ระบบจัดการรูปภาพ

IMAGE MANAGEMENT SYSTEM

โดย

นางสาวชุติมา ทรัพย์สิรินทร์ 34102098

อาจารย์ที่ปรึกษา

อาจารย์วัชระ ฉัตรวิริยะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

035001

ปริญญานิพนธ์ปีการศึกษา 2537
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบจัดการรูปภาพ

ผู้จัดทำ

นางสาวชุติมา ทรัพย์สิรินทร์ 34102098

.....อาจารย์ที่ปรึกษา
(อาจารย์วัชระ ฉัตรวิริยะ)

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบจัดการรูปภาพ

ชุติมา ทรัพย์สิรินทร์
อาจารย์วัชระ ฉัตรวิริยะ อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้จัดทำขึ้นมาเพื่อประกอบโครงงานซึ่งเป็นการประยุกต์ใช้ระบบปฏิบัติการแบบมัลติทาร์คกิง (Multitasking Operating System) และการทำงานแบบแบคกราวน์โปรเซส โดยนำมาใช้กับการจัดการไฟล์รูปภาพหรือไฟล์อื่นๆ ที่มีขนาดใหญ่ แล้วต้องการลดขนาดไฟล์ลงเพื่อประหยัดเนื้อที่ โครงงานที่พัฒนาขึ้นมามีความสามารถคือใช้ในการจัดการรูปภาพ ทั้งด้านการย่อ (Compression) การขยาย (Decompression) และแสดงผล (Display) รูปภาพ และการทำงานของระบบสามารถทำได้โดยเมื่อระบบได้รับตัวเลือกในการย่อข้อมูลตามที่ผู้ใช้ต้องการแล้วจะสร้างโปรเซสขึ้นมาและในโปรเซสนี้จะทำการเลือกวิธีการย่อข้อมูลที่เหมาะสมกับตัวเลือกที่ผู้ใช้ต้องการมากที่สุด การทำงานในด้านการขยายข้อมูลเมื่อผู้ใช้ใส่ชื่อไฟล์ที่ต้องการจะขยายแล้วระบบจะทำการตรวจสอบว่าไฟล์ดังกล่าวได้ถูกย่อเก็บไว้ด้วยวิธีใดและจะเลือกวิธีการขยายที่ถูกต้องสำหรับไฟล์นั้น ด้านการแสดงผลรูปภาพระบบจะตรวจสอบไฟล์ที่ได้รับว่าเป็นไฟล์ที่ได้ถูกย่อไว้หรือไม่ ถ้าหากเป็นไฟล์ที่ย่อไว้ระบบจะทำการขยายข้อมูลและเก็บไฟล์ที่ขยายเอาไว้เป็นไฟล์ชั่วคราว (Temporary File) เมื่อแสดงผลเสร็จสิ้นไฟล์ดังกล่าวจะถูกลบทิ้งไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IMAGE MANAGEMENT SYSTEM

Chutima Subsirin
Watchara Chatwiriya Advisor
1994

Abstract

This thesis is created for Image Management System which applied concept of multitasking operating system and background processing for manage large files that want to reduce size of files. This system develope for compression , decompression and display images. The part of compression and display can run in background process . Compression process is able to select the best way to compress file by itself. Decompression process is able to detect kind of compression for each input file and use the right method to decompress it. In display process the file if it is compressed , the program will decompress and keep the result in temporary file which will be deleted after finish.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

บทที่ 1 บทนำ

1.1 วัตถุประสงค์ 1

1.2 แนะนำเนื้อหาในปริญญานิพนธ์ 1

บทที่ 2 หลักการที่ใช้ในการพัฒนาระบบ

2.1 การเขียนโปรแกรมในสสภาพแวดล้อมแบบเอกซ์วินโดว์ 3

2.1.1 หลักการไคลน์-เซิฟเวอร์โมเดล 3

2.1.2 เอกซ์โปรโตคอล 3

2.1.3 โครงสร้างของโปรแกรมบนระบบเอกซ์วินโดว์ 4

2.1.4 การเขียนโปรแกรมโดยใช้เอกซ์ลิปฟังก์ชัน 4

2.2 การย่อ - ขยายข้อมูล 8

2.2.1 การย่อ-ขยายข้อมูลด้วยวิธี RLE 8

2.2.2 การย่อ-ขยายข้อมูลด้วยวิธี Huff-Man 12

2.2.3 การย่อ-ขยายข้อมูลด้วยวิธี LZW 15

2.3 การทำงานแบบแบคกราวน์โปรเซสในระบบหลายผู้ใช้ 19

2.3.1 คำจำกัดความพื้นฐานบนยูนิกซ์ 19

2.3.2 การสร้างโปรเซสใหม่ในระบบและการเรียกใช้โปรแกรมอื่น 22

2.3.3 สัญญาณ 22

2.3.4 การจับสัญญาณในโปรเซสและนำมาใช้งาน 25

บทที่ 3 การพัฒนาระบบ

3.1 ความต้องการของระบบ 26

3.2 ระบบที่ได้พัฒนาขึ้นและการทำงานในส่วนต่างๆของระบบ 27

3.2.1 โปรแกรมการทำงานหลัก 27

3.2.2 โปรแกรมการย่อข้อมูล 33

3.2.3 โปรแกรมการแสดงผลรูปภาพ 40

บทที่ 4 สรุปผลการทำงาน

4.1 ผลที่ได้และแนวทางการพัฒนาระบบในอนาคต 41

4.2 ปัญหาที่พบในการพัฒนาระบบ 42

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

รูป 2.1 แสดงการเขียนโปรแกรมแบบอีเวนท์ไดรเวน 8

รูป 2.2 การแทนตัวอักษรที่ถูกย่อข้อมูลแบบ RLE 9

รูป 2.3 ขั้นตอนการย่อข้อมูลโดยวิธี RLE 10

รูป 2.4 ขั้นตอนการขยายข้อมูลโดยวิธี RLE 11

รูป 2.5 วิธีย่อข้อมูลแบบ MNP5 12

รูป 2.6 ตัวอย่างแสดงความมีของตัวอักษรในไฟล์ 13

รูป 2.7 ตัวอย่างการสร้างโครงสร้างต้นไม้จากความถี่ของข้อมูล 13

รูป 2.8 ตัวอย่างการกำหนดรหัสจากโครงสร้างต้นไม้ 14

รูป 2.9 ตัวอย่างการเขียนโปรแกรมในการย่อข้อมูลแบบ LZW 16

รูป 2.10 ตัวอย่างการสร้างตารางสตริงและการย่อข้อมูล 17

รูป 2.11 ตัวอย่างการเขียนโปรแกรมในการขยายข้อมูลแบบ LZW 18

รูป 2.12 ตัวอย่างการสร้างตารางสตริงและการขยายข้อมูล 18

รูป 2.13 แสดงส่วนประกอบของโปรเซส 21

รูป 3.1 การทำงานของโปรแกรมหลัก 28

รูป 3.2 รายละเอียดหน้าจอส่วนติดต่อกับผู้ใช้ 29

รูป 3.3 การทำงานในฟังก์ชันขยายข้อมูล 32

รูป 3.4 การทำงานของโปรแกรมการย่อข้อมูล 34

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตาราง 2.1 คุณลักษณะของวินโดว์ 5

ตาราง 2.2 ความแตกต่างระหว่างพิคแมปและวินโดว์ 6

ตาราง 2.3 ชนิดของอีเวนท์ 7

ตาราง 2.4 ตัวอย่างการย่อข้อมูลด้วยวิธี RLE 9

ตาราง 2.5 ชนิดของสัญญาณ 23

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

เนื่องจากในปัจจุบันนี้มีการใช้งานไฟล์รูปภาพกันเป็นจำนวนมาก ทั้งในด้านการแสดงผลและการจัดเก็บ และจากการที่ไฟล์รูปภาพมักจะมีขนาดใหญ่ ต้องการเนื้อที่ในการเก็บไฟล์จำนวนมาก เพื่อการประหยัดเนื้อที่จึงควรมีการย่อไฟล์รูปภาพให้เล็กลงเพื่อให้เหมาะสมกับเนื้อที่ที่มีอยู่ แต่ทั้งนี้จะพิจารณาเห็นได้ว่าการย่อข้อมูลประกอบไปด้วยวิธีต่างๆมากมาย แต่ละวิธีก็มีข้อจำกัดและความเหมาะสมกับรูปแบบของข้อมูลต่างๆกัน อีกทั้งยังต้องคำนึงถึงความต้องการของผู้ใช้ที่ต้องการผลในการย่อข้อมูลในแบบใด นอกจากปัญหาในการเลือกวิธีการย่อข้อมูลที่เหมาะสมกับความต้องการของผู้ใช้แล้ว การนำไฟล์มาย่อขนาดย่อมต้องเสียเวลาในการย่อข้อมูลด้วย เพื่อตอบสนองกับความต้องการผู้ใช้ดังกล่าวระบบจัดการรูปภาพนี้จึงได้ถูกพัฒนาขึ้นมาด้วยหลักการพื้นฐานคือสามารถเลือกวิธีการย่อข้อมูลได้เองโดยที่ผู้ใช้เพียงระบุความต้องการให้แก่ระบบ และในขณะที่ระบบทำงานเกี่ยวกับการย่อข้อมูลอยู่นั้นผู้ใช้ยังสามารถใช้งานส่วนอื่นๆของระบบพร้อมกันได้ การขยายข้อมูลเมื่อผู้ใช้ต้องการจะขยายไฟล์ที่ได้ถูกย่อไว้แล้วระบบจะตรวจสอบชนิดของการย่อในไฟล์นั้นแล้วทำการขยายตามชนิดนั้นเอง นอกจากการย่อ และขยาย ยังมีการแสดงผลเพื่อให้ผู้ใช้สามารถดไฟล์รูปภาพได้จากระบบพร้อมกันได้หลายไฟล์ทั้งไฟล์ที่ย่อแล้วหรือไฟล์ปกติ นอกจากนี้ระบบยังพัฒนาการติดต่อกับผู้ใช้ให้ง่ายและสะดวกในการใช้งานระบบ

