บทที่ 3

โปรแกรมบรรณาธิกร vi

vi เป็นโปรแกรมบรรณาธิกรสำหรับอำนวยความสะดวกให้แก่ผู้ใช้ ในด้านการจัดการเกี่ยวกับการเก็บข้อความในแฟ้มข้อมูล โดยทำงานในลักษณะโต้ตอบกับผู้ใช้ รูปแบบของข้อความที่ปรากฏให้เห็นบนจอภาพ คือรูปแบบสุดท้ายของเอกสารที่ได้รับ เมื่อกล่าวถึงโปรแกรมดังกล่าว จะหมายถึงโปรแกรมบรรณาธิกรข้อความ (Text Editor) ซึ่งทำงานเกี่ยวกับแฟ้มข้อมูลโดยเก็บข้อความในลักษณะของกลุ่มตัวอักษร

โปรแกรมนี้แบ่งตามลักษณะการทำงานออกเป็น 2 ประเภทคือ การบรรณาธิกรบรรทัด (line editor) และการบรรณาธิกรทั้งจอภาพ (screen editor) โดยที่การบรรณาธิกรบรรทัดจะทำงานเป็นหน่วยของบรรทัด ในขณะหนึ่งๆ คำสั่งที่เรียกใช้จะทำงานเกี่ยวข้องกับข้อความเพียงบรรทัดเดียว เช่น การแทรกข้อความบรรทัดใหม่ ลบข้อความหนึ่งบรรทัด แสดงข้อความที่ต้องการทางจอภาพ หรือ การคัดลอกข้อความ เหล่านี้เป็นต้น การบรรณาธิกรทั้งจอภาพที่สามารถแสดงข้อความครั้งละหลายบรรทัด โดยการนำข้อความซึ่งเป็นส่วนหนึ่งที่เก็บในแฟ้มข้อมูลแสดงทางจอภาพครั้งละหนึ่งจอภาพ และสามารถแก้ไขเปลี่ยนแปลงข้อความที่ปรากฏบนจอภาพขณะนั้นได้โดยการใช้เคอร์เซอร์ (cursor) ซึ่งเป็นตัวชี้บอกตำแหน่งขณะนั้นเลื่อนไปยังตำแหน่งที่ต้องการ และทำการแก้ไขต่อไป นอกจากนี้สามารถกำหนดให้ข้อความในส่วนต่างๆ ของแฟ้มข้อมูลแสดงทางจอภาพได้ การบรรณาธิกรทั้งจอภาพนี้บางครั้งเรียกว่า display editor หรือ visual editor

อย่างไรก็ตามทั้งการบรรณาธิกรบรรทัดและการบรรณาธิกรทั้งจอภาพ จะมีพื้นฐานการทำงานเหมือนกัน คือใช้บัฟเฟอร์ (buffer) เป็นที่เก็บชั่วคราวสำหรับการแก้ไขหรือสร้างข้อความใหม่เมื่อแก้ไขเปลี่ยนแปลงข้อความแล้ว จะต้องกำหนดให้เก็บข้อความเหล่านั้นไว้ในแฟ้มข้อมูลอีกครั้งหนึ่ง ในการทำงานลักษณะนี้มีทั้งผลดีและผลเสีย คือกรณีที่แก้ไขข้อความต่างๆ ผิดจากที่ต้องการ สามารถยกเลิกข้อความเหล่านั้นได้โดยไม่ต้องเก็บข้อความที่เปลี่ยนแปลงไว้ในแฟ้มข้อมูล ดังนั้นสามารถใช้ข้อความเดิมในแฟ้มข้อมูลต่อไปได้ แต่ในกรณีที่เครื่องขัดข้องหรือหยุดกระทันหันขณะที่กำลังใช้โปรแกรม ข้อความทั้งหมดในบัฟเฟอร์จะหายไปด้วย ในกรณีนี้สามารถป้องกันได้โดยการเรียกใช้คำสั่งสำหรับเก็บข้อความไว้ในแฟ้มข้อมูลบ่อยๆ ในขณะที่ใช้โปรแกรมบรรณาธิกรเพื่อป้องกันการสูญหายของข้อความที่เก็บในบัฟเฟอร์ด้วยสาเหตุดังกล่าว

3.1 โปรแกรมบรรณาธิการพื้นฐานในระบบปฏิบัติการยูนิกซ์

เนื่องจากโปรแกรมบรรณาธิการพื้นฐานในระบบปฏิบัติการยูนิกซ์เป็นประเภทบรรณาธิกรบรรทัด ซึ่งเป็นโปรแกรมที่ใช้งานไม่สะดวกนัก ดังนั้นในปัจจุบันจึงได้มีการพัฒนาโปรแกรมที่ให้ความสะดวกแก่ผู้ใช้มากขึ้น ซึ่งต่างก็มีพื้นฐานมาจาก ed ทั้งสิ้น โปรแกรมดังกล่าวที่นิยมใช้กันในปัจจุบันเช่น ex และ vi ของ Berkeley และ rand ซึ่งพัฒนาขึ้นโดยบริษัทแรนด์ (Rand Corporation)[1] เป็นต้น ในที่นี้จะกล่าวถึงเฉพาะ vi ซึ่งปัจจุบันใช้เป็นโปรแกรมบรรณาธิกรมาตรฐานในระบบปฏิบัติการยูนิกซ์

