บทที่ 6

โปรแกรมบรรณาธิกรภาษาไทย (tvi)

tvi เป็นโปรแกรมบรรณาธิกรที่สามารถรับและแสดงผลข้อความภาษาไทยทางจอภาพภายใต้ระบบปฏิบัติการยูนิกซ์ โดยทำงานตาม vi ในส่วนงานหลักคือการรับและแสดงผลข้อความทางจอภาพ รวมทั้งการแก้ไขและเก็บในแฟ้มข้อมูล ส่วนลักษณะงานที่ซับซ้อนกว่าปกติที่เป็นความสามารถของ vi ไม่ได้รวมไว้ใน tvi ขณะนี้

6.1 โครงสร้างข้อมูล (Data Structure)

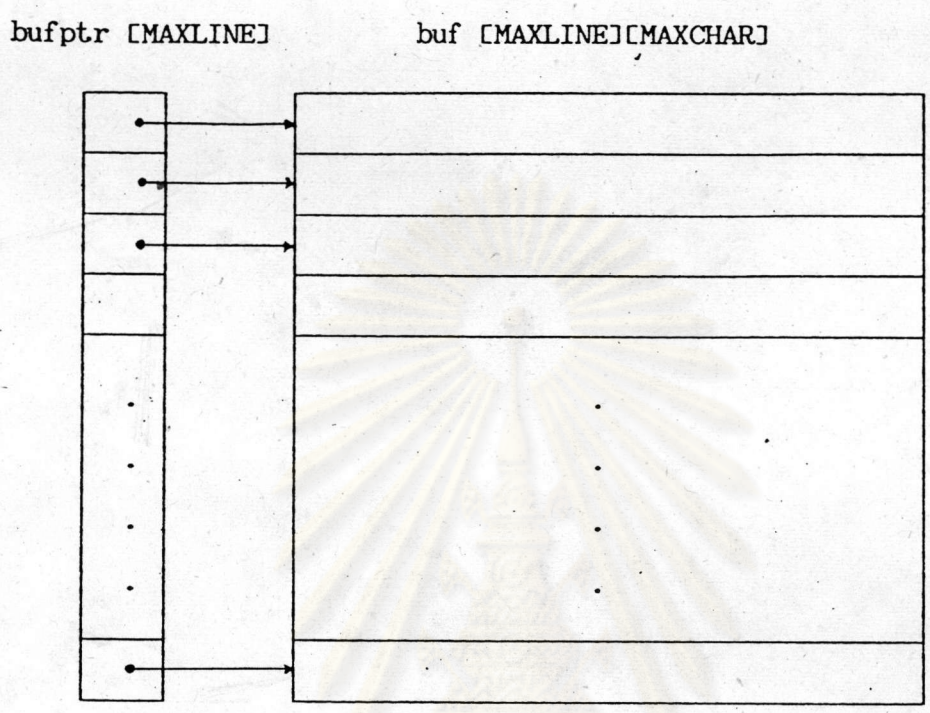
6.1.1 บัฟเฟอร์ (Buffer) โครงสร้างข้อมูลที่ใช้ในโปรแกรมบรรณาธิกรมีหลายรูปแบบ ซึ่งมีวิธีการใช้ ข้อดีข้อเสีย และความเหมาะสมของลักษณะงานที่ใช้แตกต่างกัน การเลือกใช้โครงสร้างข้อมูลที่เหมาะสม ทำให้โปรแกรมทำงานอย่างมีประสิทธิภาพ โครงสร้างของข้อมูลชนิดต่างๆ ที่ใช้กันแพร่หลาย เช่น string, stack, queue, array, tree, linked list, multilinked list, table, file เหล่านี้เป็นต้น

เนื่องจากลักษณะงานของ tvi เป็นการประมวลผลที่เกี่ยวข้องกับข้อความ ตั้งแต่การนำข้อมูลเข้า การแก้ไขเปลี่ยนแปลงข้อความในลักษณะต่าง ๆ เช่น การแทรก การลบ การคัดลอกข้อความ ดังนั้นรูปแบบของโครงสร้างข้อมูลที่ใช้จะต้องเป็นวิธีการที่เหมาะสมสำหรับงานประเภทนี้ และเก็บข้อความที่นำเข้าได้อย่างต่อเนื่องกัน รูปแบบโครงสร้างข้อมูลดังกล่าวได้แก่ array, list หรือ file นอกจากนี้การทำงานของโปรแกรมบรรณาธิกรโดยทั่วไปจะเกี่ยวข้องกับการประมวลผลข้อความในลักษณะของบรรทัด ดังนั้นการเก็บข้อมูลของโปรแกรมบรรณาธิกรจะเก็บตามความสัมพันธ์ของบรรทัดเพื่อความสะดวกในการทำงาน

โครงสร้างของข้อมูลที่ใช้ในการเขียนโปรแกรม tvi เป็นลักษณะของ Pointer to array of character โดยกำหนดให้เก็บข้อความแต่ละบรรทัดในรูปแบบของตัวแปรชนิดแถวลำดับ (array) และมีตัวชี้ (pointer) บอกลำดับของบรรทัด เพื่อความสะดวกในการเรียกข้อความบรรทัดต่าง ๆ มาประมวลผล

สำหรับการประมวลผลข้อความภายในบรรทัดใด ๆ จะทำการประมวลผลแบบตามลำดับ (sequential) ส่วนการประมวลผลในลักษณะของบรรทัด เช่น การแทรกหรือลบข้อความเป็นบรรทัด ทำโดยการเปลี่ยนลำดับค่าตัวชี้ ซึ่งชี้ไปยังบรรทัดที่เกี่ยวข้อง แทนการเปลี่ยนแปลงลำดับ