1.1 วัตถุประสงค์ของโครงงาน

ระบบที่สร้างขึ้นควรมีความสามารถดังนี้

1. สามารถจัดเก็บ แสดงผล และพิมพ์ภาพออกทางเครื่องพิมพ์ได้
2. ในการจัดเก็บรูปภาพ สามารถเลือกวิธีการย่อภาพวิธีต่างๆ เพื่อจัดเก็บให้เหมาะสมกับความต้องการของผู้ใช้
3. สามารถทำงานในขั้นการเลือกวิธีจัดเก็บรูปภาพเป็นแบคกราวน์โปรเซสได้
4. มีการติดต่อกับผู้ใช้แบบกราฟิคได้

1.2 แนะนำเนื้อหาในปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้จัดทำขึ้นโดยแบ่งเนื้อหาออกเป็น 4 บท แต่ละมีเนื้อหาเกี่ยวกับสิ่งต่างๆดังต่อไปนี้

บทที่ 1 เป็นบทที่กล่าวถึงที่มาของปัญหาที่มีส่วนทำให้เกิดความคิดในการเริ่มทำโครงงานขึ้น และการตั้งวัตถุประสงค์ของโครงงานเพื่อแก้ปัญหาที่มีอยู่นั้น

บทที่ 2 กล่าวถึงหลักการและความรู้ที่ใช้ในการพัฒนาโครงงานแต่ละส่วนขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3 เป็นบทที่มีรายละเอียดเกี่ยวกับการพัฒนาระบบขึ้นจริงๆ กล่าวถึงความต้องการของระบบ และการสร้างระบบในแต่ละส่วนขึ้นเพื่อตอบสนองตามความต้องการนั้น

บทที่ 4 ผลสรุปของการพัฒนาระบบ คือ ผลที่ได้จากระบบในการทำงาน แนวทางในการพัฒนาระบบในอนาคต ปัญหาที่พบในการพัฒนาระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
หลักการที่ใช้การพัฒนาระบบ

2.1 การเขียนโปรแกรมในสภาพแวดล้อมแบบเอกซ์วินโดว์

เอกซ์วินโดว์ (X-Windows) เป็นระบบวินโดว์ที่ไม่ขึ้นกับฮาร์ดแวร์ และ ระบบปฏิบัติการ(Operating System Independent) ที่มีการแสดงผลเป็นแบบบิทแมปกราฟิก(Bitmap Graphic) ไม่ว่าจะเป็นรูปภาพ หรือ ตัวอักษรต่างๆ ในระบบเอกซ์วินโดว์จะมีการแสดงผลที่มีการแบ่งหน้าจอออกเป็นส่วนๆ ที่เรียกว่า วินโดว์ (Windows) โดยแต่ละส่วนจะมีส่วนรับข้อมูลและส่วนแสดงผล(Input and Output) เป็นของตนเอง โดยส่วนใหญ่แล้วส่วนรับข้อมูลจะเป็นแป้นพิมพ์(Keyboard) และตัวชี้(Pointer or Mouse) และการทำงานบนสภาพแวดล้อมแบบเอกซ์วินโดว์ยังเป็นการทำงานแบบไคลน์-เซิฟเวอร์ดังจะกล่าวถึงต่อไป

ในโครงงานนี้ได้พัฒนาระบบบนระบบปฏิบัติการลีนุกซ์ (Linux) ซึ่งเป็นระบบปฏิบัติการที่มีความเหมือนกับยูนิกซ์เป็นส่วนใหญ่ แต่เป็นระบบที่ไม่ต้องเสียค่าใช้จ่ายในการนำมาใช้งาน และสามารถใช้งานบนเครื่องพีซีโดยทั่วไปได้ ส่วนระบบเอกซ์วินโดว์ที่ใช้ในการพัฒนาเป็นเอกซ์วินโดว์เวอร์ชัน 5

2.1.1 ไคลน์-เซิฟเวอร์โมเดล (Client - Server Model)

เอกซ์วินโดว์ เป็นระบบที่ทำงานแบบลูกข่ายกับผู้ให้บริการ (Client - Server) คือการที่โปรแกรม หรือ ลูกข่าย (Client) ที่ทำงานบนระบบไม่ต้องใช้งาน แป้นพิมพ์ ตัวชี้ จอภาพ วินโดว์โดยตรง หากแต่ลูกข่ายจะใช้งานผ่านโดยผู้ให้บริการ (Server) และผู้ให้บริการหนึ่งตัวสามารถให้บริการแก่ลูกข่ายได้พร้อมๆกันหลายตัวในระบบ โดยที่ลูกข่ายและผู้ให้บริการมีการติดต่อสื่อสารกันผ่านกฎเกณฑ์ที่เรียกว่าเอกซ์โปรโตคอล

2.1.2 เอกซ์โปรโตคอล (X-Protocol)

การติดต่อสื่อสารข้อมูลกันระหว่างลูกข่ายกับผู้ให้บริการเป็นส่วนสำคัญในระบบนี้ ข่าวสาร ที่ใช้สื่อสารกันในระบบสามารถแบ่งได้หลายชนิดคือ

1. การที่ลูกข่ายส่งคำร้องขอไปให้กับผู้ให้บริการเพื่อขอข้อมูลเกี่ยวกับระบบเพิ่มเติมเรียกว่า คำร้องขอ (Request) เช่น ขนาดวินโดว์ คุณสมบัติบางอย่างเกี่ยวกับหน้าจอ ชนิดของแป้นพิมพ์ที่ถูกกด โดยคำร้องขอเหล่านี้จะส่งไปให้เพื่อให้ผู้บริการได้ส่งข้อมูลมาให้

2. ข้อมูลที่ผู้ให้บริการส่งให้แก่ลูกข่ายเรียกว่า อีเวนท์ (Event) เช่น มีการกดแป้นพิมพ์ ลากตัวชี้ การเคลื่อนที่ของตัวชี้ มีการเปลี่ยนแปลงขนาดของวินโดว์ ผู้ให้บริการจะส่งข่าวสารต่างๆนี้ให้แก่ลูกข่ายเพื่อนำไปใช้งานต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ข้อมูลที่ผู้ให้บริการส่งเพื่อตอบกลับตามคำร้องขอของลูกข่ายเรียกว่า การตอบรับ (Reply) ได้แก่ข้อมูลที่ลูกข่ายต้องการใช้ในการทำงานแต่ละขั้นตอนนั่นเอง ดังนั้นเมื่อลูกข่ายมีคำร้องขอไปให้แก่ผู้ให้บริการ ลูกข่ายจะต้องรอจนกว่าผู้ให้บริการจะตอบกลับมาจึงจะทำงานขั้นต่อไป

4. ผู้ให้บริการจะส่งข่าวสารเกี่ยวกับข้อผิดพลาดที่เกิดขึ้นในระบบให้แก่ลูกข่ายเรียกว่า ความผิดพลาด (Error)

การเขียนโปรแกรมเพื่อทำงานบนเอกซ์วินโดว์จึงเป็นการใช้งานฟังก์ชันที่เกี่ยวข้องกับเอกซ์โปรโตคอลเหล่านี้

2.1.3 โครงสร้างของโปรแกรมบนระบบเอกซ์วินโดว์

โปรแกรม หรือ ลูกข่ายในระบบ ก็คือ โปรแกรมที่ผู้ใช้ได้พัฒนาขึ้นมา ในการสร้างโปรแกรมเพื่อใช้งานบนระบบ ผู้ใช้สามารถเลือกเขียนขึ้นได้จาก เอกซ์ลิป (Xlib) , เอกซ์-ทูลคิดท์ (X-Toolkit) และ วิดเจท (Widget) โดยเอกซ์ลิปจะเป็นการใช้งานฟังก์ชันที่มีในระบบโดยตรง นั่นคือการติดต่อกับเอกซ์โปรโตคอลโดยตรง ส่วนทูลคิดท์จะเป็นการเขียนโปรแกรมในระดับสูงขึ้นมาจากเอกซ์ลิปโดยจะเรียกใช้งานจากฟังก์ชันที่เรียกเอกซ์ลิปคอลอีกทอดหนึ่ง การใช้งานเอกซ์ทูลคิดท์จะใช้ร่วมกับวิดเจทซึ่งจะเป็นการเขียนโปรแกรมแบบออปเจกต์โอเรียนเตด Object Oriented Programming และในระบบที่พัฒนาขึ้นนี้ได้เลือกการพัฒนาโปรแกรมโดยใช้เอกซ์ลิปฟังก์ ชัน ทั้งนี้เพราะระบบไม่จำเป็นต้องติดต่อกับผู้ใช้และผู้ให้บริการมากนัก การใช้เอกซ์ลิปจึงพอเพียงสำหรับการพัฒนาระบบนี้

