บทที่ 8 การกำหนดการทำงานของเชลล์สคริพต์

จุดประสงค์เชิงพฤติกรรม

1 เพื่อให้สามารถใช้คำสั่ง test สร้างเงื่อนไขได้
2 เพื่อให้สามารถใช้คำสั่ง if-then-else ได้
3 เพื่อให้สามารถใช้คำสั่ง while ได้
4 เพื่อให้สามารถใช้คำสั่ง until ได้
5 เพื่อให้สามารถใช้คำสั่ง for ได้
6 เพื่อให้สามารถใช้คำสั่ง case ได้
7 เพื่อให้สามารถใช้คำสั่ง select ได้
8 เพื่อให้สามารถประยุกต์ใช้คำสั่งเงื่อนไขในเชลล์สคริพต์ได้

ในบทนี้จะได้กล่าวถึงการกำหนดการทำงานของเชลล์สคริพต์เพื่อใช้ในการควบคุมทิศทางของ เชลล์สคริพต์ซึ่งเป็นสคริพต์ที่มีความซับซ้อนมากขึ้น เชลล์สคริพต์ทั่วไปนั้น จะมีการทำงานเริ่มต้นจาก ต้นไฟล์ไปยังท้ายไฟล์ ซึ่งจะไม่มีการควบคุมใดๆ แต่สำหรับงานที่มีความซับซ้อนต้องมีการตัดสินใจ นั้น จะต้องมีการเขียนสคริพต์ที่สามารถกำหนดทิศทางการทำงานได้

8.1. คำสั่ง Test

การที่กำหนดให้เชลล์ทำงานในทิศทางใดนั้นจะอาศัยเงื่อนไขในการกำหนดทิศทาง โดยค่าของ เงื่อนไขที่ใช้ในการกำหนดทิศทางนั้นจะมีอยู่ 2 ค่า คือ ค่าจริงและค่าเท็จ สำหรับการสร้างเงื่อนไขนั้น เราจะสร้างด้วยการใช้คำสั่ง test โดยจะแบ่งลักษณะของการสร้างเงื่อนไขเป็น 3 ลักษณะ

ดังนี้คือ

- เงื่อนไขการเปรียบเทียบข้อความ
- เงื่อนไขการตรวจสอบไฟล์
- เงื่อนไขการเปรียบเทียบค่าของตัวเลข

8.1.1. การเปรียบเทียบข้อความ

การสร้างเงื่อนไขโดยการเปรียบเทียบข้อความนี้แบ่งเป็นย่อยๆ ได้อีก 4 ประเภทดังนี้

• test string1 = string2 จะเปรียบเทียบ string1 กับ string2 โดยถ้า string1 และ string2 เหมือนกัน ผลลัพธ์ที่ได้จะเป็นค่าจริง แต่ถ้าไม่เหมือนจะให้ค่าเท็จออกมา

- test string1 != string2 เป็นการเปรียบเทียบที่กำหนดไว้ว่า ถ้า string1 และ string2 ไม่เหมือนกับผลลัพธ์ที่ได้เป็นค่าจริง แต่ถ้า string ทั้งสองเหมือนกันก็จะให้ค่าเท็จออกมา
- test –n string เป็นการเปรียบเทียบ string ว่าเป็น null หรือไม่ถ้า string นี้ไม่เป็น null ผลลัพธ์จะให้ค่าจริง แต่ถ้าเป็น null จะให้ค่าเท็จออกมา
- • test –z string ถ้า string นี้เป็น null ก็จะให้ค่าจริง แต่ถ้าไม่เป็น null จะให้ค่าเท็จ ออกมา

8.1.2. การตรวจสอบไฟล์

- test –d name จะตรวจสอบ name ที่กำหนดว่าเป็นไดเร็กตอรี่ที่มีอยู่จริงหรือไม่ถ้าใช่จะให้ค่าจริงออกมา แต่ถ้าไม่ใช่จะให้ค่าเท็จ
- test -e name จะตรวจสอบว่า name เป็นไฟล์หรือไดเร็กตอรีที่มีอยู่จริงหรือไม่ถ้าใช่จะให้ค่าจริง แต่ถ้าไม่ใช่จะให้ค่าเท็จ
- test -f name จะตรวจสอบว่า name เป็นไฟล์ที่มีอยู่จริงหรือไม่ซึ่งนอกจากไฟล์นี้จะต้องมีอยู่จริงแล้วยังต้องเป็นไฟล์ธรรมดาที่ไม่ใช่ไดเร็กตอรีหรือไฟล์ชนิดพิเศษชนิดอื่นๆถึงจะให้ผลลัพธ์เป็นค่าจริง ถ้าผิดเงื่อนไขจากนี้จะให้ค่าเท็จออกมา
- test -r name ถ้า name ที่กำหนดเป็นไฟล์หรือไดเร็กตอรีที่เปิดสิทธ์ในการใช้งานให้สามารถอ่านได้ค่าจริงออกมา แต่ถ้าไม่เปิดสิทธ์ให้อ่านก็จะให้เท็จออกมาแทน
- test -w name ถ้า name เป็นไฟล์หรือไดเร็กตอรีที่เปิดสิทธ์ในการใช้งานให้สามารถ เขียนได้ก็จะให้ค่าจริง แต่ถ้าไม่เปิดสิทธ์สามารถเขียนได้ก็จะให้ค่าเท็จออกมา
- test -x name ถ้า name เป็นไฟล์หรือไดเร็กตอรีที่เปิดสิทธ์ให้สามารถเขียนได้จะ ออกมา
- test -s name ถ้า name เป็นไฟล์ที่มีอยู่จริงและมีขนาดไม่เป็น 0 ผลลัพธ์ที่ได้จะเป็น ค่าจริง แต่ถ้านอกเหนือจากนี้ผลลัพธ์ที่ได้จะเป็นเท็จแทน
- test -o name แต่ถ้าผู้ที่เรียกใช้คำสั่งนี้เป็นเจ้าของไฟล์หรือไดเร็คทอรีที่ชื่อ name แล้วผลลัพธ์เป็นค่าจริง แต่ถ้าผู้เรียกใช้คำสั่งไม่ได้เป็นเจ้าของไฟล์หรือไดเร็กตอรีนั้นผลลัพธ์จะออกเป็น ค่าเท็จแทน
- test -G name ถ้าเลขกลุ่ม (Group ID) ของไฟล์หรือไดเร็กตอรีที่ชื่อ name มีค่า เหมือนเลขกลุ่มของผู้ใช้ที่เรียกคำสั่งผลลัพธ์ที่ได้จะเป็นค่าจริง แต่ถ้าไม่ใช่จะเป็นค่าเท็จแทน
- test name1 –nt name2 ถ้า name1 เป็นไฟล์ทีใหม่ที่ชื่อ name2 จะเป็นค่าจริงออกมา แต่ถ้าไม่ใช่จะให้ค่าเท็จแทน
- test name1 –ot name2 ถ้า name1 เป็นไฟล์ที่เก่าชื่อ name2 จะให้ค่าจริง แต่ถ้า ไม่ใช่จะให้ค่าเท็จแทน

การที่จะเปรียบเทียบว่าไฟล์ไหนเก่าหรือใหม่กว่ากันนั้น จะเปรียบเทียบจากเวลาที่แก้ไขไฟล์ (Modification Time) นั้นๆ

8.1.3. การเปรียบเทียบค่าของตัวเลข

นอกจากการเปรียบเทียบใน 2 ลักษณะที่กล่าวมาแล้ว วิธีในการเปรียบเทียบค่าของตัวเลขเป็น อีกวิธีหนึ่งที่ใช้กัน เพราะในเชลล์สคริปต์นั้นจะมีการใช้ค่าของตัวเลขอยู่มาก โดยจะสามารถแบ่งการ เปรียบเทียบออกได้ดังนี้

- test num1 –lt num2 ถ้าค่าของ num1 น้อยกว่า num2 จะให้ค่าจริงออกมา แต่ถ้า มากกว่าหรือเท่ากันจะให้ค่าเท็จแทน
- test num1 –le num2 ถ้าค่าของ num1 น้อยกว่าหรือเท่ากับ num2 แล้วจะได้ผลลัพธ์ที่เป็นค่าจริง แต่ถ้ามากกว่าจะเป็นค่าเท็จ
- test num1 –gt num2 ถ้าค่าของ num1 มากกว่า num2 จะค่าจริง แต่ถ้าน้อยกว่าหรือ เท่ากันจะให้ค่าเท็จแทน
- test –ge num2 ถ้าค่าของ num1 มากกว่าหรือเท่ากับ num2 แล้ว จะได้ผลลัพธ์ที่เป็น จริง แต่ถ้าน้อยกว่าจะได้ค่าเท็จแทน
- test num1 –gq num2 ถ้าค่าของ num1 เท่ากับ จะได้ค่าจริง แต่ถ้าไม่เท่ากันจะเป็น ค่าเท็จ
- test num1 –ne num2 ถ้าค่าของ num1 ไม่เท่ากับ num2 จะได้ค่าจริง แต่ถ้าค่าทั้ง สองเท่ากันจะได้ค่าเท็จแทน