ของข้อความในบัฟเฟอร์ ซึ่งทำให้ลดเวลาที่ใช้ในการประมวลผลในรูปแบบดังกล่าว



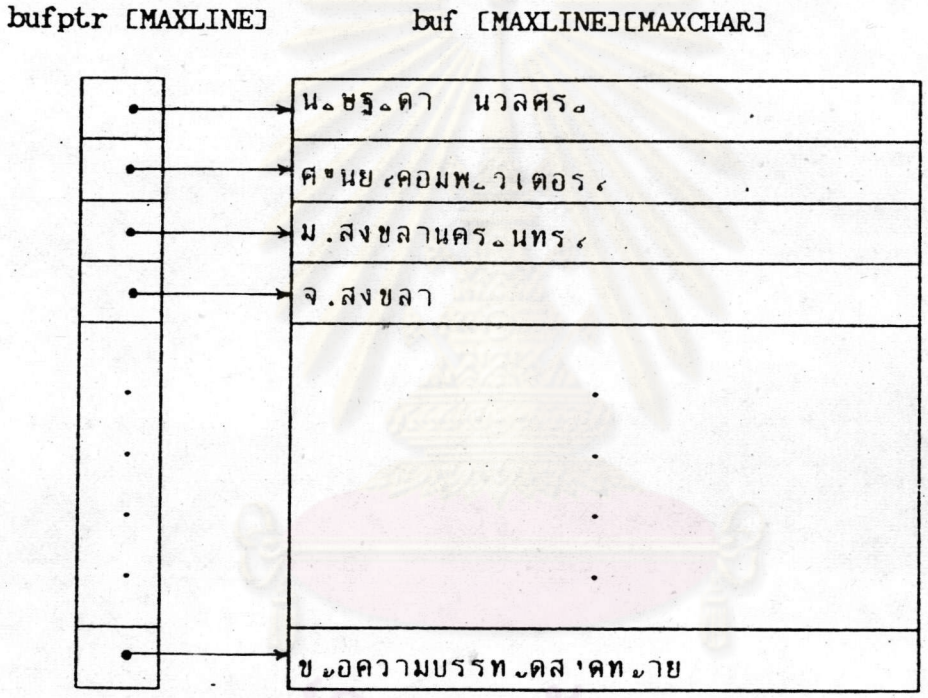
รูปที่ 6.1 รูปแบบของโครงสร้างข้อมูลชนิด Pointer to Array of Character

buf เป็นตัวแปรชนิดแถวลำดับ 2 มิติ (2-dimensional array) สำหรับเก็บข้อความเมื่อ · MAXLINE คือจำนวนบรรทัดสูงสุดที่กำหนดไว้ และ MAXCHAR คือจำนวนตัวอักษรสูงสุดในหนึ่งบรรทัด

bufptr เป็นตัวแปรชนิดแถวลำดับหนึ่งมิติ สำหรับเก็บตำแหน่งของแต่ละบรรทัดในตัวแปร buf เพื่อชี้บอกลำดับของบรรทัดต่าง ๆ

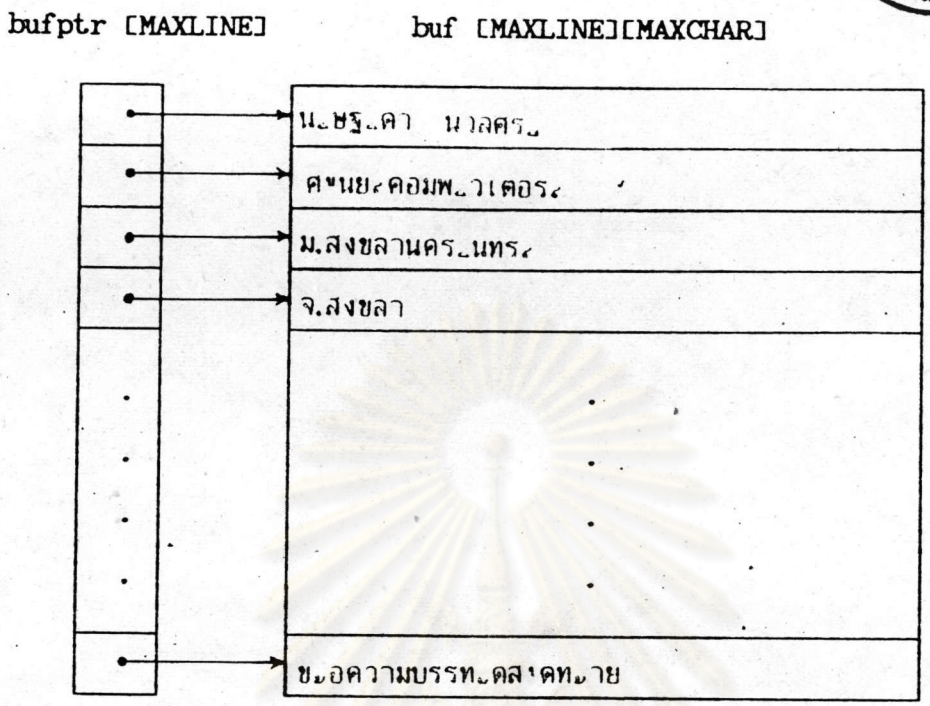
<u>การทำงาน</u> เมื่อมีการเรียกใช้ข้อความในบรรทัดใด ๆ ทำโดยการอ่านตัวชี้บอกบรรทัดนั้นจากตัวแปร bufptr เพื่อเข้าถึงข้อความของบรรทัดที่ต้องการและประมวลผลแบบตามลำดับภายในบรรทัดนั้น แต่ในกรณีที่ต้องการเปลี่ยนแปลงข้อความในลักษณะของบรรทัด จะเปลี่ยนแปลงค่าตัวชี้ในตัวแปร bufptr ให้ชี้ไปยังบรรทัดที่เก็บข้อความที่เพิ่มขึ้นหรือลดลง ดังนั้นข้อความที่เก็บในบัฟเฟอร์ไม่จำเป็นต้องเรียงตามลำดับการนำเข้า หรือลำดับของบรรทัดจริง ๆ แต่จะเรียงเฉพาะค่าของตัวแปร bufptr เท่านั้น เพื่อให้การทำงานรวดเร็ว ลดปัญหาที่เกิดจากการจัดการเนื้อที่ และลดการสูญเปล่าของงานที่เกิดจากการเคลื่อนย้ายข้อความของบัฟเฟอร์ทั้งหมด

