การใช้โปรแกรม vi เบื้องต้น

ความเป็นมาของ vi

โปรแกรมที่ทำหน้าที่ในการพิมพ์โปรแกรมต้นฉบับหรือเอกสารอื่นๆ ที่ประกอบขึ้นด้วยตัวอักษรเรียกว่า Editor โปรแกรม Editor ที่มีขีดความสามารถสูงมากโปรแกรมหนึ่งในระบบปฏิบัติการ UNIX คือ โปรแกรม vi ซึ่งเป็นโปรแกรมที่ได้รับการพัฒนาขึ้นที่มหาวิทยาลัยคาลิฟอร์เนียที่ Berkley โดยพัฒนาเพื่อให้เป็นโปรแกรมอรรถประโยชน์โปรแกรมหนึ่งในระบบปฏิบัติการ Berkley UNIX (BSD) ในปัจจุบัน vi เป็น editor ที่มีอยู่ในระบบ UNIX ทุกระบบ โปรแกรม vi พัฒนาขึ้นจากโปรแกรม ed ซึ่งเป็นโปรแกรมที่ทำงานได้ครั้งละบรรทัด ทำให้ยากต่อการแก้ไขโปรแกรมเนื่องจากผู้แก้ไขเห็นโปรแกรมครั้งละเพียงบรรทัดเดียว ด้วยข้อจำกัดเช่นนี้จึงทำให้มีการพัฒนาโปรแกรม ex ซึ่งสามารถมองเห็นข้อความได้มากขึ้นแต่ยังคงมีข้อจำกัดในการแก้ไขอยู่มาก โปรแกรม vi หรือ visual editor เป็นโปรแกรมที่สามารถแก้ไขข้อจำกัดของ ex ให้หมดไป โดยสามารถแสดงเอกสารได้เต็มหน้าจอและสามารถแก้ไขข้อความที่ส่วนใดของหน้าจอได้โดยสะดวก โดยการควบคุม cursor

เมื่อผู้ใช้ทำการ login เข้าสู่ระบบ UNIX ได้แล้ว ผู้ใช้สามารถเรียกใช้โปรแกรม vi ได้ โดยป้อนคำสั่ง

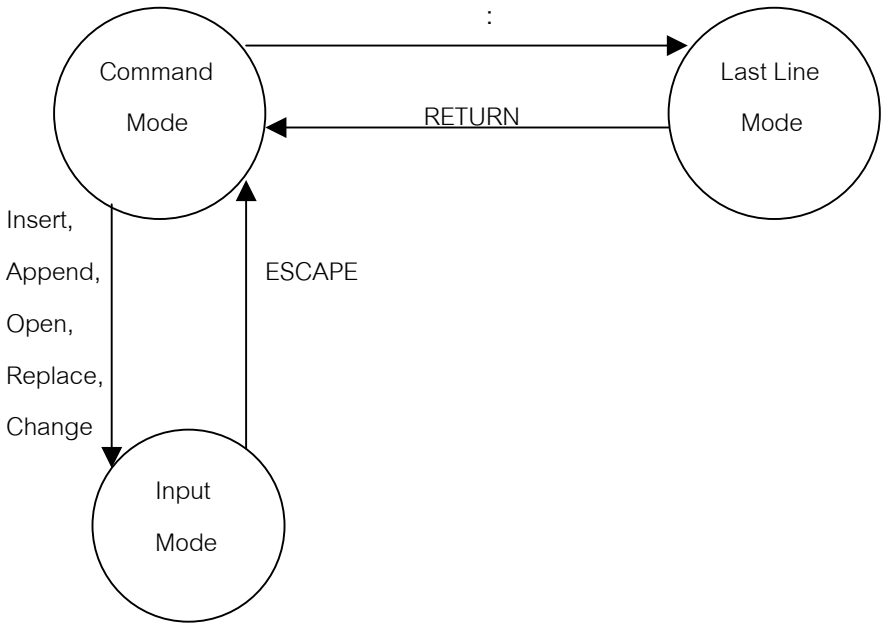
vi <ชื่อแฟ้ม>

สภาวะการทำงานของ vi

เมื่อ vi เริ่มทำงาน จะอยู่ในสภาวะ **command mode** หมายความว่า vi รอรับคำสั่งจากผู้ใช้ หากเป็นคำสั่งที่ใช้ในการพิมพ์หรือแก้ไขเอกสาร เช่น การแทรก, การลบ, การเลื่อน cursor คำสั่งที่ใช้จะเป็นตัวอักษร หรือกลุ่มของตัวอักษร เช่น คำสั่ง x ใช้ในการลบตัวอักษร ณ ตำแหน่ง cursor เมื่อมีการป้อนคำสั่งดังกล่าวเข้าไป จะทำให้ vi เปลี่ยนสภาวะจาก command mode เป็น **input mode** ซึ่งในสภาวะนี้แป้นตัวอักขระทุกตัวจะใช้ในการพิมพ์ข้อมูลที่ต้องการ การออกจาก input mode ไปสู่ command mode ทำได้โดยการกดแป้น Esc

ในขณะที่ vi อยู่ในภาวะ command mode หากผู้ใช้ป้อนคำสั่งที่ขึ้นต้นด้วยเครื่องหมาย : โปรแกรม vi จะเข้าสู่ภาวะที่เรียกว่า **last line mode** โดยแสดงเครื่องหมาย : ที่บรรทัดสุดท้ายของหน้าจอ เพื่อรอรับคำสั่งที่จะป้อนเข้าไป คำสั่งในชุดนี้เป็นกลุ่มของข้อความที่มีความสั้นยาวไม่เท่ากัน ดังนั้นเมื่อผู้ใช้ป้อนคำสั่งแล้ว จึงต้องกดแป้น Enter เพื่อให้ vi ทำงานตามคำสั่งนั้น เมื่อทำงานเสร็จแล้ว vi จะกลับเข้าสู่ภาวะ command mode โดยอัตโนมัติ คำสั่งที่ขึ้นต้นด้วยเครื่องหมาย : นี้เป็นคำสั่งที่ใช้ในการจัดการกับข้อมูลระดับแฟ้ม เช่นบันทึกข้อมูลที่พิมพ์เรียบร้อยแล้วลงในจานแม่เหล็ก, การปรับแต่งสิ่งแวดล้อมในการทำงานของ vi เพื่อให้เป็นไปตามความประสงค์ของผู้ใช้, การเปลี่ยนแทนข้อความ เป็นต้น

ความสัมพันธ์ระหว่างสภาวะการทำงานทั้งสามแบบของ vi สามารถแสดงได้ด้วย state transition diagram ดังนี้



