บทที่ 9
โครงสร้างสื่อจัดเก็บ
(Storage Structure)

ระบบปฏิบัติการมีความสามารถทางตรรกะที่มองเห็นเป็นรูปร่างที่ประกอบด้วย 3 ส่วนคือ ส่วนที่ 1 คือ เรื่องของผู้ใช้และผู้เขียนโปรแกรมในระบบปฏิบัติการ ส่วนที่ 2 คือเรื่องโครงสร้างข้อมูลภายใน ที่ใช้ในระบบปฏิบัติการโดยใช้เครื่องมือการเชื่อมต่อระหว่างคอมพิวเตอร์ที่ถ่ายโอนข้อมูลจากกันและกันได้ และส่วนที่ 3 คือเรื่องของระดับต่ำสุดของระบบที่เกี่ยวกับระบบไฟล์และโครงสร้างของหน่วยความจำช่วย ขั้นแรกเราจะอธิบายถึงการเคลื่อนย้ายหัวอ่านบันทึกดิสก์และการแลกเปลี่ยนพื้นที่ และยังครอบคลุมไปถึงเรื่องความปลอดภัยของดิสก์และความแน่นอนของพื้นที่ที่ใช้เก็บข้อมูล

โครงสร้างของดิสก์

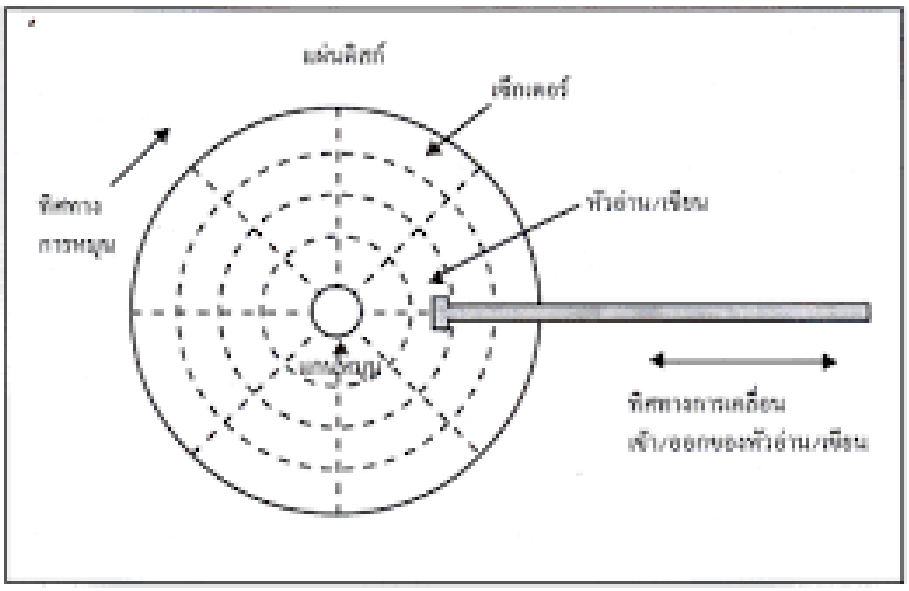
ดิสก์มีโครงสร้างที่ประกอบด้วยแผ่นจานกลมบาง ๆ หลายแผ่นเรียงซ้อนกันโดยที่บนผิวจานแต่ละแผ่นจะเคลือบด้วยอ๊อกไซด์ของแม่เหล็กทั้งสองหน้าและมีหัวอ่าน/บันทึกประจำทุกหน้า การบันทึกข้อมูลบนดิสก์จะบันทึกที่บนผิวหน้าของจาน ในการทำงานจานจะหมุนด้วยความเร็วสูงมาก บนผิวของจานจะถูกแบ่งเป็นวงช่องข้อมูล จำนวนมากถึงพัน ๆ วงเรียกว่า แทร็ก (Track) และในแต่ละวงหรือแทร็กจะถูกแบ่งออกเป็นส่วน ๆ แต่ละส่วนเรียกว่าเซ็กเตอร์ (Sector) และแต่ละวงที่อยู่ตรงกันของจานแต่ละแผ่นเรียกว่าไซลินเดอร์ (Cylinder) ดิสก์ 1 ตัวจะประกอบด้วยจานหลายแผ่นในชุดเดียวกันเรียกว่าดิสก์แพค (Disk Pack) ซึ่งจะอยู่ภายในกล่องโลหะหรือกล่องพลาสติกเพื่อป้องกันฝุ่นละอองและการกระทบกระแทก ดิสก์มี 2 ประเภทคือ

1. Remove-head disk เป็นดิสก์ที่มีหัวอ่าน/บันทึกประจำทุกหน้า ๆ ละ 1 หัว จะทำงานช้าเพราะต้องเสียเวลาในการเลื่อนหัวอ่าน/บันทึกไปยังแทร็กที่ต้องการ ดิสก์ประเภทนี้ราคาถูก
2. Fixed-head Disk เป็นดิสก์ที่มีหัวอ่าน/บันทึกประจำทุกหน้า และประจำทุกแทร็กของแต่ละหน้า มีการทำงานที่เร็วกว่าประเภทแรก เพราะการย้ายตำแหน่งการอ่าน/บันทึกของแต่ละหัวใช้วิธีการ switch หัวอ่าน/บันทึกไปยังแทร็กที่ต้องการ ดิสก์ประเภทนี้ราคาแพงกว่า

ระบบการทำงานของดิสก์แบ่งเป็น 2 ส่วน คือ ตัวเครื่องขับดิสก์ ซึ่งประกอบด้วยส่วนที่สำคัญคือ มอเตอร์ หัวอ่าน/บันทึก และวงจรอีเล็คทรอนิคต่าง ๆ ส่วนที่สอง เรียกว่า ตัวควบคุมอุปกรณ์ (Disk Controller) เป็นตัวรับคำสั่ง จากหน่วยประมวลผลกลางและสั่งให้ตัวเครื่องทำงานตามคำสั่งเหล่านั้น การแบ่งระบบจานบันทึกเป็น 2 ส่วนนี้ ทำให้ตัวควบคุมอุปกรณ์ตัวหนึ่งสามารถ

ควบคุมและสั่งการอุปกรณ์ได้หลายเครื่อง เมื่อต้องการเพิ่มเครื่องขับจานบันทึกอีก เครื่องในระบบที่มีเครื่องขับอยู่แล้ว ก็เพียงเพิ่มแต่ตัวเครื่องเข้ามาในระบบเท่านั้น ตัวควบคุมอุปกรณ์บางตัวมีหน่วยความจำพิเศษ (Cache Memory) ภายในใช้เก็บข้อมูลที่เพิ่งจะอ่านหรือเขียนมา เพื่อช่วยลดเวลาในการอ่านในกรณีที่ข้อมูลเดิมเคยอ่านมาแล้ว

การอ้างอิงข้อมูลในดิสก์ใช้การระบุตำแหน่งซึ่งประกอบด้วย หมายเลขหน้าพื้นผิว (Surface) และหมายเลขวงข้อมูล (Track) โดยทั่วไปเราสามารถอ่านข้อมูลจากวงข้อมูลแต่ละวงในจานบันทึกหนึ่งๆ ได้โดยไม่ต้องเลื่อนหัวอ่านเลยในแต่ละวงข้อมูลจะถูกแบ่งเป็นช่วง (Block) แต่ละช่วงมักมีขนาดคงที่แน่นอนซึ่งกำหนดโดยฮาร์ดแวร์ของเครื่องขับข้อมูลในแต่ละช่วงนี้ เรียกว่า เซกเตอร์ (Sectors) หัวอ่าน/เขียน สามารถอ่าน หรือเขียนข้อมูลได้เล็กที่สุดคือ 1 เซกเตอร์ ดิสก์มีโครงสร้างการทำงาน ดังรูปที่ 9.1



รูปที่ 9.1 โครงสร้างการทำงานของดิสก์

(ที่มา http://site/rabbpdibatikar1/kar-cad-tarang-wela-kar-chi-disk)

การอ่านหรือบันทึกข้อมูลลงในดิสก์อย่างน้อยที่สุด คือ 1 เซกเตอร์ ขนาดของเซกเตอร์นี้อาจเป็นได้ตั้งแต่ 32 ไบท์ จนถึง 4096 ไบท์ ใน 1 ช่องข้อมูลอาจมีได้ตั้งแต่ 4 จนถึง 32 เซกเตอร์ ดิสก์ 1 หน้า (Surface) จะมีวงข้อมูลได้ตั้งแต่ 20 จนถึง 1500 วง ในการอ้างอิงเซกเตอร์หนึ่ง ๆ นั้นจะต้องบอกตำแหน่งเป็นเลขหน้า เลขแทร็ก และเลขเซกเตอร์

