# บทที่ 5

## เทคนิคการใช้สภาพแวดล้อมของระบบยูนิกซ์ ในการพัฒนาระบบการประมวลผลข้อมูลทางธุรกิจ

ในบทนี้จะแสดงให้เห็นถึงวิธีการใช้คำสั่งงานบนระบบยูนิกซ์ แสดงเทคนิคต่าง ๆ ที่จะนำมาใช้ในการพัฒนาระบบการประมวลผลข้อมูลทางธุรกิจ ซึ่งแต่ละคำสั่งงานก็จะเหมาะกับการใช้งานที่แตกต่างกันออกไป ตลอดจนแสดงให้เห็นการใช้ทางเลือกต่าง ๆ ของคำสั่งงานแต่ละคำสั่ง และการใช้เครื่องหมายพิเศษต่าง ๆ การใช้คำสั่งงานและเทคนิคต่าง ๆ จะแยกตามการใช้งานได้ดังต่อไปนี้

### การสร้างเมนู

การสร้างเมนูใช้วิธีการแสดงข้อความต่าง ๆ ที่จอภาพ โดยใช้คำสั่ง echo หรือคำสั่ง cat การใช้คำสั่ง cat และเครื่องหมาย "<<" เพื่อใช้แสดงข้อความที่จอภาพจะต้องมีคำเพื่อบอกจุดจบของข้อความที่จะแสดง ดังตัวอย่างเป็นการแสดงข้อความบนจอภาพของเมนูหลักและใช้คำว่า "SCREEN" เป็นตัวบอกจุดจบของข้อความ

```
cat << SCREEN
        MAIN MENU

        1. DATA ENTRY
        2. REPORT
        3. UPDATE
        4. EXIT

        Please enter number or first character:
SCREEN
```

การรับคำสั่งงานจากผู้ใช้

การรับคำสั่งงานจากผู้ใช้จะใช้คำสั่ง read และใช้คำสั่ง case ตรวจสอบคำสั่งที่ผู้ใช้ป้อนเข้ามา จุดเด่นของคำสั่ง case คือสามารถตรวจสอบข้อมูลได้หลายรูปแบบโดยสามารถใช้เครื่องหมาย "*" แทนอักขระใด ๆ กี่ตัวก็ได้ เครื่องหมาย "?" แทนอักขระใด ๆ 1 ตัว เครื่องหมาย "[...]" แทนกลุ่มอักขระและเครื่องหมาย "|" แทนตัวดำเนินการ "หรือ" ( or) ดังตัวอย่าง

```
read ANSWER
case $ANSWER in
        1|D|d)   dataJentry ;;
        2|R|r)   report ;;
        3|U|u)   update ;;
        4|X|x)   exit ;;
        *)       echo "Invalid command"
esac
```

จากตัวอย่าง คำสั่ง case สามารถตรวจสอบคำสั่งที่ผู้ใช้ป้อนเข้ามาได้ว่าเป็นตัวเลข ตัวอักษรตัวใหญ่หรือตัวอักษรตัวเล็ก โดยใช้เครื่องหมาย "|" เป็นตัวแบ่ง ส่วนเครื่องหมาย "*" แทนตัวอักษรอะไรก็ได้นอกเหนือจากที่ผ่านการตรวจสอบมาแล้ว

การเก็บข้อมูลในแฟ้มข้อมูล

การเก็บข้อมูลในแฟ้มข้อมูลจะเก็บในลักษณะกระแสของไบต์ติดต่อกัน ( stream of byte) มีการแบ่งแยกแต่ละระเบียนด้วยรหัสขึ้นบรรทัดใหม่ (newline) และมีการแบ่งแยกแต่ละเขตข้อมูลด้วยอักขระที่ผู้ใช้กำหนดขึ้นมา ถ้าผู้ใช้ไม่กำหนดอักขระแบ่งแยกเขตข้อมูลใหม่ ระบบจะกำหนดตัวแยกเขตข้อมูลเป็นรหัสเว้นวรรค (blank) รหัสย่อหน้า (tab) หรือรหัสขึ้นบรรทัดใหม่ ในกรณีศึกษานี้จะใช้เครื่องหมาย ":" เป็นตัวแยกข้อมูลแต่ละเขตข้อมูล เนื่องจากว่าเครื่องหมาย ":" ไม่เป็นอักขระพิเศษของคำสั่งต่าง ๆ และเป็นอักขระที่ไม่ใช้เป็นข้อมูลทั่ว ๆ ไป ดังตัวอย่างของแฟ้มข้อมูลสินค้าคงคลัง เช่น

```
011001:NEW FAB FAMILY:4:530.00:477.00:SALE 10 %
011002:NEW SUPER JUMBO:6:560.00:560.00:
```