ข้อสังเกต คำสั่งของ vi ที่เป็นอักษรตัวใหญ่และอักษรตัวเล็กมีลักษณะการทำงานต่างกัน โดยทั่วไปคำสั่งที่ใช้ตัวอักษรเดียวกันจะมีลักษณะการทำงานที่คล้ายกัน โดยคำสั่งที่เป็นอักษรตัวใหญ่จะมีขอบเขตที่กว้างกว่า เช่น คำสั่งแทรก ถ้าเป็นคำสั่ง **i** เป็นการแทรก ณ ตำแหน่ง cursor หากเป็นคำสั่ง **I** เป็นการแทรกที่ต้นบรรทัดไม่ว่าขณะที่ป้อนคำสั่ง cursor จะอยู่ ณ ตำแหน่งใดในบรรทัด

The work buffer

เมื่อเริ่มต้นทำงาน vi อ่านข้อมูลจากแฟ้มข้อมูลที่กำหนด เข้ามายังหน่วยความจำส่วนหนึ่งเรียกว่า work buffer การแก้ไข เพิ่มเติมข้อมูลใดๆที่เกิดขึ้นกับแฟ้มนั้น การเปลี่ยนแปลงดังกล่าวจะเกิดขึ้นใน work buffer โดยไม่มีผลใดๆต่อข้อมูลที่จัดเก็บในจานแม่เหล็ก การเปลี่ยนแปลงข้อมูลในแฟ้มที่จัดเก็บในจานแม่เหล็กจะเกิดขึ้นเมื่อผู้ใช้สั่งบันทึกข้อมูลใน work buffer ลงสู่จานแม่เหล็ก ซึ่งเป็นกระบวนการที่ทำการคัดลอกข้อมูลที่อยู่ใน work buffer ลงสู่จานแม่เหล็กโดยใช้ชื่อแฟ้มเดิม ทำให้ข้อมูลที่เคยจัดเก็บอยู่แต่เดิมในแฟ้มถูกเขียนทับโดยข้อมูลจาก work buffer

การอ่านข้อมูลจากแฟ้มข้อมูลเข้ามาจัดเก็บใน work buffer มีทั้งข้อดีและข้อเสีย ถ้าผู้ใช้เลิกการทำงานโดยไม่ได้สั่งให้ vi เขียนข้อมูลใน work buffer ลงในจานแม่เหล็ก ข้อมูลที่แก้ไขเปลี่ยนแปลงจะสูญหายไป ซึ่งหากข้อมูลที่เปลี่ยนแปลงแก้ไขนั้นเป็นข้อมูลที่ต้องการให้เกิดการเปลี่ยนแปลงจริง ผู้ใช้จะเสียเวลาในการแก้ไขเปลี่ยนแปลงข้อมูลนั้นใหม่ อย่างไรก็ดีหากผู้ใช้ทำการแก้ไขข้อมูลและไม่พอใจผลการแก้ไขนั้น ผู้ใช้สามารถยกเลิกการแก้ไขได้จากการเลิกทำงานโดยไม่เขียนข้อมูลที่อยู่ใน work buffer ลงในจานแม่เหล็ก ข้อมูลในแฟ้มที่จัดเก็บอยู่ในจานแม่เหล็กจะไม่เกิดการเปลี่ยนแปลงใดๆ

หากผู้ใช้ต้องการใช้ vi ในการอ่านข้อมูลในแฟ้มโดยไม่ต้องการแก้ไข หรือหากมีการแก้ไขจะไม่สามารถบันทึกลงในจานแม่เหล็กได้ สามารถทำได้โดยใช้โปรแกรม **view** ซึ่งมีการทำงานเหมือน vi ทุกประการ ยกเว้นแต่ไม่สามารถบันทึกข้อมูลใน work buffer ลงสู่จานแม่เหล็กได้

การบันทึกข้อมูลลงในแฟ้มและการเลิกการทำงาน

เมื่อผู้ใช้ทำการแก้ไขข้อมูลเรียบร้อยแล้ว ผู้ใช้สามารถเลิกการทำงานได้สองแบบคือ เลิกการทำงานและบันทึกข้อมูลใน work buffer ลงสู่จานแม่เหล็ก โดยใช้คำสั่ง ZZ เมื่ออยู่ใน command mode หรือเลิกการทำงานโดยไม่บันทึกข้อมูลลงสู่จานแม่เหล็กโดยใช้คำสั่ง :q! สำหรับคำสั่งนี้เมื่อป้อนคำสั่งจบแล้วต้องกดแป้น Enter เนื่องจากเป็นคำสั่งในแบบ last line mode การใช้คำสั่ง :q! ควรใช้ด้วยความระมัดระวัง เนื่องจาก vi จะจบการทำงานทันทีโดยไม่มีการให้ผู้ใช้ยืนยันเช่นในบางระบบ

ในบางครั้งผู้ใช้อาจเรียก vi ขึ้นมาทำงาน โดยลืมระบุชื่อแฟ้ม เมื่อทำการแก้ไขเพิ่มเติมข้อมูลเรียบร้อยแล้ว เมื่อใช้คำสั่ง ZZ เพื่อบันทึกข้อมูลและเลิกทำงาน vi จะตอบสนองด้วยข้อความว่า No file name เนื่องจากไม่มีชื่อแฟ้มที่จะใช้บันทึก ในกรณีเช่นนี้ผู้ใช้ต้องใช้คำสั่ง :w **<ชื่อแฟ้ม>** เพื่อบันทึกข้อมูลลงในแฟ้มที่กำหนดโดย <ชื่อแฟ้ม> แล้วจึงเลิกการทำงานด้วยคำสั่ง :q

การกู้แฟ้มที่กำลังแก้ไขในขณะที่ระบบปฏิบัติการล่ม

หากระบบปฏิบัติการล่ม (crash) หรือการสื่อสารข้อมูลขาดหายไปในระหว่างที่ผู้ใช้กำลังใช้ vi แก้ไขข้อมูล ข้อมูลส่วนที่ดำเนินการไปแล้วระบบจะยังคงเก็บรักษาไว้ให้ เมื่อระบบปฏิบัติการหรือสายสื่อสารเป็นปกติแล้ว ผู้ใช้สามารถกู้แฟ้มข้อมูลที่เคยทำงานอยู่ขณะเกิดปัญหาโดยใช้คำสั่ง

$ vi –r filename

หากระบบปฏิบัติการสามารถเก็บแฟ้มในขณะ crash ไว้ได้ vi จะสามารถนำแฟ้มข้อมูลนั้นขึ้นมาให้ผู้ใช้แก้ไขต่อไปได้

