บทที่ 3
คำสั่งที่ใช้จัดระเบียบแฟ้มข้อมูล
(Commands to manipulate files)

ในบทที่แล้วได้กล่าวถึงการสร้างและการอีดิทแฟ้มข้อมูล โดยใช้ EDT อีดิเตอร์เนื้อหาในบทนี้จะอธิบายเกี่ยวกับการใช้คำสั่ง DCL เพื่อจัดระเบียบแฟ้มข้อมูล เช่น การกำหนดแฟ้มข้อมูล, การสร้างแฟ้มข้อมูล, การลบแฟ้มข้อมูล, การเก็บเฉพาะแฟ้มข้อมูลเวอร์ชันสุดท้าย การสร้างและ list ไดเรคเทอร์รี่, การก๊อปปี้และการตั้งชื่อใหม่ให้แฟ้มข้อมูล

3.1 การกำหนดแฟ้มข้อมูล (Identifying files)

file specification ที่สมบูรณ์ จะประกอบด้วย information ทั้งหมดที่จำเป็นในการหาที่อยู่และกำหนดแฟ้มข้อมูลโดยมีรูปแบบดังนี้

```
node : : device : (directory) filename. type ; version
```

เครื่องหมายกำกับวรรคตอน (colons, brackets, period, semicolon) เป็น syntax ที่ต้องมีเพื่อแยกส่วนต่าง ๆ ของ file specification

3.1.1 โหนด (Nodes)

เมื่อระบบคอมพิวเตอร์ถูกลิงค์เข้าด้วยกันเพื่อฟอร์มเป็น network ชุดหนึ่ง, แต่ละระบบใน network นั้น เรียกว่าโหนด และจะกำหนดภายใน network ด้วยชื่อโหนดที่มีความหมายเดียว (unique) ระบบของเราอาจจะเป็นหรืออาจจะไม่เป็นส่วนหนึ่งของ network ที่ใหญ่กว่า ในการหาคำตอบ พิมพ์ SHOW NETWORK ถ้าระบบของเราเป็นส่วนหนึ่งของ network เราจะเห็นว่า list ของชื่อโหนดปรากฏบนจอภาพ (สำหรับโหนดชื่อ "LOCAL" จะเป็นชื่อโหนดของเรา) ถ้าระบบของเราไม่ได้เป็นส่วนหนึ่งของ network จะมีข้อความบอกว่าไม่มี network ที่ใช้ได้ (available)

ถ้าระบบของเราเป็น network ชุดหนึ่ง เราสามารถที่จะเอาข้อมูลในแฟ้มข้อมูลที่ตั้งอยู่ที่โหนดอีกแห่งหนึ่ง บน network * ได้ โดยการใส่ node specification ไว้ที่ส่วนแรกของ file specification (specification) นี้ เราจะเอาข้อมูลในแฟ้มข้อมูลมาใช้ได้ก็ต่อเมื่อ ผู้ใช้เครื่อง (user) ที่เป็นเจ้าของ อนุญาตให้ผู้ใช้เครื่องคนอื่น, เอาข้อมูลไปใช้ได้ ถ้าเราไม่ได้ระบุโหนดไว้ เครื่องจะ default ว่าแฟ้มข้อมูลนั้นเป็นของเรา หรือ local node

3.1.2 Devices

ส่วนที่ 2 ของ file specification เป็นชื่อ device เป็นการกำหนด physical device ซึ่งจะใช้เก็บ (store) แฟ้มข้อมูล, device name แต่ละชื่อจะแบ่งเป็น 3 ส่วนคือ ส่วนแรกคือ ชนิด device ซึ่งบอกถึง hardware device ตัวอย่างเช่น DB หมายถึง RP06 disk และ MT หมายถึง TE16 magnetic tape ส่วนที่สอง controller designator ซึ่งบอกถึง hardware controller ซึ่ง device ตัวนี้ต่ออยู่ ส่วนที่สาม unit number ซึ่งบอกชัดเจนว่าเป็น device ตัวหนึ่งที่อยู่บน controller นั้น

ตัวอย่าง device names

ชื่อ (name)	**Device**
DBA2	RP06 disk บน controller A, unit 2
MTA0	TE16 magnetic tape บน controller A, unit 0
TTB2	เทอร์มินัล บน controller B, unit 2

ถ้าเราไม่ใส่ device name ใน file specification เครื่องจะ default ให้เป็นแฟ้มข้อมูลบนดิสก์ (disk)

3.1.3 ไดเรคทอรีและซับไดเรคทอรี (Directories and subdirectories)

