

ภาคผนวก L

การทดลองที่ 12 การศึกษาอุปกรณ์เก็บรักษาข้อมูลและระบบไฟล์

การทดลองนี้อธิบายและเชื่อมโยงเนื้อหาความรู้ของทุกบทเข้าด้วยกัน แต่จะเน้นบทที่ 7 และบทที่ 6 เพื่อให้ผู้อ่านมองเห็นอุปกรณ์อินพุตและเอาต์พุตเหมือนไฟล์แต่ละไฟล์ โดยมีวัตถุประสงค์ดังนี้

- เพื่อให้เข้าใจการวัดขนาดของไฟล์และไดเรคทอรีในระบบไฟล์
- เพื่อให้รู้จักโครงสร้างและระบบไฟล์ของหน่วยความจำการ์ด microSD ที่ใช้งานในปัจจุบัน
- เพื่อให้เข้าใจระบบไฟล์ (File System) ชนิดต่างๆ บนบอร์ด Pi3
- เพื่อให้สามารถเชื่อมโยงอุปกรณ์อินพุตเอาต์พุตชนิดต่างๆ กับระบบไฟล์

L.1 ขนาดของไฟล์และไดเรคทอรี

ผู้อ่านสามารถตรวจสอบขนาดของไฟล์ใดๆ ชื่อ filename ที่แท้จริง หน่วยเป็นไบต์ ด้วยคำสั่ง **du** (Disk Usage) โดยทำตามขั้นตอนต่อไปนี้

- ย้ายไดเรคทอรีปัจจุบันไปที่ /home/pi ซึ่งเป็นไดเรคทอรีหลักของผู้ใช้ชื่อ pi

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

- สร้างไฟล์ข้อความ test.txt ด้วยโปรแกรม nano ด้วยคำสั่งต่อไปนี้

```
$ nano test.txt
```

พิมพ์ข้อความ fdd ลงในไฟล์ ทำการ Write โดยกดปุ่ม Ctrl แชะตามด้วยปุ่ม o ออกจากโปรแกรมโดยกดปุ่ม Ctrl แชะตามด้วยปุ่ม x

- คำสั่ง 'df -i' จะแสดงรายการต่างๆ ดังนี้ จดบันทึก 10 รายการแรกลงในตาราง

```
$ df -i
```

Filesystem	Inodes	IUsed	IFree	IUse%	Mounted on

โดยคอลัมน์จะแสดงผลเป็นจำนวน **ไอน์ด (inode)** แทน รายละเอียดเรื่อง inode ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้ในบทที่ 7 และทาง [wikipedia](https://en.wikipedia.org/wiki/Inode)

- คำสั่ง **stat** แสดงรายละเอียดของไฟล์หรือไดเรคทอรี

```
$ stat asm
```

```
File: asm
Size: 4096      Blocks: 8      IO Block: 4096   directory
Device: b307h/45831d Inode: 521754      Links: 3
Access: (0755/drwxr-xr-x)  Uid: ( 1000/      pi)   Gid: ( 1000/      pi)
Access: 2019-03-19 19:43:05.449401732 +0700
Modify: 2019-03-19 19:43:05.449401732 +0700
Change: 2019-03-19 19:43:05.449401732 +0700
Birth: -
```

ผู้เขียนอธิบายผลลัพธ์ที่ได้ตามลำดับดังนี้

- ชื่อ asm
- ขนาด 4096 ไบต์ ใช้พื้นที่จำนวน 8 Blocks เป็นไดเรคทอรี (directory)
- มีหมายเลข Device = b307h/45831d หรือเท่ากับ b307₁₆/45831₁₀
- มีหมายเลข Inode ที่ 521754 จำนวน 3 Links
- สิทธิ์เข้าถึง (Access) ด้วยรหัส 0644 หรือ 011₂:100₂:100₂ โดยผู้ใช้หมายเลข Uid (User ID)=1000 ชื่อผู้ใช้ (Username)=pi ในกลุ่มหมายเลข Groupid=1000 ชื่อกลุ่ม pi
- เข้าถึง (Access) ณ วันที่ 19 มีนาคม 2019 เวลา 19.43.05

