บทที่ 3

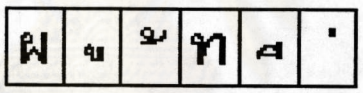
เคิร์สไลบรารี่กับภาษาไทย

ในบทนี้จะกล่าวถึงเคิร์สไลบรารีที่พัฒนาขึ้น ซึ่งประกอบด้วยฟังก์ชั่นเดิมของเคิร์ส และ ฟังก์ชั่นใหม่ที่จะรองรับการทำงานแบบ 2 มิติ

3.1 ข้อจำกัดของเคิร์สในการทำงานกับภาษาไทย

ก) ฟังก์ชันบางฟังก์ชันของเคิร์สไม่รองรับภาษาไทย เนื่องจากเป็นฟังก์ชันที่รับตัวอักขระ 7 บิท

ข) ภาษาไทยเป็นภาษาที่มีตัวอักขระอยู่ในระดับต่างๆกันในบรรทัด คือมีการแสดงผลใน 2 มิติ ในขณะที่ลักษณะการเก็บข้อมูลที่ใช้ในการแสดงผลของเคิร์สเป็นแบบ 1 มิติ คือจะเก็บข้อมูลเป็นสายอักขระ (string) ในแนวนอน ดังรูป 3.1



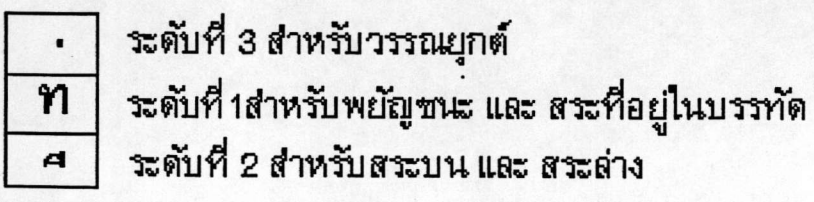
รูปที่ 3.1 การเก็บข้อมูลเป็นสายอักขระ

ลักษณะการเก็บข้อมูลเช่นนี้ ไม่สัมพันธ์กับการแสดงผลภาษาไทย 3 ระดับ ความยาวของข้อมูลและการนับคอลัมน์ของข้อมูลในบัฟเฟอร์ไม่ตรงกับการแสดงผลบนจอภาพ ทำให้แทนที่จะแสดงผลได้ 80 คอลัมน์กลับแสดงผลได้เพียง 80 ตัวอักขระ

3.2 การปรับปรุงให้เคิร์สรู้จักภาษาไทย

ปัญหาการแสดงผลได้น้อยกว่า 80 คอลัมน์แก้ไขได้โดยเก็บข้อมูลให้สัมพันธ์กับการแสดงผล คือ ใช้โครงสร้างข้อมูลแบบ 2 มิติ ซึ่งมีลักษณะการทำงานดังต่อไปนี้

ก) เก็บข้อมูล 3 ระดับใน 1 คอลัมน์



รูปที่ 3.2 การเก็บข้อมูล 3 ระดับใน 1 คอลัมน์

ในการพัฒนา กำหนดให้แต่ละคอลัมน์ ในบัฟเฟอร์สำหรับการแสดงผล มีโครงสร้างข้อมูลดังนี้

```
typedef struct
{
        /* one column has 3 levels of charactor. */
        int     alpha,
                vowel,
                tonal;
        short   attr;
} __LDATA;
```

โดยต้องมีการจัดลำดับตัวอักขระ เพื่อให้เคิร์สและโปรแกรมเลียนแบบเทอร์มินัล มีตำแหน่งสอดคล้องกัน โดยรายละเอียดจะได้กล่าวถึงต่อไป

ข) การเคลื่อนที่ของตัวชี้ตำแหน่ง จะเคลื่อนที่ตามคอลัมน์ ที่กำหนดในข้อ ก)

