

ภาคผนวก L

การทดลองที่ 12 การศึกษาอุปกรณ์เก็บรักษาข้อมูล และระบบไฟล์

การทดลองนี้อธิบายและเชื่อมโยงเนื้อหาความรู้ของทุกบทเข้าด้วยกัน แต่จะเน้นบทที่ 6 และบทที่ 7 เพื่อให้ผู้อ่านมองเห็นอุปกรณ์อินพุตและเอาต์พุตเหมือนไฟล์แต่ละไฟล์ โดยมีวัตถุประสงค์ดังนี้

- เพื่อให้เข้าใจการวัดขนาดของไฟล์และไดเรกทอรีในระบบไฟล์
- เพื่อให้รู้จักโครงสร้างและระบบไฟล์ของการวัดหน่วยความจำไม่โคร SD ที่ใช้งานในปัจจุบัน
- เพื่อให้เข้าใจระบบไฟล์ (File System) ชนิดต่างๆ บนบอร์ด Pi
- เพื่อให้สามารถเชื่อมโยงอุปกรณ์อินพุต/เอาต์พุตชนิดต่างๆ กับระบบไฟล์

L.1 ขนาดของไฟล์และไดเรกทอรี

ผู้อ่านสามารถตรวจสอบขนาดของไฟล์ใดๆ 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 ออกจากโปรแกรมโดยกดปุ่ม 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’ ผู้อ่านสามารถ**ตรวจสอบ**ขนาดของไฟล์ใดๆ ชื่อ filenameที่จัดเก็บเป็นจำนวนเท่าของ 1024 ไบต์ ในอุปกรณ์เก็บรักษาข้อมูล SD ด้วยคำสั่งต่อไปนี้

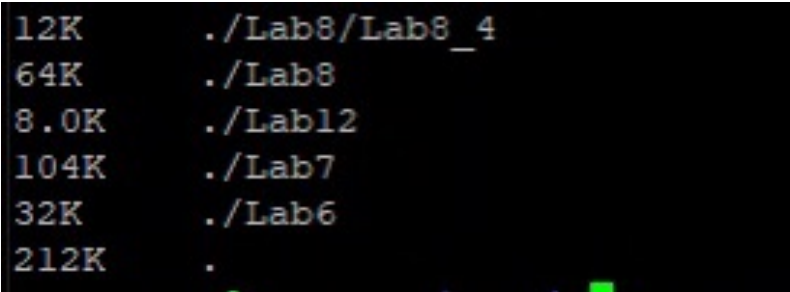
```
$ du -B1 test.txt
4096 test.txt
```

ตัวเลข 4096 หมายถึง เลขจำนวนไบต์ที่คำสั่ง du แสดงผลมาตามพารามิเตอร์ B1 ที่ส่งไป โดยผู้อ่านจะสังเกตเห็นความแตกต่าง ถึงแม้ไฟล์มีข้อมูลจำนวนน้อยเพียงไม่กี่ไบต์ แต่การจองพื้นที่ในอุปกรณ์สำรองจะมีขนาดเป็นจำนวนเท่าของ 4096 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
12k	./Lab8 /Lab8_4
64k	./Lab8
8.0k	./Lab12
104k	./Lab7
32k	./Lab6



L.2 ระบบไฟล์

ผู้ใช้หรือผู้ดูแลระบบลินุกซ์ สามารถ**ตรวจสอบ**การใช้งานอุปกรณ์เก็บรักษาข้อมูล เช่น ฮาร์ดดิสก์ไดรฟ์ โซลิดสเตตไดรฟ์ 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	2๙ ๕	6.๙ ๕	21 ๕	26%	/
devtmpfs	3.7 ๕	0	3.7 G	0%	/dev
tmpfs	3.๙ ๕	0	3.๙ ๕	0%	/dev/shm
tmpfs	3.๙ ๕	17๓	3.๙ ๕	1%	/run
tmpfs	5.0๓	4.0k	5.0๓	1%	/run/lock

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/root	29G	6.9G	21G	26%	/
devtmpfs	3.7G	0	3.7G	0%	/dev
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	3.9G	17M	3.9G	1%	/run
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/sys/fs/cgroup
/dev/mmcblk0p1	253M	49M	204M	20%	/boot
tmpfs	790M	4.0K	790M	1%	/run/user/1000
tmpfs	790M	0	790M	0%	/run/user/1004