2.1.4 การเขียนโปรแกรมโดยใช้เอกซ์ลิปฟังก์ชัน

การเขียนโปรแกรมโดยใช้ เอกซ์ลิปฟังก์ชัน ถึงแม้จะสามารถติดต่อกับเอกซ์โปรโตคอลได้โดยตรง ซึ่งจะทำให้โปรแกรมที่ได้มีความยืดหยุ่นและสามารถโต้ตอบกับระบบได้อย่างดีก็ตาม แต่ก็มีข้อเสียก็คือจะทำให้การเขียนโปรแกรมยุ่งยากซับซ้อน ไม่เหมาะกับการเขียนโปรแกรมขนาดใหญ่

ความรู้เบื้องต้นในการเขียนโปรแกรมบนโดยใช้เอกซ์ลิปฟังก์ชันที่ควรทราบคือ

2.1.4.1 ทรัพยากรในระบบ (Resource)

ทรัพยากรในระบบ หมายถึง เนื้อที่ซึ่งอยู่ภายในผู้ให้บริการที่ใช้เก็บข้อมูลที่จำเป็นสำหรับการใช้งานของลูกข่าย โดยทรัพยากรแต่ละตัวจะต้องมีตัวเลขที่ใช้ระบุถึงทรัพยากรแต่ละตัวนั้นๆเรียกว่า ค่าประจำตัวของทรัพยากร (Resource ID) ซึ่งค่าประจำตัวนี้มีไว้เพื่อให้ลูกข่ายระบุถึงทรัพยากรที่ต้องการใช้งาน ดังนั้นลูกข่ายแต่ละตัวจึงไม่จำเป็นต้องเก็บข้อมูลของทรัพยากรที่ต้องการใช้ไว้ เพราะผู้ให้บริการจะเป็นผู้เก็บข้อมูลเหล่านี้ไว้ ลูกข่ายเพียงแต่อ้างถึงค่าเหล่านี้ในการติดต่อกับผู้ให้บริการเพื่อเรียกใช้งานทรัพยากรแต่ละตัวเท่านั้น ทรัพยากรในระบบประกอบไปด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. <u>คุณลักษณะของวินโดว์ (Window Attributes)</u> หมายถึง ค่าต่างๆที่เกี่ยวข้องกับวินโดว์ เพื่อใช้ในการกำหนดลักษณะวินโดว์ที่ต้องการในแต่ละโปรแกรม ได้แก่

Window Attributes	คำอธิบาย
Background-Pixmap	ภาพเบื้องหลังของวินโดว์
Background-Pixel	สีที่จะใช้เป็นเบื้องหลังของวินโดว์
Border-Pixmap	ภาพเบื้องหลังของขอบของวินโดว์
Border-Pixel	สีของขอบของวินโดว์
Bit-Gravity	ค่านี้จะใช้ในการระบุตำแหน่งของสิ่งที่อยู่ในวินโดว์หลังจากที่วินโดว์มีการเปลี่ยนแปลงขนาด
Win-Gravity	ระบุตำแหน่งของวินโดว์เมื่อวินโดว์ผู้ปกครองมีการเปลี่ยนแปลงขนาด
Backing-Store	ค่าที่กำหนดว่าจะเก็บรายละเอียดของวินโดว์ไว้หรือไม่ มีประโยชน์ในการจัดการกับเอกซ์โพสอีเวนท์
Backing-Planes	คล้ายกับ backing-store แต่เลือกเก็บเฉพาะเพลนที่ระบุไว้เท่านั้น
Backing-Pixel	ใช้ร่วมกับ backing-planes ซึ่งจะกำหนดสีที่ต้องการเก็บ
Save-Under	ค่านี้ใช้เพื่อกำหนดว่าจะเก็บรายละเอียดของวินโดว์ที่อยู่ข้างใต้ไว้หรือไม่ ใช้ประโยชน์ในเอกซ์โพสอีเวนท์ได้
Event-Mask	ระบุอีเวนท์ที่วินโดว์ต้องการใช้งาน
Do-not-Propagate-Mask	ระบุอีเวนท์ที่วินโดว์ไม่ต้องการให้มีการส่งผ่านจากยังวินโดว์ลูกมายังวินโดว์ปัจจุบันได้
Override-Redirect	เพื่อกำหนดว่าการทำงานกับวินโดว์นั้นจะให้ผ่านวินโดว์เมเนเจอร์หรือติดต่อโดยตรงกับผู้ให้บริการโดยตรง
Colormap	เพื่อกำหนดคัลเลอร์แมปที่ใช้กับวินโดว์
Cursor	เพื่อกำหนดชนิดของเคอร์เซอร์ที่ใช้กับวินโดว์

ตาราง 2.1 คุณลักษณะของวินโดว์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. <u>พิคแมป (Pixmap)</u> พิคแมปมีลักษณะที่เหมือนและแตกต่างกับวินโดว์ดังนี้ คือ

	พิคแมป	วินโดว์
1. มีลักษณะเป็นรูปสี่เหลี่ยม	ใช่	ใช่
2. สามารถวาด หรือ เขียนสิ่งต่างๆ ลงไปได้	ใช่	ใช่
3. มองเห็นได้	ไม่ใช่	ใช่
4. มีขอบรอบ	ไม่ใช่	ใช่

ตาราง 2.2 ความแตกต่างระหว่างพิคแมปและวินโดว์

จากลักษณะที่กล่าวมาข้างต้นนี้จะเห็นได้ว่าพิคแมปมีลักษณะแตกต่างอย่างเห็นได้ชัดกับวินโดว์ ก็คือ พิคแมป ไม่สามารถมองเห็นได้ การใช้งานพิคแมปจึงมักใช้สำหรับเก็บภาพหรือกราฟิคเพื่อใช้ตั้งเป็นแบคกราวน์ของวินโดว์หรือขอบวินโดว์หรือใช้ในการเก็บภาพหรือรายละเอียดที่ต้องการก็อปปี้โดยตรงลงไปยังพื้นที่บนวินโดว์ก็ได้

3. <u>คัลเลอร์แมป (Colormap)</u> หมายถึง รูปแบบของสีที่ต้องการใช้ในโปรแกรม เช่น 8-บิท อินไดเร็กคัลเลอร์แมป , 8-บิท เกรย์สเกลคัลเลอร์แมป เป็นต้น

4. <u>รายละเอียดทางกราฟิค (Graphic Context)</u> หมายถึงข้อกำหนดที่ใช้ในการสร้างรูปแบบทางกราฟิคต่าง ๆ เช่น ความกว้าง และชนิดของเส้นที่จะสร้าง สีของแบคกราวน์ และฟอร์กราวน์ พิคแมปที่ใช้ ลักษณะของการวาดส่วนโค้ง การลงสี เป็นต้น

5. <u>เคอร์เซอร์ (Cursor)</u> หมายถึง ข้อมูลในการกำหนดการใช้สัญญลักษณ์แทนตัวชี้ในแต่ละวินโดว์ เช่น สี ขนาด รูปร่าง เป็นต้น

6. <u>ฟอนท์ (Font)</u> ข้อมูลในการกำหนดชนิดของฟอนท์ที่จะเรียกใช้

2.1.4.2 การจัดการเกี่ยวกับอีเวนท์

เนื่องจากการโปรแกรมโดยใช้เอกซ์ลิปนั้นต้องติดต่อโดยตรงกับเอกซ์โปรโตคอล ทำให้ต้องเข้าใจถึงการทำงานของอีเวนท์และการจัดการอีเวนท์เป็นอย่างดี อีเวนท์หมายถึงสิ่งที่เกิดขึ้นในระบบ อีเวนท์ในระบบมีอยู่ 33 ชนิด ได้แก่

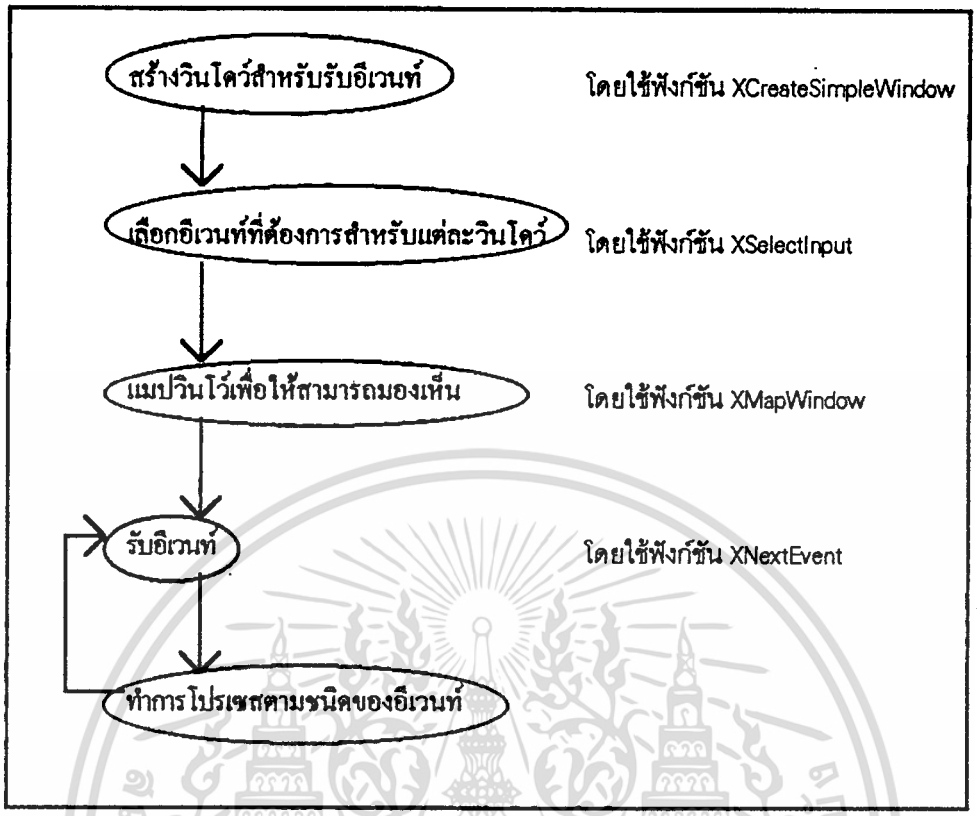
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Event Number	Event Type	Event Number	Event Type
2	KeyPress	18	UnmapNotify
3	KeyRelease	19	MapNotify
4	ButtonPress	20	MapRequest
5	ButtonRelease	21	ReparentNotify
6	MotionNotify	22	ConfigureNotify
7	EnterNotify	23	ConfigureRequest
8	LeaveNotify	24	GravityNotify
9	Focusin	25	ResizeRequest
10	FocusOut	26	CirculateNotify
11	KeymapNotify	27	CirculateRequest
12	Expose	28	PropertyNotify
13	GraphicsExpose	29	SelectionClear
14	NoExpose	30	SelectionRequest
15	VisiblityNotify	31	SelectionNotify
16	CreateNotify	32	ColormapNotify
17	DestroyNotify	33	ClientMessage
		34	MappingNotify