การแสดงผลหน้าจอ

โปรแกรม vi ใช้บรรทัดแสดงสถานะซึ่งเป็นบรรทัดท้ายสุดของหน้าจอแสดงข่าวสารที่ vi ต้องการติดต่อกับผู้ใช้ ข่าวสารดังกล่าวนี้ประกอบด้วย ข่าวสารแสดงความผิดพลาด, ข่าวสารแสดงการดำเนินการ เช่นจำนวนบรรทัดที่ลบ, จำนวนบรรทัดที่ได้เพิ่มเติมมาจากการอ่านแฟ้มอื่นเข้ามาในข้อมูลที่กำลังแก้ไขอยู่ในปัจจุบัน, สถานะของข้อมูลที่กำลังแก้ไขอยู่ในปัจจุบัน และแสดงคำสั่งที่ผู้ใช้ป้อนเข้าไปในขณะอยู่ในสภาวะ last line mode

การปรับหน้าจอเมื่อมีข่าวสารอื่นเข้ามาแทรกในข้อความที่กำลังแก้ไข

ในระบบ UNIX ซึ่งเป็นระบบปฏิบัติการที่สนับสนุนการสื่อสารข้อมูลและมีโปรแกรมอรรถประโยชน์หลายชนิดที่ช่วยให้ผู้ใช้ติดต่อสื่อสารกันได้ เช่น talk และ write เป็นต้น ดังนั้นในขณะที่ผู้ใช้กำลังใช้ vi ในการแก้ไขข้อมูลอยู่ จึงมีความเป็นไปได้ที่ผู้ใช้อื่นจะติดต่อเข้ามา ข่าวสารที่เกิดขึ้นจากการติดต่อจะมีการแสดงผลปะปนกับการแสดงผลข้อมูลที่กำลังแก้ไข อย่างไรก็ดีลักษณะเช่นนี้เป็นเฉพาะการแสดงผลหน้าจอเท่านั้น ไม่เกี่ยวข้องกับ working buffer แต่อย่างใด ผู้ใช้สามารถสั่งให้ vi แสดงผลข้อมูลที่กำลังแก้ไขอยู่ใน work buffer ใหม่ได้ โดยการกดแป้น Esc เพื่อเข้าสู่l4k;t command mode แล้วใช้คำสั่ง Ctrl-L เพื่อแสดงผลใหม่ คำสั่งนี้ในบางระบบใช้คำสั่ง Ctrl-R

เครื่องหมายแสดงจุดสิ้นสุดของข้อมูล (~)

ในการใช้งาน vi หากหน้าจอที่กำลังแก้ไขมีจุดสิ้นสุดของข้อมูลอยู่ด้วย vi ใช้สัญลักษณ์ Tilde (~) แทนจุดสิ้นสุดของข้อมูล โดยแสดงสัญลักษณ์นี้ทางด้านซ้ายมือของบรรทัดทุกบรรทัดที่เลยจุดสิ้นสุดของข้อมูลบนจอภาพ ซึ่งจะเห็นได้ชัดเมื่อเริ่มใช้ vi ในการสร้างแฟ้มข้อมูลใหม่ โดยทุกบรรทัดยกเว้นบรรทัดแรกจะมีสัญลักษณ์ Tilde ปรากฏอยู่แสดงว่ายังไม่ข้อมูลใน working buffer เลย

การเลื่อน cursor ขณะอยู่ในสภาวะ Command Mode

ในขณะที่ vi อยู่ใน command mode ผู้ใช้สามารถเลื่อน cursor ไปยังตำแหน่งของอักขระใดๆบนหน้าจอได้ และสามารถเลื่อนหน้าจอเพื่อแสดงข้อมูลที่ส่วนใดส่วนหนึ่งของ working buffer ได้โดยใช้คำสั่งเลื่อน cursor

การเลื่อน cursor ครั้งละตัวอักษร

เลื่อน cursor ไปทางขวา 1 ตัวอักษร	l หรือแป้น ลูกศรขวา
เลื่อน cursor ไปทางซ้าย 1 ตัวอักษร	h หรือแป้น ลูกศรซ้าย

การเลื่อน cursor ทั้งสองกรณีสามารถใช้ตัวเลขแสดงจำนวนตัวอักษรร่วมด้วยได้ เช่น 7h หมายถึง เลื่อน cursor ไปทางซ้าย 7 ตัวอักษร

การเลื่อน cursor ครั้งละ 1 คำ

การเลื่อน cursor ไปยังจุดเริ่มต้นของคำต่อไปโดยใช้เครื่องหมายวรรคตอนอื่นที่ไม่ใช่เว้นวรรค เช่น !) เป็นเครื่องหมายแยกคำ

เลื่อน cursor ไปทางขวา 1 คำ	w
เลื่อน cursor ไปทางซ้าย 1 คำ	b

การเลื่อน cursor ไปยังจุดเริ่มต้นของคำต่อไปโดยใช้เครื่องหมายวรรค เป็นเครื่องหมายแยกคำ

เลื่อน cursor ไปทางขวา 1 คำ	W
เลื่อน cursor ไปทางซ้าย 1 คำ	B

คำสั่ง w, W, b และ B สามารถใช้ร่วมกับตัวเลขแสดงจำนวนคำได้ เช่น 5w เลื่อนcursor ไปทางขวา 5 คำ

การเลื่อน cursor ครั้งละบรรทัด

เลื่อน cursor ขึ้นบน 1 บรรทัด	k หรือแป้นลูกศรขึ้น
เลื่อน cursor ลงล่าง 1 บรรทัด	j หรือแป้นลูกศรลง

การเลื่อน cursor ภายในบรรทัด

เลื่อน cursor ไปต้นบรรทัด	0 (ศูนย์)
เลื่อน cursor ไปท้ายบรรทัด	$

การเลื่อน cursor ครั้งละ 1 ประโยค

เลื่อน cursor ไปทางขวา 1 ประโยค	)
เลื่อน cursor ไปทางซ้าย 1 ประโยค	(

การเลื่อน cursor ครั้งละ 1 ย่อหน้า

เลื่อน cursor ไปย่อหน้าถัดไป	}
เลื่อน cursor ไปย่อหน้าก่อน	{

การเลื่อน cursor ในหน้าจอ

เลื่อน cursor ไปยังมุมบนซ้ายของจอ	H
เลื่อน cursor ไปยังบรรทัดที่อยู่กึ่งกลางจอ	M
เลื่อน cursor ไปยังบรรทัดล่างสุดของจอ	L