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	2๙๓33356	21248576	26%	/
devtmpfs	devtmpfs	387๙284	387๙284	0%	/dev
tmpfs	tmpfs	4044148	40๔๔1๔๘	0%	/dev/shm
tmpfs	tmpfs	4044148	4027192	1%	/run
tmpfs	tmpfs	5120	5116	1%	/run/lock

Filesystem	Type	1K-blocks	Used	Available	Use%	Mounted on
/dev/root	ext4	29733356	7223452	21248576	26%	/
devtmpfs	devtmpfs	3879284	0	3879284	0%	/dev
tmpfs	tmpfs	4044148	0	4044148	0%	/dev/shm
tmpfs	tmpfs	4044148	16956	4027192	1%	/run
tmpfs	tmpfs	5120	4	5116	1%	/run/lock
tmpfs	tmpfs	4044148	0	4044148	0%	/sys/fs/cgroup
/dev/mmcblk0p1	vfat	258095	49281	208814	20%	/boot
tmpfs	tmpfs	808828	4	808824	1%	/run/user/1000
tmpfs	tmpfs	808828	0	808828	0%	/run/user/1004

- คำสั่ง ‘df -i’ จะแสดงรายการต่างๆ ดังนี้ จดบันทึก 5 รายการที่ตรงกับตารางที่แล้ว

```
$ df -i
```

Filesystem	Inodes	IUsed	IFree	IUse%	Mounted on
/dev/root	1870176	164227	1705949	9%	/
devtmpfs	74939	439	74500	1%	/dev
tmpfs	157371	1	157370	1%	/dev/shm
tmpfs	157371	598	156773	1%	/run
tmpfs	157371	3	157368	1%	/run/lock

Filesystem	Inodes	IUsed	IFree	IUse%	Mounted on
/dev/root	1870176	164227	1705949	9%	/
devtmpfs	74939	439	74500	1%	/dev
tmpfs	157371	1	157370	1%	/dev/shm
tmpfs	157371	598	156773	1%	/run
tmpfs	157371	3	157368	1%	/run/lock
tmpfs	157371	15	157356	1%	/sys/fs/cgroup
/dev/nmcb1k0p1	0	0	0	-	/boot
tmpfs	157371	20	157351	1%	/run/user/1000
tmpfs	157371	13	157358	1%	/run/user/1004

โดยคอลัมน์ที่ 2 จากทางซ้ายจะแสดงผลเป็นจำนวน **ไอโนด** แทน รายละเอียดเรื่องไอโนด ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้ในบทที่ 7 และทาง [wikipedia](#)

- คำสั่ง **stat** แสดงรายละเอียดของไฟล์หรือไดเรกทอรี การทดลองนี้จะใช้ไดเรกทอรี `asm` ที่มีอยู่ และเติมตัวเลขในช่องว่าง

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

```
File: asm
Size: 4096      Blocks: 8      IO Block: 4096      directory
Device: b302h/45826d Inode: 518669      Links: 6
Access: (0755/drwxr-xr-x)  Uid: ( 1004/t63010487)  Gid: ( 1004/t63010487)
Access: 2022-02-14 13:19:34.298414955 +0700
Modify: 2022-02-14 13:35:08.431765872 +0700
Change: 2022-02-14 13:35:08.431765872 +0700
Birth: -
```

File: asm	Size: 4096	Blocks: 8	IO Block: 4096	directory
Device: b302h/45826d	Inode: 518669	Links: 6		
Access: (0755/drwxr-xr-x)	Uid: (1004/t63010487)	Gid: (1004/t63010487)		
Access: 2022-02-14 13:19:34.298414955	+0700			
Modify: 2022-04-01 13:35:08.431765872	+0700			
Change: 2022-04-01 13:35:08.431765872	+0700			
Birth: -				

ผู้อ่านจะต้องกรอกผลลัพธ์ในช่องว่าง ดังต่อไปนี้

- ชื่อ `asm`
- ขนาด 4096 ไบต์ ใช้พื้นที่จำนวน 8 Blocks ซึ่งหมายถึง 8 เซ็กเตอร์ๆ ละ 512 ไบต์ เป็น directory
- มีหมายเลข Device = b302 h/45826 d หรือเท่ากับ b302 ₁₆/45826 ₁₀
- มีหมายเลข Inode = 518669 ₁₀ จำนวน 3 Links
- สิทธิ์เข้าถึง (Access) ด้วยรหัส 0755 ₁₆ หรือ 0111 ₂:0101 ₂:0101 ₂ โดยผู้ใช้หมายเลข Uid (User ID)=1004 ชื่อผู้ใช้ (Username)=pi ในกรุปหมายเลข Groupid=1004 ชื่อกรุป t63010487