ระหว่างเซกเตอร์ จะมีช่องว่าง หรือเครื่องหมายคั่นเพื่อให้หัวอ่านสามารถทราบตำแหน่งข้อมูลที่ถูกต้องได้ หัวอ่านจะถูกเลื่อนด้วยกลไกไปยังแทร็กที่ต้องการเรียกว่า เวลาในการเลื่อนหัวอ่าน/บันทึก (Seek Time) แล้วรอจนเซกเตอร์ที่ต้องการหมุนมาถึงตำแหน่งหัวอ่านบันทึก

เรียกว่า เวลาในการหมุนหา (Latency Time) ส่วนการเลือกหน้า ไม่ต้องมีการเคลื่อนไหวใด ๆ เพราะใช้ระบบสวิทซ์ตำแหน่งภายในตัวเครื่อง จะเห็นได้ว่า เวลาในการเข้าถึงข้อมูล จะขึ้นกับระยะห่างระหว่างแทร็กและเซกเตอร์ ยิ่งตำแหน่งห่างมากก็ยิ่งเสียเวลามาก

การส่งผ่านข้อมูลระหว่างหน่วยความจำหลักกับดิสก์ จะส่งเป็นเซกเตอร์เนื่องจาก ค่าตำแหน่งของเซกเตอร์หนึ่งๆ มี 3 ส่วน คือ หน้า ช่องข้อมูล และหมายเลขเซกเตอร์เราอาจจะมองจานบันทึกเป็นแอร์เรย์ 3 มิติ ของชุดข้อมูล หรือ เซกเตอร์ ได้ระบบปฏิบัติการทั่วไป จะกำหนดให้แอร์เรย์ 3 มิตินี้ เป็นเพียงแอร์เรย์แบบ 1 มิติ โดยเรียงค่าเริ่มต้นที่หมายเลขแทร็ก ตามด้วยหมายเลขหน้าและหมายเลขเซกเตอร์

ถ้าให้ s เป็นจำนวนเซกเตอร์ใน 1 แทร็ก

t เป็นจำนวนหน้าหรือผิวหน้าจานบันทึก

i เป็นหมายเลขแทร็ก

j เป็นหมายเลขหน้า

k เป็นหมายเลขเซกเตอร์

b เป็นตำแหน่งที่แสดงชุดข้อมูลนี้

จะได้ว่า

$$b = (k / s) * (j / i * t)$$

ข้อดีของดิสก์ในการเก็บแฟ้มข้อมูล คือ

1. สามารถจะเขียนซ้ำได้ โดยการอ่านข้อมูลจากดิสก์ ทำการแก้ไข แล้วบันทึกกลับไปยังที่เดิมได้

2. สามารถอ้างอิง หรือ เข้าถึงข้อมูล บนดิสก์ ณ ตำแหน่งใด ๆ ก็ได้ ดังนั้น การอ้างอิงแฟ้มข้อมูลในดิสก์จะเป็นแบบตามลำดับหรือแบบสุ่มก็ได้ การเปลี่ยนแฟ้มข้อมูลหนึ่ง ไปสู่อีกแฟ้มหนึ่งก็ทำได้ง่ายเพียงเลื่อนหัวอ่านไปยังช่องข้อมูลที่ถูกต้อง และรอให้ เซกเตอร์ที่ต้องการหมุนมาตรงเท่านั้น

การจัดตารางเวลาของดิสก์

ด้วยเหตุที่การทำงานของดิสก์นั้นมีการบันทึกจัดเก็บข้อมูลจำนวนมาก และมีการเข้าถึงแฟ้มข้อมูลจำนวนมาก และเวลาที่เข้ถึงข้อมูลในแต่ละแฟ้มต้องใช้เวลาของการเข้าถึงตำแหน่งแทร็คและเวลาการเข้าถึงตำแหน่งเซ็กเตอร์ ดังนั้นการที่จะให้ดิสค์ ทำงานได้อย่างมีประสิทธิภาพ ระบบปฏิบัติการจะต้องมีการควบคุมให้การบริการของดิสก์รวดเร็วและมีประสิทธิภาพ

ดิสก์ที่สามารถเข้าถึงข้อมูลได้เร็วประกอบด้วย 3 ส่วน คือ ข้อมูลในบล็อกบนดิสก์ ในระบบขั้นแรกจะย้ายการอ่านที่เหมาะกับแทร็ก (Track) หรือของไซลินเดอร์และหัวอ่านนั้นจะย้ายตามเวลาที่กำหนด คือหัวอ่านจะอ่านจากขวาของแทร็ก (Track) และก็จะรอหน่วยที่ต้องการบล็อกในรอบการอ่านหรือเขียนหัวอ่าน แต่จะช้าในขั้นตอนสุดท้ายความจริงแล้วการโอนย้ายของข้อมูลระหว่างดิสก์และหน่วยความจำหลักสามารถเอาตำแหน่งและส่วนสุดท้าย การโอนในเวลา คือผลรวมในการร้องข้อการให้บริการดิสก์ คือการสรุปเวลาในการค้นหาข้อมูล การทำงานเวลา และเวลาในการโอนข้อมูล

เมื่อกระบวนการต้องการการรับข้อมูลจากดิสก์หรือส่งข้อไปสู่ดิสก์ จึงใช้คำสั่งจากการเรียกระบบของระบบปฏิบัติการ จะต้องรู้ข้อมูลที่เกี่ยวข้องหลายอย่าง คือ

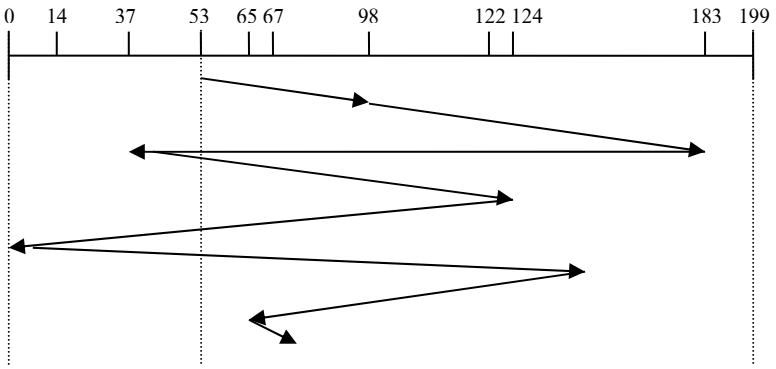
1. การปฏิบัติการนี้เป็นการนำเข้า (Input) หรือการส่งออก (Output) หรือไม่
2. ตำแหน่งของดิสก์ที่จะใช้ในการโอนย้ายข้อมูล
3. ตำแหน่งของหน่วยความจำสำหรับการโอนย้ายข้อมูล
4. จำนวนไบต์ของข้อมูลที่ถูกโอนย้าย

ถ้าเครื่องขับดิสก์และตัวควบคุม (Controller) ว่าง การร้องขอจะได้รับบริการทันที ถ้ามันไม่ว่าง การร้องขอใหม่จำเป็นต้องนำไปไว้ในแถวคอย สำหรับระบบ Multiprogramming ที่มีหลายกระบวนการแถวคอยของดิสก์ (Disk Queue) อาจมีการร้องขออยู่หลายตัว ดังนั้นเมื่อการร้องขอหนึ่งได้รับบริการแล้วระบบปฏิบัติการจึงมีโอกาสที่จะเลือกการร้องขอถัดไปมาให้บริการ

การจัดตารางเวลาแบบมาก่อนได้ก่อน (FCFS Scheduling)

รูปแบบที่ง่ายที่สุดของการจัดตารางเวลาของดิสก์ คือ มาก่อนได้ก่อน (First Come First Served : FCFS) พิจารณาตัวอย่าง ถ้าแถวคอยของดิสก์มีการร้องขอการรับส่งข้อมูลของบล็อกบนไซลินเดอร์ ดังนี้ 98, 183, 37, 122, 14, 124, 65 และ 67 ตามลำดับ ถ้าหัวอ่านเริ่มต้นที่ไซลินเดอร์ที่ 53 มันจะเริ่มเคลื่อนในครั้งแรกจาก 53 ไป 98 แล้ว จึงไป 183, 37, 122, 14, 124, 65 และสุดท้ายที่ 67 สำหรับการเคลื่อนย้ายหัวอ่านทั้งหมด 640 ไซลินเดอร์ การจัดตารางเวลานี้คือแผนภาพดังรูปที่ 9.2 ซึ่งปัญหาของการจัดตารางเวลาวิธีนี้ดูได้จากการเดินทางของหัวอ่านในแนวกว้างจากตำแหน่งที่ 122 ไป 14 แล้วกลับมา 124 ถ้าการร้องขอสำหรับไซลินเดอร์ 37 และ 17 ได้รับการบริการต่อกัน ก่อนหรือหลังการร้องขอที่ 122 และ 124 การเคลื่อนย้ายหัวอ่านทั้งหมดน่าจะลดลงอย่างมาก ดังนั้นสมรรถนะ (Performance) การควบคุมการดำเนินการควรจะได้รับการปรับปรุง