ตาราง 2.3 ชนิดของอีเวนท์

แต่เนื่องจากในแต่ละโปรแกรมไม่จำเป็นต้องใช้อีเวนท์ทั้งหมดที่มีอยู่ในระบบ ดังนั้นวินโดว์แต่ละตัวในโปรแกรมจึงสามารถกำหนดให้รับทราบเฉพาะเมื่อมีอีเวนท์ที่ต้องการก็ได้ และเมื่อมีอีเวนท์ที่กำหนดไว้เข้ามาแล้วก็นำไปดำเนินการทำงานต่อไป เมื่อเสร็จแล้ว โปรแกรมก็จะรอจนกว่าจะมีอีเวนท์ที่ต้องการตัวใดตัวหนึ่งเข้ามาอีก เช่นนี้ไปเรื่อยๆ โดยแสดงให้เห็นการเขียนโปรแกรมได้ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2.1 แสดงการเขียนโปรแกรมแบบอีเวนท์ไดรเวน

2.2 การย่อ - ขยายขนาดข้อมูล

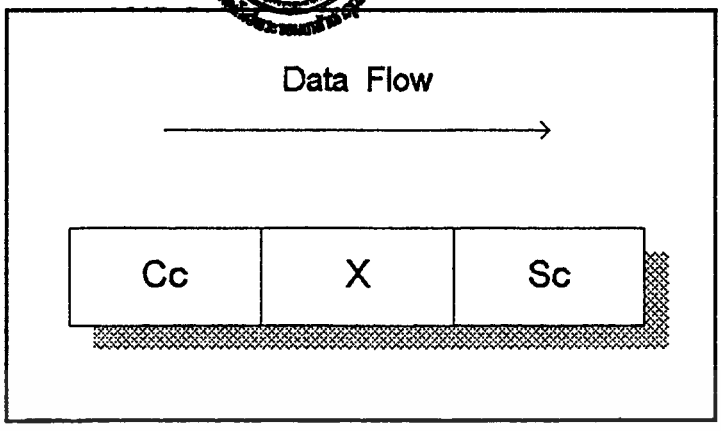
2.2.1 การย่อ - ขยายข้อมูลโดยวิธี Run - Length Encoding (RLE)

2.2.1.1 การย่อข้อมูลโดยวิธี RLE

Run-Length Encode (RLE) คือวิธีการย่อข้อมูลโดยใช้วิธีการลดจำนวนของตัวอักษรที่เกิดขึ้นซ้ำๆกันเป็นช่วง ในกรณีที่ตัวอักษรเป็น Null Character การย่อข้อมูลจะใช้ตัวอักษรพิเศษแสดงแทน Null Character ที่เกิดขึ้น

กระบวนการย่อข้อมูลแบบ RLE จะให้ผลลัพธ์ออกมาในรูปของข้อมูลที่ได้ทำการย่อแล้วดังรูปที่ 3.10 ซึ่งจะเห็นว่าเราใช้ตัวอักษร 3 ตัว แสดงแทนข้อมูลที่ถูกย่อ ดังนั้นการย่อข้อมูลแบบ RLE นี้จะใช้ได้อย่างมีประสิทธิภาพก็ต่อเมื่อข้อมูลมีการซ้ำกันของตัวอักษรตั้งแต่ 4 ตัวอักษรขึ้นไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2.2 การแทนตัวอักษรที่ถูกย่อข้อมูลแบบ RLE

โดยมีสัญลักษณ์ต่างๆดังนี้

Sc = ตัวอักษรพิเศษที่แสดงถึงการย่อข้อมูล

X = ตัวอักษรที่เกิดการซ้ำของข้อมูล

Cc = จำนวนตัวอักษรที่มีการซ้ำ

ตัวอย่างของการย่อข้อมูลด้วยวิธี RLE ได้แสดงไว้ในตาราง 2.4 โดยมี Sc เป็นตัวอักษรพิเศษที่ใช้บอกถึงกระบวนการย่อข้อมูลที่เกิดขึ้น และใช้อักษร B แสดงแทน Blank Character

Original Data String	Encoded Data String
$******55.72	$Sc*655.72
aaaaaaaaa	Sca9
GunsBBBBBBBBBBButter	GunsScB10Butter

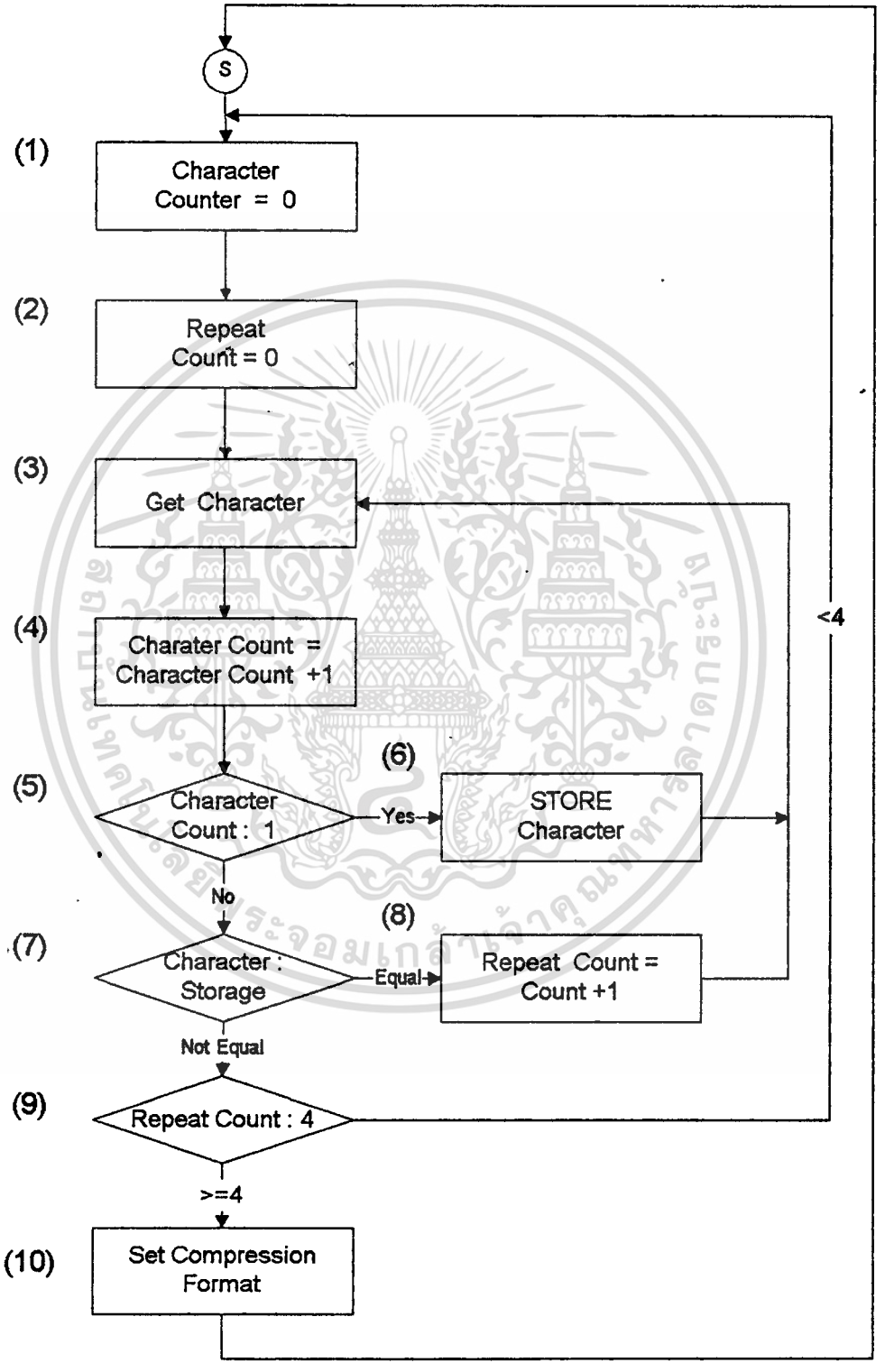
ตาราง 2.4 ตัวอย่างการย่อข้อมูลด้วยวิธี RLE

