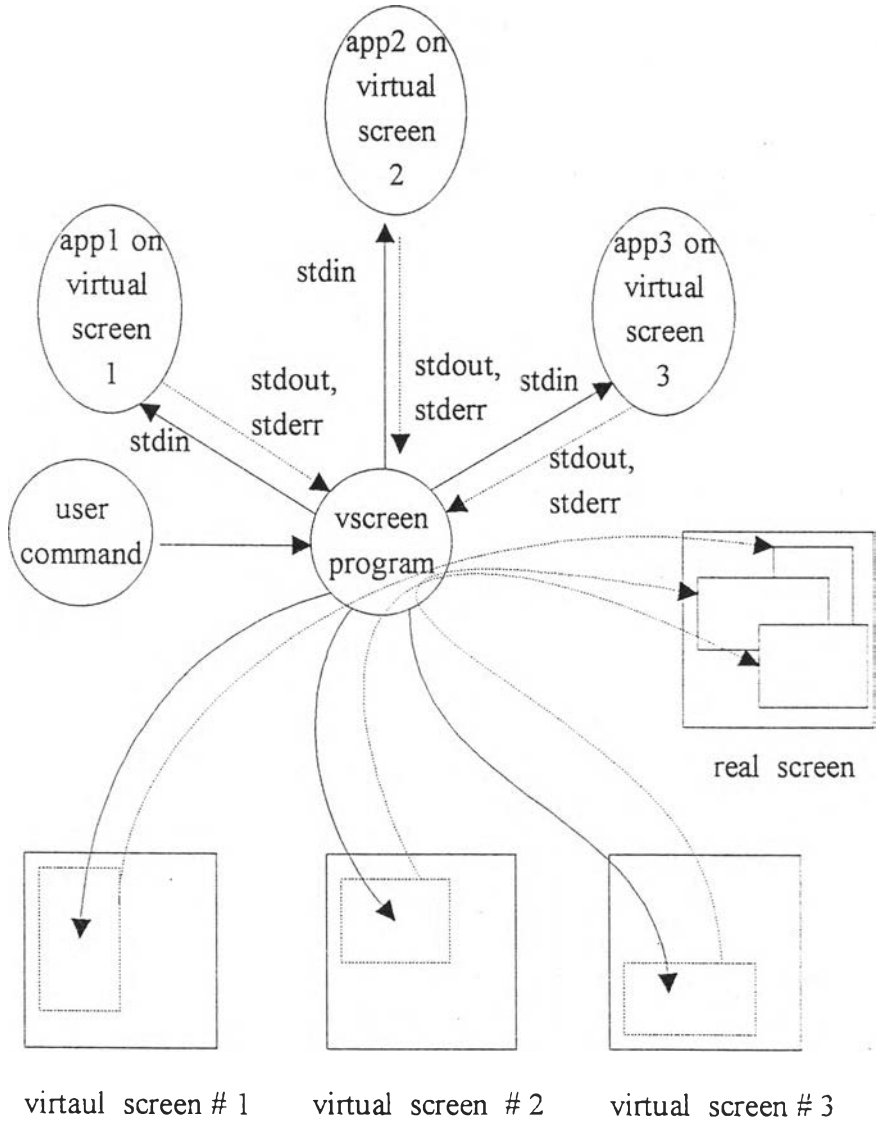
บทที่ 3

การออกแบบโปรแกรม

สำหรับโปรแกรมสร้างจอภาพเสมือนมีส่วนประกอบที่สำคัญอยู่หลายประการ ได้แก่ จอภาพเชิงกายภาพ หน้าต่าง และจอภาพเสมือน เป็นต้น นอกจากนี้ยังประกอบด้วยเรื่องของการติดต่อสื่อสารระหว่างโปรเซส เช่น ไคล์เอ็นด์-เซอร์ฟเวอร์ เป็นต้น ซึ่งจะกล่าวถึงการออกแบบส่วนต่างๆ ดังต่อไปนี้