6.1.1.1 การสร้างบรรทัดใหม่ เมื่อเริ่มเข้าสู่โปรแกรม tvi จะทำการสร้างความสัมพันธ์ของตัวแปร 2 ตัว คือ buf และ bufptr โดยทำการลบล้างข้อความในบัฟเฟอร์เพื่อกำจัดข้อความที่ไม่ต้องการที่อาจเกิดขึ้นออก แล้วสร้างตัวชี้ไปยังแต่ละบรรทัดของบัฟเฟอร์ ในครั้งแรกจะสร้างตัวชี้บอกลำดับบรรทัดต่าง ๆ ตามลำดับของตัวแปร buf ดังนั้น เมื่อทำการเก็บข้อความครั้งแรกจะเก็บตามลำดับบรรทัด ทำให้ข้อความในบัฟเฟอร์เรียงตามลำดับการเข้า เมื่อสิ้นสุดการเก็บข้อความสุดท้ายที่ตัวชี้ชี้ไปจะเป็นบรรทัดว่าง (null) ดังภาพ

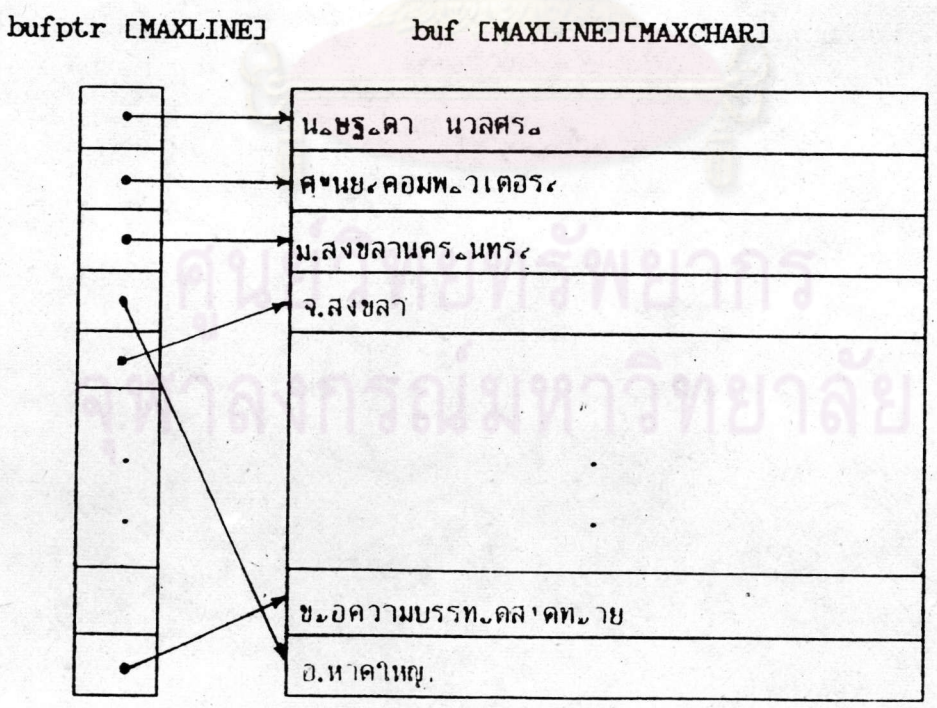


รูปที่ 6.2 แสดงการเก็บข้อความครั้งแรก

6.1.1.2 การแทรกข้อความเป็นบรรทัด การแทรกข้อความทั้งบรรทัดจะไม่ทำการเคลื่อนย้ายข้อความของบัฟเฟอร์ แต่จะทำการเก็บข้อความที่ต้องการแทรกในบรรทัดว่างบรรทัดแรกที่มี แล้วเปลี่ยนลำดับของการชี้ ดังนั้นเมื่อมีการเปลี่ยนแปลงข้อความในลักษณะนี้ทำให้ข้อความภายในบัฟเฟอร์อาจไม่เรียงตามลำดับบรรทัด