ค) การเคลื่อนที่ของตัวชี้ตำแหน่งไปยังคอลัมน์ถัดไป จะเกิดขึ้นเมื่อมีการเขียนตัวอักขระที่เป็นพยัญชนะหรือสระกลางลงในบัฟเฟอร์ และจะลบตัวอักขระในระดับอื่นๆของคอลัมน์นั้น ข้อกำหนดนี้ใช้ในการ implement ฟังก์ชันต่อไปนี้

ฟังก์ชัน int waddalpha(WINDOW *win, int ch)
คำอธิบาย เพิ่มตัวอักขระกลาง ch ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win และกำหนด attribute ของคอลัมน์ในเท่ากับ current attribute ของวินโดว์
ผลลัพธ์ 0 = เพิ่มสำเร็จ
1 = มีข้อผิดพลาด

ฟังก์ชัน int waddch(WINDOW win, int ch)
คำอธิบาย เพิ่มตัวอักขระ ch ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win โดยเรียกใช้ waddalpha กรณีที่ ch เป็นตัวอักขระ เรียกใช้ waddvowel กรณีที่ ch เป็นสระบนหรือสระล่าง เรียกใช้ waddtonal กรณีที่ ch เป็นวรรณยุกต์
ผลลัพธ์ 0 = เพิ่มสำเร็จ
1 = มีข้อผิดพลาด

ฟังก์ชัน int waddstr(WINDOW win, char *str)
คำอธิบาย เพิ่ม str ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win โดยเรียกใช้ waddch
ผลลัพธ์ 0 = เพิ่มสำเร็จ
1 = มีข้อผิดพลาด

ฟังก์ชัน int wechochar(WINDOW win, int ch)
คำอธิบาย เพิ่มตัวอักขระ ch ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win

โดยเรียกใช้ waddch และทำการ refresh จอภาพทันที

ผลลัพธ์ 1 เสมอ

ฟังก์ชัน int winsch(WINDOW win, int ch)

คำอธิบาย แทรก ch ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win โดยเรียกใช้ waddch และทำการ refresh จอภาพทันที

ผลลัพธ์ 0 = เพิ่มสำเร็จ
1 = มีข้อผิดพลาด

ง) กรณีที่มีการเขียนตัวอักขระที่ไม่อยู่ในบรรทัด เช่น วรรณยุกต์ สระบน สระล่าง ตัวอักขระเหล่านี้จะถูกเขียนไปยังระดับต่างๆของคอลัมน์ก่อนหน้าคอลัมน์ที่ตัวชี้ตำแหน่งอยู่ และตัวชี้ตำแหน่งไม่เคลื่อนที่

ในการพัฒนา ตามข้อกำหนดนี้จึงมีการเพิ่มฟังก์ชันต่อไปนี้

ฟังก์ชัน int waddvowel(WINDOW *win, int ch)

คำอธิบาย เพิ่มสระบน หรือสระล่าง ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win

ผลลัพธ์ 0 = เพิ่มสำเร็จ
1 = มีข้อผิดพลาด ซึ่งอาจเกิดจาก
- มีวรรณยุกต์อยู่ในคอลัมน์นั้นก่อนแล้ว
- ในคอลัมน์นั้นไม่มีตัวอักขระที่อยู่ในบรรทัดมาก่อน
- อ้างคอลัมน์ไม่ถูกต้อง

ฟังก์ชัน int waddtonal(WINDOW *win, int ch)

คำอธิบาย เพิ่มวรรณยุกต์ ลงในบัฟเฟอร์ ที่เก็บข้อมูลสำหรับการแสดงผล ของวินโดว์ win

ผลลัพธ์ 0 = เพิ่มสำเร็จ
1 = มีข้อผิดพลาด ซึ่งอาจเกิดจาก
- ในคอลัมน์นั้นไม่มีตัวอักขระที่อยู่ในบรรทัดมาก่อน
- อ้างคอลัมน์ไม่ถูกต้อง

