เอกสารอ้างอิง

1. อาจิน จิรชีพพัฒนา, "การพัฒนาโปรแกรมสำเร็จรูปสำหรับสร้างภาพให้สอดคล้องกับคำบรรยายจากเทปบันทึกเสียง," วิทยานิพนธ์ปริญญามหาบัณฑิต, ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2530.
2. Munro, Allen, Mac Power Using Macintosh Software, pp. 63-79, Scott, Foresman and Company Glenview, Illinois London, 1985.
3. Innovative Data Design, Inc., "Mac Draft User's Manual," Innovative Data Design, Inc., California, 1985.
4. Apple Computer, Inc., "Inside Macintosh," Apple Computer, Inc., California, 1985.
5. Audus Bob, "Mac Draw PICT Files in Basic," Mac Tutor, Vol.3, No.10, pp. 66-81, California, 1987.
6. Palmer Gray, "Reading Mac Paint Files," Mac Tutor, Vol.3, No.5, pp. 56-61, California, 1987.
7. Jernigan Ginger, Rick Blair, "Quick Draw's Internal Picture Definition," Macintosh Technical Notes, No. 21, pp. 1-6, California, 1986.
8. Jernigan Ginger, "Mac Draw's PICT File Format and Comments," Macintosh Technical Notes, No. 27, pp. 1-7, California, 1986.

ภาคผนวก

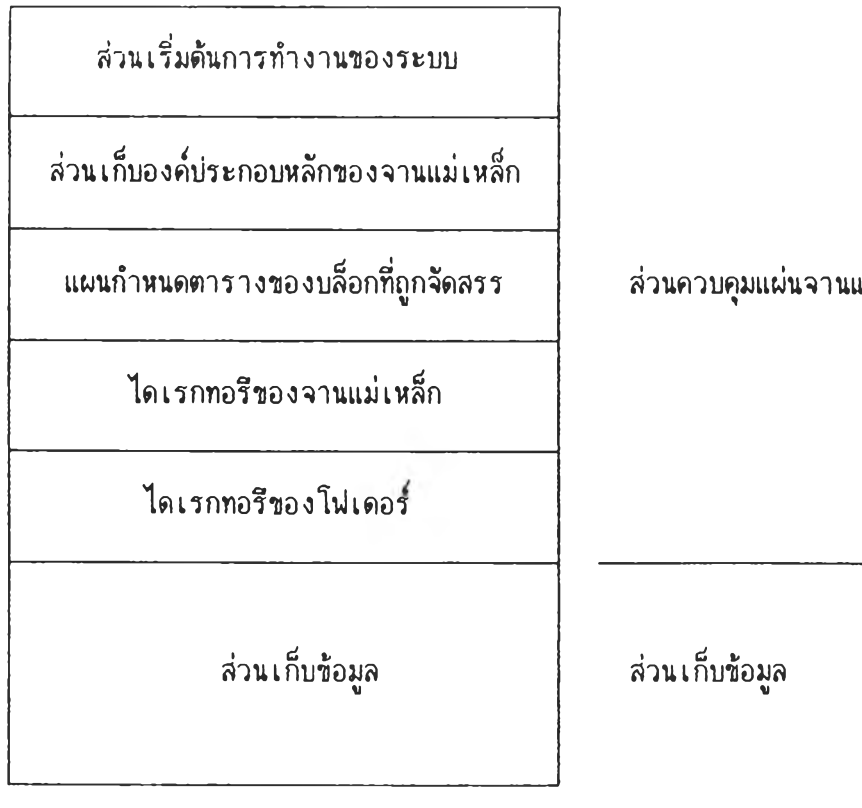
ภาคผนวก ก.

การจัดแฟ้มข้อมูลแบบลำดับชั้น

ภาคผนวก ก.

การจัดแฟ้มข้อมูลแบบลำดับชั้น

การจัดแฟ้มข้อมูลแบบเป็นลำดับชั้น (Hirearchy) สามารถทำได้โดยใช้โฟเดอร์ ซึ่งสร้างโดยใช้คำสั่ง New Folder จากรายการแฟ้มข้อมูล แล้วจากนั้นสามารถเลือกแฟ้มข้อมูลต่าง ๆ ให้อยู่ภายในโฟเดอร์ได้เช่นเดียวกับไดเรกทอรีแฟ้มข้อมูล เมื่อสร้างโฟเดอร์ 1 โฟเดอร์ โปรแกรมแมคดราฟท์จะสร้างไดเรกทอรีของโฟเดอร์นั้น ภายในส่วนของไดเรกทอรีจานแม่เหล็ก ดังนั้นโครงสร้างข้อมูลของจานแม่เหล็ก สามารถแสดงได้ดังนี้

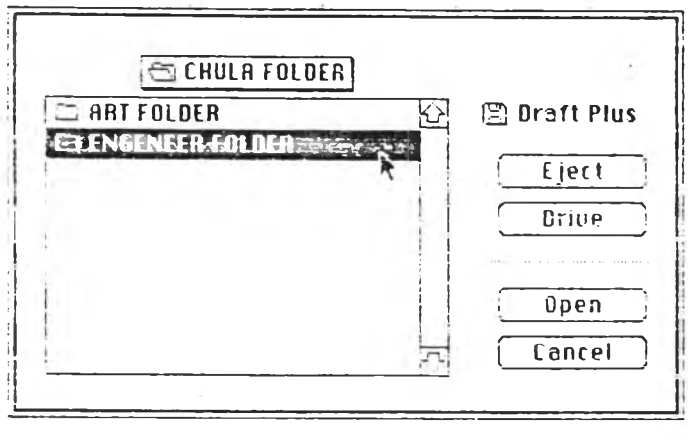


รูปที่ ก.1 แสดงโครงสร้างข้อมูลของแผ่นจานแม่เหล็ก

จากรูปที่ ก.1 จะเห็นว่าข้อมูลส่วนอื่น ๆ ของแผ่นจานแม่เหล็กยังเหมือนเดิม ส่วนที่เพิ่มขึ้นคือ ไดเรกทอรีของโฟเดอร์ ซึ่งข้อมูลประกอบด้วย