เนื่องจากในดิสก์ตัวหนึ่งสามารถเก็บแฟ้มข้อมูลต่าง ๆ ซึ่งเป็นของผู้ใช้เครื่องหลายคน ผู้ใช้เครื่องแต่ละคนของดิสก์ตัวนั้นจะมีไดเรคทอรี (directory) หนึ่งชุด ซึ่งจัดระเบียบ (catalogs) แฟ้มข้อมูลทั้งหมดที่เป็นของเขา ใน device ตัวนั้น

ใน default disk ถ้าเราไม่ได้ระบุไดเรคทอรี อีกชุดหนึ่ง หรือถ้าเราไม่ได้ระบุไดเรคทอรีอะไรเลย เครื่องจะ default โดยสมมติว่าแฟ้มข้อมูลที่เราอ้างถึงนั้น ได้จัดระเบียบใน default

---

* *ให้ดูในหนังสือคู่มือ DECnet-VAX สำหรับคำอธิบายการเอาข้อมูลในแฟ้มข้อมูลข้าม network*

directory ของเรา, เราสามารถค้นหาว่า default disk และไดเรคทอรีปัจจุบันของเราคืออะไร โดยใช้คำสั่ง SHOW DEFAULT ดังนี้

$ show default RET

DBA2 : [COBSEC2]

การตอบรับคำสั่ง SHOW DEFAULT นี้แสดงว่า default device ของเราคือ DBA2 default directory คือ [COBSEC2]

เราสามารถเอาข้อมูลจากแฟ้มข้อมูลในไดเรคทอรีอื่นได้ (รวมทั้งไดเรคทอรีซึ่งจัดระเบียบแฟ้มข้อมูลเป็นของผู้ใช้เครื่องคนอื่น) โดยระบุชื่อไดเรคทอรีใน file specification

ตัวอย่าง ให้แสดงมูลค่า (content) ของแฟ้มข้อมูลชื่อ CONTENTS.DAT ที่เป็นของผู้ใช้เครื่องที่มีไดเรคทอรีเป็น [JONES] ให้ปรากฏบนเทอร์มินัลของเรา โดยใช้คำสั่ง TYPE ดังนี้

$ type [jones] contents.dat RET

หมายเหตุ file specification จะไม่รวมชื่อ device สำหรับคำสั่งนี้ เพื่อให้ execute ได้สำเร็จ, ไดเรคทอรี [JONES] ต้องมีใน default disk device ของเรา ทั้งนี้เพราะว่าเครื่องจะจัด default ให้เมื่อเราไม่บอกชื่อ device, ถ้าไดเรคทอรี JONES ของผู้ใช้เครื่องนั้นอยู่บน disk DBB2 เราต้องใช้คำสั่งดังนี้

$ type fnn2 : [jones] contents.dat RET

ทั้ง 2 ตัวอย่างข้างต้นนี้ เราสมมติว่าผู้ใช้เครื่องชื่อ JONES อนุญาตให้ผู้ใช้เครื่องคนอื่นเอาแฟ้มข้อมูลในไดเรคทอรีของตนไปใช้ได้ เราสามารถอนุญาตบางส่วนหรือจำกัดการเอาแฟ้มข้อมูลของเราไปใช้โดยผู้อื่น โดยใช้คำสั่ง SET PROTECTION (ให้ดูเพิ่มเติม ในหนังสือคู่มือ VAX/VMS Command Language User's Guide for information about directory and file protection and for a description of the SET PROTECTION command)

แฟ้มข้อมูลสามารถจัดระเบียบในซับไดเรคทอรี (subdirectories) ได้เช่นกัน ซับไดเรคทอรีหนึ่ง, หมายถึงไฟล์ (จัดระเบียบในไดเรคทอรีที่สูงกว่า) ซึ่งประกอบด้วยแฟ้มข้อมูลเพิ่มเติมอีก ชื่อของซับไดเรคทอรี ได้จากการต่อชื่อของมันกับชื่อของไดเรคทอรีซึ่งมีมันอยู่

ตัวอย่าง

$ type [jones.datafiles] memo.sum RET

คำสั่ง TYPE ข้างต้นนี้ให้แสดงแฟ้มข้อมูลชื่อ MEMO.SUM ซึ่งจัดระเบียบไว้ในซับไดเรคทอรี [JONES.DATAFILES], ซับไดเรคทอรีไฟล์ชื่อ DATAFILES.DIR ก็ถูกจัดระเบียบไว้ในไดเรคทอรี [JONES] (เรื่องไดเรคทอรีได้บรรยายไว้อย่างละเอียดในหัวข้อ 3.8)

