ยูนิกซ์ขั้นต้น

เทพพิทักษ์ การุญบุญญานันท์

สารบัญ

บทที่

1

เริ่มต้นกับยูนิกซ์

ยูนิกซ์เป็นระบบปฏิบัติการที่มีประวัติยาวนาน ถือได้ว่าเป็นระบบที่มีอายุการใช้งานนานที่สุด และแพร่หลายในเครื่องคอมพิวเตอร์ประเภทต่างๆ มากที่สุดเท่าที่มนุษย์เคยมีมา ไม่ว่าจะเป็นเครื่องในระดับมินิคอมพิวเตอร์ไปจนถึงอุปกรณ์พกพา ยูนิกซ์ถูกใช้งานในห้องวิจัย ในสถาบันการศึกษา ในงานทางทหาร เป็นจุดกำเนิดของอินเทอร์เน็ต และของภาษา C ซึ่งเป็นภาษาสำหรับเขียนโปรแกรมระบบที่แพร่หลายที่สุดในปัจจุบัน รวมทั้งเป็นต้นแบบสำหรับระบบปฏิบัติการอื่นๆ เช่น MS-DOS, MS-Windows, OS/2 ผ่านทางมาตรฐาน POSIX อีกด้วย

ยูนิกซ์เองมีการแตกแขนงออกไปมากมาย โดยเริ่มจากที่ AT&T Bell Lab โดย เคน ทอมป์สัน (Ken Thompson) ใน พ.ศ. 2512 ไปแตกหน่อและพัฒนาอย่างมโหฬารที่มหาวิทยาลัยแคลิฟอร์เนียที่เบอร์คลีย์เป็น BSD Unix (ซึ่งเป็นแบบฉบับของ FreeBSD, NetBSD, OpenBSD และยูนิกซ์เชิงพาณิชย์หลายตัวในปัจจุบัน) โดยต่อมา AT&T ก็ได้ผนวกส่วนขยายต่างๆ กลับเข้ามาใน AT&T Unix System V เช่นกัน ผู้ผลิตต่างๆ ก็ได้นำยูนิกซ์ไปใช้กับผลิตภัณฑ์ของตนมากมาย เช่น SunOS และ Solaris บนเครื่อง Sun SPARC, Ultrix บน DEC PDP-11, HP-UX ของ HP, Aix ของ IBM, Xenix ของไมโครซอฟท์ในยุคหนึ่ง (ซึ่งต่อมากลายเป็น SCO Unix สำหรับเครื่อง PC) ฯลฯ รวมทั้ง GNU/Linux หรือที่เราเรียกกันสั้นๆ ว่าลินุกซ์ที่เรากำลังพูดถึงในเอกสารฉบับนี้ ก็นับได้ว่าเป็นแขนงหนึ่งของระบบยูนิกซ์เช่นกัน โดยเป็นการเขียนขึ้นมาใหม่ทั้งหมดภายใต้เงื่อนไขของซอฟต์แวร์เสรี

การที่ยูนิกซ์ถูกใช้งานกว้างขวางเช่นนี้ ก็เป็นเพราะยูนิกซ์ถูกออกแบบมาด้วยปรัชญาที่เรียบง่าย ชัดเจน แต่ก็ยังยืดหยุ่นและมีประสิทธิภาพในการแก้ปัญหาที่ซับซ้อน ทำให้มีวัฒนธรรมการใช้งานเป็นแบบฉบับของตัวเอง ความคงทนอยู่ในวงการมากกว่า 30 ปีจึงทำให้ยูนิกซ์น่าศึกษาแม้สำหรับผู้ที่ไม่ได้ใช้ยูนิกซ์ก็ตาม

ในปัจจุบันนี้ แม้เดสก์ทอปลินุกซ์จะอำนวยความสะดวกต่อผู้ใช้ด้วยการสั่งงานผ่าน GUI ซึ่งมีการพัฒนาขึ้นมาเป็นลำดับตามยุคสมัย แต่โดยเนื้อแท้แล้ว การสั่งงานผ่าน GUI จะทำงานได้เฉพาะในรูปของโปรแกรมสำเร็จรูปที่นักพัฒนาได้ออกแบบไว้ให้เท่านั้น จึงทำงานได้ดีสำหรับงานที่มีการออกแบบรองรับไว้แล้ว เช่น การสร้างเอกสาร งานกราฟิก การสำเนา/เคลื่อนย้ายไฟล์ ฯลฯ แต่สำหรับงานอีกหลายประเภทยังต้องการความยืดหยุ่นมากกว่านั้น เช่น การแก้ไขหรือแปลงไฟล์ทีละหลายไฟล์ การประมวลผลไฟล์โดยผ่านขั้นตอนเป็นทอดๆ การทำงานในภาวะที่ไม่มี GUI (เช่น โปรแกรมที่ทำงานอัตโนมัติขณะบูต

เครื่อง หรือการดูแล server จากระยะไกล) ทั้งหมดนี้ สามารถทำได้ด้วยการสั่งงานผ่านบรรทัดคำสั่งของยูนิกซ์ ซึ่งมีความคล่องตัวสูงแม้สำหรับงานที่ไม่ใช่งานสำเร็จรูป

1.1 วิธีต่างๆ ที่จะป้อนคำสั่งยูนิกซ์

การป้อนคำสั่งยูนิกซ์จะป้อนผ่านตัวแปลคำสั่งที่เรียกว่า **เชลล์ (shell)** ซึ่งการจะได้มาซึ่งเชลล์นั้น ก็มีอยู่หลายวิธี เช่น

1. ล็อกอินใน text console
2. เปิด terminal emulator ใน X Window
3. เชื่อมต่อไปยังเครื่องปลายทางด้วย telnet, rlogin หรือ ssh
4. เรียกเชลล์ซ้อนเชลล์
5. ใช้ shell escape จากโปรแกรมบางโปรแกรม เช่น vi

ในที่นี้ขอให้ใช้สองวิธีแรกก่อน ส่วนวิธีอื่นๆ ที่เหลือนั้น เราจะได้พบต่อไป

1.2 คำสั่งเบื้องต้น

1.2.1 เขียนข้อความออกจอภาพ

เรามาเริ่มจากคำสั่งที่ทำให้เครื่องโต้ตอบอะไรกับคุณได้ก่อนก็แล้วกัน คุณสามารถสั่งแสดงข้อความใดๆ ออกทางจอภาพได้ด้วยคำสั่ง `echo`

ตัวอย่างเช่น

```
thep@anubis:~$ echo hello
hello
thep@anubis:~$ echo how   are you
how are you
thep@anubis:~$ echo "how   are you"
how   are you
```

คำสั่ง `echo` จะแสดงข้อความที่เป็นอาร์กิวเมนต์ (argument คือค่าที่ส่งให้กับคำสั่งใดๆ) ทั้งหมด โดยคั่นระหว่างอาร์กิวเมนต์แต่ละตัวด้วยช่องว่างหนึ่งช่อง สังเกตว่า ในคำสั่งแรก เราส่งอาร์กิวเมนต์ให้กับ `echo` เพียงค่าเดียว คำสั่งถัดมาส่งให้สามค่า คือ `how`, `are` และ `you` การเว้นช่องว่างระหว่าง `how` และ `are` จึงไม่มีผลใดๆ ในขณะที่คำสั่งสุดท้าย เราใช้เครื่องหมายคำพูดครอบข้อความทั้งหมด ทำให้อาร์กิวเมนต์เหลือเพียงค่าเดียว

> **คำสั่ง** `echo` – แสดงข้อความ
>
> **รูปแบบ** `echo` [*STRING*] . . .
>
> **คำบรรยาย** แสดง *STRING* ออกทางเอาท์พุตมาตรฐาน คั่นระหว่าง *STRING* แต่ละตัวด้วยช่องว่างหนึ่งช่อง

เมื่อเขียนจอภาพแล้ว คุณก็อาจจะอยากลบทิ้ง คำสั่งสำหรับลบจอภาพในยูนิกซ์ได้แก่คำสั่ง `clear` ซึ่งจะพิจารณาประเภทของจอที่คุณใช้ (เช่น เป็น xterm, Linux console หรือ VT100 ฯลฯ) และหาวิธีลบจอให้โดยอัตโนมัติ

คำสั่ง `clear` – ลบจอภาพ

รูปแบบ `clear`

คำบรรยาย ลบจอภาพตามประเภทของเทอร์มินัลที่ใช้

1.2.2 ดูวัน-เวลา

นักคอมพิวเตอร์หลายคนไม่เคยใช้นาฬิกาข้อมือเลย เพราะอาศัยดูเวลาจากเครื่องเอา ถึงคุณมีนาฬิกา คุณก็สามารถตรวจสอบนาฬิกาของเครื่องว่าเดินตรงหรือไม่ด้วยคำสั่ง `date`

```
thep@anubis:~$ date
Sat Jul  5 16:31:45 ICT 2003
```

อย่าแปลกใจถ้าคุณเห็นปฏิทินบนเครื่องของคุณเป็นภาษาไทย ไม่เหมือนกับตัวอย่างข้างบน นั่นแสดงว่าคุณได้ใช้ระบบที่ปรับแต่งให้เป็นไทยแล้ว

คำสั่ง `date` – แสดงวันที่และเวลาของระบบ

รูปแบบ `date`

คำบรรยาย แสดงวันที่และเวลาของระบบ

นอกจากแสดงวัน-เวลาปัจจุบันแล้ว คุณอาจดูปฏิทินของเดือนปัจจุบันได้ด้วยคำสั่ง `cal`

```
thep@anubis:~$ cal
     July 2003
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

หรือลองดูปฏิทินของเดือนอื่น

```
thep@anubis:~$ cal 2 2000
   February 2000
```

```
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29
```

ลองพิมพ์ `cal 2003` ที่เครื่องของคุณดูสิว่าจะเกิดอะไรขึ้น

คำสั่ง `cal` – แสดงปฏิทิน

รูปแบบ `cal` [[*month*] *year*]

คำบรรยาย แสดงปฏิทินของเดือนและปีที่กำหนด หากไม่กำหนดอะไรเลยจะแสดงปฏิทินของเดือนปัจจุบัน และหากระบุแต่ปีไม่ระบุเดือน จะแสดงปฏิทินของทั้งปี

1.2.3 ดูรายชื่อไฟล์

เราจะเริ่มสำรวจฮาร์ดดิสก์กันด้วยคำสั่งขอดูรายชื่อไฟล์ ได้แก่คำสั่ง `ls`

```
thep@anubis:~$ ls
a.out  Download  hello.c  Mail  Projects  tmp
```

ในบางระบบอาจมีการตั้งค่าไว้ให้ใช้สีบ่งบอกชนิดของไฟล์ เช่น สีน้ำเงินแทนไดเรกทอรี สีเขียวแทนไฟล์โปรแกรม สีปกติแทนไฟล์ข้อมูล เป็นต้น หากระบบของคุณไม่ได้ตั้งค่าเช่นนี้ไว้ คุณก็ยังสามารถดูชนิดของไฟล์ต่างๆ ได้ด้วยการเพิ่มตัวเลือก `-F`

```
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  Projects/  tmp/
```

จะเห็นว่า มีการเพิ่มเครื่องหมายพิเศษต่อท้ายชื่อไฟล์เพื่อบอกชนิด กล่าวคือ

`/` หมายถึงไดเรกทอรี
`*` หมายถึงไฟล์โปรแกรม
`@` หมายถึง symbolic link
`=` หมายถึง socket
`|` หมายถึง named pipe (FIFO)

โดยถ้าไม่มีเครื่องหมายพิเศษ จะหมายถึงไฟล์ข้อมูลปกติ ชนิดต่างๆ ของไฟล์ จะกล่าวในรายละเอียดในบทที่ 3

แต่ไฟล์ต่างๆ ก็ยังมีรายละเอียดมากกว่าชนิดของไฟล์ เช่น ขนาด เวลาปรับปรุงครั้งสุดท้าย ฯลฯ ซึ่งคุณสามารถขอดูรายละเอียดทั้งหมดได้ด้วยตัวเลือก `-l`

```
thep@anubis:~$ ls -l
total 28
-rwxr-xr-x   1 thep    users      4815 2002-08-23 14:49 a.out
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Download
-rw-r--r--   1 thep    users        77 2002-08-23 14:49 hello.c
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Mail
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Projects
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 tmp
```

บรรทัดแรก (`total 28`) จะแสดงจำนวนบล็อกทั้งหมดของไฟล์ในไดเรกทอรีนั้น โดยบล็อกหนึ่งโดยปกติจะมีขนาด 1024 ไบต์ ทั้งนี้ สำหรับไฟล์ที่เป็นไดเรกทอรีย่อย (เช่น `Download`) จะไม่รวมขนาดของไฟล์ต่างๆ ที่อยู่ใต้ไดเรกทอรีย่อยนั้น แต่จะนับเพียงขนาดของตัวไดเรกทอรีย่อย (ที่เก็บชื่อไฟล์ต่างๆ) เท่านั้น

จากนั้น ก็เป็นรายละเอียดของแต่ละไฟล์ในไดเรกทอรี โดยแต่ละรายการจะแบ่งเป็นส่วนต่างๆ (จากซ้ายไปขวา) ดังนี้

1. permission
2. จำนวนลิงก์ (ดูหัวข้อ 3.3)
3. ผู้ใช้ที่เป็นเจ้าของไฟล์
4. กลุ่มที่เป็นเจ้าของไฟล์
5. ขนาดของไฟล์
6. วัน-เวลาที่แก้ไขครั้งสุดท้าย
7. ชื่อไฟล์

permission แสดงสิทธิที่อนุญาตให้ผู้ใช้แต่ละคนเข้าถึงไฟล์ได้ ส่วนนี้จะประกอบด้วยตัวอักษร 10 ตัว โดยแบ่งออกเป็น 4 ส่วน ส่วนแรกเป็นชนิดของไฟล์ แสดงด้วยตัวอักษรตัวแรกตัวเดียว ส่วนอีก 3 ส่วนถัดไป แสดงด้วยตัวอักษรส่วนละ 3 ตัว เป็นสิทธิของผู้ใช้ที่เป็นเจ้าของไฟล์ สิทธิของผู้ใช้ในกลุ่มเดียวกับเจ้าของไฟล์ และสิทธิของผู้ใช้อื่นๆ ตามลำดับ

ชนิดของไฟล์ ได้แก่

`-` หมายถึงไฟล์ปกติ

`d` หมายถึงไดเรกทอรี

`l` หมายถึง symbolic link

`s` หมายถึง socket

`c` หมายถึง character device

`b` หมายถึง block device

`p` หมายถึง named pipe (FIFO)

ชนิดต่างๆ ของไฟล์ จะกล่าวในรายละเอียดในบทที่ 3

ส่วนสิทธิของผู้ใช้ที่แสดงด้วยอักษรสามตัวนั้น ตัวแรกจะหมายถึงสิทธิในการอ่าน (`r`) ตัวถัดมาหมายถึงสิทธิในการเขียน (`w`) ตัวสุดท้ายหมายถึงสิทธิในการ execute (`x`) ซึ่งหากในตำแหน่งนั้นๆ เป็นเครื่องหมายลบ (`-`) จะหมายถึงไม่อนุญาต หากอนุญาตก็จะแสดงค่า `r`, `w` หรือ `x` ตามลำดับ โดยในส่วนของสิทธิการ execute นั้น ในบางกรณีอาจใช้อักษรอื่น ได้แก่ `s`, `t`, `S`, `T` ซึ่งจะมีความหมายพิเศษ ดังจะได้กล่าวถึงในหัวข้อ 2.7.2

permission ของไฟล์นี้ สามารถเปลี่ยนได้ด้วยคำสั่ง `chmod` ดังจะกล่าวถึงในหัวข้อ 2.7.3

เราได้เห็นการใช้คำสั่ง `ls` ในการแสดงรายชื่อไฟล์ไปแล้ว แต่ที่ผ่านมา ยังมีไฟล์อีกส่วนหนึ่งที่ถูกซ่อนไว้ไม่ได้แสดงออกมา ไฟล์ที่ถูกซ่อนเหล่านี้จะมีชื่อขึ้นต้นด้วย "." คุณสามารถดูรายชื่อไฟล์เหล่านั้นได้โดยใช้ตัวเลือก `-a`

```
thep@anubis:~$ ls -a
.   .alias  .bash_profile  .cshrc    hello.c  Projects
..  a.out   .bashrc        Download  Mail     tmp
```

นอกจากนี้ คำสั่ง `ls` ยังสามารถตั้งตัวเลือกในการแสดงผลได้อีกหลายแบบ ดังแสดงในสรุปคำสั่ง

คำสั่ง `ls` – ดูรายชื่อไฟล์

รูปแบบ `ls` [ตัวเลือก] [*path*]

คำบรรยาย แสดงรายชื่อไฟล์ใน *path* ที่กำหนด โดยถ้าไม่ระบุ path จะหมายถึงไดเรกทอรีปัจจุบัน

ตัวเลือก `-1` แสดงหนึ่งรายการต่อบรรทัด (สำหรับส่งต่อให้โปรแกรมอื่น)
- `-a` แสดงทุกไฟล์รวมทั้งไฟล์ที่ซ่อนอยู่ด้วย
- `-F` แสดงป้ายบอกชนิดของไฟล์อย่างย่อต่อท้ายชื่อ
- `-i` แสดงหมายเลข i-node ของแต่ละไฟล์
- `-l` แสดงรายละเอียดของไฟล์อย่างยาว
- `-S` เรียงลำดับตามขนาดของไฟล์
- `-t` เรียงไฟล์ใหม่ขึ้นมาก่อน

1.2.4 ดูเนื้อหาไฟล์

คำสั่งสำหรับดูเนื้อหาไฟล์ของยูนิกซ์ออกจะชื่อแปลกสักหน่อย คือคำสั่ง `cat` เช่น เมื่อคุณจะสั่งแสดงเนื้อหาของไฟล์ `/etc/hostname` (ซึ่งเก็บชื่อเครื่องของคุณ) ออกทางจอภาพ:

```
thep@anubis:~$ cat /etc/hostname
anubis
```

ความจริงแล้ว คำสั่ง `cat` มาจากคำว่า catenate คือการต่อไฟล์เข้าด้วยกัน แล้วแสดงออกทาง standard output ตัวอย่างเช่น เมื่อมีไฟล์ `a` และ `b` คุณสามารถต่อไฟล์ทั้งสองเข้าด้วยกันออกทางจอภาพได้:

```
thep@anubis:~$ cat a
hello
thep@anubis:~$ cat b
bye
thep@anubis:~$ cat a b
hello
bye
```

ดังนั้น คุณจะเห็นว่า การแสดงเนื้อหาไฟล์หนึ่งไฟล์ออกทางจอภาพ จึงเป็นเพียงกรณีพิเศษของการต่อไฟล์เท่านั้นเอง คำสั่ง `cat` สามารถจะต่อไฟล์กี่ไฟล์เข้าด้วยกันก็ได้ และถ้าคุณเรียกคำสั่ง `cat` โดยไม่ระบุไฟล์ใดๆ เลย `cat` จะหมายถึง standard input (ได้แก่แป้นพิมพ์ ถ้าไม่มีการเปลี่ยนให้เป็นอย่างอื่น)

```
thep@anubis:~$ cat
hello
hello
bye
bye
Ctrl-d
thep@anubis:~$
```

คุณจะเห็นแมวตัวนี้พูดล้อเลียนคุณไปตลอดบรรทัดต่อบรรทัด เพราะมันพยายามต่อไฟล์ที่มีเพียงไฟล์เดียว คือแป้นพิมพ์ แล้วแสดงผลลัพธ์ออกทาง standard output ซึ่งได้แก่จอภาพ จนกว่าคุณจะกด `Ctrl-d` ซึ่งยูนิกซ์กำหนดให้หมายถึง end-of-file

การปล่อยให้แมวล้อเลียนอย่างนี้ ก็ใช่จะไร้สาระเสียทีเดียว เพราะถ้าคุณเปลี่ยนทิศทาง standard output จากจอภาพเป็นไฟล์แล้วล่ะก็ มันก็คือวิธีสร้างไฟล์ข้อความอย่างง่ายนั่นเอง (เราจะกล่าวถึงการเปลี่ยนทิศทางข้อมูลอีกครั้งในหัวข้อ 4.1):