จ) กรณีที่ต้องการลบคอลัมน์ ทำได้โดยการเขียนช่องว่าง ลงไปยังคอลัมน์ที่ต้องการลบ กรณีที่ต้องการลบบางระดับ จะต้องเรียกฟังก์ชันการลบที่ระบุระดับ

ข้อกำหนดนี้ใช้ในการพัฒนา ฟังก์ชันต่อไปนี้

ฟังก์ชัน int wdelvowel(WINDOW win)

คำอธิบาย ลบสระบนหรือสระล่างที่อยู่ในตำแหน่งปัจจุบันบนวินโดว์ win โดยแทนที่ด้วยช่องว่าง ณ ตำแหน่งของสระ

ผลลัพธ์ 0 = สำเร็จ
1 = มีข้อผิดพลาด

ฟังก์ชัน int wdeltonal(WINDOW win)

คำอธิบาย ลบวรรณยุกต์ที่อยู่ในตำแหน่งปัจจุบัน บนวินโดว์ win โดยแทนที่ด้วยช่องว่าง

ณ ตำแหน่งของวรรณยุกต์

ผลลัพธ์ 0 = สำเร็จ
1 = มีข้อผิดพลาด

ฟังก์ชัน int wdech(WINDOW win)

คำอธิบาย ลบตัวอักขระกลาง สระบนหรือสระล่าง และวรรณยุกต์ที่อยู่ในตำแหน่งปัจจุบัน บนวินโดว์ win พร้อมทั้งขยับตัวอักขระทั้งหมดตั้งแต่ตำแหน่งปัจจุบัน ไปจนสิ้นสุดบรรทัด มาแทนที่ตำแหน่งที่ลบไป

ผลลัพธ์ 0 = สำเร็จ
1 = มีข้อผิดพลาด

ฉ) การป้อนสระบน สระล่าง และวรรณยุกต์ จะต้องป้อนตามลำดับที่แน่นอนคือ ป้อนสระก่อน แล้วจึงป้อนวรรณยุกต์ เพื่อให้โปรแกรมเลียนแบบเทอร์มินัล ทราบตำแหน่งก่อนหลังที่แน่นอน เพราะโดยโครงสร้างการเก็บข้อมูลแล้ว เคิร์สไม่ทราบว่าระดับไหนถูกป้อนเข้ามาก่อน แต่โปรแกรมเลียนแบบเทอร์มินัลจะทราบ

ในการพัฒนา จะตรวจสอบเรื่องนี้ในฟังก์ชัน addalpha() addvowel() addtonal()

ช) เคิร์สมีหน้าที่ในการปรับ 2 มิติ ให้สามารถแสดงอย่างถูกต้องด้วยโปรแกรมเลียนแบบเทอร์มินัลที่เป็นแบบ 1 มิติ ที่มีบัฟเฟอร์ไม่น้อยกว่า 240 ตัวอักขระต่อบรรทัด โดยเคิร์สจะต้องแปลงตำแหน่งจาก 2 มิติ ไปเป็น 1 มิติ ดังต่อไปนี้

1) การเคลื่อนที่ไปยังตำแหน่งต่างๆ แบบสัมพัทธ์ ตำแหน่งที่ส่งไปยังโปรแกรมเลียนแบบเทอร์มินัล จะต้องถูกปรับเสริมด้วยจำนวนของตัวอักขระที่ไม่ได้อยู่ในบรรทัด

2)การเคลื่อนที่แบบสมบูรณ์ ในกรณีนี้เคิร์สต้องคำนวณตำแหน่งสำหรับ โปรแกรมเลียนแบบเทอร์มินัลใหม่ ตามตัวอักขระที่เก็บในบัฟเฟอร์ 2 มิติ

ในฟังก์ชัน doupdate() จะต้องเพิ่มการคำนวนตำแหน่งให้สอดคล้องกันระหว่างบัฟเฟอร์ของเคิร์สกับตำแหน่งในบัฟเฟอร์ของเทอร์มินัล ซึ่งจะกล่าวโดยละเอียดในหัวข้อถัดไป