3.1.4 ชื่อ ชนิด และเวอร์ชันของแฟ้มข้อมูล

ข้อดีของการ default โหมด, ดิสก์ และ ไดเรคทอรี ทำให้เราสามารถกำหนดแฟ้มข้อมูลที่มีความหมายเดียวได้ โดยบอกเฉพาะชื่อแฟ้มข้อมูลและชนิดของแฟ้มข้อมูลในรูปแบบดังนี้

filename.type

ชื่อแฟ้มข้อมูลประกอบด้วยตัวอักขระ 1 ตัวขึ้นไปไม่เกิน 9 ตัวเลือกจากตัวอักษร (letter) A ถึง Z และตัวเลข (degit) 0 ถึง 9 เมื่อเราสร้างแฟ้มข้อมูลเราจะใช้ชื่ออะไรก็ได้ที่สื่อความหมายได้ดี

ชนิดของแฟ้มข้อมูล ประกอบด้วยตัวอักขระไม่เกิน 3 ตัว หรือไม่มีเลยก็ได้ และอยู่หลังเครื่องหมาย . (period) ชื่อนี้เลือกจากตัวอักษร A ถึง Z หรือตัวเลข 0 ถึง 9

อย่างไรก็ตาม ชนิดของแฟ้มข้อมูลโดยปกติจะให้รายละเอียดของชนิดของข้อมูลในแฟ้มข้อมูลนั้นและเครื่องคอมพิวเตอร์ก็ยอมรับการ default ชนิดของแฟ้มข้อมูลหลายตัว ที่ใช้ในวัตถุประสงค์เฉพาะอย่าง ตัวอย่างเช่น ภาษาระดับสูง แต่ละภาษาจะมีการ default ชนิดของแฟ้มข้อมูล สำหรับ source program (ดูหัวข้อ 5.1) นอกจากนี้แล้วยังมีการ default ชนิดของแฟ้มข้อมูลอื่น ๆ อีก ดังนี้

ชนิดของแฟ้มข้อมูล	ใช้สำหรับ
DAT	Data file
EDT	Start-up command file for EDT editor
EXE	Executable program image file
JOU	Journal file used by the EDT editor
LIS	Output listing file
MAI	Mail message file
OBJ	Object module file output from a compiler or assembler

นอกจากชื่อแฟ้มข้อมูลและชนิดของแฟ้มข้อมูลแล้ว ทุก ๆ แฟ้มข้อมูลต้องมีเลขเวอร์ชัน (version number) ซึ่งเครื่องจะกำหนดให้กับแต่ละแฟ้มข้อมูล เมื่อมีการสร้าง หรือ การปรับปรุง (revised) เมื่อเราเริ่มต้นสร้างแฟ้มข้อมูล เครื่องจะกำหนดเลขเวอร์ชันให้เป็น 1 ต่อจากนั้น ถ้าเราอีดิทแฟ้มข้อมูลหรือสร้างเวอร์ชันอื่นเพิ่มเติม, เลขเวอร์ชันจะเพิ่มขึ้นทีละ 1 อัตโนมัติ

เราไม่จำเป็นต้องบอกเลขเวอร์ชันใน file specification ก็ได้เพราะเครื่องจะ default มูลค่าสำหรับเลขเวอร์ชันให้ ขณะที่มันทำกับ devices, ไดเรคทอรี และชนิดของแฟ้มข้อมูล, การ

default เลขเวอร์ชันกำหนดไว้ดังนี้

a) สำหรับ Input file เครื่องจะใช้เลขเวอร์ชันสูงสุดที่มีอยู่ของแฟ้มข้อมูลนั้น

b) สำหรับ Output file เครื่องจะบวก 1 กับเลขเวอร์ชันสูงสุดที่มีอยู่

เมื่อเราบอกเลขเวอร์ชัน ใน file specification ให้ใส่เลขเวอร์ชันหลังเครื่องหมาย semi-colon (;) หรือ period (.)

3.1.5 Wild card characters

Wild card character แต่ละตัวเป็นสัญลักษณ์ที่เราสามารถเอามาใช้ได้ในคำสั่ง DCL หลายคำสั่ง เพื่อบอกให้คำสั่งนั้นกระทำกับแฟ้มข้อมูลหลายชุดในครั้งเดียวกันมากกว่าบอกให้ทำทีละแฟ้มข้อมูลแยกต่างหากจากกัน, wild card characters มีอยู่ 2 ตัวคือ asterisk (*) และ percent sign (%) ซึ่งจะไปใช้ใน specification ของไดเรคทอรี, ชื่อแฟ้มข้อมูลและชนิดของแฟ้มข้อมูล เครื่องหมาย asterisk ยังใช้สำหรับบอกเลขเวอร์ชันด้วย ตัวอย่างเช่น

เราสามารถบอกทุกเวอร์ชันของแฟ้มข้อมูลนั้น โดยการใส่ asterisk แทนที่เลขเวอร์ชันใน file specification

ตัวอย่าง

*$ print testfile.dat ; **

คำสั่งนี้ให้พิมพ์แฟ้มข้อมูลชื่อ TESTFILE.DAT ทุกเวอร์ชัน โดยไม่ต้องบอกเลขเวอร์ชัน แต่ละตัวแยกต่างหากจากกัน

