บทที่ 9 ตัวอย่างของเชลล์สคริพต์

จุดประสงค์เชิงพฤติกรรม

1. เพื่อให้สามารถอธิบายการทำงานของเชลล์สคริพต์ตัวอย่างได้
2. เพื่อให้สามารถเขียนเชลล์สคริพต์ได้
3. เพื่อให้สามารถประยุกต์ตัวอย่างเชลล์สคริพต์ได้

จากเนื้อหาที่ผ่านมา เราได้ศึกษาการเขียนเชลล์สคริพต์มาแล้ว สำหรับในบทนี้จะได้กล่าวถึง การเขียนสคริพต์ที่ใช้เป็นตัวอย่างในการศึกษาการเขียนเชลล์สคริพต์แบบต่างๆ ดังนี้

9.1. ตัวอย่างการกำหนดค่าเริ่มต้นให้กับผู้ใช้งาน

ในระบบลินุกซ์ที่ใช้กันนั้น เมื่อผู้ใช้ได้ทำการล็อกอินเข้าระบบแล้ว ระบบจะไปอ่านไฟล์สำคัญ ไฟล์หนึ่ง เพื่อใช้ในการกำหนดค่าต่างๆให้กับระบบผู้ใช้คนนั้น เรียกว่าไฟล์ .bash_ profile ฉะนั้นใน ตัวอย่างแรกเราจะได้ทำการศึกษาไฟล์ .bash_ profile ซึ่งไฟล์แรกที่ถูกอ่านเมื่อเมื่อผู้ใช้ได้ล็อกอินสู่ระบบนั้นเอง

รายละเอียดเชลล์สคริปต์ .bash_ profile

```
1. # .bash_profile
2. # Get the aliases and functions
3. if [ -f ~ /.bashrc ]; then
4.  .~ /.bashrc
5. fi
6. # User specific environment and startup programs
7. PATH=$PATH:$HOME/bin
8. ENV=$HOME/.bashrc
9. USERNAME="root"
```

10. export USERNAME ENV PATH
11. mesg n
12. alias xedit =‘xedit –fn thai8x13 &’

ขั้นตอนการทำงานของเชลล์สคริปต์

- **บรรทัดที่ 1-2 เป็นคำอธิบายอย่างสั้นๆ ของสคริพต์นี้**

- บรรทัดที่ 3-4 จะทำการตรวจสอบว่ามีระบบไฟล์ .bashrc อยู่ที่ไดเร็กตอรี่โฮม (home directory) ของผู้ใช้งานที่ล็อกอินเข้ามาหรือไม่ ถ้ามีก็จะทำการเรียกใช้งานไฟล์นั้น
- บรรทัดที่ 7-9 ใน 3 บรรทัดนี้จะเป็นช่วงของการกำหนดค่าตัวแปรภายใน PATH, ENV และ USERNAME ตามลำดับ
- บรรทัดที่ 10 จะทำการ export ตัวแปรที่ภายในได้กำหนดค่ามาให้เป็นตัวแปรกลางหรือตัวแปร สาธารณะ ซึ่งจะทำให้มีการถ่ายทอดค่าของตัวแปรเหล่านี้ไปให้เชลล์ลูกของเชลล์นี้
- บรรทัดที่ 11 การเรียกใช้งานคำสั่ง mesg n เป็นการปฏิเสธการใช้คำสั่ง mesg จะทำให้ผู้อื่นไม่สามารถจะติดต่อสื่อสารกับเราได้ มักจะใช้ในกรณีที่เรากำลังทำงานยุ่งไม่ต้องการให้ผู้อื่นเข้ามารบกวน
- บรรทัดที่ 12 จะเป็นการกำหนดให้การใช้คำสั่ง xedit –fn thai8 x13 & แทนด้วยคำสั้นๆว่า xedit ทำให้เมื่อเรียกใช้งานคำสั่ง xedit ก็จะมีค่าเท่ากับเรียกใช้งานคำสั่ง xedit –fn thai8x13 & นั่นเอง

นอกจากไฟล์ .bash_ profile ที่กล่าวข้างต้นแล้ว ยังมีไฟล์อื่นที่เป็นเชลล์สคริพต์ที่ระบบจะต้อง เรียกใช้งาน เช่น .bash_ logout จะเป็นไฟล์ที่ระบบจะต้องเรียกใช้ก่อนออกจากระบบ หรือ .bash_history จะเป็นไฟล์สำหรับใช้เก็บคำสั่งที่เคยใช้งานมาเพื่อเป็นประวัติ เป็นต้น

9.2. ตัวอย่างการตรวจสอบไฟล์และไดเร็กตอรี่

ตัวอย่างนี้เป็นตัวอย่างที่มีการใช้งานฟังก์ชันเป็นเชลล์สคริพต์ที่ใช้เพื่อการตรวจสอบว่าที่ไดเร็ก ตอรีหนึ่งนั้น มีไฟล์กี่ไฟล์ มีไดเร็กตอรีกี่ไดเร็กตอรี่ และมีอะไรบ้างในไดเร็กตอรี่ โดยเมื่อมีการเรียกใช้งานจะต้องระบุไดเร็กตอรีที่ต้องการทราบรายละเอียดให้เป็นอาร์กิวเมนต์ของเชลล์สคริพต์ แต่ถ้าไม่ได้ระบุอาร์กิวเมนต์ไป เชลล์สคริพต์ก็จะไปหารายละเอียดของไดเร็กตอรีปัจจุบันที่ทำงานอยู่ โดยเชลล์สคริพต์นี้จะมีรายละเอียดดังนี้

รายละเอียดเชลล์สคริปต์ check_file

```
1. function count_di {
2. t =`expr $# - 1`
3. if [ $1 = 'file' ]
4. then
5. echo "Have $t files."
6. else
7. echo "Have $t directories."
8. fi
9. i = 1
10. for di
11. do
12. if [ $1 != 1 ]
13. then
14. j=`expr $i – 1`
15. echo "$j. $di"
16. fi
17. i=`expr $i + 1
18. done
19. }
20. directory =`ls –pA $1 |grep /`
21. count_di dir $directory
22. files=`ls –pA $1 |grep – v /`
23. count_di file $files
```