การออกแบบการทำงานของโปรแกรม

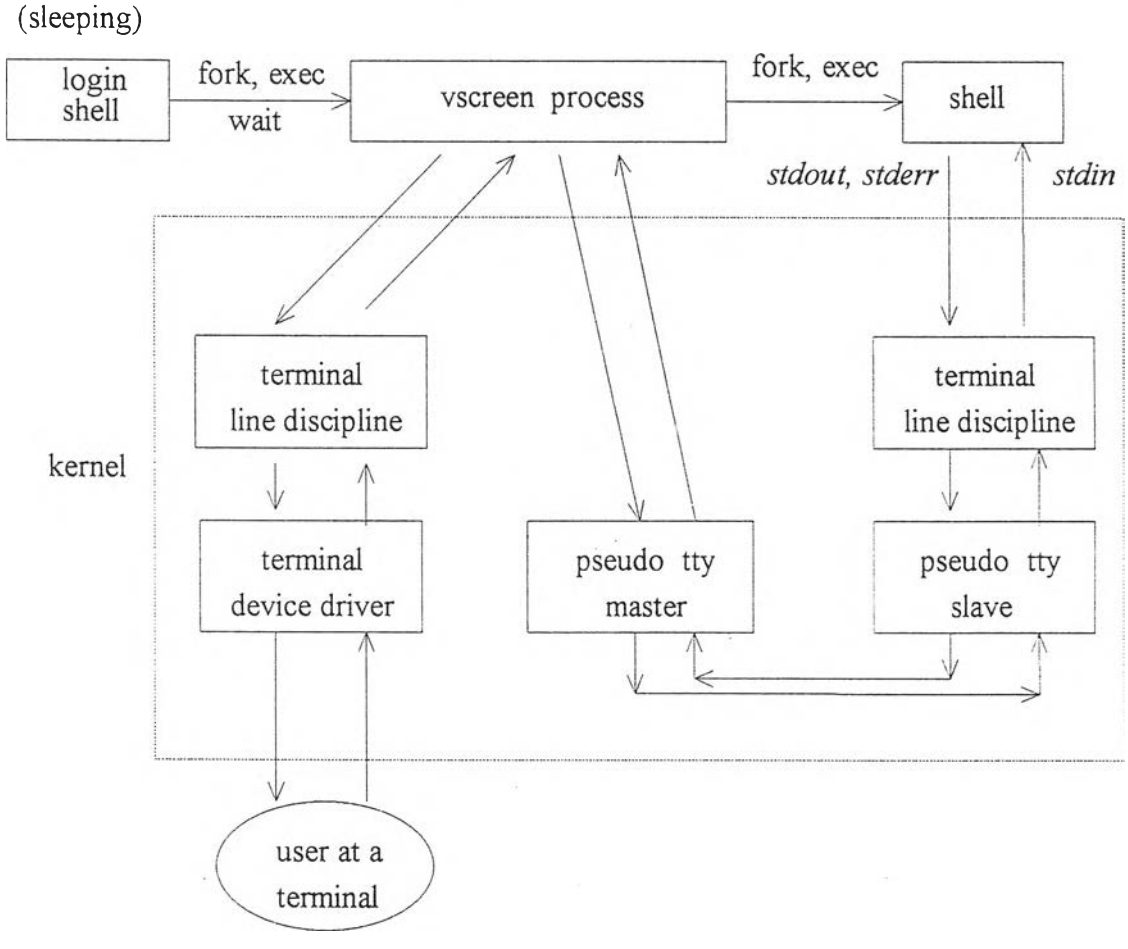


รูปที่ 3.1 แสดงหลักการทำงานของโปรแกรม

จากรูปที่ 3.1 สมมติว่าได้สร้างจอภาพเสมือนไว้ 3 จอภาพเสมือน โดยแต่ละจอภาพเสมือนมีโปรแกรมทำงานอยู่คือ app1 app2 และ app3 โปรแกรมวีสกรีนจะจองเนื้อที่ในหน่วยความจำที่มีขนาดเท่ากับเนื้อที่ที่จะต้องใช้ในการแสดงผลบนจอภาพจริงไว้แทนจอภาพจริงให้สำหรับแต่ละจอภาพ โดยแต่ละจอภาพเสมือนจะมีหน้าต่าง 1 หน้าต่างไว้แสดงผลของข้อมูลในจอภาพเสมือนนั้น โปรแกรมวีสกรีนจะคอยตรวจสอบอยู่เสมอๆ ว่ามีโปรแกรมที่ทำงานอยู่ในจอภาพเสมือนใดต้องการอ่านข้อมูลจากทางแฟ้มข้อมูลเข้ามาตรฐาน (standard input หรือ stdin) หรือต้องการเขียนผลลัพธ์ลงแฟ้มข้อมูลออกมาตรฐาน (standard output หรือ stdout) บ้าง และเมื่อโปรแกรมในจอภาพเสมือนใดต้องการอ่านข้อมูลจากทางแฟ้มข้อมูลเข้ามาตรฐาน โปรแกรมวีสกรีนจะทำการอ่านข้อมูลจากแฟ้มข้อมูลเข้ามาตรฐาน (โดยทั่วไปแฟ้มข้อมูลเข้ามาตรฐานจะหมายถึงแป้นพิมพ์) แล้วส่งต่อให้กับโปรแกรมนั้น และในทำนองเดียวกันกับการแสดงผลของโปรแกรมในจอภาพเสมือนจะถูกส่งผ่านมายังโปรแกรมวีสกรีน ซึ่งโปรแกรมวีสกรีนจะนำผลหรือข้อมูลนั้นเขียนลงในหน่วยความจำที่เตรียมไว้สำหรับแต่ละจอภาพเสมือนแล้วจึงเขียนลงในจอภาพจริง โดยจะจัดให้อยู่ภายในหน้าต่างของจอภาพเสมือนที่โปรแกรมนั้นทำงานอยู่