ตัวอย่าง

$ print testfile.dat

คำสั่งนี้ไม่มี wild card character คำสั่ง PRINT ด้วยการ default เครื่องจะใช้เวอร์ชันสูงสุดของแฟ้มข้อมูล TESTFILE.DAT นั่นคือพิมพ์แฟ้มข้อมูลชื่อ TESTFILE.DAT เฉพาะชุดที่มีเลขเวอร์ชันสูงสุด

ตัวอย่าง

*$ print * . dat ; **

คำสั่งนี้พิมพ์ทุกเวอร์ชันของแฟ้มข้อมูลทุกชุดในไดเรคทอรีปัจจุบันที่มีชนิดของแฟ้มข้อมูลเป็น DAT

ตัวอย่าง

*$ print test . * ; **

คำสั่งนี้พิมพ์ทุกเวอร์ชันของแฟ้มข้อมูลทุกชุดในไดเรคทอรี ที่มีชื่อแฟ้มข้อมูลว่า TEST

สำหรับเครื่องหมาย percent มีไว้ให้เราระบุแฟ้มข้อมูลทุกชุด ซึ่งมีตัวอักขระเป็นอะไรก็ได้ในตำแหน่งซึ่งถูกเครื่องหมาย % แทนที่ใน file specification

ตัวอย่าง

$ print chap % . txt

คำสั่งนี้พิมพ์เวอร์ชันสุดท้ายของแฟ้มข้อมูลหลายชุด ซึ่งมีชนิดของแฟ้มข้อมูลเป็น TXT และชื่อแฟ้มข้อมูลขึ้นต้นด้วย CHAP และจบด้วยชุดของเลขต่าง ๆ กัน เช่น CHAP1. TXT, CHAP2. TXT, และ CHAP3. TXT เป็นต้น

แต่อย่างไรก็ตาม ตัวอย่างนี้ เครื่องหมาย % ระบุตัวอักขระเพียงตัวเดียว นั่นคือ คำสั่งพิมพ์ข้างต้นนี้ จะไม่มีผลต่อแฟ้มข้อมูลชื่อ CHAP.TXT หรือ CHAPIX.TXT

3.2 การสร้างแฟ้มข้อมูล (Creating files)

ในบทที่สองได้อธิบายการสร้างแฟ้มข้อมูลโดยใช้คำสั่ง EDIT เพื่อเรียกอีดิเตอร์ นอกจากนี้แล้วเรายังสามารถใช้คำสั่ง CREATE เพื่อสร้างแฟ้มข้อมูลใหม่ โดยให้ชื่อแฟ้มข้อมูลเป็นพารามิเตอร์ แล้วเราใส่ text เข้าไปได้ทันที เมื่อจบการใส่ text ให้กด CTRL/Z

ตัวอย่าง

$ create myfile.dat

This is the only line. CTRL/Z

3.3 การลบแฟ้มข้อมูลทิ้ง (Deleting files)

มีอยู่บ่อย ๆ ขณะที่เราพัฒนาและปรับปรุงโปรแกรม แล้วจบด้วยเวอร์ชันหลายชุด เนื่องจากแฟ้มข้อมูลเหล่านี้จะต้องใช้เนื้อที่เก็บในดิสก์, ดังนั้นแฟ้มข้อมูลเวอร์ชันที่เราไม่ต้องการใช้อีกแล้วเราลบทิ้งได้

คำสั่ง CREATE ไม่เหมือนคำสั่ง EDIT ตรงที่เอาไปใช้ modify แฟ้มข้อมูลที่มีอยู่ไม่ได้

คำสั่ง DELETE ลบแฟ้มข้อมูลที่ไม่ต้องการออก เมื่อเราใช้คำสั่ง DELETE เราต้องบอกชื่อแฟ้มข้อมูล ชนิดของแฟ้มข้อมูล และเลขเวอร์ชัน (การระบุเลขเวอร์ชัน เป็นการป้องกันไม่ให้ลบแฟ้มข้อมูลเวอร์ชันอื่นทิ้งโดยไม่ตั้งใจ)

อย่างไรก็ตาม แต่ละส่วนของแฟ้มข้อมูลเหล่านี้ อาจจะระบุโดยใช้ wild card character ได้ นอกจากนี้เราสามารถใส่ file specification มากกว่า 1 ชุด ในบรรทัดของคำสั่งนั้น โดยใช้เครื่องหมาย comma (,) คั่นระหว่าง file specification แต่ละตัวได้