การเลื่อนหน้าจอ

เลื่อนหน้าจอขึ้นครึ่งหน้าจอ	Ctrl-U	(up)
เลื่อนหน้าจอลงครึ่งหน้าจอ	Ctrl-D	(down)
เลื่อนหน้าจอขึ้น 1 หน้าจอ	Ctrl-F	(forward)
เลื่อนหน้าจอลง 1 หน้าจอ	Ctrl-B	(backward)

การเลื่อน cursor ไปยังบรรทัดที่ต้องการ

การเลื่อน cursor ไปยังบรรทัดที่ต้องการทำได้โดยการพิมพ์หมายเลขบรรทัดตามด้วยคำสั่ง G เช่น 25G เป็นการเลื่อน cursor ไปยังบรรทัดที่ 25

หากผู้ใช้ป้อนคำสั่ง G โดยไม่มีหมายเลขบรรทัดประกอบ vi จะเลื่อน cursor ไปยังบรรทัดสุดท้ายใน work buffer

การทำงานในสภาวะ Input Mode

จากที่ได้กล่าวมาแล้วว่า vi มีการทำงานเป็น 3 สภาวะคือ command mode, input mode และ last line mode คำสั่งที่กล่าวมาแล้วเป็นคำสั่งสำหรับเลื่อน cursor ไปยังจุดที่ต้องการทำการแก้ไข ส่วนการแก้ไขเพิ่มเติมจะเกิดขึ้นเมื่อผู้ใช้ป้อนคำสั่งสำหรับการแก้ไขเพิ่มเติม คำสั่งเหล่านี้จะได้กล่าวถึงเป็นลำดับไป

คำสั่งแทรก (Insert)

การแทรกที่หน้าตำแหน่ง cursor	i
การแทรกที่ต้นบรรทัดไม่ว่า cursor จะอยู่ที่ตำแหน่งใดก็ตาม	I

เมื่อใช้คำสั่ง i หรือ I แล้ว ผู้ใช้สามารถป้อนข้อความที่ต้องการแทรกได้ทันที เมื่อเสร็จสิ้นการพิมพ์ข้อความที่ต้องการแทรกแล้ว ให้กดแป้น Esc เพื่อกลับสู่สภาวะ command mode

คำสั่งพิมพ์ข้อความต่อ (Append)

คำสั่งพิมพ์ข้อความต่อมีการทำงานคล้ายคำสั่งแทรก คือ คำสั่ง **a** เป็นการแทรกข้อความที่หลังตำแหน่ง cursor ส่วนคำสั่ง **A** เป็นการแทรกข้อความที่ท้ายบรรทัด ไม่ว่าในขณะนั้น cursor จะอยู่ ณ ตำแหน่งใดก็ตาม

คำสั่งแทรกบรรทัดใหม่ (Open)

โดยปกติเมื่ออยู่ใน input mode เมื่อผู้ใช้พิมพ์ข้อความจนกระทั่งเต็มบรรทัดแล้ว หากผู้ใช้ต้องการขึ้นบรรทัดใหม่สามารถทำได้โดยการกดแป้น Enter อย่างไรก็ตามหากมีการพิมพ์ข้อความไว้แล้ว และต้องการแทรกบรรทัดใหม่ลงระหว่างข้อความเดิมทำได้โดยเลือกใช้คำสั่งดังนี้

แทรกบรรทัดใหม่ใต้บรรทัด cursor **o**

แทรกบรรทัดใหม่เหนือบรรทัด cursor **O**

การใช้คำสั่ง o และ O ในสภาวะ command mode จะทำให้ vi แทรกบรรทัดว่างให้และเข้าสู่ input mode ผู้ใช้สามารถพิมพ์ข้อความที่ต้องการแทรกได้ทันที

คำสั่งพิมพ์ข้อความทับหรือเปลี่ยนแทนข้อความ (Replace)

เมื่อผู้ใช้ต้องการพิมพ์ข้อความใหม่ทับข้อความเดิมที่มีอยู่ การพิมพ์ทับหรือเปลี่ยนตัวอักษรเพียงตัวเดียวใช้คำสั่ง **r** เมื่อพิมพ์ตัวอักษรที่ต้องการเปลี่ยนแล้ว vi จะกลับเข้าสู่ command mode โดยอัตโนมัติ หากต้องการเปลี่ยนแทนตัวอักษรหลายตัว ใช้คำสั่ง **R** เมื่อเปลี่ยนแทนเรียบร้อยแล้ว ผู้ใช้ต้องกดแป้น Esc เพื่อกลับเข้าสู่สภาวะ command mode การใช้คำสั่ง R มีข้อควรสังเกตคือหากข้อความเดิมยาวกว่าข้อความที่เปลี่ยนแทน ข้อความใหม่ส่วนที่เกินจะพิมพ์ทับข้อความเดิม หากข้อความใหม่สั้นกว่าข้อความเดิม ผู้ใช้ต้องลบข้อความเดิมที่เหลือด้วยคำสั่ง x

การยกเลิกคำสั่ง (Undo)

คำสั่ง **u** ใช้ในการยกเลิกการทำงานที่เคยทำไปเป็นครั้งสุดท้ายก่อนหน้าที่จะใช้คำสั่งนี้ เช่นผู้ใช้เพิ่งใช้คำสั่งลบบรรทัดไป และพบว่าลบผิดบรรทัด สามารถยกเลิกผลของคำสั่งลบบรรทัดได้ โดยใช้คำสั่ง u ซึ่ง vi จะยกเลิกคำสั่งนั้นและนำบรรทัดที่ลบไปกลับคืนมา คำสั่ง u สามารถใช้ยกเลิกคำสั่งสุดท้ายที่เพิ่งใช้งานไปได้เพียงคำสั่งเดียว ส่วนคำสั่ง **U** ใช้ในการเรียกบรรทัดเดิมที่ได้แก้ไขครั้งสุดท้ายกลับคืนมาไม่ว่าผู้ใช้จะทำการเปลี่ยนแปลงแก้ไขข้อมูลในบรรทัดนั้นไปอย่างไร

การลบตัวอักษร

การลบตัวอักษร ณ ตำแหน่ง cursor ใช้คำสั่ง **x** คำสั่งนี้สามารถใช้ร่วมกับตัวเลขแสดงจำนวนตัวอักษรได้ เช่น 7x ใช้ลบตัวอักษร 7 ตัวเริ่มต้นที่ตัวอักษร ณ ตำแหน่ง cursor

การลบบรรทัด

การลบบรรทัดทั้งบรรทัด ในบรรทัดที่มี cursor อยู่ใช้คำสั่ง **dd** คำสั่งนี้สามารถใช้ร่วมกับตัวเลขแสดงจำนวนบรรทัดที่ต้องการได้ เช่น 5dd ใช้ลบบรรทัด 5 บรรทัดเริ่มต้น ณ บรรทัดที่มี cursor อยู่