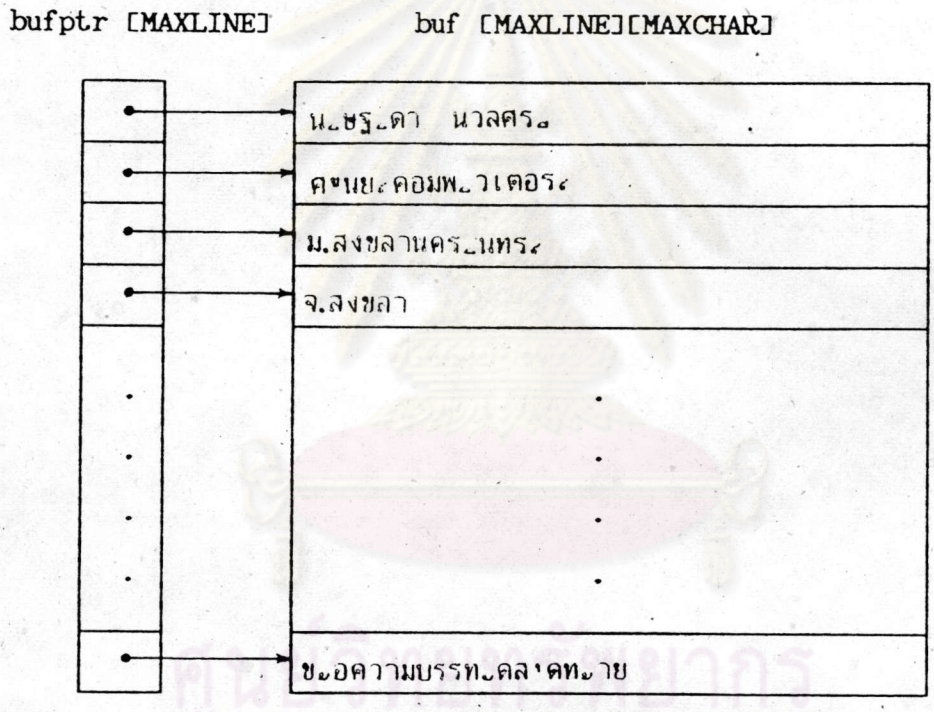


รูปที่ 6.3 แสดงการเก็บข้อความภายในบัฟเฟอร์ก่อนการแทรกข้อความ

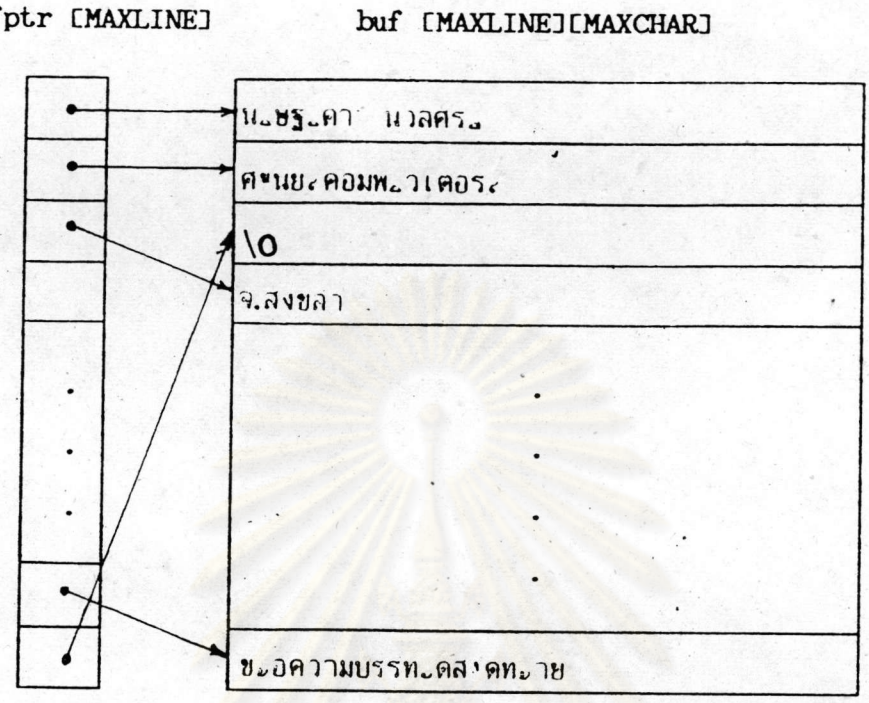


รูปที่ 6.4 แสดงการเก็บข้อความในบัฟเฟอร์หลังการแทรกข้อความ

6.1.1.3 <u>การลบข้อความทั้งบรรทัด</u> ทำงานในทำนองเดียวกันกับการแทรกข้อความ คือจะเปลี่ยนเฉพาะค่าตัวชี้เท่านั้น ไม่เคลื่อนย้ายข้อความของบัฟเฟอร์ แต่ลบล้างข้อความของบรรทัดนั้นออกจากบัฟเฟอร์ ทำให้เป็นบรรทัดว่าง เพื่อนำไปใช้ในการเพิ่มข้อความครั้งต่อไป

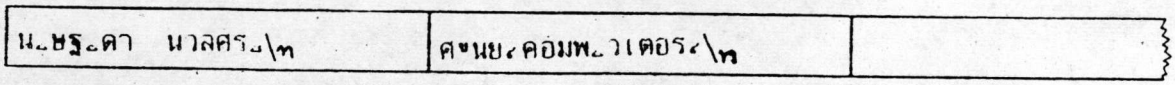


รูปที่ 6.5 แสดงการเก็บข้อความภายในบัฟเฟอร์ก่อนการลบ



รูปที่ 6.6 แสดงการเก็บข้อความภายในบัฟเฟอร์หลังการลบ

6.1.2 <u>แฟ้มข้อมูล</u> (File) แฟ้มข้อมูลที่ใช้เป็นลักษณะของแฟ้มข้อมูลแบบข้อความ (text file) สำหรับเก็บข้อมูลเรียงตามลำดับการนำเข้า หลังจากแก้ไขข้อความภายในบัฟเฟอร์แล้ว และต้องการเก็บในแฟ้มข้อมูลโดยการนำข้อความของบรรทัดตามลำดับค่าตัวชี้ เก็บในแฟ้มข้อมูลแต่ละบรรทัดจะมี \n กำหนดการสิ้นสุดบรรทัด เป็นตัวสุดท้าย ดังนั้นข้อความของแต่ละบรรทัดจะเก็บเรียงต่อกันในแฟ้มข้อมูล ดังภาพ



รูปที่ 6.7 แสดงการเก็บข้อความในแฟ้มข้อมูลแบบข้อความ

6.1.3 หน้าต่าง (Window) หน้าต่างเป็นรูปแบบของการแสดงผลข้อความทางจอภาพรูปแบบหนึ่ง โดยการกำหนดขนาดของจำนวนบรรทัด และตัวอักษรที่สามารถแสดงผลข้อความได้สูงสุดในแต่ละครั้งขนาดของหน้าต่างขึ้นอยู่กับชนิดของจอภาพที่ใช้ด้วย โดยทั่วไปจอภาพสามารถแสดงผลได้ขนาด 24x80 การกำหนดขนาดของหน้าต่างที่ใช้ สามารถกำหนดให้เล็กกว่าขนาดของจอภาพได้ ในที่นี้จะกำหนดเท่ากับขนาดของจอภาพ แต่เนื่องจากข้อความหลักที่ใช้แสดงผลเป็นข้อความภาษาไทย ซึ่งต้องใช้การแสดงผลหลายระดับ ดังที่จะได้กล่าวถึงรายละเอียดต่อไปในหัวข้อ 6.4 ดังนั้นในหนึ่งหน้าต่างจึงทำให้เห็นการแสดงผลเพียง 6 บรรทัดของข้อความภาษาไทย

6.2 การจัดการบนจอภาพ (Screen Manipulation)