ตัวอย่าง	ผลลัพธ์
$ delete average.obj ; 1	ลบแฟ้มข้อมูลชื่อ AVERAGE.OBJ ; 1 ทิ้ง
$ delete * .lis ; *	ลบแฟ้มข้อมูลทุกชุดที่มีชนิดของแฟ้มข้อมูลเป็น LIS (นั่นคือ, คำสั่งนี้ลบทุกเวอร์ชันของ program listing ทุกชุด)
$ delete a.dat ; 1, a.dat ; 2	ลบ 2 เวอร์ชันแรกของ data file เดียวกัน

3.4 การเก็บเฉพาะแฟ้มข้อมูลเวอร์ชันสุดท้าย (Purging files)

บางครั้งเราอาจต้องการล้าง (clean) ไดเรคทอรีของเราเพื่อขจัดเวอร์ชันแรก ๆ ทุกชุดของแฟ้มข้อมูลต่าง ๆ ถ้าแฟ้มข้อมูล 1 ชุดของเรามีหลายเวอร์ชัน การใส่ชื่อทั้งหมด ในคำสั่ง DELETE เป็นสิ่งที่เราอาจจะไม่ชอบ

คำสั่ง PURGE ลบแฟ้มข้อมูลทุกเวอร์ชันทิ้ง ยกเว้นเวอร์ชันสุดท้ายของแฟ้มข้อมูลนั้น คำสั่ง PURGE จึงไม่จำเป็นต้องบอกเลขเวอร์ชัน

ตัวอย่าง

$ purge average.cob

คำสั่งนี้ลบแฟ้มข้อมูลทั้งหมดที่ชื่อ AVERAGE.COB ทิ้ง ยกเว้นแฟ้มข้อมูลชื่อ AVERAGE.COB ที่มีเลขเวอร์ชันสูงสุด

/KEEP เป็น qualifier ของคำสั่ง PURGE เราอาจจะบอกว่าต้องการเก็บ (keep) แฟ้มข้อมูลนั้นมากกว่า 1 เวอร์ชันได้

ตัวอย่าง

$ purge/keep = 2 test.dat

คำสั่งนี้ลบแฟ้มข้อมูลชื่อ TEST.DAT ทั้งหมดทิ้ง ยกเว้น 2 เวอร์ชันสุดท้าย

3.5 การแสดงแฟ้มข้อมูลทางเทอร์มินัล (Display files at your terminal)

คำสั่ง TYPE จะแสดงแฟ้มข้อมูลออกมาทางเทอร์มินัล

ตัวอย่าง

```
$ type test.dat
This is the first line of a file
created with the EDT editor.
```

ขณะที่กำลังแสดงแฟ้มข้อมูลอยู่นั้น เราอาจจะขัดจังหวะ (interrupt) output โดยการใช้ CTRL คีย์อย่างใดอย่างหนึ่งดังนี้

CTRL/S หยุดการแสดงผลของแฟ้มข้อมูลและการประมวลผลของคำสั่งทางเทอร์มินัล

ถ้าต้องการแสดงผลทางเทอร์มินัลต่อไปอีก, กด CTRL/Q จอภาพก็จะแสดงผลต่อจากที่ขัดจังหวะไว้

CTRL/C หรือ CTRL/Y ขัดจังหวะการประมวลผลคำสั่ง, เครื่องจะ prompt ให้เราใส่คำสั่งอื่นต่อ

3.6 การพิมพ์แฟ้มข้อมูล (Printing files)

เมื่อเราใช้คำสั่ง PRINT ให้พิมพ์ก๊อปปี้ของแฟ้มข้อมูลชุดใดชุดหนึ่ง เครื่องจะไม่สามารถพิมพ์แฟ้มข้อมูลนั้นให้ได้ทันที เนื่องจากว่า อาจจะมี line printer เพียงตัวเดียว หรือสองตัวสำหรับผู้ใช้เครื่องทุกคน เครื่องจะใส่ชื่อแฟ้มข้อมูลที่เราต้องการพิมพ์ให้อยู่ในรูปคิว (queue) แล้วพิมพ์แฟ้มข้อมูลที่เข้ามาก่อนเป็นอันดับแรก

แฟ้มข้อมูลที่พิมพ์ออกทางกระดาษ (print file) จะตามหลังกระดาษแผ่นแรก (header page flag) ซึ่งระบุชื่อแฟ้มข้อมูล ซึ่งทำให้เราทราบว่าแฟ้มข้อมูลนั้นเป็นของเรา

ตัวอย่าง

```
$ print db2 : [jones] average.lis
  job 210 entered on queue SYS$PRINT
```

คำสั่งข้างต้นนี้กระดาษแผ่นแรก (header page) จะพิมพ์ชื่อผู้ใช้เครื่อง, ชื่อแฟ้มข้อมูล, และเลขเวอร์ชันของแฟ้มข้อมูลนั้น