การลบข้อความ (Delete)

การลบข้อความใช้คำสั่ง **d** คำสั่งนี้เป็นคำสั่งที่ต้องมีการกำหนดขอบเขตที่ต้องการลบด้วย โดยใช้คำสั่งเลื่อน cursor เป็นตัวกำหนดขอบเขต เช่น

d0 ลบข้อความตั้งแต่ตำแน่ง cursor ไปถึงต้นบรรทัด

d$ ลบข้อความตั้งแต่ตำแหน่ง cursor ไปถึงท้ายบรรทัด

d} ลบข้อความตั้งแต่ตำแหน่ง cursor ไปถึงสิ้นย่อหน้า

dG ลบข้อความตั้งแต่ตำแหน่ง cursor ไปถึงท้ายไฟล์

ข้อสังเกต คำสั่งในหมวด d เป็นคำสั่งที่มีอันตรายมากหากใช้โดยไม่เข้าใจความหมายโดยละเอียด ขอให้ระมัดระวังให้มากเวลาใช้งาน

การเปลี่ยนข้อความเดิมด้วยข้อความใหม่ (Change)

คำสั่งเปลี่ยนเป็นคำสั่งที่ใช้สำหรับเปลี่ยนแทนข้อความที่มีอยู่แล้วด้วยข้อความใหม่ โดยข้อความใหม่อาจมีความยาวสั้นกว่า หรือยาวกว่าข้อความเดิมก็ได้ หากข้อความใหม่สั้นกว่าข้อความเดิม vi จะตัดข้อความเดิมส่วนที่เกินออกไป หากข้อความใหม่ยาวกว่าข้อความเดิม vi จะแทรกข้อความส่วนที่เกินให้ ลักษณะการทำงานเช่นนี้แตกต่างจากคำสั่ง replace ซึ่งเป็นคำสั่งที่ใช้เปลี่ยนแทนข้อความที่มีความยาวเท่ากัน การเปลี่ยนแทนข้อความด้วยคำสั่ง **c** สามารถกำหนดขอบเขตได้เช่นเดียวกับคำสั่ง delete เช่น

cw เปลี่ยนคำที่มีอยู่เดิมด้วยคำใหม่

c3w เปลี่ยนคำที่มีอยู่เดิม 3 คำ ด้วยข้อความใหม่

c$ เปลี่ยนแทนข้อความตั้งแต่ตำแหน่ง cursor ไปจนถึงท้ายบรรทัดด้วยข้อความใหม่

เมื่อใช้คำสั่ง c และพิมพ์ข้อความใหม่ที่ต้องการเปลี่ยนแทนเสร็จแล้ว ผู้ใช้ต้องกดแป้น Esc เพื่อแสดงว่าการเปลี่ยนแปลงเสร็จสิ้นแล้ว vi จะเข้าสู่สภาวะ command mode

การเชื่อมบรรทัด (Join) **และการแยกบรรทัด**

หากผู้ใช้ต้องการเชื่อมบรรทัด 2 บรรทัดเข้าด้วยกันเป็นบรรทัดเดียว สามารถทำได้โดยวาง cursor ที่บรรทัดบน และใช้คำสั่ง **J** vi จะทำการเชื่อมบรรทัดทั้งสองเข้าด้วยกันโดยแทรกเว้นวรรคระหว่างข้อความทั้งสองบรรทัดให้โดยอัติโนมัติ

หากต้องการแยกบรรทัดเดียวออกเป็น 2 บรรทัดทำได้โดยการเลื่อน cursor ไปยังจุดที่ต้องการแยก ใช้คำสั่ง **i** และกดแป้น Enter

การแสดงสถานะของ vi

ในขณะทำการแก้ไขข้อมูล ผู้ใช้อาจต้องการทราบหมายเลขของบรรทัดที่ cursor อยู่ จำนวนบรรทัดทั้งหมด และตำแหน่งของ cursor ในบรรทัดนั้น การเรียกดูสถานะดังกล่าวนี้ทำได้โดยใช้คำสั่ง Ctrl-G

การเรียกใช้คำสั่งเดิมซ้ำอีกครั้งหนึ่ง

ในการแก้ไขเพิ่มเติมข้อความ บางครั้งผู้ใช้อาจต้องการทำงานแบบเดิมซ้ำหลายครั้ง เช่นเมื่อผู้ใช้ลบบรรทัดด้วยคำสั่ง dd ไปบรรทัดหนึ่งแล้ว และต้องการลบบรรทัดอื่นด้วย ผู้ใช้สามารถเลื่อน cursor ไปยังบรรทัดที่ต้องการลบ และใช้คำสั่งจุด (.) เพื่อทำคำสั่งที่เคยใช้ไปแล้วเป็นครั้งสุดท้ายได้ หรือผู้ใช้ต้องการแทรกคำบางคำเพิ่มเติมเข้าไปที่จุดจุดหนึ่ง และต้องการแทรกคำเช่นเดียวกันที่จุดอื่น สามารถทำได้โดยการเลื่อน cursor ไปยังจุดที่ต้องการแทรก และใช้คำสั่งจุด (.) เพื่อทำงานเช่นเดียวกับที่เคยทำครั้งสุดท้ายได้ คำสั่งจุดเป็นคำสั่งที่สะดวกและมีประโยชน์ในการแก้ไขมากคำสั่งหนึ่ง

การเคลื่อนย้ายและคัดลอกข้อความ

ในขณะที่ vi ทำงานจะทำการจัดสรรหน่วยความจำส่วนหนึ่งไว้เรียกว่า general-purpose buffer เพื่อใช้เก็บข้อมูลในระหว่างการแก้ไขเพิ่มเติม โดยจะทำการจัดเก็บข้อมูลที่มีการลบด้วยคำสั่ง delete, การเปลี่ยนข้อความด้วยคำสั่ง change, และการคัดลอกข้อความด้วยคำสั่ง yank ที่เคยใช้งานครั้งสุดท้ายไว้ หน่วยความจำส่วนนี้มีประโยชน์มากในกรณีที่ผู้ใช้ต้องการเคลื่อนย้ายหรือคัดลอกข้อมูลที่มีอยู่แล้วจากบริเวณหนึ่งไปยังอีกบริเวณหนึ่งโดยใช้คำสั่งในกลุ่ม put, delete และ yank

