บทที่ 4
ความเข้าใจเกี่ยวกับโครงสร้างของไดเรคทอรี
(Understanding directory structure)

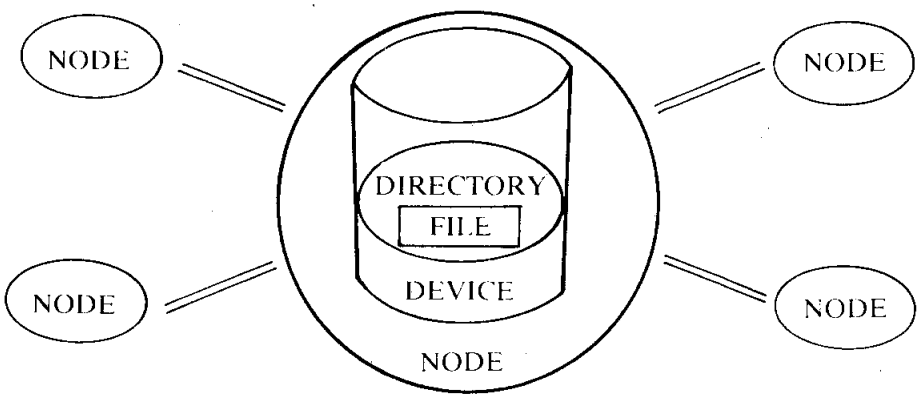
ในบทนี้จะอธิบายส่วนต่าง ๆ ของ file specification ที่สมบูรณ์

4.1 ส่วนต่าง ๆ ของ file specification ที่สมบูรณ์

file specification ที่สมบูรณ์แต่ละชุดจะประกอบด้วย information ทั้งหมดที่ระบบเครื่องคอมพิวเตอร์จำเป็นต้องทราบในการใช้หาตำแหน่งที่อยู่และกำหนดแฟ้มข้อมูลแต่ละชุด โดยมีรูปแบบดังนี้

node : : device : [directory] filename.type; version

เราต้องใส่เครื่องหมายกำกับวรรคตอน (colons, brackets, period, semicolon) แยกส่วนต่าง ๆ ของ file specification



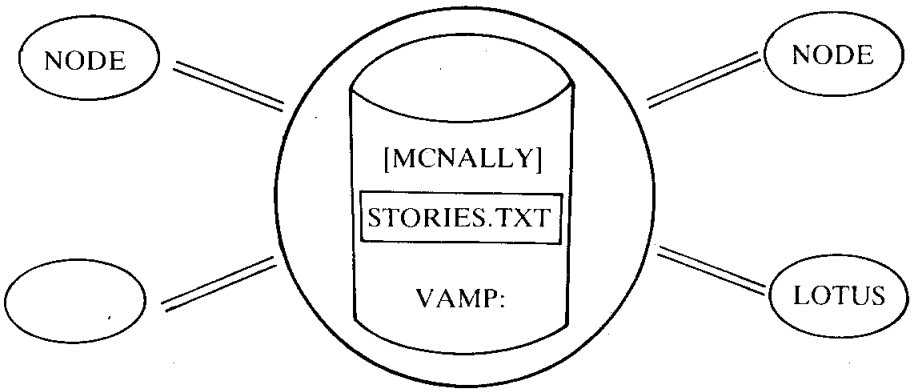
รูปแสดง ความสัมพันธ์ระหว่างส่วนต่าง ๆ ของ file specification ที่สมบูรณ์โปรดสังเกตว่ารายชื่อแฟ้มข้อมูลจะปรากฏในไดเรคทอรี, ไดเรคทอรีอยู่ใน device และ device จะอยู่ที่โหนดใดโหนดหนึ่ง