3.2 vi

vi พัฒนาขึ้นโดย William Joy แห่ง University of California at Berkeley[2] เพื่อใช้ภายใต้ระบบปฏิบัติการยูนิกซ์ โดยมีพื้นฐานมาจาก ed บางครั้งจะเรียกว่าโปรแกรมบรรณาธิกรบนจอภาพ หรือ display editor โดยเรียกตามลักษณะการทำงาน นั่นคือในการเรียกใช้ vi นี้ จะแสดงข้อความให้เห็นครั้งละหนึ่งจอภาพ ซึ่งในที่นี้จะเรียกว่าหน้าต่าง ภายในหน้าต่างจะมีเครื่องหมายชี้บอกตำแหน่งตัวอักษรขณะนั้น ซึ่งเรียกว่าเคอร์เซอร์ สามารถเลื่อนไปยังตำแหน่งที่ต้องการภายในหน้าต่างได้ ตำแหน่งที่เคอร์เซอร์ชี้ขณะใดๆ จะเรียกว่าตำแหน่งปัจจุบัน การแทรกข้อความจะทำที่ตำแหน่งปัจจุบันเสมอ

3.3 ลักษณะสำคัญของ vi

3.3.1 บัฟเฟอร์ (Buffer) เป็นที่ใช้ชั่วคราวสำหรับเก็บข้อความขณะทำงานของ vi โดยเฉพาะงานที่เกี่ยวกับการสร้าง และการแก้ไขข้อความภายในบัฟเฟอร์ ก่อนที่จะนำข้อความ

---

1. Topham D. W., Truong H. V., UNIX and XENIX, A Step-By-Step, Brady Communications Company, Inc., 1985, pp. 196

2. McGilton H., Morgan R., Introducing the UNIX System, McGraw-Hill Book Company, 1983, pp. 235

ในช่วงของการแก้ไขจะต้องออกก่อน โดยการใช้ปุ่ม ESC หลังจากนั้นจึงจะเรียกใช้คำสั่งอื่นต่อไปได้

3.3.3 รูปแบบคำสั่ง (Command Structure) การทำงานภายใต้ vi จะต้องกำหนดคำสั่งอย่างชัดเจน คำสั่งของ vi จะไม่แสดงทางจอภาพแต่จะรับเข้าไปใช้งานทันที คำสั่งของ vi มี องค์ประกอบต่อไปนี้ทั้งหมดหรือบางส่วน

- ตัวอักษรเพียงตัวเดียว
- อักษรควบคุมต่างๆ (control character)
- ตัวเลขบอกจำนวนซ้ำ และคำสั่งปกติของ vi

ในกรณีที่เรียกใช้คำสั่งที่ไม่ถูก หรือถูกแต่ vi ไม่สามารถทำงานตามคำสั่งในขณะนั้นได้ จะมีเสียงเตือนให้ทราบเสมอ

ปุ่มที่สำคัญของ vi คือ ESC นอกจากจะเป็นปุ่มที่ใช้สำหรับออกจากช่วงของการแก้ไขข้อความตามที่กล่าวมาแล้วข้างต้น ยังสามารถใช้ในความหมายอื่นได้อีก เช่น ในกรณีที่เริ่มใช้คำสั่งใดๆ แต่ต้องการยกเลิกในขณะที่กำหนดคำสั่งนั้นไม่สมบูรณ์ สามารถกดปุ่ม ESC เพื่อยกเลิกคำสั่งนั้น ในที่นี้ ESC จะเป็นปุ่มที่ปลอดภัยสำหรับการยกเลิกหรือป้องกันการทำงานของ vi หากมีความผิดพลาดของการพิมพ์คำสั่งหรือการใช้คำสั่งผิดความประสงค์ นอกจากนี้สามารถใช้ในกรณีที่ไม่แน่ใจว่าขณะนั้นกำลังอยู่ในช่วงใดโดยการกด ESC หากเกิดเสียงดังขึ้น หมายความว่า vi กลับเข้าวสู่ช่วงของการรอคำสั่งตามปกติ

3.3.4 รูปแบบคำสั่งที่ต่างจากปกติ คำสั่งบางตัวมีวิธีการใช้ต่างจากที่กล่าวมาแล้ว คือจะต้องเริ่มด้วยตัวอักษร ':' ก่อนเสมอ และคำสั่งพวกนี้จะปรากฏให้เห็นทางจอภาพด้วย โดยที่เมื่อเรียกใช้ อักษร ':' จะปรากฏที่บรรทัดสุดท้ายของจอภาพซึ่งเรียกว่าเป็นบรรทัดแสดงสถานภาพ (status line) และรอรับคำสั่งที่เกี่ยวข้องต่อไปจนกระทั่งกดปุ่มรีเทอร์น (return) จึงนำคำสั่งเหล่านั้นไปแปลและทำงานต่อไป เช่น คำสั่งสำหรับให้เขียนข้อความของบัฟเฟอร์เก็บในแฟ้มข้อมูลที่เรียกใช้ ซึ่งกำหนดในรูปแบบดังต่อไปนี้

```
:w <return>
```

3.4 ลักษณะงานที่ทำของ vi

3.4.1 การย้ายเคอร์เซอร์ การย้ายเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ ทำได้หลายลักษณะ เช่นการย้ายเคอร์เซอร์ไปทางซ้ายหรือขวา หรือบรรทัดบนล่าง ซึ่งในการย้ายเคอร์เซอร์สามารถกำหนดหน่วยหรือขนาดของการย้ายในแต่ละครั้งตามต้องการได้ เช่นการย้าย

ครั้งละตัวอักษร คำ บรรทัด ย่อหน้าหรือครั้งละหน้าต่าง โดยเรียกใช้ตามรูปแบบของคำสั่ง vi ตามที่กล่าวแล้วข้างต้น คำสั่งที่ใช้ในการย้ายเคอร์เซอร์ในลักษณะต่างๆ มีดังนี้

3.4.1.1 การย้ายเคอร์เซอร์เป็นตัวอักษร คำสั่งที่ใช้คือ

'h' ย้ายเคอร์เซอร์ไปทางซ้ายของตำแหน่งเดิมครั้งละหนึ่งตัวอักษร

'l' ย้ายเคอร์เซอร์ไปทางขวาของตำแหน่งเดิมครั้งละหนึ่งตัวอักษร

นอกจากนี้ในกรณีที่แป้นพิมพ์มีปุ่มลูกศร สามารถใช้ปุ่มเหล่านี้ในการย้ายเคอร์เซอร์ได้ด้วย ซึ่งลักษณะการทำงานจะเหมือนกับการทำงานของการเรียกใช้คำสั่งด้วยตัวอักษร 'h' 'l' คือ