- เปลี่ยนแปลง (Modify) ณ วันที่ 19 มีนาคม 2019 เวลา 19.43.05
- เวลาที่ Inode เปลี่ยนแปลง (Change) ณ วันที่ 19 มีนาคม 2019 เวลา 19.43.05

เบื้องต้นผู้เขียนขอให้ผู้อ่านสร้างไฟล์ผลลัพธ์จากคำสั่ง stat ไปเก็บในไฟล์ เพื่อมาประกอบบททดลองต่อไป โดย

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

หลังจากนั้น เราสามารถตรวจสอบสถานะของไฟล์ stat_asm.txt ได้ดังนี้

```
$ stat stat_asm.txt
```

```
File: stat\_asm.txt
Size: 341          Blocks: 8          IO Block: 4096   regular file
Device: b307h/45831d Inode: 524766      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/      pi)   Gid: ( 1000/      pi)
Access: 2019-03-19 19:45:05.459401732 +0700
Modify: 2019-03-19 19:45:05.459401732 +0700
Change: 2019-03-19 19:45:05.459401732 +0700
Birth: -
```

ผู้เขียนอธิบายผลลัพธ์ที่ได้ตามลำดับ ดังนี้

- ชื่อ stat_asm.txt
- ขนาด 341 ไบต์ ใช้พื้นที่จำนวน 8 Blocks เป็นไฟล์ธรรมดา (regular File)
- มีหมายเลข Device = b307h/45831d หรือเท่ากับ b307₁₆/45831₁₀
- มีหมายเลข Inode ที่ 524766 จำนวน 1 Links
- สิทธิ์เข้าถึง (Access Permission) ด้วยรหัส 0644 หรือ 011₂:100₂:100₂ โดยผู้ใช้หมายเลข Uid (User ID)=1000 ชื่อผู้ใช้ (Username)=pi ในกรุปหมายเลข Groupid=1000 ชื่อกรุป pi
- เข้าถึง (Access) ณ วันที่ 19 มีนาคม 2019 เวลา 19.45.05
- เปลี่ยนแปลง (Modify) ณ วันที่ 19 มีนาคม 2019 เวลา 19.45.05
- เวลาที่ Inode เปลี่ยนแปลง (Change) ณ วันที่ 19 มีนาคม 2019 เวลา 19.45.05

L.3 อุปกรณ์อินพุตและเอาต์พุตในระบบไฟล์

การทดลองในหัวข้อนี้จะเชื่อมต่อกับเนื้อหาในบทที่ 3 และ การทดลองที่ 4 ภาคผนวก D หลักการของระบบปฏิบัติการ Unix คือ การ**เมาท์** (Mount) อุปกรณ์กับไดเรกทอรีด้วยระบบไฟล์ (File System) ที่แตกต่างกัน โดยใช้ชื่อไดเรกทอรีที่ต่างกัน โดยมีรูทไดเรกทอรีหรือโพลเดอร์ (Root Directory) เป็นตำแหน่งเริ่มต้น ผู้อ่านสามารถพิมพ์คำสั่งใน Terminal

```
$ mount
```

คำสั่งนี้จะแสดงรายชื่อการเมาท์ หรือ ผูกยึด อุปกรณ์อินพุตเอาต์พุต เข้ากับไดเรกทอรีของระบบปฏิบัติการ ตัวอย่างผลลัพธ์และคำอธิบายต่อไปนี้

- /dev/mmcbk0p7 on / type ext4 (rw,noatime,data=ordered)
- devtmpfs on /dev type devtmpfs (rw,relatime,size=470116k,nr_inodes=117529,mode=755)
- sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
- proc on /proc type proc (rw,relatime)
- tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
- devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
- ...