ส่วนข้อความที่เครื่องตอบรับนั้น บอกเลขประจำ job ที่กำหนดให้กับงานที่จะพิมพ์

คำสั่ง PRINT มี qualifiers เช่นเดียวกันที่ทำให้เราสามารถกำหนดจำนวนก๊อปปี้ ของ

แฟ้มข้อมูลที่จะพิมพ์ได้ รายละเอียดต่าง ๆ เพิ่มเติมเกี่ยวกับ qualifiers เหล่านี้ ให้ดูในหนังสือคู่มือ VAX/VMS Command Language User's Guide

ตัวอย่าง

$ PRINT/QUE = LPB0 *: myfile.lis, data_text.dat/*NOFLAG, *output.lst/*NOFLAG

3.7 การเรียกรายชื่อแฟ้มข้อมูลทั้งหมดในไดเรคทอรี (Listing files in directory)

คำสั่ง DIRECTORY จะให้รายการ (list) ชื่อของแฟ้มข้อมูลทั้งหมดในไดเรคทอรีที่กำหนด, ถ้าเราพิมพ์คำสั่ง DIRECTORY โดยไม่มีพารามิเตอร์หรือ qualifier คำสั่งนี้จะ default ให้แสดงรายการชื่อแฟ้มข้อมูลในไดเรคทอรี ของเราบนเทอร์มินัล

ตัวอย่าง

$ *directory* RET

① — DIRECTORY DBA2 : [COBSEC2.WIWAT]

② — AVERAGE.EXE ; 2 AVERAGE.EXE ; 1 AVERAGE.COB ; 2 AVERAGE.COB ; 1
AVERAGE.OBJ ; 2 AVERAGE.OBJ ; 1

③ — total of 6 files

ข้อความทั้งหมดข้างต้นนี้เป็นตัวอย่าง output ของคำสั่ง DIRECTORY

หมายเลข 1 เป็นชื่อดิสก์ และชื่อไดเรคทอรี

หมายเลข 2 เป็นชื่อแฟ้มข้อมูล, ชนิดของแฟ้มข้อมูล และเลขเวอร์ชันของแฟ้มข้อมูลแต่ละชุดในไดเรคทอรีนั้น

หมายเลข 3 บอกจำนวนแฟ้มข้อมูลทั้งหมดในไดเรคทอรี

เมื่อเราใช้คำสั่ง DIRECTORY เราสามารถให้ file specification ตั้งแต่ 1 ชุดขึ้นไป เพื่อให้ได้ listing เฉพาะแฟ้มข้อมูลชุดที่ต้องการเท่านั้น ตัวอย่างเช่น ต้องการทราบว่าแฟ้มข้อมูลชื่อ AVERAGE.COB ที่มีอยู่ในขณะนั้นมีกี่เวอร์ชัน ให้ใช้คำสั่ง DIRECTORY ดังนี้

$ directory average.cob RET

DIRECTORY DBA1 : [COBSEC2.WIWAT]

AVERAGE.COB ; 2 AVERAGE.COB ; 1

total of 2 files

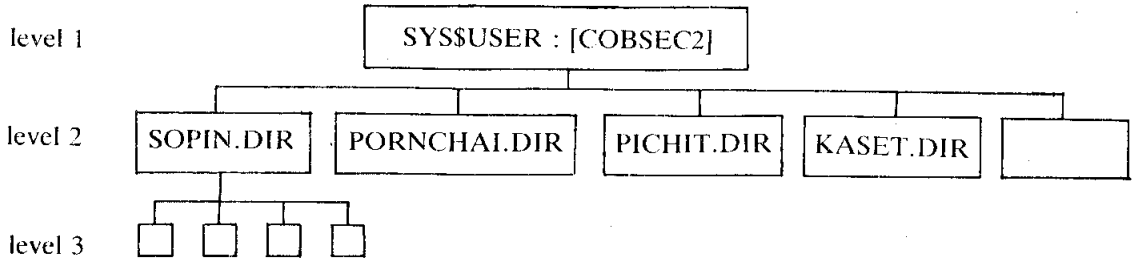
3.8 การสร้างซับไดเรคทอรี

โดยปกติ ผู้จัดการระบบจะให้ไดเรคทอรี 1 ชุดกับผู้ใช้เครื่องแต่ละคน เพื่อเก็บรักษาแฟ้มข้อมูล ถ้าเราเป็นผู้ใช้เครื่อง และงานของเรามีหลายชุด (applications) เราจะพบว่า จะสะดวกมากในการสร้างซับไดเรคทอรีขึ้นมาอีกจำนวนหนึ่ง แล้วจัดระเบียบให้มันอยู่ในไดเรคทอรีหลัก เราสามารถสร้างซับไดเรคทอรีในไดเรคทอรีใดไดเรคทอรีหนึ่ง ซึ่งเรามีไว้สำหรับสร้างแฟ้มข้อมูลได้ โดยใช้คำสั่ง CREATE/DIRECTORY