```
thep@anubis:~$ cat > d
hello
anybody here?
bye
Ctrl-d
thep@anubis:~$ cat d
hello
anybody here?
bye
```

คุณจะได้เรียนรู้การใช้เอดิเตอร์ในการสร้างไฟล์ข้อความในโอกาสต่อไป ซึ่งจะช่วยให้คุณแก้ไขข้อความได้สะดวกกว่านี้ (อย่างน้อยๆ ก็สามารถกลับไปแก้ไขบรรทัดก่อนได้)

คำสั่ง `cat` – ต่อไฟล์แล้วแสดงออกทาง standard output

รูปแบบ `cat` [*file* ...]

คำบรรยาย ต่อไฟล์ที่กำหนดตามลำดับ (หากไม่ระบุไฟล์ใดๆ จะหมายถึง standard input) แล้วแสดงผลออกทาง standard output

คำสั่ง `cat` จะแสดงเนื้อหาไฟล์ออกทางจอภาพในฐานะที่เป็นไฟล์ไฟล์หนึ่ง โดยไม่มีปฏิสัมพันธ์กับแป้นพิมพ์ใดๆ ทั้งสิ้น แต่ถ้าคุณต้องการดูไฟล์ยาวๆ โดยให้มันหยุดทีละหน้า ยูนิกซ์ก็มีคำสั่ง `more` เพื่อการนี้

คำสั่ง more จะแสดงเนื้อหาไฟล์ทีละหน้า โดยจะแสดงคำว่า "--More--" ที่ท้ายหน้าเพื่อบอกว่า "มีต่อ" คุณสามารถควบคุมการแสดงผลได้ด้วยปุ่มต่างๆ ดังนี้

Enter	เลื่อนหน้าจอไป 1 บรรทัด
Space	เลื่อนหน้าจอไป 1 หน้า
/ *text* Enter	ค้นหา text
n	ค้นหาต่อ
q	ออกจากโปรแกรม

more จะออกจากโปรแกรมทันทีที่แสดงผลมาถึงท้ายไฟล์ หรือเมื่อคุณกด q

การเรียกคำสั่ง more สามารถเรียกโดยใส่ชื่อไฟล์ที่ต้องการดู โดยใส่ได้มากกว่า 1 ไฟล์ ซึ่งถ้าระบุมากกว่า 1 ไฟล์ ก็จะแสดงชื่อไฟล์ที่ต้นข้อความให้ด้วย และถ้าไม่ระบุชื่อไฟล์ จะหมายถึง standard input คล้าย cat ยกเว้นว่า more จะไม่สามารถทำงานกับ standard input ที่เป็นแป้นพิมพ์ได้ (เพราะแป้นพิมพ์เอาไว้ใช้รับคำสั่งเลื่อนจอไปแล้ว) จึงมักใช้ more ที่ไม่ระบุชื่อไฟล์ในกรณีที่รับข้อมูลต่อจากคำสั่งอื่นเท่านั้น เช่น ในการดูผลที่พ่นออกมาจากคำสั่งอื่นผ่าน pipe (เราจะกล่าวถึงการใช้ pipe โดยละเอียดในบทที่ 4)

```
thep@anubis:~$ ls -l /bin | more
total 2872
-rwxr-xr-x   1 root    root      47464 2002-08-04 03:15 afio
-rwxr-xr-x   1 root    root       2724 2002-01-27 14:05 arch
-rwxr-xr-x   1 root    root     579816 2002-09-12 05:51 bash
-rwxr-xr-x   1 root    root      13864 2002-08-04 16:10 cat
-rwxr-xr-x   1 root    root      16232 2002-09-01 20:39 chgrp
-rwxr-xr-x   1 root    root      16008 2002-09-01 20:39 chmod
-rwxr-xr-x   1 root    root      18120 2002-09-01 20:39 chown
-rwxr-xr-x   1 root    root      42664 2002-09-01 20:39 cp
-rwxr-xr-x   1 root    root      47656 2002-06-23 03:09 cpio
-rwxr-xr-x   1 root    root      82248 2002-09-14 19:32 dash
-rwxr-xr-x   1 root    root      34408 2002-09-01 20:40 date
-rwxr-xr-x   1 root    root      28808 2002-09-01 20:39 dd
-rwxr-xr-x   1 root    root      26184 2002-09-01 20:39 df
-rwxr-xr-x   1 root    root      59496 2002-09-01 20:39 dir
-rwxr-xr-x   1 root    root       4076 2002-01-27 14:05 dmesg
-rwxr-xr-x   1 root    root      11080 2002-09-01 20:40 echo
-rwxr-xr-x   1 root    root      43292 2000-11-27 09:05 ed
--More--
```

คำสั่ง more สามารถช่วยให้คุณดูไฟล์ยาวๆ โดยหยุดทีละหน้าได้ แต่ไม่สามารถย้อนกลับไปดูหน้าก่อนได้ (ในที่นี้หมายถึง more มาตรฐานของยูนิกซ์ ไม่รวมถึง more ในบางระบบที่มีการแก้ไขให้ดูหน้าก่อนได้) จึงได้มีการสร้างคำสั่งใหม่ คือ less ซึ่งสามารถเลื่อนหน้าขึ้น-ลงได้ การเรียกใช้คำสั่ง less ก็คล้ายกับ more ผิดกันที่ปุ่มที่ใช้ควบคุมหน้าจอของ less มากกว่า ตัวอย่างเช่น

Enter	เลื่อนหน้าจอลง 1 บรรทัด
Space	เลื่อนหน้าจอลง 1 หน้า
j	เลื่อนหน้าจอลง 1 บรรทัด
k	เลื่อนหน้าจอขึ้น 1 บรรทัด
f	เลื่อนหน้าจอลง 1 หน้า
b	เลื่อนหน้าจอขึ้น 1 หน้า
g	ไปที่ต้นไฟล์
G	ไปที่ท้ายไฟล์
m *letter*	mark ตำแหน่งปัจจุบันด้วย *letter*
' *letter*	ไปที่ตำแหน่งที่ mark ไว้
/ *text* Enter	ค้นหา text
? *text* Enter	ค้นหา text ย้อนหลัง
n	ค้นหาต่อในทิศทางเดิม
N	ค้นหาต่อในทิศทางตรงข้าม
q	ออกจากโปรแกรม

1.2.5 สำเนาไฟล์

บนยูนิกซ์ คุณสามารถทำสำเนาไฟล์ได้ด้วยคำสั่ง cp (copy)

```
thep@anubis:~$ ls
a
thep@anubis:~$ cat a
hello
thep@anubis:~$ cp a b
thep@anubis:~$ ls
a  b
thep@anubis:~$ cat b
hello
```

ตัวอย่างข้างต้นเป็นการสำเนาไฟล์ a ไปเป็นไฟล์ใหม่ชื่อ b โดยปกติแล้ว ถ้ามีไฟล์ b อยู่แล้ว b ก็จะถูกเขียนทับด้วยเนื้อหาใหม่ทันที ดังนั้น ควรระวังเรื่องนี้ให้ดี ถ้าจะให้ cp เตือนก่อนเขียนทับ ก็ใช้ตัวเลือก -i

```
thep@anubis:~$ ls
a  b
thep@anubis:~$ cp -i a b
cp: overwrite ‘b’? y
```

แต่ถ้าจะบังคับให้เขียนทับไฟล์ที่มีอยู่แล้วเสมอ ก็ใช้ตัวเลือก -f

นอกจากสำเนาไฟล์แล้ว cp ยังสามารถใช้สำเนาไดเรกทอรีทั้งชุดได้ด้วย โดยใช้ตัวเลือก -r ดังจะกล่าวต่อไปในหัวข้อที่ 1.3

1.2.6 เปลี่ยนชื่อไฟล์

อีกครั้งหนึ่งที่เราจะพบคำสั่งยูนิกซ์ที่ออกแบบมาในรูปทั่วไป คำสั่งที่ใช้เปลี่ยนชื่อไฟล์ในยูนิกซ์ ได้แก่คำสั่ง mv (move)

ความจริงแล้ว คำสั่ง mv มีหน้าที่ในการเคลื่อนย้ายไฟล์จากที่หนึ่งไปยังอีกที่หนึ่ง โดยสามารถตั้งชื่อใหม่ได้ เช่น คำสั่งต่อไปนี้เป็นการย้ายไฟล์ a ไปไว้ที่ /tmp ภายใต้ชื่อใหม่คือ b

```
thep@anubis:~$ mv a /tmp/b
```

ดังนั้น การย้ายไฟล์มาไว้ในไดเรกทอรีเดิมในชื่อใหม่ ก็คือการเปลี่ยนชื่อไฟล์นั่นเอง

```
thep@anubis:~$ mv a b
```

1.2.7 ลบไฟล์

คำสั่งสำหรับลบไฟล์บนยูนิกซ์ได้แก่คำสั่ง rm

```
thep@anubis:~$ rm a b
```

เป็นการลบไฟล์ a และ b

1.2.8 พิมพ์ไฟล์ออกเครื่องพิมพ์

เนื่องจากระบบยูนิกซ์เป็นระบบ multi-user จึงสนับสนุนการพิมพ์ผ่าน print queue ผู้ใช้แต่ละคนสามารถส่งงานไปเข้าคิวเพื่อรอส่งออกทางเครื่องพิมพ์ได้ด้วยคำสั่ง lpr เมื่อส่งไปแล้ว ถ้าคิวว่างอยู่ ก็จะถูกส่งไปยังเครื่องพิมพ์ทันที แต่ถ้าระบบกำลังพิมพ์งานชิ้นอื่นอยู่ ก็ต้องรอในคิวไปก่อน

ตัวอย่างเช่น ถ้าจะพิมพ์ไฟล์ output.ps ออกทางเครื่องพิมพ์ ก็ใช้คำสั่ง

```
thep@anubis:~$ lpr output.ps
```

ในบางกรณี ระบบของคุณอาจมีเครื่องพิมพ์มากกว่าหนึ่งตัว ซึ่งตรวจดูได้ที่ไฟล์ /etc/printcap

```
thep@anubis:~$ cat /etc/printcap
# Entry edited Tue Jun 10 15:28:13 2003 by foomatic-configure.
# Additional configuration atop /etc/foomatic/lpd/c82-foo.ppd
c82-foo||Epson Stylus C82|:\
    :filter_options= --lprng $J $Z /etc/foomatic/lpd/c82-foo.ppd:\
    :lf=/var/log/lp-errs:\
```

```
    :ppdfile=/etc/foomatic/lpd/c82-foo.ppd:\
    :sd=/var/spool/lpd/c82-foo:\
    :lp=/dev/lp0:\
    :if=/usr/bin/foomatic-rip:\
    :sh:\
    :force_localhost:\
    :mx#0:

c82:\
    :NUP=1:\
    :PRINTER_TYPE=LOCAL:\
    :RTLFTMAR=18:\
    :TOPBOTMAR=18:\
    :ASCII_TO_PS=YES:\
    :TEXT_SEND_EOF=NO:\
    :lp=/dev/lp0:\
    :RESOLUTION=1440x720:\
    :GSDEVICE=uniprint:\
    :DESIRED_TO=ps:\
    :PAPERSIZE=a4:\
    :lf=/var/log/lp-errs:\
    :sd=/var/spool/lpd/c82:\
    :PS_SEND_EOF=NO:\
    :sh:\
    :force_localhost:\
    :mx#0:
:
```

ซึ่งในที่นี้มี print queue สองคิว คือ `c82-foo` และ `c82` หากต้องการเจาะจงให้ออกยังคิวที่ต้องการ ก็สามารถระบุได้ด้วยตัวเลือก `-P` *`queue`* เช่น

```
thep@anubis:~$ lpr -P c82-foo output.ps
```

การพิมพ์โดยไม่ระบุชื่อคิว จะพิมพ์ออกทางเครื่องพิมพ์ที่ผู้ดูแลระบบตั้งไว้ให้เป็นคิวปริยาย

การตรวจสอบ print queue ว่ามีงานพิมพ์อะไรรออยู่บ้าง ใช้คำสั่ง `lpq`

```
thep@anubis:~$ lpq -P c82-foo
Printer: c82-foo@localhost 'Epson Stylus C82'
 Queue: 1 printable job
 Server: pid 2257 active
 Unspooler: pid 2258 active
 Status: processing 'dfA256localhost', size 674097, format 'f',
```

```
IF filter 'master-filter' at 11:32:58.948
 Rank    Owner/ID           Class Job Files        Size Time
active thep@localhost+256    A   256 output.ps  674097 11:32:58
```

จะเห็นว่า มี active job อยู่หนึ่งงาน หมายเลข job คือ 256 ไฟล์ที่พิมพ์คือ `output.ps` ขนาด 674,097 ไบต์ สั่งพิมพ์เมื่อเวลา 11.32 น.

หากทิ้งไว้นานแล้ว เห็นว่ายังไม่ออกที่เครื่องพิมพ์ อาจมีอะไรขัดข้องบางอย่าง หรือต้องการจะเลิกพิมพ์กลางคัน คุณก็สามารถยกเลิกได้ด้วยคำสั่ง `lprm` โดยระบุหมายเลข job ที่ต้องการ

```
thep@anubis:~$ lprm -P c82-foo 256
Printer: c82-foo@localhost 'Epson Stylus C82'
  checking perms 'thep@localhost+256'
  dequeued 'thep@localhost+256'
thep@anubis:~$ lpq -P c82-foo
Printer: c82-foo@localhost 'Epson Stylus C82'
 Queue: no printable jobs in queue
 Status: job 'thep@localhost+256' saved at 11:45:13.299
```

คำสั่ง `lprm -a` จะยกเลิกงานทุกงานในคิวที่เป็นของคุณ คุณจะไม่มีสิทธิยกเลิกงานพิมพ์ของผู้ใช้อื่น นอกเสียจากคุณจะเป็นผู้ดูแลระบบ

เช่นเดียวกับคำสั่ง `lpr` คำสั่ง `lpq` และ `lprm` จะระบุคิวด้วยตัวเลือก `-P` หรือไม่ก็ได้ โดยถ้าไม่ระบุ จะหมายถึงคิวปริยายที่ผู้ดูแลระบบตั้งไว้ (ปกติก็คือคิวแรกใน `/etc/printcap` นั่นเอง) หรือถ้าคุณต้องการตั้งค่าเครื่องพิมพ์ปกติของคุณเอง ก็สามารถทำได้โดยเซ็ตตัวแปรระบบชื่อ `PRINTER`

```
thep@anubis:~$ export PRINTER=c82
thep@anubis:~$ lpq
Printer: c82@localhost
 Queue: no printable jobs in queue
 Status: finished 'thep@localhost+870', status 'JFAIL' at 20:09:25.185
```

สังเกตที่บรรทัด `Printer:` ว่าได้เปลี่ยนเป็น `c82` โดยไม่ต้องใช้ตัวเลือก `-P`

1.3 ไดเรกทอรี

ระบบไฟล์ของยูนิกซ์มีการแบ่งแหล่งเก็บไฟล์เป็นชั้นๆ ทำให้จัดเก็บไฟล์ได้อย่างมีระเบียบ เปรียบเสมือนตู้เอกสารที่มีการแบ่งหมวดหมู่ ทำให้หาไฟล์ได้ง่าย หมวดหมู่หนึ่งๆ ที่ใช้เก็บไฟล์นี้ เรียกว่า *ไดเรกทอรี (directory)*[1] หน้าที่ของไดเรกทอรีก็คือ เก็บรายชื่อของไฟล์ และอาจเก็บไดเรกทอรีย่อยลงไปอีกก็ได้เช่นกัน

ระบบยูนิกซ์จะมีโครงสร้างของไดเรกทอรีหลักที่ตายตัว โดยเริ่มจากชั้นบนสุดที่เรียกว่า *ไดเรกทอรีราก (root directory)* จะอ้างถึงโดยเครื่องหมายขีดทับ หรือ slash (`/`)

[1] ใน MS-DOS จะใช้คำว่าไดเรกทอรีเช่นเดียวกับยูนิกซ์ ในขณะที่ MS Windows จะเริ่มใช้คำว่า *โฟลเดอร์ (folder)* ซึ่งหมายความถึงสิ่งเดียวกัน

```
thep@anubis:~$ ls -F /
bin/     etc/         lib/         root/  vmlinuz@
boot/    floppy/      lost+found/  sbin/  vmlinuz.old@
cdrom/   home/        mnt/         tmp/
cdrom0/  initrd/      opt/         usr/
dev/     initrd.img@  proc/        var/
```

จะเห็นว่า ไดเรกทอรีรากยังมี *ไดเรกทอรีย่อย (subdirectory)* อยู่ในนั้นอีกจำนวนหนึ่ง เมื่อจะอ้างถึงไดเรกทอรีย่อยนับจากรากก็อ้างได้ด้วยชื่อไดเรกทอรีย่อยหลัง slash และเมื่ออ้างถึงไฟล์หรือไดเรกทอรีย่อยลงไปอีก ก็คั่นแต่ละชั้นด้วย slash จนถึงชื่อไฟล์หรือไดเรกทอรีย่อยที่ต้องการ[2]

```
thep@anubis:~$ ls -F /usr/local
bin/  games/  include/  lib/  man/  sbin/  share/  src/
thep@anubis:~$ cat /etc/hostname
anubis
```

1.3.1 ไดเรกทอรีปัจจุบัน

เมื่อคุณอยู่ในเชลล์ของยูนิกซ์ คุณจะ “อยู่” ที่ไดเรกทอรีใดไดเรกทอรีหนึ่งเสมอ โดยที่เมื่อคุณอ้างชื่อไฟล์ลอยๆ โดยไม่ได้ระบุตำแหน่งแห่งที่ มันจะหมายถึงไฟล์ใน *ไดเรกทอรีปัจจุบัน (current directory)*

คุณสามารถขอดูค่าของไดเรกทอรีปัจจุบันได้ด้วยคำสั่ง `pwd` (print working directory)

```
thep@anubis:~$ pwd
/home/thep
```

1.3.2 การเปลี่ยนไดเรกทอรี

คุณสามารถเปลี่ยนไดเรกทอรีปัจจุบันไปที่อื่นได้ด้วยคำสั่ง `cd` (change directory)

```
thep@anubis:~$ cd /usr
thep@anubis:/usr$ pwd
/usr
thep@anubis:/usr$ ls -F
bin/    include/  local/        sbin/   X11R6/
doc/    info@     m68k-palmos/  share/
games/  lib/      openwin/      src/
thep@anubis:/usr$ cd local
```

[2]ใน MS-DOS และ MS Windows จะใช้ backslash แทน slash เพราะได้กำหนดให้ใช้ slash ในการระบุตัวเลือกในบรรทัดคำสั่งไปแล้ว เช่น `dir /w` เป็นต้น

```
thep@anubis:/usr/local$ pwd
/usr/local
```

จะเห็นว่า คุณสามารถอ้างไดเรกทอรีในแบบสัมบูรณ์ (absolute) เช่น `/usr` หรือในแบบสัมพัทธ์ (relative) กับไดเรกทอรีปัจจุบัน เช่น `local` ก็ได้

นอกจากนี้ การสั่ง `cd` โดยไม่ระบุอาร์กิวเมนต์จะหมายถึงการเปลี่ยนไดเรกทอรีกลับไปที่ *ไดเรกทอรีบ้าน (home directory)* ของคุณ ซึ่งไดเรกทอรีบ้านนี้ ผู้ใช้แต่ละคนจะมีเป็นของตัวเอง กำหนดให้โดยผู้ดูแลระบบ

การเปลี่ยนไดเรกทอรีด้วย `cd` จะเปลี่ยนไดเรกทอรีปัจจุบันไป ณ ตำแหน่งใหม่ แต่เมื่อคุณใช้ยูนิกซ์ไปสักพัก คุณจะเริ่มพบว่ามีหลายครั้งที่คุณแค่อยากเปลี่ยนไดเรกทอรีไปทำงานที่อื่นเพียงชั่วคราวแล้วกลับมาทำงานต่อที่เดิม ถ้าใช้คำสั่ง `cd` คุณก็จำ (หรือจด) ไดเรกทอรีปัจจุบันไว้ก่อนเปลี่ยนไปที่ใหม่ จากนั้นก็ค่อยเปลี่ยนกลับมาที่เดิมที่จำ (หรือจด) ไว้ หรือไม่ก็ใช้ชุดคำสั่ง `pushd` และ `popd`

`pushd` และ `popd` จะทำงานเป็นสแต็ก (stack–กองซ้อนที่วางทับๆ กัน โดยเวลาหยิบ ชิ้นบนสุดที่วางหลังสุดจะถูกหยิบก่อน) โดย `pushd` จะเก็บค่าไดเรกทอรีปัจจุบันใส่สแต็กไว้ ก่อนจะเปลี่ยนไดเรกทอรีไป และคำสั่ง `popd` จะหยิบค่าในสแต็กล่าสุดขึ้นมาแล้วเปลี่ยนไดเรกทอรีไป

```
thep@anubis:~$ pushd /usr/local
/usr/local ~
thep@anubis:/usr/local$ popd
~
thep@anubis:~$
```

สังเกตว่า `popd` ไม่ต้องการอาร์กิวเมนต์ และหลังการสั่ง `pushd` และ `popd` แต่ละครั้ง เชลล์จะรายงานค่าในสแต็กให้ทราบ ดังเช่นในการสั่ง `pushd` ในตัวอย่างข้างต้นจะทำให้สแต็กเก็บค่าสองค่า คือ ไดเรกทอรีใหม่ที่ย้ายไป และไดเรกทอรีเก่า เมื่อสั่ง `popd` ก็จะเลิกค่าล่าสุดในสแต็กและใช้ค่าถัดมาแทน

คุณสามารถขอดูค่าสแต็กของไดเรกทอรีนี้ได้ทุกเวลาด้วยคำสั่ง `dirs`