6.2.1 Screen Updating and Curser Movement Optimization ("curses")

ในการเขียนโปรแกรมที่เกี่ยวข้องกับการกำหนดตำแหน่งของเคอร์เซอร์ การย้ายเคอร์เซอร์บนจอภาพของเทอร์มินอลชนิดต่าง ๆ จะต้องใช้คุณสมบัติและความสามารถในการทำงานลักษณะดังกล่าวของเทอร์มินอลที่ใช้ ซึ่งคุณสมบัตินี้แตกต่างกันตามชนิดของเทอร์มินอล ทำให้โปรแกรมที่เขียนต้องพึ่งพิงกับอุปกรณ์ชนิดใดชนิดหนึ่งโดยเฉพาะ (device dependent) แต่ในระบบปฏิบัติการยูนิกซ์ ทำได้โดยไม่ต้องพึ่งพิงกับเทอร์มินอลใด ๆ จากที่กล่าวแล้วว่า ในยูนิกซ์จะมีแฟ้มข้อมูลสำหรับเก็บคุณลักษณะของเทอร์มินอลที่ใช้ เมื่อต้องการทำงานที่เกี่ยวกับเทอร์มินอล สามารถอ่านคุณสมบัติของเทอร์มินอลเหล่านั้นจากแฟ้มข้อมูล และเพื่อให้การเขียนโปรแกรมในลักษณะงานดังกล่าวสะดวกและใช้ได้กับเทอร์มินอลทุกชนิดที่นำมาใช้ในระบบได้ จึงมีการสร้างฟังก์ชั่นทั่วไปขึ้น เพื่อจัดการเกี่ยวกับการใช้จอภาพ การย้ายเคอร์เซอร์ การเขียนข้อความบนจอภาพ ฟังก์ชั่นเหล่านี้จัดเป็นฟังก์ชั่นประจำไลบรารี สามารถเรียกใช้ได้ดังนี้

```
#include <curses.h>
```

จากฟังก์ชั่นที่อำนวยความสะดวกดังกล่าว ทำให้การเขียนโปรแกรมสำหรับงานในส่วนที่เกี่ยวกับการจัดข้อความบนจอภาพ การย้ายเคอร์เซอร์ ทำได้โดยไม่ต้องพึ่งพิงอุปกรณ์ชนิดใดชนิดหนึ่งโดยเฉพาะ ทำให้โปรแกรมบรรณาธิกรสามารถใช้กับเทอร์มินอลทุกชนิดที่ใช้ในระบบปฏิบัติการยูนิกซ์ได้

การใช้ ในการเรียกใช้ฟังก์ชั่นดังกล่าว จะต้องกำหนดชื่อเทอร์มินอล ขนาดของหน้าต่างที่ต้องการแสดงข้อความ ครั้งแรกที่จะเรียกใช้ฟังก์ชั่นที่เกี่ยวข้อง ซึ่งการกำหนดคุณสมบัติดังกล่าวของเทอร์มินอล กำหนดได้โดยการเรียกใช้ฟังก์ชั่น getttmode() setterm() และ initscr()

ฟังก์ชั่นที่เกี่ยวข้อง ในที่นี้จะกล่าวถึงเฉพาะฟังก์ชั่นที่นำมาใช้ในการเขียนโปรแกรมบรรณาธิกรครั้งนี้เท่านั้น รายละเอียดนอกเหนือจากนี้สามารถอ่านได้จากคู่มือของ curses และคู่มือ การใช้ยูนิกซ์ทั่วๆ ไป

initscr ()

กำหนดค่าแรกเริ่มที่เกี่ยวกับคุณสมบัติของเทอร์มินอล ขนาดของหน้าต่างที่ต้องการใช้ และจะต้องเรียกใช้ ก่อนที่จะเรียกใช้ฟังก์ชั่นอื่น

addch (ch)
char ch;

นำค่าของตัวแปร ch เขียนในหน้าต่างที่ใช้ ณ ตำแหน่งปัจจุบัน

clear ()

ลบล้างข้อความทั้งหมดที่ปรากฏบนจอภาพ และย้ายเคอร์เซอร์ไปยังตำแหน่งแรกของจอภาพ

clrtobot ()

ลบล้างข้อความตั้งแต่ตำแหน่งปัจจุบันไปจนถึงจุดสุดท้ายของจอภาพ

clrtoeol ()

ลบล้างข้อความตั้งแต่ตำแหน่งปัจจุบันไปจนสิ้นสุดบรรทัด

delch ()

ลบข้อความที่ตำแหน่งปัจจุบันหนึ่งตัวอักษร และเลื่อนตัวอักษรถัดไปมาแทนที่ โดยที่เคอร์เซอร์ปรากฏที่ตำแหน่งเดิม

deleteln ()

ลบข้อความในบรรทัดปัจจุบันหนึ่งบรรทัด เลื่อนข้อความในบรรทัดลำดับถัดไปเข้าแทนที่จนกระทั่งสิ้นสุดจอภาพ ทำให้บรรทัดสุดท้ายของจอภาพว่าง

move (y,x)

int y,x;

เปลี่ยนตำแหน่งปัจจุบันของเคอร์เซอร์ไปยังตำแหน่งที่กำหนดด้วยค่า y และ x

refresh ()

แสดงข้อความของหน้าต่างบนจอภาพ เพื่อให้ข้อความทั้งสองตรงกัน

crmode ()

กำหนดโมดของเทอร์มินอลให้เป็นชนิด cbreak ซึ่งจะรับตัวอักษรทุกตัวที่พิมพ์และส่งไปประมวลผลทันทีโดยไม่ต้องกดปุ่ม return

noecho ()

กำหนดไม่ให้แสดงข้อความที่รับจนกว่าจะมีการใช้คำสั่งแสดงผลข้อความ

getch()

รับตัวอักษรนำเข้าจากเทอร์มินอล และแสดงทางจอภาพ ถ้าไม่กำหนดเป็นอย่างอื่น

endwin()