คำสั่ง test ที่ใช้ในการทดสอบเงื่อนไขนั้น สามารถใช้เครื่องหมาย [ ] แทนได้ เช่น test num1 –ge num2 ถ้าใช้เครื่องหมาย [ ] จะได้ดังนี้ [num1 -ge num2] เป็นต้น

การทดสอบเงื่อนไขที่กล่าวมาแล้วทั้ง 3 แบบ จะให้ค่าของการทดสอบอยู่ 2 ค่า คือ ค่าจริงและ ค่าเท็จ ซึ่งเราสามารถที่จะนำผลลัพธ์ของ

เงื่อนไขแต่และเงื่อนไขมาประมวลผลร่วมกันโดยใช้สัญลักษณ์ต่างๆ ดังนี้

! เป็นการเปลี่ยนค่าของผลลัพธ์จากจริงเป็นเท็จ หรือ จากเท็จเป็นจริง (not)

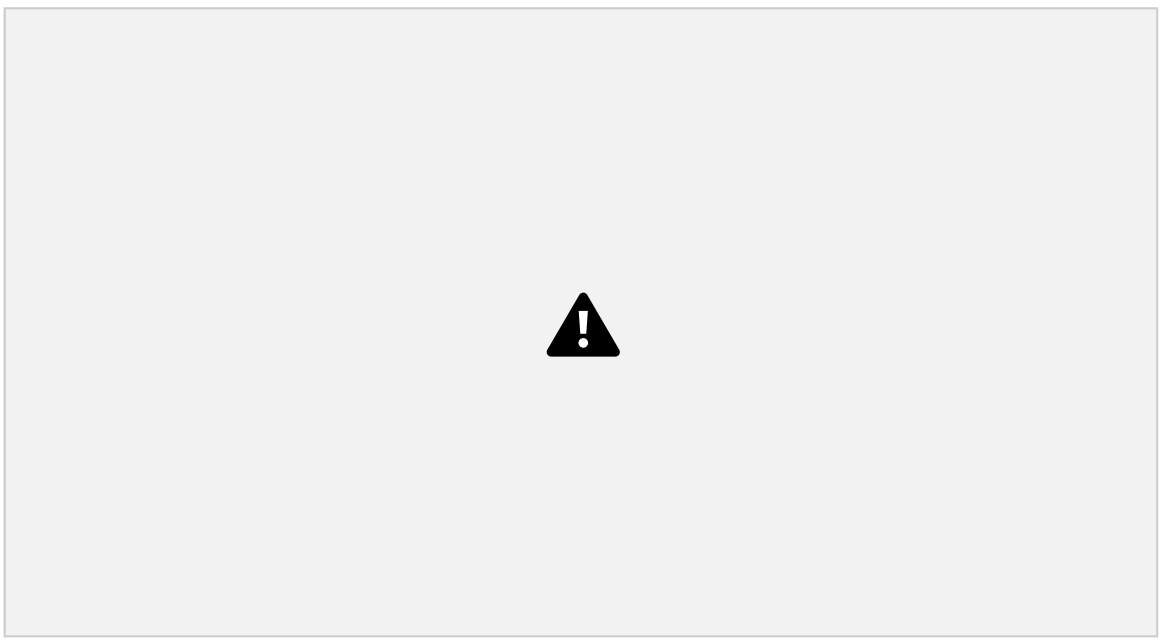
-a เป็นการนำผลลัพธ์ของเงื่อนไข 2 เงื่อนไขมาประมวลผลร่วมกันแบบแอนด์ (and) คือ ถ้า เงื่อนไขแรกและเงื่อนไขที่สองเป็นจริงทั้งคู่ผลลัพธ์ที่ได้จากการประมวลผลร่วมกันจะเป็นจริง แต่ถ้า เงื่อนใดเงื่อนไขหนึ่งหรือทั้งคู่เป็นเท็จ ผลลัพธ์ที่ได้จากการมาประมวลผลร่วมกันจะเป็นเท็จ

-o เป็นการนำผลลัพธ์ของเงื่อนไขมาประมวลผลร่วมกันแบบออร์ (or) คือ ถ้าเงื่อนไขแรกและ เงื่อนไขที่สองเป็นเท็จทั้งคู่ผลลัพธ์ที่ได้จากการประมวลผลร่วมกันจะเป็นเท็จ แต่ถ้าเงื่อนไขใดเงื่อนไข หนึ่งเป็นจริง ผลลัพธ์ที่ได้จากการประมวลผลร่วมกันเป็นจริง

ถ้าเราใช้การประมวลแบบ –a และ –o ร่วมกันแล้ว เชลล์จะให้ความสำคัญกับแบบ –a มากกว่า คือ จะประมวลผลแบบ –a ก่อน แล้วค่อยนำผลลัพธ์ที่ได้มาประมวลผลกับแบบ –o อีกครั้ง

เพื่อให้เข้าใจเรื่องการสร้างเงื่อนไข ให้พิจารณาจากตัวอย่างที่แสดงไว้ในรูปที่ 8.1 ซึ่งเมื่อสร้าง เงื่อนไขเสร็จแล้วจะสามารถตรวจสอบผลลัพธ์ของเงื่อนไขได้โดยใช้คำสั่ง echo $? ถ้าผลลัพธ์ของ เงื่อนไขเป็นจริงก็จะให้ค่า 0 ออกมา แต่ถ้าเงื่อนไขเป็นเท็จจะให้ค่า 1 ออกมาแทน

รูปที่ 8.1 ตัวอย่างการใช้คำสั่ง test ที่กล่าวผ่านมาข้างต้นนั้น เป็นวิธีการสร้างเงื่อนไข ต่อไปจะมาทำความเข้าใจคำสั่งที่จะนำมาใช้ในการควบคุมทิศทางการทำงานของเชลล์สคริปต์ ซึ่งคำสั่งที่ใช้ในการควบคุมนี้บางคำสั่งก็ต้องใช้ความรู้ในการสร้างเงื่อนไขที่ได้ศึกษากันมาแล้ว โดยคำสั่งต่าง ๆ มีดังนี้



- if – then – else
- while
- until
- for
- case
- select

คำสั่งทั้งหมดที่ได้กล่าวมานี้จะมีลักษณะการทำงานที่แตกต่างกัน บางคำสั่งจะทำงานแบบวน ไปเรื่อย ๆ แต่บางคำสั่งก็จะทำงานเพียงครั้งเดียว ซึ่งเนื้อหาในหัวข้อต่อไปกล่าวถึงคำสั่งเหล่านี้

8.2. คำสั่ง if - then

- else รูปแบบการใช้

คำสั่ง

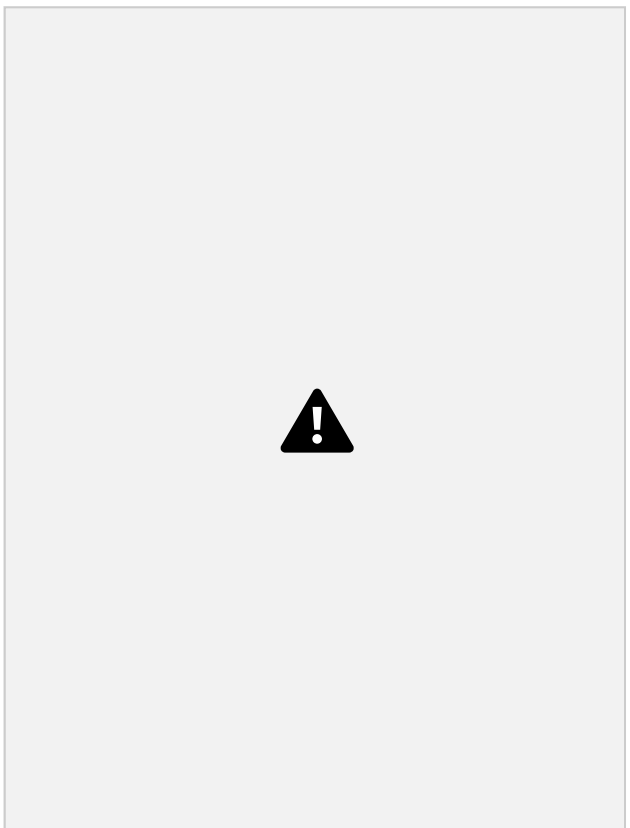
if เงื่อนไข then

ชุดคำสั่ง

[elif เงื่อนไข then ชุดคำสั่ง]