ตัวอย่าง

$ *create/directory* [COBSEC2.*sopin*] RET

คำสั่งนี้สร้างซับไดเรคเตอร์ไฟล์ชื่อ SOPIN.DIR ในไดเรคทอรีชื่อ [COBSEC2] เราสามารถเรียกชื่อซับไดเรคทอรี [COBSEC2.SOPIN] ไปใช้ได้ในคำสั่งต่าง ๆ หรือในโปรแกรม



3.9 การเปลี่ยนการ default ในไดเรคทอรี

การสร้างซับไดเรคทอรีขึ้นมาอีก 1 ชุด ใน default ไดเรคทอรีของเราให้ใช้คำสั่ง SET DEFAULT ตัวอย่างเช่น เราสามารถสร้างแฟ้มข้อมูลใหม่ 1 ชุดในซับไดเรคทอรี [COBSEC2.SOPIN] โดยการเปลี่ยน default ไดเรคทอรีของเรา แล้วสร้างแฟ้มข้อมูลด้วยคำสั่ง EDIT

ตัวอย่าง

$ *set default* [COBSEC2.SOPIN] RET

$ *edit newfile.txt* RET

Input file does not exist

[EOB]

*

แฟ้มข้อมูลใหม่ที่เกิดขึ้นจะถูก cataloged ในซับไดเรคทอรีชื่อ [COBSEC2.SOPIN] (เราสามารถทำเช่นนี้ได้โดยการกำหนดซับไดเรคทอรี ให้เป็นส่วนหนึ่งของ file specification เมื่อเราใช้คำสั่ง EDIT)

เราสามารถใช้คำสั่ง SET DEFAULT เปลี่ยนการ default ดิสค์ของเราได้ด้วย

ตัวอย่าง

$ *set default dbb2 :* RET

หลังจากเราใช้คำสั่งนี้ เครื่องจะใช้ดิสค์ DBB2 เป็น default ดิสค์สำหรับแฟ้มข้อมูลทั้งหมดที่เราเอามาใช้ หรือที่เราสร้างขึ้นมา

เราสามารถเปลี่ยนการ default ของดิสค์ และไดเรคทอรีได้บ่อยเท่าที่ทำได้สะดวก การเปลี่ยนแปลงที่เกิดจากการใช้คำสั่ง SET DEFAULT จะยังคงมีผลอยู่จนกว่าเราจะใช้คำสั่ง SET DEFAULT อื่นหรือ log out

3.10 การก๊อปปี้แฟ้มข้อมูล (Copying files)

คำสั่ง COPY ทำให้เกิดแฟ้มข้อมูลหลายก๊อปปี้ (copies) เราใช้เพื่อทำให้เกิดก๊อปปี้ของแฟ้มข้อมูลในไดเรคทอรีของเราหรือเพื่อก๊อปปี้แฟ้มข้อมูลจากไดเรคทอรีหนึ่งไปยังอีกไดเรคทอรีหนึ่ง, หรือเพื่อก๊อปปี้แฟ้มข้อมูลจาก devices อื่น ๆ หรือเพื่อสร้างแฟ้มข้อมูลที่ประกอบด้วย input ไฟล์มากกว่า 1 ไฟล์ขึ้นไป

เมื่อเราใช้คำสั่ง COPY, สิ่งแรกที่ต้องบอกคือชื่อของ input ไฟล์ที่เราต้องการก๊อปปี้ และชื่อของ output ไฟล์

ตัวอย่าง

$ COPY *payroll.tst payroll.old* RET

คำสั่งข้างต้นนี้ ก๊อปปี้มูลค่า (contents) ของแฟ้มข้อมูลชื่อ PAYROLL.TST ไปให้แฟ้มข้อมูลชื่อ PAYROLL.OLD

ถ้าแฟ้มข้อมูลชื่อ PAYROLL.OLD มีอยู่แล้ว, เวอร์ชันใหม่ของแฟ้มข้อมูลนี้จะถูกสร้างขึ้นและมีเลขเวอร์ชันเพิ่มขึ้นอีกหนึ่ง

เราสามารถก๊อปปี้แฟ้มข้อมูลชุดหนึ่งจากไดเรคทอรี [COBSEC2] ไปยังซับไดเรคทอรี [COBSEC2.SOPIN] และให้ชื่อใหม่เป็น OLDFILE.DAT ได้ดังนี้

$ COPY *newfile.dat* [COBSEC2.SOPIN] *oldfile.dat* RET

เมื่อเราก๊อปปี้แฟ้มข้อมูลจาก devices ที่ไม่ใช่ default ดิสค์ของเรา เราต้องบอกชื่อ device ในคำสั่ง COPY