ตามตัวอย่าง แฟ้มข้อมูลประกอบด้วยเขตข้อมูล 6 เขตข้อมูล แต่ละเขตข้อมูลจะมีเครื่องหมาย ":" เป็นตัวแบ่งแยก กรณีที่เขตข้อมูลใดไม่มีข้อมูลก็จะต้องมีเครื่องหมาย ":" คั่นไว้เพื่อให้ข้อมูลเขตข้อมูลถัดไปอยู่ในตำแหน่งที่ถูกต้อง

การค้นหาหมายเลขระเบียนของข้อมูลที่กำหนด

การค้นหาหมายเลขระเบียนของข้อมูลที่กำหนด เนื่องเป็นการกำหนดตัวชี้ระเบียน (record pointer) ซึ่งจะใช้ในการดึงข้อมูลมาทำการประมวลผลต่อไป จะใช้คำสั่ง sed (stream editor) เช่น ถ้าต้องการหาหมายเลขระเบียนของข้อมูลที่ระบุโดยเขตข้อมูลที่ 1 มีค่าตามตัวแปรชื่อ batch_no จากแฟ้มข้อมูลชื่อ batc_file จะใช้คำสั่งดังนี้

```
REC_POI=`sed -n "/^${batch_no}:/=" batch_file`
```

เครื่องหมาย "=" ในคำสั่ง sed เป็นตัวบอกหมายเลขระเบียนที่ตรงกับข้อมูลในตัวแปรชื่อ batch_no ทางเลือก "-n" เป็นการบอกคำสั่ง sed ให้ไม่ต้องแสดงรายละเอียดข้อมูลแต่ละระเบียนออกมา เครื่องหมาย "^" เป็นการระบุจุดเริ่มต้นของระเบียนนั้น

การดึงข้อมูลตามตัวชี้ระเบียนที่กำหนด

การดึงข้อมูลโดยระบุตัวชี้ระเบียนจะใช้คำสั่ง tail และคำสั่ง sed ดังตัวอย่าง

```
tail +$REC_POI $INPUT_FILE | sed 1q > $OUTPUT_FILE
```

คำสั่ง tail +$REC_POI จะดึงข้อมูลจากแฟ้มข้อมูลซึ่งมีชื่อเก็บอยู่ในตัวแปร INPUT_FILE ตั้งแต่ระเบียนที่กำหนดโดยตัวแปร REC_POI จากนั้นก็จะส่งข้อมูลไปเป็นข้อมูลรับเข้าของคำสั่ง sed 1q ซึ่งจะคัดเอามาเฉพาะระเบียนแรกเท่านั้น ก็จะได้ข้อมูลที่ต้องการ

การแยกข้อมูลแต่ละเขตข้อมูลโดยมีอักขระแยกเขตข้อมูลตามที่กำหนด

การแยกข้อมูลออกมาแต่ละเขตข้อมูลจะต้องมีการกำหนดอักขระแยกเขตข้อมูลให้ตรงกับอักขระแยกเขตข้อมูลของข้อมูล จากนั้นใช้คำสั่ง set เพื่อแยกข้อมูลแต่ละเขตข้อมูล โดยกำหนดให้พารามิเตอร์ตำแหน่ง (positional parameter) $1 เก็บข้อมูลเขตข้อมูลที่ 1 $2 เก็บข้อมูลเขตข้อมูลที่ 2 เป็นเช่นนี้ไปเรื่อย ๆ ดังตัวอย่างต่อไปนี้

```
IFS=':'
set - `tail +$REC]POI $INPUT]FILE | sed 1q ` ''
BATCH]NO=$1
TRAN]DATE=$2
SALEMAN=$3
```

จากตัวอย่าง มีการกำหนดอักขระแยกเขตข้อมูลเป็นเครื่องหมาย ":" โดยกำหนดให้กับตัวแปรมาตรฐานของระบบยูนิกซ์ชื่อ IFS ใช้คำสั่ง tail และคำสั่ง sed ดึงข้อมูลจากแฟ้มข้อมูลขึ้นมาส่งให้คำสั่ง set เพื่อแยกข้อมูลออกมาเก็บในพารามิเตอร์ตำแหน่งแต่ละตัว แล้วก็จะนำค่าในพารามิเตอร์ตำแหน่งไปเก็บในตัวแปรที่กำหนด