[else ชุดคำสั่ง] fi จากรูปแบบของคำสั่งด้านบนหลัง if จะตามด้วยเงื่อนไขที่ระบุ โดยถ้าเงื่อนไขเป็นจริงก็จะ ทำงานในชุดคำสั่งหลัง then แต่ถ้าเงื่อนไขเป็นเท็จก็จะไปทำงานในชุดคำสั่งหลัง else แทน แต่ถ้าหาก ไม่ได้กำหนด else ไว้ก็จะไม่มีการทำคำสั่งใด ๆ ในกรณีที่เงื่อนไขเป็นเท็จส่วน elif นั้นจะใช้ในกรณีที่เงื่อนไขที่ if เป็นเท็จก็จะมาตรวจสอบเงื่อนไขที่ elif อีกทีหนึ่ง ซึ่งถ้าเป็นจริงก็จะทำงานในชุดคำสั่งหลัง then ที่อยู่ต่อจาก elif แทน

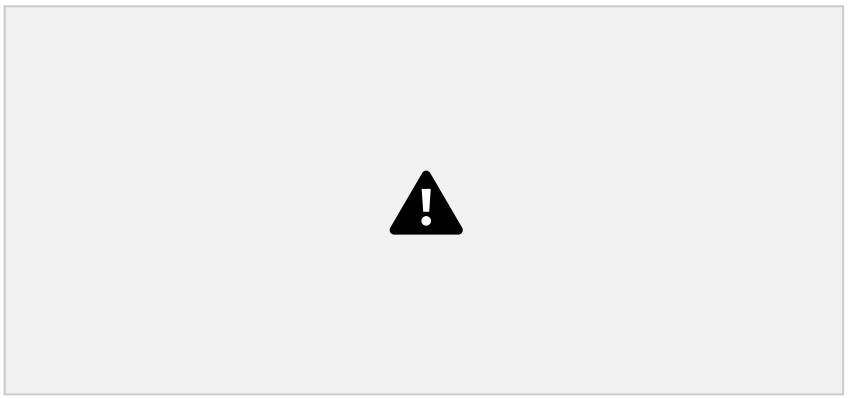
รูปที่ 8.2 การทำงานของคำสั่ง if - then - else



ตัวอย่างไฟล์ Ex1 การใช้คำสั่ง if - then - else

```
if  [ $# = 0 ] then echo ” Usage : Ex1 your_name ” elif
[ $1 = ’David’ ]   then echo ” David, Please contact me
now. ” else echo ” Hello, $1. ” fi
```

เมื่อเรียกให้สคริพต์ในตัวอย่างทำงาน ผลลัพธ์จะเป็นดังรูปที่ 8.3



รูปที่ 8.3 ตัวอย่างผลลัพธ์การเรียกเชลล์สคริพต์

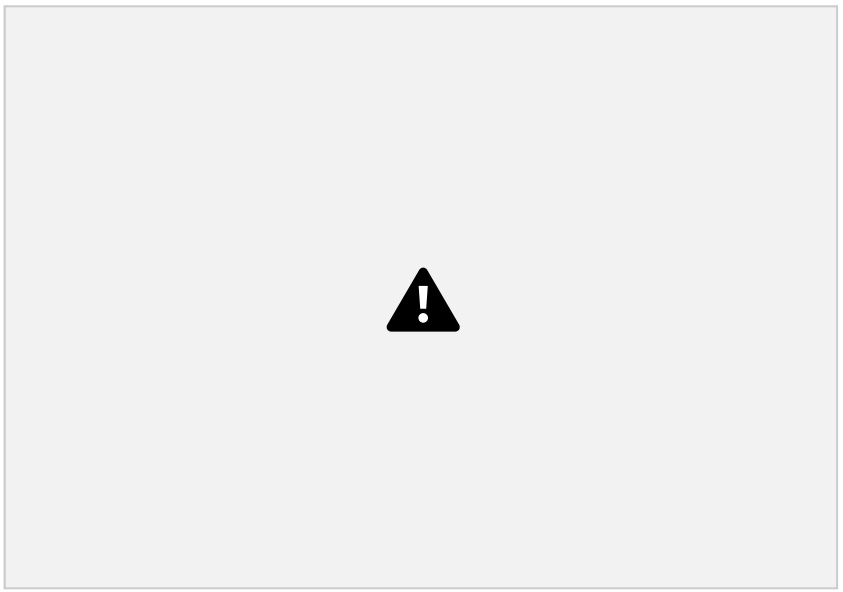
จากเชลล์สคริพต์ตามตัวอย่าง เมื่อเราเรียกให้เชลล์สคริพต์มีการทำงาน เชลล์สคริพต์จะ ตรวจสอบว่าว่า ผู้ใช้ได้ป้อนอาร์กิวเมนต์มาด้วยหรือไม่ถ้าไม่(นั่นคือ $# เป็น 0) จะแสดงข้อความ "Usage : Ex1 your_name" ออกมา แต่ถ้าผู้ใช้ป้อนอาร์กิวเมนต์มาด้วย ก็จะตรวจสอบว่าเป็น David หรือไม่ถ้าใช่ก็จะแสดงข้อความว่า "David, Please contact me now." แต่ถ้าไม่ใช่ก็จะแสดง ข้อความว่า "Hello, ชื่อที่ป้อนเข้ามา" แทน นอกจากหลังคำสั่ง if จะตามด้วยเงื่อนไขแล้ว เรายังใช้คำสั่งต่าง ๆ แทนเงื่อนไขได้ด้วย โดยถ้า คำสั่งที่กำหนดไว้หลัง if ทำได้สำเร็จก็จะให้ค่าเป็นจริง และจะทำคำสั่งหลัง then แต่ถ้าทำคำสั่งไม่สำเร็จก็จะให้ค่าเท็จออกมา และจะทำคำสั่งหลัง else แทน เราสามารถนำคำสั่งหลาย ๆ คำสั่งมาสร้าง เป็นเงื่อนไขร่วมกันได้โดยจะใช้สัญลักษณ์ || แทนออร์ และใช้สัญลักษณ์ && แทนแอนด์ ตัวอย่างไฟล์ Ex2

```
if cd /book 2>/dev/null && ls /book/unix.txt > /dev/null 2>&1   then echo " Have unix.txt in book's
    directory. "   else   echo " Don't have /book/unix.txt or book's directory. " fi
```

เมื่อสร้างเชลล์สริปต์ตามตัวอย่างแล้วตั้งชื่อเชลล์สคริปต์ว่า Ex2 จาก

นั้นเรียกคำสั่งของไฟล ์สคริพต ์โดยเริ่มแรกเชลล์สคริปต์จะทำการตรวจสอบว่ามีไดเร็กตอรี book หรือไม่โดยใช้คำสั่ง cd book ในการตรวจสอบ และจะทำการเรียกคำสั่ง ls/book/unix.txt ถ้าทั้งสองคำสั่งทำสำเร็จ ก็จะ ทำงานในชุดคำสั่งหลัง then แต่ถ้ามีคำสั่งใดคำสั่งหนึ่งที่ทำไม่สำเร็จ ก็จะทำคำสั่งหลัง else แทนส่วน คำสั่ง ls/book/unix.txt นั้นเป็นคำสั่งที่จะให้ผลลัพธ์ออกมา ดังนั้นถ้าเราไม่ทำการรีไดเร็กชันผลลัพธ์ที่ได้ให้ไปที่ใดที่หนึ่งแล้วก็จะเกิดข้อพลาดขึ้น ซึ่งจากตัวอย่างจะทำการรีไดเร็กชันลงไฟล์ /dev/null ซึ่ง ไฟล์นั้นจะเปรียบเสมือนถังขยะของระบบนั่นเอง ส่วนข้อผิดพลาดที่อาจจะเกิดขึ้น ก็ทำการรีไดเร็คชัน ไปเก็บไว้ที่ไฟล์ /dev/null เช่นกัน