ขั้นตอนการทำงานของเชลล์สคริปต์

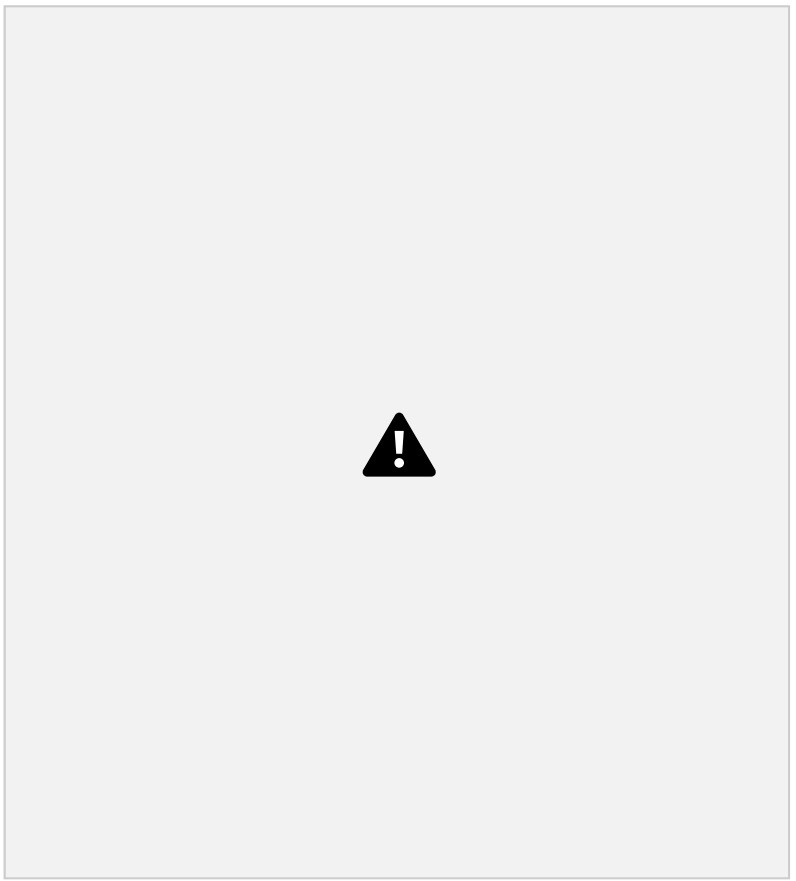
- บรรทัดที่ 1-19 ในช่วง 19 บรรทัดแรกนั้นจะเป็นการกำหนดฟังก์ชัน
- บรรทัดที่ 20 จะเรียกใช้งานคำสั่ง ls –pA $1 |grep / ผลลัพธ์ที่ได้จะเป็นชื่อไดเร็กตอรีทั้งหมดที่มีอยู่ในไดเร็กตอรีที่รับค่าเป็นอาร์กิวเมนต์เข้ามา ปกติถ้าเรียกแค่คำสั่ง ls –pA $1 จะเป็นการเรียกดูรายชื่อไฟล์และไดเร็กตอรีที่อยู่ในเส้นทาง $1 โดยชื่อไฟล์และไดเร็กตอรีที่แสดงนั้น จะแสดงแตกต่าง กันตรงชื่อไดเร็กตอรีจะมีเครื่องหมาย / ตามหลัง ส่วนไฟล์จะไม่มี และเมื่อนำผลลัพธ์ที่ได้จากการใช้คำสั่ง ls – pA $1 มาผ่านเข้าคำสั่ง grep / ระบบก็จะหาค่าที่มีเครื่องหมาย / อยู่เท่านั้น ซึ่งก็จะมีแต่ชื่อไดเร็กตอรีเพียงอย่างเดียว สำหรับค่า $1 นั้นจะเก็บชื่อไดเร็กตอรีที่ต้องการทำการดูรายละเอียด ซึ่งค่านี้จะ เป็นอาร์กิวเมนต์ที่ผู้ใช้ต้องป้อนมาพร้อมกับการเรียกสคริปต์นี้ ส่วนค่าของผลลัพธ์ที่ได้จากการรันคำสั่ง ls –pA $1 | grep / นี้จะเก็บอยู่ในตัวแปร directory
- บรรทัดที่ 21 จะทำการเรียกฟังก์ชัน count_di ซึ่งได้ทำการกำหนดไว้ก่อนแล้วนี้โดยจะมี dir และค่าตัวแปร directory (ซึ่งเป็นรายชื่อไดเร็กตอรีทั้งหมด) ถูกส่งไปเป็นอาร์กิวเมนต์ของฟังก์ชันนี้ด้วย โดยการทำงานของฟังก์ชันจะเป็นดังนี้
- บรรทัดที่ 2 จะเป็นการกำหนดค่าให้กับตัวแปร t ซึ่งเป็นตัวแปรนี้จะเก็บค่าของจำนวนอาร์กิว เมนต์ของฟังก์ชันแล้วลบออกไป 1 ตัว โดยค่า t นี้คือจำนวนไดเร็กตอรีหรือไฟล์ที่ได้จากการเรียกใช้งานคำสั่ง ls –pA $1 |grep / นั่นเอง
- บรรทัดที่ 3 - 8 จะทำการตรวจสอบว่าอาร์กิวเมนต์ตัวแรกที่ส่งมานั้นเป็นคำว่า directory หรือ file โดยถ้าเป็นคำว่า directory ก็จะแสดงข้อความออกทางหน้าจอว่ามีจำนวนไดเร็กตอรีอยู่ t ไดเร็ก ตอรี แต่ถ้าอาร์กิวเมนต์ตัวแรกเป็น file ก็จะแสดงข้อความว่ามีไฟล์อยู่ t ไฟล์แทน
- บรรทัดที่ 10 - 18 จะนำค่าตั้งแต่อาร์กิวเมนต์ตั้งแต่ตัวที่ 2 ไปถึงจนตัวสุดท้ายของฟังก์ชัน มา แสดงออกทางหน้าจอ โดยจะมีตัวเลขลำดับอยู่ด้านหน้าของแต่ละบรรทัดด้วย ซึ่งจริงๆแล้วก็คือการ แสดงชื่อไดเร็กตอรี หรือชื่อไฟล์นั้นเอง โดยหลักการทำงานนั้นลูป for จะรับค่าอาร์กิวเมนต์ของฟังก์ชัน

ทั้งหมดเข้ามาเก็บไว้ในตัวแปรชื่อ di ทีละตัว หลังจากนั้นก็จะเข้าเงื่อนไข if ซึ่งสำหรับอาร์กิวเมนต์ตัว แรกตัวแปร i ยังมีค่าเป็น 1 ก็ยังไม่แสดงผลอะไร แต่พอผ่านไปถึงอาร์กิวเมนต์ตัวที่ 2 ถึงตัวสุดท้าย เมื่อ เข้าเงื่อนไข if แล้ว ตัวแปร i มีค่าไม่เท่ากับ 1 แล้ว เชลล์ก็จะแสดงค่าอาร์กิวเมนต์เหล่านั้นออกมาทีละ บรรทัด