- ไบต์ที่เป็นตัวชี้ว่าเป็นไดเรกทอรีของโฟเดอร์
- หมายเลขอ้างอิงของโฟเดอร์
- จำนวนแฟ้มข้อมูลที่บรรจุอยู่ในโฟเดอร์นั้น
- หมายเลขของแฟ้มข้อมูลที่อยู่ในโฟเดอร์
- โฟเดอร์ที่เก็บโฟเดอร์นี้อยู่ (สำหรับโฟเดอร์ที่มีหลายลำดับชั้น)
- วันและเวลาที่สร้างโฟเดอร์นี้
- วันและเวลาที่ทำการปรับปรุงโฟเดอร์นี้ครั้งสุดท้าย
- ความยาวของชื่อโฟเดอร์
- ชื่อโฟเดอร์ (จำนวนมากที่สุด 31 ตัวอักษร)
- ชื่อโฟเดอร์ที่อยู่ภายใต้โปรแกรมประยุกต์

เมื่อสร้างแฟ้มข้อมูลอยู่ในโฟเดอร์แล้ว เวลาเรียกคำสั่งเปิดแฟ้มข้อมูล จะปรากฏกรอบสนทนาแสดงรายชื่อโฟเดอร์ภายใต้โปรแกรมแมคดราฟท์ จากนั้นจะสามารถเลือกโฟเดอร์เพื่อแสดงรายชื่อแฟ้มข้อมูลที่อยู่ภายในโฟเดอร์นั้น ทำให้ผู้ใช้สามารถแยกแฟ้มข้อมูลเป็นหมวดหมู่ได้ นอกจากนี้ชื่อแฟ้มข้อมูลต่างโฟเดอร์กัน สามารถตั้งชื่อเหมือนกันได้



รูปที่ ก.2 แสดงกรอบสนทนาการเปิดแฟ้มข้อมูลแบบลำดับชั้น

ลักษณะความสัมพันธ์ของข้อมูลที่เก็บอยู่ในแผ่นจานแม่เหล็ก มีดังนี้

	หมายเลขโฟเดอร์แม่	หมายเลขแฟ้มข้อมูล	บล็อกเริ่มต้น		ชื่อแฟ้มข้อมูล
	23	5	3		BLD1
	23	17	8		BLD2
	45	11	20		BLD4
	23	7	5		BLD3
	45	3	16		BLD5

ไดเรกทอรีจานแม่เหล็ก

หมายเลขอ้างอิง	หมายเลขโฟเดอร์แม่	จำนวนแฟ้มข้อมูล	หมายเลขแฟ้มข้อมูล				ชื่อโฟเดอร์
10			23	45			CHULA
23	10	3	5	17	7		ART
45	10	2	11	3			ENGINEER

ไดเรกทอรีโฟเดอร์

จำนวนโฟเดอร์	ชื่อโปรแกรมประยุกต์	ชื่อโฟเดอร์	
3	MacDraft	CHULA FOLDER	ART FOLDER
ENGINEER FOLDER			

รูปที่ ก.3 แสดงตัวอย่างความสัมพันธ์ของแฟ้มข้อมูลแบบลำดับชั้น

จากข้อมูลไดเรกทอรีของโฟเดอร์ จะยกตัวอย่างเพื่อแสดงความสัมพันธ์ของข้อมูลในแผ่นจานแม่เหล็กดังนี้

ตัวอย่างจากรูปที่ ก.3 แสดงโฟเดอร์ภายใต้โปรแกรมแมกดราฟต์ ซึ่งมี 3 โฟเดอร์ คือ CHULA โฟเดอร์, ARTS โฟเดอร์ และ ENGINEER โฟเดอร์ ซึ่ง ARTS และ ENGINEER อยู่ใน CHULA โฟเดอร์ และใน ARTS โฟเดอร์ จะมีแฟ้มข้อมูล BLD1, BLD2 และ BLD3 ส่วน ENGINEER โฟเดอร์ ประกอบด้วยแฟ้มข้อมูล BLD3, BLD4

การจัดการเกี่ยวกับโฟเดอร์ มีลักษณะเช่นเดียวกับแฟ้มข้อมูล คือจะนำข้อมูลของโฟเดอร์เก็บไว้ในหน่วยความจำ แล้วจึงนำข้อมูลในหน่วยความจำของโฟเดอร์ไปค้นหาแฟ้มข้อมูลอีกที ซึ่งการจัดการโฟเดอร์ที่แตกต่างไปจากแฟ้มข้อมูล คือส่วนที่เชื่อมโยงกับแฟ้มข้อมูล คือการแสดงรายชื่อโฟเดอร์และแฟ้มข้อมูล

ส่วนการอ่านแฟ้มข้อมูล และการเขียนแฟ้มข้อมูล เป็นส่วนที่จัดการเกี่ยวกับแฟ้มข้อมูลโดยตรง จึงไม่มีการเปลี่ยนแปลง

1. <u>โครงสร้างการจัดแฟ้มข้อมูลแบบลำดับชั้น</u>

1.1 การแสดงรายชื่อแฟ้มข้อมูล

ขั้นตอนมีดังนี้

1.1.1 ขั้นตอนแรกต้องทำการ mount แผ่นจานแม่เหล็ก และจะนำข้อมูลจากส่วนควบคุมแผ่นจานแม่เหล็ก มาเก็บไว้ในหน่วยความจำ เช่นเดียวกับในรูปที่ 3.2

1.1.2 เมื่อได้ข้อมูลส่วนควบคุมแผ่นจานแม่เหล็กไว้ในหน่วยความจำแล้ว โปรแกรมแมกดราฟต์จะทำการอ่านไดเรกทอรีจานแม่เหล็ก โดยเริ่มหาจากโฟเดอร์ที่อยู่ภายใต้โปรแกรมแมกดราฟต์โดยหาโฟเดอร์ลำดับชั้นแรก (ไดเรกทอรีโฟเดอร์ไม่มีหมายเลขโฟเดอร์แม่) ซึ่งในตัวอย่างนี้คือ CHULA โฟเดอร์