```
thep@anubis:~$ pushd /usr/local
/usr/local ~
thep@anubis:~$ dirs
/usr/local ~
thep@anubis:/usr/local$ popd
~
thep@anubis:~$ dirs
~
thep@anubis:~$
```

1.3.3 “.” และ “..”

ในแต่ละไดเรกทอรี จะมีการอ้างอิงไปยังไดเรกทอรีปัจจุบันและไดเรกทอรีเบื้องบนภายใต้ชื่อ “.” และ “..” ตามลำดับ ซึ่งทำให้คุณสามารถอ้างไดเรกทอรีแบบสัมพัทธ์ได้ยืดหยุ่นยิ่งขึ้น

เราทราบแล้วว่า การอ้างชื่อพาธโดยไม่ได้ขึ้นต้นด้วย "/" นั้น จะอ้างอิงเทียบกับไดเรกทอรีปัจจุบันเสมอ ซึ่งทำให้เราอ้างถึงไฟล์หรือไดเรกทอรีที่อยู่ในระดับล่างลงไปได้ แต่เมื่อจะอ้างถึงไฟล์หรือไดเรกทอรีโดยอ้างอิงจากไดเรกทอรีเบื้องบนขึ้นไป ก็สามารถใช้ ".." ช่วยได้

```
thep@anubis:~$ cd /usr/local
thep@anubis:/usr/local$ ls ..
bin  games    info  local        openwin  share  X11R6
doc  include  lib   m68k-palmos  sbin     src
thep@anubis:/usr/local$ ls ../games
3Dc             gnome-stones  mahjongg       xdemineur
banner          gnometris     nethack        xdigger
cgoban          gnomine       nethack-gnome  xgal
cmail           gnuchess      pxboard        xgalaga
debugboard      gnuchessx     same-gnome     xtet42
gnibbles        gnugo         sol            zic2xpm
gnobots2        grab_cgoban   tint
gnome-freecell  iagno         xboard
thep@anubis:/usr/local$ cd ..
thep@anubis:/usr$ pwd
/usr
```

ส่วน "." นั้นจะอ้างอิงถึงไดเรกทอรีปัจจุบัน ซึ่งดูเผินๆ อาจจะดูไม่มีประโยชน์อะไรนัก เพราะการอ้างถึง "./file" ก็มักจะเทียบเท่ากับการอ้างถึง "file" เฉยๆ อยู่แล้ว แต่คุณจะเห็นประโยชน์ของ "." เมื่อใช้ยูนิกซ์ไปสักพัก ตัวอย่างเช่น การเรียกโปรแกรมหรือสคริปต์ที่อยู่ในไดเรกทอรีปัจจุบัน หรือการลบไฟล์ที่ชื่อขึ้นต้นด้วยเครื่องหมายลบ (-) เป็นต้น

1.3.4 การสร้างไดเรกทอรี

คำสั่งสร้างไดเรกทอรีสำหรับยูนิกซ์คือ `mkdir` ซึ่งสามารถสั่งสร้างในลักษณะสัมบูรณ์หรือสัมพัทธ์ก็ได้

```
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  Projects/  tmp/
thep@anubis:~$ mkdir MyDoc
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  MyDoc/  Projects/  tmp/
thep@anubis:~$ mkdir /var/tmp/MyDoc
thep@anubis:~$ ls -F /var/tmp
MyDoc/
```

ทั้งนี้ ในการสั่งสร้างไดเรกทอรีแบบสัมบูรณ์ จะต้องมีไดเรกทอรีเบื้องบนอยู่ก่อน มิฉะนั้นจะไม่สามารถสร้างได้

```
thep@anubis:~$ mkdir /var/tmp/where/is/MyDoc
mkdir: cannot create directory ‘/var/tmp/where/is/MyDoc’: No such
 file or directory
```

อย่างไรก็ดี mkdir ของ GNU มีตัวเลือก -p เพื่อสร้างไดเรกทอรีเบื้องบนให้ด้วย

```
thep@anubis:~$ mkdir -p /var/tmp/where/is/MyDoc
```

1.3.5 การสำเนาและย้ายไฟล์ข้ามไดเรกทอรี

คำสั่ง cp และ mv นั้น มีอีกรูปแบบหนึ่งของการเรียกนอกเหนือจากการระบุชื่อไฟล์ต้นทางและปลายทาง คือการระบุปลายทางเป็นไดเรกทอรี ซึ่งในรูปแบบนี้ จะระบุไฟล์ต้นทางได้มากกว่าหนึ่งไฟล์ โดย cp (หรือ mv) จะสำเนา (หรือย้าย) ไฟล์ต้นทางทั้งหลายนั้นไปยังไดเรกทอรีปลายทางภายใต้ชื่อไฟล์เดิม

```
thep@anubis:~$ ls MyDoc
thep@anubis:~$ cp /etc/hostname /etc/hosts MyDoc
thep@anubis:~$ ls MyDoc
hostname  hosts
thep@anubis:~$ ls -F .
a.out*  Download/  hello.c  Mail/  MyDoc/  Projects/  tmp/
thep@anubis:~$ mv a.out hello.c MyDoc
thep@anubis:~$ ls -F .
Download/  Mail/  MyDoc/  Projects/  tmp/
thep@anubis:~$ ls MyDoc
a.out  hello.c  hostname  hosts
```

1.3.6 การสำเนาไดเรกทอรี

คำสั่งที่ใช้สำเนาไดเรกทอรีอย่างง่าย ได้แก่ cp -r โดยระบุต้นทางและปลายทางเป็นไดเรกทอรีแทนที่จะเป็นไฟล์อย่างการสำเนาไฟล์ ตัวเลือก -r (recursive) จะสำเนาทุกๆ ไฟล์และไดเรกทอรีย่อยในไดเรกทอรีต้นทางไปยังไดเรกทอรีปลายทาง

```
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  MyDoc/  Projects/  tmp/
thep@anubis:~$ cp -r /usr/games mygames
thep@anubis:~$ ls -F
a.out*     hello.c  MyDoc/    Projects/
Download/  Mail/    mygames/  tmp/
thep@anubis:~$ ls mygames
3Dc             gnome-stones  mahjongg       xdemineur
banner          gnometris     nethack        xdigger
```

```
cgoban          gnomine       nethack-gnome  xgal
cmail           gnuchess      pxboard        xgalaga
debugboard      gnuchessx     same-gnome     xtet42
gnibbles        gnugo         sol            zic2xpm
gnobots2        grab_cgoban   tint
gnome-freecell  iagno         xboard
```

1.3.7 การลบไดเรกทอรี

ยูนิกซ์จะลบไดเรกทอรีด้วยคำสั่ง `rmdir` ซึ่งจะใช้สำหรับไดเรกทอรีเปล่าเท่านั้น กล่าวคือ คุณจะต้องลบไฟล์ทั้งหลายภายใต้ไดเรกทอรีที่จะลบนั้นให้หมดเสียก่อน

```
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  MyDoc/  Projects/  tmp/
thep@anubis:~$ cp /etc/hostname MyDoc
thep@anubis:~$ ls MyDoc
hostname
thep@anubis:~$ rmdir MyDoc
rmdir: 'MyDoc': Directory not empty
thep@anubis:~$ rm MyDoc/*
thep@anubis:~$ rmdir MyDoc
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  Projects/  tmp/
```

การไม่อนุญาตให้ลบไดเรกทอรีที่มีข้อมูลนี้ก็มีข้อดีตรงที่ลดโอกาสที่ข้อมูลจะหายโดยไม่เจตนาลงได้ อย่างไรก็ดี ยังมีวิธีที่จะลบไดเรกทอรีพร้อมทั้งไฟล์และไดเรกทอรีย่อยทั้งหมดที่อยู่ในนั้น โดยใช้ตัวเลือก `-r` ในคำสั่ง `rm` ธรรมดา

```
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  MyDoc/  Projects/  tmp/
thep@anubis:~$ ls MyDoc
hostname
thep@anubis:~$ rm -r MyDoc
thep@anubis:~$ ls -F
a.out*  Download/  hello.c  Mail/  Projects/  tmp/
```

1.4 การตั้งชื่อไฟล์

ยูนิกซ์กำหนดให้ไฟล์มีชื่อยาวได้ถึง 14 ตัวอักษร ถึงแม้หลายระบบ (เช่น GNU/Linux) ได้ขยายให้ใช้ได้ถึง 256 ตัวอักษร แต่ชื่อไฟล์ที่ยาวไม่เกิน 14 ตัวอักษรจะใช้ได้กับยูนิกซ์ทุกระบบ ตัวอักษรที่อนุญาตให้ใช้ตั้งชื่อไฟล์ได้อย่างปลอดภัย ได้แก่

- ตัวอักษรภาษาอังกฤษ a-z และ A-Z โดยตัวเล็ก-ตัวใหญ่จะถือว่าต่างกัน กล่าวคือ `abc`, `ABC` และ `Abc` ถือว่าเป็นไฟล์ที่ต่างกัน
- ตัวเลข 0-9
- เครื่องหมาย `. , _`

ส่วนเครื่องหมายอื่นๆ เช่น `; $ ‘ " ’ | ( ) ~ * & # - < > ? / ` อาจมีปัญหาเมื่อนำมาตั้งเป็นชื่อไฟล์ เนื่องจากเป็นเครื่องหมายพิเศษที่เชลล์ตีความ โดยเฉพาะเครื่องหมาย `/` นั้น ระบบจะตีความเป็นเครื่องหมายแบ่งชื่อไดเรกทอรีเสมอ

เครื่องหมายจุด (`.`) นั้น นิยมใช้ตั้งนามสกุลของไฟล์เพื่อบอกชนิด เช่น `foo.c` เป็นไฟล์ซอร์สโค้ดภาษาซี `foo.jpg` เป็นไฟล์รูปภาพแบบ JPEG เป็นต้น แต่ถ้าเครื่องหมาย `.` อยู่หน้าสุด เช่น `.bashrc` จะทำให้ไฟล์นั้นถูกซ่อนจากคำสั่ง `ls` ปกติ ต้องใช้ตัวเลือก `-a` จึงจะเห็น

1.5 wildcard

หลายต่อหลายครั้งที่เราต้องสั่งงานกับไฟล์เป็นชุดๆ ในยูนิกซ์จะมีวิธีอ้างไฟล์เป็นชุดๆ ที่มีชื่อสอดคล้องกับรูปแบบที่กำหนด รูปแบบดังกล่าวเรียกว่า *wildcard* เป็นการอ้างชื่อไฟล์โดยละบางส่วนให้แปรเปลี่ยนเป็นค่าต่างๆ ได้ เชลล์จะเป็นผู้กระจาย wildcard โดยค้นหาไฟล์ทั้งหมดในระบบไฟล์หรือไดเรกทอรีที่กำหนดที่มีชื่อตามรูปแบบที่กำหนดโดย wildcard

การกำหนด wildcard ใช้สัญลักษณ์พิเศษต่อไปนี้

`*` แทนตัวอักษรใดๆ ตั้งแต่ 0 ตัวขึ้นไปกี่ตัวก็ได้

`?` แทนตัวอักษรใดๆ 1 ตัว

`[ ]` แทนตัวอักษรใดๆ ในชุดที่กำหนด 1 ตัว

เพื่อเป็นตัวอย่าง เราจะสร้างไดเรกทอรีทดสอบดังนี้

```
thep@anubis:~/starwars$ touch starwars prologue.txt episode1.txt
episode2.txt episode3.txt episode4.txt episode5.txt episode6.txt
episodeX.txt epilogue.txt
thep@anubis:~/starwars$ ls
epilogue.txt  episode3.txt  episode6.txt  starwars
episode1.txt  episode4.txt  episodeX.txt
episode2.txt  episode5.txt  prologue.txt
```

`*.txt` หมายถึงไฟล์ที่ลงท้ายด้วย “`.txt`”

```
thep@anubis:~/starwars$ ls *.txt
epilogue.txt  episode3.txt  episode6.txt
episode1.txt  episode4.txt  episodeX.txt
episode2.txt  episode5.txt  prologue.txt
```

`episode?.txt` หมายถึงไฟล์ที่ขึ้นต้นด้วย “`episode`” ตามด้วยอักษรอะไรก็ได้ 1 ตัว ตามด้วย “`.txt`”

```
thep@anubis:~/starwars$ ls episode?.txt
episode1.txt  episode3.txt  episode5.txt  episodeX.txt
episode2.txt  episode4.txt  episode6.txt
```

`episode[0-9].txt` หมายถึงไฟล์ที่ขึ้นต้นด้วย “episode” ตามด้วยตัวเลข 1 ตัว ตามด้วย “.txt”

```
thep@anubis:~/starwars$ ls episode[0-9].txt
episode1.txt  episode3.txt  episode5.txt
episode2.txt  episode4.txt  episode6.txt
```

`epi*` หมายถึงไฟล์ที่ขึ้นต้นด้วย “epi”

```
thep@anubis:~/starwars$ ls epi*
epilogue.txt  episode2.txt  episode4.txt  episode6.txt
episode1.txt  episode3.txt  episode5.txt  episodeX.txt
```

`*logue*` หมายถึงไฟล์ที่มีคำว่า “logue” อยู่ในชื่อ

```
thep@anubis:~/starwars$ ls *logue*
epilogue.txt  prologue.txt
```

เพื่อเป็นการแสดงให้ผู้อ่านมองเห็นกลไกของการกระจาย wildcard ใกล้ชิดขึ้นอีกนิด เราลองใช้ wildcard กับคำสั่ง echo ดังนี้

```
thep@anubis:~/starwars$ echo *.txt
epilogue.txt episode1.txt episode2.txt episode3.txt episode4.txt
episode5.txt episode6.txt episodeX.txt prologue.txt
```

เราได้ทราบในหัวข้อ 1.2.1 แล้วว่าคำสั่ง echo จะทำหน้าที่เพียงแค่แสดงค่าต่างๆ ที่เป็นอาร์กิวเมนต์ออกทางจอภาพ คั่นด้วยช่องว่างหนึ่งช่อง ผลลัพธ์ที่ได้จึงแสดงให้เห็นว่า echo ไม่ได้รับอาร์กิวเมนต์เป็น “*.txt” แต่เป็นชื่อไฟล์ทั้งหมดที่กระจายแล้ว ซึ่งการกระจายดังกล่าวเป็นหน้าที่ของเชลล์ ดังนั้น คุณคงจะพอมองเห็นแล้วว่า คำสั่ง ls หรือโปรแกรมใดๆ บนยูนิกซ์ ไม่จำเป็นต้องรู้วิธีกระจาย wildcard เลย

ข้อดีของการออกแบบยูนิกซ์ให้เชลล์เป็นผู้กระจาย wildcard แบบนี้ก็คือ ลดความซ้ำซ้อนของโปรแกรมต่างๆ ในอันที่จะต้องกระจาย wildcard รวมทั้งทำให้รูปแบบการใช้ wildcard เหมือนกันหมดทุกโปรแกรม ทำให้ง่ายต่อการเรียนรู้ของผู้ใช้อีกด้วย

จากตัวอย่างของ echo ข้างต้น หากคุณไม่ต้องการให้เชลล์กระจาย wildcard เพื่อให้ echo แสดงข้อความ “*.txt” จริงๆ ก็มีวิธีทำได้หลายวิธี ได้แก่

1. ใช้ backslash (\) กำกับหน้าเครื่องหมายพิเศษของ wildcard

```
thep@anubis:~/starwars$ echo \*.txt
*.txt
```

```
thep@anubis:~/starwars$ echo episode\[0-9\].txt
episode[0-9].txt
```

2. ใช้อัญประกาศคู่ (" ") คร่อมนิพจน์ทั้งหมด

```
thep@anubis:~/starwars$ echo "*.txt"
*.txt
thep@anubis:~/starwars$ echo "episode[0-9].txt"
episode[0-9].txt
```

3. ใช้อัญประกาศเดี่ยว (' ') คร่อมนิพจน์ทั้งหมด

```
thep@anubis:~/starwars$ echo '*.txt'
*.txt
thep@anubis:~/starwars$ echo 'episode[0-9].txt'
episode[0-9].txt
```

การใช้อัญประกาศคู่และอัญประกาศเดี่ยวจะให้ผลเหมือนกันในกรณีของ wildcard แต่จะต่างกันเมื่อมีการใช้ตัวแปรเชลล์ ดังจะกล่าวในโอกาสต่อไป

บทที่

2

ระบบรักษาความปลอดภัย

ยูนิกซ์เป็นระบบที่ถือกำเนิดมาเพื่อใช้กับเครื่องขนาดใหญ่ที่รองรับผู้ใช้หลายคนในเวลาเดียวกัน (เรียกว่าเป็นระบบ multi-user) ลักษณะการใช้ยูนิกซ์ของผู้ใช้ในระบบเหล่านั้นจึงคล้ายกับการเช่าคอนโดมิเนียมร่วมกันมากกว่าการเป็นเจ้าของบ้านทั้งหลังเหมือนการใช้ระบบดอสหรือวินโดวส์กับเครื่องคอมพิวเตอร์ส่วนตัว ดังนั้น ยูนิกซ์จึงต้องมีระบบรักษาความปลอดภัยเพื่อปกป้องความเป็นส่วนตัวของผู้ใช้แต่ละคนเป็นคุณสมบัติขั้นพื้นฐาน แม้ปัจจุบันเมื่อนำมาใช้งานแบบ stand-alone ตามเครื่องพีซีตามบ้านก็ตาม ระบบดังกล่าวก็ยังมีประโยชน์ในด้านการลดความเสียหายที่ผู้ใช้อาจกระทำต่อระบบโดยไม่ตั้งใจ รวมถึงการจำกัดขอบเขตของผลกระทบของโปรแกรมประเภทไวรัส จนทำให้แรงจูงใจในการสร้างไวรัสสำหรับยูนิกซ์แทบไม่มีเอาเสียเลย

2.1 ผู้ใช้และกลุ่ม

สิ่งแรกที่จะใช้จำแนกแยกแยะผู้ใช้ในระบบก็คือ *บัญชีผู้ใช้ (user account)* ซึ่งโดยหลักการก็คล้ายกับบัญชีธนาคารมาก ผู้ใช้แต่ละคนจะมีชื่อบัญชีและหมายเลขประจำตัว (user ID หรือ uid) ทรัพยากรทุกอย่างในระบบจะมีเจ้าของทั้งนั้น โดยระบุเจ้าของด้วยหมายเลขประจำตัวนี้เอง

โดยปกติ เมื่อผู้ใช้เปิดบัญชีในระบบแล้ว ก็จะมี “บ้าน” ของตัวเองในระบบ เรียกว่า *ไดเรกทอรีบ้าน (home directory)* ไว้สำหรับใช้งานด้วย เมื่อจะเข้าใช้ระบบ ผู้ใช้จะต้อง *ลงบันทึกเข้าใช้ระบบ (login)* เสียก่อน เมื่อลงบันทึกแล้ว ก็จะเข้ามาอยู่ที่ไดเรกทอรีบ้านเป็นอันดับแรก

นอกจากบัญชีผู้ใช้แล้ว ในระบบยูนิกซ์ยังสามารถตั้ง *กลุ่ม (group)* ของผู้ใช้ได้อีกด้วย โดยผู้ดูแลระบบจะเป็นผู้ดำเนินการให้ กลุ่มที่ตั้งขึ้นอาจจะตั้งเพื่อทำงานชิ้นหนึ่งๆ ร่วมกัน หรือเพื่อแยกประเภทผู้ใช้ให้มีสิทธิเข้าถึงทรัพยากรต่างกันก็ได้ ไฟล์ต่างๆ ในระบบนอกจากจะมีผู้ใช้เป็นเจ้าของแล้ว ยังมีกลุ่มเป็นเจ้าของอีกด้วย โดยผู้ใช้สามารถกำหนดได้ว่าจะให้สิทธิผู้ใช้ในกลุ่มเดียวกันเข้าถึงได้หรือไม่ ดังจะกล่าวต่อไปในหัวข้อ 2.7

ผู้ใช้คนหนึ่งๆ สามารถอยู่ในกลุ่มมากกว่าหนึ่งกลุ่มได้ โดยจะมีกลุ่มโดยปริยายอยู่กลุ่มหนึ่งเสมอ

2.2 การตั้งรหัสผ่าน

เพื่อเป็นการแสดงสิทธิในไฟล์ต่างๆ ของระบบ ผู้ใช้แต่ละคนจึงต้องมี *รหัสผ่าน (password)* ในการเข้าใช้ระบบ ซึ่งโดยปกติ ผู้ดูแลระบบจะตั้งรหัสผ่านมาให้ แต่ผู้ใช้ก็สามารถเปลี่ยนรหัสผ่านของตัวเองได้ (ซึ่งควรเปลี่ยนทันทีที่มีโอกาส) ด้วยคำสั่ง `passwd`