รูปที่ 8.4 การเรียกใช้สคริพต์ Ex2 ตามตัวอย่าง



8.3. คำสั่ง

while รูปแบบ

การใช้คำสั่ง

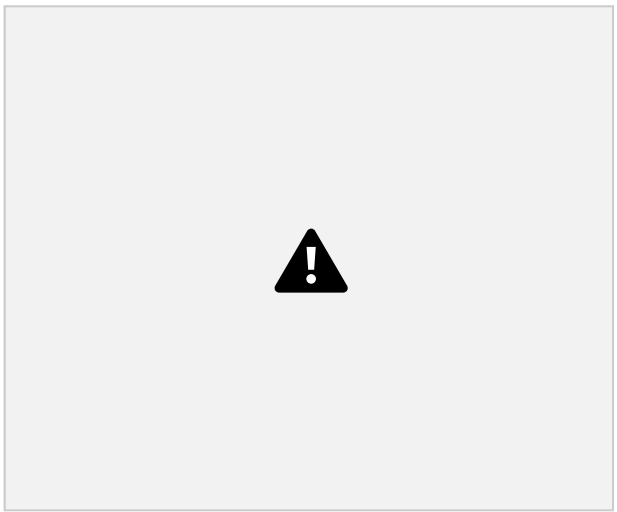
while เงื่อนไข do

ชุดคำสั่ง

done

จากรูปแบบของคำสั่ง while นี้เชลล์จะทำการตรวจสอบเงื่อนไขที่ระบุไว้หลัง while โดย เงื่อนไขเป็นจริงก็จะทำงานในชุดคำสั่งที่ระบุไว้หลัง do จนถึง done แล้วก็จะกลับไปตรวจสอบ เงื่อนไขหลัง while อีกครั้งหนึ่ง ถ้ายังเป็นจริงอีกก็จะไปทำชุดคำสั่งนั้นอีก โดยจะทำงานอย่างนี้ไป เรื่อยๆ จนกว่าผลลัพธ์ของเงื่อนไขหลัง while จะเป็นเท็จ

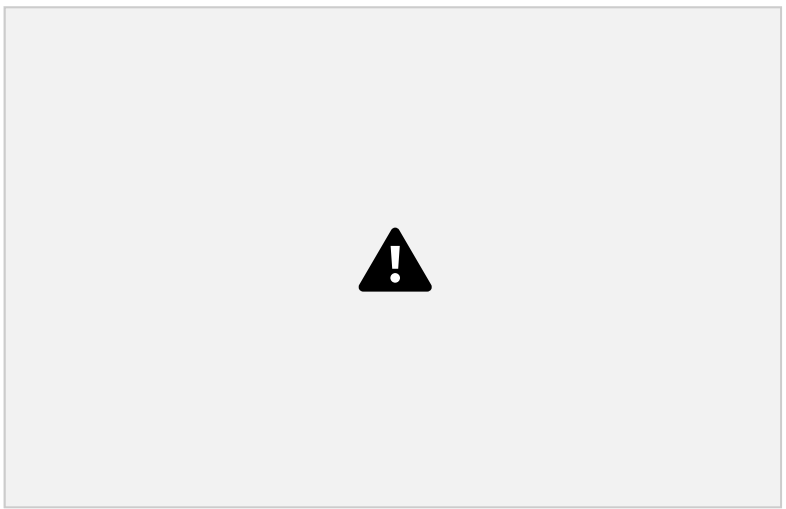
รูปที่ 8.5 การทำงานของ while



ตัวอย่างไฟล์ Ex3

```
command=$1 path=$PATH: while [ $path ] do
        cd ${path%%:*} 2>/dev/null   result=` ls $1 2>/dev/null`  if [ -n "$result" ]
      then
            echo " Have $1 in ${path%%:*} "   fi path=${path#*:}
done
```

ตัวอย่างนี้เป็นเชลล์สคริพต์ Ex3 **จะใช้ในการตรวจสอบว่าคำสั่งที่ต้องการนั้นอยู่ในไดเร็กตอรี ใดที่ระบุไว้ในตัวแปร** path **ฉะนั้นการใช้เชลล์สคริปต์** Ex3 **นี้จะต้องตามด้วยอาร์กิวเมนต์** 1**ตัวซึ่งเป็น คำสั่งที่ต้องการทราบว่าอยู่ในไดเร็กตอรีไหน**



รูปที่ 8.6 **ผลการทำงานของสตริพต์** Ex3 **การทำงานของเชลล์สคริปต์นี้มีขั้นตอนดังนี้ โดยเริ่มแรกจะทำการเก็บค่าอาร์กิวเมนต์ซึ่งเป็น ชื่อคำสั่งเก็บลงตัวแปรชื่อ** command **ต่อมาก็สร้างตัวแปร** path **ซึ่งเก็บค่าในตัวแปร** PATH **ของระบบ แล้วเติมเครื่องหมาย : เข้าไปที่ท้ายสุดของตัวแปร หลังจากนั้นก็จะเข้าสู่ลูป** while **ซึ่งเงื่อนไขของลูปนี้ คือจะทำงานไปเรื่อย ๆ ตราบเท่าที่ตัวแปร** path **ยังมีค่าที่ไม่เท่ากับ** null **อยู่ เมื่อเข้ามาในลูปแล้วก็จะทำ การย้ายไดเร็กตอรีไปอยู่ที่** ${path%%:*} **โดย** ${path%%:*} **จะเก็บค่า**

พาธตัวแรกของตัวแปร PATH และตัวแปร result จะเก็บค่าที่ได้จากการทำคำสั่ง ls $1 2>/dev/ เมื่อทำการ ls แล้ว ถ้าคำสั่งนี้มีอยู่ก็จะแสดงชื่อคำสั่งนั้นออกมาอีกครั้งแล้วเก็บลงตัวแปร result แต่ถ้าไม่คำสั่งนี้ในไดเร็กตอรีตัวแปร result ก็จะเป็น null ส่วนข้อความแสดงความผิดพลาดที่อาจเกิดขึ้นได้นั้นจะถูกส่งไปเก็บที่ /dev/null เมื่อได้ค่าตัวแปร result เรียบร้อยแล้ว ก็จะทำการตรวจสอบค่าในตัวแปร result ว่าเป็น null หรือไม่

ซึ่งถ้าไม่เป็นก็จะสรุปได้ว่ามีคำสั่งที่กำหนดอยู่ในไดเร็กตอรีนี้จริง ก็จะแสดงข้อความออกทางหน้าจอ ต่อมาก็จะเปลี่ยนค่าตัวแปร path ใหม่โดยจะตัดไดเร็กตอรีที่ได้ทำการตรวจสอบไปแล้วออก แล้ว กลับไปตรวจสอบเงื่อนไขเพื่อเข้าสู่ลูป while อีกครั้ง โดยมันจะทำงานอยู่ในลูปจนกระทั่งค่าในตัวแปร path เป็น null

8.4. คำสั่ง until

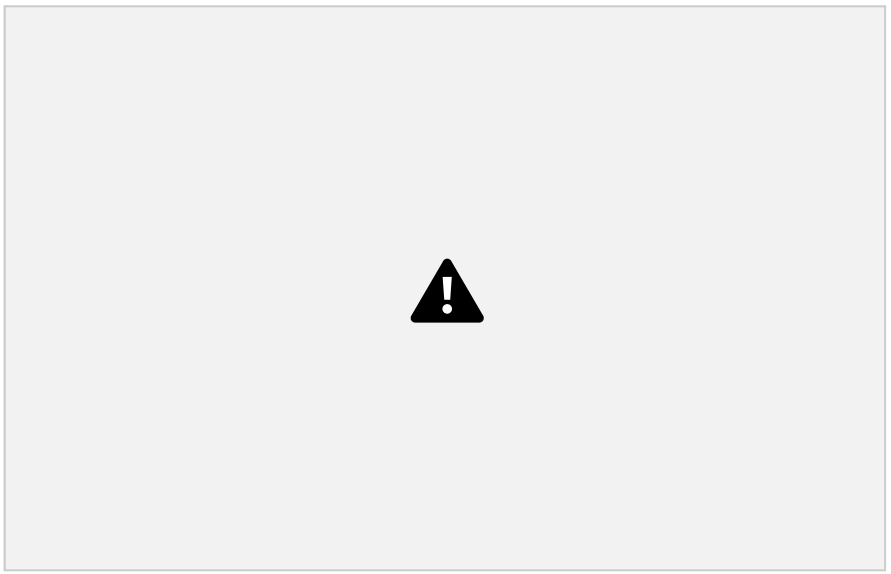
รูปแบบการใช้

คำสั่ง

until เงื่อนไข do
    ชุดคำสั่ง
done

การทำงานของคำสั่ง until นี้จะทำงานคล้ายกับคำสั่ง while แต่จะทำงานในลักษณะตรงข้าม กัน คือ ถ้าทำงานมาถึงลูป until เมื่อไหร่จะทำการตรวจสอบเงื่อนไขหลัก until ถ้าเงื่อนไขเป็นเท็จก็จะ ทำงานในชุดคำสั่งหลัง do ไปจนถึง done หลังจากนั้นก็จะมาตรวจสอบเงื่อนไขหลัง until อีกครั้ง ถ้าเป็นจริงก็จะหลุดออกจากลูป แต่ถ้าเงื่อนไขเป็นเท็จก็จะทำงานในชุดคำสั่งไปเรื่อย ๆ จนกว่าผลลัพธ์ของ เงื่อนไขหลัง until จะเป็นจริง