1.1.3 ทำการย้ายข้อมูลของไดเรกทอรีโฟเดอร์จากจานแม่เหล็ก มาเก็บในหน่วยความจำ จากตัวอย่าง CHULA โฟเดอร์ จะมีหมายเลขอ้างอิงของโฟเดอร์คือ 10 แต่ไม่มีหมายเลขของโฟเดอร์แม่ ซึ่งแสดงว่าเป็นโฟเดอร์ลำดับแรกที่อยู่ภายใต้โปรแกรมแมกดราฟต์ (โฟเดอร์ที่อยู่ภายใต้โปรแกรมแมกดราฟต์ ดูได้จากส่วนท้ายสุดของไดเรกทอรีโฟเดอร์)

ถ้าต้องการแสดงรายชื่อของแฟ้มข้อมูลหรือโฟเดอร์ที่อยู่ภายใต้ CHULA โฟเดอร์ จะหาจากหมายเลขแฟ้มข้อมูล หรือโฟเดอร์ลูก โดยใช้หมายเลขอ้างอิงของโฟเดอร์แม่ ตัวอย่างนี้ CHULA โฟเดอร์มีหมายเลขอ้างอิง 10 จะหาในแต่ละไดเรกทอรีที่มีหมายเลขโฟเดอร์แม่เป็น 10

ซึ่งลำดับชั้นของการอ่านจะทำเป็นชั้น ๆ ไป โดยจะใช้หมายเลขอ้างอิงของแต่ละโฟเดอร์ และหมายเลขของโฟเดอร์แม่เป็นตัวกำหนด จากนั้นจะใช้หมายเลขของโฟเดอร์ลูก หรือสมาชิกในโฟเดอร์นั้นเป็นตัวเรียกแฟ้มข้อมูลนั้น ๆ

แผ่นจานแม่เหล็ก

ไดเรกทอรีจานแม่เหล็ก

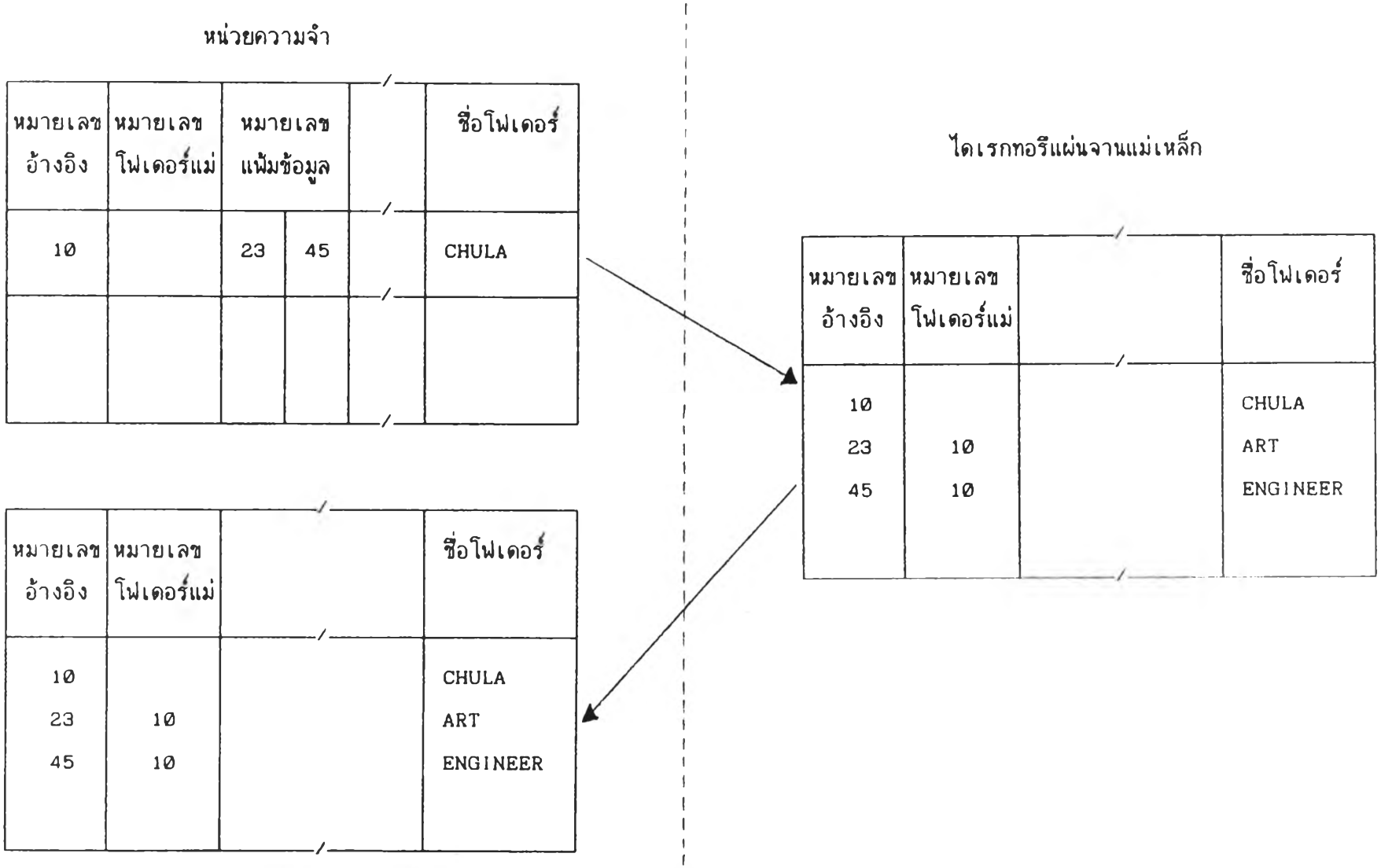
	หมายเลขโฟเดอร์แม่	หมายเลขเริ่มต้น	บล็อกเริ่มต้น		ชื่อแฟ้มข้อมูล
	23	5	3		BLD1
	23	17	8		BLD2
	45	11	20		BLD4
	23	7	5		BLD3
	45	3	16		BLD5