Queue = 98, 183, 37, 122, 14, 124, 65, 67
หัวอ่านเริ่มต้นอยู่ที่ตำแหน่ง 53



รูปที่ 9.2 การจัดตารางเวลาของดิสก์แบบมาก่อนได้ก่อน (FCFS)

การจัดตารางเวลาแบบการค้นหาสั้นที่สุดได้ก่อน (SSTF Scheduling)

การบริการการร้องขอทั้งหมดที่อยู่ใกล้ตำแหน่งของหัวอ่านในขณะนั้น ก่อนจะย้ายหัวอ่านไกลออกไปเพื่อบริการการร้องขออื่นเป็นเหตุผลที่ดี จึงมีการคิดสมมติฐานใหม่ขึ้นมา คือพื้นฐานสำหรับวิธีเวลาในการค้นหาสั้นที่สุดได้รับการบริการก่อน (Shortest Seek Time First : SSTF Algorithm)

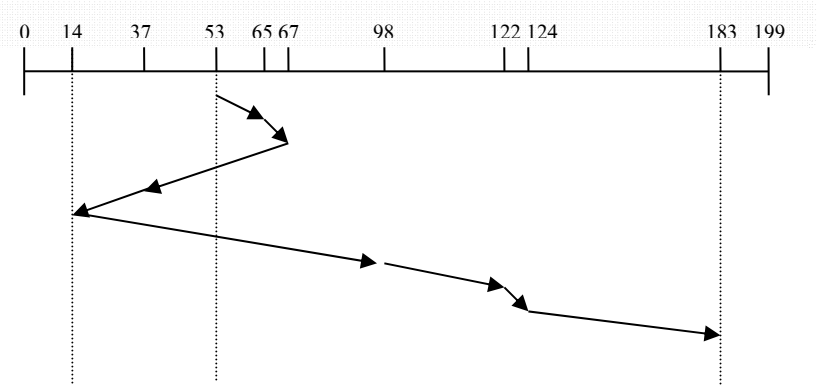
วิธี SSTF จะเลือกการร้องขอที่มีเวลาในการค้นหาน้อยที่สุดจากตำแหน่งปัจจุบันของหัวอ่านเมื่อเวลาในการค้นหาเพิ่มขึ้นด้วยจำนวนของไซลินเดอร์ที่ถูกอ่านโดยหัวอ่าน SSTF จะเลือกการร้องขอที่ใกล้ที่สุดกับตำแหน่งปัจจุบันของหัวอ่าน

จากตัวอย่างที่แล้ว การร้องขอที่ใกล้กับตำแหน่งเริ่มต้นของหัวอ่านที่สุด (53) คือที่ไซลินเดอร์ 65 เมื่อเราเคลื่อนมาที่ไซลินเดอร์ 65 การร้องขอที่ใกล้ที่สุดถัดไป คือ ที่ไซลินเดอร์ 67 จากนั้นการร้องขอที่ไซลินเดอร์ 37 จะใกล้กว่า 98 ดังนั้น 37 จะได้รับบริการถัดไป ต่อมารับบริการการร้องขอที่ไซลินเดอร์ 14 แล้วเป็น 98, 122, 124 และสุดท้ายที่ 183 (ดังรูปที่ 9.3) ผลลัพธ์ของวิธีการจัดตารางแบบนี้การเคลื่อนย้ายหัวอ่านทั้งหมดมีเพียง 236 ไซลินเดอร์เท่านั้นน้อยกว่า 1 ใน 3 ของระยะทางของวิธี FCFS

การจัดตารางแบบ SSTF จำเป็นที่สุดสำหรับรูปแบบของการจัดตารางแบบ SJF และเหมือนกับ SJF มันอาจจะเกิดปัญหาการรอแบบไม่มีกำหนด (Starvation) ซึ่งอาจจะมีการร้องขอเข้ามาถึงที่เวลาต่าง ๆ กัน สมมติว่าเรามีการร้องขอ 2 ตัวในแถวคอย สำหรับไซลินเดอร์ 14 และ 186 และขณะที่กำลังบริการการร้องขอจาก 14 การร้องขอใหม่ใกล้ 14 มาถึง การร้องขอใหม่นี้จะได้รับบริการเป็นรายถัดไป ทำให้การร้องขอที่ 186 ต้องรอก่อน ในขณะที่การร้องขอนี้กำลังได้รับบริการ การร้องขออีกตัวที่ใกล้ 14 มาถึงอีก ในทางทฤษฎีถ้าสายการร้องขออย่างต่อเนื่องที่ใกล้อีก

ตัวหนึ่งมาถึง จะเป็นเหตุให้การร้องขอสำหรับไซลินเดอร์ 186 จะต้องรออย่างไม่สิ้นสุดสิ่งนี้จะเป็นการเพิ่มแถวคอยการร้องขอให้ยาวขึ้น

Queue = 98, 183, 37, 122, 14, 124, 65, 67
หัวอ่านเริ่มต้นอยู่ที่ตำแหน่ง 53



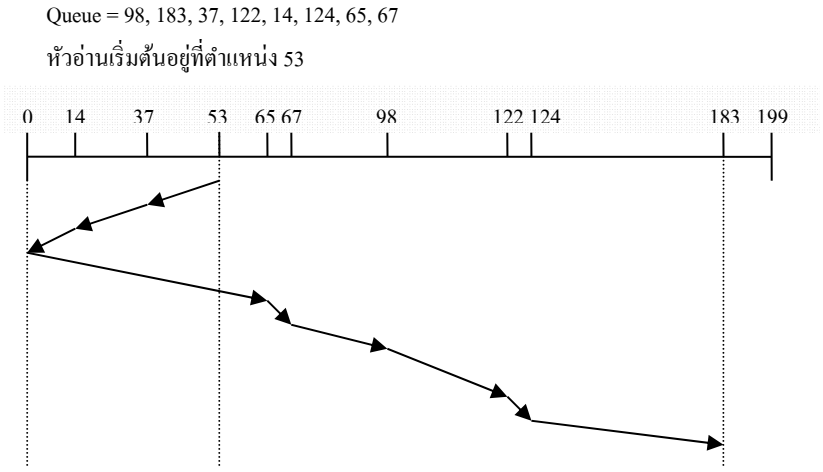
รูปที่ 9.3 การจัดตารางของดิสก์แบบเวลาในการค้นหาสั้นที่สุดได้ก่อน (SSTF)

ถึงแม้ว่าวิธี SSTF คือการปรับปรุงอย่างมากจาก FCFS แต่มันยังไม่ดีที่สุด จากตัวอย่างเราสามารถทำได้ดีกว่าโดยการเคลื่อนหัวอ่านจาก 53 ไป 37 ถึงแม้ว่า 37 จะไม่ใกล้ที่สุด จากนั้นไป 14 ก่อนจะกลับไป 65, 67, 98, 122, 124 และ 183 กลยุทธ์นี้เป็นการลดการเคลื่อนย้ายหัวอ่านทั้งหมดเหลือ 208 ไซลินเดอร์

การจัดตารางเวลาแบบกวาด (SCAN Scheduling)

ในวิธีแบบกวาด (SCAN) แขนของดิสก์เริ่มต้นที่จุดสิ้นสุดจุดหนึ่งของดิสก์ (A) และเคลื่อนไปยังจุดสิ้นสุดอื่น (B) การบริการการร้องขอจะทำได้เมื่อมันมาถึงในแต่ละไซลินเดอร์จนกระทั่งมันไปถึงจุดสิ้นสุดอื่นของดิสก์ (B) ณ ที่จุดสิ้นสุดอื่น (B) ทิศทางของการเคลื่อนของหัวอ่านจะถูกย้อนกลับและให้บริการต่อไป หัวอ่านจะกวาดไปข้างหน้าและข้างหลังผ่านดิสก์ไปเรื่อย ๆ พิจารณาตัวอย่างเดิม

ก่อนจะใช้วิธีจัดตารางแบบกวาดกับการร้องขอบนไซลินเดอร์ 98, 183, 37, 122, 14, 124, 65 และ 67 เราจำเป็นต้องรู้ทิศทางของการเคลื่อนหัวอ่านก่อน โดยถ้าตอนนี้ตำแหน่งปัจจุบันของหัวอ่านคือ 53 ถ้าแขนของดิสก์กำลังเคลื่อนไปทาง 0 หัวอ่านจะบริการ 37 และจากนั้นเป็น 14 ที่ไซลินเดอร์ 0 แขนจะย้อนกลับและจะเคลื่อนผ่านจุดสิ้นสุดอื่นของดิสก์ (ที่ไม่ใช่ 0) การบริการการร้องขอจะทำที่ 65, 67, 98, 122, 124 และ 183 (ดังรูปที่ 9.4)