ส่วนประกอบที่สำคัญของโปรแกรม

1. การสร้างและการดูแลจอภาพเสมือน

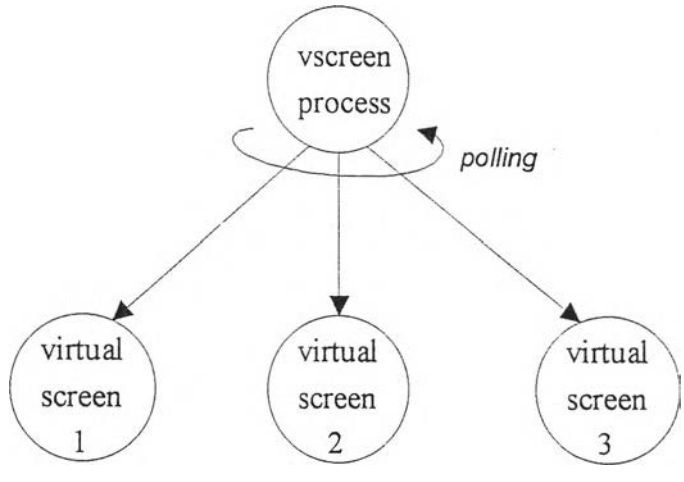


รูปที่ 3.2 แสดงการสร้างจอภาพเสมือน

เมื่อโปรแกรมวีสกรีนต้องสร้างจอภาพเสมือน โปรแกรมวีสกรีนจะสร้างลูก(fork) และทำการเชื่อมโยงเส้นทางการรับ-ส่งข้อมูลมาตรฐานกับโปรเซสลูกโดยผ่านทางเทอร์มินัลเทียม ซึ่งจะทำให้โปรเซสลูกสามารถทำงานได้เหมือนกับกำลังทำงานอยู่กับเทอร์มินัลจริง สำหรับข้อมูล

เข้าที่โปรเซสลูกต้องการจากทางแฟ้มข้อมูลเข้ามาตรฐาน จะถูกรับเข้ามาผ่านโปรแกรมวีสกรีนก่อนแล้วจึงส่งต่อทางเทอร์มินัลเทียมเพื่อให้กับโปรเซสลูก และข้อมูลที่โปรเซสลูกต้องการส่งออกทางแฟ้มข้อมูลออกมาตรฐานจะถูกส่งลงไปทางเทอร์มินัลเทียม ซึ่งโปรแกรมวีสกรีนจะรับไว้ก่อนที่จะส่งออกมาทางจอภาพ

ด้วยหลักการดังกล่าวทำให้โปรแกรมวีสกรีนสามารถจัดการกับข้อมูลที่ต้องแสดงบนจอภาพเชิงกายภาพให้อยู่ภายในบริเวณหน้าต่างของแต่ละโปรเซสลูกได้



รูปที่ 3.3 แสดงการหยั่งสัญญาณ

การติดต่อกับโปรเซสลูกๆ หลายโปรเซส โปรแกรมวีสกรีนจะใช้คำสั่ง select ของยูนิกซ์แบบบีเอสดี ซึ่งระบบจะคอยรับสัญญาณการติดต่อที่ผ่านทางเทอร์มินัลเทียมและ

ซอกเกตจากโปรเซสลูกๆไว้ให้ เมื่อโปรแกรมวีสกรีนจะให้บริการโปรเซสลูกจะใช้วิธีหยั่งสัญญาณ (polling) กับสัญญาณที่ระบบเก็บไว้ให้มาตรวจสอบว่าต้องให้บริการแก่โปรเซสลูกโปรเซสใดบ้าง ซึ่งจะให้บริการเรียงกันไปทีละโปรเซส

สำหรับการสร้างจอภาพเสมือนใหม่นั้นจะสามารถจะทำได้ 2 วิธีคือ วิธีแรกนั้นผู้ใช้กดปุ่มคำสั่งสร้างหน้าต่างขณะใช้งานโปรแกรมวีสกรีน วิธีที่สองคือสร้างโปรเซสเพื่อส่งข้อความว่าต้องการสร้างจอภาพเสมือนไปยังโปรแกรมวีสกรีนที่กำลังทำงานอยู่ ซึ่งโปรแกรมวีสกรีนจะทำการสร้างและดูแลจอภาพเสมือนให้ดังที่ได้กล่าวมาข้างต้น ในวิธีที่สองนี้จะใช้ในลักษณะไคล์เอนต์-เซอร์ฟเวอร์ โดยที่ไคล์เอนต์คือโปรเซสที่ส่งข้อความขอสร้างจอภาพเสมือน และเซอร์ฟเวอร์คือโปรแกรมวีสกรีน ในวิทยานิพนธ์นี้ได้ใช้ซอกเกตไลบรารีฟังก์ชั่นแบบบีเอสดีในการทำส่วนนี้

2. <u>การซ้อนทับของหน้าต่าง (window overlap)</u>

ระบบหน้าต่างที่ใช้ในวิทยานิพนธ์นี้เป็นระบบที่แต่ละหน้าต่างสามารถวางซ้อนทับกันได้ ซึ่งข้อมูลส่วนที่ถูกทับด้วยหน้าต่างที่อยู่ข้างบนย่อมไม่จำเป็นต้องแสดง ดังนั้นการที่จะหาส่วนตัดใช้ (clipping boundary) จึงไม่ใช่เรื่องง่ายนัก เพราะว่าส่วนตัดใช้อาจมีลักษณะเป็นรูปหลายเหลี่ยม

การหาส่วนตัดใช้นั้นมีด้วยกันอยู่หลายวิธี สำหรับในวิทยานิพนธ์นี้จะใช้วิธีที่เรียกว่า ระเบียบวิธีมาสค์ (mask method) (Rochkind,1988) ซึ่งมีวิธีการดังนี้คือเริ่มต้นด้วยการจองหน่วยความจำเป็นข้อมูลประเภทแถวลำดับ 2 มิติที่มีขนาดเท่ากับจอภาพเชิงกายภาพไว้เก็บค่ามาสค์ การสร้างแถวลำดับมาสค์จะใช้วิธีคล้ายกับระเบียบวิธีเขียนทับ (overwriting method) คือจะเริ่มจากหน้าต่างชั้นล่างขึ้นไปชั้นบนโดยเขียนเฉพาะหมายเลขหน้าต่างลงในแถวลำดับเพื่อเป็นค่ามาสค์แทนการเขียนข้อมูลจริง หน้าต่างที่วางซ้อนอยู่ข้างบนจะเขียนหมายเลขหน้าต่างนั้นทับกันไป และเมื่อทำครบทุกหน้าต่าง แถวลำดับของมาสค์ที่ได้จะเป็นการบอกให้ทราบว่าหน้าต่างใดเป็นเจ้าของเซลที่อยู่บนจอภาพเชิงกายภาพ ตัวอย่างเช่น