หมายเลขอ้างอิง	หมายเลขโฟเดอร์แม่	จำนวนแฟ้มข้อมูล	หมายเลขแฟ้มข้อมูล			ชื่อโฟเดอร์
10			23	45		CHULA
23	10	3	5	17	7	ART
45	10	2	11	3		ENGINEER

หน่วยความจำ

หมายเลขอ้างอิง	หมายเลขโฟเดอร์แม่	หมายเลขแฟ้มข้อมูล			
10		23	45		CHULA

รูปที่ ก.4 แสดงการเคลื่อนย้ายไดเรกทอรีโฟเดอร์ลงในหน่วยความจำ



รูปที่ ก.5 การแสดงรายชื่อโฟเดอร์แบบลำดับชั้น

1.2 การเปิดแฟ้มข้อมูล

การจะเปิดแฟ้มข้อมูลต่าง ๆ ได้นั้น ต้องแสดงรายชื่อแฟ้มข้อมูลขึ้นมาก่อน หรือกล่าวคือจะต้องแสดงรายชื่อจนถึงลำดับชั้นของแฟ้มข้อมูล (ปรากฏชื่อแฟ้มข้อมูลนั้น) เมื่อแสดงรายชื่อของแฟ้มข้อมูลได้แล้ว ภายในส่วนควบคุมแฟ้มข้อมูลในหน่วยความจำ จะเป็นดังรูปที่ ก.6

จากตัวอย่างในรูปที่ ก.6 เมื่อทำการแสดงรายชื่อแฟ้มข้อมูลภายใน ART โฟเดอร์แล้ว ถ้าต้องการเปิดแฟ้มข้อมูลชื่อ BLD2 ซึ่งมีหมายเลขแฟ้มข้อมูลเป็น 17 แล้วจะนำหมายเลขแฟ้มข้อมูล ไปทำการหาจากไดเรกทอรีแฟ้มข้อมูลที่มีหมายเลขตรงกัน แล้วนำข้อมูลบางส่วนที่จำเป็นเก็บไว้ในส่วนควบคุมการ access แฟ้มข้อมูลในหน่วยความจำ ซึ่งในส่วนนี้จะเหมือนกับการจัดการแฟ้มข้อมูลที่ไม่มีลำดับชั้น

ส่วนควบคุมการ access แฟ้มข้อมูลในหน่วยความจำ จะเก็บข้อมูล ที่ใช้ในการอ่านแฟ้มข้อมูล ซึ่งประกอบด้วยหมายเลขของแฟ้มข้อมูล, ขนาดเชิงตรรก, ขนาดเชิงกายภาพ และบล็อกเริ่มต้นของแฟ้มข้อมูล

หน่วยความจำ

ส่วนควบคุมแฟ้มข้อมูล

	หมายเลขโฟเดอร์	หมายเลขแฟ้มข้อมูล		ชื่อแฟ้มข้อมูล
	23	5		BLD1
	23	17		BLD2
	23	7		BLD3

จานแม่เหล็ก

ไดเรกทอรีแฟ้มข้อมูล

	หมายเลขโฟเดอร์	หมายเลขแฟ้มข้อมูล		ชื่อแฟ้มข้อมูล
	23	5		BLD1
	23	17		BLD2
	45	11		BLD4
	23	7		BLD3
	45	3		BLD5

ส่วนควบคุมการ access แฟ้มข้อมูล

	หมายเลขแฟ้มข้อมูล	บล็อกเริ่มต้นแฟ้มข้อมูล	ขนาดเชิงตรรก	ขนาดเชิงกายภาพ	
เอ็นทรี 1	17	8	1500	2048	

รูปที่ ก.6 แสดงการเปิดแฟ้มข้อมูล

1.3 การอ่านแฟ้มข้อมูล

การอ่านแฟ้มข้อมูล จะใช้ข้อมูลในส่วนควบคุมการ access แฟ้มข้อมูลในหน่วยความจำ ซึ่งการอ่านแฟ้มข้อมูลนี้ เป็นการเข้าถึงในระดับแฟ้มข้อมูล ซึ่งไม่มีส่วนเปลี่ยนแปลง ขั้นตอนในการอ่านจะยังคงเหมือนเดิมคือ