ขั้นตอนของกระบวนการย่อข้อมูลมีลักษณะตามรูป 2.3 โดยสามารถที่จะอธิบายถึงขั้นตอนต่างๆ ได้ดังนี้ เริ่มแรก ตัวแปร Character Count (1) และ Repetition Character (2) จะถูกกำหนดค่าเริ่มต้นให้เป็น 0 ในขั้นต่อมาโปรแกรมจะอ่านตัวอักษรจากแหล่งข้อมูล (3) ซึ่งทำให้ตัวแปร Character Count ถูกเพิ่มค่าขึ้น (4) ตัวแปรตัวนี้จะถูกเปรียบเทียบกับค่า '1' ซึ่งการทำงานในรอบแรกผลของการเปรียบเทียบจะเป็นจริงเสมอ ทำให้ตัวอักษรที่อ่านเข้ามาจะถูกนำไปเก็บไว้ใน บัฟเฟอร์ (Buffer) (6) การทำซ้ำในรอบต่อๆไปของการเปรียบเทียบกับค่า '1' ตัวอักษรจะถูกอ่านเข้ามา และ ถูกนำมาเทียบกับตัวอักษรที่เก็บไว้ในบัฟเฟอร์ (7) ถ้ามีค่าเท่ากัน ค่าของตัวแปร Repeat Counter จะถูกเพิ่มค่าขึ้นอีก 1 (8) และโปรแกรมจะอ่านตัวอักษรตัวใหม่เข้ามา (3) ในทางกลับกันถ้าตัวอักษรที่อ่านเข้ามามีค่าไม่ตรงกับตัวอักษรที่อยู่ใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

035001

บัฟเฟอร์ (7) ตัวแปร Repeat Counter จะถูกเปรียบเทียบกับ '4' (9) ถ้ามีค่าน้อยกว่า '4' ก็จะไม่มีการย่อข้อมูล แต่ถ้า Repeat Counter มีค่ามากกว่าหรือเท่ากับ '4' (9) กระบวนการย่อข้อมูลก็จะเกิดขึ้น (10)

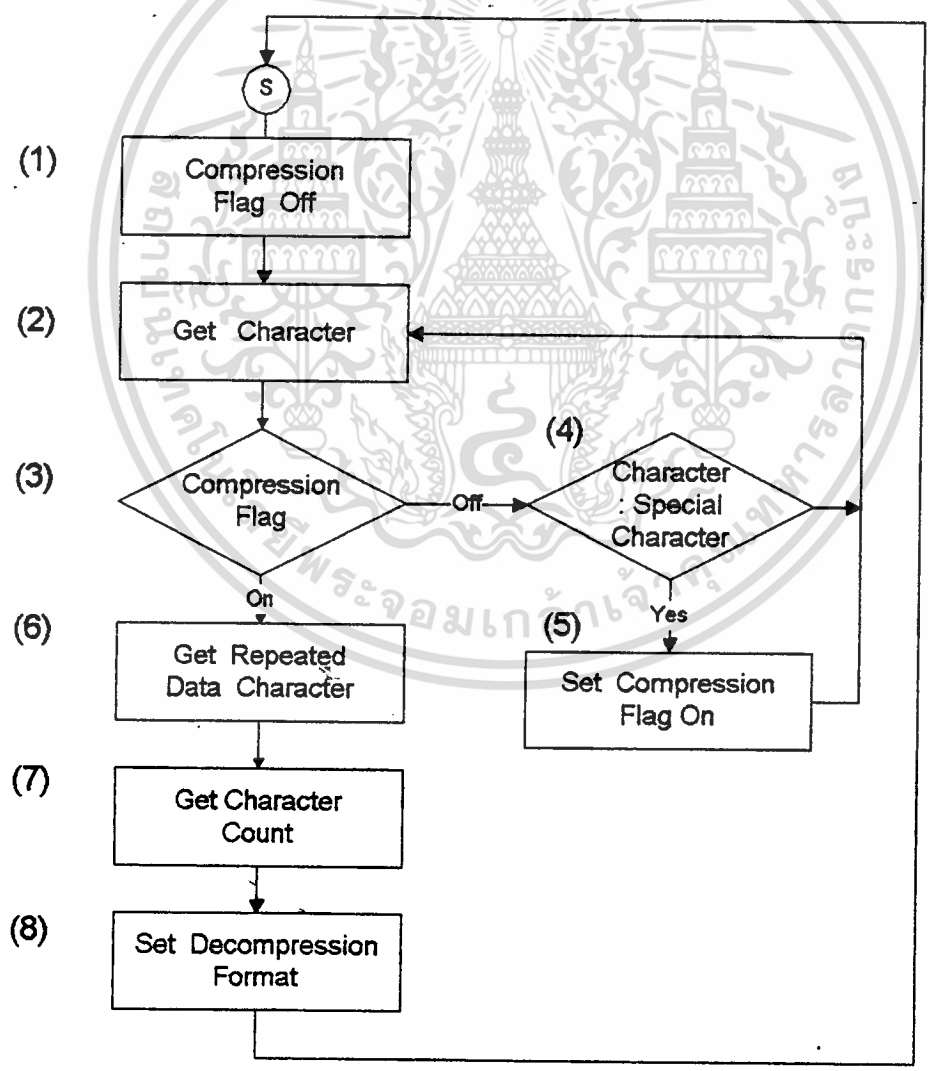


รูป 2.3 ขั้นตอนการเข้ารหัสโดยวิธี RLE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1.2 การขยายข้อมูลโดยใช้วิธี RLE

กระบวนการในการถอดรหัสแบบ RLE ได้ถูกแสดงไว้ดังรูป 2.4 ในขั้นแรก ตัวแปร Compression Flag จะถูกกำหนดสถานะให้เป็นค่าปิดอยู่ (1) โปรแกรมจะเริ่มอ่านข้อมูลที่ถูกย่อไว้จากแหล่งข้อมูล (2) ถ้า Compression Flag มีสถานะปิดอยู่ (3) ตัวอักษรที่อ่านเข้ามาจะถูกเปรียบเทียบกับตัวอักษรพิเศษที่แสดงถึงการย่อข้อมูล เช่น Sc เพื่อตรวจสอบว่าเป็นข้อมูลที่ถูกย่อไว้หรือไม่ (4) ถ้าตรวจไม่พบตัวอักษรพิเศษ โปรแกรมก็จะอ่านข้อมูลตัวต่อไป (2) ในทางกลับกัน ถ้ามีการตรวจพบตัวอักษรพิเศษนี้ ตัวแปร Compression Flag จะถูกกำหนดให้มีสถานะเปิด (5) แล้วโปรแกรมจะอ่านข้อมูลเข้ามาใหม่ (2) ในรอบการทำงานต่อมา ซึ่งตัวแปร Compression Flag มีสถานะเป็นเปิดอยู่ (3) ตัวอักษรที่อ่านเข้ามาก็จะเป็นตัวอักษรที่มีการซ้ำกันของข้อมูลเกิดขึ้น ส่วนตัวถัดไป (7) ก็คือข้อมูลจำนวนตัวอักษรที่มีการซ้ำกัน ซึ่งเมื่อข้อมูลทั้งสามแบบได้รับเข้ามาแล้ว กระบวนการ การถอดรหัสก็จะเริ่มขึ้น (8)



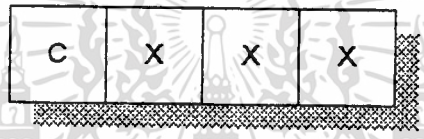
รูป 2.4 ขั้นตอนการขยายข้อมูลโดยวิธี RLE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1.3 การประยุกต์ใช้ การย่อข้อมูลแบบ RLE

การย่อข้อมูลด้วยวิธี RLE มีการนำไปประยุกต์กันอย่างแพร่หลาย แต่แนวทางที่นิยมกันมากที่สุดก็คือ การย่อข้อมูลแบบ MNP (Microcom Network Protocol) Class 5 ซึ่งใช้กันทางใน MODEM

การย่อข้อมูลแบบ MNP Class 5 เกิดจากการผสมผสาน ระหว่างเทคนิคการย่อข้อมูลสองแบบ คือ RLE และ Adaptive-Frequency Encoding วิธีการย่อข้อมูลแบบ MNP 5 จะใช้การส่งข้อมูลของตัวอักษรที่มีการซ้ำกัน สาม ตัวแล้วตามด้วยจำนวนของตัวอักษรที่เกิดการซ้ำกัน ดังรูปที่ 3.14 ซึ่งจะเห็นว่าถ้ามีการซ้ำกันของข้อมูลน้อยกว่า สาม ตัวแล้ว ก็จะไม่มีการใส่ตัวเลขเข้าไปด้วย ในขณะที่มีการซ้ำกัน สามตัวพอดี เราก็จะเติมเลข 0 เข้าไปแทน ซึ่งการย่อข้อมูลแบบ MNP 5 นี้เราจะกำหนดขอบเขตของตัวเลขไว้ที่ 250 ตัวอักษรเท่านั้น



C = จำนวนตัวอักษร

X = ตัวอักษรที่ซ้ำ

riginal Data Stream	Compressed Data Stream
A	A
BB	BB
CCC	0CCC
DDDD	4DDD
EEEEE	5EEE