เริ่มจากหน้าต่างชั้นล่างสุดคือหน้าต่างหมายเลข 0 ซึ่งจะได้ค่าแถวลำดับมาสค์ดังนี้

```
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
```

ถัดมาเขียนทับด้วยบริเวณของหน้าต่างหมายเลข 1

0000000000000011111111111111111100
0000000000000011111111111111111100
0000000000000011111111111111111100
0000000000000011111111111111111100
0000000000000011111111111111111100
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000

และต่อมาเป็นหน้าหมายเลข 2

0000000000000011111111111111111100
0000000000000011111111111111111100
2222222222222222222221111111111100
2222222222222222222221111111111100
2222222222222222222221111111111100
2222222222222222222220000000000000
2222222222222222222220000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000

สุดท้ายหน้าต่างชั้นบนสุดคือหน้าหมายเลข 3

```
00000000000000111111111111111111100
00000000000000111111111111111111100
22222222222222222222211111111111100
22222222222222222222211111111111100
22223333333333333333333333311111100
22223333333333333333333333300000000
22223333333333333333333333300000000
00003333333333333333333333300000000
00000000000000000000000000000000000
00000000000000000000000000000000000
```

สมมติว่าต้องการจะเขียนหน้าต่างหมายเลข 2 แถวลำดับมาสค์จะบอกให้ทราบได้ว่าควรจะเขียนข้อมูลบริเวณใดบ้าง ค่ามาสค์นี้จะอยู่ไปจนกว่าจะมีการสร้างหน้าต่างใหม่ การยกเลิกหน้าต่าง การเปลี่ยนขนาดของหน้าต่าง การย้ายตำแหน่งของหน้าต่าง หรือมีการเปลี่ยนแปลงลำดับของหน้าต่าง

โดยทั่วไปจะเป็นเพียงเฉพาะบางส่วนของจอภาพเชิงกายภาพเท่านั้นที่ต้องเปลี่ยนค่ามาสค์ ความรวดเร็วของการทำงานจะขึ้นอยู่กับการสร้างค่ามาสค์เฉพาะบริเวณใกล้เคียง (localizing) และการเขียนหน้าต่างใหม่ลงในบริเวณที่เกี่ยวข้องบนจอภาพเชิงกายภาพ เช่น เมื่อหน้าต่างถูกยกเลิกการใช้งาน ส่วนที่ต้องถูกปรับ (updated) จะเป็นเฉพาะบริเวณที่หน้าต่างนั้นปิดบังไว้เท่านั้นเป็นต้น

3. ความสัมพันธ์ระหว่างจอภาพเสมือน หน้าต่าง และ จอภาพเชิงกายภาพ

สิ่งที่สำคัญอย่างมากสิ่งหนึ่งในวิทยานิพนธ์นี้คือการจัดการเกี่ยวกับการแสดงผล ซึ่งมีส่วนประกอบหลัก 3 อย่างคือ จอภาพเสมือน หน้าต่าง และจอภาพเชิงกายภาพ ดังจะได้อธิบายต่อไป

จอภาพเชิงกายภาพมีคุณสมบัติดังนี้

1. ประกอบด้วยบริเวณสี่เหลี่ยมสำหรับแสดงตัวอักขระ และสำหรับแต่ละผู้ใช้จะมีเพียง 1 จอภาพเชิงกายภาพเท่านั้น

2. จำนวนแถวและสดมภ์สามารถเปลี่ยนแปลงได้ โดยปกติมีขนาด 24 แถว และ 80 สดมภ์เป็นอย่างน้อย

3. ตัวอักขระที่แสดงผลได้ตามมาตรฐานแอสกี ได้แก่ตัวอักขระที่มีค่ารหัสตั้งแต่ 32 ถึง 126 จะสามารถแสดงผลได้ตามปกติ แต่สำหรับตัวอักขระอื่นๆ อาจถูกเปลี่ยนแปลงหรือถูกแทนด้วยตัวอักขระว่าง

4. ลักษณะประจำของตัวอักขระที่ไม่ใช่ลักษณะประจำแบบปกติอาจถูกแทนด้วยลักษณะประจำตามแบบผกผัน (ตัวอักขระสีดำบนพื้นขาว)

5. มีตัวชี้เพื่อบอกถึงตำแหน่งที่จะใช้แสดงผล โดยปกติจะเป็นตัวอักขระขีดเส้นใต้แบบกระพริบ หรือสี่เหลี่ยมทึบ ตัวชี้สามารถถูกย้ายตำแหน่งได้ หรืออาจถูกซ่อนได้แต่บางครั้งการซ่อนตัวชี้อาจทำให้ประสิทธิภาพของการแสดงผลลดลง ดังนั้นจึงมักเห็นตัวชี้ปรากฏอยู่ตลอดเวลา โปรแกรมที่ใช้จอภาพเชิงกายภาพจะไม่สามารถสอบถามตำแหน่งของตัวชี้ได้ จะต้องจำตำแหน่งของตัวชี้ไว้เอง