'->' เรียกใช้เหมือนปุ่มอักษร 'l'

'<-' เรียกใช้เหมือนปุ่มอักษร 'h'

3.4.1.2 การย้ายเคอร์เซอร์เป็นบรรทัด

'j' ย้ายเคอร์เซอร์ลงหนึ่งบรรทัดในตำแหน่งตรงกันกับตำแหน่งปัจจุบัน

'k' ย้ายเคอร์เซอร์ขึ้นหนึ่งบรรทัดในตำแหน่งตรงกันกับตำแหน่งปัจจุบัน

'↑' เรียกใช้เหมือนการเรียกปุ่มอักษร 'k'

'↓' เรียกใช้เหมือนการเรียกปุ่มอักษร 'j'

'return' ย้ายเคอร์เซอร์จากตำแหน่งปัจจุบันไปยังตัวอักษรตัวแรกของบรรทัดถัดไป

'-' ย้ายเคอร์เซอร์จากตำแหน่งปัจจุบันไปยังตัวอักษรตัวแรกของบรรทัดก่อน

การย้ายเคอร์เซอร์ในทางซ้ายหรือขวาด้วยคำสั่งดังกล่าว เมื่อเคอร์เซอร์ถึงจุดเริ่มต้นหรือจุดสุดท้ายของบรรทัดแล้ว จะไม่ย้ายไปบรรทัดถัดไปหากไม่มีการใช้คำสั่งโดยชัดเจน

การย้ายเคอร์เซอร์ขึ้นหรือลงนั้น ตามปกติจะย้ายไปยังตำแหน่งที่ตรงกันในบรรทัดที่ต้องการ หากบรรทัดที่ย้ายไปนั้นมีข้อความน้อยกว่าตำแหน่งที่เคอร์เซอร์ควรจะปรากฏ ณ ตำแหน่งนั้น เคอร์เซอร์จะชี้ที่ตัวอักษรสุดท้ายของบรรทัดนั้นแทน

การย้ายเคอร์เซอร์ไปยังข้อความที่ไม่ปรากฏบนหน้าต่าง vi จะเปลี่ยนข้อความที่แสดงบนหน้าต่างใหม่

3.4.1.3 การย้ายเคอร์เซอร์ในหน้าเดียวกัน

'H' ย้ายเคอร์เซอร์ไปยังบรรทัดแรกของข้อความบนหน้าต่าง

'M' ย้ายเคอร์เซอร์ไปยังบรรทัดกลางของข้อความบนหน้าต่าง

'L' ย้ายเคอร์เซอร์ไปยังบรรทัดสุดท้ายของข้อความบนหน้าต่าง

3.4.1.4 การย้ายเคอร์เซอร์เป็นคำ

'w' ย้ายเคอร์เซอร์ไปยังตัวอักษรแรกของคำถัดไป

'b' ย้ายเคอร์เซอร์ไปยังตัวอักษรแรกของคำก่อนหน้า

'e' ย้ายเคอร์เซอร์ไปยังตัวอักษรสุดท้ายของคำถัดไป

3.4.1.4 การย้ายเคอร์เซอร์เป็นประโยคและย่อหน้า

')' ย้ายเคอร์เซอร์ไปยังจุดเริ่มต้นของประโยคถัดไป

'(' ย้ายเคอร์เซอร์ไปยังจุดเริ่มต้นของประโยคปัจจุบัน

'}' ย้ายเคอร์เซอร์ไปยังจุดเริ่มต้นของย่อหน้าถัดไป

'{' ย้ายเคอร์เซอร์ไปยังจุดเริ่มต้นของย่อหน้าปัจจุบัน

ในการใช้คำสั่งสำหรับย้ายเคอร์เซอร์ตามลักษณะที่กล่าวมาทั้งหมด ถ้าต้องการย้ายเคอร์เซอร์ในลักษณะเดียวกันมากกว่าหนึ่งครั้ง สามารถบอกจำนวนของการย้ายพร้อมกับการเรียกใช้คำสั่งเหล่านั้นได้เช่น

'3w' ย้ายเคอร์เซอร์ไปข้างหน้า 3 คำ

'10h' ย้ายเคอร์เซอร์ไปทางซ้าย 10 ตัวอักษร

3.4.1.4 การย้ายเคอร์เซอร์ไปยังตัวอักษรที่กำหนด

'fx' ย้ายเคอร์เซอร์ไปที่อักษร 'x' ตัวแรกถัดจากตำแหน่งปัจจุบันในทิศทางข้างหน้า

'Fx' เหมือนกับ 'fx' แต่ในทิศทางตรงกันข้าม

3.4.1.5 การย้ายหน้าต่าง ในกรณีที่ต้องการดูข้อความในบัฟเฟอร์ซึ่งไม่ปรากฏบนหน้าต่างขณะนั้น สามารถทำได้โดยการใช้คำสั่งที่ทำงานเปรียบเสมือนกับการย้ายหน้าต่าง ไปยังข้อความในบัฟเฟอร์ส่วนที่ต้องการ ซึ่งคำสั่งเหล่านี้คือ

control-D ย้ายหน้าต่างครั้งละครึ่งหน้าต่างไปยังข้อความในทิศทางข้างหน้า

control-U ย้ายหน้าต่างครั้งละครึ่งหน้าต่างไปยังข้อความในทิศทางย้อนกลับ

control-F ย้ายหน้าต่างครั้งละหนึ่งหน้าต่างไปยังข้อความในทิศทางข้างหน้า

control-B ย้ายหน้าต่างครั้งละหนึ่งหน้าต่างไปยังข้อความในทิศทางย้อนกลับ

3.4.2 <u>การสร้างแฟ้มข้อมูลใหม่</u> (Creating a new File) ทำได้ 2 วิธีคือ โดยการเรียกใช้คำสั่ง vi พร้อมกับกำหนดชื่อแฟ้มข้อมูลที่ต้องการให้เก็บข้อความไว้ และการสร้างโดยการใช้คำสั่งสำหรับเขียนข้อความลงในแฟ้มข้อมูลของ vi ในขณะที่ใช้ vi