รูป 2.5 วิธีย่อข้อมูลแบบ MNP 5

ข้อดีของการย่อข้อมูลแบบ RLE นี้ก็คือไม่จำเป็นที่จะต้องมี ตัวอักษรพิเศษที่บอกถึงการย่อข้อมูล ส่วนข้อเสียก็คือ การย่อข้อมูลแบบนี้ มีโอกาสที่จะทำให้ข้อมูลมีความยาวมากขึ้น เช่นในกรณี ที่มีการซ้ำของข้อมูลเพียง สาม ตัวเท่านั้น

2.2.2 การย่อ - ขยายข้อมูลด้วยวิธี Huff-Man Compression

การเข้ารหัสด้วยวิธี Huff-Man จัดเป็นวิธีการย่อข้อมูลโดยให้กรรมวิธีการทาง สถิติ เข้ามา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ช่วย ซึ่งจะทำให้สามารถลดจำนวนข้อมูลโดยเฉลี่ยลงไปได้

2.2.2.1 การเข้ารหัสแบบ Huff-Man

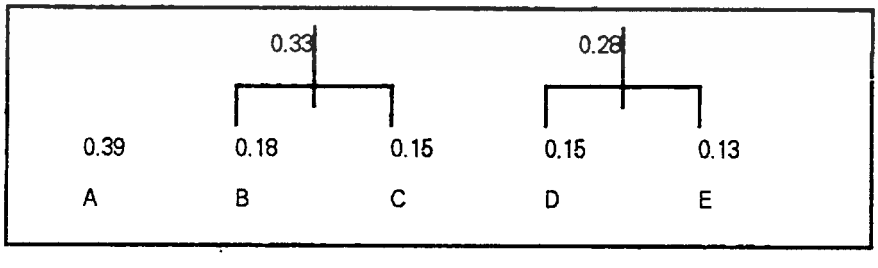
คุณสมบัติการเข้ารหัสด้วยวิธี Huff-Man ที่พิเศษไปกว่าการเข้ารหัสแบบอื่นๆก็คือ กลุ่มของรหัสตัวที่มีความยาวสั้นๆ จะไม่มีการซ้ำกับรหัสที่เป็นส่วนเริ่มต้นของ กลุ่มรหัสที่มีความยาวกว่า ตัวอย่างเช่น ตัวอักษร 'A' ถูกเข้ารหัสด้วยวิธี Huff-Man ได้ผลลัพธ์ออกมาเป็นเลขฐานสอง '100' เพราะฉะนั้นการเข้ารหัสในตัวอักษรอื่นๆจะไม่มีการเริ่มต้นด้วยรหัส '100' อีก เช่น '10001' เป็นต้น

ต่อไปนี้เป็นการแสดงการเข้ารหัสด้วยวิธี Huff-Man โดยดูได้จากรูปที่ 2.6 ขั้นแรกเริ่มจากการนำตัวอักษรที่ต้องการเข้ารหัสมาจัดลำดับตามความถี่ในการใช้ในข้อมูลที่ต้องการจะเข้ารหัสดังรูป

ความถี่	15	7	6	6	5
ตัวอักษร	A	B	C	D	E

รูป 2.6 แสดงความถี่ของตัวอักษรในไฟล์ที่ต้องการย่อขนาดโดยวิธี Huff-Man

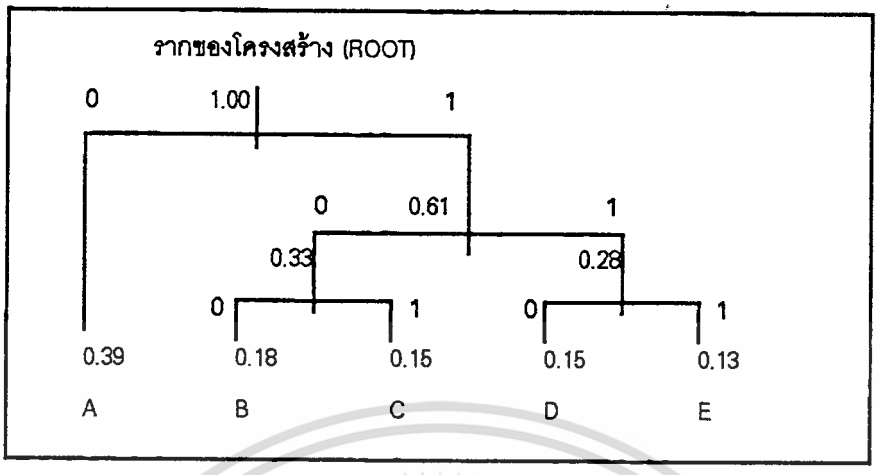
แต่โดยทั่วไปเรามักใช้ค่าความถี่เป็นทศนิยม โดยเอาความถี่ในการใช้งานของแต่ละตัวอักษรหารด้วยจำนวนตัวอักษรในข้อมูลทั้งหมด ก็จะได้ค่าความถี่เป็นทศนิยมแทน ในขั้นตอนต่อไปเป็นการสร้างโครงสร้างต้นไม้เพื่อใช้ในการกำหนดรหัสให้กับตัวอักษรแต่ละตัว ในกลุ่มอักษรเหล่านี้ เราจะนำตัวอักษรสองตัวที่มีค่าความถี่ต่ำที่สุดมาจับคู่ และบวกค่าความถี่ทั้งสองเข้าด้วยกัน ซึ่งในรูปนั้นก็คือ คู่ของ D,E ได้ผลบวกค่าความถี่เท่ากับ 0.28 ผลลัพธ์ที่ได้จะถูกนำไปเปรียบเทียบกับค่าความถี่ที่เหลือ แล้วจับคู่กับค่าที่ต่ำที่สุด และบวกเข้าด้วยกัน ซึ่งจากรูปที่ 2.6 ก็คือ B,C ได้ค่าผลบวกเท่ากับ 0.33 ทำเช่นนี้ไปเรื่อยๆจนได้ค่าผลบวกค่าเดียวคือจำนวนตัวอักษรทั้งหมดในข้อมูลนั้นดังรูป 2.8



รูป 2.7 ตัวอย่างการสร้างโครงสร้างต้นไม้จากความถี่ของข้อมูล

จากนั้นก็กำหนดค่าของ 0 และ 1 ให้กับแต่ละโหนด (Node)ดังรูป และ การไล่รหัสเราจะเริ่มจากรากของโครงสร้างต้นไม้ก่อน ก่อนแล้วจึงไล่ต่อไปจนถึง ตัวอักษรแต่ละตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2.8 ตัวอย่างการกำหนดรหัสจากโครงสร้างต้นไม้

จะได้รหัสของตัวอักษรแต่ละตัวดังนี้

A : 0

B : 100

C : 101

D : 110

E : 111

จำนวนบิทเฉลี่ยที่ใช้ในการเข้ารหัสสามารถคำนวนได้จาก การคูณค่าความยาวของบิทที่ให้กับค่าความถี่เข้าด้วยกัน ดังต่อไปนี้

1*0.39 + 3*0.18 + 3*0.15 + 3*0.15 + 3*0.13 หรือเท่ากับ 2.23 บิทต่อตัวอักษร

จำนวนบิทที่ใช้ในการเข้ารหัสคำนวนได้จากสูตรดังต่อไปนี้

$b = f(-\log_2 P)$

โดย:

P=ค่าความถี่เป็นทศนิยม

f(x)=ฟังก์ชั่นที่ให้ค่าจำนวนเต็มที่มีค่ามากกว่าหรือเท่ากับ X

เช่นค่าความน่าจะเป็นของตัวอักษร 'E' เท่ากับ 0.13 จะได้จำนวนบิทของตัวอักษรเท่ากับ f(2.94) เท่ากับ '3' ตัวอักษร

อย่างไรก็ตามในบางโอกาส ในขั้นตอนการเข้ารหัสจะเกิดเหตุการณ์ที่โหนดทั้ง 2 หรือ มากกว่ามีความถี่เท่าๆกัน ซึ่งจะทำให้ต้องมีการตัดสินใจว้าจะใช้โหนดใดก่อนดีกว่ากัน แต่อย่างไรก็ตามผลปรากฏออกมาว่า ไม่ว่าจะเลือกวิถีทางใดก็ตาม เราจะได้ค่าความยาวเฉลี่ยต่อตัวอักษรออกมาเท่ากันทั้งคู่

จากตัวอย่างที่แสดงให้เห็นการกำหนดค่ารหัสของตัวอักษรในข้อมูล การย่อข้อมูลวิธีนี้จะเสียเวลาในการหาค่าความถี่ของตัวอักษรแต่ละตัวและการสร้างโครงสร้างต้นไม้และขั้นตอนในการกำหนดค่ารหัสจากโครงสร้างต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม้ จากนั้นจึงจะนำรหัสที่ได้มาแทนตัวอักษรในไฟล์ที่ต้องการย่อข้อมูล และในขั้นตอนการขยายข้อมูลจำเป็นต้องใช้โครงสร้างต้นไม้ที่สร้างขึ้นในขั้นตอนการย่อข้อมูลเพื่อค้นหาตัวอักษรมาแทนรหัสที่มีอยู่ในไฟล์ที่ถูกย่อแล้ว ดังนั้นโครงสร้างต้นไม้จึงจะต้องถูกเก็บไว้ในไฟล์ที่ย่อข้อมูลไว้ด้วย