3) การแทนที่คอลัมน์ด้วยช่องว่าง เคิร์สจะต้องตรวจสอบว่า คอลัมน์นั้นมีกี่ตัวอักขระต้องแทนที่ช่องว่าง ที่ตำแหน่งที่เป็นตัวอักขระกลาง ของบัฟเฟอร์ของโปรแกรมเลียนแบบเทอร์มินัล และลบตำแหน่งที่เป็นสระและวรรณยุกต์ทิ้ง

กรณีที่เป็นการลบเฉพาะระดับ เคิร์สต้องสั่งให้โปรแกรมเลียนแบบเทอร์มินัล เลื่อนไปยังตำแหน่ง ที่จะลบ ก่อนทำการลบ

4) กรณีที่มีการพิมพ์สระบน สระล่าง หรือ วรรณยุกต์ เคิร์สจะต้อง สั่งให้ โปรแกรมเลียนแบบเทอร์มินัล แทรกตัวอักขระเหล่านี้เสมอ ไม่ว่าขณะนั้นจะอยู่ในโหมดพิมพ์ทับหรือพิมพ์แทรก ยกเว้นกรณีที่พิมพ์ตัวอักขระทับ ในระดับที่มีอยู่แล้ว โดยตรวจสอบจากเคิร์สบัฟเฟอร์ว่ามีตัวอักขระในระดับนั้นๆหรือไม่ เคิร์สต้องสั่ง ให้โปรแกรมเลียนแบบเทอร์มินัลแทนที่ตัวเดิมด้วยตัวอักขระใหม่

ข้อบังคับเหล่านี้จะถูกเพิ่มเข้าไปในส่วนของฟังก์ชัน doupdate() ซึ่งจะกล่าวถึงอัลกอริทึมในรายละเอียดต่อไป

3.3 โครงสร้างข้อมูลสำหรับภาษาไทย 80 คอลัมน์

ภาษาไทยมีลักษณะ 2 มิติ โครงสร้างข้อมูลที่ใช้ในการพัฒนาเคิร์สจึงต้องรองรับลักษณะเช่นนี้ได้ ในหัวข้อนี้จะกล่าวถึงโครงสร้างข้อมูลที่ใช้ในเคิร์สเพื่อให้รองรับภาษาไทย

ก.) โครงสร้างข้อมูลของแต่ละคอลัมน์

แต่ละคอลัมน์ ในบัฟเฟอร์สำหรับการแสดงผล มีโครงสร้างข้อมูลดังนี้

```
typedef struct
{
        /* one column has 3 levels of charactor. */
        int     alpha,
                vowel,
                tonal;
        short   attr;
} __LDATA;
```

alpha เป็น integer สำหรับเก็บตัวอักขระที่เป็นพยัญชนะ และ สระกลาง

vowel เป็น integer สำหรับเก็บตัวอักขระที่เป็นสระบน และ สระล่าง

tonal เป็น integer สำหรับเก็บตัวอักขระที่เป็นวรรณยุกต์

attr เป็น short integer สำหรับเก็บ attribute ของคอลัมน์หนึ่งๆ

การเก็บข้อมูลลงในโครงสร้างนี้ จะต้องเก็บเรียงลำดับก่อนหลังดังนี้ คือ เก็บค่าพยัญชนะ ก่อน ถ้ามีสระเก็บค่าสระ ถ้ามีวรรณยุกต์เก็บค่าวรรณยุกต์เป็นตัวสุดท้าย เหตุที่ต้องเรียงลำดับการจัดเก็บ ก็เพื่อให้สอดคล้องกับการทำงานของโปรแกรมเลียนแบบเทอร์มินัล เนื่องจาก โปรแกรมเลียนแบบเทอร์มินัล เก็บข้อมูลเป็นสายอักขระ (string) ลำดับที่ จึงเป็นสิ่งสำคัญ เพื่อให้สามารถอ้างถึง ตัวอักขระ โดยเฉพาะ ตัวอักขระที่ไม่ได้อยู่ในบรรทัด ระหว่างเคิร์ส และ โปรแกรมเลียนแบบเทอร์มินัลให้ได้ตรงกัน จึงต้องกำหนด ลำดับก่อนหลังในการเก็บข้อมูล ลงในโครงสร้างข้อมูลของเคิร์ส