```
thep@anubis:~$ passwd
Changing password for thep
(current) UNIX password: (ป้อนรหัสผ่านเดิม ซึ่งจะไม่ปรากฏบนจอภาพ)
Enter new UNIX password: (ป้อนรหัสผ่านใหม่ ซึ่งจะไม่ปรากฏเช่นกัน)
Retype new UNIX password: (ป้อนรหัสผ่านใหม่ซ้ำอีกครั้งเพื่อยืนยัน)
passwd: password updated successfully
```

รหัสผ่าน ถือเป็นสิ่งสำคัญที่เป็นความลับของทั้งผู้ใช้และของระบบเอง การที่ผู้บุกรุกที่ไม่ประสงค์ดีสามารถเข้าระบบได้ แม้จะไม่ถึงกับขโมยสิทธิผู้ดูแลระบบได้ แต่ก็เป็นการเพิ่มโอกาสในการวางกับดักเพื่อเจาะทำลายระบบได้เป็นอย่างดี ดังนั้น ในระบบที่มีผู้ใช้หลากหลาย ผู้ใช้จึงไม่ควรบอกรหัสผ่านกับใคร รวมทั้งไม่ควรตั้งรหัสผ่านให้เดาง่ายเกินไป (เช่น ใช้ชื่อตัว ชื่อแฟน ชื่อสัตว์เลี้ยง สะกดกลับหน้ากลับหลัง) รหัสผ่านที่ดีควรมีความยาวตั้งแต่แปดตัวอักษรขึ้นไป และไม่ควรเป็นคำที่มีในพจนานุกรม เพื่อป้องกันการแกะรหัสผ่านด้วยโปรแกรมอัตโนมัติ โดยถ้ามีตัวเลขหรือเครื่องหมายพิเศษปนอยู่ด้วย จะทำให้เดาได้ยากยิ่งขึ้น

ในบางระบบ ผู้ดูแลระบบจะบังคับนโยบายในการตั้งรหัสผ่านด้วยคำสั่ง `passwd` โดยอาจเตือนหรือไม่ยอมให้ผ่านถ้ารหัสสั้นเกินไปหรือเดาได้ง่ายด้วยโปรแกรมอัตโนมัติ (เช่น เป็นคำในพจนานุกรมเป็นต้น)

2.3 สารบบผู้ใช้

รายชื่อผู้ใช้ทั้งหมดในระบบจะเก็บไว้ที่ไฟล์ `/etc/passwd` หนึ่งคนหนึ่งบรรทัด โดยแบ่งเป็นฟิลด์ต่างๆ คั่นด้วยทวิภาค (colon) ดังตัวอย่าง

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
...
thep:x:500:100:Theppitak Karoonboonyanan,,,:/home/thep:/bin/bash
```

ข้อมูลในฟิลด์ต่างๆ ได้แก่

1. *ชื่อบัญชี (user)* เช่น thep
2. *รหัสผ่าน (password)* โดยจะเก็บในรูปแบบที่เข้ารหัสทางเดียวไว้ แต่สามารถแยกเก็บในไฟล์ `/etc/shadow` ต่างหากซึ่ง root อ่านได้คนเดียวได้ เพื่อความปลอดภัยจากการเดารหัสผ่านด้วยโปรแกรมอัตโนมัติ ดังในตัวอย่างข้างต้น โดยฟิลด์นี้จะมีค่าเป็น `x` หมด

3. *หมายเลขประจำตัว (user id)*
4. *หมายเลขกลุ่ม (group id)* หมายถึงกลุ่มโดยปริยาย แต่ผู้ใช้สามารถเข้าร่วมในกลุ่มอื่นอีกได้ ซึ่งจะกำหนดไว้ในไฟล์ `/etc/group` ดังจะกล่าวต่อไป
5. *ข้อมูลผู้ใช้ (comment)* เก็บชื่อ-สกุลของผู้ใช้ โดยอาจเก็บข้อมูลอื่นๆ อีกสามอย่าง คั่นด้วยจุลภาค (comma) ได้แก่ หมายเลขห้องทำงาน หมายเลขโทรศัพท์ที่ทำงาน หมายเลขโทรศัพท์ที่บ้าน
6. *ไดเรกทอรีบ้าน (home directory)*
7. *เชลล์ (shell)* ได้แก่โปรแกรมที่จะทำงานเพื่อรับคำสั่งของผู้ใช้เมื่อลงบันทึกเข้าระบบ

ผู้ใช้รายแรกๆ ถัดจาก root ในตัวอย่างข้างต้น เช่น daemon, bin, sys เป็นผู้ใช้ที่สงวนไว้ใช้วิ่งโปรแกรมให้บริการต่างๆ ไม่ใช่ผู้ใช้ที่เป็นบุคคล ข้อมูลในฟิลด์ของเชลล์จึงอาจเป็นคำสั่งที่ไม่ใช่เชลล์ โดยบัญชีที่ห้ามลงบันทึกเข้าใช้เด็ดขาดมักจะมีเชลล์เป็น `/bin/false`

2.4 ใครเป็นใคร

2.4.1 ฉันคือใคร

เพื่อที่จะตรวจสอบให้แน่ใจว่าคุณกำลังทำงานอยู่ด้วยบัญชีของใคร คุณสามารถถามได้ด้วยคำสั่ง `whoami`

```
thep@anubis:~$ whoami
thep
```

2.4.2 เหลือบแลเพื่อนบ้าน

คำสั่ง `who` จะแสดงรายชื่อผู้ที่กำลังเข้าใช้ระบบทั้งหมด

```
thep@anubis:~$ who
thep     tty2         Jun 26 23:59
root     tty3         Jun 27 00:01
somsak   tty4         Jun 27 00:02
```

คอลัมน์แรกจะเป็นชื่อบัญชีผู้ใช้ที่เข้าใช้ระบบ คอลัมน์ที่สองเป็นชื่อเทอร์มินัลที่ใช้ติดต่อกับระบบ และคอลัมน์ถัดมาเป็นเวลาที่เข้าใช้ระบบ

2.4.3 รู้จักเพื่อนบ้าน

ข้อมูลต่างๆ ของผู้ใช้แต่ละคนที่เก็บไว้ในสารบบ สามารถเรียกดูได้ด้วยคำสั่ง `finger`

```
thep@anubis:~$ finger thep
Login: thep                         Name: Theppitak Karoonboonyanan
Directory: /home/thep               Shell: /bin/bash
On since Thu Jun 26 23:59 (ICT) on tty2 5 minutes 48 seconds idle
     (messages off)
```

```
No mail.
No Plan.
```

คุณสามารถแก้ไขรายละเอียดข้อมูลของตัวคุณเองได้บางส่วนด้วยคำสั่ง chfn

```
thep@anubis:~$ chfn
Password:  (ป้อนรหัสผ่าน)
Changing the user information for thep
Enter the new value, or press ENTER for the default
        Full Name: Theppitak Karoonboonyanan
        Room Number []: 201
        Work Phone []: 1150
        Home Phone []: 1112
thep@anubis:~$ finger thep
Login: thep                         Name: Theppitak Karoonboonyanan
Directory: /home/thep               Shell: /bin/bash
Office: 201, x1150                  Home Phone: x1112
On since Thu Jun 26 23:59 (ICT) on tty2 17 minutes 3 seconds idle
     (messages off)
No mail.
No Plan.
```

สังเกตที่สองบรรทัดสุดท้ายจะตรวจสอบถุงเมล์ว่ามีเมล์ที่ยังไม่ได้อ่านอยู่หรือไม่ ถุงเมล์นี้ ใช้สำหรับเมล์ที่เก็บไว้ที่เครื่องยูนิกซ์ที่ให้บริการอยู่เท่านั้น ถ้าคู่สนทนาของคุณใช้เว็บเมล์หรืออยู่ในระบบอื่น คุณต้องใช้วิธีอื่นในการตรวจสอบ

ส่วนบรรทัดสุดท้าย “No Plan.” นั้น เป็นข้อมูลแผนการของคุณที่อยากเปิดเผยให้เพื่อนบ้านรู้ ซึ่งทำได้โดยเขียนไว้ในไฟล์ ~/.plan และอีกไฟล์หนึ่งที่ finger จะตรวจสอบคือ ~/.project เป็นการให้ข้อมูลว่าคุณกำลังทำโครงงานอะไรอยู่

```
thep@anubis:~$ cat > ~/.plan
To rule the world.
Ctrl-d
thep@anubis:~$ cat > ~/.project
Neuclear Weapon.
Ctrl-d
thep@anubis:~$ finger thep
Login: thep                         Name: Theppitak Karoonboonyanan
Directory: /home/thep               Shell: /bin/bash
Office: 201, x1150                  Home Phone: x1112
On since Thu Jun 26 23:59 (ICT) on tty2 17 minutes 3 seconds idle
     (messages off)
No mail.
```

```
Project:
Neuclear Weapon.
Plan:
To rule the world.
```

นอกจากนี้ ยังสามารถเปลี่ยนเชลล์ได้ด้วยคำสั่ง chsh

```
thep@anubis:~$ chsh
Password:  (ป้อนรหัสผ่าน)
Changing the login shell for thep
Enter the new value, or press return for the default
        Login Shell [/bin/bash]: /bin/tcsh
```

คำสั่ง finger นอกจากจะใช้ดูข้อมูลทั่วไปของผู้ใช้ ยังมีรูปแบบการเรียกแบบไม่ระบุชื่อผู้ใช้ โดยจะแสดงข้อมูลอย่างย่อของผู้ใช้ที่กำลังเข้าใช้ระบบทั้งหมด

```
thep@anubis:~$ finger
Login    Name               Tty      Idle  Login Time
root     root              *tty3    15:54  Jun 27 00:01
thep     Theppitak Karoon  *tty2    15:55  Jun 26 23:59
somsak   Somsak Saetae      tty4     1:58  Jun 27 00:02
```

2.5 ผู้ดูแลระบบ

ดังที่กล่าวไปแล้วว่าผู้ใช้แต่ละคนจะถูกจำกัดขอบเขตของสิทธิในการเข้าถึงไฟล์ต่างๆ แต่จะมีผู้ใช้พิเศษอยู่คนหนึ่งที่มีสิทธิในการเข้าถึงไฟล์ทุกไฟล์ในระบบ เขาคือภารโรงผู้ถือกุญแจห้องทุกห้อง เพื่อจะได้ช่วยแก้ปัญหาให้กับผู้ใช้ และเพื่อปรับแต่งระบบ ติดตั้งโปรแกรมที่ใช้งานร่วมกัน ฯลฯ เรียกว่า *ผู้ดูแลระบบ (system administrator)* หรือบางครั้งเรียกอย่างยกย่องว่า *super user* เนื่องเพราะเอกสิทธิ์ที่ได้มาพร้อมกับหน้าที่บริการนั่นเอง

อย่างไรก็ดี ในระบบทั่วไป ผู้ดูแลระบบมักจะมีบัญชีผู้ใช้ต่างหากของตัวเอง เพื่อใช้ในการทำงานปกติทั่วไป และจะใช้บัญชีผู้ดูแลระบบก็ต่อเมื่อต้องการทำงานดูแลระบบเท่านั้น เพื่อหลีกเลี่ยงการทำลายทรัพย์สินของผู้อื่นโดยไม่ตั้งใจ ข้อแนะนำนี้ ใช้ได้แม้กับระบบที่คุณเป็นเจ้าของเครื่องเพียงผู้เดียวด้วย การลดสิทธิของคุณลงมา ย่อมเป็นการลดโอกาสเสี่ยงที่คุณจะทำระบบพัง ไวรัสบนยูนิกซ์ไม่คุ้มที่จะเขียนก็เนื่องด้วยกฎเหล็กนี้

ด้วยเหตุผลตามนัยประวัติ บัญชีของผู้ดูแลระบบคือ *root* เนื่องจากเป็นผู้ที่มีสิทธิในไฟล์ทั้งหมดตั้งแต่รากของระบบไฟล์เป็นต้นมา ในขณะที่ผู้ใช้ปกติจะมีสิทธิในไฟล์นับจากกิ่งที่เป็น “บ้าน” (home directory) ของตัวเองเป็นต้นมาเท่านั้น

2.6 แปลงร่าง

ในบางครั้ง คุณอาจจำเป็นต้องทำงานบางอย่างในนามของผู้ใช้อื่นชั่วคราว เช่น เมื่อคุณได้รับความไว้วางใจจากผู้ใช้อื่นให้ช่วยแก้ปัญหาบางอย่างให้ และโดยเฉพาะเมื่อคุณเป็น super user ซึ่งโดยปกติคุณจะใช้บัญชี

ต่างหากของคุณทำงาน แต่เมื่อผู้ใช้ในระบบมาร้องขอให้แก้ปัญหาให้ (เช่น ลืมรหัสผ่าน) คุณไม่จำเป็นต้องหาตู้โทรศัพท์ เพียงแต่ใช้คำสั่ง `su` เพื่อเปลี่ยนเป็น super user

```
thep@anubis:~$ su
Password:  (ป้อนรหัสผ่านของ super user)
anubis:/home/thep#
```

สังเกตว่า เครื่องหมายรับคำสั่ง (prompt) ของคุณได้เปลี่ยนจาก `$` เป็น `#` ซึ่งเป็นเครื่องหมายของ super user เรียบร้อยแล้ว

```
anubis:/home/thep# whoami
root
```

คุณสามารถกำหนดชื่อบัญชีผู้ใช้อื่นที่ต้องการเปลี่ยนไปเป็นให้กับคำสั่ง `su` ได้ โดยหากไม่กำหนดจะหมายถึง root จากนั้น `su` จะถามรหัสผ่านของผู้ใช้ที่คุณต้องการเปลี่ยนไปเป็นนั้น

```
thep@anubis:~$ su somsak
Password:  (ป้อนรหัสผ่านของ somsak)
somsak@anubis:/home/thep$
```

การแปลงร่างด้วย `su` นี้ เป็นการสร้างเชลล์ใหม่ทับเชลล์เดิม ดังนั้น เมื่อคุณปฏิบัติภารกิจเสร็จ คุณสามารถกลับไปสู่เชลล์เดิมได้โดยออกจากเชลล์ `su` นี้ ด้วยคำสั่ง `exit` หรือกด Ctrl-d

```
anubis:/home/thep# whoami
root
anubis:/home/thep# exit
thep@anubis:~$ whoami
thep
```

2.7 การปกป้องไฟล์

2.7.1 สิทธิการเข้าถึงไฟล์

ดังที่กล่าวไปแล้วในหัวข้อ 2.1 ว่าไฟล์ทุกไฟล์ในระบบจะมีผู้ใช้และกลุ่มเป็นเจ้าของ โดยสามารถกำหนดสิทธิการเข้าถึงของผู้ใช้ประเภทต่างๆ ได้ คำสั่ง `ls -l` สามารถแสดงรายละเอียดต่างๆ ดังกล่าว

```
thep@anubis:~$ ls -l
total 28
-rwxr-xr-x   1 thep    users      4815 2002-08-23 14:49 a.out
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Download
-rw-r--r--   1 thep    users        77 2002-08-23 14:49 hello.c
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Mail
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Projects
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 tmp
```

ดังได้เกริ่นไปแล้วในหัวข้อ 1.2.3 ว่าคอลัมน์แรกของผลลัพธ์ของคำสั่ง `ls -l` จะแสดงชนิดของไฟล์และสิทธิการเข้าถึงไฟล์ (permission) ซึ่งสิทธิการเข้าถึงไฟล์นี้ มีบทบาทสำคัญในระบบรักษาความปลอดภัยของยูนิกซ์ เนื่องจาก I/O ทุกอย่างที่เรียกผ่านระบบจะมีการตรวจสอบสิทธิก่อนเสมอ โดยเทียบ user id และ group id ของผู้เรียกใช้กับค่าที่เป็นคุณสมบัติของไฟล์ โดยถ้าตรวจสอบสิทธิไม่ผ่าน ระบบจะถือเป็นข้อผิดพลาดและไม่ทำงานให้ โดยข้อความแจ้งข้อผิดพลาดมักจะเป็น "Permission denied." ดังนั้น ไฟล์ข้อมูลของผู้ใช้จึงได้รับการปกป้องเป็นอย่างดีถ้ามีการกำหนดสิทธิอย่างเหมาะสม

อย่างไรก็ดี ระบบจะยอมให้ root ผ่านการตรวจสอบสิทธิทุกอย่างเสมอ ดังนั้น ระบบจะไม่สามารถป้องกัน root จากการแพร่ไวรัสหรือทำลายแฟ้มข้อมูลอย่างไม่ตั้งใจ อาจจะด้วยกับดักที่ซ่อนอยู่ในโปรแกรม หรือด้วยความเลินเล่อก็ตามแต่ ดังนั้น ผู้ดูแลระบบจึงควรเข้าใช้ระบบด้วยบัญชี root เท่าที่จำเป็นจริงๆ เท่านั้น

สิทธิการเข้าถึงไฟล์จะมีสามส่วน ได้แก่ 1. สิทธิของเจ้าของไฟล์เอง 2. สิทธิของผู้ใช้ในกลุ่มเดียวกับกลุ่มเจ้าของไฟล์ และ 3. สิทธิของผู้ใช้นอกกลุ่ม และสิทธิแต่ละส่วนนั้นสามารถกำหนดสิทธิได้สามประการ คือ 1. สิทธิในการอ่านไฟล์ (`r`) 2. สิทธิในการเขียนไฟล์ (`w`) 3. สิทธิในการ execute ไฟล์ (`x`)

เมื่อเขียนเรียงกันจึงเป็นรหัส 9 บิต แทนได้ด้วยเลขฐานแปด 3 หลัก โดยหลักบนสุดแทนสิทธิของเจ้าของไฟล์ ถัดมาแทนสิทธิของผู้ใช้ในกลุ่ม และผู้ใช้นอกกลุ่ม ตามลำดับ

2.7.2 สิทธิการเข้าถึงไฟล์แบบพิเศษ

นอกจากการกำหนดสิทธิเข้าถึงปกติ ยังมีการกำหนดสิทธิในลักษณะพิเศษอีกลักษณะหนึ่งสำหรับไฟล์ที่ยอมให้ execute ได้ คือการยอมให้ execute โปรแกรม *ด้วยสิทธิของเจ้าของไฟล์* มีบางสถานการณ์ที่จำเป็นต้องทำเช่นนั้น เช่น โปรแกรมตั้งรหัสผ่าน `passwd` ซึ่งอนุญาตให้ผู้ใช้เปลี่ยนรหัสผ่านของตัวเองได้ แต่การบันทึกรหัสผ่านที่เข้ารหัสแล้วในไฟล์ของระบบจำเป็นต้องอาศัยสิทธิของ root ดังนั้น ไฟล์ `/usr/bin/passwd` นอกจากจะอนุญาตให้ execute แล้ว ยังอนุญาตให้ execute ด้วยสิทธิของ root ซึ่งเป็นเจ้าของไฟล์ได้อีกด้วย

```
thep@anubis:~$ ls -l /usr/bin/passwd
-rwsr-xr-x  1 root  root   24248 2003-04-27 02:50 /usr/bin/passwd
```

สังเกตว่า สิทธิของเจ้าของไฟล์มีค่าเป็น `s` ไม่ใช่ `x`

การกำหนดสิทธิลักษณะพิเศษนี้ สามารถใช้กับ group ได้ด้วย กล่าวคือ ผู้ execute จะสามารถ execute โดยได้สิทธิเทียบเท่ากับกลุ่มที่เป็นเจ้าของไฟล์ ไฟล์ที่มีการให้สิทธิพิเศษนี้ `ls -l` จะรายงาน permission ของการ execute ของกลุ่มเป็น `s` เช่นกัน

ทั้งการ execute ด้วยสิทธิของเจ้าของไฟล์และด้วยสิทธิของกลุ่มเจ้าของไฟล์นี้ เป็นเรื่องที่ค่อนข้างเสี่ยงที่โปรแกรมหนึ่งๆ จะยอมให้มีได้ เพราะหากป้องกันไม่ดี ก็เท่ากับเป็นช่องโหว่ให้กับการขโมยสิทธิ โปรแกรมลักษณะนี้จึงมิได้มีให้เห็นบ่อยๆ และถ้าจำเป็นต้องใช้ ก็จะต้องตรวจสอบด้วยความระมัดระวัง ยูนิกซ์เปิดโอกาสให้โปรแกรมประเภทนี้รู้ตัวได้ว่ากำลังถูก execute แบบพิเศษ โดยโปรเซสจะทำงานด้วย effective user/group id ของเจ้าของไฟล์เท่านั้น แต่ real user/group id จะยังคงเป็นของผู้เรียกโปรแกรม โปรแกรมจึงมีโอกาสแยกแยะว่าจะให้สิทธิใดๆ ตาม effective หรือ real user/group id

มาถึงตรงนี้ ผู้อ่านอาจกำลังนึกถึงบิตสุดท้าย คือสิทธิของการ execute ของผู้ใช้นอกกลุ่ม ว่ามีการกำหนดสิทธิพิเศษเช่นนี้หรือไม่ เมื่อคิดโดยตรรกะแล้ว การให้สิทธิ execute "ด้วยสิทธิเทียบเท่าผู้ใช้นอกกลุ่ม" ก็เปรียบเสมือนไม่ให้สิทธิอะไรพิเศษเลย และอีกประการหนึ่ง ไฟล์มีเจ้าของและกลุ่มเจ้าของที่จะให้ค่าแก่ effective user/group id เท่านั้น ไม่มีอะไรเกี่ยวข้องกับผู้ใช้นอกกลุ่มเลย

คำตอบก็คือ ยูนิกซ์ใช้บิตพิเศษที่ว่านี้เพื่อจุดประสงค์อื่นที่ไม่ใช่การให้สิทธิโดยตรง บิตนี้เรียกว่า "sticky bit" และไฟล์ที่ถูกเซ็ตบิตนี้ก็เรียกว่า "sticky file" โดยไฟล์โปรแกรมที่ sticky นี้ จะค้างอยู่ในหน่วยความจำของระบบเมื่อจบการทำงานแล้ว ทำให้ไม่จำเป็นต้องเริ่มโหลดโปรแกรมใหม่ในการเรียกใช้ครั้งต่อไป[1]