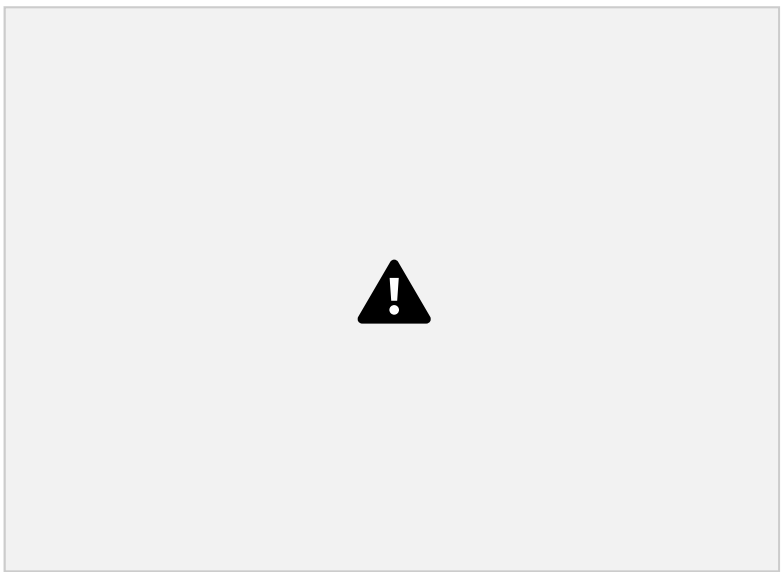
รูปที่ 8.7 การทำงานของลูป until ตัวอย่างไฟล์ Ex4



```
n=0 until [ $n -ge 5 ] do
      echo ” $n. Now, variable is less than 5. ”   n=`expr $n +
      1` done  echo ” $n. Now, variable is not less than 5. ”
```

ตัวอย่างนี้เป็นตัวอย่างง่าย ๆ เมื่อเรียกใช้เชลล์สคริปต์ Ex4 นี้ เชลล์สคริพต์จะตรวจสอบค่าของ ตัวแปร n ซึ่งเริ่มแรกกำหนดให้เป็น 0 ถ้าตราบใดที่ค่าของตัวแปร n ยังน้อยกว่า 5 อยู่ก็จะแสดงข้อความ ว่า “Now, variable is less than 5.” ไปเรื่อย ๆ จนกว่าค่าของตัวแปร n จะมากกว่าหรือเท่ากับ 5

รูปที่ 8.8 ผลการทำงานของเชลล์สคริพต์ Ex4



8.5. คำสั่ง for รูป

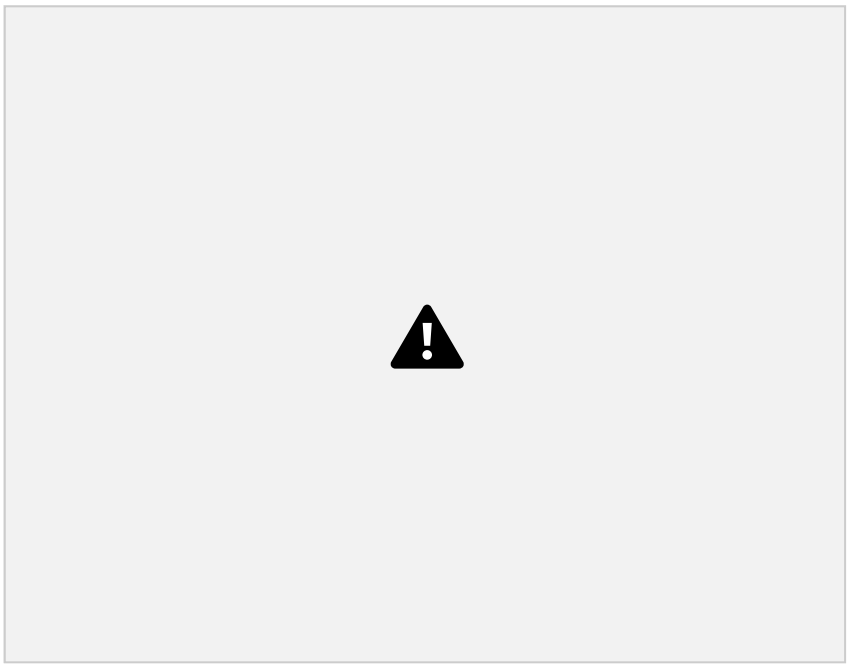
แบบการใช้คำสั่ง

```
for name [ in list] do
        ชุดคำสั่ง
done
```

การทำงานของลูป for นี้จะพิจารณาจากค่าของ name ถ้าตัวแปร name ยังมีค่าเก็บอยู่ที่ไม่ใช่ค่า null ก็จะเข้าไปทำงานในชุดคำสั่งที่กำหนด แต่ถ้าค่าในตัวแปร name เป็น null แล้วก็จะหลุดออก จากลูป for ทันที ส่วน in list จะมีหรือไม่ก็ได้ โดยถ้ามีจะหมายความว่าให้นำค่าที่ถูกเก็บอยู่ใน list มา เป็นค่า

ของ name แต่ถ้าไม่ได้กำหนด in list ก็จะหมายถึงให้นำค่าของตัวแปร name มาจากอาร์กิว เมนต์ที่ถูกป้อนเข้ามาพร้อมกับเชลล์สคริปต์

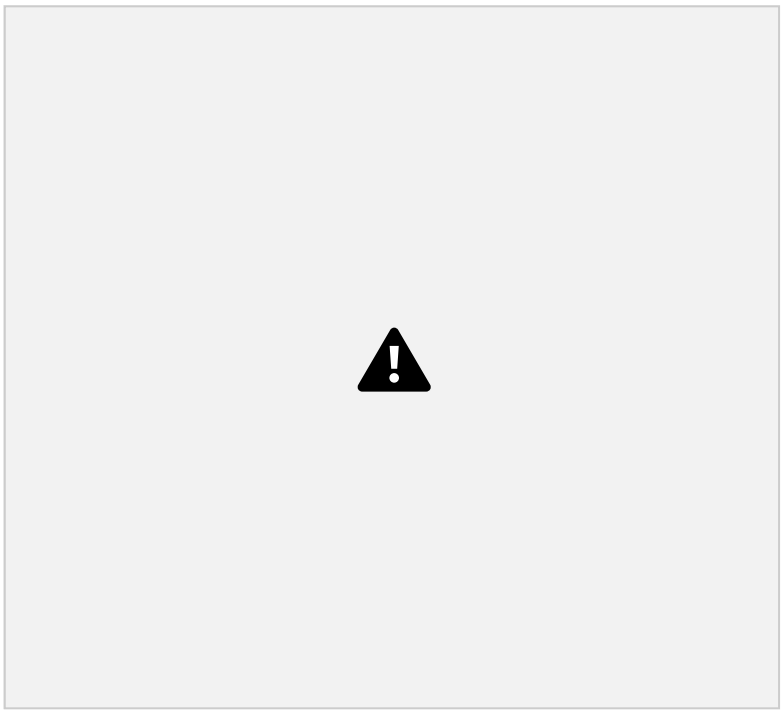
รูปที่ 8.9 การทำงานของลูป for ตัวอย่างไฟล์ Ex5



```
i=1
        echo  "This shell script have $# arguments..."
for vari
        do
                echo "Arguments $i is $vari"
                i=`expr $i+1`
        done
```

เชลล์สคริปต ์Ex5 จะทำการรับค่าอาร์กิวเมนต์ที่ถูกป้อนมากับการทำงานของเชลล์สคริปต ์แล้ว แสดงผลออกมาว่ามีอาร์กิวเมนต์ทั้งหมดกี่ตัว และมีอะไรบ้าง โดยใช้คุณสมบัติของลูป for เข้าช่วย