การเรียกคืนรูปแบบเดิมของเทอร์มินอล เพื่อใช้งานตามปกติ ฟังก์ชั่นนี้จะต้องเรียกใช้ก่อนออกจากโปรแกรมทุกครั้ง

6.3 การจัดการเกี่ยวกับการค้นหาข้อความ (Searching Technique)

ในการประมวลผลข้อความของโปรแกรมบรรณาธิกร งานที่สำคัญอีกส่วนหนึ่งนอกเหนือจากการนำเข้าและแก้ไขข้อความคือ การค้นหาข้อความที่ต้องการ ซึ่งรวมทั้งการค้นหาและ

เปลี่ยนแปลงข้อความ เนื่องจากการค้นหาส่วนใหญ่จะทำการค้นหาในลักษณะของชุดข้อความมากกว่าจะหาเป็นตัวอักษรเดี่ยว ๆ ดังนั้นในการจัดการเกี่ยวกับการค้นหาข้อความจะต้องใช้วิธีการที่รวดเร็วและถูกต้อง จึงจะทำให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพ

จากการศึกษาเกี่ยวกับวิธีการค้นหาข้อความ มีผู้ทดลองค้นคว้าและสรุปเป็นอัลกอริทึมที่ใช้หลายวิธี ซึ่งแต่ละวิธีทำให้ประสิทธิภาพของการทำงานแตกต่างกันและขึ้นอยู่กับลักษณะของโครงสร้างของข้อมูลและลักษณะของข้อมูลที่ใช้ด้วย อัลกอริทึมที่ใช้ในการค้นหาข้อความซึ่งเป็นที่รู้จักกันแพร่หลาย เช่น อัลกอริทึมที่ชื่อ Knuth-Morris-Pratt (Elementary pattern matching), Boyer-Moore Algorithm (A Substring Algorithm) และ Brute-Force Algorithm

สำหรับการค้นหาข้อความของ tvi ในครั้งนี้จะเลือกใช้อัลกอริทึมของ Brute-Force สำหรับเป็นอัลกอริทึมในการค้นหาข้อความที่ต้องการ ซึ่งมีรายละเอียดคือ[1]

ให้ $S = s_1,s_2,s_3,....,s_n$ เป็นข้อความทั้งหมด $P = p_1,p_2,p_3,....,p_m$ เป็นข้อความที่ต้องการค้นหา i และ j เป็นตัวชี้บอกตำแหน่งของข้อความ S และ P ตามลำดับ จะมีวิธีการค้นหาดังนี้

```
set i <- 1
set j <- 1
REPEAT
   IF s[i] = p[j]
      THEN
            set i <- i+1
            set j <- j+1
      ELSE
            set i <- i-j+2
            set j <- 1
```

---

1. Sedgewick. R., **Algorithms**, Addison-Wesley Publishing Company, 1983, pp.243

```
        ENDIF
    UNTIL (j>m) OR (i>n)
    IF (j>m)                (* end of pattern is reached, a match has
                               been found *)
      THEN
        return (i-m)
      ELSE
        return (i)          (* end of text is reached, there is no
                               match *)
    ENDIF
```

โดยที่อัลกอริทึมนี้หากทำการค้นหาข้อความที่มีความยาว M ตัวอักษร จากข้อความที่มีความยาว N ตัวอักษร จะได้ว่าเวลาที่ใช้ในการค้นหามากที่สุด MN ครั้ง และเวลาเฉลี่ยจะเป็น M+N ครั้ง [2]

6.4 การแสดงผลข้อความภาษาไทย

6.4.1 โครงสร้างการเขียนภาษาไทย เนื่องจากการเขียนภาษาไทยใช้หลายระดับ โดยที่ตัวอักษรแต่ละตัวปรากฏในระดับใดระดับหนึ่งเพียงระดับเดียวเท่านั้น ซึ่งแบ่งออกได้ดังนี้

ระดับที่ 1 ประกอบด้วยอักษรประเภทวรรณยุกต์และตัวการันต์ ได้แก่

่ ้ ๊ ๋ ์

ระดับที่ 2 ประกอบด้วยสระบางตัวได้แก่

ั ิ ี ึ ื ็ และ ุ (หยาดน้ำค้าง)

---

2. Sedgewick. R., **Algorithms**, Addison-Wesley Publishing Company, 1983, pp.243

ระดับที่ 3 ประกอบด้วยพยัญชนะไทยทั้งหมด สระ และอักษรพิเศษบางตัว ได้แก่

ก ข ฃ ค ฅ ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ ท ธ น บ ป ผ ฝ พ ฟ ภ ม ย ร ล ว ศ ษ ส ห ฬ อ ฮ ๅ เ แ ไ ใ โ ะ า ๆ และ ฯ

ระดับที่ 4 ประกอบด้วยสระ ุ และ ู

ตัวอย่างการเขียนข้อความภาษาไทยที่ประกอบด้วยตัวอักษรหลายระดับ

ระดับที่ 1

ี ระดับที่ 2

ทสด ระดับที่ 3

ุ ระดับที่ 4

6.4.2 <u>การแสดงผลข้อความภาษาไทยทางจอภาพของโปรแกรม tvi</u> ในการแสดงผลข้อความภาษาไทยทางจอภาพ รวมขั้นตอนการทำงานตั้งแต่การรับข้อมูลนำเข้า ตรวจสอบความถูกต้องและแสดงผล โดยจะรับข้อมูลนำเข้าครั้งละหนึ่งตัวอักษร นำไปตรวจสอบความถูกต้องและแสดงผลเมื่อเป็นข้อมูลนำเข้าที่ถูกต้อง

6.4.2.1 การตรวจสอบ วิธีการตรวจสอบความถูกต้องจะตรวจสอบตามหลักการเขียนภาษาไทย ดังนี้