เมื่อระบบคอมพิวเตอร์ถูกลิงค์เข้าด้วยกัน เราเรียกว่า network แต่ละระบบใน network นั้นเรียกว่า โหนด (node) และถูกกำหนดภายใน network ด้วยชื่อที่มีความหมายเดียว (a unique node name) ระบบคอมพิวเตอร์ของเราอาจจะเป็นส่วนหนึ่งของ network ที่ใหญ่กว่าหรือไม่เป็นก็ได้ สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับ networks และ โหนดให้ดูในหนังสือ Guide to networking on VAX/VMS

```
$ copy RET
- FROM: DRACUL : : VAMP : [MCNALLY] STORIES.TXT;7
         ↑       ↑       ↑          ↑        ↑       ↑
       node    device  directory  filename  file type  version
```

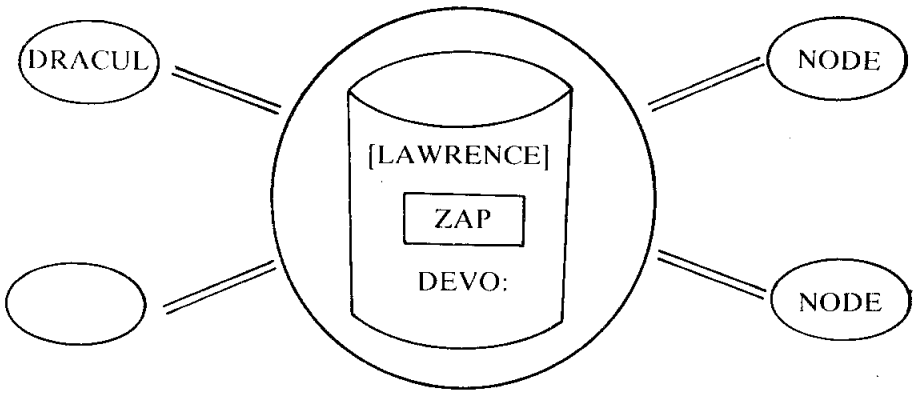
รูปแสดง full file specification ชุดหนึ่ง มี STORIES.TXT เป็นแฟ้มข้อมูลอยู่ในไดเรค-ทอรี [MCNALLY] บน device ชื่อ VAMP ติดตั้งที่โหนด DRACUL และสามารถวาดรูปแสดงความสัมพันธ์ระหว่างส่วนต่าง ๆ ของ

DRACUL : : VAMP : [MCNALLY] STORIES.TXT ดังนี้



ตัวอย่าง ต้องการเข้าถึงข้อมูลในแฟ้มข้อมูลชื่อ ZAP.LIS ซึ่งปรากฏชื่อในไดเรคทอรี [LAWRENCE] เก็บบน device ชื่อ DEVO บนโหนด LOTUS ให้ใช้คำสั่งข้างล่างนี้

```
$ type lotus : : devo : [lawrence] zap.lis
```



ถ้าเราไม่ระบุชื่อโหนด เครื่องจะถือว่าแฟ้มข้อมูลนั้นเป็นของเรา หรือ local node โดยการ default

4.1.2 Devices

ส่วนที่สองของ file specification คือ device name ซึ่งบอกให้ทราบ physical device ซึ่งเก็บแฟ้มข้อมูลนั้น ตัวอย่างของ devices เช่น เทปและดิสค์ ชื่อ device แต่ละชื่อแบ่งเป็น 3 ส่วนคือ

ส่วนแรก ชนิดของ device เป็นการระบุ hardware device ตัวอย่างเช่น DB หมายถึง ดิสค์ RP06, MT หมายถึง เทปแม่เหล็ก TE 16

ส่วนที่สองเรียกว่า a controller designator หมายถึง hardware controller ซึ่งต่อกับ device ตัวนั้น

และส่วนที่สาม เป็น unit number หมายถึง device ตัวหนึ่งที่อยู่บน controller ตัวใดตัวหนึ่ง