- บรรทัดที่ 22 ในบรรทัดนี้ก็จะคล้ายๆ กับบรรทัดที่ 21 แต่ผลลัพธ์ที่ได้จากการรันคำสั่ง ls –pA $1 |grep –v / จะเป็นชื่อไฟล์ที่อยู่ในไดเร็กตอรี่แทน เพราะผลลัพธ์ที่ได้จากการรันคำสั่ง ls -pA $1 เมื่อ นำมาผ่านคำสั่ง grep –v / ก็จะเป็นการหาบรรทัดที่ไม่มีเครื่องหมาย / ซึ่งก็คือบรรทัดที่เป็นชื่อไฟล์นั่นเอง หลังจากนั้นจึงนำผลลัพธ์ที่เป็นชื่อไฟล์ที่ได้เก็บใว้ในตัวแปรชื่อ files
- บรรทัดที่ 23 จะทำการเรียกฟังก์ชัน count_di โดยจะมี file และค่าที่เก็บอยู่ในตัวแปร file เป็น อาร์กิวเมนต์ของฟังก์ชัน ส่วนการทำงานของฟังก์ชันก็จะเหมือนกับบรรทัด 21

กำหนดให้เชลล์สคริปต์นี้ชื่อ check_file ก็จะเรียกเชลล์สคริปต์ด้วยคำสั่งดังนี้ ./check_fie / จะเป็นการขอดูรายชื่อของไฟล์และไดเร็กตอรีที่เก็บอยู่ในไดเร็กตอรีราก ดังแสดงไว้ในรูปที่ 9.1

รูปที่ 9.1 ผลการรันเชลล์สคริปต์ check _file



9.3. ตัวอย่างการใช้สคริพต์ทดแทนฟังก์ชัน

สำหรับตัวอย่างนี้จะเป็นตัวอย่างของการใช้เงื่อนไข if, case และซับสคริพต์ (subscript) ซึ่งจะ การเรียกเชลล์สคริพต์อื่นมาใช้งานด้วย โดยเชลล์สคริพต์นี้จะเป็นเชลล์สคริพต์ที่ใช้แสดงชื่อวันของ วันนี วันพรุ่งนี และเมื่อวาน

สำหรับตัวอย่างนี้จะประกอบด้วยเชลล์สคริปต์ 2 ไฟล์ที่ใช้งานรวมกัน คือ Show_day และ case_day ดังนี้

รายละเอียดชองเชลล์สคริปต์ Show_day

```
1.  today=`date lcut –d’ ` -f1`
2.  case “$today” in
3.   Mon) a=1;;
4.   Tue) a=2;;
5.   Wed) a=3;;
6.   Thu) a=4;;
7.   Fir) a=5;;
8.   Sat) a=6;;
9.   Sun) a=7;;
10. esac
11. b=`expr $a + 1`
12. if (test $b = 8)
13. then
14. b=1
15. fi
16. c=`expr $a – 1`
17. if (test $c = 0)
18. then
19. c=7
20. fi
21. echo –n “Today is ”
22. ./case_day  $a
23. echo –n “Yesterday is \c”
24. ./case_daty  $c
25. echo-n “Tomorrow is \c”
26. ./case_day $b
```

รายละเอียดของเชลล์สคริปต์ case_day

```
1. case $1 in
2. 1) echo "Monday";;
3. 2) echo "Tuesday";;
4. 3) echo "Wednesday";;
5. 4) echo "Thursday";;
6. 5) echo "Friday";;
7. 6) echo "Saturday";;
8. 7) echo "Sunday";;
9. esac
```

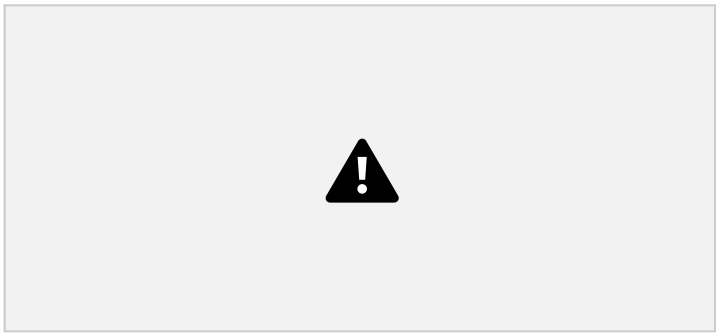
ขั้นตอนการทำงานของเชลล์สคริปต์ Show_day

- บรรทัดที่ 1 จะเป็นการกำหนดค่าให้กับตัวแปรชื่อ today ซึ่งจะเป็นตัวอักษร 3 ตัวแรกของวัน เช่น ถ้าวันนี้เป็นวันจันทร์ ตัวแปร today ก็จะเก็บค่า Mon เป็นต้น โดยค่าที่ได้จะเป็นผลมาจากการเรียก คำสั่ง date |cut-d``-f1 คือ เมื่อเรียกคำสั่ง date ผลที่ได้จะแสดงรายละเอียดของวันและเวลาในขณะที่เรียกคำสั่ง จากนั้นจะนำผลลัพธ์ที่ได้มาผ่านคำสั่ง cut โดยจะเลือกมาเฉพาะฟิลด์ที่ 1 โดยใช้ช่องว่างเป็น ตัวแบ่งซึ่งก็คือชื่อวันนั่นเอง
- • บรรทัดที่ 2 - 10 จะเป็นส่วนของเงื่อนไข case โดยมันจะรับค่าของตัวแปร today เข้ามา เปลี่ยนให้เป็นตัวเลขเพื่อนำไปใช้งานแทน แล้วนำค่าตัวเลขนั้นเก็บไว้ในตัวแปร a
- • บรรทัดที่ 11 จะนำค่าของ a มาเพิ่มอีก 1 แล้วเก็บไว้ในตัวแปร b
- • บรรทัดที่ 12 -16 จะทำการตรวจสอบค่า b ว่าเป็น 8 หรือไม่ ถ้าเป็น 8 ก็จะป้อนค่าใหม่ให้เป็น 1 แทน
- • บรรทัดที่ 17 จะนำค่าของ a มาลดลง 1 ค่า แล้วทำการเก็บไว้ในตัวแปร b
- บรรทัดที่ 18 - 21 จะนำค่าตัวแปร c มาตรวจสอบว่ามีค่าเป็น 0 หรือไม่ โดยถ้าเป็น 0 ก็จะป้อน ค่าให้เป็น 7 แทน
- บรรทัดที่ 21, 23, 25 จะเป็นการเรียกคำสั่ง echo เพื่อให้แสดงข้อความว่า Today is, Yesterday is และ Tomorrow is ออกทางหน้าจอตามลำดับ
- บรรทัดที่ 22, 24, 26 จะทำการเรียกซับสคริปต์ case_day ขึ้นมาใช้งานโดยมีตัวแปร a, c และ b เป็นอาร์กิวเมนต์ตามลำดับ

เชลล์สคริปต์นี้จะมีเพียงเงื่อนไข case เพียงอย่างเดียว โดยจะนำค่าอาร์กิวเมนต์ที่ถูกป้อนให้มา จากเชลล์สคริปต์ Show_day มาตรวจสอบว่าเป็นค่าเท่าไร แล้วแสดงผลออกทางหน้าจอเป็นชื่อวันตาม เงื่อนไขที่ได้กำหนดไว้