- ถ้าเป็นตัวอักษรตัวแรกของบรรทัด จะต้องเป็นอักษรในระดับที่ 3 เท่านั้น
- ถ้าเป็นอักษรในระดับที่ 2 หรือ 4 ตัวอักษรก่อนหน้าจะต้องเป็นอักษรในระดับที่ 3
- ถ้าเป็นตัวอักษรในระดับที่ 1 ตัวอักษรก่อนหน้าอาจเป็นได้ทั้งระดับที่ 2 หรือ 3 หากเป็นตัวการันต์ จะต้องเขียนตามหลังตัวอักษรในระดับที่ 3 หรือ ิ หรือ ุ เท่านั้น เช่น สวัสดิ์ พันธุ์
- ไม่มีอักษรในระดับที่ 2 และ 4 ปรากฏในตำแหน่งที่ตรงกัน

6.4.2.2 การแสดงผล จะแสดงผลเฉพาะข้อมูลนำเข้าที่ถูกต้องเท่านั้น โดยการจัดระดับของตัวอักษรเหล่านั้น ในที่นี้หากตัวอักษรในระดับที่ 1 ไม่มีอักษรในระดับที่ 2 นำหน้าจะนำอักษรนั้นแสดงผลในระดับที่ 2 แทน เพื่อให้ได้ผลลัพธ์ที่สวยงาม

ตัวอย่าง

◌ั	ระดับที่ 1
◌่	ระดับที่ 2
พนอง	ระดับที่ 3

จะแสดงผลเป็น

◌ั	ระดับที่ 1
◌่	ระดับที่ 2
พนอง	ระดับที่ 3

6.5 คำสั่งของ tvi

คำสั่งที่สามารถใช้ได้ ในโปรแกรม tvi แบ่งตามลักษณะการทำงานได้ดังนี้

6.5.1 คำสั่งเกี่ยวกับการเลื่อนเคอร์เซอร์

6.5.1.1 การเลื่อนเคอร์เซอร์เป็นจำนวนตัวอักษร จะเลื่อนเคอร์เซอร์ไปครั้งละตัวอักษร และเลื่อนไปบนอักษรในระดับที่ 3 ของภาษาไทยเท่านั้น คำสั่งที่ใช้คือ

SPACE, l เลื่อนเคอร์เซอร์ไปทางขวา

BS, h เลื่อนเคอร์ไปทางซ้าย

6.5.1.2 การเลื่อนเคอร์เป็นบรรทัด บรรทัดในที่นี้จะสิ้นสุดด้วยตัวอักษร '\n' เสมอ คำสั่งที่เกี่ยวข้องคือ

j เลื่อนเคอร์เซอร์ไปยังบรรทัดก่อนหน้าหนึ่งบรรทัด ในตำแหน่งที่ตรงกันกับตำแหน่งปัจจุบัน

k เลื่อนเคอร์เซอร์ไปยังบรรทัดถัดไปหนึ่งบรรทัด ในตำแหน่งที่ตรงกันกับตำแหน่งปัจจุบัน

หากบรรทัดที่เลื่อนไปมีความยาวน้อยกว่าตำแหน่งปัจจุบัน จะเลื่อนเคอร์เซอร์ไปยังตัวอักษรสุดท้ายของบรรทัดนั้นแทน

H เลื่อนเคอร์เซอร์ไปยังบรรทัดแรกของข้อความบนหน้าต่าง

M เลื่อนเคอร์ไปยังบรรทัดกลางของข้อความบนหน้าต่าง

L เลื่อนเคอร์เซอร์ไปยังบรรทัดสุดท้ายของข้อความบนหน้าต่าง

6.5.1.3 การเลื่อนเคอร์เป็นประโยคและย่อหน้า โดยกำหนดให้

ประโยค ในข้อความภาษาไทยจะใช้ช่องว่างเป็นตัวสิ้นสุดประโยค

ย่อหน้า กำหนดให้บรรทัดว่างเป็นตัวสิ้นสุดย่อหน้า

คำสั่งที่ใช้ คือ

) เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของประโยคถัดไป

(เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของประโยคก่อนหน้า

} เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของย่อหน้าถัดไป

{ เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของย่อหน้าก่อนหน้า

6.5.1.4 การเลื่อนหน้าต่าง เพื่อแสดงข้อความใหม่บนหน้าต่าง เหมือนกับการเลื่อนหน้าต่างไปยังข้อความในบัฟเฟอร์นั่นเอง คำสั่งที่ใช้มีดังนี้

^F เลื่อนหน้าต่างไปยังข้อความถัดไปครั้งละหนึ่งหน้าต่าง

^B เลื่อนหน้าต่างไปยังข้อความก่อนหน้าครั้งละหนึ่งหน้าต่าง

^U เลื่อนหน้าต่างไปยังข้อความก่อนหน้าครั้งละครึ่งหน้าต่าง

^D เลื่อนหน้าต่างไปยังข้อความถัดไปครั้งละครึ่งหน้าต่าง

6.5.2 <u>การเปลี่ยนแปลงข้อความ</u>

6.5.2.1 การแทรกข้อความ คำสั่งที่ใช้คือ

i แทรกข้อความหน้าเคอร์เซอร์

I แทรกข้อความหน้าตัวอักษรแรกของบรรทัดปัจจุบัน

a แทรกข้อความหลังเคอร์เซอร์

A เพิ่มข้อความหลังตัวอักษรสุดท้ายของบรรทัดปัจจุบัน

o แทรกบรรทัดว่างหลังบรรทัดปัจจุบัน

O แทรกบรรทัดว่างหน้าบรรทัดปัจจุบัน

6.5.2.2 การลบข้อความ คำสั่งที่ใช้คือ

x ลบข้อความครั้งละตัวอักษรที่ตำแหน่งเคอร์เซอร์แล้วเลื่อนข้อความถัดไปแทนที่

D ลบข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปจนสิ้นสุดบรรทัดปัจจุบัน

dd ลบข้อความบรรทัดปัจจุบันหนึ่งบรรทัด แล้วย้ายข้อความในบรรทัดถัดไปแทนที่

d) ลบข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปจนสิ้นสุดประโยคปัจจุบัน

d(ลบข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นประโยคปัจจุบัน

d) ลบข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปจนสิ้นสุดย่อหน้าปัจจุบัน