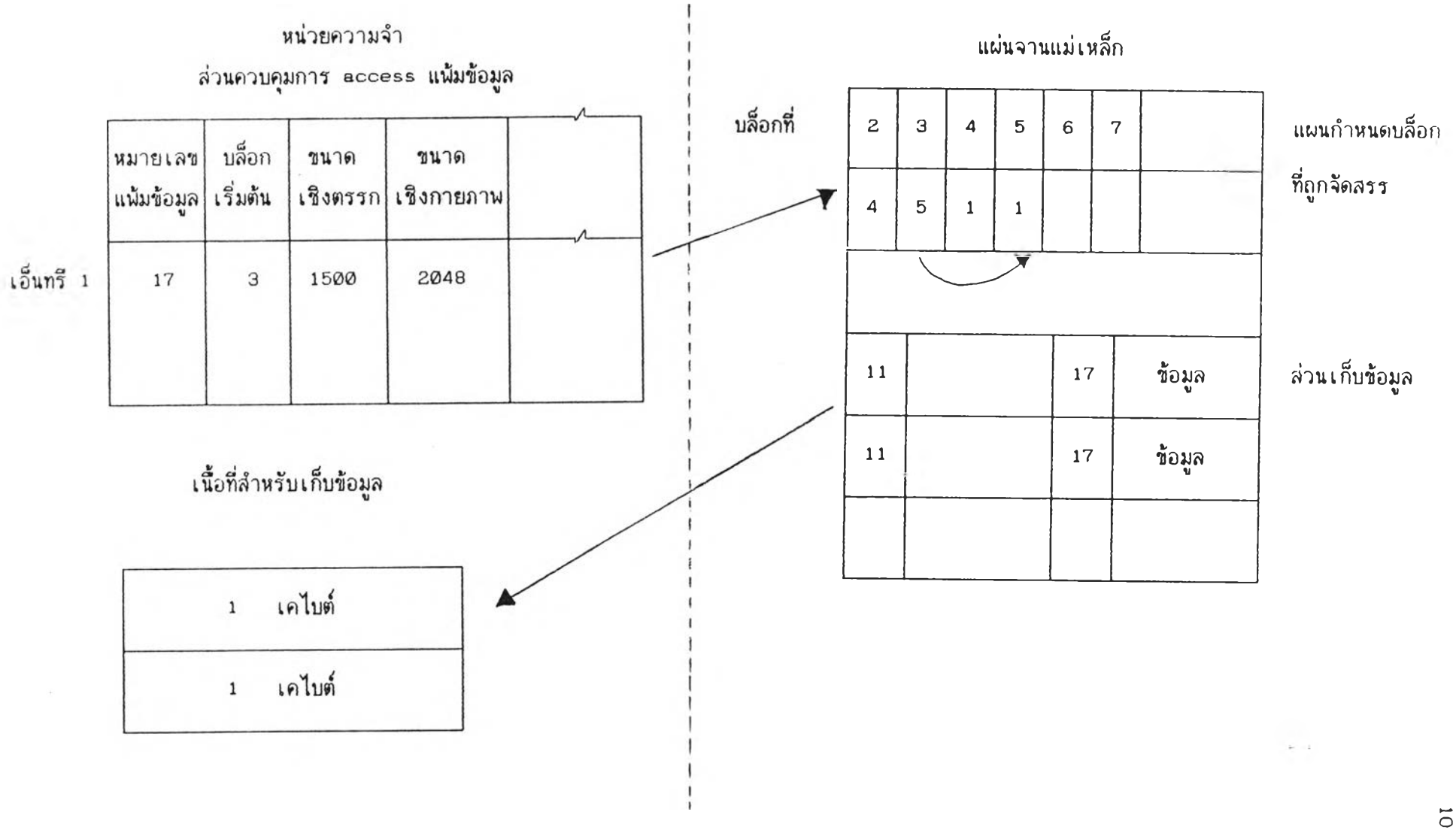
1.3.1 ใช้หมายเลขแฟ้มข้อมูลจากส่วนควบคุมการ access แฟ้มข้อมูลในหน่วยความจำในการเข้าถึงแฟ้มข้อมูลนั้น

1.3.2 การเตรียมเนื้อที่ในหน่วยความจำ จะใช้ขนาดเชิงกายภาพของแฟ้มข้อมูล เพื่อเตรียมที่ที่จะใช้เก็บข้อมูลทั้งหมดของแฟ้มข้อมูล และใช้ขนาดเชิงตรรกในการหาจำนวนการอ่าน

1.3.3 บล็อกเริ่มต้นของข้อมูล เพื่อนำไปหาบล็อกต่อไป จนกระทั่งถึงบล็อกสุดท้าย

1.3.4 อ่านข้อมูลแต่ละบล็อก มาเก็บไว้ในหน่วยความจำ

จากตัวอย่างในรูปที่ ก.7 แฟ้มข้อมูลที่ต้องการมีหมายเลข 17 บล็อกเริ่มต้นคือ 3 บล็อกถัดไปคือ 5 มีขนาด 2 เคไบต์ ข้อมูลเก็บอยู่ในส่วนเก็บข้อมูล โดยมีหมายเลขแฟ้มข้อมูลกำกับอยู่หน้าข้อมูล



รูปที่ ก.7 ตัวอย่างแสดงการอ่านแฟ้มข้อมูล

1.4 การเขียนแฟ้มข้อมูล

การเขียนแฟ้มข้อมูลนี้เป็นการจัดการแฟ้มข้อมูลเป็นอีกส่วนที่ไม่มีการเปลี่ยนแปลง ซึ่งขั้นตอนเป็นดังนี้คือ

1.4.1 เมื่อทำการสร้างแฟ้มข้อมูลเป็นที่เรียบร้อย ข้อมูลทั้งหมดจะอยู่ในหน่วยความจำ ต้องคำนวนขนาดเชิงตรรกและเชิงกายภาพของแฟ้มข้อมูลนั้น เพื่อหาเนื้อที่ในแผ่นจานแม่เหล็ก โดยคำนวณว่าแฟ้มข้อมูลนั้นมีใช้เนื้อที่กี่บล็อก และบล็อกที่ว่างมีพอหรือไม่ เช่นต้องการเขียนข้อมูลขนาด 3 บล็อก (3 เคไบต์) แต่บล็อกที่ว่างมีเพียง 2 บล็อก ทำให้ไม่สามารพจัดเขียนแฟ้มข้อมูลนั้นลงแผ่นจานแม่เหล็กได้ ต้องทำการเขียนลงแผ่นจานแม่เหล็กอื่น

1.4.2 ในกรณีที่มีบล็อกว่างพอในจานแม่เหล็กแผ่นนั้น การจัดสรรบล็อกจะหาบล็อกที่ว่างแล้ว หาบล็อกต่อไปที่ว่าง เมื่อเจอแล้วจะ เลขของบล็อกถัดไปลงในบล็อกแรก และบล็อกสุดท้ายจะใส่ค่า 1

ตัวอย่างเช่น ต้องการเขียนแฟ้มข้อมูลมีขนาด 2 เคไบต์ และบล็อกที่ใช้ไปมีบล็อก 2, 3, 5, 6, 8 ดังนั้นจึงแสดงได้ดังนี้