เมื่อสร้างเชลล์สคริปต์ทั้ง 2 ไฟล์นี้ตามตัวอย่างแล้ว เรียกเชลล์สคริปต์ Show_day โดยไม่ต้องมีอาร์กิวเมนต์อะไรเลย ผลลัพธ์ที่ได้จะเป็นดังรูปด้านล่าง

รูปที่ 9.2 ผลการรันเชลล์สคริปต์ Show_day



9.4. ตัวอย่างการค้นหาไฟล์และคำที่ต้องการ

เป็นตัวอย่างของการใช้เงื่อนไข if ธรรมดา แต่จะเป็นเงื่อนไข if ในลักษณะมี if ซ้อนกันหลาย ชั้น ฉะนั้นในกรณีที่ต้องการจะสร้างเชลล์สคริปต์ที่มีลูปหรือเงื่อนไขซ้อนกันหลายชั้นนั้น สิ่งที่ควร คำนึงถึงนอกจากหลักการเขียนโปรแกรมปกติโดยทั่วไปคือ การจัดระเบียบในการเขียนโปรแกรมของลูปต่างๆ เพื่อให้สามารถแก้ไขสคริพต์ในภายหลังได้อย่างไม่สับสนว่าคำสั่งนี้อยู่ในลูปไหน เงื่อนไขใด สำหรับในเชลล์สคริปต์นี้จะช่วยในการหาคำที่ต้องการในไฟล์ที่ระบุ โดยผู้ใช้จะต้องระบุอาร์กิวเมนต์ให้กับเชลล์สคริปต์ 2 ตัว คือ ชื่อไฟล์และคำที่ต้องการค้นหาตามลำดับ

รายละเอียดของเชลล์สคริปต์ search

```
1. if [ $# = 0 -o $# = 1 ]
2. then
3. echo "Usage: search filename word..."
4. else
5. filename=$1
6. word1=$2
7. if [ -r $filename ]
8. then
```

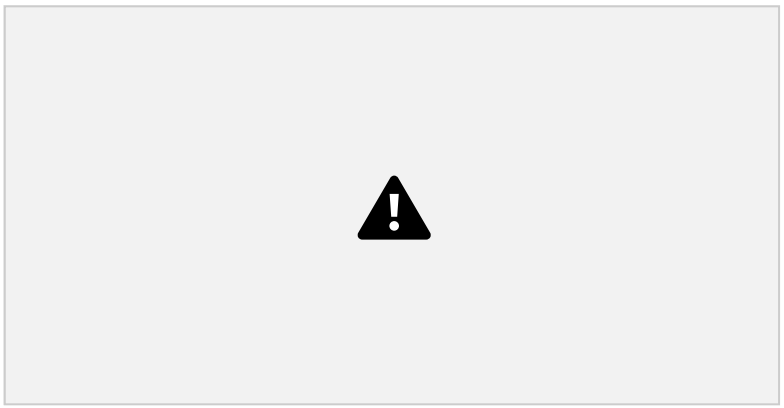
```
9.  if grep –n $word1 $filename > /tmp/search
10.   then
11. echo "$word1 is in $filename."
12.   echo "ln line:"
13. cat /tmp/search
14. rm /tmp/search
15. fi
16. else echo "$filename not exist !!!"
17. fi
18. fi
```

ขั้นตอนการทำงานของเชลล์สคริปต์

- บรรทัดที่ 1-3 จะเป็นเงื่อนไข if ชั้นนอกสุด โดยเงื่อนไขนี้จะทำการตรวจสอบว่าผู้ใช้อาร์กิว เมนต์เข้ามาถึง 2 ตัวหรือไม่ถ้าป้อนมาไม่ถึง 2 ตัว ก็จะแสดงข้อความเพื่อบอกวิธีใช้เชลล์สคริปต์นี้
- บรรทัดที่ 5-6 แต่ถ้าผู้ใช้ป้อนอาร์กิวเมนต์เข้ามาถึง 2 ตัว (อาจจะมากกว่า 2 ตัวก็ได้แต่จะใช้แค่ 2 ตัวเท่านั้น ก็จะทำการป้อนค่าอาร์กิวเมนต์ตัวแรกให้กับตัวแปรชื่อ filename และตัวที่ 2 ให้กับชื่อตัว แปรชื่อ word1
- บรรทัดที่ 7-8 จะเป็นเงื่อนไข if ชั้นที่ 2 โดยในเงื่อนไข if ชั้นนี้จะทำการตรวจสอบชื่อไฟล์ที่เก็บอยู่ในตัวแปรชื่อ filename ว่ามีอยู่ในระบบจริงหรือไม่ โดยชื่อไฟล์ที่ผู้ใช้จะต้องทำการระบุตำแหน่ง ของไฟล์มาให้ถูกต้อง ถ้าผู้ใช้ไม่ได้ระบุตำแหน่งระบบจะถือว่าเป็นตำแหน่งปัจจุบันขณะที่เรียกใช้เชลล์สคริปต์
- บรรทัดที่ 9-10 เพื่อทำการตรวจสอบเงื่อนไขที่แล้ว ปรากฏว่ามีไฟล์ที่ระบุอยู่จริงก็จะเข้าสู่เงื่อนไข if ชั้นสุดท้าย โดยจะทำการเรียกใช้งานคำสั่ง grep –n $word1 $filename>/tmp/search เพื่อให้คำสั่งนี้จะทำการค้นหาคำที่ต้องการในไฟล์ที่ระบุ แล้วเก็บผลลัพธ์ที่ได้ชื่อ serch ที่เส้นทาง /tmp
- • บรรทัดที่ 11-14 ถ้าในเงื่อนไขที่แล้วสามารถเรียกใช้งานคำสั่งได้ก็จะแสดงข้อความที่ต้องการ อยู่ในไฟล์ที่ระบุจริง และแสดงคำว่า In line: ตามลำดับหลังจากนั้นก็จะแสดงข้อมูลที่เก็บอยู่ในไฟล์ search ที่เส้นทาง /tmp ซึ่งเก็บผลลัพธ์ที่ได้จากการเรียกใช้งานคำสั่ง grep –n $word1 $filename > /tmp/search และสุดท้ายเมื่อแสดงข้อความในไฟล์นี้เรียบร้อยก็จะลบคำสั่งนี้ทิ้ง
- • บรรทัดที่ 15 ปิดเงื่อนไข if ชั้นในสุด
- • บรรทัดที่ 16 ในเงื่อนไข if ชั้นที่ 2 ที่ทำการตรวจสอบชื่อไฟล์ที่เก็บข้อมูลอยู่ในตัวแปรชื่อ filename นั้น ถ้าไฟล์นี้ไม่มีอยู่ในระบบจริง จะแสดงข้อความออกมาว่าไฟล์นั้นไม่มีอยู่ในระบบ
- บรรทัดที่ 17 ปิดเงื่อนไข if ชั้นที่ 2
- บรรทัดที่ 18 ปิดเงื่อนไข if ชั้นแรก