ข.) โครงสร้างข้อมูลสำหรับแต่ละบรรทัด

```
typedef struct
{
        __LDATA     *line;              /* pointer to the line text. */
        int         firstch, lastch;    /* first and last changed columns. */
} __LINE;
```

ค.) โครงสร้างข้อมูลสำหรับแต่ละวินโดว์

```
typedef struct __window
{
        int         begy, begx;     /* window home. */
        int         cury, curx;     /* current x, y coordinates. */
        int         maxy, maxx;     /* maximum values for curx, cury. */
        __LINE      **lines;        /* array of pointers to the lines */
        __LINE      *lspace;        /* line space */
```

```
    __LDATA         *wspace;        /* window space */
    struct __window
                    *nextp, *orig;  /* subwindows list and parent. */
    short           ch_off;         /* x offset for firstch/lastch.*/
    unsigned        _use_keypad : 1;
    short           _attrs;         /* current window attributes */
    short           _clear;         /* clear window */
    short           _scroll;        /* scroll ok info */
    short           flags;
} WINDOW;
```

โดยที่

begy - ตำแหน่งบรรทัดเริ่มต้นของวินโดว์ เมื่อเทียบกับหน้าจอจริง

begx - ตำแหน่งคอลัมน์เริ่มต้นของวินโดว์ เมื่อเทียบกับหน้าจอจริง

cury - ตำแหน่งบรรทัดปัจจุบันของวินโดว์ เมื่อเทียบกับ begy

curx - ตำแหน่งคอลัมน์ปัจจุบันของวินโดว์ เมื่อเทียบกับ begx

maxy - จำนวนบรรทัดที่แสดงผลของวินโดว์ในขณะหนึ่งๆ

maxx - จำนวนคอลัมน์ที่ใชแสดงผลของวินโดว์ในขณะหนึ่งๆ

lines - array ของ pointer ที่ชี้ไปยัง lines ต่างๆ ของวินโดว์

lspace - pointer ที่ชี้ไปยัง address ของพื้นที่ในหน่วยความจำที่ใช้เก็บ data ในการแสดงผลของแต่ละบรรทัด

wspace - pointer ที่ชี้ไปยังพื้นที่ในหน่วยความจำที่ใช้เก็บ data ในการแสดงผลของทั้งวินโดว์

_use_keypad - มีค่าเป็นจริงเมื่อมีการใช้ keypad มีค่าเป็นเท็จเมื่อไม่ใช้ keypad

_attrs - attribute ของวินโดว์ในปัจจุบัน

_clear - flag ใช้สำหรับตรวจสอบว่าต้องการ clear screen หรือไม่

_scroll - flag ว่าสามารถ scroll screen ได้หรือไม่

flags - วินโดว์ flag

3.4 การกำหนดให้เคิร์สรู้จักภาษาไทย

เพื่อให้เคิร์สรู้จักภาษาไทย ได้กำหนดไฟล์(configulation file)ชื่อ curses.cfg ไฟล์นี้จะเก็บตัวอักขระที่ไม่ได้อยู่ในบรรทัด โดยแบ่งเป็น 2 ประเภทคือสระ และ วรรณยุกต์ โดยมีรูปแบบการจัดเก็บดังนี้

```
TONAL  0x?? 0x??  ...  0x??
VOWEL      0x?? 0x?? ... 0x??
```