คำสั่ง Delete

เมื่อผู้ใช้ใช้คำสั่งในกลุ่ม delete ตามที่ได้กล่าวมาแล้ว vi จะทำการเก็บข้อมูลที่ผู้ใช้ทำการลบไว้ใน general purpose buffer ซึ่งการกระทำดังกล่าวนี้เองช่วยให้ผู้ใช้สามารถใช้คำสั่ง undo เรียกข้อความที่ถูกลบแล้วกลับคืนมาได้ นอกจากนี้แล้วหากผู้ใช้ทำการลบข้อความส่วนหนึ่งออก ผู้ใช้สามารถนำข้อความนี้ซึ่งอยู่ใน general-purpose buffer ไปแทรกลงที่จุดอื่นใน work buffer ได้ จากความรู้ดังกล่าวนี้สามารถนำมาประยุกต์ใช้ในการเคลื่อนย้ายข้อความจากส่วนหนึ่งไปยังส่วนอื่นของเอกสารได้ โดยทำการลบเอกสารที่ต้องการเคลื่อนย้ายโดยใช้คำสั่งในกลุ่ม delete และทำการแทรกลง ณ จุดที่ต้องการได้โดยใช้คำสั่ง put

คำสั่ง Put

คำสั่ง put เป็นคำสั่งที่ใช้ในการคัดลอกข้อมูลที่มีอยู่ใน general-purpose buffer ไปแทรกลงในจุดที่กำหนดโดย cursor โดยคำสั่ง P จะทำการแทรกข้อความใน buffer ลงไปในตำแหน่งก่อนหน้าของ cursor ที่ปรากฏอยู่ในขณะนั้น และคำสั่ง p จะทำการให้เกิดการแทรกลงหลังตำแหน่ง cursor หากเป็นข้อความที่ลบด้วยคำสั่งลบบรรทัดหรือย่อหน้า คำสั่ง P จะทำการแทรกบรรทัดหรือย่อหน้านั้นลงเหนือบรรทัดตำแหน่ง cursor หากเป็นคำสั่ง p จะทำการแทรกลงใต้บรรทัดตำแหน่ง cursor

เนื่องจาก vi มี general-purpose buffer เพียงหน่วยเดียว ดังนั้นข้อความที่เก็บอยู่ใน general-purpose buffer จึงเกิดการเปลี่ยนแปลงไปทุกครั้งที่ผู้ใช้มีการใช้คำสั่งในกลุ่ม delete, change และ yank ดังนั้นจึงถือเป็นหลักในการปฏิบัติว่าเมื่อผู้ใช้ทำการลบข้อความที่ต้องเคลื่อนย้ายแล้ว สามารถใช้ได้แต่เฉพาะคำสั่งสำหรับเคลื่อนย้าย cursor เท่านั้น เมื่อเลื่อน cursor ถึงจุดที่ต้องการแล้วใช้คำสั่ง put เพื่อแทรกข้อความลง หากมีการใช้คำสั่งอื่นข้อความใน general-purpose buffer อาจเปลี่ยนแปลงไป ไม่อาจเรียกกลับคืนมาได้อีก

คำสั่ง Yank

คำสั่ง yank เป็นคำสั่งที่มีลักษณะการทำงานคล้ายคำสั่ง delete เพียงแต่คำสั่ง yank ทำการคัดลอกข้อความที่กำหนดลงใน general-purpose buffer โดยไม่ได้ลบข้อมูลนั้นออกเหมือนการใช้คำสั่ง delete คำสั่ง yank จึงสามารถประยุกต์ใช้ในการคัดลอกข้อความที่มีอยู่แล้ว ณ จุดหนึ่งในเอกสารไปแทรกลงยังจุดอื่นได้ หากต้องการคัดลอกข้อความบรรทัดเดียวไปยัง general-purpose buffer ใช้คำสั่ง **yy** หากข้อความมีหลายบรรทัด สามารถใช้ตัวเลขบอกจำนวนบรรทัดร่วมด้วยได้ เช่น 10yy เป็นการคัดลอกข้อความ 10 บรรทัดเริ่มตั้งแต่บรรทัด cursor ลงใน general-purpose buffer และสามารถแทรกลงที่จุดอื่นได้โดยใช้คำสั่ง put

หากข้อความที่ต้องการคัดลอกมีลักษณะเป็นคำ, ประโยค หรือย่อหน้า สามารถใช้คำสั่งเคลื่อนย้าย cursor ประกอบเป็นขอบเขตที่ต้องการเช่น y$ ใช้คัดลอกข้อความจากตำแหน่ง cursor ไปจนถึงท้ายบรรทัดลงสู่ general-purpose buffer

การเขียนและอ่านไฟล์

โดยปกติเมื่อผู้ใช้เรียกใช้โปรแกรม vi พร้อมกับระบุชื่อไฟล์ vi จะอ่านไฟล์ที่ระบุเข้ามาใน work buffer เพื่อให้ผู้ใช้แก้ไขเพิ่มเติม และเมื่อผู้ใช้แก้ไขข้อมูลเสร็จแล้วสามารถบันทึกข้อมูลใน work buffer กลับคืนลงสู่จานแม่เหล็กด้วยคำสั่ง ZZ สำหรับการเขียนและอ่านไฟล์ในหัวข้อนี้เป็นเรื่องของการอ่านข้อมูลที่มีอยู่ในอีกไฟล์หนึ่งเข้ามารวมกับข้อมูลเดิมที่มีอยู่ใน work buffer และการเขียนข้อมูลบางส่วนใน work buffer ลงสู่ไฟล์ใหม่ คำสั่งในส่วนนี้ประกอบด้วยคำสั่ง read และ write ซึ่งเป็นคำสั่งทำงานในสภาวะ last line mode

คำสั่ง Read

คำสั่ง read ใช้อ่านข้อมูลจากแฟ้มข้อมูลอื่นที่มีอยู่ในจานแม่เหล็กเข้าใน work buffer โดยอ่านข้อมูลจากไฟล์ที่ระบุเข้ามาแทรกข้อมูลเดิม ณ ตำแหน่ง cursor หากต้องการอ่านเข้ามาแทรกที่ตำแหน่งอื่นสามารถระบุได้ รูปแบบของคำสั่ง read เป็นดังนี้

: **[address] r [filename]**

เมื่อ address เป็นหมายเลขบรรทัดที่ต้องการจะอ่านข้อมูลเข้ามาแทรก และเนื่องจากเป็นคำสั่งในแบบ last line mode ผู้ใช้จึงต้องกดแป้น Enter เมื่อป้อนคำสั่งจบแล้ว หากผู้ใช้ไม่ระบุตำแหน่ง vi จะทำการอ่านแฟ้มข้อมูลที่ระบุเข้ามาแทรก ณ ตำแหน่ง cursor ในขณะนั้น