ในกรณีที่เรียกใช้ชื่อแฟ้มข้อมูลใหม่ เมื่อเข้าสู่ช่วงของการรับคำสั่ง vi จะแสดงให้ทราบว่าเป็นแฟ้มข้อมูลใหม่ และแสดงข้อความบนหน้าต่างที่มีลักษณะดังนี้

~

~

~

~

~

"ชื่อแฟ้มข้อมูล" [New File]

เคอร์เซอร์จะปรากฏที่มุมหน้าต่างด้านบนซ้าย และที่บรรทัดสำหรับแสดงสถานะ จะมีข้อความแสดงว่าเป็นแฟ้มข้อมูลใหม่ เมื่อต้องการจะพิมพ์ข้อความในบัฟเฟอร์จะต้องเรียกใช้คำสั่งสำหรับพิมพ์ข้อความดังที่จะกล่าวต่อไป

ในกรณีที่ต้องการสร้างแฟ้มข้อมูลใหม่ด้วยคำสั่งของ vi จะใช้คำสั่งดังนี้

':w ชื่อแฟ้มข้อมูล'

3.4.3 <u>การลบข้อความ</u> ทำได้หลายลักษณะโดยการใช้คำสั่งดังนี้ คือ

'x' ลบข้อความครั้งละหนึ่งตัวอักษร โดยที่จะลบตัวอักษรที่เคอร์เซอร์ขณะนั้น แล้วเลื่อนตัวอักษรถัดไปมาแทนที่ ในกรณีที่เป็นตัวอักษรสุดท้ายของบรรทัด จะลบตัวอักษรนั้นและเลื่อนเคอร์เซอร์ไปทางซ้ายหนึ่งตำแหน่งเสมอ จะไม่ลบข้อความในบรรทัดถัดไป

นอกจากนี้ยังมีคำสั่งสำหรับลบข้อความครั้งละหลายตัวอักษร เช่นลบเป็นจำนวนคำ ประโยค หรือ บรรทัด เช่น

'dw' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดคำขณะนั้น

'db' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นของคำ

'd)' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดประโยค

'd(' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นประโยค

'd}' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดย่อหน้า

'd{' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นย่อหน้า

'dd' ลบข้อความทั้งบรรทัด

'd$' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดบรรทัด

'd0' ลบตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นบรรทัด

'D' มีความหมายเหมือน 'd$'

3.4.4 <u>การเปลี่ยนข้อความ</u> (Changing Text) การเปลี่ยนข้อความเก่าเริ่มที่ตำแหน่งเคอร์เซอร์และสิ้นสุดที่ตำแหน่งที่กำหนด ด้วยข้อความใหม่ที่พิมพ์เพิ่มขึ้นและสิ้นสุดด้วย ESC

คำสั่งเปลี่ยนข้อความคล้ายกับคำสั่งลบข้อความ คือสามารถกำหนดให้เปลี่ยนเป็นจำนวนตัวอักษร คำ ประโยค หรือย่อหน้าได้เช่นเดียวกัน คือ

'cc' เปลี่ยนข้อความเก่าทั้งบรรทัด

'cw' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดคำขณะนั้น

'cb' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นของคำ

'c)' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดประโยค

'c(' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นประโยค

'c}' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดย่อหน้า

'c{' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นย่อหน้า

'cc' เปลี่ยนข้อความทั้งบรรทัด

'c$' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดบรรทัด

'c0' เปลี่ยนข้อความตั้งแต่ตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นบรรทัด

'C' มีความหมายเหมือน 'c$'

การเปลี่ยนข้อความในลักษณะที่กล่าวมานี้ ข้อความเก่าและข้อความใหม่ไม่จำเป็นต้องมีจำนวนตัวอักษรเท่ากัน แต่จะทำการแทนที่ข้อความในขอบเขตที่กำหนด ด้วยข้อความใหม่ที่พิมพ์เข้าไปจนกระทั่งกดปุ่ม ESC ดังนั้นข้อความใหม่อาจมีความยาวมากกว่าหรือน้อยกว่าข้อความเดิมได้

การเปลี่ยนข้อความในอีกลักษณะคือ การเปลี่ยนเป็นจำนวนตัวอักษร โดยจะเปลี่ยนเฉพาะตำแหน่งที่เคอร์เซอร์ปรากฏอยู่ขณะนั้นเท่านั้น จำนวนตัวอักษรของข้อความเดิมที่เปลี่ยนและข้อความใหม่จะต้องเท่ากัน คำสั่งที่ใช้เปลี่ยนในลักษณะนี้คือ

'r' เปลี่ยนข้อความครั้งละหนึ่งตัวอักษร โดยจะเปลี่ยนข้อความที่เคอร์เซอร์ชี้อยู่ด้วยตัวอักษรที่ตามหลังคำสั่งนี้ หลังจากนั้น vi จะรอรับคำสั่งอื่นต่อไป

'R' เปลี่ยนอักษรที่เคอร์เซอร์ชี้ด้วยอักษรที่พิมพ์ใหม่ครั้งละหนึ่งตัวอักษรพร้อมย้ายเคอร์เซอร์ไปทางขวาเพื่อเปลี่ยนตัวอักษรถัดไป จนกระทั่งสิ้นสุดคำสั่งนี้ด้วย ESC