เชลล์สคริปต์นี้มีชื่อว่า search สามารถจะทำการเรียกใช้งานเชลล์สคริปต์เช่นถ้าต้องการหาคำ ว่า case ในไฟล์ชื่อ Show_day ก็ให้เรียกใช้งานคำสั่ง search Show_day case เชลล์สคริปต์ก็จะทำการ ค้นหาคำว่า case ในไฟล์ชื่อ Show_day แล้วแสดงออกมาทางหน้าจอพร้อมระบุหมายเลขบรรทัด ดัง ตัวอย่างในรูปที่ 9.3

รูปที่ 9.3 ผลการเรียกใช้งานเชลล์สคริปต์ชื่อ search



9.5. ตัวอย่างการสร้างไดเร็กตอรี่

ตัวอย่างนี้เป็นตัวอย่างสั้นๆ ที่ใช้ในการสร้างไดเร็กตอรี่ ซึ่งปกติถ้าเราจะสร้างไดเร็กตอรี่นั้นเรา จะคำสั่ง mkdir แต่คำสั่งนี้จะสร้างไดเร็กตอรี่ทีละชั้นเท่านั้น เช่น ถ้าเราต้องการใช้คำสั่ง mkdir เพื่อสร้าง ไดเร็กตอรี่ /etc/mylinux/textbook ไดเร็กตอรี่ textbook จะถูกสร้างได้ก็ต่อเมื่อไดเร็กตอรี่ mylinux ถูก สร้างไว้แล้วใน /etc เสียก่อนแล้ว แต่ถ้ายังไม่มีไดเร็กตอรี่ mylinux ใน /etc ไดเร็กตอรี่ textbook ก็จะไม่สามารถถูกสร้างได้ แต่เชลล์สคริปต์ในตัวอย่างนี้จะสามารถสร้างไดเร็กตอรี่ที่มีจำนวนชั้นได้หลายชั้น

รายละเอียดของเชลล์สคริปต์ makedir

```
1.  if [ $# !=1 ]
2.  then
3.  echo "Usage: makedir directory..."
4.   exit 1
5.  fi
6.  if [-d $1]
7.  then exit 0
8.  else
9.  makedir `dirname $1`
10.  mkdir $1
11. fi
```

ขั้นตอนการทำงานของเชลล์สคริปต์

• บรรทัดที่ 1-5 จะเป็นเงื่อนไข if เพื่อทำการตรวจสอบจำนวนอาร์กิวเมนต์ที่ถูกป้อนเข้ามา เนื่องจากเชลล์สคริปต์นี้ใช้สำหรับสร้างไดเร็กตอรี่ผู้ใช้จะต้องระบุชื่อไดเร็กตอรี่เป็นอาร์กิวเมนต์เข้ามา ให้กับเชลล์สคริปต์ โดยถ้า

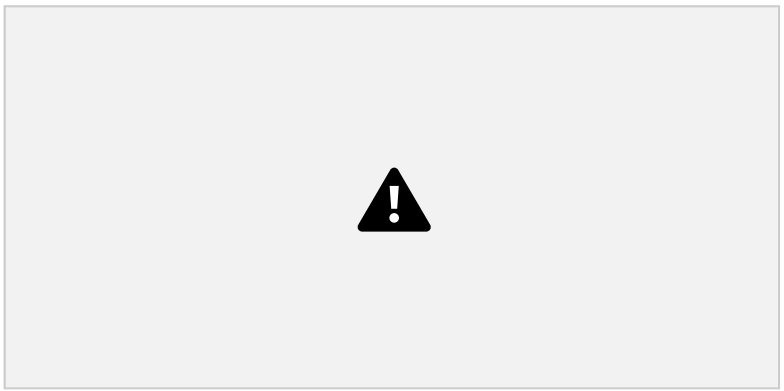
จำนวนอาร์กิวเมนต์ไม่เท่ากับ 1 ก็จะแสดงข้อความเพื่อบอกวิธีใช้เชลล์สคริปต์นี้แล้วออกเชลล์สคิปต์โดยใช้คำสั่ง exit 1 ซึ่งหมายถึงให้ออกจากเชลล์สคริปต์แล้วค่าที่ถูก ส่งออกไปเป็น 1 นั่นคือถ้าหลุดออกจากเชลล์สคริปต์นี้แล้วลองใช้คำสั่ง echo $? ดูจะได้ค่า 1 ออกมา

นั่นหมายความว่าคำสั่งที่เพิ่งทำมาล่าสุดมีข้อผิดพลาด ถ้าในเชลล์สคริปต์ใช้เพียงคำสั่ง exit อย่างเดียว ก็จะหลุดออกจากเชลล์สคริปต์เช่นกัน แต่ถ้าเราใช้คำสั่ง echo $? ตรวจสอบจะได้ค่า 0 ออกมาแทนซึ่งไม่ถูกต้อง

- บรรทัดที่ 6-7 จะเป็นเงื่อนไข if อีกเงื่อนไขหนึ่ง โดยถ้าผู้ใช้ป้อนจำนวนอาร์กิวเมนต์เข้ามา 1 ตัวก็จะผ่านเข้ามาสู่เงื่อนไขนี้ได้ สำหรับเงื่อนไขนี้จะทำการตรวจสอบว่าไดเร็กตอรี่ที่ต้องการสร้างนั้นมีอยู่แล้วหรือยัง ถ้ามีแล้วก็จะออกจากเชลล์สคริปต์แล้วส่งค่าออกไปเป็น 0 ซึ่งหมายถึงเชลล์สคริปต์นี้สามารถทำงานได้ถูกต้องไม่มีข้อผิดพลาด
- • บรรทัดที่ 8-9 ถ้าปรากฏว่าไดเร็กตอรี่ที่ระบุยังไม่ถูกสร้าง ก็จะการเรียกใช้งานเชลล์สคริปต์นี้อีกครั้ง โดยมีผลลัพธ์จากการเรียกคำสั่ง dirname $1 เป็นอาร์กิวเมนต์สำหรับคำสั่ง dirname นั้นจะใช้สำหรับทำการตัดไดเร็กตอรี่ที่อยู่หลังสุดออกไป เช่น ถ้าผู้ใช้ระบุอาร์กิวเมนต์ที่ต้องการสร้างเป็น /etc/mylinux/textbook แล้วใช้คำสั่ง dirname /etc/mylinux/textbook ก็จะได้ผลเป็น /etc/mylinux แทน
- บรรทัดที่ 10 จะทำการสร้างไดเร็กตอรี่ที่เป็นอาร์กิวเมนต์เข้ามา
- บรรทัดที่ 11 ปิดเงื่อนไข if