รูปที่ 9.4 การจัดตารางของดิสก์แบบกวาด (SCAN)

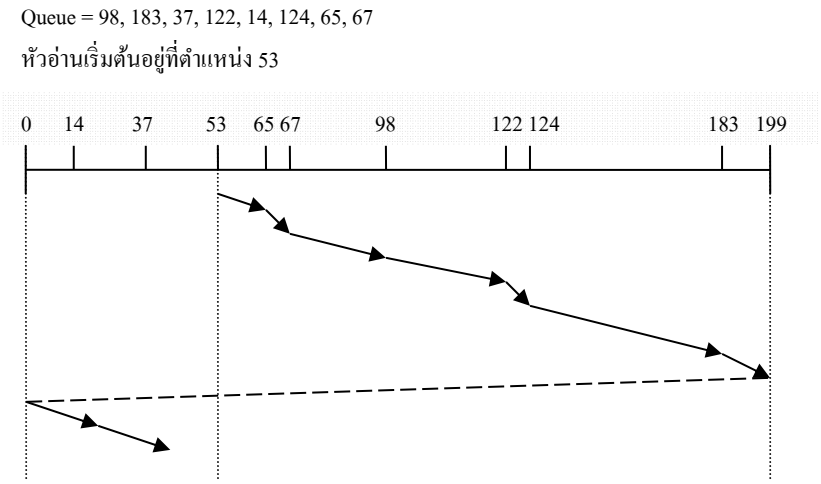
ถ้าการร้องขอที่พึ่งจะมาอยู่ในแถวคอยหน้าของหัวอ่าน มันก็จะได้รับบริการเกือบจะทันที ส่วนการร้องขอที่พึ่งจะมาถึงหลังหัวอ่าน ก็จะต้องรอจนกระทั่งแขนเคลื่อนไปถึงจุดสิ้นสุดของดิสก์แล้วย้อนกลับมาอีกครั้ง

วิธีแบบกวาด (SCAN Algorithm) นี้บางครั้งก็เรียกว่า วิธีแบบบันไดเลื่อน (Elevator Algorithm) เพราะแขนของดิสก์มีพฤติกรรมคล้ายบันไดเลื่อนในตึก การบริการเริ่มต้นกับการร้องขอทั้งหมดในขาขึ้น และจากนั้นจะบริการการร้องขอย้อนกลับมาอีกทาง

สมมติว่าการกากระจายของการร้องขอสำหรับไซลินเดอร์เป็นแบบเดียวกัน พิจารณาความหนาแน่นของการร้องขอเมื่อหัวอ่านมาถึงจุดสิ้นสุดจุดหนึ่งแล้วย้อนทิศทางกลับ ณ จุดนี้มีการร้องขอเพียงเล็กน้อยที่สัมพันธ์กันที่อยู่หน้าหัวอ่าน ดังนั้นไซลินเดอร์เหล่านี้ก็จะได้รับบริการ แต่ความหนาแน่นที่มากที่สุดของการร้องขออยู่ที่อีกจุดสิ้นจุดหนึ่งของดิสก์ การร้องขอเหล่านี้ก็ต้องรอนานมาก ดังนั้นทำไมเราไม่ไปทางด้านนั้นก่อน นั้นคือความคิดของวิธีถัดไป

การจัดตารางเวลาแบบกวาดเป็นวง (C – SCAN Scheduling)

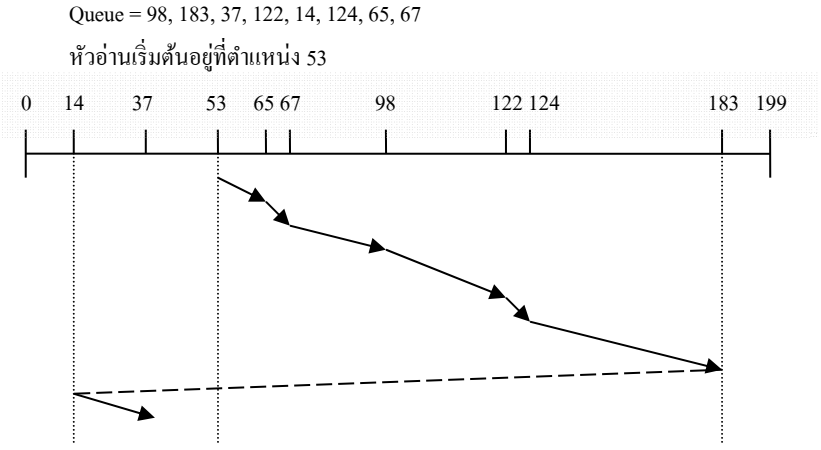
การกวาดเป็นวง (Circular SCAN : C - SCAN) คือ การเปลี่ยนแปลงของการกวาดซึ่งถูกออกแบบมาเพื่อจัดการกับเวลารอคอยที่มากกว่า 1 รูปแบบ เหมือนกับแบบกวาด แบบกวาดเป็นวงจะเคลื่อนหัวอ่านจากจุดสิ้นสุดจุดหนึ่ง (A) ของดิสก์ไปสู่อีกจุดหนึ่ง (B) มันจะกลับไปสู่จุดเริ่มต้นของดิสก์ในทันที โดยไม่บริการการร้องขอใด ๆ ในขากลับนี้ (ดังรูปที่ 9.5) การจัดตารางเวลาแบบกวาดเป็นวงนี้เป็นสิ่งจำเป็นสำหรับการจัดการกับไซลินเดอร์แบบวงกลม ซึ่งล้อมรอบจากไซลินเดอร์สุดท้ายไปสู่ไซลินเดอร์แรก



รูปที่ 9.5 การจัดตารางเวลาของดิสก์แบบกวาดเป็นวง (C – SCAN)

การจัดตารางเวลาแบบ LOOK (LOOK Scheduling)

จะเห็นว่า ทั้งแบบกวาดและกวาดเป็นวงจะเคลื่อนแขนดิสก์ไปทั่วทั้งดิสก์ ในทางปฏิบัติไม่มีวิธีไหนที่เอาไปใช้จริงได้ ในความเป็นจริงแขนดิสก์จะไปไกลเพียงแค่การร้องขอสุดท้ายเท่านั้นในแต่ละทิศทาง จากนั้นมันก็จะย้อนกลับไปอีกทางทันทีโดยไม่ต้องไปจุดสิ้นสุดของดิสก์ การทำเช่นนี้เรียกว่า LOOK และ C – LOOK เพราะมันมองการร้องขอก่อนจะเคลื่อนไปยังทิศทางที่ได้ดังรูปที่ 9.6



รูปที่ 9.6 การจัดตารางเวลาของดิสก์แบบ C - LOOK

การเลือกวิธีการจัดตารางเวลาของดิสก์ (Selection of a Disk – Scheduling Algorithm)

วิธี SSTF ดูจะธรรมดาและความเป็นไปได้ตามธรรมชาติ วิธี SCAN และ C – SCAN ใช้ได้ดีสำหรับระบบที่มีภาระหนักบนดิสก์ เพราะมักจะเกิดปัญหาการแช่เย็นได้ยาก สำหรับรายการของการร้องขอโดยเฉพาะ มันเป็นไปได้ที่จะกำหนดลำดับที่ดีทีสุด แต่ในแง่ของการคำนวณจำเป็นต้องหาการจัดตารางเวลาที่ดีที่สุด ซึ่งอาจจะไม่ประหยัดไปกว่าแบบ SSTF หรือ SCAN

สำหรับดิสก์ยุคใหม่ เวลาในการหมุนหัวอ่าน (Rotational Latency) มีขนาดใกล้เคียงกับเวลาในการค้นหาเฉลี่ย (Average Seek Time) แต่เป็นเรื่องยากสำหรับระบบปฏิบัติการที่จะจัดตารางทางตรรกะ ผู้ผลิตดิสก์ได้ช่วยแก้ปัญหานี้โดยใช้วิธีการจัดตารางเวลาของดิสก์ในฮาร์ดแวร์ควบคุม (Controller Hardware) ที่สร้างไว้ในเครื่องขับดิสก์ ถ้าระบบปฏิบัติการส่งการร้องขอเป็นกลุ่มมาที่ตัวควบคุม ตัวควบคุมสามารถจัดแถวคอยพวกมัน และจากนั้นก็ทำการจัดตารางเวลาเพื่อปรับปรุงทั้งเวลาในการค้นหาและเวลาในการหมุนหัวอ่าน ถ้ามีแต่สมรรถนะในการรับส่งข้อมูลเท่านั้นที่เราสนใจ ระบบปฏิบัติการจะยินดีมากที่จะโอนความรับผิดชอบของการจัดตารางเวลาของดิสก์ไปให้ฮาร์ดแวร์ของดิสก์ทำแทน