บล็อกที่	2	3	4	5	6	7	8	9		สุดท้าย
	3	5	0	6	8	0	1	0		0

รูปที่ ก.8 แสดงบล็อกที่ถูกจัดสรรก่อนเขียนแฟ้มข้อมูล

แฟ้มข้อมูลที่ต้องการเขียน ใช้ 2 บล็อก ซึ่งเมื่อจัดสรรแล้วจะได้ดังนี้

บล็อกที่	2	3	4	5	6	7	8	9		สุดท้าย
	3	5	7	6	8	1	1	0		0

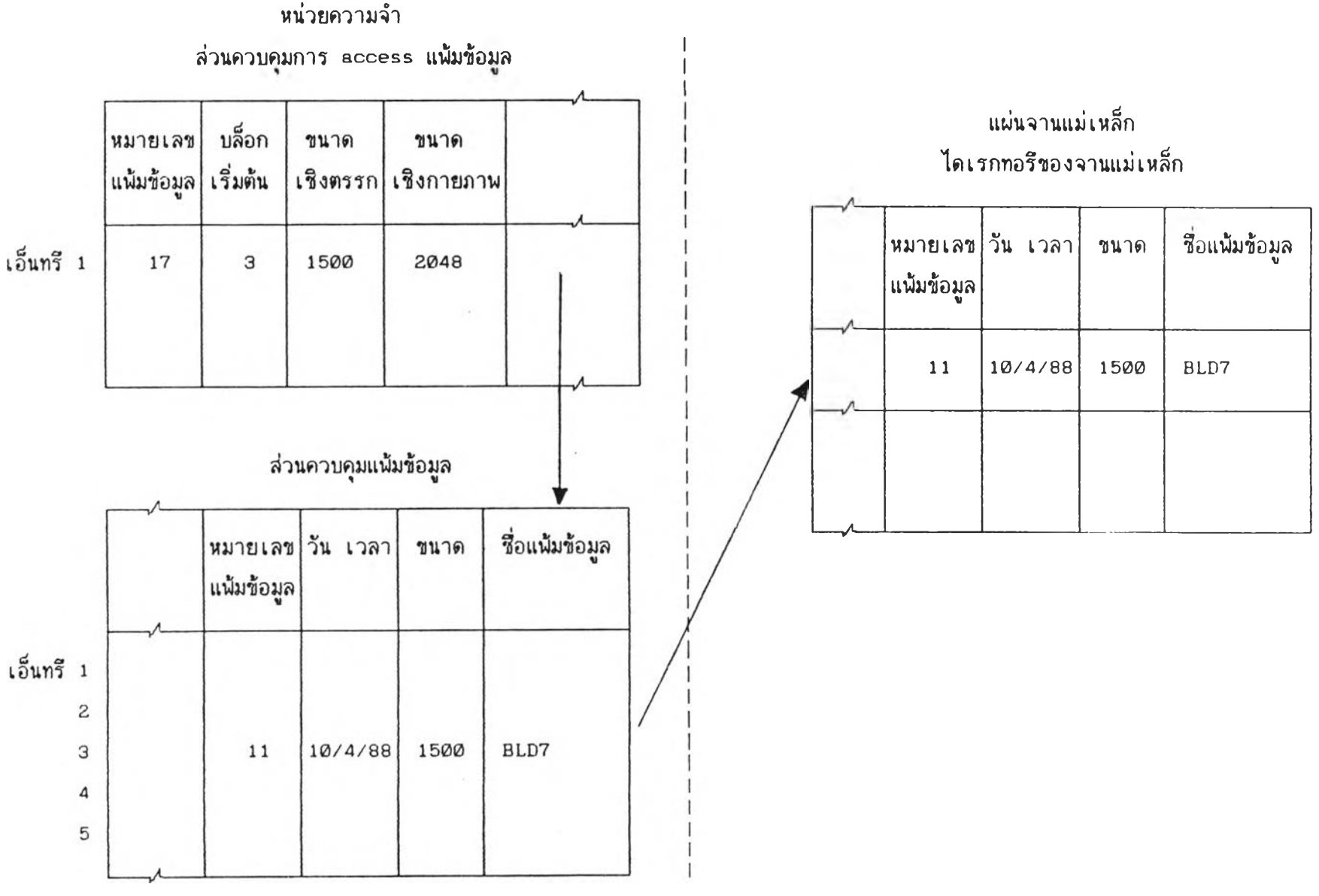
รูปที่ ก.9 แสดงบล็อกที่ถูกจัดสรรหลังเขียนแฟ้มข้อมูล

1.4.3 เมื่อจัดสรรบล็อกเป็นที่เรียบร้อยแล้ว จะย้ายข้อมูลจากหน่วยความจำลงในแผ่นจานแม่เหล็ก จากนั้นจะทำการปิดแฟ้มข้อมูล

1.5 การปิดแฟ้มข้อมูล

การปิดแฟ้มข้อมูลเป็นการปรับปรุงข้อมูลจากหน่วยความจำลงในแผ่นจานแม่เหล็ก ซึ่งขั้นตอนจะแสดงดังรูปที่ ก.10

จากตัวอย่าง ต้องการปิดแฟ้มข้อมูล หมายเลข 17 ที่ทำการ access อยู่ จะมาหาแฟ้มข้อมูลนั้นในส่วนควบคุมแฟ้มข้อมูล โดยใช้หมายเลขแฟ้มข้อมูล จากนั้นจะทำการหาแฟ้มข้อมูลนั้นในไดเรกทอรีแฟ้มข้อมูล เพื่อหาแฟ้มข้อมูลหมายเลขเดียวกัน ถ้าพบจะทำการย้ายข้อมูลจากส่วนควบคุมแฟ้มข้อมูลในหน่วยความจำ ลงในแผ่นจานแม่เหล็ก แต่ถ้าไม่พบจะทำการสร้างไดเรกทอรีของแฟ้มข้อมูลใช้อีก 1 แฟ้มข้อมูล



รูปที่ ก.10 แสดงตัวอย่างการเปิดแฟ้มข้อมูล

ภาคผนวก ข.