รูปที่ 8.10 ผลการทำงานของเชลล์สคริพต์ Ex5



8.6. คำสั่ง case

รูปแบบการใช้

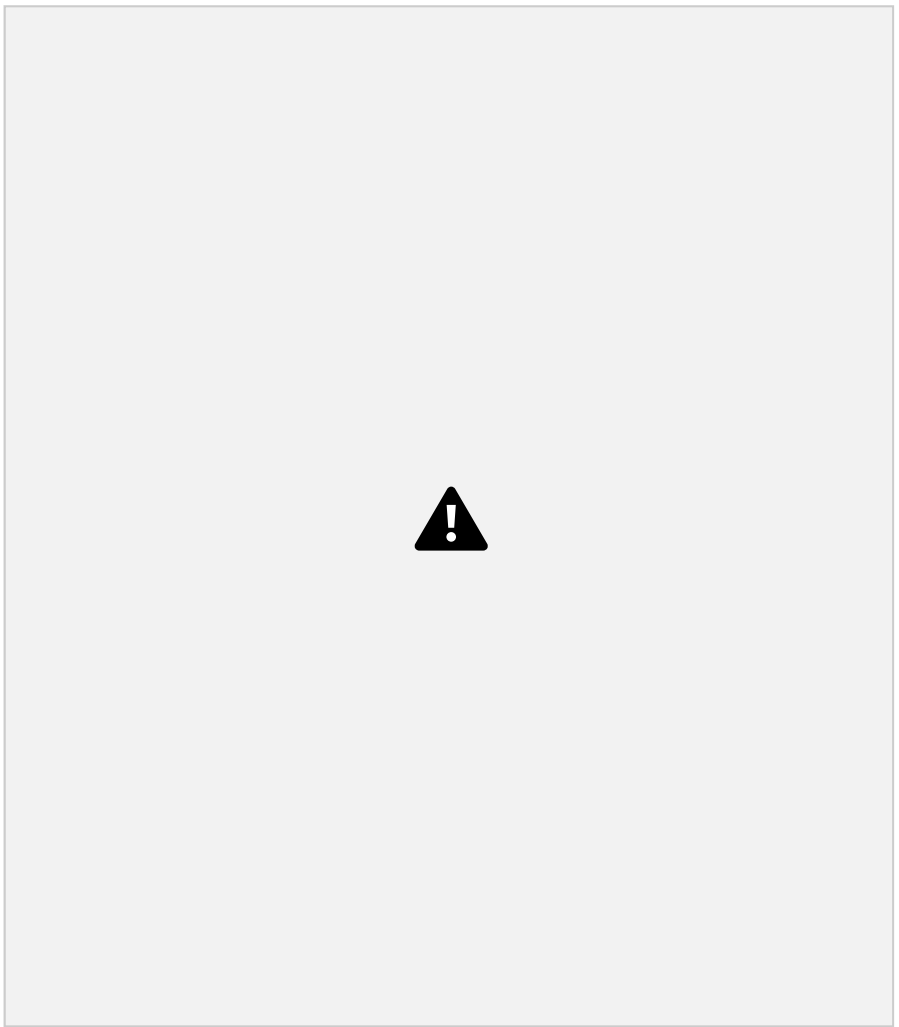
คำสั่ง

```
case name in choice 1 )
        ชุดคำสั่ง
1
        ;;
choice 2 )
        ชุดคำสั่ง2
        ;;
*)
        ชุดคำสั่งสุดท้าย
        ;;
esac
```

สำหรับคำสั่ง case นี จะทำการนำค่าที่เก็บอยู่ในตัวแปร name มาทำการตรวจสอบว่าตรงกับ ตัวเลือกใดที่ได้กำหนดไว้ใน choice ถ้าค่าของตัวแปร name นี้ตรงกับเงื่อนไขใดก็จะทำงานใน ชุดคำสั่งของตัวเลือกนั้น และแต่ละชุดคำสั่งของแต่ละตัวเลือกจะต้องจบด้วยเครื่องหมาย ;;

สำหรับตัวเลือกนั้นสามารถจะกำหนดเป็นค่าเดียว หรือหลายค่าก็ได้ โดยใช้เครื่องหมาย | คั่น ระหว่างค่า 2 ค่าที่นำมาใช้เป็นตัวเลือก นอกจากนี้ยังนำคุณสมบัติของไวด์การ์ด * หรือ ? หรือการ กำหนดค่าเป็นช่วงโดยใช้ [...] มาช่วยในการกำหนดตัวเลือกได้ด้วย และตัวเลือกสุดท้ายมักจะกำหนดให้เป็น * เพื่อว่าถ้าค่าของตัวแปร name ไม่ตรงกับตัวเลือกใดที่ผ่านมาเลยก็ให้ทำงานใน ชุดคำสั่งทีกำหนดไว้ในชุดคำสั่งสุดท้ายแทน

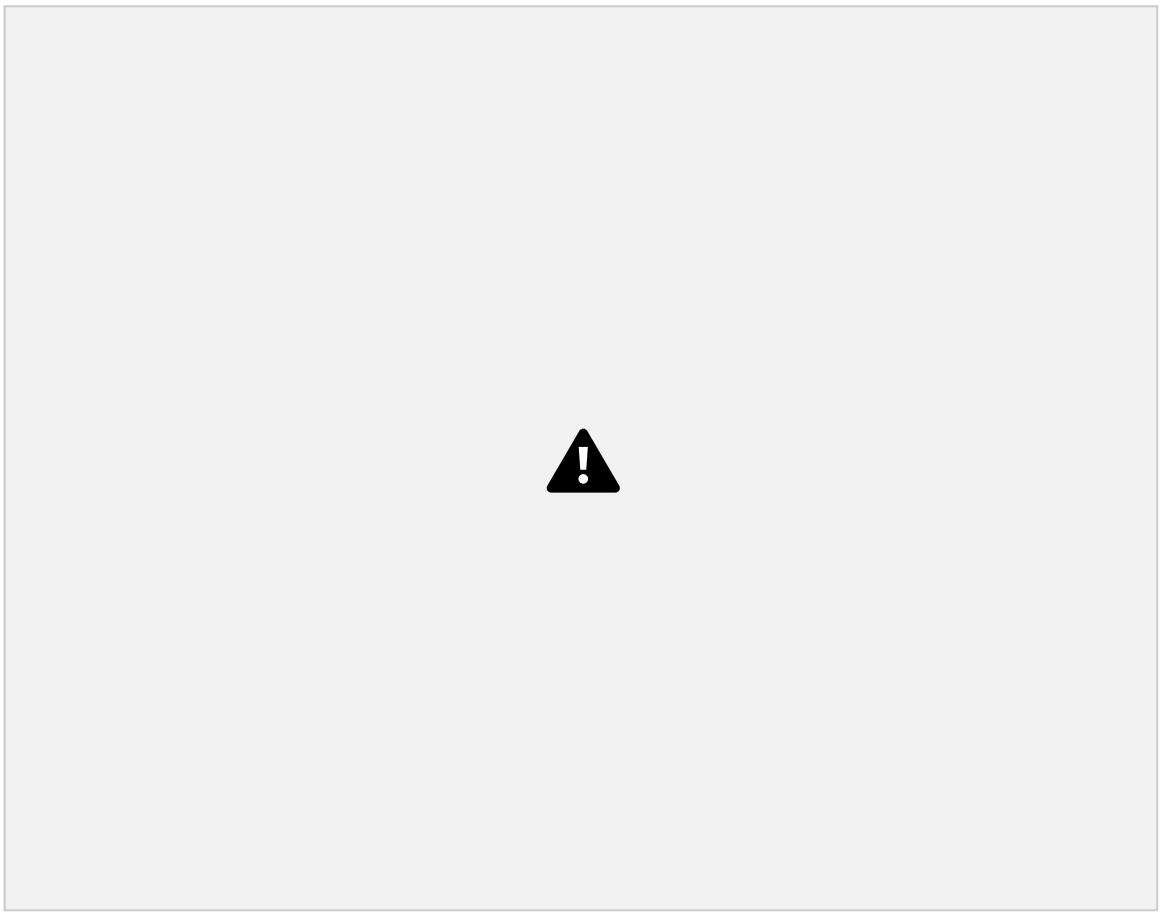
รูปที่ 8.11 การทำงานของเงื่อนไข case ตัวอย่างไฟล์ Ex6



```
echo “Are you Daved? (y/n)” read ans case $ans in
        y|Y) echo “You are in my group.”
                ;;  n|N) echo ”You are in John’s group”
                ;; *) echo ”Please enter y or n only”    ;;
        esac
```

เมื่อเรียกใช้งานเชลล์สคริปต์ Ex6 จะมีผลลัพธ์ดังรูปที่ 8.12

รูปที่ 8.12 การทำงานของเชลล์สคริพต์ไฟล์ Ex6 ตัวอย่าง Ex6 นี้จะแตกต่างกับตัวอย่างที่ผ่าน ๆ มาเล็กน้อยตรงที่ในตัวอย่างจะเป็นตัวอย่างเชลล์สคริพต์ที่มีการโต้ตอบกับผู้ใช้ โดยเชลล์สคริปต์จะมีคำถาม ถามผู้ใช้แล้วให้ผู้ใช้ตอบ ถ้าคำตอบของผู้ใช้ตรงกับตัวเลือกใด ก็จะทำงานในชุดคำสั่งที่กำหนดไว้สำหรับตัวเลือกที่ได้กำหนดไว้ ในเชลล์นั้นจะใช้เครื่องหมาย | เข้ามาช่วยเพื่อให้แต่ละตัวเลือกนั้นมีค่าของตัวเลือกได้หลายค่า