ในทางปฏิบัติ ระบบปฏิบัติการมีข้อจำกัดอื่น ๆ สำหรับการบริการการร้องขอ ตัวอย่างเช่น การจัดสรรหน้าตามคำขอ (Demand Paging) อาจมีศักดิ์สูงกว่า การรับส่งข้อมูลของโปรแกรมประยุกต์ (Application I/O) และการเขียนก็เป็นเรื่องที่เร่งด่วนกว่าการอ่าน ถ้า Cache กำลังทำงานออกนอกหน้าว่าง ดังนั้นอาจจะต้องการการรับประกันลำดับของกลุ่มของดิสก์ที่จะเขียน เพื่อทำให้ระบบแฟ้มข้อมูลแข็งแรงเพื่อเผชิญกับการชนของระบบ (System Crash) ลองพิจารณาดูว่าอะไรจะเกิดขึ้น ถ้าระบบปฏิบัติการจัดสรรหน้าของดิสก์ให้แฟ้มข้อมูลแล้วโปรแกรมประยุกต์เขียนข้อมูลลงหน้าก่อนที่ระบบปฏิบัติการจะมีโอกาสปรับปรุง หรือแจ้งให้ดิสก์ทราบว่าหน้านั้นว่าง เพื่อทำให้ความต้องการเหมาะสม ระบบปฏิบัติการอาจจะเลือกที่จะทำการจัดตารางเวลาของดิสก์เองแล้วป้องกันการร้องขอให้ตัวควบคุมดิสก์ทีละตัว

การจัดการดิสก์

การควบคุมการใช้งานดิสก์มีหลายรูปแบบที่ระบบปฏิบัติการที่จะดำเนินการกับดิสก์ ในส่วนนี้จะอธิบายในการควบคุมขั้นพื้นฐานของระบบปฏิบัติการในการจัดการดิสก์ ดังต่อไปนี้

การจัดรูปแบบของดิสก์ (Disk Formatting)

ก่อนที่ดิสก์จะสามารถบรรจุข้อมูลได้ มันต้องถูกแบ่งเป็นเซกเตอร์ซึ่งตัวควบคุมดิสก์สามารถอ่านและเขียนได้ กระบวนการนี้เรียกว่า การจัดรูปแบบระดับต่ำ (Low-Level Formatting)

หรือการจัดรูปแบบเชิงกายภาพ (Physical Formatting) การจัดรูปแบบระดับต่ำจะเติมโครงสร้างข้อมูลพิเศษลงไปในแต่ละเซกเตอร์ของดิสก์ โครงสร้างข้อมูลสำหรับเซกเตอร์ประกอบด้วยหัวอ่าน (Header) พื้นที่ของข้อมูล (Data Area มักมีขนาด 512 ไบต์) Trailer หัวอ่าน และ Trailer บรรจุสารสนเทศที่ถูกใช้โดยตัวควบคุมดิสก์ เช่น หมายเลขเซกเตอร์ และรหัสถูก - ผิด [Error Correcting Code (ECC)] เมื่อตัวควบคุมเขียนเซกเตอร์ของข้อมูลในระหว่างการรับส่งปกติ ECC จะถูกแก้ไขด้วยค่าที่คำนวณจากไบต์ทั้งหมดในพื้นที่ข้อมูล เมื่อเซกเตอร์ถูกอ่าน ECC จะถูกคำนวณใหม่และจะถูกเปรียบเทียบกับค่าที่เก็บเอาไว้ ถ้าหมายเลขที่ถูกเก็บไว้และที่ได้จากการคำนวณต่างกัน การไม่เข้าคู่นี้ชี้ให้เห็นว่า พื้นที่ข้อมูลของเซกเตอร์กลายเป็นข้อผิดพลาด และเซกเตอร์ของดิสก์นั้นอาจไม่ดี เกิดเซกเตอร์เสีย

ECC คือ รหัสถูก – ผิด เพราะมันบรรจุสารสนเทศที่เพียงพอ ซึ่งถ้ามีข้อมูล 1 หรือ 2 บิต ถูกทำให้ผิดพลาด ตัวควบคุมจะสามารถแยกได้ว่า บิตไหนถูกเปลี่ยนแปลงและสามารถคำนวณได้ว่า ค่าที่ถูกต้องควรจะเป็นค่าใด กระบวนการของ ECC จะถูกทำอย่างอัตโนมัติโดยตัวควบคุมไม่ว่าเซกเตอร์จะถูกอ่านหรือถูกเขียน

บูตบล็อก (Boot Block)

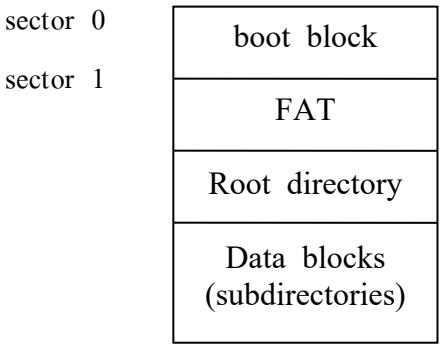
เมื่อคอมพิวเตอร์เริ่มทำงาน (เช่น เปิดเครื่องบูตเครื่องใหม่) มันจำเป็นต้องมีโปรแกรมเริ่มต้นเพื่อการทำงาน โปรแกรมนี้คือ Bootstrap มันเป็นการเริ่มต้นทั้งหมดของระบบ เริ่มจากซีพียู ตัวควบคุมอุปกรณ์ และหน่วยความจำหลักแล้วจึงเริ่มระบบปฏิบัติการ โดยมันจะหาแก่น (Kernel) ของระบบปฏิบัติการในดิสก์แล้วนำ (Load) แก่นนั้นเข้าสู่หน่วยความจำ และกระโดดไปสู่ตำแหน่งเริ่มต้นเพื่อเริ่มการทำงานของระบบปฏิบัติการ

คอมพิวเตอร์ส่วนใหญ่ โปรแกรม Bootstrap จะถูกเก็บไว้ในรอม (ROM : Read – Only Memory) เพราะรอมไม่จำเป็นต้องใช้ตั้งแต่แรก และมันเป็นตำแหน่งที่แน่นอนคงที่ซึ่งหน่วยประมวลผลสามารถเริ่มทำงานเพื่อเปิดเครื่องหรือรีเซต (Reset) เครื่อง และเพราะรอมใช้อ่านได้อย่างเดียว มันจึงไม่สามารถจะติดไวรัสได้ แต่ปัญหาก็คือ การเปลี่ยนแปลงโค้ดของ Bootstrap จำเป็นต้องเปลี่ยนชิปของรอมด้วย ด้วยเหตุนี้ระบบส่วนใหญ่จะเก็บโปรแกรม Bootstrap Loader เล็ก ๆ ไว้ในรอมบูต (Boot ROM) เมื่อต้องการใช้งานก็ค่อยนำโปรแกรม Bootstrap แบบเต็มเข้ามาจากดิสก์ ดังนั้นโปรแกรม Bootstrap แบบเต็มจะสามารถเปลี่ยนแปลงได้ง่าย เราเพียงแค่เอาเวอร์ชั่นใหม่เขียนลงบนดิสก์เท่านั้นโปรแกรม Bootstrap แบบเต็มนั้นถูกเก็บไว้ในส่วนที่เรียกว่า บูตบล็อก (Boot Blocks) ที่ตำแหน่งตายตัวบนดิสก์ ดิสก์ที่มีส่วนที่ใช้บูตคือ บูตดิสก์ (Boot Disk) หรือชิสเต็มดิสก์ (System Disk)

ตัวควบคุมดิสก์จะอ่านคำสั่งในบูตรอมจากบูตบล็อกเข้าสู่หน่วยความจำ (ยังไม่มีตัวควบคุมอุปกรณ์ใด ๆ ถูกนำเข้าหน่วยความจำในตอนนี้) จากนั้นก็เริ่มทำงานตามคำสั่งดังกล่าว โปรแกรม Bootstrap แบบเต็มสามารถที่จะนำระบบปฏิบัติการจากตำแหน่งที่ไม่ตายตัวมาไว้บนดิสก์ได้ แล้วจึงเริ่มให้ระบบปฏิบัติการทำงาน ดังนั้นโปรแกรม Bootstrap แบบเต็มจึงมีขนาดเล็กได้ ตัวอย่างเช่น MS – DOS ใช้บล็อกขนาด 512 ไบต์ สำหรับการบูตเครื่อง (ดังรูปที่ 9.7)

บล็อกเสีย (Bad Block)