"sticky bit" นี้ ค่อนข้างพิเศษสักหน่อยเมื่อนำไปใช้กับไดเรกทอรี "sticky directory" ที่ยอมให้ทุกคนเขียน (เช่น `/tmp`) จะยอมให้ผู้ใช้ที่เป็นเจ้าของไฟล์เท่านั้นสามารถลบไฟล์ในไดเรกทอรีได้ (หากไม่เซ็ต sticky bit กับไดเรกทอรีสาธารณะก็หมายความว่า ทุกคนมีสิทธิลบไฟล์ใดๆ ก็ได้ทั้งนั้นในไดเรกทอรีนั้น ซึ่งไม่เหมาะสมนัก)

`ls -l` จะแสดงสิทธิการ execute ของผู้ใช้นอกกลุ่มของ sticky file/directory ด้วย `t` แทน `x` ตัวอย่างที่ดีก็คือ `/tmp`

```
thep@anubis:~$ ls -ld /tmp
drwxrwxrwt    7 root     root         4096 2003-06-28 17:19 /tmp
```

การกำหนดสิทธิแบบพิเศษทั้งหมดนี้ จะอาศัยบิตเพิ่มเติมอีก 3 บิต (เรียกว่า *setuid bit, setgid bit* และ *sticky bit* ตามลำดับ) รวมกับบิตเดิม 9 บิตเป็น 12 บิต โดยแทนด้วยเลขฐานแปด 4 หลัก การกำหนดสิทธิพิเศษนี้จะอยู่หลักบนสุด โดยถ้ากำหนดแบบ 3 หลักปกติก็จะหมายถึงค่าศูนย์ในบิตพิเศษทั้งหมด

2.7.3 การกำหนดสิทธิเข้าถึงไฟล์

permission ของไฟล์ สามารถเปลี่ยนได้ด้วยคำสั่ง `chmod` โดยเข้ารหัส permission เป็นเลขฐานแปด 3 หลัก (ในกรณีปกติ) แทนสิทธิสามส่วน (เจ้าของ, กลุ่ม, ผู้อื่น) โดยตัวเลขฐานแปดแต่ละหลัก เมื่อแปลงเป็นเลขฐานสองสามบิต ก็จะแทนการอนุญาตสามประการ (read, write, execute) สำหรับผู้ใช้แต่ละประเภท ตัวอย่างเช่น

`750` เทียบเท่ากับ `rwxr-x---` คือเจ้าของสามารถอ่าน เขียน execute ได้; ผู้ใช้ในกลุ่มเดียวกันอ่านและ execute ได้ แต่เขียนไม่ได้; ผู้ใช้นอกกลุ่มทำอะไรไม่ได้เลยแม้แต่อ่าน

`644` เทียบเท่ากับ `rw-r--r--` คือเป็นไฟล์ที่ execute ไม่ได้ แต่อ่านได้ทุกคน และมีเพียงเจ้าของเท่านั้นที่เขียนได้

`777` เทียบเท่ากับ `rwxrwxrwx` คือเป็นไฟล์สาธารณะ

ตัวอย่างการตั้งค่า

```
thep@anubis:~$ touch a  # สร้างไฟล์เปล่า
thep@anubis:~$ chmod 750 a
thep@anubis:~$ ls -l a
-rwxr-x---    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod 644 a
thep@anubis:~$ ls -l a
-rw-r--r--    1 thep     users           0 2003-06-28 14:49 a
```

[1] คู่มือของ GNU/Linux ระบุว่า ความสามารถนี้แทบไม่จำเป็นสำหรับ Linux kernel แล้ว เพราะระบบ virtual memory สามารถจัดการให้ได้ แต่อย่างไรก็ดี มันก็เป็นข้อกำหนดหนึ่งของยูนิกซ์ที่ต้องคงไว้

```
thep@anubis:~$ chmod 777 a
thep@anubis:~$ ls -l a
-rwxrwxrwx    1 thep     users           0 2003-06-28 14:49 a
```

อย่างไรก็ดี เพื่อความง่ายต่อการสั่งงาน chmod ยังรับค่าอีกลักษณะหนึ่ง คือเป็นคำสั่งเซ็ตบิตในรูปแบบ [ugoa] [+-] [rwx]

- อักษรตัวแรก หมายถึง ส่วนของสิทธิที่จะตั้งค่า (เลือกได้มากกว่าหนึ่งส่วน)
  - u = user (เจ้าของไฟล์)
  - g = group (ผู้ใช้ในกลุ่ม)
  - o = other (ผู้ใช้นอกกลุ่ม)
  - a = all (เซ็ตค่าทั้งสามส่วนพร้อมกัน)
- อักษรตัวที่สอง หมายถึง วิธีตั้งค่า
  - + = set (อนุญาต)
  - - = clear (ไม่อนุญาต)
- อักษรตัวที่สาม หมายถึง บิตที่จะตั้งค่า (เลือกได้มากกว่าหนึ่งบิต)
  - r = read (สิทธิในการเขียน)
  - w = write (สิทธิในการอ่าน)
  - x = execute (สิทธิในการ execute)

ตัวอย่างเช่น

```
thep@anubis:~$ ls -l a
-rwxrwxrwx    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod g-w a  # ห้ามผู้ใช้ในกลุ่มเขียน
thep@anubis:~$ ls -l a
-rwxr-xrwx    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod o-rwx a  # ห้ามผู้ใช้นอกกลุ่มทำทุกอย่าง
thep@anubis:~$ ls -l a
-rwxr-x---    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod go+rw a  # อนุญาตผู้ใช้ในกลุ่มและนอกกลุ่มอ่านและเขียน
thep@anubis:~$ ls -l a
-rwxrwxrw-    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod a-x a  # ห้ามทุกคน execute
thep@anubis:~$ ls -l a
-rw-rw-rw-    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod +x a  # อนุญาตให้ทุกคน execute
thep@anubis:~$ ls -l a
-rwxrwxrwx    1 thep     users           0 2003-06-28 14:49 a
```

จะเห็นว่า ส่วนแรกที่เลือกส่วนของสิทธินั้น ถ้าละก็จะหมายถึงทุกส่วน

สำหรับการกำหนดสิทธิแบบพิเศษ จะกำหนด setuid, setgid และ sticky bit ได้โดยเพิ่มเลขฐานแปดหลักบนอีกหนึ่งหลัก โดย 4 = setuid, 2 = setgid และ 1 = sticky เช่น

4750 เทียบเท่ากับ rwsr-x---

2750 เทียบเท่ากับ rwxr-s---

1755 เทียบเท่ากับ rwxr-xr-t

เนื่องจากบิตพิเศษทั้งสามใช้ที่แสดงผลร่วมกับค่า x ปกติ คำสั่ง ls -l จึงมีวิธีแยกความแตกต่างเมื่อบิตพิเศษถูกเซ็ตโดยที่ x ไม่ถูกเซ็ต (ซึ่งออกจะเป็นกรณีที่แปลกประหลาด แต่ ls ก็พยายามจัดการ) โดยแสดงค่า s หรือ t ด้วยอักษรตัวใหญ่

```
thep@anubis:~$ chmod 4644 a  # พิกล
thep@anubis:~$ ls -l a
-rwSr--r--    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod +x a
thep@anubis:~$ ls -l a
-rwsr-xr-x    1 thep     users           0 2003-06-28 14:49 a
```

คุณสามารถใช้รูปแบบ [guoa][+-][rwxst] เซ็ตบิตพิเศษดังกล่าวได้เช่นกัน

```
thep@anubis:~$ chmod g+s a
thep@anubis:~$ ls -l a
-rwsr-sr-x    1 thep     users           0 2003-06-28 14:49 a
thep@anubis:~$ chmod g+t a
thep@anubis:~$ ls -l a
-rwsr-sr-t    1 thep     users           0 2003-06-28 14:49 a
```

2.7.4 การโอนเจ้าของไฟล์

ผู้ดูแลระบบสามารถเปลี่ยนเจ้าของไฟล์ได้ด้วยคำสั่ง chown และเปลี่ยนกลุ่มเจ้าของไฟล์ได้ด้วย chgrp

```
anubis:/home/thep# ls -l a
-rwxrwxrwx    1 thep     users           0 2003-06-28 14:49 a
anubis:/home/thep# chown somsak a
anubis:/home/thep# ls -l a
-rwxrwxrwx    1 somsak   users           0 2003-06-28 14:49 a
anubis:/home/thep# chgrp wheel a
-rwxrwxrwx    1 somsak   wheel           0 2003-06-28 14:49 a
```

คำสั่ง chown และ chgrp เป็นคำสั่งที่สงวนไว้สำหรับ root เท่านั้น ผู้ใช้ปกติไม่สามารถสั่งได้

บทที่

3

เกี่ยวกับไฟล์และไดเรกทอรี

เอกลักษณ์ที่สำคัญอย่างหนึ่งในปรัชญาการออกแบบของยูนิกซ์ก็คือ มีการกำหนด interface ของการติดต่อกับอุปกรณ์ภายนอกและองค์ประกอบต่างๆ ของระบบในลักษณะของไฟล์เหมือนกันหมด แป้นพิมพ์ คือไฟล์ เทอร์มินัลคือไฟล์ เมาส์ เครื่องพิมพ์ ช่องสัญญาณเสียง การเชื่อมต่อผ่านระบบเครือข่าย ท่อส่งข้อมูลระหว่างโปรเซส[1] ทุกอย่างคือไฟล์ รายละเอียดปลีกย่อยภายในของแต่ละส่วนถูกซ่อนไว้ภายใต้การดูแลของเคอร์เนลและ device driver ทำให้การเขียนโปรแกรมติดต่อกับอุปกรณ์ต่างๆ บนยูนิกซ์ง่ายลงมาก

ลักษณะพิเศษของระบบไฟล์บนยูนิกซ์อีกอย่างหนึ่งก็คือ การมีโครงสร้างแบบต้นไม้ที่สามารถเชื่อมโยงข้ามกิ่งได้ ทั้งหมดคือสิ่งที่เรากำลังจะกล่าวถึงในบทนี้

3.1 ชนิดของไฟล์บนยูนิกซ์

คำสั่ง `ls -l` สามารถช่วยแสดงชนิดของไฟล์ต่างๆ ได้ ดังได้กล่าวไปแล้วในหัวข้อ 1.2.3 สัญลักษณ์ที่ใช้แทนชนิดต่างๆ ของไฟล์ ได้แก่

- `-` หมายถึงไฟล์ปกติ
- `d` หมายถึงไดเรกทอรี เป็นไฟล์ที่เก็บ “ต้นขั้ว” ของไฟล์ต่างๆ ที่บรรจุอยู่ภายใน
- `l` หมายถึง symbolic link เป็นไฟล์ที่ชี้ไปยังไฟล์อื่นอีกทอดหนึ่ง ดังจะกล่าวต่อไปในหัวข้อ 3.4
- `s` หมายถึง socket เป็นช่องติดต่อสื่อสารกับโปรเซสอื่นผ่านระบบเครือข่าย
- `c` หมายถึง character device มักเป็นอุปกรณ์ภายนอกที่รับส่งข้อมูลเป็นลำดับต่อๆ กัน เช่น โมเด็ม เมาส์ เครื่องพิมพ์ เทอร์มินัล เทป ช่องสัญญาณเสียง ฯลฯ

[1] โปรเซส (process) คือโปรแกรมที่กำลังทำงานอยู่ในขณะหนึ่งๆ

b หมายถึง block device ได้แก่อุปกรณ์ที่รับส่งข้อมูลเป็นบล็อกๆ เช่น ฮาร์ดดิสก์ ฟลอปปีดิสก์ ซีดีรอม แรมดิสก์

p หมายถึง named pipe (FIFO) เป็นท่อส่งข้อมูลระหว่างโปรเซสที่โปรเซสหนึ่งเปิดไว้เพื่อให้โปรเซสอื่นในระบบสามารถรับหรือส่งข้อมูลได้

3.2 ระบบไฟล์บนยูนิกซ์

ระบบไฟล์ทั้งหมดในยูนิกซ์จะเริ่มต้นจากไดเรกทอรีรากเพียงรากเดียว ซึ่งจะบรรจุไฟล์และไดเรกทอรีย่อยลงไปเป็นลำดับชั้นเหมือนต้นไม้ที่แตกกิ่งก้านสาขา ส่วนที่เป็นใบซึ่งไม่แตกแขนงออกไปอีกแล้วก็ได้แก่ไฟล์ข้อมูลหรือไฟล์พิเศษอื่นดังกล่าวข้างต้นที่ไม่ใช่ไดเรกทอรีนั่นเอง

หากคุณคุ้นเคยกับระบบดอสหรือวินโดวส์ที่มีหลายราก แต่ละรากเริ่มจากไดรฟ์ A: B: C: . . . ซึ่งแต่ละไดรฟ์จะหมายถึงอุปกรณ์เป็นตัวๆ หรือเป็นพาร์ทิชันในฮาร์ดดิสก์ หรือเป็น network drive ที่ mount ผ่าน netware ล่ะก็ คุณจะพบแนวคิดที่ต่างออกไปในยูนิกซ์

ยูนิกซ์นั้น จะมองระบบไฟล์ทั้งระบบเป็นระบบเดียว โดยมีการกำหนดหน้าที่ของกิ่งต่างๆ ในระบบไฟล์อย่างชัดเจน เป็นหน้าที่ของผู้ดูแลระบบที่จะจัดสรรอุปกรณ์มาเติมในส่วนต่างๆ ของระบบไฟล์ เช่น อาจจะใช้ฮาร์ดดิสก์หลายลูกประกอบกัน หรืออาจจะแบ่งใช้พาร์ทิชันกับส่วนต่างๆ ของระบบไฟล์ตามความจำเป็นในการสำรองข้อมูล ฯลฯ แต่ผู้ใช้จะมองไม่เห็นรายละเอียดที่อยู่ข้างล่างนั้นเลย โปรแกรมต่างๆ สามารถกำหนดตำแหน่งของไฟล์ต่างๆ อย่างตายตัวได้ตามที่ต้องการได้โดยไม่จำเป็นต้องรู้ว่าอยู่ในฮาร์ดดิสก์ลูกไหน

ไดเรกทอรีที่สำคัญๆ ใน GNU/Linux ที่ควรทราบ ได้แก่

`/boot` เป็นไดเรกทอรีเก็บไฟล์ที่ใช้บูตระบบ เคอร์เนลลินุกซ์จะถูกเก็บไว้ที่นี่

`/bin` (binary) เป็นไดเรกทอรีเก็บโปรแกรมคำสั่งขั้นพื้นฐานของระบบ เป็นโปรแกรมที่จำเป็นเพียงพอสำหรับการใช้งานระบบในกรณีที่ระบบไฟล์ส่วนอื่นมีปัญหา

`/sbin` (system admin binary) เก็บโปรแกรมคำสั่งขั้นพื้นฐานที่ใช้สำหรับผู้ดูแลระบบเท่านั้น เป็นโปรแกรมที่จำเป็นเพียงพอสำหรับการดูแลแก้ไขระบบในกรณีที่ระบบไฟล์ส่วนอื่นมีปัญหา

`/lib` (library) เก็บไลบรารีพื้นฐานที่โปรแกรมใน `/bin` และ `/sbin` ต้องใช้

`/etc` (et cetera) เดิมมีชื่อเช่นนี้เพราะเก็บไฟล์จิปาถะของระบบ แต่การใช้งานในปัจจุบันจะเก็บการตั้งค่าของระบบและของโปรแกรมต่างๆ

`/usr` (user) ชื่อบอกว่าเป็นไดเรกทอรีสำหรับบริการผู้ใช้ ไดเรกทอรีนี้ใช้เก็บส่วนต่างๆ ของระบบที่จำเป็นต้องใช้ในโหมด multi-user ในระบบเก่าๆ จะบรรจุไดเรกทอรีบ้านของผู้ใช้ด้วย แต่ปัจจุบันแยกออกไปที่ `/home` ต่างหาก

`/usr/bin` เก็บโปรแกรมคำสั่งที่ใช้ในระบบเพิ่มเติมจาก `/bin`

`/usr/sbin` เก็บโปรแกรมคำสั่งที่ใช้สำหรับผู้ดูแลระบบเพิ่มเติมจาก `/sbin`

`/usr/lib` เก็บไลบรารีต่างๆ เพิ่มเติมจาก `/lib`

`/usr/include` เก็บ header file ที่ใช้สำหรับคอมไพล์โปรแกรม

`/usr/share` เก็บไฟล์ข้อมูลที่ใช้โดยโปรแกรมต่างๆ

`/usr/local` เป็นไดเรกทอรีที่มีโครงสร้างคล้าย `/usr` แต่ใช้สำหรับการติดตั้งโปรแกรมที่แยกต่างหากจากระบบติดตั้งปกติ (เช่น คอมไพล์เองจากซอร์สโค้ดที่ดาวน์โหลดมาต่างหาก) การแยกมาติดตั้งในที่ต่างหากทำให้สะดวกต่อการจัดการโดยไม่กระทบต่อระบบติดตั้งปกติ

`/usr/X11R6` ใช้สำหรับระบบ X Window ทั้งหมด ไม่ว่าจะเป็นโปรแกรม ไลบรารี ไฟล์ข้อมูล หรือเอกสารประกอบ

`/tmp` ไดเรกทอรีเก็บไฟล์ชั่วคราวสำหรับทุกคนในระบบ โดยปกติจะถูกล้างทิ้งทุกครั้งที่บูตระบบ

`/var` ไดเรกทอรีเก็บข้อมูลของระบบที่จะต้องมีการเปลี่ยนแปลงอยู่ตลอดเวลา เช่น log file ต่างๆ ถุงเมล์ หรือคิวของเครื่องพิมพ์ เป็นต้น

`/dev` เก็บไฟล์ที่แทนอุปกรณ์ทุกอย่าง เป็นที่อยู่ของไฟล์ชนิด character device และ block device โดยเฉพาะ

`/proc` เป็นที่อยู่ของไฟล์เสมือนที่ใช้ติดต่อกับเคอร์เนลลินุกซ์ ไฟล์ต่างๆ ในไฟล์นี้ไม่ใช่ไฟล์จริง แม้จะปรากฏใน `ls -l` ว่าเป็นไฟล์ปกติก็ตาม

`/home` เก็บไดเรกทอรีบ้านของผู้ใช้ทุกคน

`/root` ไดเรกทอรีบ้านของ root เดิมในระบบเก่านั้น บ้านของ root ก็คือ "/" แต่ปัญหาคือทำให้ไฟล์ส่วนตัวของ root ต้องไปกระจุกกองอยู่ที่ไดเรกทอรีราก ต่อมาจึงเริ่มแยกบ้านของ root ออกมาเป็นสัดส่วน

3.3 i-node

หัวข้อนี้จะกล่าวถึงลักษณะพิเศษอีกประการหนึ่งของระบบไฟล์ของยูนิกซ์ คือความสามารถในการ *ตั้งฉายา (alias)* ให้กับไฟล์ได้ กล่าวคือ ไฟล์หนึ่งๆ สามารถตั้งชื่อได้มากกว่าหนึ่งชื่อ การที่จะเข้าใจแนวคิดนี้เราต้องทำความเข้าใจกับคำว่า *i-node* เสียก่อน

ระบบไฟล์โดยทั่วไปจะแยกเนื้อหาของไฟล์กับชื่อไฟล์เป็นคนละส่วนกัน แต่ยูนิกซ์มีความพิเศษเพิ่มขึ้นตรงที่การเชื่อมโยงระหว่างชื่อไปยังเนื้อหาเป็นแบบ many-to-one (แทนที่จะเป็น one-to-one) กล่าวคือในขณะที่การอ้างถึงชื่อหนึ่งๆ จะหมายถึงเนื้อหาไฟล์เพียงหนึ่งเท่านั้น แต่เนื้อหาไฟล์หนึ่งๆ สามารถอ้างมาจากชื่อได้หลายชื่อ

ตัวเนื้อหาของไฟล์ทุกชิ้นจะมีหมายเลข i-node ประจำ ระเบียนชื่อไฟล์ในไดเรกทอรีจะเชื่อมโยงไปยังค่า i-node ที่เป็นเนื้อหา เป็นการเชื่อมโยงในทิศทางเดียว ซึ่งทำให้ i-node หนึ่งๆ สามารถถูกเชื่อมโยงจากชื่อไฟล์ได้หลายชื่อ (ในขณะที่ระเบียนชื่อไฟล์ชื่อหนึ่งจะเก็บค่า i-node เพียงค่าเดียว) ที่ตัว i-node เองก็จะเก็บ reference count ไว้ว่ามีการเชื่อมโยงมายังตัวมันเองเป็นจำนวนเท่าไร การลบไฟล์ในยูนิกซ์จะหมายถึงการ unlink โดยตัดการเชื่อมโยงนี้ และลดค่า reference count ใน i-node ลงหนึ่ง เมื่อใดที่ reference count ถูกลดลงเหลือศูนย์ i-node จึงจะถูกลบออกจากดิสก์จริง