ตัวอย่าง device names

LPA0
LPB0 } หมายถึง line printer บน controller A, B, C ตามลำดับ, unit 0
LPC0

DBA2 หมายถึง ดิสค์ RP06 บน controller A, unit 2

MTA0 หมายถึง เทปแม่เหล็ก TE16 บน controller A, unit 0

TTB2 หมายถึง เทอร์มินัล บน controller B, unit 2

โปรดสังเกตว่า ชื่อ device ของเราเป็นคำที่ประกอบด้วยรหัส 4 หลัก คำเหล่านี้เป็น logical name (ให้ดูคำอธิบายในบทที่ 6 เรื่อง logical names)

ตัวอย่าง

$ sh def

SYS$USER : [RUO2]

หมายถึงเราอยู่ใน ไดเรคทอรี [RUO2] ซึ่งเป็น level ที่ 1 และ SYS$USER เป็น logical device name ต่อไปใช้คำสั่ง

$ *dir*

เครื่องจะ list รายชื่อแฟ้มข้อมูลทั้งหมดที่อยู่ในไดเรคทอรี

ตัวอย่าง

$ show default

BOOK1 : [HARVEY]

คำสั่งข้างต้นนี้ logical device name คือ BOOK1:

ตัวอย่าง

$ type tape1 : trees.dat

คำสั่งข้างต้นนี้ คือการเข้าถึงแฟ้มข้อมูลชื่อ TREES.DAT ซึ่งเก็บในเทปแม่เหล็กมี label เป็น TAPE1 และถ้าแฟ้มข้อมูล TREES.DAT อยู่ในโหนดของเราเอง เช่นตัวอย่างนี้เราจึงไม่ระบุชื่อโหนด

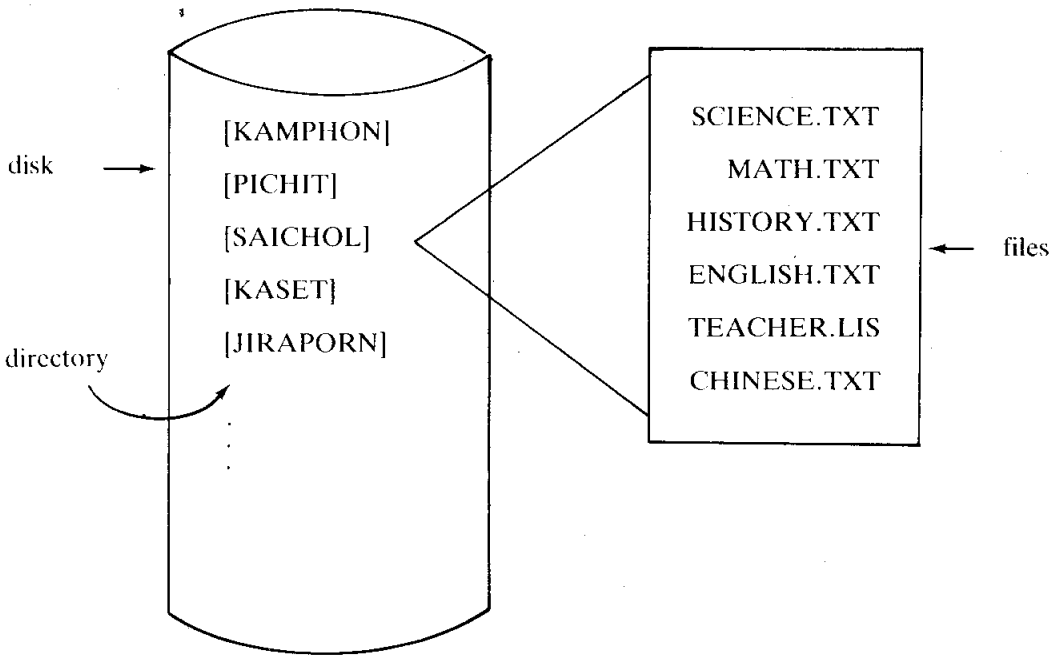
ถ้าเราไม่ใส่ชื่อ device ใน file specification เครื่องจะ default มูลค่าให้เป็นดิสค์, ดิสค์ตัวนี้จะเป็น default disk ของเรา