- เข้าถึง (Access) ... ๑๐๑๑-๐๑-๑๔ ๑๓:๑๖:๓๔ . ๑๑๘๔๑๔๙๕๕ +๐๗๐๐
- เปลี่ยนแปลง (Modify) ... ๑๐๑๑-๐๑-๑๔ ๑๓:๓๕:๐๘.๔๓๑๖๕๘๓๒ +๐๗๐๐
- เวลาที่ Inode เปลี่ยนแปลง (Change) ... ๑๐๑๑-๐๑-๑๔ ๑๓:๓๕:๐๘.๔๓๑๖๕๘๓๒ +๐๗๐๐

เบื้องต้นผู้เขียนขอให้ผู้อ่านสร้างไฟล์ผลลัพธ์จากคำสั่ง stat ไปเก็บในไฟล์ เพื่อมาใช้ในการประกอบการทดลองต่อไป โดย

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

หลังจากนั้น เราสามารถตรวจสอบสถานะของไฟล์ stat_asm.txt ได้ดังนี้

```
$ stat stat_asm.txt
```

```
File: stat_asm.txt
Size: ๓๔๓      Blocks: ๘      IO Block: 4096   regular file
Device: b๑๐๒_h/458๒๖_d Inode: ๓๙๙๐๓๐      Links: 1
Access: (๐๖๔๔/-rw-r--r--)  Uid: ( 1๐๐๔ /๓๖๓๐1๐๔๘7)  Gid: ( 1๐๐๔ /๓๖๓๐1๐๔๘7)
Access: ... ๑๐๑๑-๐๔-๐1 ๑๓:๔๓:๒๙.8๒๐2๐๐4๙1 +๐๗๐๐
Modify: ... ๑๐๑๑-๐๔-๐1 ๑๓:๔๓:๒๙.8๒๐2๐๐4๙1 +๐๗๐๐
Change: ... ๑๐๑๑-๐๔-๐1 ๑๓:๔๓:๒๙.8๒๐2๐๐4๙1 +๐๗๐๐
Birth: -
```

```
t63010487@Pi432b:~ $ stat stat_asm.txt
File: stat_asm.txt
Size: 343      Blocks: 8      IO Block: 4096   regular file
Device: b302h/45826d Inode: 399030      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1004/t63010487)  Gid: ( 1004/t63010487)
Access: 2022-04-01 13:43:29.820200491 +0700
Modify: 2022-04-01 13:43:29.820200491 +0700
Change: 2022-04-01 13:43:29.820200491 +0700
Birth: -
```

ผู้อ่านจะต้องกรอกผลลัพธ์ในช่องว่าง ดังต่อไปนี้

- ชื่อ stat_asm.txt
- ขนาด ๓๔๓ ไบต์ ใช้พื้นที่จำนวน ๘ Blocks ซึ่งหมายถึง 8 เซ็กเตอร์ๆ ละ 512 ไบต์ เป็น regular file
- มีหมายเลข Device = b๑๐๒_h/458๒๖_d หรือเท่ากับ b๑๐๒_16/458๒๖_10
- มีหมายเลข Inode = ๓๙๙๐๓๐ จำนวน 1 Links
- สิทธิ์เข้าถึง (Access Permission) ด้วยรหัส ๐๖๔๔₁₆ หรือ ๐1๐๐₂:๐1๐๐₂:๐1๐๐₂ โดยผู้ใช้หมายเลข Uid (User ID)= 1๐๐๔ ชื่อผู้ใช้ (Username)=__ ในกรุปหมายเลข Groupid= 1๐๐๔ ชื่อกรุป ๓๖๓๐1๐๔๘๗
- เข้าถึง (Access) ... ๑๐๑๑-๐๔-๐1 ๑๓:๔๓:๒๙.8๒๐2๐๐4๙1 +๐๗๐๐
- เปลี่ยนแปลง (Modify) ... ๑๐๑๑-๐๔-๐1 ๑๓:๔๓:๒๙.8๒๐2๐๐4๙1 +๐๗๐๐
- เวลาที่ Inode เปลี่ยนแปลง (Change) ... ๑๐๑๑-๐๔-๐1 ๑๓:๔๓:๒๙.8๒๐2๐๐4๙1 +๐๗๐๐
- หมายเลข Inode ของ asm กับ หมายเลข Inode ของ stat_asm.txt ตรงกันหรือไม่ เพราะเหตุใด ไม่ตรงกัน พอๆ กับตัวเลขประจำตัว ของ file หรือ directory ใดๆ
- asm เป็น ไทเรททอรี ในขณะที่ stat_asm.txt เป็น regular file