คำสั่ง Write

คำสั่ง write เป็นคำสั่งที่ใช้เขียนข้อมูลเป็นบางส่วน หรือข้อมูลทั้งหมดลงในแฟ้มข้อมูลใหม่ ดังนั้นก่อนจะใช้คำสั่ง write จึงต้องผู้ใช้จะต้องทราบว่าจะเขียนข้อมูลจากบรรทัดใดถึงบรรทัดใด และชื่อของแฟ้มข้อมูลที่ต้องการจัดเก็บข้อมูล เช่น หากต้องการบันทึกข้อมูลใน working buffer จากบรรทัดที่ 23 ถึง 85 ลงสู่แฟ้มชื่อ part.txt ใช้คำสั่งดังนี้

: **23,85 w part.txt**

ในกรณีที่แฟ้มข้อมูลชื่อ part.txt มีอยู่บนจานแม่เหล็กแล้ว จะไม่สามารถเขียนข้อมูลทับได้ หากต้องการให้ข้อมูลใหม่ทับข้อมูลเดิม ต้องใช้คำสั่งดังนี้

: **23,85 w! part.txt**

ในกรณีที่แฟ้มข้อมูลชื่อ part.txt มีอยู่บนจานแม่เหล็กแล้ว และต้องการนำข้อมูลใหม่เขียนต่อท้ายข้อมูลเดิมที่มีอยู่ในแฟ้ม ทำได้ดังนี้

: 23,85 w>> part.txt

ข้อควรระวัง คำสั่ง w เป็นคำสั่งที่หากใช้โดยไม่มีชื่อแฟ้มจะเขียนข้อมูลลงแฟ้มเดิมที่เปิดไว้ หากมีการระบุหมายเลขบรรทัดด้วย จะทำให้ข้อมูลในแฟ้มเดิมเปลี่ยนแปลงไป หากไม่แน่ใจสามารถขอดูชื่อแฟ้มที่กำลังแก้ไขได้โดยใช้คำสั่ง :f

การกำหนดสิ่งแวดล้อมในการทำงานของ vi

เนื่องจาก vi เป็นโปรแกรมพิมพ์เอกสารสารพัดประโยชน์ ดังนั้นเพื่อให้ vi สามารถตอบสนองความต้องการของผู้ใช้ได้อย่างเหมาะสมสำหรับงานแต่ละประเภท เช่น การพิมพ์บทความ, การพิมพ์โปรแกรมต้นฉบับ ผู้พัฒนา vi จึงออกแบบให้ vi สามารถปรับแต่งสิ่งแวดล้อมที่เหมาะสมสำหรับการทำงานในแต่ละประเภทได้ โดยกำหนด parameter ที่เหมาะสมผ่านคำสั่ง set ใน last line mode หากต้องการดูค่า parameter เหล่านี้สามารถดูได้โดยใช้คำสั่งใน last line mode คือคำสั่ง :set all

การใช้คำสั่ง set ในขณะที่ vi อยู่ในสภาวะ last line mode จะมีผลเฉพาะในการใช้งานในครั้งนั้น เมื่อเลิกการทำงานและเรียกใช้ vi ใหม่ต้องทำการปรับแต่ง parameter ใหม่ หากต้องการให้มีผลถาวร สามารถนำคำสั่งเหล่านี้มารวมกันไว้ในแฟ้ม .exrc ใน home directory เมื่อมีการเรียกใช้ vi ครั้งต่อไป vi จะอ่าน parameter จากแฟ้มมาทำการปรับแต่งสิ่งแวดล้อมโดยอัตโนมัติ

Parameters

Parameter ของ vi มีมากมาย และอาจมีความแตกต่างกันในระหว่าง vi รุ่นต่างๆขอให้ตรวจสอบกับโปรแกรมจริงที่ต้องใช้งาน อย่างไรก็ตาม parameter ที่เป็นพื้นฐานและควรทราบมีดังนี้

Line Numbers

ในการใช้งานโดยทั่วไป vi จะไม่แสดงหมายเลขบรรทัดของข้อความ หากผู้ใช้ต้องการให้ vi แสดงเลขบรรทัด เช่นในกรณีของการตรวจสอบโปรแกรม สามารถทำได้โดยใช้คำสั่ง :set number และสามารถยกเลิกได้โดยใช้คำสั่ง :set nonumber อย่างไรก็ดีไม่ว่าผู้ใช้จะกำหนดให้มีการแสดงหมายเลขบรรทัดหรือไม่ก็ตาม หมายเลขบรรทัดนี้ไม่ได้เป็นส่วนหนึ่งของเอกสารและไม่อาจบันทึกลงในแฟ้มได้

Line Wrap Margin

Line wrap margin เป็น parameter ที่ใช้ในการกำหนดระยะห่างจากขอบด้านขวามือของจอภาพที่ต้องการให้ vi ตัดข้อความส่วนที่เกินมาขึ้นบรรทัดใหม่โดยอัตโนมัติ เช่น หากผู้ใช้กำหนดค่า line wrap margin = 5 เมื่อผู้ใช้พิมพ์ข้อความในบรรทัดไปจนกระทั่งเหลือระยะทางอีก 5 ตัวอักษรจะถึงขอบขวา vi จะตัดคำขึ้นบรรทัดใหม่ให้โดยอัตโนมัติ การกำหนดค่า line wrap margin มีรูปแบบคำสั่งดังนี้

: set wrapmargin=*nn*

เมื่อ *nn* เป็นจำนวนตัวอักษรก่อนถึงขอบด้าน**ขวา**มือของจอภาพที่ต้องการให้ vi ตัดคำขึ้นบรรทัดใหม่ หากค่าของ *nn* เป็นศูนย์ ผู้ใช้ต้องกดแป้น Enter เพื่อขึ้นบรรทัดใหม่ด้วยตนเอง

Show mode

โดยปกติในระหว่างการทำงาน vi จะไม่แสดงสถานะให้ผู้ใช้ทราบว่าขณะนี้อยู่ในภาวะใด หากต้องการให้ vi แสดงสถานะให้ทราบเมื่ออยู่ในสภาวะ input mode ทำได้โดยการใช้คำสั่ง **:set showmode** หากต้องการยกเลิก ใช้คำสั่ง **:set noshowmode**

Invisible Characters

โดยปกติการแสดงผลของ vi จะแสดงผลเฉพาะตัวอักษร ส่วนรหัสควบคุมแสดงผลที่เกิดจากการควบคุมนั้น เช่น tab หรือ Ctrl-I เวลาแสดงผลจะแสดงในรูปของย่อหน้า หากต้องการให้ vi แสดงผลอักขระควบคุม เช่น tab ในรูปของ ^I เพื่อตรวจสอบข้อมูลที่มีอยู่ในแฟ้ม สามารถทำได้โดยใช้คำสั่ง **:set list** และยกเลิกได้ด้วยคำสั่ง **:set nolist**