3.4.5 <u>การค้นหาข้อความที่ต้องการ</u> (Searching for String) การค้นหาข้อความที่กำหนด vi จะทำการค้นหาข้อความที่ต้องการแล้วย้ายเคอร์เซอร์ไปยังตัวอักษรตัวแรกของข้อความที่หาได้ ถ้าไม่มีข้อความนั้น จะแสดงให้ทราบว่าไม่มีข้อความนั้นในบัฟเฟอร์

คำสั่งในการค้นหาข้อความจะต่างจากคำสั่งโดยทั่วไปคือ จะปรากฏให้เห็นคำสั่งและข้อความที่ต้องการให้ค้นหาที่บรรทัดสุดท้ายของหน้าต่าง รูปแบบของคำสั่งที่ใช้คือ

'/ข้อความที่ต้องการค้นหา'

ในกรณีที่ใช้เครื่องหมาย '/' สำหรับค้นหานั้น vi จะทำการค้นหาตั้งแต่บรรทัดขณะนั้นไปจนสิ้นสุดบัฟเฟอร์ แต่ถ้าต้องการค้นหาในทิศทางย้อนกลับ จะต้องใช้เครื่องหมาย '?' แทนเครื่องหมาย '/'

หลังจากใช้คำสั่งนี้แล้วครั้งหนึ่ง สามารถกำหนดให้หาข้อความเดิมซ้ำโดยไม่ต้องพิมพ์ข้อความนั้นใหม่ โดยการใช้คำสั่งดังนี้

'n' สำหรับค้นหาข้อความเดิมไปในทิศทางเดียวกันกับคำสั่งค้นหาครั้งสุดท้าย

'N' สำหรับค้นหาข้อความเดิมไปในทิศทางตรงกันข้ามกับคำสั่งค้นหาครั้งสุดท้าย

คำสั่งสำหรับค้นหาข้อความของ vi สามารถทำงานในลักษณะต่างๆ ได้อย่างกว้างขวางมาก ผู้ใช้สามารถกำหนดตำแหน่งที่ปรากฏของข้อความเหล่านั้นในบัฟเฟอร์ เพื่อเป็นการกำหนดขอบเขตของการค้นหาให้แคบลงได้ ลักษณะการกำหนดคุณสมบัติพิเศษของการค้นหา ในกรณีนี้เช่น

'^' ค้นหาข้อความบรรทัดที่ขึ้นต้นด้วยข้อความที่กำหนด เช่น

/^the ค้นหาบรรทัดที่ขึ้นต้นด้วยคำว่า the

'$' ค้นหาบรรทัดที่สิ้นสุดด้วยคำที่กำหนด ความหมายคล้ายกับการใช้ '^' โดยที่เป็นการหาที่จุดสิ้นสุดของบรรทัด เช่น

/$es กำหนดให้หาบรรทัดที่มีข้อความ 'es' เป็นข้อความสุดท้าย

'.' แทนตัวอักษรใดๆ เช่น

/l..e หาข้อความที่ประกอบด้วยตัวอักษร 'l' และตัวอักษรใดๆ อีก 2 ตัวและตามด้วยอักษร 'e' ดังนั้นในกรณีนี้ สามารถหาข้อความได้หลายข้อความ เช่น line, followed, include, replace ฯลฯ

'\<' และ '\>' ค้นหาข้อความที่เริ่มต้นและสิ้นสุดคำตามลำดับ ใช้สำหรับการค้นหาข้อความที่ขึ้นต้นหรือสิ้นสุดคำด้วยข้อความที่กำหนดหลังเครื่องหมายนี้ เช่น

/\<the หาคำที่ขึ้นต้นด้วย 'the'

/\>ed หาคำที่จบด้วย 'ed'

'[]' กำหนดกลุ่มของตัวอักษรใดๆ และเลือกใช้เพียงตัวเดียว เช่น

/dis[ck] หาข้อความที่สะกดด้วย 'c' หรือ 'k' ซึ่งเป็นไปได้ทั้ง 'disc' และ 'disk'

การค้นหาและแทนที่ด้วยข้อความชุดใหม่ เป็นคำสั่งที่ประกอบด้วยคำสั่งค้นหาและคำสั่งเปลี่ยนแปลงข้อความ vi จะทำการค้นหาข้อความที่กำหนดด้วยเครื่องหมาย '/' ดังกล่าว และเปลี่ยนเป็นข้อความใหม่ที่กำหนดให้โดยมีรูปแบบของคำสั่ง คือ

:[address]s/search-string/replace-string/[/g] โดยที่

address คือขอบเขตของข้อความที่ต้องการหา โดยกำหนดเป็นหมายเลขบรรทัดของข้อความเหล่านั้น สามารถกำหนดด้วยหมายเลขบรรทัดเพียงบรรทัดเดียว หรือช่วงของบรรทัดก็ได้ นอกจากกำหนดด้วยตัวเลขดังกล่าวแล้ว สามารถใช้ตัวอักษรพิเศษได้คือ

'.' ใช้แทนบรรทัดขณะนั้น

'$' ใช้แทนบรรทัดสุดท้ายของบัฟเฟอร์

นอกจากนี้สามารถใช้เครื่องหมาย +,- เพื่อกำหนดขอบเขตของบรรทัดที่ต้องการหา โดยกำหนดเป็นค่าสัมพัทธ์กับหมายเลขบรรทัดใดๆ

ตัวอย่างการใช้	ความหมาย
5	บรรทัดที่ 5
77,100	ตั้งแต่บรรทัดที่ 77 ถึงบรรทัดที่ 100
1,.	ตั้งแต่บรรทัดแรกถึงบรรทัดขณะนั้น
.,$	ตั้งแต่บรรทัดขณะนั้นถึงบรรทัดสุดท้าย
%	ตั้งแต่บรรทัดแรกถึงบรรทัดสุดท้าย
.,.+10	ตั้งแต่บรรทัดขณะนั้นไปอีก 10 บรรทัด

search-string และ replace-string หมายถึงข้อความที่กำหนดให้หา และข้อความใหม่ที่ใช้แทนที่ตามลำดับ