การค้นหาข้อมูลในแฟ้มข้อมูล

การค้นหาข้อมูลในแฟ้มข้อมูลเพื่อนำข้อมูลมาใช้ในการประมวลผล หรือเป็นการตรวจสอบความซ้ำซ้อนของข้อมูล สามารถใช้คำสั่ง grep โดยสามารถระบุรูปแบบของข้อมูลที่ต้องการจะค้นได้หลายรูปแบบ ดังตัวอย่างเช่น

```
grep "^[^:]*:[^:]*:${DOC]NO}:" $INPUT]FILE
```

จากตัวอย่างเป็นการค้นหาข้อมูลในแฟ้มข้อมูล โดยเขตข้อมูลที่ 3 มีค่าตามตัวแปร DOC]NO และแต่ละเขตข้อมูลแยกด้วยเครื่องหมาย ":" เครื่องหมาย "^" ตัวแรกเป็นการบอกจุดเริ่มต้นของข้อมูล ส่วนเครื่องหมาย "^" ตัวอื่น ๆ ที่อยู่ภายในเครื่องหมาย "[...]" แทนตัวดำเนินการ "ไม่ใช่" (not) และเครื่องหมาย "*" แทนอักขระที่อยู่ข้างหน้ากี่ตัวก็ได้

การเพิ่มข้อมูลในแฟ้มข้อมูล

การเพิ่มข้อมูลในแฟ้มข้อมูลสามารถทำได้โดยใช้เครื่องหมาย ">>" ซึ่งเป็นการเบี่ยงเบนส่งออกข้อมูลไปต่อท้ายแฟ้มข้อมูลที่ระบุ ถ้าแฟ้มข้อมูลที่ระบุยังไม่มี ระบบยูนิกซ์ก็จะสร้างแฟ้มข้อมูลนั้นขึ้นมาโดยอัตโนมัติ ดังตัวอย่างเช่น

```
echo "$TRAN_DATE:$BATCH_NO:$SALEMAN" >> $INPUT_FILE
```

การลบข้อมูลในแฟ้มข้อมูล

การลบระเบียนในแฟ้มข้อมูลใช้คำสั่ง sed โดยการระบุข้อมูลของระเบียนที่ต้องการจะลบ มีรูปแบบของคำสั่งคือ sed '/รูปแบบของข้อมูลที่ต้องการลบ/d' ดังตัวอย่าง

```
sed "/^[^:]*:[^:]*:${DOC_NO}:/d" $INPUT_FILE > $OUTPUT_FILE
```

จากตัวอย่างเป็นการลบระเบียนจากแฟ้มข้อมูล INPUT_FILE ซึ่งเขตข้อมูลที่ 3 มีค่าตามตัวแปรชื่อ DOC_NO ตัวอักษร d ในคำสั่ง sed เป็นการระบุการลบข้อมูล

การแก้ไขข้อมูลในแฟ้มข้อมูล

การแก้ไขข้อมูลในแฟ้มข้อมูลใช้คำสั่ง sed โดยการแทนข้อมูลเก่าด้วยข้อมูลใหม่มีรูปแบบของคำสั่งคือ sed 's/ข้อมูลเก่า/ข้อมูลใหม่/' ดังตัวอย่าง

```
sed "s/^$TRAN_DATE:$BATCH_NO/$tran_date:$batch_no/"
        $INPUT_FILE > $OUTPUT_FILE
```

จากตัวอย่างเป็นการเปลี่ยนข้อมูลในแฟ้มข้อมูล โดยเปลี่ยนเฉพาะระเบียนที่มีค่าในเขตข้อมูลที่ 1 และ 2 ตรงกับค่าในตัวแปรที่กำหนด ($TRAN_DATE และ $BATCH_NO) ตัวอักษร s ในคำสั่ง sed เป็นการระบุการเปลี่ยนข้อมูล

การคำนวณตัวเลข

การคำนวณตัวเลขใช้คำสั่ง expr bc และ awk ตัวอย่างเช่น

```
NUM=`expr $NUM + 1`
AMOUNT=`echo "scale=2; (($TOTUNIT * $PURCJPRT) / $CONTENER )"
       | bc`
TOTAL=`awk '{ total += $3 } END { print total }' $INPUTJFILE`
TOTALNUM=`awk ' $1 ~ /'$ENDJDATE'/ { totnum += $2 }
                END { print totnum }' $INPUTJFILE`
```

ตัวอย่างแรกเป็นการคำนวณเพิ่มค่าตัวแปรชื่อ NUM โดยใช้คำสั่งงาน expr

ตัวอย่างที่สองเป็นการคำนวณโดยใช้คำสั่ง bc โดยนำค่าในตัวแปร TOTUNIT คูณกับค่าในตัวแปร PURCJPRT และนำไปหารกับค่าในตัวแปร CONTENER และกำหนดให้มีจุดทศนิยม 2 หลัก (scale=2)