ในระบบดิสก์อย่างง่าย เช่น ดิสก์ที่มีตัวควบคุมแบบ IDE เราสามารถจัดการกับบล็อกเสียได้ (โดยผู้ใช้สั่งให้ทำ) เช่น คำสั่ง Format ของ MS – DOS เป็นการจัดรูปแบบทางตรรกะและจะทำการหาบล็อกเสียบนดิสก์ ถ้าหากเจอมันจะเขียนค่าพิเศษไว้ในช่องของ FAT ที่ตรงกันเพื่อบอกให้โปรแกรมย่อยต่าง ๆ ไม่ให้ใช้บล็อกนั้น ถ้าบล็อกเกิดเสียในระหว่างการทำงานปกติ โปรแกรมพิเศษ (เช่น Chkdsk) ต้องถูกสั่งให้ทำงานเพื่อค้นหาบล็อกเสีย และล็อคบล็อกเหล่านั้นไว้ไม่ให้ใช้ ข้อมูลที่อยู่ในบล็อกเสียก็จะเสียไปโดยปริยาย



รูปที่ 9.7 โครงร่างของดิสก์ในระบบ MS – DOS

สำหรับดิสก์แบบ SCSI สามารถกู้บล็อกเสียคืนได้ โดยตัวควบคุมจะเก็บรายการของบล็อกเสียบนดิสก์ รายการนี้ถูกเก็บมาตั้งแต่ตอนจัดรูปแบบดิสก์ระดับต่ำจากโรงงาน และถูกทำให้แก้ไขตลอดเวลาของการใช้ดิสก์ การจัดรูปแบบระดับต่ำจะจัดเซกเตอร์อะไรไหล่ที่ระบบปฏิบัติการมองไม่เห็นไว้ แล้วตัวควบคุมจะทำการแทนที่เซกเตอร์ทางตรรกะที่เสียแต่ละเซกเตอร์ด้วยเซกเตอร์อะไหล่ วิธีการนี้เรียกว่า การเก็บเซกเตอร์อะไหล่ (Sector Sparing) หรือการเปลี่ยนที่ (Forwarding)

ตัวอย่างของการเปลี่ยนแปลงเซกเตอร์เสียอาจเป็นดังนี้

1. ระบบปฏิบัติการพยายามอ่านบล็อกทางตรรกะที่ 87 (สมมุติว่าเป็นเซกเตอร์ที่เสีย)

2. ตัวควบคุมจะทำการคำนวณรหัสถูก – ผิด (ECC) และพบว่าเซกเตอร์นั้นเสีย มันจะรายงานการค้นพบนี้ไปสู่ระบบปฏิบัติการ

3. คราวหน้าเมื่อบูตระบบใหม่ คำสั่งพิเศษจะทำงานเพื่อบอกตัวควบคุม SCSI เพื่อแทนเซกเตอร์ที่เสียด้วยเซกเตอร์อะไหล่

4. หลังจากนั้น เมื่อระบบร้องขอบล็อกทางตรรกะที่ 87 การร้องขอนั้นจะถูกแปลไปยังตำแหน่งของเซกเตอร์ที่ถูกแทนที่โดยตัวควบคุม

จะเห็นได้ว่าการเปลี่ยนทิศทางโดยตัวควบคุม อาจจะทำให้การจัดตารางเวลาดิสก์ของระบบปฏิบัติการผิดพลาด ด้วยเหตุนี้ดิสก์ส่วนใหญ่ถูกจัดรูปแบบเพื่อเตรียมเซกเตอร์อะไหล่ไว้เล็กน้อยในแต่ละไซลินเดอร์ เมื่อพบบล็อกเสีย ตัวควบคุมจะใช้เซกเตอร์อะไหล่จากไซลินเดอร์เดียวกัน ถ้าทำได้อีกทางเลือกหนึ่งของการเก็บเซกเตอร์อะไหล่ ตัวควบคุมสามารถสั่งให้แทนที่บล็อกเสียและย้ายเซกเตอร์

การแทนที่ของบล็อกเสียโดยทั่วไปยังไม่เป็นกระบวนการอย่างอัตโนมัติเพราะข้อมูลในบล็อกเสียมักจะเสียไปด้วย ดังนั้นแฟ้มข้อมูลที่ใช้งานบล็อกนั้นต้องถูกซ่อมแซม (เช่นการกู้คืนจากเทปสำรอง) โดยต้องการให้คนเป็นผู้แก้ไข

การจัดการการสับเปลี่ยน

หน่วยความจำเสมือนใช้พื้นที่ในดิสก์เสมือนส่วนขยายของหน่วยความจำหลัก เพราะว่าการเข้าถึงดิสก์ทำได้ช้ากว่าการเข้าถึงหน่วยความจำ ดังนั้นการใช้พื้นที่ที่ใช้ในการสับเปลี่ยน (Swap Space) จะมีผลกระทบมากต่อสมรรถนะของระบบ เป้าหมายหลักสำหรับการออกแบบและการนำพื้นที่ที่ใช้ในการสับเปลี่ยนไปใช้ คือเตรียมอัตรางานที่ได้ต่อหนึ่งหน่วยเวลาได้ดีที่สุดสำหรับระบบหน่วยความจำเสมือน

การใช้พื้นที่ผู้ใช้ที่ใช้ในการสับเปลี่ยน (Swap – Space Use)

พื้นที่ที่ใช้ในการสับเปลี่ยนถูกใช้ได้หลายวิธีโดยหลายระบบปฏิบัติการที่ต่างกัน ขึ้นอยู่กับการใช้อัลกอริทึมในการจัดการหน่วยความจำ ตัวอย่างเช่น ระบบที่ใช้การสับเปลี่ยน (Swapping) อาจใช้พื้นที่ที่ใช้ในการสับเปลี่ยนเพื่อเก็บภาพของกระบวนการทั้งหมดไว้ รวมทั้งตอน (Segment) ของรหัสโปรแกรมและข้อมูลด้วยระบบจะแบ่งเป็นหน้า (Paging System) อาจจะง่ายในการเก็บหน้าซึ่งถูกผลักออกจากหน่วยความจำหลัก จำนวนของพื้นที่ที่ใช้ในการสับเปลี่ยนที่ต้องการของระบบ ขึ้นอยู่กับจำนวนของหน่วยความจำทางกายภาพ

ตำแหน่งของพื้นที่ที่ใช้ในการสับเปลี่ยน (Swap – Space Location)

มีพื้นที่ที่ใช้ในการสับเปลี่ยนอยู่ 2 พื้นที่ คือพื้นที่ที่ใช้ในการสับเปลี่ยนสามารถถูกตัดออกจากระบบแฟ้มข้อมูลปกติ หรืออยู่ในส่วนของดิสก์ที่แยกออกมา (Separate Disk Partition) ถ้าการสับเปลี่ยนพื้นที่ทำได้ง่ายกับแฟ้มข้อมูลขนาดใหญ่ภายในระบบแฟ้มข้อมูล โปรแกรมย่อยระบบแฟ้มข้อมูลปกติ (Normal File System Routines) ควรจะถูกใช้เพื่อสร้าง เพื่อระบุชื่อ (ตั้งชื่อ) และ

พื้นที่เพื่อจัดสรรให้กับแฟ้มนั้น ๆ วิธีนี้นำไปใช้จริงได้ง่าย แต่ไม่มีประสิทธิภาพ การสำรวจโครงสร้างของไดเรกทอรี และโครงสร้างของข้อมูลการนจัดสรรดิสก์ต้องใช้เวลา การสูญเสียพื้นที่ย่อยภายนอกทำให้เพิ่มเวลาในการสับเปลี่ยนขึ้นมาก เพราะต้องค้นหาหลายครั้งในช่วงของการอ่านหรือเขียนภาพของกระบวนการ

อีกวิธีหนึ่ง คือใช่ส่วนของดิสก์ที่แยกออกมา เพื่อสร้างพื้นที่ซึ่งใช้ในการสับเปลี่ยน จึงไม่มีระบบแฟ้มข้อมูลหรือ โครงสร้างของไดเรกทอรีบนพื้นที่นี้ ตัวจัดการหน่วยเก็บพื้นที่ซึ่งจะใช้ในการสับเปลี่ยนจะถูกใช้ในการจัดสรรหรือยึดบล็อกคืนมาสู่ระบบ ตัวจัดการนี้ใช้วิธีที่ดีที่สุดสำหรับความเร็ว เพื่อให้ได้ประสิทธิภาพการสูญเสียพื้นที่ย่อยของภายในอาจจะเพิ่มขึ้น แต่ก็พอรับได้ เพราะข้อมูลในพื้นที่ซึ่งใช้ในการสับเปลี่ยนมักจะอยู่ในช่วงสั้น ๆ กว่าแฟ้มข้อมูลในระบบแฟ้มข้อมูล และพื้นที่ซึ่งใช้ในการสับเปลี่ยนมันจะถูกเข้าถึงอยู่บ่อย ๆ