3.4.6 <u>คำสั่งทำซ้ำ</u> คำสั่งที่ใช้คือ

'.' ทำสั่งคำสั่งสุดท้ายที่ทำให้เกิดการเปลี่ยนแปลงของข้อความภายในบัฟเฟอร์ เช่น ถ้าใช้ 'dd' เป็นคำสั่งสุดท้าย เมื่อเรียกใช้คำสั่ง '.' จะทำการลบข้อความอีกหนึ่งบรรทัด

3.4.7 <u>คำสั่งยกเลิกการแก้ไขข้อความ</u> คำสั่งที่ใช้คือ

'u' สำหรับยกเลิกการทำงานของคำสั่งสุดท้ายที่แก้ไขข้อความ คำสั่งนี้มีประโยชน์และใช้มากในกรณีที่ทำการเปลี่ยนแปลงแก้ไขข้อความในบัฟเฟอร์แล้ว และต้องการยกเลิกเพื่อนำข้อความเดิมคืนมา

3.4.8 คำสั่งรวมข้อความในบรรทัดที่ติดกันเข้าด้วยกัน (Join Command) คำสั่งที่ใช้คือ

'J' รวมข้อความในบรรทัดที่ติดกันเข้าด้วยกัน โดยใช้บรรทัดที่มีเคอร์เซอร์เป็นหลัก แล้วนำข้อความในบรรทัดถัดไปมาต่อ ใช้ช่องว่างเป็นตัวแยกข้อความในสองบรรทัด เคอร์เซอร์จะชี้ที่ช่องว่างนั้น

3.4.9 คำสั่งแสดงสถานะของแฟ้มข้อมูล (Status) คำสั่งที่ใช้คือ

':f' หรือ 'control-G' สำหรับแสดงสถานะต่างๆ ของแฟ้มข้อมูลที่กำลังใช้งาน โดยแสดงที่บรรทัดสุดท้ายของหน้าต่าง ซึ่งรายละเอียดที่แสดงคือ ชื่อแฟ้มข้อมูล หมายเลขบรรทัดปัจจุบัน จำนวนบรรทัดและจำนวนตัวอักษรทั้งหมด

3.4.10 คำสั่งคัดลอกข้อความ มีคำสั่งที่เกี่ยวข้อง 2 คำสั่งคือ 'yank' และ 'put'

3.4.10.1 yank คัดลอกข้อความที่ต้องการเก็บไว้ในบัฟเฟอร์ทั่วไป เพื่อนำไปใช้งานด้วยคำสั่ง put คำสั่ง yank สามารถกำหนดจำนวนหน่วยของข้อความที่ต้องการคัดลอกในลักษณะเดียวกันกับการกำหนดสำหรับการลบข้อความหรือเปลี่ยนข้อความ จะกำหนดด้วยคำสั่งต่อไปนี้

'yy' คัดลอกข้อความหนึ่งบรรทัด

'yw' คัดลอกข้อความหนึ่งคำ

'y)' คัดลอกข้อความจากตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดประโยค

'y(' คัดลอกข้อความจากตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นประโยค

'y}' คัดลอกข้อความจากตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดย่อหน้า

'y{' คัดลอกข้อความจากตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นย่อหน้า

'y$' คัดลอกข้อความจากตำแหน่งเคอร์เซอร์ไปถึงจุดสิ้นสุดบรรทัด

'y0' คัดลอกข้อความจากตำแหน่งเคอร์เซอร์ไปถึงจุดเริ่มต้นบรรทัด

'Y' เหมือนการใช้คำสั่ง 'y$'

3.4.10.2 put นำข้อความที่เก็บในบัฟเฟอร์ทั่วไปด้วยคำสั่ง yank มาแทรกในตำแหน่งที่ต้องการโดยใช้เคอร์เซอร์เป็นตัวบอกตำแหน่ง ดังนั้นในการคัดลอกข้อความในแต่ละครั้งจะต้องย้ายเคอร์เซอร์ไปยังข้อความที่ต้องการ แล้วใช้คำสั่งสำหรับคัดลอกข้อความเก็บในบัฟเฟอร์ทั่วไป แล้วย้ายเคอร์เซอร์ไปยังตำแหน่งที่ต้องการแทรก และเรียกใช้คำสั่ง put

ข้อความในบัฟเฟอร์ทั่วไปสามารถนำมาใช้ได้หลายครั้งเท่าที่ต้องการ หรือจนกว่าจะเรียกใช้คำสั่งคัดลอกครั้งใหม่ กรณีนี้จะลบข้อความในบัฟเฟอร์ออก และแทนที่ด้วยข้อความที่คัดลอกใหม่

คำสั่ง put เรียกใช้ได้ 2 รูปแบบคือ

'p' แทรกข้อความที่คัดลอกหน้าตำแหน่งเคอร์เซอร์

'P' แทรกข้อความที่คัดลอกหลังตำแหน่งเคอร์เซอร์

3.4.11 คำสั่งสำหรับอ่านข้อความจากแฟ้มข้อมูลอื่น (Read Command) ในขณะที่ใช้ vi สามารถกำหนดให้อ่านข้อความจากแฟ้มข้อมูลอื่นมาแทรกในตำแหน่งที่ต้องการได้ โดยการเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการแล้วใช้คำสั่งอ่านจากแฟ้มข้อมูล คือ