เชลล์สคริปต์ที่สร้างตามตัวอย่าง จะมีชื่อเชลล์สคริปต์ว่า makedir เมื่อเราเรียกใช้งานเชลล์สคริปต์จะต้องระบุชื่อไดเร็กตอรี่ที่ต้องการสร้างเป็นอาร์กิวเมนต์เมื่อเรียกใช้งานเสร็จเรียบร้อยให้ลองใช้คำสั่ง ls แสดงรายชื่อไฟล์และไดเร็กตอรี่ดูเพื่อตรวจสอบว่าเชลล์สคริปต์นี้ทำการสร้างไดเร็กตอรีที่ระบุได้แล้วหรือยัง

รูปที่ 9.4 ผลการรันเชลล์สคริปต์ makedir



9.6. ตัวอย่างการสร้างเมนูเพื่อเลือกใช้งานโปรแกรม

ตัวอย่างนี้จะเป็นการนำเชลล์สคริปต์ในตัวอย่างตั้งแต่ตัวอย่างที่ 3 ถึงตัวอย่างที่ 5 มาสร้างเป็น เชลล์สคริปต์เดียว โดยมีเมนูให้เลือกว่าต้องการทำอะไร เช่น ถ้าเลือก 1 ก็จะแสดงชื่อวัน (ตามตัวอย่างที่ 3) เลือก 2 ก็จะทำการค้นหาคำที่ต้องการในไฟล์ที่ระบุ (ตามตัวอย่างที่ 4) หรือเลือก 3 ก็จะทำการสร้าง ไดเร็กตอรีที่ระบุ (ตามตัวอย่างที่ 5) โดยทุกอย่างจะต้องรวมอยู่ในไฟล์เดียวกัน และจะต้องทำการสร้าง

การทำงานในแต่ละตัวเลือกเป็นฟังก์ชัน ไม่ใช้การเรียกจากเชลล์สคริปต์ที่มีอยู่ฉะนั้นสิ่งที่ต้องทำคือ สร้างเมนูเพื่อรอรับค่าจากผู้ใช้และแก้ไขเชลล์สคริปต์ในตัวอย่างมาทำเป็นฟังก์ชันเพื่อรวมอยู่ในไฟล์เดียวกัน นอกจากนี้อาจจะต้องแก้ไขเชลล์สคริปต์เดิมบางส่วนเพื่อทำงานได้เหมาะสมยิ่งขึ้นด้วย

นอกจากจะนำความสามารถของเชลล์สคริปต์ตั้งแต่ตัวอย่างที่ 3 ถึง 5 มาสร้างเป็นตัวเลือกให้ผู้ใช้ได้เลือกทำงานแล้ว ยังจะเพิ่มตัวเลือกอีก 1 ตัวเลือกเพื่อออกจากเชลล์สคริปต์เมื่อผู้ใช้ทำงานเสร็จ ในแต่ละตัวเลือกแล้วก็จะสามารถเลือกได้ว่าจะทำงานต่อหรือไม่และบางส่วนของเชลล์สคริปต์เดิมที่ต้องการระบุอาร์กิวเมนต์ให้กับเชลล์สคริปต์ก็จะเปลี่ยนเป็นการถามแล้วรอรับคำตอบจากผู้ใช้แทน

รายละเอียดของเชลล์สคริปต์ menu_exam

```
1.  function case_day {
2.  case $1 in
3.  1) echo "Monday";;
4.  2) echo "Tuesday";;
5.  3) echo "Wednesday";;
6.  4) echo "Thursday";;
7.  5) echo "Firday";;
8.  6) echo "Saturday";;
9.  7) echo "Sunday";;
10. esac
11. }
12. function show_day {
13. today=`date |cut –d' ` -f1`
14. case "$today" in
15.  Mon) a=1;;
16.  Tue) a=2;;
17.  Wed) a=3;;
18.  Thu) a=4;;
19.  Fir) a=5;;
20.  Sat) a=6;;
21.  Sun) a=7;;
22. esac
23. b=`expr $a + 1 `
24. if (test $b = 8)
```

```
25. then b=1
26. fi
27. c=`expr $a - 1`
28. if (test $c = 0)
29. then c=7
30. fi
31. echo –n "Today is "
32. case_day $a
33. echo –n "Yesterday is "
34. case_day $c
35. echo –n "Tomorrow is "
36. case_day $b
37. }
38. function search {
39. if [ -r $filename ]
40. then
41. if grep –n $word $filename > /tmp/search
42.  then
43. echo "$word is in $filename."
44. echo –n "In line"
45. cat /tmp/search
46. rm /tmp/search
47.  fi
48. else
49. echo "$filename not exist !!!"
50.  exit 1
51. fi
52. }
53. function makedir {
54. if ! [ -d $1 ]
55. then
56. makedir `dirname $1`
57.  mkdir $1
58. fi
59. }
60. function check_cont {
61. echo –n "Do you want to continue  this shell script (y/n) :"
62. read ans
63. case "$ans" in
64.  y|Y) menu_exam
65.  ;;
66. n|N) exit 0
```

```
 ;;
 *) check_cont
 ;;
esac
}
clear
echo –n ”
################################################
## ##
##              Example Shell Script          ##
## ##
################################################
## ##
## 1. Show Day ##
## ##
## 2. Search ##
## ##
##          3.Make Directory    ##
## ##
## 4. Exit ##
## ##
################################################

Please select the choice from menu : “
read choice
case “$choice”  in
1) echo “Your choice is 1”
  echo
  show_day
 check_cont
 ;;
2) echo “Your choice is 2”
  echo
echo –n “Enter the word for search :”
  read word
while [ -z $word ]
  do
       echo “Invalid word for search”
echo –n “Enter the word for search :”
       read word
  done
echo –n “Enter the name of file which you want :”
  read filename
while [ -z $filename ]
  do
 echo “Invalid file’s name”
       echo –n “Enter the name of file which you want :”
 read filename
  done
  echo
search  $filename  $word
 check_cont
 ;;
3) echo “Your choice is 3”
 echo –n “Enter the name of directory which you want to create :”
  read direct
while [-z $direct ]
  do
     echo “Invalid directory ‘s name”
```

```
37.       echo –n "Enter the name of directory which you want create :"
38.   read direct
39.    done
40.   makedir $direct
41.   check_cont
42.   ;;
43.    4) exit 0
44.   ;;
45.    *) menu_exam
46.   ;;
47.  esac
```

ขั้นตอนการทำงานของเชลล์คริพต์ menu_exam

- **บรรทัดที่** 1-71 **จะเป็นช่วงของการกำหนดฟังก์ชันให้กับเชลล์สคริพต์**
- **บรรทัดที่** 72 **เป็นการลบหรือล้างหน้าจอทั้งหมด**

• บรรทัดที่ 73-90 จะเป็นการสร้างเมนูแบบปกติ โดยใช้คำสั่ง echo ธรรมดา จึงทำให้ไม่มีเทคนิค มากนัก