d{ ลบข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นย่อหน้าปัจจุบัน

6.5.2.3 การแทนที่ข้อความ คำสั่งที่ใช้คือ

r แทนที่ข้อความครั้งละหนึ่งตัวอักษรที่ตำแหน่งเคอร์เซอร์

R แทนที่ข้อความตั้งแต่ตำแหน่งเคอร์เซอร์จนกระทั่งกด ESC

cc เปลี่ยนข้อความเก่าหนึ่งบรรทัดด้วยข้อความใหม่

c) เปลี่ยนข้อความเก่าตั้งแต่ตำแหน่งเคอร์เซอร์จนสิ้นสุดประโยคปัจจุบัน

c(เปลี่ยนข้อความเก่าตั้งแต่ตำแหน่งเคอร์เซอร์จนถึงจุดเริ่มต้นประโยคปัจจุบัน

c} เปลี่ยนข้อความเก่าตั้งแต่ตำแหน่งเคอร์เซอร์จนสิ้นสุดย่อหน้าปัจจุบัน

c{ เปลี่ยนข้อความเก่าตั้งแต่ตำแหน่งเคอร์เซอร์จนถึงจุดเริ่มต้นย่อหน้าปัจจุบัน

6.5.3 <u>การคัดลอกข้อความ</u> คำสั่งที่ใช้คือ

yy คัดลอกข้อความหนึ่งบรรทัด

y) คัดลอกข้อความจากตำแหน่งเคอร์เซอร์จนสิ้นสุดประโยคปัจจุบัน

y(คัดลอกข้อความจากตำแหน่งเคอร์เซอร์จนถึงจุดเริ่มต้นประโยคปัจจุบัน

y} คัดลอกข้อความจากตำแหน่งเคอร์เซอร์จนสิ้นสุดย่อหน้าปัจจุบัน

y{ คัดลอกข้อความจากตำแหน่งเคอร์เซอร์จนถึงจุดเริ่มต้นย่อหน้าปัจจุบัน

p นำข้อความที่คัดลอกแทรกหน้าตำแหน่งเคอร์เซอร์

P นำข้อความที่คัดลอกแทรกหลังตำแหน่งเคอร์เซอร์

6.5.4 <u>คำสั่งอื่น ๆ</u>

6.5.4.1 การแสดงผลข้อความใหม่บนหน้าต่าง ^L ทำการล้างข้อความบนหน้าต่างแล้วแสดงผลใหม่ด้วยข้อความเดิมของบัฟเฟอร์

6.5.4.2 คำสั่งแสดงสถานะของบัฟเฟอร์ ^G สำหรับแสดงสถานะของบัฟเฟอร์ที่กำลังใช้งานโดยจะแสดงชื่อแฟ้มข้อมูลที่กำลังแก้ไข จำนวนบรรทัดของข้อความ หมายเลขบรรทัดปัจจุบัน

6.5.4.3 การรวมข้อความสองบรรทัดเข้าด้วยกัน คำสั่งที่ใช้คือ

J รวมข้อความบรรทัดปัจจุบันและบรรทัดถัดไปเข้าด้วยกันใช้ช่องว่างแยกระหว่างข้อความของสองบรรทัด และเลื่อนเคอร์ไปยังช่องว่างนั้น

6.5.4.4 คำสั่งค้นหาข้อความ คำสั่งที่ใช้

/ ข้อความ ค้นหาข้อความที่ต้องการที่กำหนดหลังเครื่องหมาย / ในทิศทางตั้งแต่ตำแหน่งปัจจุบันไปทางจุดสิ้นสุดของบัฟเฟอร์

? ข้อความ ค้นหาข้อความเหมือนกับคำสั่ง / แต่ในทิศทางตรงกันข้าม

n ใช้หลังจากเรียกใช้คำสั่งค้นหาข้อความรูปแบบใดแบบหนึ่งแล้ว เพื่อทำการค้นหาข้อความเดิมในทิศทางเดียวกันกับคำสั่งที่กำหนดให้ค้นหาครั้งสุดท้าย

6.5.4.5 คำสั่งอ่านข้อความจากแฟ้มข้อมูล คำสั่งที่ใช้คือ

:r ชื่อแฟ้มข้อมูล ทำการอ่านข้อความจากแฟ้มข้อมูลที่กำหนด ถ้ามีจะนำข้อความที่อ่านได้มาแทรกหน้าตำแหน่งปัจจุบัน

6.5.4.6 คำสั่งเขียนข้อความในแฟ้มข้อมูล คำสั่งที่ใช้คือ

:w ชื่อแฟ้มข้อมูล ทำการเขียนข้อความจากบัฟเฟอร์เก็บในแฟ้มข้อมูลที่กำหนดหลังคำสั่ง w

:w! ชื่อแฟ้มข้อมูล ทำการเขียนข้อความจากบัฟเฟอร์เก็บในแฟ้มข้อมูลที่กำหนดหลังคำสั่ง w โดยการเขียนทับข้อความเดิมในแฟ้มข้อมูลนั้น

:w ทำการเขียนข้อความจากบัฟเฟอร์เก็บในแฟ้มข้อมูลที่เรียกใช้ด้วยคำสั่ง tvi

6.5.4.7 คำสั่งออกจากโปรแกรม tvi คำสั่งที่ใช้คือ

:q ออกจากโปรแกรม tvi หลังจากที่ใช้คำสั่งเขียนข้อความแล้ว

:q! ออกจากโปรแกรม tvi โดยไม่ต้องทำการเขียนข้อความในแฟ้มข้อมูล

ZZ ทำการตรวจสอบ หากมีการเปลี่ยนแปลงข้อความในบัฟเฟอร์จะเขียนข้อความของบัฟเฟอร์ในแฟ้มข้อมูลที่เรียกใช้ด้วยคำสั่ง tvi แล้วออกจากโปรแกรม แต่ถ้าไม่มีการแก้ไขข้อความจะออกจากโปรแกรม tvi ได้ทันที