':r ชื่อแฟ้มข้อมูล'

vi จะทำการตรวจสอบชื่อแฟ้มข้อมูลที่กำหนดในคำสั่งนี้ ถ้าสามารถอ่านได้ก็จะอ่านข้อความจากแฟ้มข้อมูลนั้น นำเข้ามาแทรกในตำแหน่งที่ต้องการ กรณีที่อ่านไม่ได้ อาจจะเนื่องมาจากไม่มีชื่อแฟ้มข้อมูล หรือมีแต่ไม่อนุญาตให้อ่าน จะแสดงข้อความให้ทราบที่บรรทัดสุดท้ายของหน้าต่าง

3.4.12 คำสั่งสำหรับเขียนข้อความเก็บในแฟ้มข้อมูล (Write Command) นำข้อความในบัฟเฟอร์เก็บในแฟ้มข้อมูลเป็นการถาวร สามารถกำหนดชื่อแฟ้มข้อมูลที่ต้องการเก็บได้ โดยไม่จำเป็นต้องเก็บในแฟ้มข้อมูลเดิมที่เรียกใช้เสมอไป รูปแบบของคำสั่งที่ใช้ คือ

':w' เก็บข้อความไว้ในแฟ้มข้อมูลที่เรียกใช้ด้วยคำสั่ง vi

':w ชื่อแฟ้มข้อมูล' เก็บข้อความไว้ในแฟ้มข้อมูลที่กำหนดใหม่ ในการเรียกใช้รูปแบบนี้ถ้าเป็นแฟ้มข้อมูลเก่า vi จะไม่ทำงานทันทีแต่จะแสดงข้อความว่าเป็นแฟ้มข้อมูลที่มีแล้วถ้าต้องการจะเขียนข้อความใหม่จะต้องเรียกใช้ด้วยคำสั่งเพิ่มเติม คือ

':w! ชื่อแฟ้มข้อมูล'

3.4.13 คำสั่งสำหรับออกจาก vi (Quit Command) เมื่อต้องการออกจาก vi ทำได้หลายวิธีด้วยคำสั่งต่อไปนี้

':q' ออกจาก vi หลังจากเก็บข้องความของบัฟเฟอร์ไว้ในแฟ้มข้อมูลแล้ว โดยมีข้อกำหนดทั่วไปว่า หากมีการเปลี่ยนแปลงข้อความในบัฟเฟอร์ จะทำการเก็บข้อความนั้นในชื่อแฟ้มข้อมูลเดิมเสมอ แต่ถ้าไม่กำหนดคำสั่งเก็บข้อความก่อนที่จะออกจาก vi จะมีข้อความเตือนว่ายังไม่ได้ทำการเก็บข้อความที่แก้ไข และจะยังคงอยู่ใน vi การออกจาก vi โดยไม่เก็บข้อความในแฟ้มข้อมูลจะต้องใช้ '!' กำกับ

คำสั่ง 'q' นี้ สามารถใช้ร่วมกับคำสั่ง 'w' ได้ดังนี้

':wq' หรือ ':wq ชื่อแฟ้มข้อมูล' จะทำงานตามคำสั่ง 'w' ก่อนแล้วออกจาก vi

'ZZ' ออกจาก vi โดยทำงาน 2 ลักษณะคือ ทำการตรวจสอบว่ามีการเปลี่ยนแปลงข้อความในบัฟเฟอร์หรือไม่ หากจะเก็บข้อความของบัฟเฟอร์ไว้ในแฟ้มข้อมูลที่เรียกใช้แล้วจึงออกจาก vi หากไม่มีการเปลี่ยนแปลงข้อความจะออกจาก vi ได้ทันที

3.4.14 <u>คำสั่งสำหรับแทรกหรือสร้างข้อความใหม่</u> เป็นการรับข้อความที่ต้องการบันทึกไว้ในบัฟเฟอร์ ในการใช้คำสั่งเหล่านี้ vi จะเปลี่ยนจากช่วงของการรับคำสั่ง เป็นช่วงของการแก้ไขข้อความ และรับตัวอักษรทุกตัวเป็นส่วนหนึ่งของข้อความที่เก็บในบัฟเฟอร์ จนกระทั่งกด ESC เพื่อออกจากช่วงของการแก้ไขข้อความ คำสั่งที่เกี่ยวข้องในการทำงานลักษณะนี้คือ

'i' แทรกข้อความที่ตำแหน่งหน้าเคอร์เซอร์

'I' แทรกข้อความที่ตำแหน่งแรกของบรรทัด

'a' แทรกข้อความที่ตำแหน่งหลังเคอร์เซอร์

'A' แทรกข้อความที่ตำแหน่งสุดท้ายของบรรทัด

'o' แทรกบรรทัดว่างหนึ่งบรรทัดหลังบรรทัดขณะนั้น

'O' แทรกบรรทัดว่างหนึ่งบรรทัดก่อนบรรทัดขณะนั้น

ที่กล่าวมาข้างต้นเป็นเพียงโครงสร้างคำสั่งและการทำงานของ vi โดยย่อ มีคำสั่งต่างๆ เพื่อใช้ในการทำงานในลักษณะอื่น ซึ่งให้ความสะดวกแก่ผู้ใช้ในการใช้ vi เป็นโปรแกรมบรรณาธิกร รายละเอียดของคำสั่งเพิ่มเติม สามารถศึกษาได้จากหนังสือเกี่ยวกับยูนิกซ์ทั่วไป หรือจากคู่มือการใช้ vi ของยูนิกซ์แต่ละระบบได้

3.5 <u>ลักษณะเด่นอื่นๆ ของ vi</u>

3.5.1 <u>การกลับเข้าสู่ shell ชั่วคราว</u> (Escaping to a Shell) ในขณะที่กำลังใช้ vi และต้องการใช้คำสั่งของ shell ทำได้โดยไม่ต้องออกจาก vi และเมื่อเรียกคำสั่งของ shell มาใช้แล้วกลับไปทำงานใน vi ต่อไปได้ การเรียกใช้คำสั่งของ shell ทำโดยกำหนดคำสั่งให้ shell สร้างโปรเซสขึ้นมาใหม่ แล้วเรียกใช้คำสั่งของ shell จนกระทั่งต้องการกลับเข้าสู่ vi อีกครั้ง