หน้าต่างมีคุณสมบัติดังนี้

1. หน้าต่างสามารถมีขนาดเท่ากับหรือเล็กกว่าจอภาพเชิงกายภาพ ในขณะที่มีขนาดเล็กกว่า อาจมีกรอบหรือไม่มีกรอบได้

2. ข้อมูลที่ถูกเขียนลงในหน้าต่างจะไม่ออกไปนอกขอบเขตของหน้าต่าง ซึ่งเหมือนกับที่ข้อมูลถูกเขียนลงบนจอภาพเชิงกายภาพที่ไม่สามารถเขียนออกไปนอกจอภาพได้

3. ส่วนต่างๆ ของหน้าต่าง หรืออาจเป็นทั้งหน้าต่าง อาจถูกปิดบังไว้ด้วยหน้าต่างอื่นได้ ขณะที่โปรแกรมประยุกต์เขียนข้อมูลลงในหน้าต่างไม่จำเป็นต้องคำนึงถึงกรณีเหล่านี้ข้อมูลที่ไม่ควรจะปรากฏจะถูกจัดการอย่างอัตโนมัติโดยผู้จัดการหน้าต่าง

4. ตำแหน่งที่อยู่ของหน้าต่างบนจอภาพเชิงกายภาพสามารถถูกเปลี่ยนแปลงได้โดยที่โปรแกรมประยุกต์ไม่ต้องรับทราบหรือสนใจกับสิ่งนี้ แต่โปรแกรมประยุกต์สามารถสอบถามตำแหน่งของหน้าต่างได้หากต้องการ ในทำนองเดียวกันโปรแกรมประยุกต์สามารถจะไม่สนใจหรือสอบถามขนาดของหน้าต่างได้เช่นกัน

5. เมื่อส่วนของหน้าต่างถูกเขียนใหม่เพื่อถูกทำให้มองเห็นโปรแกรมประยุกต์จำเป็นต้องสนับสนุนข้อมูลที่ต้องแสดงให้กับผู้จัดการหน้าต่าง เพราะว่าผู้จัดการหน้าต่างจะไม่จำส่วนที่ไม่ได้ถูกทำให้เห็นไว้

6. หน้าต่างมีตัวชี้เพื่อบอกตำแหน่งที่ใช้แสดงผลในหน้าต่างซึ่งสามารถย้ายถูกทำให้ปรากฏ หรือซ่อนไว้ได้

7. มีเพียงหน้าต่างบนสุดเท่านั้นที่ตัวชี้จะปรากฏให้ผู้ใช้เห็น

จอภาพเสมือนมีคุณสมบัติที่ได้เปรียบหน้าต่างอยู่ 2 ประการใหญ่คือ

1. จอภาพเสมือนสามารถมีขนาดใดๆ แม้กระทั่งใหญ่กว่าจอภาพเชิงกายภาพได้และต้องมีบางส่วนของจอภาพเสมือนปรากฏในหน้าต่างซึ่งอาจมีขนาดใดๆ จอภาพเสมือนจะมีส่วนการเลื่อนข้อมูลอัตโนมัติไปตามการเคลื่อนที่ของการชี้ตำแหน่ง ดังนั้นผู้ใช้จอภาพเสมือนจึงไม่ต้องคำนึงถึงเรื่องนี้

2. การเขียนข้อมูลของจอภาพเสมือนลงในหน้าต่างจะถูกทำโดยอัตโนมัติปัญหาเรื่องของหน้าต่างที่วางซ้อนทับกันจะหมดไป และผู้ใช้จอภาพเสมือนไม่ต้องสนับสนุนการเขียนข้อมูลลงในหน้าต่างใหม่ ซึ่งจะทำให้การเขียนโปรแกรมง่าย

คุณสมบัติข้างต้นที่ได้กล่าวมานั้นเป็นคุณสมบัติที่ได้กำหนดขึ้น เพื่อใช้ในการทำวิทยานิพนธ์นี้ อาจมีความแตกต่างไปจากโปรแกรมอื่นได้ ทั้งนี้ขึ้นอยู่กับผู้พัฒนาโปรแกรมนั้นจะกำหนดตามความเหมาะสม

การใช้งานจอภาพเชิงกายภาพนั้นอาจมีความแตกต่างกันตามชนิดของเทอร์มินัล สำหรับฟังก์ชั่นที่เตรียมไว้ เพื่อใช้งานจอภาพเชิงกายภาพในวิทยานิพนธ์นี้เป็นฟังก์ชั่นที่ใช้กับเทอร์มินัล วีที100 (VT-100)