โดยที่ ?? คือตัวเลขฐานสิบหกของตัวอักขระนั้น

คำแรกในบรรทัดจะเป็นคำสำคัญ ที่ระบุว่าตัวอักขระที่ระบุโดยเลขฐานสิบหก ต่อไปนี้ เป็น สระ หรือ วรรณยุกต์

เคิร์สจะใช้ข้อมูลในไฟล์นี้ ตัดสินว่าตัวอักขระใดเป็นสระ เป็นวรรณยุกต์ หรือ เป็นพยัญชนะ กรณี

ที่พบรหัสควบคุม เช่น control-A control-B หรือ escape ต่างๆ เคิร์สจะปฏิบัติกับรหัสควบคุมเหล่านี้เสมือนว่าเป็นพยัญชนะตัวหนึ่ง คือเคิร์สจะไม่ตีความรหัสควบคุม

ฟังก์ชันที่เกียวข้อง

ฟังก์ชัน int read_cfg(void)
คำอธิบาย อ่านค่าจากไฟล์ curses.cfg มาเก็บใน memory ในตัวแปรอาร์เรย์ ชือ int tonal_tab[] int vowel_tab[]
ผลลัพธ์ 0 = ทำสำเร็จ
1 = มีข้อผิดพลาด

ฟังก์ชัน int char_type(char ch)
คำอธิบาย ตรวจสอบว่า ch เป็น พยัญชนะ สระ หรือ วรรณยุกต์
ผลลัพธ์ 0 = เป็นสระบน หรือ สระล่าง
1 = เป็นวรรณยุกต์
2 = เป็นพยัญชนะ หรือ สระกลาง
3 = มีข้อผิดพลาด

3.5 การแปลงตำแหน่งระหว่างบัฟเฟอร์ของเคิร์สกับ บัฟเฟอร์ของเทอร์มินัล

เคิร์สจะมีการส่งข้อมูลให้กับเทอร์มินัลทุกครั้งที่มีการเรียกใช้ฟังก์ชั่น doupdate() ฟังก์ชันนี้เดิมเปรียบเทียบ จอภาพเสมือน (curscr) กับ จอภาพจริง (stdscr) ถ้าจอภาพเสมือนมีการเปลียนแปลงไปจากจอภาพจริง doupdate จะส่งข้อมูลที่มีการเปลียนแปลงไปยังเทอร์มินัล

สิ่งที่ต้องมีการเปลี่ยนแปลงใน doupdate() คือ

ก) ก่อนส่งตำแหน่งไปยังเทอร์มินัล ต้องมีการคำนวนหา ตำแหน่งคอลัมน์บนบัฟเฟอร์ของเทอร์มินัล ที่สัมพันธ์กับตำแหน่งคอลัมน์บนบัฟเฟอร์ของเคิร์ส เนื่องจากการเก็บข้อมูลระหว่างเทอร์มินัลซึ่งเป็นแบบ 1 มิติ คอลัมน์ที่ใช้อ้างถึงจะไม่ตรงกับ การเก็บข้อมูลในเคิร์สที่เป็นแบบ 2 มิติ

ฟังก์ชันที่เกียวข้อง

ฟังก์ชัน int cal_1d_col(int row, int col)
คำอธิบาย คำนวนหาคอลัมน์ถ้าเก็บข้อมูลแบบ 1 มิติ ที่ตรงกับ col ในการเก็บข้อมูลแบบ 2 มิติ
ผลลัพธ์ ตำแหน่งคอลัมน์ถ้าเก็บข้อมูลแบบ 1 มิติ

ข) ส่งตำแหน่งที่คำนวนได้ไปให้เทอร์มินัล เพื่อเลื่อนตัวชี้ตำแหน่ง จะเห็นว่าโปรแกรมเลียนแบบเทอร์มินัลต้องสามารถอ้างอิงคอลัมน์ได้ถึง 240 คอลัมน์ แทนที่จะเป็น 80 คอลัมน์ตามปกติ

ค) จากนั้นส่งตัวอักขระที่ต้องการแสดงไปยังเทอร์มินัล โดยในแต่ละคอลัมน์ ต้องส่งตามลำดับก่อนหลังดังต่อไปนี้