• บรรทัดที่ 91 จะเป็นคำสั่งที่รอรับค่าตัวเลือกจากผู้ใช้แล้วเก็บลงตัวแปรชื่อ choice

• บรรทัดที่ 92 เข้าสู่เงื่อนไข case โดยนำเอาตัวเลือกที่ผู้ใช้ได้เลือกไว้มาใช้

• บรรทัดที่ 93-94 ถ้าผู้ใช้เลือกตัวเลือกที่ 1 จะแสดงข้อความออกทางหน้าจอ เพื่อบอกว่าผู้ใช้ได้เลือกใช้ตัวที่ 1 ไป

• บรรทัดที่ 95 ใช้เรียกฟังก์ชัน show_day ขึ้นมา ซึ่งเป็นฟังก์ชันที่ได้ถูกกำหนดไว้ในตอนต้น (บรรทัดที่ 12-37) โดยฟังก์ชัน show_day นี้ก็จะมีการเรียกใช้ฟังก์ชัน case_day (บรรทัดที่ 1-11) เหมือนกับตอนที่อยู่ในรูปแบบของเชลล์สคริพต์เช่นกัน

• บรรทัดที่ 96 ใช้เรียกฟังก์ชัน check_cont ที่ได้ถูกกำหนดไว้ในบรรทัดที่ 60-71 ซึ่งฟังก์ชันนี้จะเป็นฟังก์ชันที่ใช้สอบถามผู้ใช้เมื่อทำงานเสร็จงานหนึ่งว่าต้องการจะทำงานต่อหรือไม่ ถ้าทำงานต่อก็จะกลับไปหน้าเมนูเพื่อให้ผู้ใช้เลือกการทำงานอีกครั้ง แต่ถ้าเลือกไม่ทำงานต่อออกจากเชลล์สคริพต์

• บรรทัดที่ 98-99 เมื่อผู้ใช้เลือกตัวที่ 2 คือการค้นหาคำที่ต้องการในเชลล์สคริพต์ที่ระบุก็จะ แสดงข้อความว่าผู้ใช้ได้เลือกตัวที่ 2

• บรรทัดที่ 100-101 เนื่องจากฟังก์ชัน search ต้องมีการระบุอาร์กิวเมนต์ให้กับเชลล์สคริพต์ใน 2 บรรทัดนี้ จึงต้องมีการถามที่ผู้ใช้ต้องการค้นหาแล้วรอรับคำตอบเพื่อเก็บไว้ในตัวแปรชื่อ word

• บรรทัดที่ 102-107 เป็นลูป while เพื่อตรวจสอบค่าที่เก็บอยู่ในตัวแปรชื่อ word ว่าเป็น null หรือไม่ ซึ่งถ้าเป็น null ก็จะรอรับค่าใหม่จนกระทั่งไม่เป็น null

• บรรทัดที่ 108-109 จะเป็นการถามชื่อไฟล์ที่ต้องการใช้ค้นหาและรอรับคำตอบเพื่อเก็บไว้ใน ตัวแปรชื่อ filename

• บรรทัดที่ 110-115 จะเป็นลูป while อีกเช่นกันเพื่อตรวจสอบค่าที่เก็บอยู่ในตัวแปร filename ว่าเป็น null หรือไม่ ซึ่งถ้าเป็น null ก็จะรอรับค่าใหม่จนกระทั่งไม่เป็น null

• • บรรทัดที่ 117 จะเรียกฟังก์ชัน search ที่ได้ระบุไว้ในบรรทัดที่ 38-52 ขึ้นมาใช้ ซึ่งในฟังก์ชัน นี้ได้ทำการตัดในส่วนของการตรวจสอบจำนวนอาร์กิวเมนต์ของสคริพต์เดิมออก เนื่องจากเชลล์สคริปต์เป็นผู้กำหนดค่าอาร์กิวเมนต์เองอยู่แล้ว ผู้ใช้ไม่ได้เป็นผู้กำหนดจึงไม่มีความจำเป็นต้องตรวจสอบอีก โดยจะมีค่าที่เก็บอยู่ในตัวแปร filename และตัวแปร word เป็นอาร์กิวเมนต์ตามลำดับ

• • บรรทัดที่ 118 เรียกฟังก์ชัน check_cont ขึ้นมาเพื่อสอบถามว่าผู้ใช้ต้องการจำทำงานต่อหรือไม่

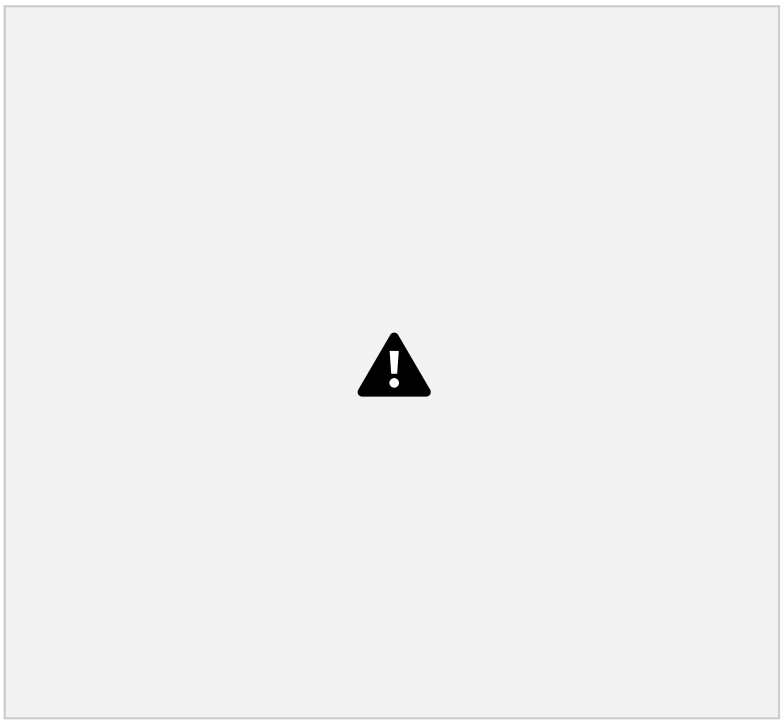
- • บรรทัดที่ 120 ถ้าผู้ใช้เลือกตัวที่ 3 เพื่อสร้างไดเร็กตอรี่ก็จะมีการแสดงข้อความว่า ผู้ใช้ได้ตัวเลือกที่ 3
- • บรรทัดที่ 121-122 จะมีการรอรับชื่อไดเร็กตอรีที่ต้องการสร้าง แล้วเก็บในตัวแปรชื่อ direct
- บรรทัดที่ 123-128 จะเป็นลูป while ที่ทำการตรวจสอบค่าที่เก็บอยู่ในตัวแปร direct ว่าเป็น null หรือไม่ ซึ่งถ้าเป็น null ก็จะใส่ค่าใหม่จนกว่าจะไม่เป็น null