หน้าต่างถูกกำหนดโครงสร้างในลักษณะของภาษาซีดังนี้

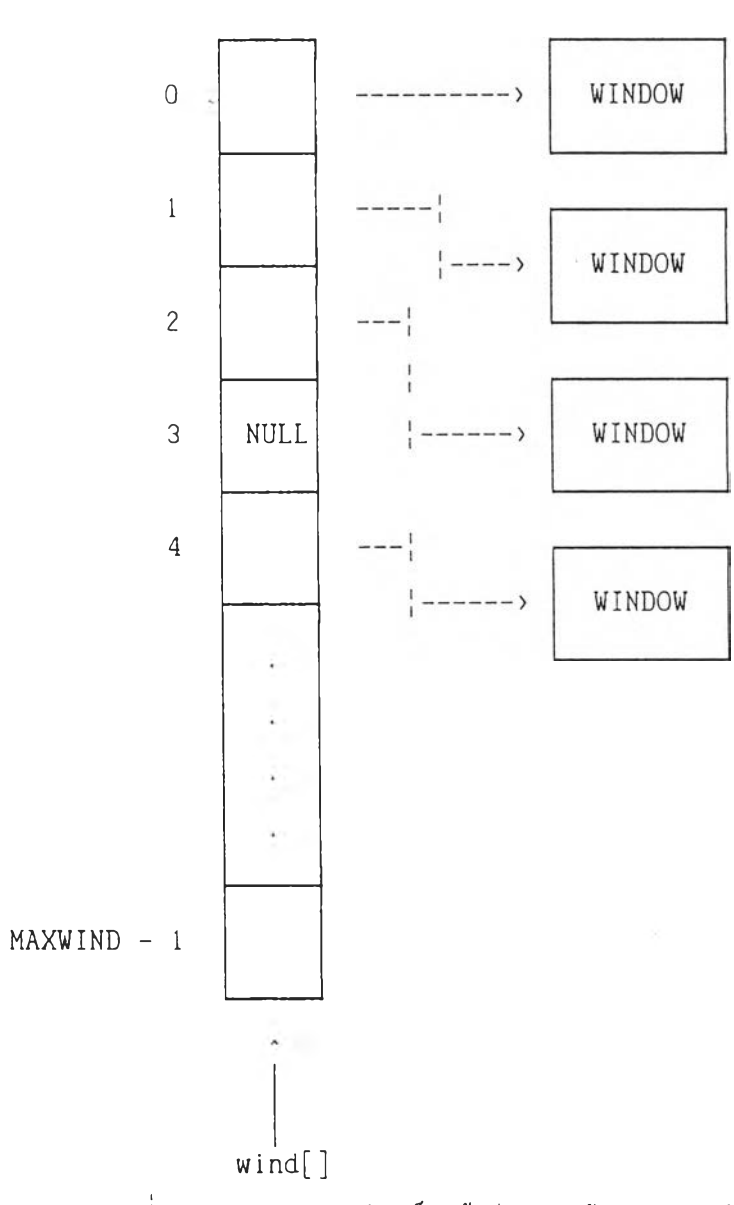
```
typedef struct {
     RECT srect;         /* physical screen rectangle */
     RECT svsrect;       /* saved rectangle (for zooming) */
     FRMTYPE frmtype;    /* frame type */
     short frmatt;       /* frame attribute */
     char *title;        /* window title */
     FRMTYPE svfrmtype; /* save frame type (for zooming) */
     BOOLEAN zoomed;     /* is window zoomed? */
     short currow;       /* current row in window */
     short curcol;       /* current column in window */
     BOOLEAN showcur;    /* is cursor is visible? */
     int (*drawfcn)();  /* redrawing function */
} WINDOW;

static WINDOW *wind[MAXWIND];
```

ส่วนโครงสร้างของจอภาพเสมือนเป็นดังนี้

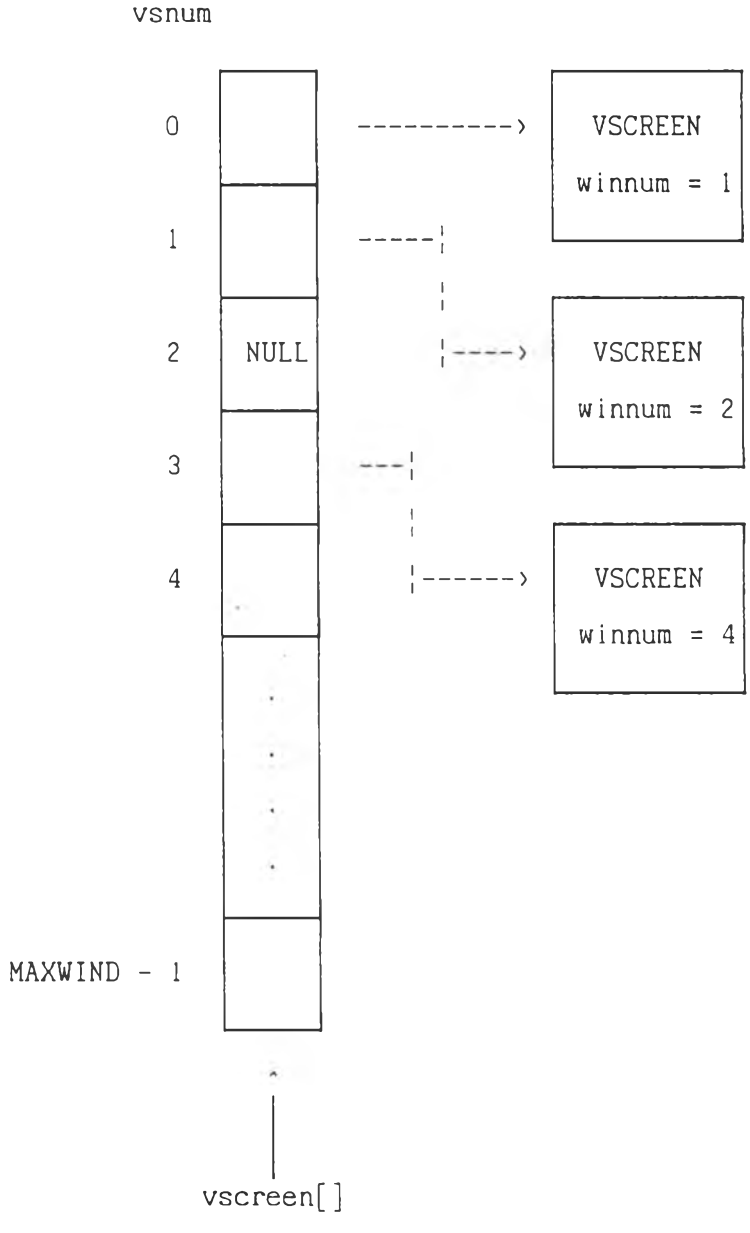
```
typedef struct {
     short winnum;  /* associated window */
     short vwinrow1; /* location of window first row */
     short vwincol1; /* location of window first column */
     short vrow2;    /* size of vscreen; last row */
     short vcol2;    /* size of vscreen; last column */
     short vcurrow;  /* cursor row */
     short vcurcol;  /* cursor column */
     CELL **ca;      /* cell array */
} VSCREEN;

static VSCREEN *vscreen[MAXVSCREEN];
```