คำสั่ง `ls -l` จะแสดงจำนวนลิงก์ของไฟล์ดังกล่าวในคอลัมน์ที่สอง

```
thep@anubis:~$ ls -l
total 28
-rwxr-xr-x   1 thep    users      4815 2002-08-23 14:49 a.out
```

```
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Download
-rw-r--r--   1 thep    users        77 2002-08-23 14:49 hello.c
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Mail
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Projects
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 tmp
```

จากตัวอย่างข้างต้น จะเห็นว่าไฟล์ปกติ คือ `a.out` และ `hello.c` โดยปกติจะมีจำนวนลิงก์เป็น 1 และเมื่อสั่งลบด้วยคำสั่ง `rm` ก็จะเป็นการลดจำนวนลิงก์นี้ลงหนึ่ง กลายเป็นศูนย์ และ i-node ของไฟล์ก็จะถูกลบจากฮาร์ดดิสก์ทันที การเชื่อมโยงไฟล์ปกติเข้ากับชื่อมากกว่าหนึ่ง ทำได้ด้วยการสร้าง hard link ด้วยคำสั่ง `ln` ซึ่งจะกล่าวในหัวข้อ 3.4

หากคุณสังเกตพบว่า ในตัวอย่างข้างต้น ไฟล์ที่เป็นไดเรกทอรีปลายทางมีจำนวนลิงก์เป็น 2 ก็มีคำอธิบายว่า นอกเหนือจากการเชื่อมโยงจากระเบียนชื่อในไดเรกทอรีที่อยู่เหนือขึ้นไปแล้ว ยังมีลิงก์จากชื่อ "." ในไดเรกทอรีตัวเองด้วย ส่วนไดเรกทอรีที่มีไดเรกทอรีย่อยภายในตัวมันเอง ก็จะถูกเชื่อมโยงจากชื่อ ".." จากไดเรกทอรีย่อยเพิ่มขึ้นมาอีกด้วย เช่น จากตัวอย่างข้างต้น ถ้าขอดูข้อมูลของไดเรกทอรีปัจจุบัน จะพบจำนวนลิงก์เพิ่มจากเดิมอีก 4 ลิงก์ ตามจำนวนไดเรกทอรีย่อยที่มี

```
thep@anubis:~$ ls -ld
drwxr-xr-x   6 thep    users      4096 2002-08-23 14:28 .
```

อนึ่ง หากต้องการดูหมายเลข i-node ของแต่ละไฟล์ คุณก็เพียงระบุตัวเลือก `-i` ให้กับคำสั่ง `ls`

```
thep@anubis:~$ ls -i
 204580 a.out      204583 hello.c   142158 Projects
 142156 Download   142157 Mail      142159 tmp
```

3.4 การลิงก์ไฟล์

ดังที่ได้กล่าวไปแล้ว ว่าไฟล์บนยูนิกซ์สามารถมีชื่อได้มากกว่าหนึ่งชื่อ โดยอาศัยการเชื่อมโยง อย่างไรก็ตาม การเชื่อมโยงไฟล์ไม่จำเป็นต้องใช้ i-node ร่วมเสมอไป วิธีเชื่อมโยงมีอยู่ 2 ลักษณะ คือ

1. **hard link** ได้แก่การเชื่อมโยงโดยฝังในโครงสร้างของระบบไฟล์โดยตรง เป็นการเพิ่มชื่อใหม่ให้กับไฟล์ โดยที่ทั้งไฟล์เก่าและไฟล์ใหม่จะมีฐานะของความเป็นไฟล์เท่าๆ กัน ในทางเทคนิคก็คือ ไฟล์ทั้งสองจะใช้ i-node ร่วมกัน หรือถ้าจะกล่าวให้ละเอียดกว่านั้นก็คือ ไฟล์ทั้งสองจะเป็น entry ที่ต่างกันในไดเรกทอรี แต่ชี้ไปยัง i-node เดียวกัน การสร้าง hard link จะเพิ่ม reference count ใน i-node ที่ถูกเชื่อมโยงนั้น และทุกครั้งที่ลบไฟล์ reference count จะถูกลดลง และ i-node นั้นจะถูกลบออกไปจากดิสก์จริงๆ เมื่อ reference count ลดลงเหลือศูนย์

2. **symbolic link** หรือที่เรียกสั้นๆ ว่า **symlink** จะเป็นการเชื่อมโยงผ่านไฟล์ชนิดพิเศษที่เก็บชื่อของไฟล์ที่เชื่อมโยงไปหาอีกต่อหนึ่ง เมื่อมีการอ้างถึงไฟล์ที่เป็น symlink เพื่อการอ่านหรือเขียน ยูนิกซ์จะอ่านชื่อไฟล์ที่เชื่อมโยงแล้วอ้างไปถึงชื่อนั้นๆ แทน ดังนั้น symlink จึงไม่ได้มีฐานะเป็นไฟล์ที่เก็บข้อมูลด้วยตัวของมันเอง แต่เป็นเพียงตัวชี้ไปหาไฟล์อื่นเท่านั้น

โดยปกติจะไม่สามารถทำ hard link กับไดเรกทอรี และไม่สามารถทำ hard link ข้ามอุปกรณ์ได้ (เช่น ไม่สามารถทำ hard link ข้ามไดเรกทอรีที่อยู่ในฮาร์ดดิสก์คนละพาร์ทิชันได้ และไม่สามารถทำ hard link จากไฟล์ในซีดีรอมลงมาที่ฮาร์ดดิสก์ได้ เป็นต้น) แต่สามารถทำ symlink ได้

คำสั่งสำหรับเชื่อมโยงไฟล์ทั้งแบบ hard link และ symlink ใช้คำสั่งเดียวกัน คือ `ln` (link) โดยหากไม่ระบุตัวเลือกใดเลยจะหมายถึง hard link และถ้าระบุตัวเลือก `-s` จะหมายถึง symlink ส่วนรูปแบบการส่งค่า ก็จะเหมือนคำสั่ง `cp` คือชื่อไฟล์ต้นฉบับมาก่อนชื่อไฟล์ปลายทางที่จะสร้างใหม่

ตัวอย่างเช่น มีไฟล์ `a` ซึ่งมีลักษณะดังนี้

```
thep@anubis:~ $ ls -li
total 8
 125565 -rw-r--r--   1 thep   users    6 2002-09-22 20:46 a
thep@anubis:~$ cat a
hello
```

เมื่อทำ hard link `b` ให้ชี้มายังเนื้อหาเดียวกับ `a`

```
thep@anubis:~$ ln a b
thep@anubis:~$ ls -li
total 8
 125565 -rw-r--r--   2 thep   users    6 2002-09-22 20:46 a
 125565 -rw-r--r--   2 thep   users    6 2002-09-22 20:46 b
```

สังเกตว่า ทั้ง `a` และ `b` มีหมายเลข i-node เดียวกัน คือ 125565 และจำนวนลิงก์ ของ `a` ถูกเพิ่มจาก 1 เป็น 2 และจำนวนลิงก์ของ `b` ก็เท่ากับ 2 เช่นกัน (เพราะมาจาก i-node เดียวกัน)

เมื่อเปลี่ยนแปลงเนื้อหาของไฟล์ `b` ก็จะเห็นว่า `a` ก็เปลี่ยนตามด้วย (เพราะใช้ i-node เดียวกัน)

```
thep@anubis:~$ echo bye > b
thep@anubis:~$ cat a
bye
```

รวมทั้งการเปลี่ยน permission (เพราะ permission เป็นคุณสมบัติประจำ i-node)

```
thep@anubis:~$ chmod a+w b
thep@anubis:~$ ls -li
total 8
 125565 -rw-rw-rw-   2 thep   users    4 2002-09-22 20:50 a
 125565 -rw-rw-rw-   2 thep   users    4 2002-09-22 20:50 b
```

และเมื่อลบ `a` จำนวนลิงก์ของ `b` ก็จะลดลงหนึ่ง

```
thep@anubis:~ $ rm a
thep@anubis:~ $ ls -li
total 4
 125565 -rw-rw-rw-   1 thep   users    4 2002-09-22 20:50 b
```

ในขณะที่ถ้าทำ symlink `c` ให้ชี้มายัง `b`

```
thep@anubis:~$ ln -s b c
thep@anubis:~$ ls -li
total 4
 125565 -rw-rw-rw-   1 thep   users    4 2002-09-22 20:50 b
 127799 lrwxrwxrwx   1 thep   users    1 2002-09-22 21:10 c -> b
```

จะเห็นว่า c เป็นไฟล์ใหม่ที่ใช้คนละ i-node กับ b โดยเป็นไฟล์ชนิด symlink ที่เก็บชื่อ "b" (สังเกตว่า ขนาดของ c เท่ากับ 1 คือเท่ากับความยาวของชื่อ "b" ที่มันเก็บ) และสังเกตว่า จำนวนลิงก์ของ b ไม่ได้เพิ่มเหมือนตอนที่สร้าง hard link

และการเปลี่ยนแปลงใดๆ ใน c ก็จะมีผลกับ b ด้วยเช่นกัน

```
thep@anubis:~$ echo hi > c
thep@anubis:~$ cat b
hi
```

รวมทั้งการเปลี่ยน permission

```
thep@anubis:~ $ chmod go-w c
thep@anubis:~ $ ls -li
total 4
 125565 -rw-r--r--   1 thep   users    4 2002-09-22 20:50 b
 127799 lrwxrwxrwx   1 thep   users    1 2002-09-22 21:10 c -> b
```

แต่สังเกตว่า permission ของ c เองไม่มีความหมาย และจะแสดงเป็น lrwxrwxrwx เสมอ ทั้งนี้เพราะมันไม่ได้เป็นเจ้าของ i-node ของไฟล์ที่อ้างถึงเองนั่นเอง

โดยปกติแล้ว ในชีวิตประจำวันเรามักจะใช้ symlink มากกว่า hard link ด้วยเหตุผลหลายประการ เช่น

- ไม่สามารถทำ hard link กับไดเรกทอรีได้
- ไม่สามารถทำ hard link ข้าม device ได้
- symlink สามารถมองเห็นได้ง่ายผ่านคำสั่ง ls ธรรมดา ทำให้จัดการได้ง่าย

อย่างไรก็ดี ปัญหาที่คุณอาจจะพบเมื่อใช้ symlink ก็คือ ปัญหา broken link เมื่อไฟล์ต้นฉบับถูกลบ

```
thep@anubis:~$ rm b
thep@anubis:~$ ls -l
total 0
 127799 lrwxrwxrwx   1 thep    user    1 2002-09-22 21:10 c -> b
thep@anubis:~$ cat c
cat: c: No such file or directory
```

3.5 ตรวจสอบการใช้เนื้อที่ดิสก์

ว่ากันว่า ฮาร์ดดิสก์เป็นทรัพยากรที่มีเท่าไรก็ไม่เคยพอ ไม่ว่าจะเป็นสมัย 40 MB หรือ 40 GB ฮาร์ดดิสก์ของคุณจะถูกใช้จนเต็มเสมอ แน่นอน คุณต้องการเคลียร์เนื้อที่สักหน่อย คุณสามารถตรวจสอบเนื้อที่ว่างของดิสก์ต่างๆ ที่ถูกใช้ในระบบได้ด้วยคำสั่ง `df` (disk free)

```
thep@anubis:~$ df
Filesystem   1K-blocks      Used Available Use% Mounted on
/dev/hda2      3937252   3548548    188696  95% /
/dev/hda5     19251804  18236840     37008 100% /home
```

คอลัมน์ต่างๆ มีความหมายตามหัวตาราง ในที่นี้มีอุปกรณ์ที่ใช้เก็บข้อมูลสองตัว คือ

1. `/dev/hda2` คือฮาร์ดดิสก์ IDE primary master, primary partition ที่ 2 mount ไว้ที่ `/` หรือรากของระบบไฟล์ มีขนาดทั้งหมด 3,937,252 KB (หรือ 3.8 GB) ใช้ไปแล้ว 3,548,548 KB (หรือ 3.4 GB) เหลือ 188,696 KB (หรือ 185 MB) ใช้เนื้อที่ไปคิดเป็น 95% โดยประมาณ
2. `/dev/hda5` คือฮาร์ดดิสก์ IDE primary master, logical partition ที่ 1 mount ไว้ที่ `/home` มีขนาดทั้งหมด 19,251,804 KB (หรือ 19 GB) ใช้ไปแล้ว 18,236,840 KB (หรือ 18 GB) เหลือ 37,008 KB หรือ 37 MB ใช้เนื้อที่ไปคิดเป็น 100% โดยประมาณ

คุณอาจใช้ตัวเลือก `-h` เพื่อแสดงขนาดต่างๆ ในรูปแบบที่อ่านง่าย แทนการแสดงเป็นจำนวนบล็อก (ตัวเลือก `-h` ในคำสั่งเกี่ยวกับเนื้อที่ดิสก์เป็นความสามารถพิเศษของ GNU ที่เพิ่มเติมจากยูนิกซ์ทั่วไป)

```
thep@anubis:~$ df -h
Filesystem   Size  Used Avail Use% Mounted on
/dev/hda2    3.8G  3.4G  185M  95% /
/dev/hda4     19G   18G   37M 100% /home
```

เมื่อพบว่าฮาร์ดดิสก์ปริ่มๆ จะเต็มเช่นนี้ คุณจะเริ่มลงมือเคลียร์ฮาร์ดดิสก์ แต่ก่อนอื่น คุณคงอยากทราบว่าไดเรกทอรีไหนที่กินที่ คำสั่ง `ls -l` นั้น ช่วยได้แค่นิดหน่อยเท่านั้น โดยในบรรทัดแรกจะแสดงจำนวนบล็อกทั้งหมดที่ไฟล์ต่างๆ ในไดเรกทอรีนั้นใช้ แต่จะไม่ตรวจสอบลงไปถึงไดเรกทอรีย่อยที่อยู่ข้างใน

```
thep@anubis:~$ ls -l
total 28
-rwxr-xr-x   1 thep    users      4815 2002-08-23 14:49 a.out
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Download
-rw-r--r--   1 thep    users        77 2002-08-23 14:49 hello.c
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Mail
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 Projects
drwxr-xr-x   2 thep    users      4096 2002-08-23 14:48 tmp
```

บรรทัด “`total 28`” นั้น แสดงจำนวนบล็อก (ในที่นี้บล็อกมีขนาด 1 KB) ที่ไดเรกทอรีปัจจุบันใช้ทั้งหมด แต่ไม่ได้รวมเนื้อที่ต่างๆ ในไดเรกทอรีย่อย

เครื่องมือตรวจสอบการใช้เนื้อที่ที่มีประสิทธิภาพกว่าคือคำสั่ง `du` (disk usage) ซึ่งจะนับเนื้อที่ทั้งหมดในไดเรกทอรีย่อยด้วย

```
thep@anubis:~$ du
6064    ./Download/nautilus
956     ./Download/metatheme
1700    ./Download/gtk2
8760    ./Download/gdm
928     ./Download/icon
316     ./Download/metacity
18728   ./Download
4       ./Mail
4       ./Projects
4       ./tmp
18756   .
```

หรือคุณสามารถใช้ตัวเลือก -h เพื่อขอดูผลลัพธ์แบบเข้าใจง่าย

```
thep@anubis:~$ du -h
6.0M    ./Download/nautilus
956K    ./Download/metatheme
1.7M    ./Download/gtk2
8.6M    ./Download/gdm
928K    ./Download/icon
316K    ./Download/metacity
19M     ./Download
4.0K    ./Mail
4.0K    ./Projects
4.0K    ./tmp
19M     .
```

ตัวเลือก -s ของ du จะแสดงค่าสรุปของไดเรกทอรีปัจจุบันเท่านั้น

```
thep@anubis:~$ du -sh
19M     .
```

บทที่

4

เครื่องมือประมวลผลไฟล์

นอกจากจะโต้ตอบกับผู้ใช้ทางแป้นพิมพ์และจอภาพแล้ว ยูนิกซ์ยังมีความยืดหยุ่นในการควบคุมเส้นทางของข้อมูลระหว่างโปรแกรมต่างๆ เหมือนสายพานในโรงงาน ทำให้คุณสามารถใช้เครื่องมือย่อยๆ ประกอบกันเพื่อทำงานที่ซับซ้อนได้ตามความต้องการ โปรแกรมคำสั่งต่างๆ ในยูนิกซ์จึงมักไม่ได้ออกแบบมาเป็น "โปรแกรมอเนกประสงค์" ที่ทำได้ทุกอย่างในโปรแกรมเดียว แต่จะเป็นเหมือนตัวต่อเลโก้ชิ้นเล็กๆ ที่ทำงานเฉพาะด้านในด้านหนึ่ง และพยายามทำให้ดีที่สุด ยืดหยุ่นที่สุด เพื่อที่คุณจะสามารถนำมาประกอบกับชิ้นส่วนอื่นเพื่อแก้ปัญหาได้

ในบทนี้ เราจะมาทำความรู้จักกลไกการยักย้ายข้อมูลระหว่างโปรแกรมต่างๆ ที่ทำงานประกอบกัน และทำความรู้จักเครื่องมือบางส่วนของยูนิกซ์ที่คุณสามารถหยิบฉวยมาใช้ได้

4.1 การเปลี่ยนทิศทางข้อมูล

โปรแกรมบนยูนิกซ์ทุกโปรแกรมจะมีช่องทางติดต่อกับอุปกรณ์มาตรฐานสามอย่าง คือ standard input สำหรับรับข้อมูลเข้า, standard output สำหรับแสดงผลลัพธ์ และ standard error สำหรับแสดงข้อผิดพลาด

โดยปกติ ช่องทางทั้งสามจะเชื่อมต่อกับเทอร์มินัลของผู้ใช้ โดย standard input จะรับข้อมูลจากแป้นพิมพ์ standard output และ standard error ก็จะแสดงออกทางจอภาพ แต่ก็สามารถสั่งให้เชื่อมต่อกับไฟล์อื่นที่ไม่ใช่เทอร์มินัลได้เช่นกัน (คงจำได้ว่า ในมุมมองของยูนิกซ์ เทอร์มินัลก็คือไฟล์ชนิดหนึ่ง) ซึ่งถ้าเชื่อมทาง standard input โปรแกรมก็จะอ่านข้อมูลจากไฟล์แทนแป้นพิมพ์ ถ้าเชื่อมทาง standard output หรือ standard error โปรแกรมก็จะแสดงผลลัพธ์หรือข้อผิดพลาดออกทางไฟล์แทนจอภาพตามลำดับ

การเปลี่ยนทิศทางข้อมูล (redirection) สำหรับ standard input และ standard output ในเชลล์ต่างๆ บนยูนิกซ์ จะใช้เครื่องหมาย < > >> ดังนี้

`command < file` เปลี่ยน standard input ให้มารับข้อมูลจาก *file*

`command > file` เปลี่ยน standard output ให้แสดงผลลัพธ์ออกทาง *file* โดยเขียนทับเนื้อหา

เดิมถ้ามีอยู่ ถ้ายังไม่มีก็จะสร้างไฟล์ใหม่

`command >> file` เปลี่ยน standard output ให้แสดงผลลัพธ์ออกทาง *file* โดยเขียนต่อท้ายเนื้อหาเดิมถ้ามีอยู่ ถ้ายังไม่มีก็จะสร้างไฟล์ใหม่

สำหรับการเปลี่ยนทิศทาง standard error นั้น ออกจะต่างกันไปในเชลล์แต่ละชนิด ถ้าคุณใช้เชลล์ในตระกูลเดียวกับ Bourne Shell (`sh`) เช่น Korn Shell (`ksh`) Zsh (`zsh`) หรือ Bourne Again Shell (`bash` คือเชลล์ที่ใช้เป็นเชลล์หลักใน GNU/Linux นั่นเอง) จะใช้รูปแบบ `command 2> file` และ `command 2>> file` คล้ายกับการเปลี่ยนทิศทาง standard output[1] ระวังว่า "2>" และ "2>>" นั้น เขียนติดกันหมด ไม่มีช่องว่างคั่นระหว่าง 2 และ >

ส่วนเชลล์ในตระกูลเดียวกับ C Shell (`csh`) เช่น TENEX C Shell (`tcsh`) จะมีเพียงวิธีเปลี่ยนทิศทาง standard error ให้ออกยังไฟล์เดียวกับ standard output เท่านั้น ไม่สามารถเปลี่ยนทิศทางแยกกันโดยตรงได้[2] รูปแบบที่ใช้คือ `command >& file` และ `command >>& file`

เห็นความสามารถในการเปลี่ยนทิศทาง standard error ออกทางไฟล์เดียวกับ standard output ของ C Shell แล้ว คุณอาจเกิดคำถามขึ้นบ้างว่าจะทำเช่นนั้นใน `bash` ที่คุณกำลังใช้ใน GNU/Linux อย่างไร เพราะคงมีบ่อยครั้งที่คุณอยากจะโยนทุกสิ่งทุกอย่างที่ปรากฏบนจอภาพลงไฟล์ ทั้งผลลัพธ์และข้อผิดพลาด ที่แน่ๆ คือ คุณทำเช่นนี้ไม่ได้

```
thep@anubis:~$ ls -R /root
log  Mail  minicom.log  XF86Config.new
ls: /root/Mail: Permission denied
thep@anubis:~$ ls -R /root > res 2> res
thep@anubis:~$ cat res
ls: /root/Mail: Permission denied
nfig.new
```