- พยัญชนะหรือสระกลาง
- สระบนหรือสระล่าง ถ้ามี
- วรรณยุกต์ ถ้ามี

ง) ก่อนส่งตัวอักขระที่ไม่ได้อยู่ในบรรทัดออกไป ต้องมีการเช็คก่อนว่า เป็นการส่งเพื่อไปแทนที่ตัวอักขระที่ไม่ได้อยู่ในบรรทัดที่มีอยู่เดิมแล้ว หรือ เป็นการส่งไปใหม่ ถ้าเป็นการส่งใหม่ ต้องส่งคำสั่งให้โปรแกรมเลียนแบบเทอร์มินัล แทรกตัวอักขระที่ส่งไปให้ใหม่ แต่ถ้าเป็นการแทนที่ก็สามารถส่งตัวอักขระไปแทนที่ได้เลย

จ) กรณีที่พบว่ามีการลบตัวอักขระที่ไม่ได้อยู่ในบรรทัด เคิร์สต้องส่งคำสั่ง ลบตัวอักขระไปยังโปรแกรมเลียนแบบเทอร์มินัล

อัลกอริทึม ของฟังก์ชัน doupdate เป็นไปดังนี้

```
doupdate()
{
    for (บันทัดแรกถึงบันทัดสุดท้าย)
    {
        if ( บันทัดนั้นมีการเปลี่ยนแปลง )
        {
            หาค่าคอลัมน์แบบ 1มิติ ของตัวอักขระแรกที่มีการเปลียนแปลงใน row;
            สั่งให้โปรแกรมเลียนแบบเทอร์มินัลเลื่อนตัวชี้ตำแหน่งไปที่ตำแหน่งนั้น;
            for ( col= ตำแหน่งแรกที่มีการเปลียน;
                col <= ตำแหน่งสุดท้ายที่มีการเปลียน; col++)
            {
                ส่ง attribute ของ col ไปยังเทอร์มินัล;
                ส่ง ตัวอักขระระดับที่ 1 ไปยังเทอร์มินัล;
                if ( ตัวอักขระระดับที่ 2 != ' ' )
                {
                    if ( เดิมไม่เคยมีตัวอักขระระดับที่ 2 ที่ col นี้ )
                        ส่งคำสั่งแทรกตัวอักขระไปยังเทอร์มินัล;
                    ส่ง ตัวอักขระระดับที่ 2 ไปยังเทอร์มินัล;
                }
                else
                {
                    if ( เดิมเคยมีตัวอักขระระดับที่ 2 ที่ col นี้ )
                        ส่งคำสั่งลบอักขระไปยังเทอร์มินัล;
                }
                if ( ตัวอักขระระดับที่ 3 != ' ' )
                {
                    if ( เดิมไม่เคยมีตัวอักขระระดับที่ 3 ที่ col นี้ )
                        ส่งคำสั่งแทรกตัวอักขระไปยังเทอร์มินัล;
                    ส่ง ตัวอักขระระดับที่ 3 ไปยังเทอร์มินัล;
                }
                else
```

```
                                {
                                    if ( เดิมเคยมีตัวอักขระระดับที่ 3 ที่ col นี้ )
                                        ส่งคำสั่งลบอักขระไปยังเทอร์มินัล;
                                }
                            }
                        }
                    }
                }
```

การปรับลักษณะการเก็บข้อมูลและการเพิ่มฟังก์ชั่นของเคิร์สดังที่กล่าวในบทนี้ ทำให้ด้านยูนิกซ์รู้จักภาษาไทย แต่เพื่อให้การแสดงผลภาษาไทยได้อย่างสมบูรณ์ ต้องปรับปรุงให้โปรแกรมเลียนแบบเทอร์มินัล รองรับลักษณะการทำงานของเคิร์สด้วย ดังจะกล่าวถึงในบทต่อไป