8.7. คำสั่ง select

รูปแบบการใช้

คำสั่ง

```
select name [ in list ] do
        ชุดคำสั่ง
done
```

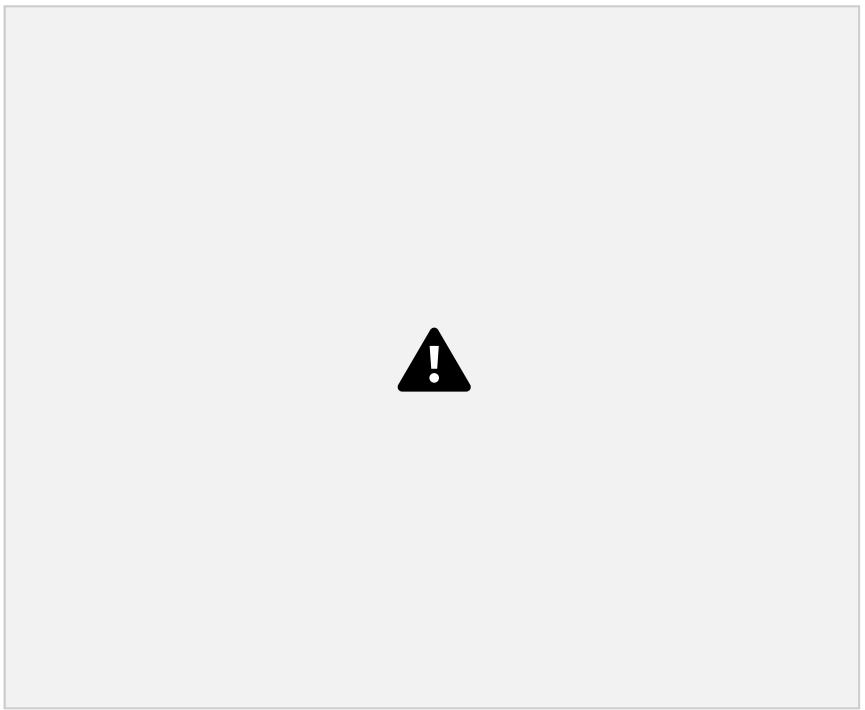
คำสั่ง select **นี้เป็นคำสั่งที่ไม่ค่อยจะมีการนำมาใช้กันเท่ากับคำสั่งที่ผ่านๆมา เพราะคำสั่ง** select **นี้ไม่สามารถใช้ได้กับทุกเชลล์เหมือนกับคำสั่งที่ผ่านมา แต่สามารถใช้ได้เฉพาะกับคอร์นเชลล์และบอร์นอะเชลล์ตั้งแต่เวอร์ชัน** 1.14 **ขึ้นไปเท่านั้น**

โดยที่คำสั่ง select **จะเป็นคำสั่งที่ช่วยในการสร้างเมนูให้กับผู้ใช้ตามค่า**

ของตัวแปร name ที่ให้มา หรือถ้ามีการกำหนด in list มาด้วย ก็จะนำค่าใน list นี้มาสร้างเป็นเมนูโดยจะมีค่าเริ่มตั้งแต่ 1 ไป เรื่อย ๆ เมื่อสร้างเป็นเมนูเรียบร้อยแล้วก็จะขึ้นพร้อมป์ เพื่อรอรับค่าผู้ใช้ต้องการจะเลือก โดยพร้อมป์ทีรอรับค่านี้จะได้มาจากค่าที่ถูกกำหนดไว้ในตัวแปรระบบ PS3 เมื่อผู้ใช้ทำการป้อนค่าตัวเลือกที่ต้องการ แล้ว เชลล์ก็จะนำค่านั้นไปทำในชุดคำสั่งที่ระบุไว้หลัง do จนถึง done

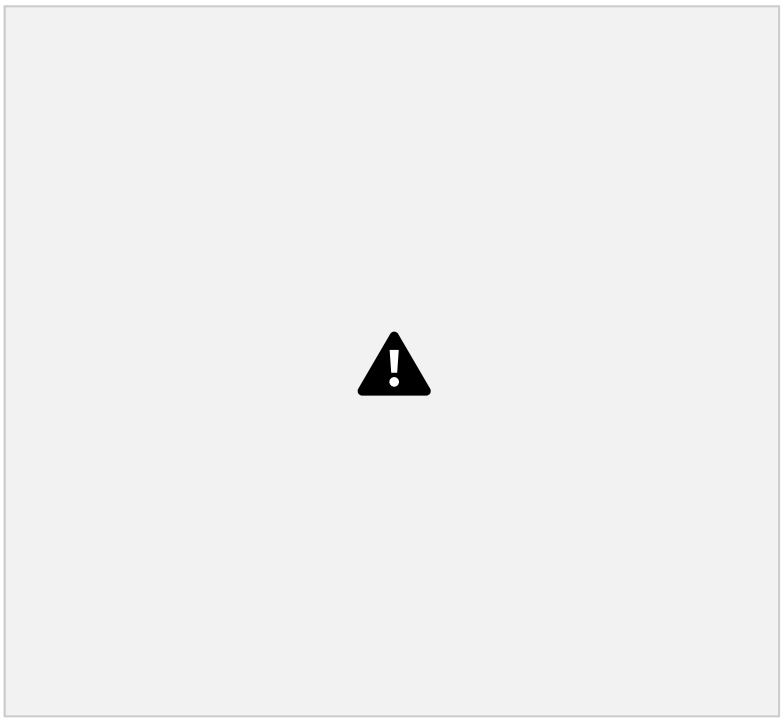
รูปที่ 8.13 การทำงานของคำสั่ง

select ตัวอย่างไฟล์ Ex7



PS3=’directory is ’ direct=”/usr /bin /home” select choice in $direct do
if [$choice]
then echo ”You choose $choice” ls -la $choice break
else echo ’ invalid directory’ fi done

สร้างเชลล์สคริปต์ตามตัวอย่างแล้วตั้งชื่อเชลล์สคริปต์นี้ว่า Ex7 **แล้วลองรันเชลล์สคริปต์นี้ดู จะได้ผลรันเชลล์สคริปต์นี้ดูจะได้ดังรูปที่** 8.14



รูปที่ 8.14 ผลการทำงานของเชลล์สคริพต์ Ex7 สำหรับการทำงานของเชลล์สคริปต์นี้เริ่มแรกเป็นช่วงของการกำหนดค่าโดยค่าของ PS3 จะ เป็นค่าที่จะนำไปใช้เป็นพร้อมป์สำหรับรอรับค่าจากผู้ใช้ หลังจากนั้นก็จะทำการสร้างเมนูตามค่าที่ได้กำหนดไว้ในตัวแปร $direct และก็จะมีพร้อมป์ขึ้นมาเพื่อรอรับค่าจากผู้ใช้ เมื่อผู้ใช้ป้อนตัวเลือกเข้าไป แล้ว ซึ่งถ้าเป็นตัวเลือกที่อยู่ในช่วงที่ถูกต้องแล้ว เชลล์ก็จะนำตัวเลือกไปทำงานกับชุดคำสั่งหลัง do จนถึง done แต่ถ้าผู้ใช้ป้อนค่าตัวเลือกที่ไม่ได้อยู่ในช่วงทีกำหนดแล้วก็จะไปทำคำสั่งหลัง else แทน ส่วนคำสั่ง break ที่อยู่เป็นคำสั่งสั่งสุดท้ายของชุดคำสั่งหลัง do นั้นมีไว้เพื่อให้เชลล์ทำงานคำสั่ง select เพียงครั้งเดียว เพราถ้าไม่มี คำสั่ง break แล้วล่ะก็เมื่อเชลล์ทำงานในชุดคำสั่งหลัง do เรียบร้อยแล้ว ก็จะ กลับไปที่พร้อมป์เพื่อรอรับค่าใหม่อีกครั้ง โดยจะวนลูปอย่างนี้ไปไม่รู้จบ