โปรแกรมการย้ายข้อมูลจากโปรแกรมแมกดรอว์ไปยังโปรแกรมแมกดราฟต์

```
program DRAWFILES ;
{
    purpose          provides a desk accessory template
    last update      29 APR 1988
}


{$D PasDeskAcc}                         { compile as desk accessory w/ ID=12 }
{$U-}                                   { don't automatically include units  }


uses MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,MacPrint;

const

  accEvent        =    64;              { DA needs to handle event           }
  accRun          =    65;              { DA needs/wants periodic update     }
  accCursor       =    66;              { DA needs to change cursor shape    }
  accMenu         =    67;              { DA needs to handle a menu item     }
  accUndo         =    68;              { DA needs to handle Undo command    }
  accCut          =    70;              { DA needs to handle Cut command     }
  accCopy         =    71;              { DA needs to handle Copy command    }
  accPaste        =    72;              { DA needs to handle Paste command   }
  accClear        =    73;              { DA needs to handle Clear command   }
  goodBye         =    -1;              { heap is about to be reinitialized  }

  dCtlEnable      = $0400;              { DA can respond to control calls    }
  dNeedGoodBye    = $1000;              { DA wants call before heap re-init  }
  dNeedTime       = $2000;              { DA needs time for periodic action  }
  dNeedLock       = $4000;              { DA needs to be locked in memory    }

type

  DAGlobals       =                     { this is 'global' data for the DA   }
    record
      theMenu     : MenuHandle;                   { holds DA's menu          }
      theString   : String;                       { holds a string.          }
      thePlace    : Point;                        { spot to write a char     }
      CHandle     : CursHandle;                   { the cursor you want      }
     {   you can define this record to be whatever you want; use its        }
     {   fields to declare your "global" variables, that is, those data     }
     {   structures which need to pass values between Open, Control and     }
     {   Close, and which need to be preserved between calls to Open.       }
     {   The fields above are just examples, not necessities.               }
    end;

  DAGlobalsP      = ^DAGlobals;                   { pointer to globals       }
  DAGlobalsH      = ^DAGlobalsP;                  { handle to globals        }

  EventRecP       = ^EventRecord;


(*  here's the definition of the device control entry record.  It's found
    in the interface for the unit OSIntf, but is repeated here (within a
    comment) for your reference.

  DCtlEntry       =
    record
      DCtlDriver      : Ptr;        { ptr to ROM or handle to RAM driver  }
      DCtlFlags       : Integer;    { flags}
      DCtlQHdr        : QHdr;       { driver's i/o queue}
      DCtlPosition    : LongInt;    { byte pos used by read and write calls }
      DCtlStorage     : Handle;     { hndl to RAM drivers private storage  }
      DCtlRefNum      : Integer;    { driver's reference number  }
      DCtlCurTicks    : LongInt;    { long counter for timing system task calls }
      DCtlWindow      : Ptr;        { ptr to driver's window if any  }
```

```
      DCtlDelay       : Integer;   { number of ticks between sysTask calls }
      DCtlEMask       : Integer;   { desk acessory event mask }
      DCtlMenu        : Integer;   { menu ID of menu associated with driver }
  END;

  DCtlPtr          = ^DCtlEntry;
  DCtlHandle       = ^DCtlPtr;

end of commented section *)


procedure  Open(var  Device: DCtlEntry);  forward;
procedure  Control(var  Device: DCtlEntry;  Param: LongInt;  Code: Integer);  forward;
procedure  Close(var  Device: DCtlEntry);  forward;

procedure  Open;
{
    purpose          this routine is called when the desk accessory is first
                     launched.  It needs to set the appropriate fields in
                     Device (a variable of type DCtlEntry), allocate any global
                     storage, bring up the window (if there is one), initialize
                     any global variables, and generally set up housekeeping

}

var
  SavePort        : GrafPtr;
  TempVal         : Integer;
  theImagePtr     : Ptr;
  ImagePtr  : Ptr;

  {$I DrawFileMgr}


begin

      GetDrawImage(theImagePtr);
 end;

procedure  Control;
{
    purpose          handles events, periodic updates

}
var
  SavePort        : GrafPtr;

{$I nongl.Inc}     { include event-handling routines for desk accessory     }

begin  { main body of proc Control }
  with Device do begin                                  { access DCtlEntry   }
    HLock(dCtlStorage);                                 { freeze reloc block}
    with DAGlobalsH(dCtlStorage)^^    do begin          { access globals     }
      GetPort(SavePort);                                { save current port  }
      SetPort(GrafPtr(dCtlWindow));                     { switch to ours     }
      case Code of  { most of these routines are found in MyDA.Inc        }
        accEvent      : HandleEvent(EventRecP(Param));{     deal with event   }
        accRun        : HandleTick;                     { deal with timer    }
        accCursor     : CursorAdjust;                   { change cursor      }
        accMenu       : HandleMenu(Param);              { item from menu     }
        accUndo..AccClear
                      : HandleEdit(Code);               { standard edit cmds}
        goodBye       : Close(Device)                   { about to shut down}
      end;
      SetPort(SavePort)                                 { restore old port   }
    end;
    HUnLock(dCtlStorage)                                { unfreeze rel block}
  end
```

```
end; { of proc Control }

procedure Close;
{
    purpose          put stuff away and turn off desk acc

}
var
  MHandle            : MenuHandle;
begin
  with Device, DAGlobalsH(dCtlStorage)^^    do begin