ถ้าต้องการให้เครื่อง list รายชื่อ devices ทั้งหมด ในระบบของเรา ให้ใช้คำสั่ง SHOW DEVICE

4.1.3 ไดเรคทอรี (Directories)

เนื่องจากดิสค์หนึ่งตัวสามารถเก็บแฟ้มข้อมูลต่าง ๆ ที่เป็นของผู้ใช้เครื่องหลาย ๆ คน ผู้ใช้เครื่องแต่ละคนของดิสค์ที่กำหนดให้ จะมีไดเรคทอรีของตนเองหนึ่งชุด สำหรับจัดระเบียบ (catalog) แฟ้มข้อมูลทั้งหมดที่เป็นของตน ไดเรคทอรี เป็นแฟ้มข้อมูลซึ่งใช้จัดระเบียบแฟ้มข้อมูลอื่น ๆ ไดเรคทอรีไฟล์แต่ละชุด จะประกอบด้วยชื่อ และตำแหน่งของแฟ้มข้อมูลต่าง ๆ ซึ่งอยู่ในรูปแบบที่เครื่องเข้าใจได้

ตัวอย่าง แฟ้มข้อมูลทั้งหลายที่อยู่ในไดเรคทอรี [COBSEC3]



รูปแสดง ดิสค์ตัวหนึ่งคือ TEACHERS1 : ประกอบด้วย รายชื่อของไดเรคทอรี 5 ชุด (ของผู้ใช้เครื่อง ชื่อ KAMPHON, PICHIT, SAICHOL, KASET และ JIRAPORN) และมีรายชื่อแฟ้มข้อมูลทั้งหมดที่อยู่ในไดเรคทอรี [SAICHOL]

ถ้าต้องการเข้าถึงแฟ้มข้อมูลชื่อ SCIENCE.TXT. ให้ใช้คำสั่งดังนี้

```
$ type teacher1 : [saichol] science.txt
```

ตัวอย่าง

```
$ show default
TEACHERS1 : [COBSEC3]
```

หมายถึง default device ของผู้ใช้เครื่อง คือ TEACHERS1 :

และ default directory คือ [COBSEC3]

ถ้าต้องการเข้าถึงแฟ้มข้อมูล ในไดเรคทอรีอื่น (หมายถึง ไดเรคทอรี ซึ่งจัดระเบียบแฟ้มข้อมูลของผู้ใช้เครื่องคนอื่น) ต้องกำหนดชื่อไดเรคทอรีใน file specification

ตัวอย่าง

$ type [jones] contents.dat RET

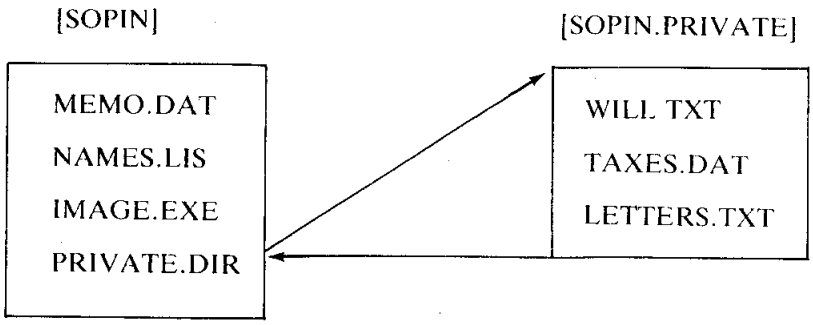
หมายถึง ให้ display มูลค่าของแฟ้มข้อมูลชื่อ CONTENTS.DAT ซึ่งเป็นของผู้ใช้เครื่องที่มีไดเรคทอรี เป็น [JONES]