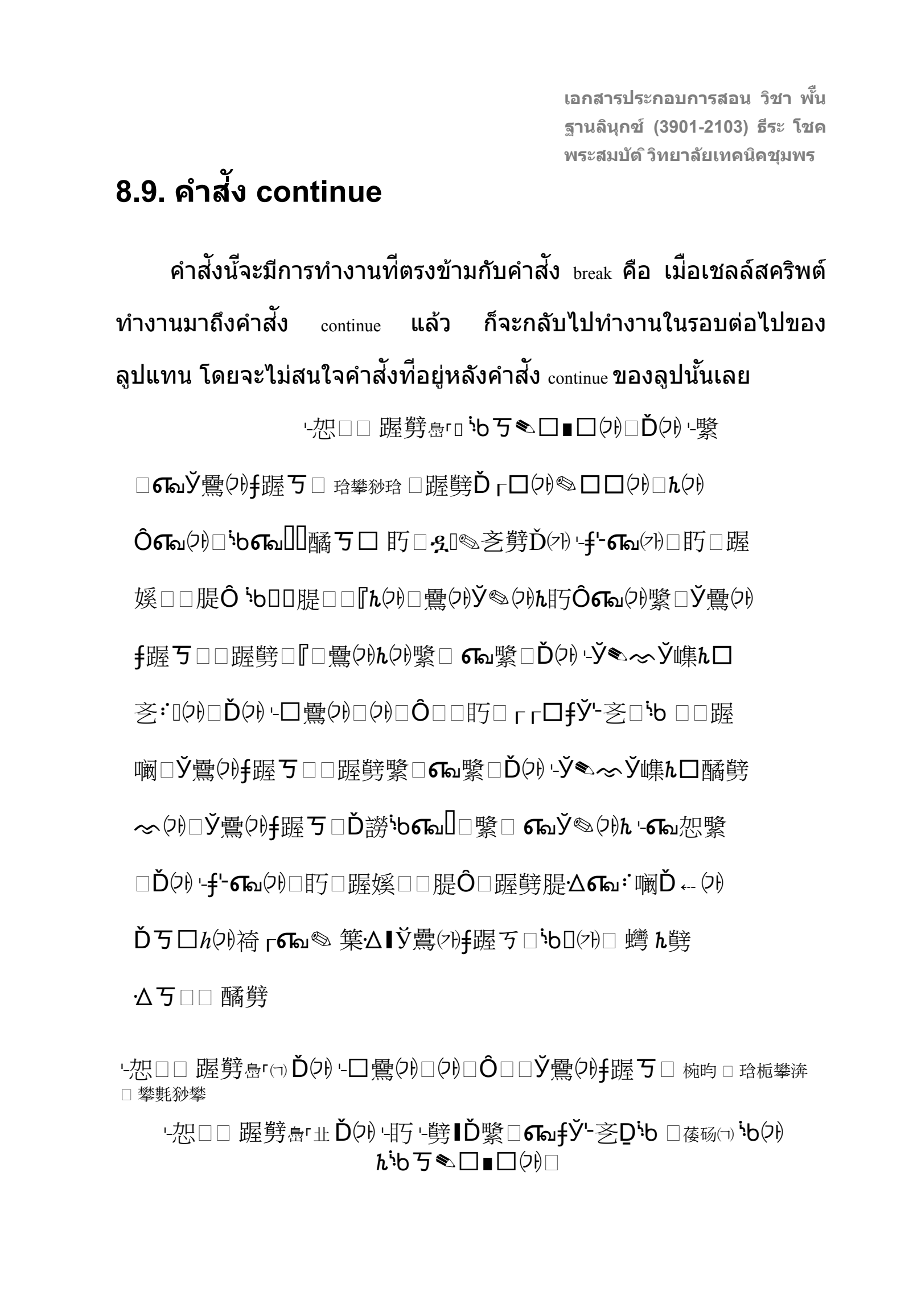
8.8. คำสั่ง break

คำสั่งที่ผ่านมาเหล่านี้ล้วนเป็นคำสั่งที่ทำงานขึ้นอยู่กับเงื่อนไขที่กำหนดไป แต่ยังมีคำสั่งบาง คำสั่งจะใช้ในการควบคุมทิศทางการทำงานของเซลล์สคริพต์ให้โดยไม่ขึ้นกับเงื่อนไข โดยคำสั่งเหล่านี้คือ break และ continue

เมื่อเชลล์สคริปต์ทำงานมาถึงบรรทัดที่มีคำสั่ง break อยู่ก็จะหลุดออกจากลูปที่ทำงานอยู่ทันที ให้ลองสังเกตจากตัวอย่างที่ผ่านมา ในลูปของ select ถ้าเราเอาคำสั่ง break ออก เชลล์สคริปต์ก็จะ ทำงานวนไปเรื่อยๆ ไม่รู้จบ แต่เมื่อใส่คำสั่ง break เข้าไปแล้ว เมื่อเชลล์สคริปต์ทำงานมาถึงบรรทัดนี้ก็จะออกจากลูป select ทันที

8.9. คำสั่ง continue

คำสั่งนี้จะมีการทำงานที่ตรงข้ามกับคำสั่ง break **คือ เมื่อเชลล์สคริพต์ทำงานมาถึงคำสั่ง** continue **แล้ว ก็จะกลับไปทำงานในรอบต่อไปของลูปแทน โดยจะไม่สนใจคำสั่งที่อยู่หลังคำสั่ง** continue **ของลูปนั้นเลย**



ㄣ怨□□ 踝劈蟲ㄏ劇 Ď㇒ ㄣ□鷺㇒□㇒□Ô□□ 明梔椀氈攀

ㄣ怨□□ 踝劈蟲ㄏ嫻 Ď㇒ ㄣ□鷺㇒□㇒□Ô□□ ㄏ怨□

甒湋琀椀氈 ᛃㄎ✎□■□㇒□腥ᾼㄏ □萎砀止

ㄣ怨□□ 踝劈蟲ㄏ蟲 □ ㄏĎ㇒ ㄣ□鷺㇒□㇒□Ô□□盱□ㄏㄏ□ƒў'-乺Ḏᛃb □萎砀止

ㄣ怨□□ 踝劈蟲ㄏ恣 Ď㇒ ㄣ□

鷺㇒□㇒□Ô□□ ㄏ怨□ 昀褰嫲 ᛃㄎ✎□■□㇒□腥

ᾼㄏ □萎砀劘

ㄣ怨□□ 踝劈蟲ㄏ▯ □ ㄏĎ㇒ ㄣ□鷺㇒□㇒□Ô□□盱□ㄏㄏ□ƒў'-乺Ḏᛃb □萎砀劘

ㄣ怨□□ 踝劈蟲ㄏ▯▯ Ď㇒ ㄣ□鷺㇒□㇒□Ô□□盱□踝

嫁□□腥Ô 挀愀狆攀 ᛃㄎ✎□■□㇒□腥ᾼㄏ □萎砀嗥

ㄣ怨□□ 踝劈蟲ㄏ▯㈠ Ď㇒ ㄣ□鷺㇒□㇒□Ô□□盱

□ㄏㄏ□ƒў'-乺Ḏᛃb▯腥ᾼㄏ □萎砀嗥 ᛃㄎ✎□■□㇒□ 萎砀嗥 □醐劈□

『裔ᛃbĎᛃb▯㇒□Ď丐ᨒᛃb丐✎□■□㇒□□踝劈□□㇒□ 蠻 ㄟ㇒盱ㄏ

謗Ď□ᎮᏨ▯▮ᛃb ㄣ□□踝劈繫□ᛃb丐✎□■□㇒□□『盱

□ᦟ▯ᛃb丐✎□■□㇒□盱□ㄏㄏ □ƒў'-乺Ḏᛃb▯▯踝劈ㄟ劈Ď㇒ ㄣ箕

ᛃbᎮᏨᛃb▯ᨒĎ丐ᨒ▯ᎮᏨ怨繫□ ᎮᏨ箕·△▮盱□ㄏㄏ□ƒў'-乺□ᛃb▯□『ㄟ

劈Ў鷺㇒㇐㇒ㄟ ㇐㇒ㄟ□ᎮᏨ怨繫□ᎮᏨ裔 ㄏᎮᏨ✎繫→ᎮᏨ▯ᎮᏨ怨繫

□ᎮᏨᛃb▯ᨒ ㇐ᎮᏨ㇒ㄟЎ鷺㇒ᛃb▯ᨒÔ□□□ᎮᏨ怨繫

□ ᏮᏫᦓᦟ ᠆□Ď ᠻ ෴ᦓᦟ ᠻ✎盯 ┌嫒□Ď繴·Δ Ď謗□『□鱞�hh□�hh□繴□□

嶕·ΔǏ鱞�hhʄ踁 ᠻ □□踁劈Ď鱞�hh→□·Δ腥✎ ᏮᏫʄ鱞�hh

→᠆ ᠻ ෴ᦓᦟ ᠻ✎盯 ┌嫒□Ď□踁劈腥·ΔᏮᏫĎ鱞�hh→□·Δ腥✎ᏮᏫ繴□盯

□ ┌ ┌□□醐 ᠻ□□『繴□ ᏮᏫ盯Ǐ ᠆踁嫒□□→h�hh▌簣 盯ÔᏮᏫ㈦h㈦

□□✎▌盯Ḏ踁嫒□繴→ᏮᏫ祷ᦓ□ ┌『ᦓᦟ ᠻ✎盯 ┌嫒□Ď□醐 ᠻ　 h劈

Ǐ□㈦Ô□□ᦓᦟ ᠻ✎盯 ┌嫒□Ď腥·ΔᏮᏫ→ ┏㈦▌Ǐ□㈦

᠆怨□□ 踁劈蟲┌□ｱﾍﾄ Ď㈦ ᠆□鱞㈦

□㈦□Ô□□Ǐ鱞㈦ʄ踁 ᠻ□ 狆攀氈攀振琀 ᦓᦟ ᠻ✎□■□㈦

□腥ᷝA ┌ □蔆砀婶