Automatic Indentation

ในงานบางประเภท เช่น การพิมพ์โปรแกรมต้นฉบับที่เขียนขึ้นตามข้อกำหนดของภาษาต่างๆ นิยมใช้การย่อหน้าเพื่อแสดงโครงสร้างของการควบคุม ผู้ใช้สามารถกำหนดให้ vi ย่อหน้าอัตโนมัติได้เมื่อมีการขึ้นบรรทัดใหม่ โดย vi ทำการวาง cursor ตรงกับตำแหน่งแรกของบรรทัดบนซึ่งสะดวกในการเขียนโปรแกรม การดำเนินการเช่นนี้ทำได้โดยใช้คำสั่ง : **set autoindent** หรือ **:set ai** และยกเลิกด้วยคำสั่ง **:set noautoindent** หรือ **:set noai**

เมื่อมีการกำหนดให้ทำงานแบบ automatic indentation และอยู่ใน input mode การเลื่อน cursor ไปยังย่อหน้า (tab) ถัดไป ใช้คำสั่ง **Ctrl-T** และหากต้องการเลื่อน cursor ถอยหลังกลับไปยังย่อหน้า (tab) ที่ผ่านมาแล้วใช้ **Ctrl-D** จำนวนตัวอักขระที่เลื่อนไปในแต่ละครั้งสามารถกำหนดได้ ผ่านคำสั่ง set shiftwidth

Shift Width

Shift Width ใช้ควบคุมจำนวนตัวอักขระที่ใช้เป็นย่อหน้า (tab) ที่ใช้ร่วมกับคำสั่ง autoindent และทำงานร่วมกับ Ctrl-T และ Ctrl-D ในสภาวะ input mode เช่นหากต้องการให้มีการเลื่อน cursor ไปข้างหน้าครั้งละ 5 ตัวอักขระเมื่อมีการกดแป้น Ctrl-T ใช้คำสั่ง **:set shiftwidth=5**

คำสั่งของ vi ดังได้กล่าวมาแล้วเป็นคำสั่งพื้นฐานซึ่งเพียงพอสำหรับการใช้งานในระยะเริ่มต้น นอกจากคำสั่งเหล่านี้แล้ว vi ยังมีคำสั่งในระดับที่สูงขึ้นและทำงานได้ซับซ้อนยิ่งขึ้น เช่น คำสั่งในการค้นหาและเปลี่ยนแทนข้อความ (search and replace), คำสั่งกำหนด buffer ย่อยที่สามารถกำหนดชื่อได้ (named buffer), การออกจาก vi ไปทำงานที่ระบบปฏิบัติการชั่วคราว, การเรียกใช้คำสั่งของระบบปฏิบัติการและนำผลลัพธ์ที่ได้มาใช้ใน vi, การเปิดแฟ้มเพื่อแก้ไขพร้อมกันหลายแฟ้มเป็นต้น คำสั่งต่างๆเหล่านี้หากสนใจสามารถหาอ่านได้จากหนังสือวิธีการใช้ vi โดยละเอียด เช่น

Lamb, L. (1990) Learning the vi Editor, O'Reilly & Associates, Inc., CA

เรียบเรียงจาก

Sobell, M. G. (1995) A Practical Guide to the UNIX System, 3rd Edition, The Benjamin/Cummings Publishing Company, Inc. CA

Lamb, L. (1990) Learning the vi Editor, O'Reilly & Associates, Inc., CA

ผู้เรียบเรียง

จิระ จตุรานนท์, ภาควิชาวิทยาการคอมพิวเตอร์, คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา ชลบุรี

ภาคผนวก

เพื่อความสะดวกการใช้โปรแกรม vi สำหรับการพิมพ์โปรแกรมต้นฉบับภาษา C ในระบบปฏิบัติการ Unix ควรทำการสร้างแฟ้ม **.exrc** ซึ่งเป็นแฟ้มควบคุมการทำงานของ vi ไว้ที่ home directory โดยมีข้อมูลในแฟ้มดังนี้

set autoindent	กำหนดให้มีการย่อหน้าแบบอัตโนมัติ
set shiftwidth=5	กำหนดให้มีการย่อหน้าเมื่อใช้ ^D และ ^T เป็น 5 ตัวอักขระ
set showmatch	กำหนดให้มีการเลื่อน Cursor เพื่อแสดงวงเล็บที่สมนัยกัน
set showmode	กำหนดให้มีการแสดงสภาวะการทำงานของ vi ที่บรรทัดที่ 25

การใช้โปรแกรม Terminal Emulation เช่น Telnet หรือ Tera Term สำหรับจำลองหน้าจอ เพื่อเข้าใช้เครื่องแม่ข่ายจากระยะไกล ควรกำหนดให้จำลองแบบ (Emulation) เป็น VT100/ANSI และกำหนดจำนวนบรรทัด (Buffer Size) เป็น 24 บรรทัด เพื่อให้การใช้โปรแกรมได้ผลดีที่สุด และไม่เกิดปัญหาในการแป้นลูกศรสำหรับเลื่อน Cursor

เมื่อกำหนดการจำลองหน้าจอได้ถูกต้องแล้ว หากมีปัญหาในการใช้งาน ขอให้ตรวจสอบการกำหนดหน้าจอของระบบปฏิบัติการ Unix โดยใช้คำสั่ง **set** และมองหาบรรทัดที่มีตัวแปรระบบ TERM ซึ่งควรจะมีค่าเป็น TERM=vt100 หากตัวแปรนี้เป็นค่าอื่นให้ตั้งค่าเป็น vt100 โดยใช้คำสั่งดังนี้

```
$ TERM=vt100
```

การตรวจสอบดังกล่าวนี้จำเป็นมากในการใช้งานโปรแกรม vi และควรได้ตรวจสอบทุกครั้งเมื่อทำการ Login เข้าใช้ระบบจากเครื่องคอมพิวเตอร์ในห้องปฏิบัติการที่ใช้ระบบปฏิบัติการ Microsoft Windows

เมื่อมีการใช้งานเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการ Linux ในห้องปฏิบัติการ Linux ของภาควิชา และต้องการขีดความสามารถในการแสดงสีของคำสำคัญในภาษา C ให้ทำการกำหนดจอดังนี้

```
$ TERM=linux
```

และใช้คำสั่ง syntax on ที่ last line mode เมื่อเข้าสู่โปรแกรม vi แล้ว