    DisposeWindow(GrafPtr(dCtlWindow));                  { get rid of window }
    DeleteMenu(dCtlMenu);                                { remove from bar   }
    DisposeMenu(theMenu);                                { get rid of menu   }
    DrawMenuBar;                                         { and update menubar}
    DisposHandle(dCtlStorage);                           { get rid of globals}
    dCtlWindow  := NIL                                   { clear window ptr  }
  end
end; { of proc Close }

begin
  { 'main body' of desk accessory }
end.
```

```
Procedure  DoMessage  (mes0  : Str255;
mes1  : str255;  mes2  : str255  ;mes3  : str255);
const
   MessageDialog   = 258;
var
   dialogP  : DialogPtr;
   item  : integer;
   dlogRect   : rect;
begin
  ParamText(mes0,mes1,mes2,mes3);
  SetRect(dlogRect,    100,  100,  400,200);
  dialogP  := GetNewDialog(MessageDialog,     NIL,pointer(-1));
  IF dialogP  = NIL  Then
    Begin
      SysBeep(5);
      ExitToShell;
    end;
  InitCursor;
  ModalDialog(Nil,    item);
  DisposDialog(dialogP);
End;

Procedure  SFGetDraw(var   theReply   : SFReply);
Const
  SFPutLeft   = 100;
  SFPutTop   = 100;
var
  SFPutPt   : Point;
  PICT_list   : SFTypeList;
Begin
  PICT_List[0]   := 'PICT';
  SetPt(SFPutPt,SFPutLeft,SFPutTop);
  SFGetFile(SFPutPt,    '',NIL,  1, PICT_list,  NIL,  theReply);
end;

Procedure  CloseOldFile(refNum    : integer;  vRefNum   : integer);
var
  err  : OSErr;
Begin
  err  := FSClose(refNum);
  if err <> noerr  Then
  begin
    doMessage('FSClose    error','CloseOldFile    routine',
    'Could  not close  file',  '');
  end;
  err  := FlushVol(Nil,    vRefNum);
  if err <> noErr  Then
  begin
    doMessage('FlushVol    error','CloseOldFile    routine',
    'Could  not  flush  volume','');
  end;
end;

Procedure  ReadDrawFile(refNum    : integer;  var  PackedBitsPtr    : Ptr);
Label
   1;
var
  bytes  : LongInt;
  str1  : str255;
  err  : OSErr;
Begin
  PackedBitsPtr    := NIL;
  err  := GetEOF(refNum,bytes);
  IF err <> noErr  Then
  begin
    doMessage('GetEOF    error',  'ReadDrawFile   routine',
    'Could  not  find  file  end','');
  end;
```

```
  bytes  := bytes  - 512;

  {=============================================================}
  {
  if  odd(bytes)   then
    begin
      NumToString(bytes,str1);
      Str1  := concat('Bytes   - header  = ',str1);
      doMessage('Logical    EOF  Odd',str1,'Not   a MacDraw  File','')

    end
  else
    begin
      NumToString(bytes,str1);
      str1  := concat('Bytes   - header  =' ,str1);
      doMessage('Reading    Draw  type...',str1,'','');

    end;
  }
  {=============================================================}
  PackedBitsPtr   := NewPtr(bytes);

  if MemError  <> noErr  THEN
  begin
     PackedBitsPtr   := NIL;
     doMessage('PackBitsPtr    Memory  err','ReadDrawFile    routine',
     'No room  to read  in data','');
     Goto  1;
  end;

  err  := SetFPos(refNum,   FSFromstart,512);
  if err  <> noErr  Then
  begin
     doMessage('SetFPos    error','ReadDrawFile    routine',
     'Could  not  set  file',  'at  start  of  data');
  end;

  err  := FSRead(refNum,bytes,PackedBitsPtr);
  if err  <> noerr  then
  begin
    doMessage('FSRead    error','ReadDrawFile    routine',
    'Problem  reading  in file','');
    goto  1;
  end;
1:
end;

Procedure  GetDrawImage(Var     ImagePtr   : Ptr);
Label
  2;
const    MaxDrawSize   = 3072;
type
   MacD    =    Record ,
                   MacSize   : Integer;
                   MacLoc    : Rect;
                   MacVers   : LongInt;
                   MacDat    : Array  [1..MaxDrawSize]    of  byte;
               end;
var
  bytes  : LongInt;
  refNum  : integer;
  theReply  : SFReply;
  err  : OSErr;
  packedBitsPtr    : Ptr;
  destPtr,SrcPtr    : Ptr;
  MacPtr    : ^MacD;
```

```
begin
  ImagePtr  := NIL;
  SFGetDraw(theReply);
  with  theReply  do
  begin
    if Not Good  then
      goto 2
    else  begin
      err  := FSOpen(fname,   vRefNum,refNum);
      if err  <> 0 then
      begin
        doMessage('FSOpen   error  on file', 'GetDrawImage   routine',
        'Can  not  Open  file','');
        goto  2;
      end;
    {-------------------------------------------------------------}

      ReadDrawFile(refNum,    PackedBitsPtr);

    {-------------------------------------------------------------}
      CloseOldFile(refNum,vrefNum);
      if PackedBitsPtr   = NIL  then
      begin
        Goto  2;
      end;

      MacPtr  := Pointer(packedBitsPtr);
      err  := ZeroScrap;
      err  := PutScrap(MacPtr^.MacSize,'PICT',packedBitsPtr);
      {err  := PutScrap(52,'PICT',packedBitsPtr);}

      {ImagePtr  := NewPtr(SizeOfDrawImage);
      If Memerror  <> 0  then
      begin
        doMessage('ImagePtr   Memory  err', 'GetDrawImage   routine',
        'No room  for  image','');
        goto  2;
      end;
      SrcPtr  := PackedBitsPtr;
      DestPtr  := IMagePtr;
      saveStart  := ord(destPtr);}
      {UnpackBits(SrcPtr,DestPtr,SizeOfDrawImage       DIV 2);}
      {bytesUnPacked   := Ord(DestPtr)  -SaveStart;}
      {UnPackBits(SrcPtr,DestPtr,SizeOfDrawImage       - bytesUnPacked);}


      DisposPtr(packedBitsPtr);

    end;
  end;
 2:
 end;

{Procedure  DisplayDrawFile(ImagePtr     : Ptr);
LABEL
   3;
var
  pageBits  : BitMap;
  drawRect  : Rect;
  screen    : Rect;

Begin
  If ImagePtr  = NIL Then
  begin
    Goto 3;
  end;
  with pageBits  do
  begin
```

```
        baseAddr   := ImagePtr;
        rowBytes   := 72;
        SetRect(bounds,0,0,576,720);                                    118
    end;
    screen  := screenBits.bounds;
    SetRect(drawRect,0,0,576,720);

    (*
    SetRect(drawRect,   screen.left  + 148,screen.top   +0,screen.right   - 148,  screen.bottom   -72);

*)
    CopyBits(pageBits,   thePort^.portbits,pagebits.bounds,drawRect,SrcCopy,NIL);

  3:
  end;)
```

ประวัติผู้เขียน

นางสาว กุณฑิรา สมุทรกลิน เกิดวันที่ 24 กันยายน พ.ศ. 2504 สำเร็จการศึกษาเศรษฐศาสตรบัณฑิต สาขาปริมาณวิเคราะห์ คณะเศรษฐศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2526