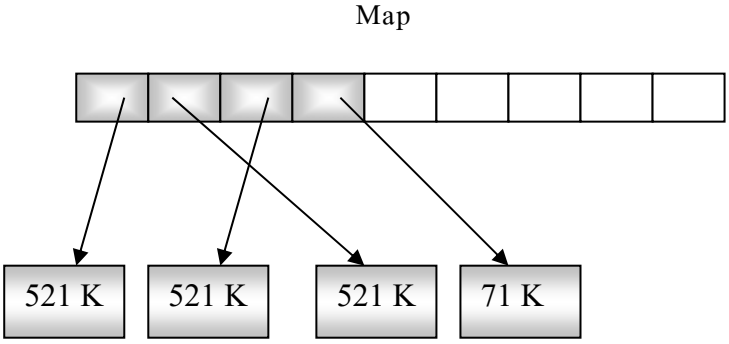
ข้อเสียของวิธีนี้ คือสร้างพื้นที่ซึ่งใช้ในการสับเปลี่ยนอย่างคงที่เอาไว้มากในระหว่างการแบ่งส่วนของดิสก์ การเพิ่มพื้นที่ซึ่งใช้ในการสับเปลี่ยนให้มากขึ้นทำได้โดยการแบ่งส่วนของดิสก์ใหม่อีกครั้ง ซึ่งเกี่ยวกับการย้ายหรือการทำลาย และการกู้คืนส่วนของระบบแฟ้มข้อมูลอื่นจากแบ็กอัป หรือโดยการเพิ่มพื้นที่ซึ่งใช้ในการสับเปลี่ยนจากที่ไหนสักแห่ง

การจัดการพื้นที่ที่ใช้ในการสับเปลี่ยน (Swap – Space Management)

เริ่มต้นพื้นที่ใช้ในการสับเปลี่ยนถูกจัดสรรให้แก่กระบวนการ เมื่อกระบวนการเริ่มต้นพื้นที่จะถูกจัดสรรให้เพื่อเก็บโปรแกรมอย่างเพียงพอ โดยเก็บหน้าของข้อความ (Text Page) หรือตอนของข้อความ (Text Segment) และตอนของข้อมูล (Data Segment) ของกระบวนการ ก่อนการจัดสรรพื้นที่ต้องการทั้งหมดด้วยวิธีนี้เป็นการป้องกันกระบวนการจากการทำงานนอกพื้นที่ที่ใช้ในการสับเปลี่ยนในขณะที่กำลังทำงาน (Execute) เมื่อกระบวนการเริ่มต้นข้อมูล (Text) จะถูกแบ่งเป็นหน้าจากระบบแฟ้มข้อมูล หน้าเหล่านี้จะถูกเขียนเมื่อจำเป็นและถูกอ่านย้อนหลังจากตรงนั้น ดังนั้นแฟ้มข้อมูลจะถูกพิจารณาเพียงครั้งเดียว สำหรับในแต่ละหน้าของข้อความ หน้าจากตอนของข้อมูลจะถูกอ่านจากระบบแฟ้มข้อมูล หรือถูกสร้างขึ้น (ถ้ายังไม่มี) และถูกเขียนที่พื้นที่ซึ่งใช้ในการสับเปลี่ยน และย้อนกลับไปยังหน้าที่ต้องการ วิธีที่ดีที่สุดวิธีหนึ่ง เช่น เมื่อผู้ใช้ 2 คน ทำงานเอดิเตอร์ (Editor) ตัวเดียวกัน คือกระบวนการในหน้าของข้อความที่เหมือนกันควรร่วมกันใช้หน้าเหล่านี้ ทั้งในหน่วยความจำทางตรรกะและในพื้นที่ที่ใช้ในการสับเปลี่ยนแก่น (Kernel) จะใช้แผนที่ในการสับเปลี่ยน (Swap Map) เพื่อตามรอยพื้นที่ซึ่งใช้ในการสับเปลี่ยน

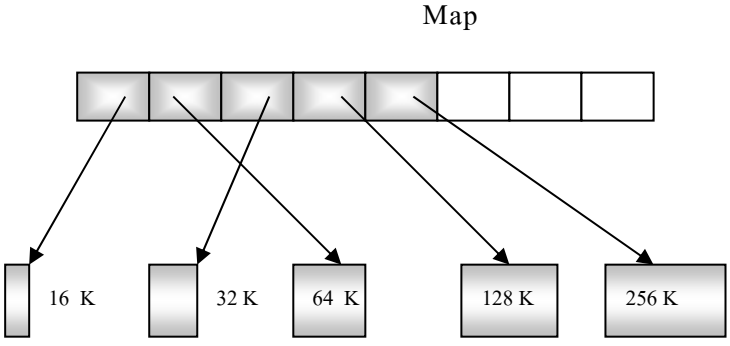
ตอนของข้อความจะมีขนาดคงที่ ดังนั้นพื้นที่ซึ่งใช้ในการสับเปลี่ยนจะถูกจัดสรรให้ก้อนละ 521 K ยกเว้นก้อนสุดท้าย จะเก็บส่วนที่เหลือ(อาจไม่ถึง 521 K) ดังรูปที่ 9.8

แผนที่การสับเปลี่ยนตอนของข้อมูลค่อยข้างซับซ้อน เพราะตอนของข้อมูลสามารถเพิ่มขึ้นได้ตลอดเวลา แผนที่ดังกล่าวมีขนาดคงที่ แต่เก็บตำแหน่งที่ใช้ในการสับเปลี่ยนที่เป็นบล็อกที่มีขนาดไม่คงที่ให้ดัชนี i คือ บล็อกที่แผนที่ที่ใช้ในการสับเปลี่ยนชี้อยู่ที่ขนาด 2 * 16 K โดยมากที่สุดได้ 2 เมกะไบต์



รูปที่ 9.8 การจับคู่ของการสับเปลี่ยนตอนของข้อความ

ขนาดของบล็อกที่เล็กที่สุดและใหญ่ที่สุดไม่คงที่ และสามารถเปลี่ยนแปลงได้โดยการบูตเครื่องใหม่ เมื่อกระบวนการพยายามเพิ่มตอนของข้อมูลก่อนบล็อกสุดท้ายในพื้นที่ซึ่งใช้ในการสับเปลี่ยนระบบปฏิบัติการจะจัดบล็อกให้อีกบล็อกหนึ่ง ที่มีขนาดเป็น 2 เท่าของบล็อกก่อนหน้านี้ ด้วยผลลัพธ์ของวิธีการนี้ การบวนการเล็กก็จะใช้บล็อกเล็กตามไปด้วย นั้นเท่ากับเป็นการลดการเสียพื้นที่ บล็อกของกระบวนการขนาดใหญ่ก็จะหาพบได้รวดเร็ว ดังนั้นการจับคู่ของเนื้อที่ ที่ใช้ในการสับเปลี่ยนจึงมีขนาดเล็กตามไปด้วย ดังรูปที่ 9.9



รูปที่ 9.9 การจับคู่ของการสับเปลี่ยนตอนตามขนาดของข้อความ

ความน่าเชื่อถือของดิสก์

ดิสก์เคยเป็นองค์ประกอบที่น่าเชื่อถือน้อยที่สุดในระบบคอมพิวเตอร์ เพราะดิสก์ยังคงมีอัตราความผิดพลาดสูง และความผิดพลาดเหล่านั้นทำให้ข้อมูลสูญหายและเสียเวลา การกู้คืนอาจต้องใช้เวลาหลายชั่วโมง

การปรับปรุงเทคนิคในการใช้ดิสก์หลาย ๆ ทางเกี่ยวข้องกับการใช้ดิสก์หลายตัวทำงานประสานกันเพื่อเป็นการปรับปรุงในด้านความเร็ว การแกะดิสก์ (Disk Striping) เราใช้กลุ่มของดิสก์เป็นเหมือนหน่วยเก็บข้อมูล 1 หน่วย บล็อกของข้อมูลแต่ละบล็อกถูกแบ่งเป็นบล็อกย่อยหลาย ๆ บล็อกซึ่งบล็อกย่อยจะถูกเก็บในแต่ละดิสก์ เวลาที่ต้องการเพื่อใช้ในการย้ายบล็อกไปสู่หน่วยความจำจะลดลงอย่างเหลือเชื่อ เพราะดิสก์ทุกตัวโอนย้ายบล็อกย่อยของพวกมันแบบขนาน ถ้าดิสก์มีการหมุนพร้อมกัน (Synchronized) สรรถนะจะดีขึ้นเพราะดิสก์ทั้งหมดจะพร้อมที่จะโอนย้าย บล็อกย่อยของมันในเวลาเดียวกัน