โดยมีชนิด (type) หรือระบบไฟล์ที่แตกต่างกัน เช่น

- ชนิด **ext4** ซึ่งเป็นระบบไฟล์หลักของลินุกซ์ ย่อมาจากคำว่า Fourth Extended File System เป็นเวอร์ชันที่ 4 พัฒนาจากชนิด ext3 [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- ชนิด sysfs เป็นระบบไฟล์เสมือน (Virtual File System) [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- ชนิด devfs เป็นระบบไฟล์เสมือน (Virtual File System) สำหรับอุปกรณ์อินพุตเอาต์พุตต่างๆ [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- ชนิด tmpfs ย่อมาจากคำว่า Temporary File System [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- ชนิด proc เป็นระบบไฟล์เสมือน (Virtual File System) สำหรับระบบสำคัญต่างๆ เช่น CPU, โดยจะสร้างขึ้นเมื่อบูทเครื่อง และลบทิ้งเมื่อชัตดาวน์ระบบ [รายละเอียดเพิ่มเติมที่ wikipedia](#)

รายชื่อต่อไปนี้ คือ ตัวเลือกคุณสมบัติ (Attribute) ที่สำคัญของระบบไฟล์ เช่น

- rw : read/write สามารถอ่านและเขียนได้
- noatime และ atime: No/ Access Time หมายถึง ไม่มี/มีการบันทึกเวลาในการสร้าง อ่านหรือเขียนไฟล์ทุกครั้ง
- relatime หมายถึง มีการบันทึกเวลาในการสร้าง อ่านหรือเขียนไฟล์ เมื่อเกิดการแก้ไขไฟล์ หรือ การอ่านหรือเข้าถึงไฟล์มากกว่าเวลาที่บันทึกไว้ก่อนหน้านี้อย่างน้อย 24 ชั่วโมง

- **nosuid**: No SuperUser ID เป็นการป้องกันไม่ให้ผู้ดูแลระบบ (SuperUser) กระทำการใดๆ ได้ เพื่อความมั่นคงปลอดภัย
- **noexec**: No Execution เพื่อตั้งค่าไม่ให้รันไฟล์ที่อยู่ในไดเรกทอรีนี้ได้ เช่น ไฟล์ที่เป็นไวรัสหรือมัลแวร์ (Malware) ที่แอบแฝงเข้ามา
- **nodev**: No Device หมายถึง การไม่อนุญาตให้สร้างหรืออ่านโนหนด (Node) ซึ่งเป็นไฟล์ชนิดพิเศษ
- **mode** หมายถึง Group 3 บิต คือ บิตควบคุม Read Write Execute รวมทั้งหมด 9 บิต

ผู้อ่านสามารถแสดงรายชื่อไดเรกทอรีหรือไดเรกทอรีหรือชื่ออุปกรณ์ภายใต้ไดเรกทอรี `/dev` โดยพิมพ์คำสั่งบนโปรแกรม Terminal

```
$ ls /dev
```