2.2.2.2 การขยายข้อมูลแบบ Huff-Man

การขยายข้อมูล มีขั้นตอนที่ไม่ยุ่งยากซับซ้อนเท่ากับกับการย่อข้อมูลเพราะเนื่องจากโครงสร้างต้นไม้ที่ใช้ในการกำหนดรหัสแทนแต่ละตัวอักษรได้ถูกเก็บรวมไว้ในไฟล์ที่ย่อไว้ ในขั้นตอนการขยายข้อมูลจึงต้องอ่านโครงสร้างต้นไม้นั้นออกมา และอ่านรหัสจากอินพุทไฟล์เข้ามาทีละหนึ่งบิทแล้วใช้ค่าของบิทนั้นเป็นตัวกำหนดตำแหน่งที่จะเดินทางในโครงสร้างต้นไม้ว่าซ้ายหรือขวา (เช่น ถ้าเป็น 1 ให้ไปทางซ้าย ถ้าเป็น 0 ให้ไปทางขวา) และทุกครั้งที่เดินทางต่อไปในโครงสร้างต้นไม้จะต้องดูว่าถึงโหนดที่เป็นตัวอักษรหรือยัง เมื่อถึงโหนดที่เป็นตัวอักษรแล้วก็อ่านตัวอักษรที่โหนดนั้นเก็บลงในเอาท์พุทไฟล์ จากนั้นเริ่มอ่านบิทต่อไปเข้ามาและเริ่มเดินทางจากรากของโครงสร้างต้นไม้ใหม่ ทำเช่นนี้ไปเรื่อยๆจนหมดบิทในอินพุทไฟล์

2.2.3 การย่อ - ขยายข้อมูลตามวิธีของ Ziv and LemPel โดย Terry Welch (LZW)

ในปี คศ. 1980 การย่อขนาดข้อมูลที่เป็นที่แพร่หลายในขณะนั้นล้วนแต่เป็นการใช้เทคนิคทางด้านสถิติ (Statistical Modeling) แต่ในปี คศ. 1977 และ 1978 ได้มี Jacob Ziv และ Abraham Lempel ได้นำหลักการที่เรียกว่า Dictionary -based Model มาดัดแปลงทำให้เกิดปรากฏการณ์ทางด้านการย่อข้อมูลที่ได้ผลน่าประทับใจอีกอันหนึ่ง

LZ77 เป็นการย่อขนาดข้อมูลแบบแรกที่ได้เสนอขึ้นโดยมีหลักการคือ ดิคชันเนรีที่ใช้มีขนาดคงที่ของวินโดว์และบรรจุอินพุทสตรีมก่อนหน้านั้นไว้ ในขณะที่มีการนำอินพุทสตริงที่ถูกอ่านเข้ามาใหม่มาเทียบกับอินพุทสตรีมที่เก็บไว้ถ้าพบว่ามีตรงกันอินพุทสตริงนั้นจะถูกเก็บไว้เป็นตัวชี้ ซึ่งจากหลักการนี้ทำให้ได้ผลการย่อข้อมูลที่ดี อีกทั้งง่ายต่อการเขียนโปรแกรมเพื่อใช้งานจริงด้วย

LZ78 มีประเด็นที่แตกต่างที่ใช้กับการสร้างดิคชันเนรี โดยแทนที่จะจำกัดขนาดวินโดว์ของอินพุทสตรีมก่อนหน้านั้นไว้ ใน LZ78 จะสร้างดิคชันเนรีเพื่อเก็บสตริงที่เข้ามาก่อนหน้านั้นไว้ทั้งหมด แต่แทนที่ดิคชันเนรีจะเข้าถึงทุกๆอักขระของสตริงใดๆในก่อนหน้านั้น ดิคชันเนรีจะสร้างเพียงหนึ่งอักขระต่อครั้งเท่านั้น ซึ่งผลที่ได้จาก LZ78 นั้นดีกว่า LZ77 ทั้งด้านอัตราการย่อข้อมูลและขนาดความยาวของสตริงที่ย่อได้ ทำให้หลักการนี้เป็นที่ยอมรับและแพร่หลาย ทำให้ Terry Welch นำมาดัดแปลงและให้ชื่อว่า LZW

2.2.3.1 หลักการย่อขนาดข้อมูลแบบ LZW

ขั้นตอนในการย่อข้อมูลแบบ LZW เป็นการทำงานที่ไม่ซับซ้อน กล่าวคือเอาท์พุทที่ได้จากการย่อจะประกอบไปด้วยรหัสที่มาแทนที่อินพุทสตริงที่เคยผ่านเข้ามาในระบบแล้ว และเมื่อใดก็ตามที่ระบบให้เอาท์พุทเป็นรหัสที่ไม่เคยมีมาก่อนออกมา สตริงที่ถูกแทนด้วยรหัสนั้นจะถูกเพิ่มไว้ในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
old_string[0] = getc( input );
old_string[1] = '\0';
while ( !feof(input) ) {
        character = getc( input );
        strcpy( new_string , old_string );
        strncat( new_string , &character , 1 );
        if( in_dictionary(new_string) )
                strcpy( old_string , new_string )
        else {
                code = look_up_dictionary( old_string );
                output_code( code );
                add_to_dictionary( new_string );
                old_string[0] = character;
                old_string[1] = '\0';
        }
}
code = look_up_dictionary( old_string );
output_code( code );
```

รูป 2.9 ตัวอย่างการเขียนโปรแกรมในการย่อข้อมูลแบบ LZW

ตัวอย่างการทำงานตามหลักการที่กล่าวมา โดยให้อินพุทสตรีมข้ามาเป็นภาษาอังกฤษ หลังจากเข้ามาลูปแรกจะได้สตริงคือ " W" และจะไม่พบสตริงนี้ในตารางสตริงที่เก็บไว้ในระบบ ดังนั้นเอาท์พุทรหัสจะเป็น ' ' และสตริง " W" จะถูกเก็บไว้ในตารางสตริง เนื่องจากรหัสที่เก็บไว้ในดิคชันเนรีจะเก็บรหัส 0 - 255 เอาไว้เพื่อแทนอักขระ 256 ตัว สตริงแรกที่ถูกเก็บไว้ในตารางสตริงจะมีค่ารหัส 256 หลังจากนั้นอินพุทตัวต่อไปจะถูกอ่านเข้ามา "E" และจะได้สตริงเก็บในตารางสตริงเป็นตัวที่สองคือ "WE" และเอาท์พุทรหัสจะเป็นรหัสของ 'W' เมื่อมาถึงคำที่สองในอินพุทสตรีม อักขระที่อ่านเข้ามาจะเป็น ' ' และ 'W' ซึ่งตรงกับรหัสในตารางสตริง ดังนั้นจะให้เอาท์พุทรหัสเป็น 256 และสตริง " WE" จะถูกเก็บไว้ในตารางสตริงต่อไป ทำเช่นนี้ไปเรื่อยๆ จนหมดอินพุทสตรีม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Input String : " WED WE WEE WEB WET"

Characters Input	Code Output	New code value and associated string
" W"	" "	256 = " W"
"E"	'W'	257 = "WE"
"D"	'E'	258 = "ED"
" "	'D'	259 = "D "
"WE"	256	260 = " WE"
" "	'E'	261 = "E "
"WEE"	260	262 = " WEE"
" W"	261	263 = "E W"
"EB"	257	264 = "WEB"
" "	'B'	265 = "B "
"WET"	260	266 = " WET"
<EOF>	'T'	

รูป 2.10 ตัวอย่างการสร้างตารางสตริงและการย่อข้อมูลแบบ LZW

จากตัวอย่างข้างต้นจะแสดงเอาท์พุทที่ได้และตารางสตริงที่สร้างขึ้นในขณะที่อินพุทสตริงถูกอ่านเข้ามา จะเห็นว่าตารางสตริงจะมีขนาดเพิ่มขึ้นอย่างรวดเร็วเพราะว่าจะมีสตริงใหม่ๆถูกเพิ่มเข้ามาทุกครั้งที่มีการเอาท์พุทรหัสออกไป จากตัวอย่างอินพุทที่ให้มานั้นมีความซ้ำซ้อนของสตริงมากเราจะได้เอาท์พุทที่เป็นรหัส 5 ตัวและเอาท์พุทที่เป็นตัวอักษร 7 ตัว ถ้าเราใช้รหัสที่มีขนาด 9 บิต จะได้ 13.5 ไบท์เอาท์พุทสตริง

2.2.3.2 การขยายข้อมูลโดยวิธี LZW

การย่อและขยายข้อมูลแบบ LZW นี้มีข้อดีอีกข้อหนึ่งคือเราไม่ต้องเก็บดิคชันเนรีหรือตารางสตริงตามตัวอย่างเข้ามาในเอาท์พุทที่ได้จากการย่อขนาดด้วย เพราะตารางดังกล่าวสามารถสร้างให้ตรงกันได้จากขั้นตอนการขยายข้อมูลโดยใช้ข้อมูลที่ย่อแล้ว เทคนิคนี้สามารถทำได้เพราะในขั้นตอนการย่อข้อมูลจะให้เอาท์พุทที่เป็นตัวอักษรออกมาก่อนที่จะนำไปเก็บไว้ในตารางสตริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
old_string[0] = input_bits();
old_string[1] = '\0';
putc( old_string[0] , output );
while ( (new_code = input_bits() ) != EOF ) {
        new_string = dictionary_lookup( new_code );
        fputs( new_string , output  );
        append_char _to_string( old_string , new_string[0] );
        add_to_dictionary( old_string );
        strcpy( old_string , new_string );
}
```