ความสัมพันธ์ระหว่างหน้าต่างและจอภาพเสมือนถูกกำหนดไว้ในตาราง wtable ดังนี้

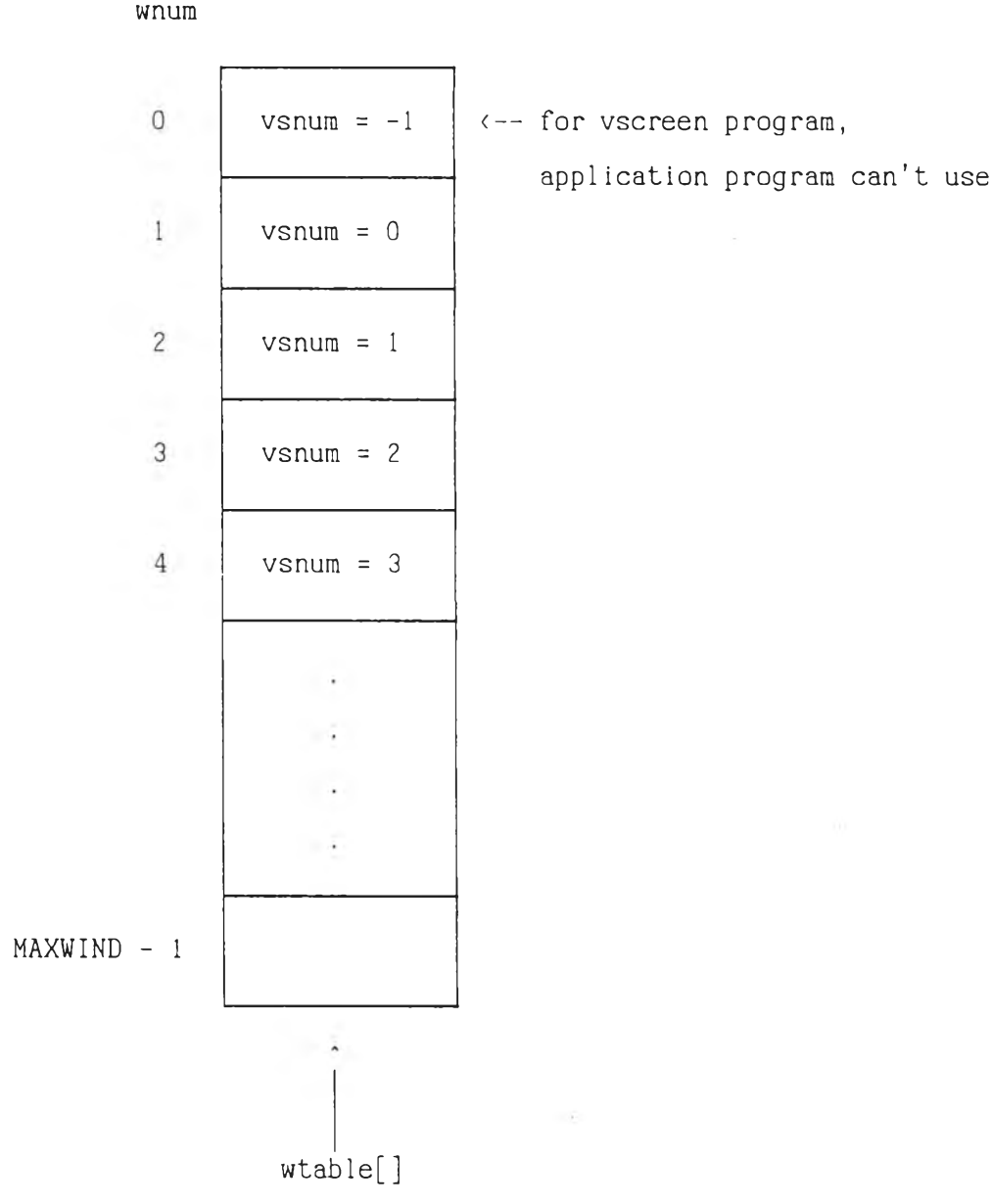
```
static wtable[MAXWIND];  /* mapping table */
                         /* window --> vscreeen number */
```