4.1.4 ซับไดเรคทอรี (Subdirectories)

แฟ้มข้อมูลทั้งหลายสามารถจัดระเบียบในซับไดเรคทอรีได้ ซับไดเรคทอรีแต่ละชุดเป็นแฟ้มข้อมูล (จัดระเบียบในไดเรคทอรีที่สูงกว่า) ซึ่งประกอบด้วยแฟ้มข้อมูลอื่น ๆ

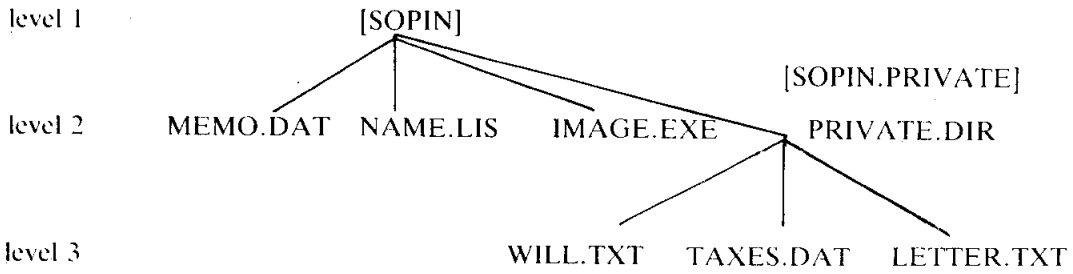
ตัวอย่าง

$ create/directory [sopin. private]



รูปแสดงไดเรคทอรีชื่อ [SOPIN] ประกอบด้วย ซับไดเรคทอรีหนึ่งชุดชื่อ [SOPIN. PRIVATE]

เขียนเป็น tree diagram ดังนี้



ชื่อของซับไดเรคทอรีเกิดจากการเอาชื่อของมันไปต่อกับ ชื่อของไดเรคทอรีแยกกันด้วยเครื่องหมาย period หนึ่งตัว

ตัวอย่าง

$ type [jones.datafiles] memo.sum

หมายถึง ให้ display แฟ้มข้อมูลชื่อ MEMO.SUM ซึ่งจัดระเบียบในซับไดเรคทอรี [JONES. DATAFILES] และซับไดเรคทอรีไฟล์ ชื่อ DATAFILES.DIR ถูกจัดระเบียบในไดเรคทอรี [JONES]

เราอาจจะใช้ซับไดเรคทอรีจัดการ (organize) และแยกแฟ้มข้อมูลต่าง ๆ ของเราได้

4.1.4.1 การสร้างซับไดเรคทอรี

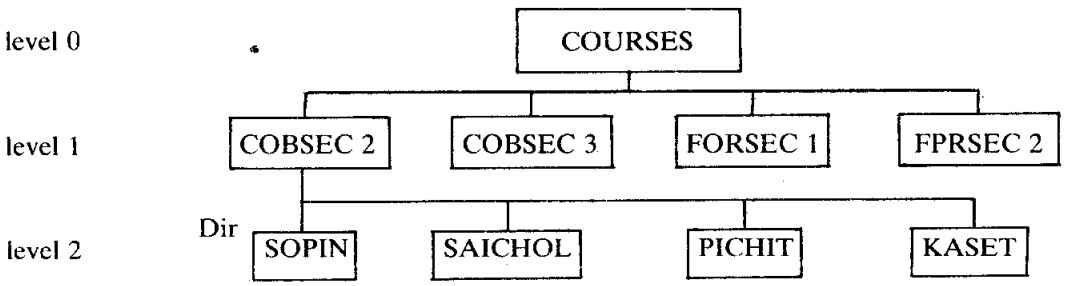
โดยปกติ ผู้จัดการระบบเครื่องคอมพิวเตอร์จะให้ไดเรคทอรีหนึ่งชุดแก่ผู้ใช้เครื่องแต่ละคนเพื่อใช้สำหรับบำรุงรักษา (maintain) แฟ้มข้อมูลต่าง ๆ ถ้าเราเป็นผู้ใช้เครื่องบ่อย และมีงานหลาย application เราจะพบว่า การสร้างซับไดเรคทอรีขึ้นมาหลายชุด เพื่อจัดระเบียบงานของเราในไดเรคทอรีหลัก (main directory) จะสะดวกขึ้น เราสามารถสร้างซับไดเรคทอรีกี่ชุดก็ได้ในไดเรคทอรีใด ไดเรคทอรีหนึ่ง เพื่อใช้ในการสร้างแฟ้มข้อมูลต่าง ๆ