3.5.2 กำหนดให้ vi ทำงานจากหลายแฟ้มข้อมูล (Multiple Filenames on Command Line) vi มีคุณสมบัติเด่นที่ต่างจากโปรแกรมบรรณาธิการทั่วไปที่ใช้ในระบบคอมพิวเตอร์เครื่องใหญ่ทั่วไปคือสามารถกำหนดให้ทำการแก้ไขเปลี่ยนแปลงข้อความจากหลายแฟ้มข้อมูลโดยการเรียกใช้คำสั่ง vi เพียงครั้งเดียวจาก shell โดย vi จะอ่านข้อความจากแฟ้มข้อมูลแรกที่กำหนดเข้าสู่บัฟเฟอร์ และทำงานด้วยข้อความนั้น จนกระทั่งกำหนดให้ทำงานในแฟ้มข้อมูลถัดไป ซึ่งจะทำตามลำดับแฟ้มข้อมูลที่กำหนดในคำสั่งของ shell แต่สามารถกำหนดให้ย้อนกลับไปทำงานในแฟ้มข้อมูลแรกได้

การทำงานในลักษณะนี้ทำให้สะดวกในการแก้ไขข้อความจากหลายแฟ้มข้อมูลในคราวเดียวกัน และนอกจากนี้ ยังสามารถกำหนดชื่อแฟ้มข้อมูลตามวิธีการของยูนิกซ์ได้ด้วย เช่น

vi *.c

กำหนดให้ vi ทำงานด้วยข้อความจากทุกแฟ้มข้อมูลที่มีชื่อตามด้วย '.c'

3.5.3 ใช้กับเทอร์มินอลหลายประเภท vi เป็นโปรแกรมบรรณาธิการที่ใช้กับเทอร์มินอลได้หลายชนิดที่สามารถติดต่อกับระบบปฏิบัติการยูนิกซ์ เนื่องจากเมื่อเริ่มทำงาน จะอ่านคุณสมบัติของเทอร์มินอลจากแฟ้มข้อมูลชื่อ /etc/termcap ซึ่งเป็นแฟ้มข้อมูลที่ใช้เก็บคุณสมบัติของเทอร์มินอลที่ใช้กับระบบปฏิบัติการยูนิกซ์ จากนั้นจึงใช้คุณสมบัติที่อ่านได้สำหรับทำงานในส่วนที่เกี่ยวข้องกับการแสดงผลทางจอภาพของ vi ทำให้ vi สามารถใช้ได้กับเทอร์มินอลหลายชนิด โดยการกำหนดชื่อของเทอร์มินอลที่ต้องการใช้ก่อนจะเรียกใช้ vi เสมอ

3.5.4 การเรียกคืนแฟ้มข้อมูลที่หาย (Recorvering Lost File) ในขณะที่ใช้ vi ถ้าระบบเกิดหยุดกระทันหัน โดยที่ไม่ได้เก็บข้อความของบัฟเฟอร์ไว้ในแฟ้มข้อมูล ทำให้ข้อความส่วนที่แก้ไขจากเดิมไม่ได้เก็บด้วย ถ้าใช้โปรแกรมบรรณาธิการบางชนิดจะต้องทำการแก้ไขข้อความเหล่านั้นใหม่ แต่ vi จะเก็บข้อความนี้ไว้เพื่อเรียกคืนได้ โดยมีการแจ้งให้ผู้ใช้ทราบ และทำการเรียกข้อความกลับคืน ทำให้ไม่ต้องแก้ไขใหม่ทั้งหมด

3.5.5 การคืนสภาพ (Undoing) จากที่กล่าวแล้วว่า vi ใช้บัฟเฟอร์สำหรับทำงานต่างๆ นอกจากจะมีประโยชน์ในการทำซ้ำดังได้กล่าวมาแล้ว ยังช่วยให้ผู้ใช้สามารถเรียกคืนสภาพข้อมูลที่พึ่งทำการเปลี่ยนแปลงถ้ามีความจำเป็น โดยใช้คำสั่ง undo (u)

3.5.6 การกำหนดข้อความมาโคร (Macro) ในกรณีที่มีชุดคำสั่งที่ใช้เป็นประจำใน vi สามารถกำหนดในรูปของมาโครได้ เพื่อความสะดวกในการเรียกใช้ และลดจำนวนตัวอักษรที่ต้องพิมพ์ในแต่ละครั้งที่เรียกใช้ชุดคำสั่งเหล่านั้น การกำหนดคำสั่งมาโครนี้ทำได้ 2 รูปแบบคือ

- กำหนดไว้ในบัฟเฟอร์ และเรียกใช้ด้วยการนำหน้าชื่อบัฟเฟอร์ด้วย '@'
- กำหนดโดยการใช้คำสั่ง map ดังนี้

:map lhs rhs

หลังจากที่กำหนดแล้ว เมื่อเรียกใช้ lhs ทุกครั้งจะแทนที่ด้วยค่าของ rhs ให้ ซึ่ง lhs ควรเป็นอักษรที่ต้องการกดปุ่มแป้นพิมพ์เพียงครั้งเดียว ในที่นี้อาจเป็นได้ทั้งปุ่มตัวอักษรปกติ และปุ่มฟังก์ชัน และมีสูงสุดไม่เกิน 10 ตัวอักษร ค่า rhs เป็นข้อความที่ต้องการกำหนดเป็นมาโคร ประกอบด้วยตัวอักษรไม่เกิน 100^{3}

---

3. Joy W., **An Introduction to Display Editing with Vi,** Ultrix-32 Supplementary Documents Volume I General Users, 1984, pp. 3-68