การจัดการนี้มักเรียกว่า อาร์เรย์ที่เหลือเฟือของดดิสก์อิสระ (Redundant Array of Independent Disk : RAID) วิธี RAID เป็นการเพิ่มความน่าเชื่อถือของระบบหน่วยเก็บข้อมูลโดยการเก็บข้อมูลที่เหลือเฟือ

การจัดการ RAID ที่ง่ายที่สุด เรียกว่า การสำรองข้อมูล (Mirroring หรือ Shadowing) โดยเก็บสำเนาของแต่ละดิสก์ วิธีนี้คุ้มค่าเพราะดิสก์หลายตัวถูกใช้เก็บข้อมูลเดียวกัน 2 ครั้ง แต่ในทางกลับกันมันจะเร็วเป็น 2 เท่าเมื่อเป็นการอ่าน เพราะครึ่งหนึ่งของการร้องขอที่จะอ่านจะถูกส่งไปยังแต่ละดิสก์

การจัดการ RAID เรียกว่า การตรวจสอบบล็อกขาออก (Block Interleaved Parity) โดยใช้พื้นที่ของดิสก์เพียงเล็กน้อยในการเก็บบล็อกตรวจสอบ (Rarity Block) ตัวอย่างเช่น สมมติว่ามีดิสก์ 9 ตัว ในอาร์เรย์โดยบล็อกของข้อมูลทุก ๆ 8 ตัว ที่ถูกเก็บในอาร์เรย์จะมีอยู่ 1 บล็อกที่เป็นบล็อกตรวจสอบที่ถูกเก็บไปด้วยตำแหน่งของบิตแต่ละบิตในบล็อกตรวจสอบนี้ ควรจะเก็บค่าสำหรับตำแหน่งของบิตที่ตรงกันในแต่ละบล็อกของข้อมูลทั้ง 8 เหมือนกับการคำนวณบิตตรวจสอบที่ 9 ในหน่วยความจำหลักสำหรับแต่ละ 8 บิต - ไบต์ ถ้าบล็อกของดิสก์หนึ่งเสีย บิตของข้อมูลทั้งหมดต้องถูกลบ แต่ยังสามารถคำนวณใหม่จากบล็อกของข้อมูลบล็อกอื่นบวกกับบล็อกตรวจสอบ ดังนั้นดิสก์เสียตัวเดียวจึงไม่ทำให้ข้อมูลเสียหาย

การใช้งานหน่วยเก็บข้อมูลชนิดถาวร

โดยนิยามแล้วข้อมูลที่อยู่ในหน่วยเก็บข้อมูลชนิดคงที่จะไม่มีทางสูญหายไปไหนการนำหน่วยเก็บข้อมูลไปใช้ เราจำเป็นต้องคัดลอกข้อมูลที่ต้องการไปไว้ยังอุปกรณ์ที่ใช้เก็บข้อมูลหลาย ๆ

ข้อมูลที่อาจเก็บในดิสก์หรือในเทป ด้วยโหนดที่เป็นอิสระจากความล้มเหลว เราต้องการวิธีที่จะทำให้ข้อมูลทันสมัยที่รับประกันได้ว่า ความผิดพลาดในระหว่างการทำข้อมูลให้ทันสมัยจะไม่ทำให้ข้อมูลเสียหาย และเมื่อเราทำการกู้คืนจากที่ผิดพลาด เราต้องสามารถทำให้ข้อมูลทั้งหมดมีค่าที่ถูกต้อง และถ้าเกิดมีการล้มเหลวอีกในระหว่างการกู้คืนเราจะต้องดำเนินการอย่างไร

ดิสก์จะทำงานจนได้ผลลัพธ์หนึ่งในสามข้อนี้คือ

1. สำเร็จอย่างสมบูรณ์แบบ (Successful Completion) ข้อมูลถูกเขียนลงดิสก์ได้อย่างถูกต้อง

2. ล้มเหลวบางส่วน (Partial Failure) ความล้มเหลวเกิดขึ้นระหว่างการโอนย้ายข้อมูล ดังนั้นเซกเตอร์บางส่วนจะถูกเขียนด้วยข้อมูลใหม่ และเซกเตอร์ที่ถูกเขียนในระหว่างการล้มเหลวอาจจะผิดพลาด

3. ล้มเหลวทั้งหมด (Total Failure) ความล้มเหลวเกิดขึ้นก่อนที่ดิสก์จะเริ่มเขียน ดังนั้นค่าของข้อมูลบนดิสก์ก็จะเหมือนเดิมก่อนที่จะเกิดการเขียนใด ๆ

ถ้าความล้มเหลวเกิดในระหว่างการเขียนบล็อก เมื่อระบบตรวจพบแล้วจะทำการกู้บล็อกนั้นคืนระบบต้องเก็บบล็อกทางกายภาพ 2 บล็อกไว้ให้บล็อกทางตรรกะแต่ละบล็อก ผลลัพธ์ที่ได้มีดังนี้

1. เขียนข้อมูลลงบนบล็อกทางกายภาพบล็อกแรก

2. เมื่อเขียนบล็อกแรกเรียบร้อย ก็เขียนข้อมูลเดียวกันลงยังบล็อกทางกายภาพอีกบล็อก

3. ประกาศว่า การทำงานเสร็จสิ้นหลังจากการเขียนบล็อกที่สองสำเร็จเท่านั้น ถ้าเขียนไม่สำเร็จจะไม่ประกาศ

ในระหว่างการกู้ระบบจากความล้มเหลว บล็อกทางกายภาพแต่ละคู่จะถูกทดสอบ ถ้าทั้ง 2 บล็อกเหมือนกันและไม่มีการพบข้อมูลผิดพลาด ก็ไม่ต้องทำอะไร ถ้ามีบล็อกหนึ่งพบว่ามีข้อผิดพลาดเราจะแทนที่เนื้อหาในบล็อกนั้นด้วยข้อมูลของอีกบล็อกหนึ่ง ถ้าทั้งสองบล็อกไม่พบข้อผิดพลาดแต่เนื้อหาต่างกัน เราจะเอาเนื้อหาของบล็อกทั้งสองเข้าไปเก็บแทนที่ในบล็อกที่หนึ่ง กระบวนการในการกู้คืนนี้ประกันได้ว่า การเขียนหน่วยเก็บข้อมูลชนิดคงที่ต้องเสร็จสมบูรณ์หรือไม่ผลลัพธ์ที่ได้ต้องไม่เปลี่ยนแปลง

ปฏิบัติการที่ 9

1. นักศึกษาแต่ละกลุ่มเรียนรู้คำสั่งเกี่ยวกับการจัดการดิสค์ ของระบบปฏิบัติการวินโดว์ (ใน System Tools)
2. นักศึกษาแต่ละกลุ่มแรกเปลี่ยนเรียนรู้คำสั่งต่าง ๆ เหล่านั้น
3. ลองใช้คำสั่งนั้น ๆ ดำเนินการจัดการดิสค์ สังเกตและบันทึกผลที่เกิดขึ้น และนำมาแลกเปลี่ยนเรียนรู้กัน
4. ลองใช้ระบบปฏิบัติการอื่น ๆ ในการจัดการสื่อจัดเก็บที่เกี่ยวข้อง

คำถามท้ายบท

1. จงอธิบายโครงสร้างของดิสก์
2. จงอธิบายการวิธีการเข้าถึงข้อมูลในดิสก์
3. เมื่อกระบวนการต้องการการรับหรือส่งข้อมูลกับดิสก์ คำสั่งเรียกระบบของระบบปฏิบัติการจะต้องรู้ข้อมูลที่เกี่ยวข้องอะไรบ้าง
4. การเลือกใช้อัลกอริทึมการจัดตารางเวลาของดิสก์สำหรับข้อมูลที่มีการจัดเก็บกระจายอยู่ทั่วดิสก์และมีการเรียกใช้บ่อย ๆ ควรเลือกใช้อัลกอริทึมใดเพราะเหตุใด
5. จงอธิบายการที่ระบบปฏิบัติการจัดการรูปแบบของดิสก์ (Disk Formatting)
6. จงอธิบายการที่ระบบปฏิบัติการจัดการบูตบล็อก (Boot Block) และบล็อกเสีย (Bad Block) ของดิสก์
7. จงอธิบายการจัดการความน่าเชื่อถือของดิสก์ (Disk Reliability)
8. ถ้าการกู้คืนข้อมูลเกิดความล้มเหลวในระหว่างการกู้คืน จะต้องดำเนินการอย่างไร
9. จงอธิบายวิธีการจัดการพื้นที่ที่ใช้ในการสับเปลี่ยน (Swap – Space Management)
10. จงแสดงความคิดเห็นว่าอัลกอริทึมการจัดตารางเวลาของดิสก์วิธีไหนเป็นวิธีที่ดีที่สุดเพราะอะไร