ตัวอย่าง

$ *COPY payroll.tst dmal :* RET

คำสั่งข้างต้นนี้ก็อปปี้แฟ้มข้อมูลจาก default ไดเรคทอรีของเรา ไปไว้ใน RK06 ดิสค์

หมายเหตุ output file specification ไม่จำเป็นต้องมีชื่อแฟ้มข้อมูล, ชนิดของแฟ้มข้อมูล, คำสั่ง COPY ที่ใช้ในไดเรคทอรีเดียวกัน, ชื่อแฟ้มข้อมูล, และชนิดของแฟ้มข้อมูล จะเหมือนกับใน input ไฟล์ ด้วยการ default

ก่อนที่เราจะก็อปปี้แฟ้มข้อมูลใด ๆ จาก devices ที่ไม่ใช่ดิสค์ เราต้องเตรียมอุปกรณ์เหล่านี้ให้พร้อมดังนี้

a) mounting the volume ด้วยคำสั่ง MOUNT

b) ให้แน่ใจว่า volume นั้นมีไดเรคทอรี สำหรับการ catalog แฟ้มข้อมูล ถ้าไม่มีไดเรคทอรีอยู่เลย ใช้คำสั่ง CREATE สร้างไดเรคทอรีขึ้นมาหนึ่งชุด

3.11 การเปลี่ยนชื่อแฟ้มข้อมูล (Renaming files)

คำสั่ง RENAME ใช้เปลี่ยน identification ของแฟ้มข้อมูลตั้งแต่ 1 ชุดขึ้นไป

ตัวอย่าง

$ *rename payroll.dat test.old* RET

คำสั่งข้างต้นนี้เปลี่ยนชื่อแฟ้มข้อมูลชื่อ PAYROLL.DAT เวอร์ชันล่าสุดให้เป็นชื่อ TEST.OLD

และเราสามารถใช้คำสั่ง RENAME ย้ายแฟ้มข้อมูลจากไดเรคทอรีหนึ่งไปยังไดเรคทอรีอีกชุดหนึ่งได้

ตัวอย่าง

$ *rename* [COBSEC2] *test.old* [COBSEC2.SOPIN]

คำสั่งข้างต้นนี้ย้าย test.old จากไดเรคทอรี [COBSEC2] ไปไว้ในซับไดเรคทอรี [COBSEC2.SOPIN]

เราอาจจะใช้ wild card character ถ้าเราต้องการเปลี่ยนแฟ้มข้อมูลจำนวนหนึ่งซึ่งมีชื่อแฟ้มข้อมูลหรือชนิดของแฟ้มข้อมูลอย่างใดอย่างหนึ่ง

ตัวอย่าง

$ *rename payroll.* * : * [COBSEC2.SOPIN] *.* : * RET

คำสั่งข้างต้นนี้เปลี่ยนชื่อไดเรคทอรีของทุกเวอร์ชันของแฟ้มข้อมูลทุกชุดที่มีชื่อแฟ้มข้อมูลว่า PAYROLL แล้วแฟ้มข้อมูลเหล่านี้จะถูกนำไป cataloged ในซับไดเรคทอรี [COBSEC 2.SOPIN]

แบบฝึกหัด

1. จงบอกความหมายของคำศัพท์ต่อไปนี้

Information message	Default
File	DCL command
CTRL/Z	CTRL/Y
Subdirectory	Directory
HELP key	*HELP
*EXIT	*QUIT

2. จงบอกความหมายของคำสั่งต่อไปนี้

```
$ PRINT testfile.dat;*
$ PRINT test.* ; *
$ PRINT *.dat;*
$ PRINT chap %.text
$ DELETE * .lis;*
* DELETE ALL "1986"
$ PURGE average.cob
$ RENAME [COBSEC2] test.old [COBSEC2.SOPIN]
* SUBSTITUTE/DEC/Digital/WHOLE
$ DELETE a.dat; 1, a.dat; 2
$ PURGE/KEEP = 2 test.dat
$ RENAME payroll. * ; * [COBSEC3.testfile] *.*;*
* MOVE REST TO 27
* COPY 4 THRU 10 TO.
```

3. จงบอกความแตกต่างระหว่างคำสั่ง EDIT และคำสั่ง CREATE
   จงเขียนคำสั่งสร้างซับไดเรคทอรีไฟล์ TESTFILE.DIR ในไดเรคทอรี [COBSEC 2]

4. จงบอกความหมายของกลุ่มคำสั่ง 2 ชุดข้างล่างนี้

```
$ DEFINE myfile [chuck] personnel.rec
$ TYPE myfile
```

และ ชุดต่อไปนี้

```
$ DEFINE test SCIENCE4: [malcolm. testfiles]
$ RUN test: memo
$ PRINT test: memo.lis
```