- บรรทัดที่ 129 จะเป็นการเรียกฟังก์ชัน makedir ที่ได้กำหนดไว้ในบรรทัดที่ 53-59 ขึ้นมาใช้งานโดยมีค่าที่เก็บอยู่ในตัวแปร direct เป็นอาร์กิวเมนต์ และในฟังก์ชัน makedir นี้ได้ทำการตัดส่วนที่ใช้ตรวจสอบอาร์กิวเมนต์ออกเช่นกัน เพราะเชลล์สคริปต์จะเป็นผู้กำหนดจำนวนอาร์กิวเมนต์ให้กับ ฟังก์ชันอยู่แล้ว จึงไม่จำเป็นต้องมีการตรวจสอบอีกว่าจำนวนอาร์กิวเมนต์จะเป็น 1 หรือไม่
- บรรทัดที่ 130 เรียกฟังก์ชัน check_cont เพื่อสอบถามผู้ใช้ว่าต้องการทำงานในเชลล์สคริพต์นี้ ต่อหรือไม่
- บรรทัดที่ 132-133 ถ้าผู้ใช้เลือกตัวเลือกที่ 4 ก็จะออกจากการทำงานของเชลล์สคริพต์ พร้อม กับส่งค่า 0 ออกไปด้วย เพื่อบอกว่าผู้ใช้สามารถทำงานในเชลล์สคริพต์นี้ได้ถูกต้องไม่มีข้อผิดพลาด
- บรรทัดที่ 134-135 ถ้าผู้ใช้เลือกตัวเลือกอื่น นอกจาก 1-4 แล้วก็เรียกเชลล์สคริพต์ menu_exam ใหม่เพื่อให้ผู้ใช้เลือกใหม่จนกว่าจะเลือกตัวเลือกได้ถูกต้อง

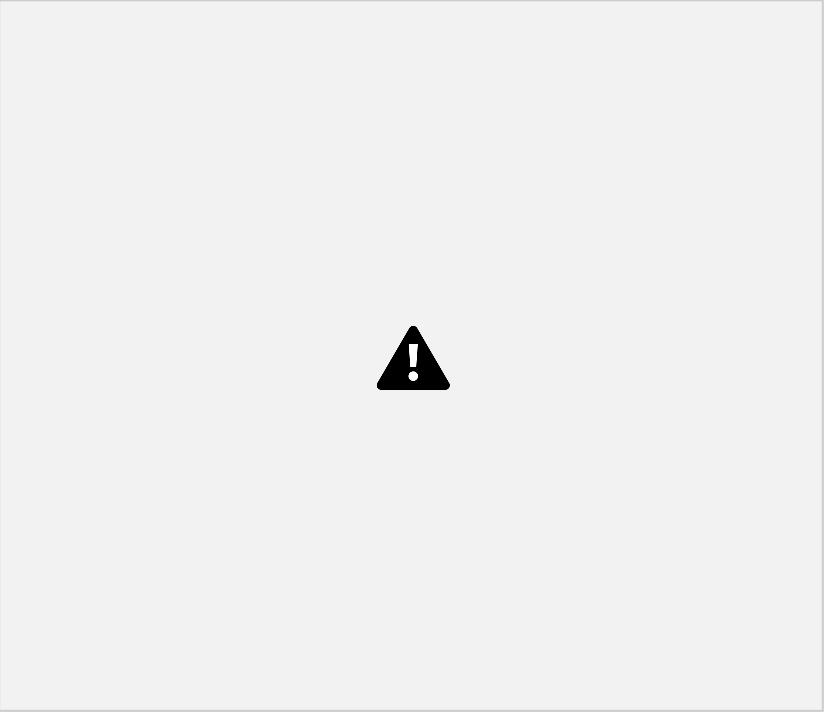
สำหรับการเรียกใช้เชลล์สคริปต์นี้ ผู้ใช้สามารถใช้ได้ 2 แบบ ดังที่ได้เคยกล่าวมาแล้วในบท ต้นๆ คือสามารถเรียกใช้งานแบบสร้างเชลล์สคริพต์ขึ้นมาใหม่หรือทับเชลล์สคริพต์เดิมก็ได้ โดยถ้า เรียกใช้แบบทับเชลล์สคริพต์เดิมเมื่อผู้ใช้เลือกตัวเลือกที่ 4 เพื่อออกจากเชลล์สคริพต์ก็จะเป็นการออก จากเชลล์สคริพต์ปัจจุบันที่เรียกเชลล์สคริปต์ด้วย ซึ่งถ้าเชลล์ปัจจุบันที่เรียกสคริพต์เป็นเชลล์แรกที่เรียกใช้ก็จะเป็นการออกจากระบบเลยทันที

ในตัวอย่างนี้ให้จะเรียกใช้งานแบบสร้างเชลล์สคริปต์ขึ้นมาใหม่ โดยใช้คำสั่ง ./menu_exam ซึ่ง จะได้ผลการทำงานดังรูปที่ 9.5 และ 9.6

รูปที่ 9.5 ผลการรันเชลล์สคริปต์ menu_exam



รูปที่ 9.6 ผลการรันเชลล์สคริปต์ menu_exam แล้วมีการเลือกตัวเลือกที่2



╙恕□□ 蹀艻▯▯▯ □ ╓Ď㈎ ╌╌ㄎ□肟□ ╓╓□ɟŸ╌乭□ᵎb □振栀攀振欀 开昀椀甀攀

╙恕□□ 蹀艻▯㈠ □ ╓Ď㈎ ╌╌ㄎ□肟□ ╓╓□ɟŸ╌乭□ᵎb □匀栀漀瀀开挽愀礀

╙恕□□ 蹀艻▯▯ □ ╓Ď㈎ ╙肟 ╌艻▮Ď繄□ดิ□㈎□肟□ ╓╓□ɟŸ╌乭
□ᵎb□□蹀媄□ 猀攀愀瀀振栀

╙恕□□ 蹀艻▯▯ □ ╓Ď㈎ ╌╌ㄎ□肟□ ╓╓□ɟŸ╌乭□ᵎb □渀愀欀攀挽洀

╙恕□□ 蹀艻▯▯ □ ╓Ď㈎ ╌╌ㄎ□肟□ ╓╓□ɟŸ╌乭□ᵎb □渀攀渀琀开攀琀愀渀

╙恕□□ 蹀艻▯▯ □ ╓Ď㈎ ╌╌ㄎ□肟□ ╓╓□ɟŸ╌乭□ᵎb □渀攀渀琀开攀琀愀渀
裿 ╓ดิ✎ℎ艻Ď㈎ ╙肟 ╓媄□Ďᵎbㄎ✎肟 ╓媄□Ď□蹀艻㈠