- สิทธิ์เข้าถึง (Access Permission) รหัส 0765_{16} มีความหมายดังต่อไปนี้
 - 111_2 : เป็นของใคร [เจ้าของ file](#)
 - 110_2 : เป็นของใคร [ผู้ใช้กลุ่มเดียวกับเจ้าของ file](#)
 - 101_2 : เป็นของใคร [ผู้ใช้ อื่น ๆ](#)

L.3 อุปกรณ์อินพุต/เอาต์พุตในระบบไฟล์

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

\$ mount

คำสั่งนี้จะแสดงรายชื่อการเมาท์ หรือ ผูกยึด อุปกรณ์อินพุต/เอาต์พุต เข้ากับระบบไฟล์ที่เหมาะสมกับอุปกรณ์นั้นๆ 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](#)
- `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) สำหรับระบบสำคัญต่างๆ เช่น CPU, โดยจะสร้างขึ้นเมื่อบูตเครื่อง และลบทิ้งเมื่อชัตดาวน์ระบบ [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- `sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)` เป็นระบบไฟล์เสมือน (Virtual File System) [รายละเอียดเพิ่มเติมที่ wikipedia](#)
- `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) สำหรับอุปกรณ์อินพุตเอาต์พุตต่างๆ [รายละเอียดเพิ่มเติมที่ wikipedia](#)

• ...

```
t63010487@Pi432b:~$ mount
/dev/mmcblk0p2 on / type ext4 (rw,noatime)
devtmpfs on /dev type devtmpfs (rw,relatime,size=3879284k,nr_inodes=74939,mode=755)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,relatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,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)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
none on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpuacct)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=31,pgpr=1,timeout=0,minproto=5,maxproto=5,direct)
sunrpc on /run/rpc_pipefs type rpc_pipefs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/mmcblk0p1 on /boot type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed,errors=remount-ro)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=808828k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
tmpfs on /run/user/1004 type tmpfs (rw,nosuid,nodev,relatime,size=808828k,mode=700,uid=1004,gid=1004)
```

- `/dev/mmcblk0p1` on `/boot` type `vfat` ระบบไฟล์ `vfat` เป็นส่วนต่อขยายของระบบไฟล์ FAT ซึ่งย่อมาจากคำว่า File Allocation Table เพื่อรองรับชื่อไฟล์ที่ยาวกว่า FAT ที่มา: [wikipedia](https://en.wikipedia.org/wiki/VFAT)
- ...

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 เพื่อดังค่าไม่ให้รันไฟล์ที่อยู่ในไดเรกทอรีนี้ได้ เช่น ไฟล์ที่เป็นไวรัสหรือมัลแวร์ (Malware) ที่แอบแฝงเข้ามา
- `nodev`: No Device หมายถึง การไม่อนุญาตให้สร้างหรืออ่านโนด (Node) ซึ่งเป็นไฟล์ชนิดพิเศษ
- `mode` หมายถึง สิทธิ์การเข้าถึงไฟล์หรือไดเรกทอรี ประกอบด้วย บิตควบคุม Read Write Execute 3 ชุด รวมทั้งหมด 9 บิต ซึ่งได้อธิบายแล้วในหัวข้อที่ 7.1.4

ผู้อ่านสามารถ แสดง ราย ชื่อ ไฟล์ หรือ ไดเรกทอรี หรือ ชื่อ อุปกรณ์ ภายใต้ไดเรกทอรี `/dev` โดยพิมพ์ คำสั่ง บน โปรแกรม Terminal

```
$ ls /dev
```