รูปที่ 3.4 แสดงการจัดเก็บหน้าต่างของตัวแปรแถวลำดับ wind



รูปที่ 3.5 แสดงการจัดเก็บจอภาพเสมือนของตัวแปรแถวลำดับ vscreen



รูปที่ 3.6 แสดงความสัมพันธ์ระหว่างหน้าต่างและจอภาพเสมือนด้วยตัวแปรแถวลำดับ wtable

จากลักษณะโครงสร้างและความสัมพันธ์ของหน้าต่างและจอภาพเสมือนที่ได้กำหนดไว้ เพื่อให้สามารถทราบได้โดยง่ายว่าหน้าต่างใดเป็นหน้าต่างสำหรับแสดงข้อมูลของจอภาพเสมือนใด หรือในทางกลับกันคือ จอภาพเสมือนใดมีหน้าต่างใดที่ใช้แสดงข้อมูลของจอภาพเสมือนนั้นๆ จึงได้กำหนดฟังก์ชันการแปลงส่ง (mapping function) ไว้ดังนี้

```
/*  get window's vitual screen number */
int VSgetvsnum( int wnum )
{
     return (wtable[wnum]);
}

/*  get virtual screen's window number */
int VSgetwnum( int vsnum )
{
     return (vscreen[vsnum]->winnum);
}
```

4. ตัวประสานของจอภาพเชิงกายภาพ หน้าต่าง และจอภาพเสมือน

เนื่องจากการออกแบบส่วนแสดงผลได้ออกแบบแยกเป็นส่วนๆ คือ ส่วนของจอภาพเชิงกายภาพ ส่วนของหน้าต่าง และส่วนของจอภาพเสมือน ดังนั้นจึงจำเป็นต้องมีตัวประสานเพื่อเชื่อมโยงการใช้งานแต่ละส่วนเข้าด้วยกัน ตัวประสานของจอภาพเชิงกายภาพ หน้าต่าง และจอภาพเสมือน ได้แก่ฟังก์ชันต่อไปนี้

ฟังก์ชันสำหรับการประสานจอภาพเชิงกายภาพ:-

```
void PSbegin()          - initialize display
void PSend()            - terminate display
void PSsynch()          - bring screen up to date
void PSheight()         - get height of physical screen
void PSwidth()          - get width of physical screen
void PSwrite()          - write string
void PSwrtcells()       - write vector of CELLs
void PSfill()           - fill a rectangle
BOOLEAN PSslide()       - slide a rectangle
void PSsetcur()         - set cursor position
void PSshowcur()        - turn cursor on or off
void PSbeep()           - sound bell
```

ฟังก์ชันสำหรับการประสานหน้าต่าง:-

```
static void dimensions()    - get dimension of window
static short order[MAXWIND]; - order of windows,
                               bottom to top
static  short numwinds;     - number of windows in
                               order array
#define get_top()  (order[numwinds-1])  - get number of
                               top window
int Wgettop()               - get top window
static void addwind()       - add window to top of order
                               array, if necessary
static void remwind()       - remove window from order array
static BOOLEAN hidden()     - determine if window is hidden
void Wtop()                 - force window to top
void Whide()                - hide window
void Wshuffle()             - force next window to top
static void allocmask()     - allocate mask rectangle
static void reset()         - recalculate mask rectangle
static void setmask()       - fill mask rectangle with
                               window number
void Wwrite()               - write string
void Wwrtcells()            - write vector of CELLs
static int draw()           - draw all windows in physical
                               screen rectangle
```

```
static void drawwind()      - draw one window's frame and
                                interior
void Wsetphys()             - set window's physical size
void Wgetphys()             - get window's physical size
void Wzoom()                - make window to full screen
void Wunzoom()              - restore window to former size
BOOLEAN Wiszoomed()         - determine if window is zoomed
static void hline()         - draw horizontal line
static void vline()         - draw vertical line
static void drawfrm()       - draw complete or partial
                                frame around window
static int draw0()          - redrawing function for
                                window zero
void Wend()                 - terminate window module
int Wnew()                  - create new window
void Wdispose()             - eliminate window
void Wfill()                - fill rectangle
BOOLEAN Wslide()            - slide rectangle
void Wsetcur()              - set cursor position
void Wshowcur()             - turn cursor on or off
```

ฟังก์ชันสำหรับการประสานจอภาพเสมือน:-

void VSbegin()	- initialize virtual screen module
void VSend()	- terminate virtual screen module
int VSnew()	- create virtual screen
void VSdispose()	- eliminate virtual screen and window
int VSgetwnum()	- get virtual screen's window number
int VSgetvsnum()	- get window's virtual screen number
void VSgetwloc()	- get window's location relative to virtual screen
void VSpan()	- move window relative to virtaul screen
void VSsetcur()	- set cursor position relative to virtual screen
void VSshowcur()	- turn cursor on or off
void VSgetcur()	- get cursor position relative to virtual screen
void VSgetsize()	- get size of virtual screen
void VSwrite()	- write string
void VSfill()	- fill rectangle
void VSslide()	- slide rectangle