ตัวอย่าง

$ create/directory [cobsec2. sopin]

หมายถึง สร้างซับไดเรคทอรีไฟล์ชื่อ SOPIN.DIR ในไดเรคทอรี [COBSEC2] ผลลัพธ์ในซับไดเรคทอรีจะเป็นชื่อ [COBSEC2.SOPIN] จากนั้นเราสามารถนำชื่อซับไดเรคทอรี [COBSEC2.SOPIN] ไปใช้ได้ในคำสั่งต่าง ๆ หรือโปรแกรมต่าง ๆ

4.1.4.2 การเปลี่ยน default ไดเรคทอรีของเรา



การสร้างไดเรคทอรี หรือซับไดเรคทอรีอีกชุดหนึ่ง ให้เป็น default ไดเรคทอรีของเราใช้คำสั่ง SET DEFAULT

ตัวอย่าง

```
$ set default [cobsec2.sopin]
$ edit newfile.txt
Input file does not exist
[EOB]
*
```

ตัวอย่าง

```
$ sh def
SYS$USER : [COBSEC2.SOPIN]
$ dir
```

ต้องการทราบว่าใน SOPIN.DIR มีแฟ้มข้อมูลอะไรอยู่บ้าง

หมายถึง สร้างแฟ้มข้อมูลใหม่หนึ่งชุดในซับไดเรคทอรี [COBSEC2.SOPIN] โดยการเปลี่ยน default ไดเรคทอรีของเรา จากนั้นสร้างแฟ้มข้อมูลชื่อ NEWFILE.TXT ด้วยคำสั่ง EDIT, แฟ้มข้อมูลนั้นจะถูกจัดระเบียบในซับไดเรคทอรี [COBSEC2.SOPIN]

นอกจากนี้แล้ว เรายังอาจใช้คำสั่ง SET DEFAULT เปลี่ยน default ดิสค์ของเราได้เช่นกัน

ตัวอย่าง คำสั่งกำหนดให้ PILOT เป็น default ไดเรคทอรีของเรา

```
$ set default pilot :
```

ในที่นี้ PILOT: เป็น device name

หลังจากเราใช้คำสั่งนี้แล้ว เครื่องคอมพิวเตอร์จะใช้ดิสค์ PILOT : เป็น default ดิสค์สำหรับแฟ้มข้อมูลทั้งหมดที่เราเอามาใช้หรือสร้างใหม่

เราสามารถเปลี่ยน default ดิสค์และไดเรคทอรีของเราได้บ่อยเท่าที่สะดวกตามต้องการ การเปลี่ยนแปลงที่เราทำด้วยคำสั่ง SET DEFAULT ยังมีผลอยู่จนกว่าจะมีการใช้คำสั่ง SET DEFAULT อีกชุดหนึ่ง หรือ log out