ตัวอย่างที่สามเป็นการคำนวณหาค่าผลรวมของค่าในเขตข้อมูลที่ 2 จากแฟ้มข้อมูลซึ่งมีชื่อเก็บอยู่ในตัวแปร INPUTJFILE โดยใช้คำสั่ง awk

ตัวอย่างสุดท้ายเป็นการคำนวณโดยใช้คำสั่ง awk เช่นกันแต่หาค่าผลรวมของค่าในเขตข้อมูลที่ 2 เฉพาะระเบียนซึ่งค่าในเขตข้อมูลที่ 1 มีค่าตรงกับค่าในตัวแปร ENDJDATE เท่านั้น

การคำนวณโดยใช้คำสั่ง expr สามารถระบุนิพจน์ (expression) และตัวดำเนินการ (operator) ได้เลย การคำนวณโดยใช้คำสั่ง bc จะต้องมีการส่งค่านิพจน์ไปทางหน่วยรับเข้ามาตรฐานโดยใช้คำสั่ง echo ส่วนการคำนวณโดยใช้คำสั่ง awk จะเป็นการอ่านนิพจน์จากแฟ้มข้อมูลซึ่งเหมาะกับการคำนวณข้อมูลหลาย ๆ ชุด

การพิมพ์ตัวเลขในลักษณะ comma (,)

การพิมพ์ตัวเลขในลักษณะ comma ใช้หลักการแทนตัวเลขที่มีรูปแบบตัวเลข 4 ตัวตามด้วยเครื่องหมาย "." หรือ "," ด้วยตัวเลขตัวแรกตามด้วยเครื่องหมาย "," และตามด้วยตัวเลขอีก 3 ตัว การแทนค่าตัวเลขจะใช้คำสั่ง sed ดังตัวอย่าง

```
while :
do
        case $number in
          *[0-9][0-9][0-9][0-9][.,]*)
                number='echo $number |
                sed 's/\(.\)\(...\)\([\.,]\)/\1,\2\3'' ;;
          *)    break ;;
        esac
done
```

การเรียงลำดับข้อมูล

การเรียงลำดับข้อมูลใช้คำสั่ง sort โดยการระบุอักขระแยกเขตข้อมูลและตำแหน่งเขตข้อมูลหรือบอกช่วงของเขตข้อมูล เพื่อใช้เป็นกุญแจ (key) ในการเรียงลำดับ ตัวอย่างเช่น

```
sort -t: +0n $INPUT_FILE
sort -t: +0n -2 $INPUT_FILE
```

จากตัวอย่างเป็นการเรียงลำดับข้อมูลตามเขตข้อมูลที่ 1 และเรียงข้อมูลตามช่วงของเขตข้อมูลที่ 1 ถึงเขตข้อมูลที่ 3 โดยมีการเรียงแบบตัวเลข ทางเลือก "-t" เป็นการกำหนดอักขระแยกเขตข้อมูลของคำสั่ง sort ซึ่งกำหนดเป็นเครื่องหมาย ":"

การเตรียมแฟ้มข้อมูลเพื่อออกรายงาน

การเตรียมแฟ้มข้อมูลเพื่อออกรายงาน เป็นการเลือกเฉพาะระเบียนที่ต้องการจะออกรายงาน หรือมีการสร้างแฟ้มข้อมูลใหม่โดยเลือกเฉพาะเขตข้อมูลที่ต้องการจากหลาย ๆ แฟ้มข้อมูล การทำเช่นนี้เพื่อให้การออกรายงานกระทำได้ง่ายขึ้น ตัวอย่างเช่น

```
awk -F: '$1 <= '$END_DATE' { print $0 }' $IN_FILE > $OUT_FILE
```

ตัวอย่างข้างบนเป็นการเลือกเฉพาะระเบียนที่มีเขตข้อมูลที่ 1 น้อยกว่าหรือเท่ากับค่า

ในตัวแปรชื่อ END]DATE ทางเลือก "-F" เป็นการกำหนดอักขระแยกเขตข้อมูลของคำสั่ง awk ซึ่งในกรณีนี้กำหนดเป็นเครื่องหมาย ":"

```
VISITSTORE=`awk -F: '$3 ~ /'$SALEMAN'/ { visitsum += $6 }
                    END { print visitsum }' $IN]FILE`
```