เพราะผลลัพธ์จะปนกันเนื่องจากการเขียนทับกันไปมาผ่านไฟล์ที่เปิดแยกกัน จึงไม่เป็นลำดับเหมือนที่เคยปรากฏบนจอ เราจำเป็นต้องทำให้ standard output และ standard error กลายเป็นไฟล์เดียวกันในมุมมองของโปรแกรม ซึ่งรูปแบบการสั่งการเช่นนั้นใน `bash` และเชลล์ในตระกูลเดียวกัน คือ `2>&1`

```
thep@anubis:~$ ls -R /root > res 2>&1
thep@anubis:~$ cat res
/root:
log
Mail
minicom.log
XF86Config.new
ls: /root/Mail: Permission denied
```

ความหมายของ "2>&1" ก็คือ ให้ file descriptor 2 เปลี่ยนทิศทางออกทางไฟล์เดียวกับ file descriptor 1 ซึ่งก็คือ ให้ standard error ออกทางไฟล์เดียวกับ standard output นั่นเอง (รูปแบบ

[1] เลข 2 ใน 2> และ 2>> มาจากค่า file descriptor ที่โปรแกรมทุกโปรแกรมในยูนิกซ์ใช้ติดต่อกับ standard error โดยค่า 0 หมายถึง standard input และค่า 1 หมายถึง standard output แต่จะไม่มีรูปแบบ 0< 1> หรือ 1>> มีเพียง standard error เท่านั้นที่ใช้

[2] อย่างไรก็ดี มีวิธีแก้ขัดอยู่ทางหนึ่งคือ `(command > file1) >& file2`

นี้สามารถใช้กับ file descriptor ค่าอื่นที่มากกว่า 2 ได้อีกด้วย แต่เรื่องนี้อยู่นอกเหนือขอบเขตของยูนิกซ์พื้นฐานเล่มนี้)

รูปแบบสุดท้ายของการเปลี่ยนทิศทางข้อมูลก็คือ การป้อน input ในบรรทัดคำสั่งเลย (เรียกว่า here-document) โดยใช้รูปแบบ

```
command << delimitor
... here-document ...
delimitor
```

โดยที่ *delimitor* คือข้อความที่ใช้จบเอกสาร ข้อความทุกบรรทัดก่อนบรรทัดนี้จะถูกป้อนเข้าทาง standard input ของ *command* โดยตรง

```
thep@anubis:~$ cat << EOF
> At $HOME, I saw this poetry:
>
> A keeper who worked at the zoo,
> Was given a gnu to see to;
>   He cries: ‘That’s the gnu
>   That I knew at Bellevue --
> I knew that I knew that new gnu.’
> EOF
At /home/thep, I saw this poetry:

A keeper who worked at the zoo,
Was given a gnu to see to;
  He cries: ‘That’s the gnu
  That I knew at Bellevue --
I knew that I knew that new gnu.’
thep@anubis:~$
```

สังเกตว่า $HOME ซึ่งเป็นตัวแปรระบบจะถูกแทนค่าเหมือนบรรทัดคำสั่งปกติ ถ้าไม่ต้องการให้เชลล์แทนค่า คุณต้องใส่เครื่องหมายคำพูด (คู่หรือเดี่ยวก็ได้) คร่อม delimitor ที่ตำแหน่งเริ่ม

```
thep@anubis:~$ cat << "EOF"
> At $HOME, I saw this poetry:
> ...
> EOF
At $HOME, I saw this poetry:
...
thep@anubis:~$
```

4.2 เชื่อมคำสั่งด้วย pipe

นอกจากการยักย้ายข้อมูลจากเทอร์มินัลให้เชื่อมกับไฟล์แทนแล้ว สิ่งอำนวยความสะดวกอีกอย่างหนึ่งของยูนิกซ์ก็คือ การต่อ *ท่อ (pipe)* เพื่อส่งข้อมูลจากโปรเซสหนึ่งไปยังอีกโปรเซสหนึ่ง ทำให้คุณสามารถประกอบหลายคำสั่งให้ทำงานรับช่วงกันเป็นทอดๆ ได้

การเชื่อมท่อในคำสั่งเชลล์มีรูปแบบดังนี้

```
command1 | command2 | ...
```

ผลก็คือ standard output ของ *command1* จะถูกเชื่อมตรงเข้ากับ standard input ของ *command2* และหากมีคำสั่งเชื่อมท่อต่อจาก *command2* อีก ก็จะมีการเชื่อมในลักษณะเดียวกันกับคำสั่งถัดๆ ไป

ด้วยกลไกการทำงานเช่นนี้ คำสั่งแต่ละคำสั่งในยูนิกซ์จะทำงานง่ายๆ คือ อ่านข้อมูลเข้าทาง standard input ประมวลผล เขียนผลลัพธ์ออกทาง standard output และแสดงข้อผิดพลาดออกทาง standard error เท่านั้น ไม่ต้องมี interface ซับซ้อน จากนั้น ผู้ใช้จะเป็นผู้ประกอบคำสั่งต่างๆ เข้าด้วยกันเองตามความต้องการ

เราได้เห็นตัวอย่างของการใช้ pipe ไปบ้างแล้วเมื่อพูดถึงการใช้ `more`

```
thep@anubis:~$ ls -l /bin | more
total 2872
-rwxr-xr-x   1 root    root      47464 2002-08-04 03:15 afio
-rwxr-xr-x   1 root    root       2724 2002-01-27 14:05 arch
-rwxr-xr-x   1 root    root     579816 2002-09-12 05:51 bash
-rwxr-xr-x   1 root    root      13864 2002-08-04 16:10 cat
-rwxr-xr-x   1 root    root      16232 2002-09-01 20:39 chgrp
-rwxr-xr-x   1 root    root      16008 2002-09-01 20:39 chmod
-rwxr-xr-x   1 root    root      18120 2002-09-01 20:39 chown
-rwxr-xr-x   1 root    root      42664 2002-09-01 20:39 cp
-rwxr-xr-x   1 root    root      47656 2002-06-23 03:09 cpio
-rwxr-xr-x   1 root    root      82248 2002-09-14 19:32 dash
-rwxr-xr-x   1 root    root      34408 2002-09-01 20:40 date
-rwxr-xr-x   1 root    root      28808 2002-09-01 20:39 dd
-rwxr-xr-x   1 root    root      26184 2002-09-01 20:39 df
-rwxr-xr-x   1 root    root      59496 2002-09-01 20:39 dir
-rwxr-xr-x   1 root    root       4076 2002-01-27 14:05 dmesg
-rwxr-xr-x   1 root    root      11080 2002-09-01 20:40 echo
-rwxr-xr-x   1 root    root      43292 2000-11-27 09:05 ed
--More--
```

และเราจะได้เห็นประโยชน์ของ pipe มากขึ้นอีก เมื่อได้กล่าวถึงคำสั่งตัวกรองต่างๆ

4.3 ตัวกรอง

ความสามารถเรื่องการเปลี่ยนทิศทางข้อมูลหรือ pipe จะไม่มีประโยชน์อะไรมากนัก ถ้าขาดโปรแกรมคำสั่งสำหรับงานต่างๆ มารองรับ เปรียบเสมือนโรงงานที่มีแต่สายพาน แต่ขาดเครื่องจักรนั่นเอง ในหัวข้อนี้เราจะมาทำความรู้จักกับโปรแกรมตัวกรอง (filter) ต่างๆ บางส่วนของที่มีในยูนิกซ์ที่คุณสามารถหยิบมาใช้ได้

4.3.1 head

มีหลายกรณีที่คุณไม่ได้สนใจเนื้อหาของไฟล์ทั้งไฟล์ ยูนิกซ์มีคำสั่งสำหรับตัดเอาเฉพาะส่วนต้นหรือส่วนท้ายของไฟล์มาใช้ ได้แก่คำสั่ง `head` และ `tail` ตามลำดับ

คำสั่ง `head` – แสดงส่วนต้นของไฟล์

รูปแบบ `head` [ตัวเลือก] [*file*]

คำบรรยาย แสดงส่วนต้นของไฟล์ *file* ถ้าไม่ระบุ *file* จะแสดงข้อมูลจาก standard input

ตัวเลือก `-`*n* แสดงผล *n* บรรทัดแรก (หากไม่ระบุ จะแสดง 10 บรรทัดแรก)
`-c` *n* แสดงผล *n* ไบต์แรก

ตัวอย่างเช่น เมื่อคุณจะหาไฟล์ที่ใหญ่ที่สุด 10 ไฟล์แรกในไดเรกทอรีของคุณ:

```
thep@anubis:~$ ls -S /bin | head
bash
tcsh
tar
netstat
dash
mount
dir
ls
vdir
ps
```

4.3.2 tail

นอกจากจะดูเฉพาะต้นไฟล์ คุณสามารถขอดูเฉพาะท้ายไฟล์ก็ได้เช่นกัน ด้วยคำสั่ง `tail`

คำสั่ง `tail` – แสดงส่วนท้ายของไฟล์

รูปแบบ `tail` [ตัวเลือก] [*file*]

คำบรรยาย แสดงส่วนท้ายของไฟล์ *file* ถ้าไม่ระบุ *file* จะแสดงข้อมูลจาก standard input

ตัวเลือก `-n` แสดงผล *n* บรรทัดสุดท้าย (หากไม่ระบุ จะแสดง 10 บรรทัดสุดท้าย)
`-c` *n* แสดงผล *n* ไบต์สุดท้าย
`-f` รอแสดงผลแบบทันทีเมื่อไฟล์ถูกเขียนต่อท้าย

ตัวอย่างเช่น ถ้าจะดู error log รายการล่าสุดจาก log file:

```
thep@anubis:~$ tail -6 /var/log/httpd/error_log
[Fri Sep 20 19:57:15 2002] [error] [client 203.185.2.14] File
does not exist: /var/www/html/d/winnt/system32/cmd.exe
[Fri Sep 20 19:57:19 2002] [error] [client 203.185.2.14] File
does not exist: /var/www/html/scripts/..%5c../winnt/system32/c
md.exe
[Fri Sep 20 19:57:19 2002] [error] [client 203.185.2.14] File
does not exist: /var/www/html/_vti_bin/..%5c../..%5c../..%5c
../winnt/system32/cmd.exe
[Fri Sep 20 19:57:29 2002] [error] [client 203.185.2.14] File
does not exist: /var/www/html/_mem_bin/..%5c../..%5c../..%5c..
/winnt/system32/cmd.exe
[Fri Sep 20 19:57:30 2002] [error] [client 203.185.2.14] File
does not exist: /var/www/html/msadc/..%5c../..%5c../..%5c/..x.
./..x../..x../winnt/system32/cmd.exe
[Fri Sep 20 19:57:34 2002] [error] [client 203.185.2.14] File
does not exist: /var/www/html/scripts/..../winnt/system32/cmd
.exe
```

ตัวเลือก `-f` มีประโยชน์มากในการดู log file ที่มีการเพิ่มข้อมูลต่อท้ายอยู่ตลอดเวลา โดยจะติดตามขนาดของไฟล์และแสดงข้อความที่เพิ่มที่ท้ายไฟล์ทันทีที่ไฟล์เปลี่ยนขนาด ช่วยในการติดตาม error log ของเซิร์ฟเวอร์แบบทันเหตุการณ์ได้เป็นอย่างดี

4.3.3 grep

คำสั่งหนึ่งที่ใช้บ่อยมากในยูนิกซ์คือคำสั่งค้นหาข้อความในไฟล์ ได้แก่คำสั่ง `grep` คุณไม่จำเป็นต้องเปิดพจนานุกรมหาความหมายของมันหรอก เพราะมันคือตัวย่อของ Global Regular Expression Print กล่าวคือ เป็นคำสั่งที่ใช้พิมพ์บรรทัดที่มีข้อความที่เข้ากับรูปแบบ regular expression ที่กำหนด[3]

[3]ที่มาของชื่อ `grep` ก็คือ คำสั่งค้นคำใน line editor ของยูนิกซ์เช่น `ed`, `ex` รวมทั้ง full screen editor ที่สามารถใช้คำสั่งของ `ex` ได้อย่าง `vi` ซึ่งมีรูปแบบเป็น

`g/`*re*`/p`

โดยที่ `g` คือคำสั่ง global สำหรับกระทำทั้งเอกสาร *re* หมายถึง regular expression และ `p` (print) เป็นคำสั่งที่ให้กระทำกับบรรทัดที่ match ด้วยการพิมพ์ออกทางจอภาพ

คำสั่ง `grep` – แสดงบรรทัดที่มีข้อความเข้ากับรูปแบบที่กำหนด

รูปแบบ `grep` [ตัวเลือก] {*pattern*|`-e` *pattern*} [*file* . . .]

คำบรรยาย แสดงบรรทัดที่มีข้อความที่เข้ากับ regular expression *pattern*

ตัวเลือก `-e` *pattern* ใช้เมื่อ *pattern* ขึ้นต้นด้วย '`-`'

`-c` นับบรรทัดแต่ไม่แสดงบรรทัด
`-h` ไม่แสดงชื่อไฟล์
`-H` บังคับแสดงชื่อไฟล์เสมอ
`-i` อักษรตัวใหญ่ตัวเล็กถือว่าไม่ต่างกัน
`-l` แสดงแต่ชื่อไฟล์ที่ match
`-L` แสดงแต่ชื่อไฟล์ที่*ไม่* match
`-n` แสดงหมายเลขบรรทัดด้วย
`-o` แสดงเฉพาะข้อความที่ match แทนการแสดงทั้งบรรทัด
`-v` หาบรรทัดที่ไม่ตรงกับ *pattern* แทน

คำว่า *"regular expression"* หากพูดเป็นภาษาคณิตศาสตร์ก็หมายถึงรูปแบบที่แทนเซ็ตของข้อความเซ็ตหนึ่ง ซึ่งข้อความใดๆ ที่อยู่ในเซ็ตนี้ถือว่า *"เข้ากันได้" (match)* กับ regular expression นั้น รูปแบบการเขียน regular expression จะแบ่งเป็นส่วนกำหนดเซ็ตของอักขระที่จะ match กับโอเปอเรเตอร์

ส่วนของการกำหนดเซ็ตของอักขระที่จะ match ได้แก่

1. *ตัวอักษร* เป็นเซ็ตที่ง่ายที่สุด อักษรทุกตัวและตัวเลขเป็น regular expression ที่ match กับตัวมันเอง โดยถ้าอักขระนั้นๆ เป็นเครื่องหมายพิเศษที่มีความหมายใน regular expression จะต้อง escape ด้วยการเขียน backslash (\) นำหน้า

2. *จุด* '`.`' จะ match กับอักขระใดๆ ก็ได้หนึ่งตัว

3. *วงเล็บเหลี่ยม* กลุ่มของตัวอักษรที่ครอบด้วยวงเล็บเหลี่ยม จะ match กับอักษรใดก็ได้ในชุดนั้นหนึ่งตัว เช่น `[abc]` จะ match กับ `a` หรือ `b` หรือ `c` ก็ได้

การลิสต์อักขระสามารถเขียนเป็นช่วงได้โดยใช้ '`-`' ก็ได้ เช่น เช่น `[a-z]` จะ match กับอักขระใดก็ได้ตั้งแต่ `a` ถึง `z` หนึ่งตัว `[a-zA-Z0-9]` จะ match กับอักษรภาษาอังกฤษทั้งหมดและตัวเลขอารบิกทุกตัว

หากอักขระแรกของเซ็ตเป็น circumflex (`^`) จะเป็นเซ็ตนิเสธ คือจะ match กับอักขระใดๆ ที่ไม่อยู่ในเซ็ตที่กำหนด เช่น `[^0-9]` จะ match กับอักขระใดๆ ที่ไม่ใช่ตัวเลขอารบิก ถ้าคุณต้องการเขียนเซ็ตที่มี `^` ก็สามารถเขียนได้โดยไม่ลิสต์มันเป็นตัวแรก และถ้าต้องการเขียนเซ็ตที่มี `]` ก็เขียนโดยลิสต์มันเป็นตัวแรกของเซ็ต

ส่วนของโอเปอเรเตอร์ ได้แก่

1. *ดอกจัน* '`*`' เมื่อเขียนต่อท้ายเซ็ตจะหมายถึงการซ้ำของเซ็ตนั้นตั้งแต่ 0 ครั้งขึ้นไป เช่น

- `a*` จะ match กับข้อความว่างเปล่า หรือ `a` หรือ `aa` . . .
- `a[0-9]*` จะ match กับ `a` หรือ `a` ที่ตามด้วยเลขอารบิกกี่ตัวก็ได้

2. *circumflex* '`^`' เมื่อเขียนที่ต้น regular expression จะหมายถึงการ match กับข้อความที่ต้นบรรทัด เช่น `^Richard` จะ match กับบรรทัดที่ขึ้นต้นด้วย `Richard`

3. *dollar* '`$`' เมื่อเขียนที่ท้าย regular expression จะหมายถึงการ match กับข้อความที่ท้ายบรรทัด เช่น `Bazaar$` จะ match กับบรรทัดที่ลงท้ายด้วย `Bazaar`

สมมติว่า คุณต้องการหาชื่อไดเรกทอรีย่อยทั้งหมดในไดเรกทอรีปัจจุบัน ก็สามารถทำได้โดยสั่ง "`ls -l`" แล้วเลือกเอาเฉพาะบรรทัดที่ขึ้นต้นด้วย 'd'

```
thep@anubis:~$ ls -l | grep ^d
drwxr-xr-x   8 thep    users      4096 2003-06-30 21:38 Download
drwxr-xr-x   2 thep    users      4096 2003-06-30 14:58 Mail
drwxr-xr-x   2 thep    users      4096 2003-07-01 22:19 Projects
drwxr-xr-x   2 thep    users      4096 2003-07-01 22:19 tmp
```

หรือจะหาบรรทัดที่ลงท้ายด้วย slash (/) ในผลลัพธ์ของ "`ls -F`" ก็ได้

```
thep@anubis:~$ ls -F | grep '/$'
Download/
Mail/
Projects/
tmp/
```

กลับกัน ถ้าคุณต้องการชื่อไฟล์ปกติ ไม่ใช่ไดเรกทอรี คุณก็ใช้ตัวเลือก `-v` กลับผลลัพธ์เสีย

```
thep@anubis:~$ ls -F | grep -v '/$'
a.out*
bye.c
hello.c
```

4.3.4 wc

ไม่ใช่ห้องน้ำ แต่คือ word count เป็นคำสั่งนับจำนวนอักขระ จำนวนคำ และจำนวนบรรทัดในไฟล์

คำสั่ง `wc` – นับจำนวนอักขระ จำนวนคำ และจำนวนบรรทัด

รูปแบบ `wc` [ตัวเลือก] [*file* . . .]

คำบรรยาย แสดงจำนวนอักขระ จำนวนคำ และจำนวนบรรทัดใน *file* หรือถ้าไม่กำหนด *file* ก็จะนับข้อมูลใน standard input

ตัวเลือก `-l` แสดงจำนวนบรรทัด
`-w` แสดงจำนวนคำ
`-c` แสดงจำนวนอักขระ

เช่น เมื่อใช้กับไฟล์ปกติ

```
thep@anubis:~$ cat hello.c
#include <stdio.h>
```

```
int main()
{
  printf("Hello, world.\n");
  return 0;
}

thep@anubis:~$ wc hello.c
      8      10      77 hello.c
thep@anubis:~$ wc -l hello.c
      8 hello.c
thep@anubis:~$ wc *.c
      8      11      85 bye.c
      8      10      77 hello.c
     16      21     162 total
```

คุณสามารถสั่งนับจำนวนไฟล์ในไดเรกทอรีได้

```
thep@anubis:~$ ls | wc -l
     10
```

4.3.5 sort

คุณสามารถเรียงลำดับข้อมูลได้ด้วยคำสั่ง `sort`

คำสั่ง `sort` – เรียงลำดับบรรทัดต่างๆ ในไฟล์ข้อความ

รูปแบบ `sort` [ตัวเลือก] [*file* . . .]