4.2 การใช้ logical names เพื่อประหยัดเวลา

A logical name เป็นชื่อซึ่งกำหนดให้เท่ากับ equivalence string หรือ list ของ equivalence strings ส่วน an equivalence string อาจจะเป็นกรุ๊ปใดกรุ๊ปหนึ่งของตัวอักขระ แต่ส่วนใหญ่แล้ว an equivalence string คือ file specification, device name หรือ logical name อีกชื่อหนึ่ง เราใช้ logical names ในวัตถุประสงค์ต่อไปนี้

a) ลดการพิมพ์โดยใช้ logical names เป็นวิธีลัด (a short way) ของการกำหนดแฟ้ม

ข้อมูล หรือไดเรคทอรีซึ่งเราอ้างถึงบ่อย ๆ

b) หลีกเลี่ยงความสับสนเกี่ยวกับตำแหน่งของ volumes

c) รักษาโปรแกรมและ command procedurs ให้เป็นอิสระจาก physical file specifications

เราสามารถใช้ได้ทั้งคำสั่ง DEFINE หรือคำสั่ง ASSIGN ในการสร้าง logical name แต่ละชื่อ โดยบอกความสัมพันธ์ของชื่ออีกชื่อหนึ่ง และ equivalence name

แต่ 2 คำสั่งนี้มีรูปแบบแตกต่างกัน ในบทนี้จะแสดงการกำหนด logical name ด้วยคำสั่ง DEFINE สำหรับรายละเอียดของคำสั่ง ASSIGN ให้ดูในหนังสือ VAX/VMS DCL dictionary

ตัวอย่าง

```
$ define zap dracul :: doc1 :: [malvolid] zapists.dat; 21
```

หมายถึง กำหนดให้ logical name ZAP ให้เท่ากับ file specification DRACUL :: DOC1 : [MALVOLIO] ZAPISTS.DAT;21

หลังจากเราใส่คำสั่งนี้แล้ว เราสามารถใช้ logical name ZAP แทนที่ทุกแห่งของ file specification ชื่อยาวนั้น

ตัวอย่าง คำสั่ง display มูลค่าของแฟ้มข้อมูลข้างต้น

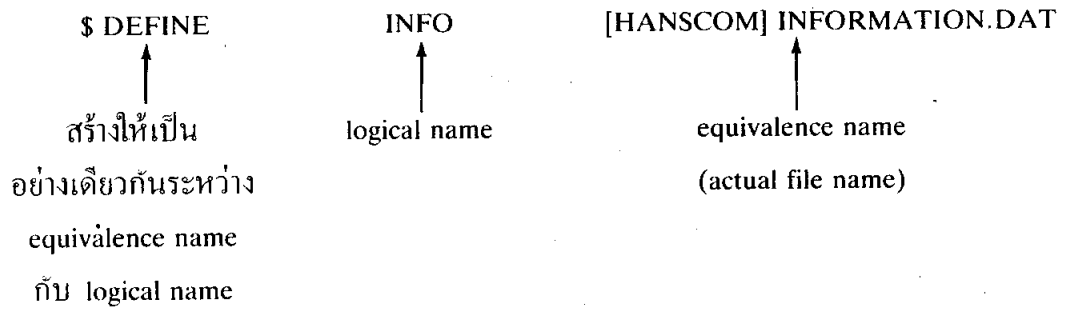
```
$ type zap
```

4.2.1 การใช้ Logical names

เราใช้คำสั่งที่มีรูปแบบข้างล่างนี้กำหนด logical name ชื่อใดชื่อหนึ่ง

```
$ DEFINE logical-name equivalence-name
```

ตัวอย่าง



คำสั่งนี้ หมายถึง กำหนด logical name INFO ให้กับแฟ้มข้อมูลชื่อ INFORMATION. DAT ซึ่งอยู่ในไดเรคทอรี ชื่อ [HANSCOM]

เมื่อเราต้องการ access แฟ้มข้อมูลชื่อ INFORMATION.DAT ในไดเรคทอรี [HANSCOM] เราสามารถใช้ logical name INFO