ผู้อ่านจะเห็นผลลัพธ์ที่ได้ทั้งหมดซึ่งมีจำนวนมากพอสมควร แต่ผู้เขียนได้พิมพ์ชื่ออุปกรณ์ที่สำคัญๆ ด้วยตัวหนา เพื่อให้ผู้อ่านมองเห็นชัดว่า **mmcblk0p7** มีอยู่จริงและระบบได้ทำการเมาท์เข้ากับไดเรกทอรีรูท (Root) นั่นคือ ไดเรกทอรี / ด้วยชนิด `ext4` ตามที่ได้แสดงในคำสั่งก่อนหน้านี้แล้ว

```
autofs block btrfs-control bus cachefiles char console cpu_dma_latency cuse disk fb0
fd full fuse gpiochip0 gpiochip1 gpiochip2 gpiomem hidraw0 hidraw1 hwrng initctl input kmsg
log loop0 loop1 loop2 loop3 loop4 loop5 loop6 loop7 loop-control mapper mem mem-
ory_bandwidth mmcblk0 mmcblk0p1 mmcblk0p2 mmcblk0p5 mmcblk0p6 mmcblk0p7
mqueue net network_latency network_throughput null ppp ptmx pts ram0 ram1 ram10
ram11 ram12 ram13 ram14 ram15 ram2 ram3 ram4 ram5 ram6 ram7 ram8 ram9 random raw
rfkill serial1 shm snd stderr stdin stdout tty tty0 tty1 tty10 tty11 tty12 tty13 tty14 tty15
tty16 tty17 tty18 tty19 tty2 tty20 tty21 tty22 tty23 tty24 tty25 tty26 tty27 tty28 tty29 tty3
tty30 tty31 tty32 tty33 tty34 tty35 tty36 tty37 tty38 tty39 tty4 tty40 tty41 tty42 tty43 tty44
tty45 tty46 tty47 tty48 tty49 tty5 tty50 tty51 tty52 tty53 tty54 tty55 tty56 tty57 tty58 tty59
tty6 tty60 tty61 tty62 tty63 tty7 tty8 tty9 ttyAMA0 ttyprintk uhid uinput urandom vchiq vcio
vc-mem vcs vcs1 vcs2 vcs3 vcs4 vcs5 vcs6 vcs7 vcsa vcsa1 vcsa2 vcsa3 vcsa4 vcsa5 vcsa6
vcsa7 vcsn vhci watchdog watchdog0 zero
```

นอกจากนี้ อุปกรณ์สำคัญอื่นๆ 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) ต่างๆ 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 ตัวอักษรๆ ละ 1 ไบท์เท่านั้น จงหาว่าไบท์ที่ 4 คือตัวอักษรใดในรูปที่ 2.12
4. เพิ่มจำนวนตัวอักษรไปเรื่อยๆ ใน test.txt จนทำให้ไฟล์มีขนาดมากกว่าเท่ากับ 4096 ไบท์ แล้วใช้คำสั่ง `du -B1 test.txt` ตรวจสอบขนาดไฟล์อีกรอบ บันทึกและอธิบายผลที่ได้
5. จงเปรียบเทียบผลลัพธ์ของคำสั่ง `stat` ระหว่าง ไดเรกทอรี และ ไฟล์
6. สิทธิ์การเข้าถึง (Permission) ของไดเรกทอรีหรือของไฟล์ประกอบด้วยบิตจำนวน 9 บิต แบ่งเป็น 3 ชุดๆ ละ 3 บิต จงเรียงลำดับชุดต่างๆ ว่าเป็นของสิทธิ์ของใครบ้าง
7. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายชื่อไดเรกทอรีและไฟล์ และอธิบายผลว่าหมายเลขที่อยู่ด้านซ้ายสุดคืออะไร และเหตุใดจึงมีค่าซ้ำ

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

8. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายละเอียดของชื่อไดเรกทอรีคู่ที่ซ้ำจากข้อที่แล้ว และอธิบายผลว่ามีอะไรที่แตกต่างกัน เพราะเหตุใด

```
$ stat /proc
```

```
$ stat /sys
```

```
$ stat /dev
```

```
$ stat /run
```

9. จงใช้คำสั่งต่อไปนี้ เพื่อแสดงรายละเอียดของอุปกรณ์ และอธิบายว่ามีผลลัพธ์ที่แตกต่างกันหรือไม่ เพราะเหตุใด

```
$ stat /dev/mmcblk0p7
```

```
$ stat /
```

10. จงอธิบายว่าเหตุใดไดเรกทอรี asound จึงอยู่ใต้ /proc ในหัวข้อที่ 1.2.1 การทดลองที่ 9 ภาคผนวก I
11. จงอธิบายความเชื่อมโยงระหว่าง `gpiomem` ที่ได้จากคำสั่ง `ls /dev` กับกิจกรรมท้ายการทดลองที่ 10 ภาคผนวก J