ผู้อ่านต้องเปรียบเทียบกับชื่ออุปกรณ์ที่ผู้เขียนตัวหนาไว้ว่าตรงกันหรือไม่ อย่างไร เพื่อให้ผู้อ่านมองเห็นชัดว่า `mmcblk0p2` มีอยู่จริงและระบบได้ทำการแมทช์เข้ากับไดเรกทอรีรูท (Root) นั่นคือ ไดเรกทอรี `/` ด้วยชนิด `ext4` ตามที่ได้แสดงในคำสั่งก่อนหน้านี้แล้ว

```
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
mmcblk0p1 mmcblk0p2 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-hevcmm 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 อุปกรณ์สำคัญอื่นๆ เช่น `stdin` (standard input) `stdout` (standard output) และ `stderr` (standard error) นั้นเกี่ยวข้องกับโปรแกรม Terminal ซึ่งเชื่อมโยงกับประโยคในภาษา C ในการทดลองที่ 5 ภาคผนวก E

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

```
t63010487@Pi432b:~ $ ls /dev
autofs      hwrng      net        random     ttyl6      tty3
block       initctl   null       raw        ttyl7      tty3
btrfs-control input     port       rfkill     ttyl8      tty3
bus         kmsg      ppp        rpivid-h264mem ttyl9      tty3
cachefiles  log       ptmx       rpivid-hevcmm tty2        tty4
char        loop0     pts        rpivid-intcmm tty20       tty4
console     loop1     ram0       rpivid-vp9mem tty21       tty4
cuse        loop2     ram1       serial1    tty22       tty4
disk        loop3     ram10      shm        tty23       tty4
dma_heap    loop4     ram11      snd        tty24       tty4
dri         loop5     ram12      stderr     tty25       tty4
fb0         loop6     ram13      stdin      tty26       tty4
fd          loop7     ram14      stdout     tty27       tty4
full        loop-control ram15      tty        tty28       tty4
fuse        mapper    ram2       tty0       tty29       tty4
gpiochip0   media0    ram3       tty1       tty3        tty5
gpiochip1   media1    ram4       tty10      tty30       tty5
gpiomem     mem       ram5       tty11      tty31       tty5
hidraw0     mmcblk0   ram6       tty12      tty32       tty5
hidraw1     mmcblk0p1 ram7       tty13      tty33       tty5
hidraw2     mmcblk0p2 ram8       tty14      tty34       tty5
hidraw3     mqueue    ram9       tty15      tty35       tty5
```

```
t63010487@Pi432b:~/asm/Lab12 $ ls -l
total 8
-rw-r--r-- 1 t63010487 t63010487 2 Apr  1 13:46 test2.txt
-rw-r--r-- 1 t63010487 t63010487 4 Apr  1 13:38 test.txt
```

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 *มีขนาด 2 bytes*
3. โปรดสังเกตว่าใน test.txt ที่สร้างด้วยโปรแกรม nano เราได้พิมพ์ประโยค fdd คิดเป็นจำนวน 3 ตัวอักษรฯ ละ 1 ไบต์เท่านั้น จงหาว่าไบต์ที่ 4 คือตัวอักษรใดในรูปที่ *2.12 null (เป็นช่องว่าง)*
4. เพิ่มจำนวนตัวอักษรไปเรื่อยๆ ใน test.txt จนทำให้ไฟล์มีขนาดมากกว่าเท่ากับ 4096 ไบต์ แล้วใช้คำสั่ง `du -B1 test.txt` **ตรวจสอบขนาดไฟล์อีกรอบ** บันทึกและอธิบายผลที่ได้โดยเฉพาะจำนวน Blocks ที่ได้จากคำสั่งว่าเท่ากับกี่**เซกเตอร์** *ได้ 48 block และ block 5 size sector*
5. จงเปรียบเทียบผลลัพธ์ของคำสั่ง `stat` ระหว่าง ไดเรกทอรี และ ไฟล์
6. สิทธิ์การเข้าถึง (Permission) ของไดเรกทอรีหรือของไฟล์ประกอบด้วยบิตจำนวน 9 บิต แบ่งเป็น 3 ชุดๆ ละ 3 บิต จงเรียงลำดับชุดต่างๆ ว่าเป็นของสิทธิ์ของใครบ้าง *owner, group of owner, Everyone*
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/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](#)