ตัวอย่าง คำสั่ง display แฟ้มข้อมูลข้างต้น

```
$ type info
.
.
```

เมื่อเราสร้าง logical name แต่ละชื่อ ชื่อนั้นจะถูกเก็บรักษาในตาราง logical name, ตาราง logical name จะประกอบด้วยเซทของ logical names กับ equivalence names ของมันสำหรับรายละเอียดเพิ่มเติมเกี่ยวกับตาราง logical name ให้ดูในหนังสือ VAX/VMS DCL dictionary

เราใช้คำสั่ง SHOW LOGICAL ให้แสดง logical name และ equivalence name ของมัน

ตัวอย่าง คำสั่งให้ display logical name HOME

```
$ show logical home
.
.
```

เครื่องจะค้นหา logical name HOME ในตาราง logical name ถ้าพบมันจะแสดง logical name และ equivalence name ของมัน ออกมาและบอกชื่อของตาราง logical name ซึ่งมันพบ logical name

ตัวอย่าง information message

```
0 "HOME" [super] = "BOOK2: [JACK]" (LNM$PROCESS_TABLE)
```

หมายถึง logical name HOME อยู่ในตาราง logical name มี equivalence name ชื่อ BOOK2: [JACK]

ตัวอย่าง

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

หมายถึง เราใช้คำสั่ง DEFINE กำหนด logical name MYFILE ให้กับแฟ้มข้อมูลชื่อ PERSONNEL.REC ที่มีรายชื่อในไดเรคทอรี CHUCK แล้ว display แฟ้มข้อมูลนี้บนเทอร์มินัล ด้วยคำสั่ง TYPE

logical name แต่ละชื่ออาจกำหนดให้เฉพาะส่วนแรกของ file specification ได้

ตัวอย่าง

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

หมายถึง คำสั่ง DEFINE กำหนด logical name TESTให้กับ disk device และไดเรคทอรี SCIENCE4: [MALCOLM.TESTFILES] ต่อจากนั้น คำสั่ง RUN execute โปรแกรม อิเมท (program image) MEMO.EXE ซึ่งจัดระเบียบในซับไดเรคทอรีและคำสั่ง PRINT พิมพ์แฟ้มข้อมูลอีกชุดหนึ่ง

เครื่องจะตรวจสอบ file specification เสมอ เพื่อดูว่า ถ้าส่วนของ file specification นั้นข้างหน้าเครื่องหมาย colon (:) เป็น logical name เช่นในตัวอย่างข้างต้นเครื่องจะแทนที่ด้วย equivalence name

4.2.2 System default logical names

เมื่อเรา log in เครื่อง หรือส่ง a batch job เครื่องจะจัด default logical names ให้จำนวนหนึ่ง, ชื่อเหล่านี้จะถูกใช้โดย command interpreter เพื่ออ่านคำสั่งของเราและพิมพ์คำตอบรับ (responses) หรือข้อความผิดพลาด (error messages)

เครื่องคอมพิวเตอร์ส่วนใหญ่จะ default logical names ในรูปแบบดังนี้

```
XXX$name
```

เมื่อ XXX หมายถึง ส่วนประกอบของเครื่องที่ใช้ logical name

การใช้เครื่องหมาย $ หนึ่งตัวภายใน logical name ถูกสงวน (reserved) ไว้สำหรับ DIGITAL

คอมพิวเตอร์หลายเครื่อง default logical names ดังนี้

ชื่อ	ใช้สำหรับ
SYS$COMMAND	เป็น default device name ของเทอร์มินัล
SYS$INPUT	เป็น default input stream สำหรับเครื่องอ่านคำสั่งและโปรแกรมอ่านข้อมูล
SYS$OUTPUT	เป็น default output stream สำหรับเครื่องพิมพ์คำตอบรับ คำสั่ง และโปรแกรมพิมพ์ข้อมูล
SYS$ERROR	เป็น default device สำหรับเครื่องพิมพ์ error ทั้งหมดและทำ information messages
SYS$DISK	เป็น default disk device ของเรา