จากตัวอย่างข้างบนเป็นการคำนวณหาผลรวมของค่าในเขตข้อมูลที่ 6 โดยมีการคำนวณเฉพาะระเบียนที่มีเขตข้อมูลที่ 3 มีค่าตามตัวแปรชื่อ SALEMAN

```
STORETOT=`awk -F: '$4 ~ /'$SALEMAN'/ { print $5 }' $IN]FILE |
        sort | uniq | wc -l`
```

จากตัวอย่างข้างบน คำสั่ง awk จะนิมน์ค่าของเขตข้อมูลที่ 5 เฉพาะระเบียนที่มีเขตข้อมูลที่ 4 มีค่าตามตัวแปรชื่อ SALEMAN จากนั้นส่งข้อมูลไปเรียงลำดับโดยใช้คำสั่ง sort ส่งข้อมูลไปยังคำสั่ง uniq เพื่อดึงเฉพาะข้อมูลที่ไม่ซ้ำกัน และส่งไปนับจำนวนของบรรทัดโดยใช้คำสั่ง wc และใช้ทางเลือก "-l"

```
while read เขตข้อมูล1 เขตข้อมูล2 เขตข้อมูล3
do
    set `sed -n "/^$เขตข้อมูล2:/p" $IN]FILE1
    echo "$เขตข้อมูล1:$5:$6" >> $OUT]FILE
done < $IN]FILE2
```

จากตัวอย่างข้างบนเป็นการสร้างแฟ้มข้อมูลขึ้นมาใหม่โดยการนำเขตข้อมูลที่ 2 ของแฟ้มข้อมูล IN]FILE2 เป็นตัวระบุข้อมูลเริ่มต้นของระเบียนของแฟ้มข้อมูล IN]FILE1 จากนั้นเอาเขตข้อมูลที่ 5 และ 6 ของแฟ้มข้อมูล IN]FILE1 มารวมกับเขตข้อมูลที่ 1 ของแฟ้มข้อมูล IN]FILE2 ได้เป็นแฟ้มข้อมูลใหม่

```
awk -F: '$1 ~ /'$END]DATE'/ && $7 !~ /0/
                { print $0 }' $IN]FILE > $OUT]FILE
```

จากตัวอย่างเป็นการเลือกข้อมูลเฉพาะระเบียนที่มีเขตข้อมูลที่ 1 ตรงกับค่าในตัวแปรชื่อ END]DATE และเขตข้อมูลที่ 7 ไม่เท่ากับ 0

การออกรายงาน

การออกรายงานใช้คำสั่ง awk ซึ่งมีรูปแบบการออกรายงานดังนี้

```
awk 'BEGIN { พิมพ์หัวรายงาน }
           { เลือกข้อมูล คำนวณและพิมพ์รายละเอียด }
      END  { พิมพ์ผลรวมและสรุป }' $IN]FILE
```

ตัวอย่างการออกรายงานโดยใช้คำสั่ง awk

```
awk 'BEGIN { print "SUM OF SALE TYPE 1 OR 2" }
     $1 ~ /1/ { TOTAL1 += $1 ; print $0 }
     $1 ~ /2/ { TOTAL2 += $1 ; print $0 }
     END { print "Total" TOTAL1 , TOTAL2 }' $IN]FILE
```

จากตัวอย่างเป็นการคำนวณค่า 2 ค่า คือหาผลรวมของของระเบียนที่มีเขตข้อมูลที่ 1 เป็น 1 และหาผลรวมของระเบียนที่มีเขตข้อมูลที่ 1 เป็น 2

การเปลี่ยนรูปแบบของคำสั่งงานในชุดคำสั่ง

การเปลี่ยนรูปแบบของคำสั่งงานในชุดคำสั่ง กระทำเพื่อปรับเปลี่ยนชุดคำสั่งให้สามารถปฏิบัติการบนระบบยูนิกซ์อื่น ๆ ได้ ในกรณีศึกษานี้เป็นการเปลี่ยนคำสั่งงานจาก echo -n เป็น echo \c และเปลี่ยนจาก echo \c เป็น echo -n การเปลี่ยนชุดคำสั่งใช้คำสั่ง sed ดังตัวอย่าง

```
sed 's/\(echo\)\( *\)\(\".*\)\(\\c.*\"\)/\1 -n \3\"/' $1
sed 's/\(echo\)\( *\)\(-n\)\( *\)\(\".*\)\(\"\)/\1 \5\\c\6/' $1
```

ตัวอย่างแรกเป็นการเปลี่ยนชุดคำสั่งจาก echo \c เป็น echo -n
ตัวอย่างที่สองเป็นการเปลี่ยนชุดคำสั่งจาก echo -n เป็น echo \c