คำบรรยาย ต่อไฟล์ทั้งหมดที่กำหนดและเรียงลำดับบรรทัดออกทาง standard output ถ้าไม่กำหนดไฟล์จะอ่านข้อมูลจาก standard input

ตัวเลือก `-k` *POS1*[`,`*POS2*]
กำหนดคีย์ที่ใช้เรียงลำดับโดยเริ่มจากฟิลด์ที่ *POS1* จนถึงฟิลด์ที่ *POS2* (เริ่มนับจาก 1) เช่น '`-k2,3`' หมายถึงใช้ฟิลด์ที่สองและสามเป็นคีย์ หากไม่กำหนด *POS2* จะหมายถึงใช้ตั้งแต่ฟิลด์ที่ *POS1* จนจบบรรทัดเป็นคีย์

`-t` *delim* กำหนดอักขระที่ใช้แบ่งฟิลด์ในไฟล์ (ค่าปกติหากไม่ระบุจะเป็น space หรือ tab)

`-n` ให้ตีความคีย์เป็นตัวเลขก่อนเรียง

`-r` เรียงลำดับย้อนกลับ

`-u` เลือกเอาบรรทัดที่ซ้ำกันเพียงบรรทัดเดียว

ตัวอย่างเช่น มีไฟล์ข้อมูลต่อไปนี้

```
thep@anubis:~$ cat tel
6402    thep    0-1234-5678
```

```
9876    ott     0-9876-5432
4444    pruet   0-5555-5555
2847    chanop  0-6789-0123
```

เมื่อสั่ง sort โดยไม่ใส่ตัวเลือก ก็หมายถึงการเปรียบเทียบทั้งบรรทัด

```
thep@anubis:~$ sort tel
2847    chanop  0-6789-0123
4444    pruet   0-5555-5555
6402    thep    0-1234-5678
9876    ott     0-9876-5432
```

หากต้องการเรียงชื่อตามลำดับอักษรโดยอาศัยฟิลด์ที่สอง

```
thep@anubis:~$ sort -k2,2 tel
2847    chanop  0-6789-0123
9876    ott     0-9876-5432
4444    pruet   0-5555-5555
6402    thep    0-1234-5678
```

หรือเรียงย้อนลำดับอักษร

```
thep@anubis:~$ sort -k2,2r tel
6402    thep    0-1234-5678
4444    pruet   0-5555-5555
9876    ott     0-9876-5432
2847    chanop  0-6789-0123
```

ลองอีกสักตัวอย่าง สมมติว่า คุณต้องการหาไดเรกทอรีที่กินเนื้อที่มากที่สุดเรียงตามลำดับ คำสั่ง du ให้ผลลัพธ์ว่า

```
thep@anubis:~$ du
6064    ./Download/nautilus
956     ./Download/metatheme
1700    ./Download/gtk2
8760    ./Download/gdm
928     ./Download/icon
316     ./Download/metacity
18728   ./Download
4       ./Mail
4       ./Projects
4       ./tmp
18780   .
```

คุณต้องการเรียงลำดับตามฟิลด์แรกเป็นหลักจากมากไปน้อย จากนั้นใช้ฟิลด์ที่สองเปรียบเทียบตามลำดับอักษรถ้าฟิลด์แรกเท่ากัน

```
thep@anubis:~$ du | sort -k1,1nr -k2,2
18780   .
18728   ./Download
8760    ./Download/gdm
6064    ./Download/nautilus
1700    ./Download/gtk2
956     ./Download/metatheme
928     ./Download/icon
316     ./Download/metacity
4       ./Mail
4       ./Projects
4       ./tmp
```

คำสั่ง `sort` นี้ จะมีผลตามโลแคล `LC_COLLATE` และ `LC_CTYPE` ดังนั้น ถ้าคุณเซ็ตตัวแปรระบบทั้งสอง หรือเซ็ต `LC_ALL` เป็น `th_TH` และระบบของคุณสนับสนุนโลแคลไทย (เช่น ด้วย GNU C library 2.1.1 ขึ้นไป) คุณก็จะสามารถเรียงลำดับคำไทยตามพจนานุกรมฉบับราชบัณฑิตยสถานได้ และหากคุณต้องการการเรียงลำดับแบบ ASCII ธรรมดา คุณก็เซ็ตโลแคลให้เป็น `C` เสีย

4.3.6 uniq

คำสั่ง `uniq` ใช้กำจัดบรรทัดซ้ำในไฟล์ข้อความที่เรียงลำดับแล้ว

คำสั่ง `uniq` – กำจัดบรรทัดซ้ำในไฟล์ข้อความที่เรียงแล้ว

รูปแบบ `uniq` [ตัวเลือก] [*`input`* [*`output`*]]

คำบรรยาย กำจัดบรรทัดซ้ำที่อยู่ติดกันใน `input` (หรือ standard input ถ้าไม่ระบุ) และแสดงผลลัพธ์ออกทาง `output` (หรือ standard output ถ้าไม่ระบุ)

ตัวเลือก `-c` แสดงจำนวนบรรทัดที่ซ้ำไว้ข้างหน้าแต่ละบรรทัดด้วย
`-d` แสดงเฉพาะบรรทัดที่ซ้ำเท่านั้น
`-u` แสดงเฉพาะบรรทัดที่ไม่ซ้ำเท่านั้น

เนื่องจาก `uniq` จะกำจัดบรรทัดซ้ำที่อยู่ติดกันเท่านั้น การใช้ `uniq` กับไฟล์ที่ยังไม่เรียงลำดับจึงยังมีโอกาสมีบรรทัดซ้ำกันหลงเหลือในผลลัพธ์ ดังนั้น ถ้าคุณต้องการไม่ให้เหลือบรรทัดซ้ำเลย คุณควรจะ `sort` เสียก่อนส่งให้ `uniq`

สมมติว่าคุณมีไฟล์ข้อมูลบทความน่าอ่านซึ่งคั่นฟิลด์ด้วย bar (`|`) ดังนี้

```
thep@anubis:~$ cat readings
Eric Raymond|How To Become A Hacker
Eric Raymond|The Cathedral and the Bazaar
```

```
Havoc Pennington|Working on Free Software
Havoc Pennington|Free software maintenance: Adding Features
Advogato|Advogato's Number: Advice for young free software developers
Richard Stallman|The GNU Project
Richard Stallman|The Free Software Definition
Richard Stallman|Why GNU/Linux?
Richard Stallman|We Can Put an End to Word Attachments
Open Source Initiative|The Open Source Definition
Robert Chassel|Making a Living with Free Software
Andy Tai|Free Software: Hackers Comeback
Richard Stallman|The GNU Manifesto
Richard Stallman|Why Software should not have owners
Richard Stallman|Copyleft: Pragmatic Idealism
Eric Raymond|Homesteading the Noosphere
```

คุณตัดเอาเฉพาะชื่อผู้แต่งด้วยคำสั่ง cut (เราจะพูดถึงคำสั่ง cut โดยละเอียดในหัวข้อ 4.3.8 ในที่นี้ขอให้เชื่อว่าคำสั่งนี้สามารถตัดเอาเฉพาะฟิลด์แรกออกมาได้ไปก่อน)

```
thep@anubis:~$ cut -d "|" -f1 readings
Eric Raymond
Eric Raymond
Havoc Pennington
Havoc Pennington
Advogato
Richard Stallman
Richard Stallman
Richard Stallman
Richard Stallman
Open Source Initiative
Robert Chassel
Andy Tai
Richard Stallman
Richard Stallman
Richard Stallman
Eric Raymond
```

หากใช้คำสั่ง uniq เพื่อตัดชื่อซ้ำโดยตรง คุณจะยังได้ชื่อซ้ำหลงเหลืออยู่

```
thep@anubis:~$ cut -d "|" -f1 readings | uniq
Eric Raymond
Havoc Pennington
Advogato
```

```
Richard Stallman
Open Source Initiative
Robert Chassel
Andy Tai
Richard Stallman  (ซ้ำ)
Eric Raymond  (ซ้ำ)
```

คุณจึงควร sort ก่อน uniq เพื่อที่บรรทัดซ้ำจะได้มาอยู่ติดกัน

```
thep@anubis:~$ cut -d "|" -f1 readings | sort | uniq
Advogato
Andy Tai
Eric Raymond
Havoc Pennington
Open Source Initiative
Richard Stallman
Robert Chassel
```

หรืออีกวิธีหนึ่งคือ ใช้ตัวเลือก -u (uniq) กับคำสั่ง sort โดยตรง

```
thep@anubis:~$ cut -d "|" -f1 readings | sort -u
Advogato
Andy Tai
Eric Raymond
Havoc Pennington
Open Source Initiative
Richard Stallman
Robert Chassel
```

ในตัวอย่างข้างต้น คุณอาจเห็นว่าผลของคำสั่ง “sort | uniq” กับ “sort -u” นั้นเหมือนกัน แต่อย่างไรก็ดี ผลจะต่างกันถ้ามีการเลือกใช้คีย์ในคำสั่ง sort โดยถ้าเทียบคีย์ของสองบรรทัดแล้วได้เท่ากัน sort จะถือว่าสองบรรทัดนั้นซ้ำกัน แม้ว่าส่วนอื่นของบรรทัดจะไม่ได้เหมือนกันก็ตาม พิจารณาผลของการสั่งต่อไปนี้

```
thep@anubis:~$ sort -t "|" -k1,1 readings | uniq
Advogato|Advogato's Number: Advice for young free software developers
Andy Tai|Free Software: Hackers Comeback
Eric Raymond|Homesteading the Noosphere
Eric Raymond|How To Become A Hacker
Eric Raymond|The Cathedral and the Bazaar
Havoc Pennington|Free software maintenance: Adding Features
Havoc Pennington|Working on Free Software
Open Source Initiative|The Open Source Definition
```

```
Richard Stallman|Copyleft: Pragmatic Idealism
Richard Stallman|The Free Software Definition
Richard Stallman|The GNU Manifesto
Richard Stallman|The GNU Project
Richard Stallman|We Can Put an End to Word Attachments
Richard Stallman|Why GNU/Linux?
Richard Stallman|Why Software should not have owners
Robert Chassel|Making a Living with Free Software
```

เทียบกับผลของคำสั่ง sort เหมือนเดิมแต่เพิ่มตัวเลือก -u แทนการใช้ uniq

```
thep@anubis:~$ sort -u -t "|" -k1,1 readings
Advogato|Advogato's Number: Advice for young free software developers
Andy Tai|Free Software: Hackers Comeback
Eric Raymond|How To Become A Hacker
Havoc Pennington|Working on Free Software
Open Source Initiative|The Open Source Definition
Richard Stallman|The GNU Project
Robert Chassel|Making a Living with Free Software
```

ซึ่งจะเห็นว่า เป็นการพยายาม unique ฟิลด์ผู้แต่ง โดยเลือกบรรทัดอย่างสุ่ม (ขึ้นอยู่กับการทำงานของอัลกอริทึมที่ใช้เรียงลำดับว่าจะเจอบรรทัดไหนก่อน) มาหนึ่งบรรทัดแทนผู้แต่งแต่ละคน (กรณีอย่างนี้เรียกว่าคีย์ไม่สมบูรณ์ที่ไม่สามารถแยก record ต่างๆ ออกจากกันได้)

4.3.7 tee

เรามาสลับฉากด้วยคำสั่งเบาๆ ที่มีประโยชน์พอควรคำสั่งหนึ่ง ที่ช่วยให้คุณสามารถดักข้อมูลในท่อที่ส่งระหว่างโปรเซสได้ โดยต่อท่อรูปตัวที (T) แทรกกลางเข้าไป คำสั่งที่ทำหน้าที่เช่นนี้คือ tee

> **คำสั่ง** tee – อ่าน standard input และส่งผ่านออกทาง standard output และไฟล์
>
> **รูปแบบ** tee [ตัวเลือก] [*file* ...]
>
> **คำบรรยาย** ก็อปปี้ standard input ไปยัง standard output และไฟล์ที่กำหนด
>
> **ตัวเลือก** -a เขียนข้อมูลต่อท้ายไฟล์แทนการเขียนทับ

สมมติว่าจากไฟล์ readings ในหัวข้อที่แล้ว คุณใช้คำสั่ง cut ตัดฟิลด์ผู้แต่งออกมาแล้วสั่งต่อด้วย "sort -u | wc -l" เพื่อนับจำนวนผู้แต่งทั้งหมด แต่คุณต้องการเก็บรายชื่อผู้แต่งที่ได้ไว้ด้วย คุณสามารถแทรกคำสั่ง tee ระหว่างทางได้

```
thep@anubis:~$ cut -d "|" -f1 readings | sort -u | tee authors \
| wc -l
```

```
      7
thep@anubis:~$ cat authors
Advogato
Andy Tai
Eric Raymond
Havoc Pennington
Open Source Initiative
Richard Stallman
Robert Chassel
```

4.3.8 cut และ paste

คำสั่ง cut และ paste จะทำงานกับไฟล์ในแนวดิ่ง คำสั่ง cut ใช้ตัดเฉพาะฟิลด์ที่ต้องการออกมาจากทุกบรรทัดในไฟล์ ส่วนคำสั่ง paste ใช้ต่อไฟล์ที่แยกฟิลด์นั้นเข้ามาเป็นไฟล์เดียว

คำสั่ง cut – เลือกคอลัมน์ที่ต้องการออกมาจากไฟล์

รูปแบบ cut [ตัวเลือก] [*file* . . .]

คำบรรยาย เลือกคอลัมน์ที่ต้องการจากไฟล์ (กำหนดได้มากกว่าหนึ่งไฟล์ หรือเลือกจาก standard input ถ้าไม่ระบุ) แล้วเขียนออกทาง standard output โดยเลือกได้สามแบบ คือเลือกเป็นไบต์ (ด้วย -b) เลือกเป็นอักขระ (ด้วย -c) หรือเลือกเป็นฟิลด์ (ด้วย -f)

ตัวเลือก -b *list* เลือกไบต์ที่ต้องการ

-c *list* เลือกอักขระที่ต้องการ (อาจจะต่างจาก -b ในภาษาที่อักขระยาวกว่าหนึ่งไบต์)

-f *list* เลือกฟิลด์ที่ต้องการ

การระบุส่วนที่ต้องการด้วย *list* ในตัวเลือก -b, -c และ -f จะระบุด้วยตัวเลขลำดับที่ต้องการ (เริ่มนับจาก 1) หรือเป็นช่วงในรูปแบบ *m*-*n* ก็ได้ และสามารถละ *m* หรือ *n* ส่วนใดส่วนหนึ่งได้ โดย -*n* หมายถึง 1-*n* และ *m*- หมายถึงตั้งแต่ *m* ถึงท้ายบรรทัด

-d *delim* กำหนดตัวคั่นฟิลด์เป็น *delim* (ใช้สำหรับตัวเลือก -f)

-s ตัดบรรทัดที่ไม่มีตัวคั่นฟิลด์ปรากฏทิ้ง (ใช้สำหรับตัวเลือก -f)

เราได้เห็นตัวอย่างการใช้ cut ตัดฟิลด์ไปแล้วในหัวข้อ 4.3.5 และ 4.3.6 ซึ่งน่าจะเป็นกรณีที่พบบ่อยที่สุด

อย่างไรก็ดี หากคุณต้องการหาอักษรนำของชื่อต้นของผู้แต่งทุกคน คุณสามารถใช้ cut ตัดได้เช่นกัน

```
thep@anubis:~$ cut -b1 readings
E
E
H
H
A
```

```
R
R
R
R
O
R
A
R
R
R
E
```

คำสั่ง `paste` – รวมบรรทัดของชุดของไฟล์

รูปแบบ `paste` [ตัวเลือก] [*file* . . .]

คำบรรยาย รวมบรรทัดต่างๆ ของชุดของไฟล์ที่กำหนด (หากไม่กำหนด หรือใช้ - แทนชื่อไฟล์ จะหมายถึง standard input) ตามลำดับขนานกันไปในแต่ละบรรทัด โดยคั่นส่วนของแต่ละไฟล์ด้วยชุด delimiter ที่กำหนด (หรือ TAB หากไม่กำหนดเป็นอย่างอื่น)

ตัวเลือก `-d` *delimiters* ใช้ชุดอักขระใน *delimiters* ตามลำดับเป็นตัวคั่นระหว่างฟิลด์ กล่าวคือ ตัวแรกคั่นระหว่างฟิลด์แรกกับฟิลด์ที่สอง ตัวที่สองคั่นระหว่างฟิลด์ที่สองกับฟิลด์ที่สาม ฯลฯ และย้อนกลับมาใช้ตัวแรกใหม่ถ้า *delimiters* สั้นกว่าจำนวน delimiter ที่ต้องการ

`-s` รวมไฟล์แบบหนึ่งไฟล์ต่อหนึ่งบรรทัดแทนการรวมบรรทัดแบบขนาน

`-f` หยุดเมื่อสุดไฟล์ใดไฟล์หนึ่ง

สมมติว่ามีไฟล์ข้อมูลสองไฟล์ดังนี้

```
thep@anubis:~$ cat id
1
2
thep@anubis:~$ cat name
bruce
richard
eric
```

เมื่อ `paste` ตามปกติ

```
thep@anubis:~$ paste id name
1 TAB bruce
2 TAB richard
 TAB  eric
```

เมื่อ paste โดยกำหนด *delimiters*

```
thep@anubis:~$ paste -d '!@' id name id
1!bruce@1
2!richard@2
!eric@
```

เมื่อ paste แบบ serial

```
thep@anubis:~$ paste -s id name
1    TAB    2
bruce TAB richard TAB eric
```

เมื่อ paste แบบหยุดที่ไฟล์แรกที่สิ้นสุด

```
thep@anubis:~$ paste -f id name
1 TAB bruce
2 TAB richard
```

4.3.9 comm

คำสั่ง comm จะช่วยคุณเปรียบเทียบไฟล์สองไฟล์ที่เรียงลำดับแล้วได้ โดยคัดแยกบรรทัดที่มีเหมือนกันออกจากบรรทัดที่ไม่เหมือนกัน โดยแยกเป็นสามกอง

คำสั่ง comm – เทียบไฟล์สองไฟล์ที่เรียงแล้วบรรทัดต่อบรรทัด

รูปแบบ comm [ตัวเลือก] *file1 file2*

คำบรรยาย เปรียบเทียบไฟล์ที่เรียงแล้วสองไฟล์บรรทัดต่อบรรทัด คัดแยกออกเป็นสามคอลัมน์ บรรทัดที่มีเฉพาะในไฟล์แรกอยู่คอลัมน์แรก บรรทัดที่มีเฉพาะในไฟล์ที่สองอยู่คอลัมน์ที่สอง และบรรทัดที่มีในทั้งสองไฟล์อยู่คอลัมน์ที่สาม คอลัมน์จะถูกแยกกันด้วย TAB หนึ่งตัว ผลลัพธ์แต่ละบรรทัดจะมีข้อมูลเพียงคอลัมน์เดียวเท่านั้น คอลัมน์ที่เหลือจะว่างเปล่า

ตัวเลือก -1 ไม่แสดงคอลัมน์ที่ 1
-2 ไม่แสดงคอลัมน์ที่ 2
-3 ไม่แสดงคอลัมน์ที่ 3

สมมติว่า เรามีไฟล์สองไฟล์ที่เรียงลำดับแล้วดังนี้

```
thep@anubis:~$ cat gnome
hp
miguel
otaylor
timj
```

```
tml
thep@anubis:~$ cat rh
drepper
hp
otaylor
```

เมื่อสั่งเปรียบเทียบไฟล์ทั้งสองด้วย comm

```
thep@anubis:~$ comm gnome rh
        drepper
                hp
miguel
                otaylor
timj
tml
```

จะเห็นว่า คำที่ปรากฏในทั้งสองไฟล์ คือ hp และ otaylor จะถูกแยกไว้คอลัมน์ที่สาม ส่วนที่เหลือถูกแยกไว้ในคอลัมน์แรกสำหรับไฟล์แรก และคอลัมน์ที่สองสำหรับไฟล์ที่สอง หากต้องการแค่บางคอลัมน์ เราก็สั่งระงับคอลัมน์ที่ไม่ต้องการได้

```
thep@anubis:~$ comm -3 gnome rh
        drepper
miguel
timj
tml
thep@anubis:~$ comm -12 gnome rh
hp
otaylor
```

ทีนี้ มาทำอะไรที่เป็นเรื่องเป็นราวดีกว่า สมมติว่าคุณมีไฟล์เก็บ e-mail address ที่แยกออกมาได้จากไฟล์อื่น

```
thep@anubis:~$ cat devlist
hp@rathead.com
miguel@elevenmian.com
iliad@odessey.com
otaylor@rathead.com
timj@geetk.org
rms@gnusnotunix.org
```

คุณต้องการตรวจว่า ใครมีชื่อในไฟล์ gnome บ้าง ก่อนอื่นคุณต้อง cut เอาแต่ชื่อออกมาก่อน แล้ว sort ก่อน comm เอาเฉพาะบรรทัดที่ซ้ำกัน

```
thep@anubis:~$ cut -d '@' -f1 devlist | sort | comm -12 - gnome
hp
miguel
otaylor
timj
```

และใครไม่อยู่บ้าง ก็ทำเช่นเดิม เพียงแต่เลือกเอาเฉพาะบรรทัดที่ไม่มีในไฟล์ gnome

```
thep@anubis:~$ cut -d '@' -f1 devlist | sort | comm -23 - gnome
iliad
rms
```

สังเกตการใช้ '-' แทน standard input ในคำสั่ง comm

4.4 ส่งท้าย

ในบทนี้เราได้ทราบถึงเครื่องมือต่างๆ ของยูนิกซ์ในการประมวลผลข้อมูลเป็นสายงานผ่าน pipe แต่ความสามารถในการประกอบเครื่องมือใหม่ของยูนิกซ์ยังไม่จบลงแค่นี้ เชลล์ของยูนิกซ์อนุญาตให้คุณใช้คำสั่งเหล่านี้เขียนเป็นโปรแกรมใหม่ขึ้นมาเลยทีเดียว ซึ่งหากคุณคุ้นเคยกับ batch file ของดอสหรือวินโดวส์มาแล้วล่ะก็ คุณจะได้พบความยืดหยุ่นอย่างที่ไม่เคยพบมาก่อนในเชลล์ของยูนิกซ์ เพราะมันไม่ใช่ฉบับย่อที่จำกัดจำเขี่ยเหมือนที่คุณคุ้นเคยอีกต่อไป