 2023-04-03 16:26:03.039999947 +0700
 Birth: 2023-01-30 17:11:38.285481752 +0700
```

stat directory (asm) แสดงผลของ file ภาพรวมใน directory นั้นๆ stat file (stat_asm.txt) แสดงผลในส่วนของ file นั้นโดยเฉพาะ

- size ของ directory มีขนาดใหญ่กว่า file
- เลข Blocks ตรงกัน
- ประเภท (directory , file) ต่างกัน
- device ตรงกัน
- Inode ต่างกัน
- Link ของ directory มากกว่า
- การ access ต่างกัน
- stat file การ Access , Modify , Change , Birth มีเวลาตรงกัน
- stat directory การ Access , Modify , Change , Birth มีเวลาต่างกัน