รูป 2.11 ตัวอย่างการเขียนโปรแกรมในการขยายข้อมูลแบบ LZW

ขั้นตอนข้างต้นนี้ ทำงานคล้ายกับขั้นตอนที่ใช้ในการย่อข้อมูล โดยจะเก็บสตริงที่เข้ามาใหม่ลงในตารางสตริงทุกครั้งที่พบสตริงใหม่ หรือพูดง่ายๆ ก็คือการแปลรหัสอินพุทที่เข้ามาให้เป็นสตริงใช้เป็นเอาท์พุทต่อไป จากเอาท์พุทที่ได้จากการย่อข้อมูลเรานำมาพิจารณาในการขยายข้อมูลจะได้ดังนี้

Input Codes : " WED<256>E<260><261><257>B<260>T"

Input / New_code	Old_code	String / Output	Character	New table entry
' '	' '	" "		
'W'	' '	"W"	'W'	256 = " W"
'E'	'W'	"E"	'E'	257 = "WE"
'D'	'E'	"D"	'D'	258 = "ED"
256	'D'	" W"	' '	259 = "D "
'E'	256	"E"	'E'	260 = " WE"
260	'E'	" WE"	' '	261 = "E "
261	260	"E "	'E'	262 = " WEE"
257	261	"WE"	'W'	263 = "E W"
'B'	257	"B"	'B'	264 = "WEB"
260	'B'	" WE"	' '	265 = "B "
'T'	260	"T"	'T'	266 = " WET"

รูป 2.12 ตัวอย่างการสร้างตารางสตริงและการขยายข้อมูลแบบ LZW

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตัวอย่างที่แสดงให้เห็นเป็นตัวอย่างที่ง่ายๆ แต่จริงๆแล้วการย่อข้อมูลแบบ LZW นี้อาจทำให้เกิดปัญหาได้ในขั้นตอนการขยายข้อมูล แต่ละครั้งในการย่อข้อมูลเพิ่มสตริงลงในตารางสตริงจะเกิดขึ้นก่อนที่ทุกตัวอักษรของสตริงนั้นจะเป็นเอาท์พุทออกไป ถ้าหากมีการใช้รหัสของสตริงนั้นเป็นเอาท์พุทในขณะที่ยังไม่ได้มีการเอาท์พุทสตริงนั้นออกไปจนครบจะทำให้เกิดปัญหาในขั้นตอนการขยายข้อมูลเพราะไม่สามารถขยายข้อมูลที่ยังไม่ได้อยู่ในตารางสตริงได้ ซึ่งเหตุการณ์เช่นนี้อาจเกิดขึ้นได้จริงๆในอินพุทสตริง นั่นคือในอินพุทสตริงของการขยายข้อมูลจะประกอบไปด้วยลำดับของ ตัวอักษร , รหัส , ตัวอักษร , รหัส , ... เช่นนี้ไปเรื่อยๆทำให้ไม่สามารถทราบถึงสตริงในแต่ละรหัสที่มีอยู่ได้ ตัวอย่างเช่น

Input String : ...IWOMBAT.....IWOMBATIWOMBATIX.....

สมมุติให้ IWOMBAT ได้ถูกเก็บในตารางสตริงและมีรหัสเป็น 300 พิจารณาขั้นตอนการย่อข้อมูล

Character Input	New code value and associated string	Code Output
.....	.....	.....
WOMBATA	300 = IWOMBAT	288 (IWOMBA)
.	.	.
.	.	.
.	.	.
WOMBATI	400 = IWOMBATI	300 (IWOMBAT)
WOMBATIX	401 = IWOMBATIX	400 (IWOMBATI)

ในการขยายข้อมูลเมื่อถึงรหัส 300 ในตารางสตริงจะเก็บสตริงถึงรหัส 399 และเมื่อได้อ่านอินพุทรหัสต่อไปคือ 400 จะไม่พบว่ามีสตริงรหัสดังกล่าวในตารางสตริง

เราสามารถดัดแปลงขั้นตอนการขยายข้อมูลเพื่อจัดการกับรหัสที่ไม่มีในตารางสตริงได้ จากตัวอย่างข้างต้นเมื่ออินพุทเป็นรหัส 400 ซึ่งไม่มีในตารางสตริง โปรแกรมจะกลับไปที่รหัสก่อนหน้านั้นในอินพุทสตริงซึ่งก็คือ 300 และหมายถึงสตริง "IWOMBAT" ซึ่งตัวอักษรตัวแรกของสตริงจะเป็นตัวอักษรสุดท้ายของสตริงในรหัส 400 นั่นคือ รหัส 400 จะหมายถึงสตริง "IWOMBATI"

จากที่กล่าวมาทั้งด้านการย่อและขยายข้อมูลแบบ LZW เป็นการพูดถึงแต่เพียงพื้นฐานเท่านั้นซึ่งตัวอย่างโปรแกรมและการทำงานก็เป็นเพียงกรณีที่ง่ายๆเท่านั้นหากแต่การใช้งานจริงสิ่งที่ต้องคำนึงถึงมากที่สุดคือการจัดการเกี่ยวกับดิคชันเนรี ซึ่งมักจะนำหลักการเกี่ยวกับ อาร์เรย์ (Array) , แฮชฟังก์ชัน (Hash function) , แสตก (Stack) มาใช้ด้วย

2.3 การทำงานแบบแบคกราวน์โปรเซสในระบบหลายผู้ใช้

2.3.1 คำจำกัดความพื้นฐานบนยูนิกซ์

ก่อนจะกล่าวถึงการสร้างแบคกราวน์โปรเซส เราควรทำความรู้จักกับคำจำกัดความต่างๆในระบบยูนิกซ์ที่เกี่ยวข้องกับการใช้งานโปรเซสก่อน ซึ่งได้แก่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.1.1 เคอเนล (Kernel)

เคอเนล เป็นระบบปฏิบัติการ (Operating System) ซึ่งเคอเนลจะทำหน้าที่ให้บริการเกี่ยวกับระบบการจัดการไฟล์ (Filesystem) การจัดการหน่วยความจำ (Memory Management) การจัดเวลาทำงานของซีพียู และ อุปกรณ์อินพุท เอาท์พุท (CPU and I/O Scheduling)

2.3.1.2 โปรแกรม (Program)

โปรแกรม หมายถึงไฟล์ที่สามารถทำงานได้ (Executable file) และในระบบยูนิกซ์นี้การที่จะเรียกไฟล์มาใช้งานนั้นสามารถทำได้เพียงวิธีเดียวคือใช้ คำสั่งระบบ (System Call) เอกซ์เซค (exec) ซึ่งจะพูดถึงรายละเอียดต่อไป

2.3.1.3 โปรเซส (Process)

โปรเซส ก็คือ โปรแกรมที่กำลังทำงานอยู่ในระบบ และวิธีที่สามารถจะสร้างโปรเซสใหม่ในระบบได้ก็ด้วยการใช้ คำสั่งระบบ ฟอร์ค (fork) บางครั้งเราจะพบคำว่า ทาส์ค (task) หมายถึงโปรเซสนั่นเอง ในระบบยูนิกซ์เป็นระบบที่สามารถมีโปรเซสทำงานอยู่ในระบบได้พร้อมๆกันหลายโปรเซส จึงเรียกว่าเป็น ระบบปฏิบัติการแบบมัลติโปรเซสซิง หรือมัลติทาส์ค และสำหรับโปรแกรมหนึ่งๆก็สามารถจะทำงานพร้อมๆกันเป็นหลายโปรเซสได้ในเวลาเดียวกัน ตัวอย่างเช่น อาจมีผู้ใช้หลายคนกำลังใช้งานโปรแกรมอิดิเตอร์เดียวกันอยู่ก็ได้

โดยทั่วไปโปรเซสแต่ละตัวที่ทำงานในระบบจะประกอบไปด้วย

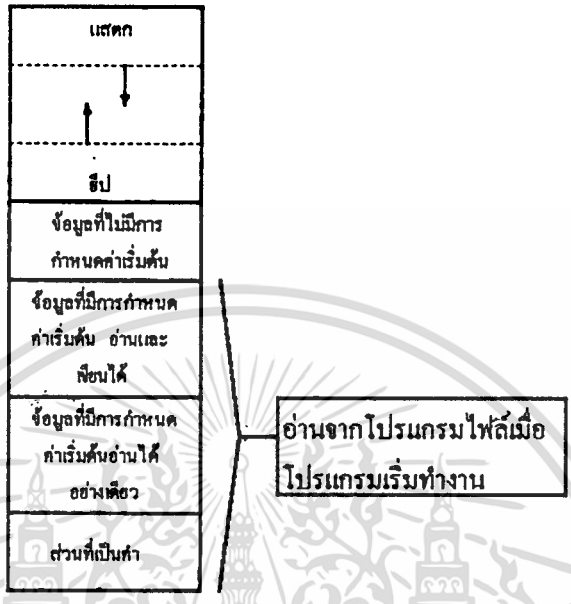
1. ส่วนที่เป็นคำสั่งที่ใช้ในการทำงานของโปรเซส (Machine Instructions) ซึ่งโปรเซสจะใช้ในการทำงานตามคำสั่งที่มีอยู่นี้ เมื่อโปรเซสเริ่มการทำงานขึ้นในระบบส่วนของคำสั่งนี้จะถูกอ่านเข้ามาสู่หน่วยความจำในระบบหรือถ้าหากระบบนั้นสามารถอนุญาตให้มีการทำงานของโปรแกรมได้เป็นหลายโปรเซสในขณะเดียวกันอาจมีการใช้ส่วนของคำสั่งนี้ร่วมกันระหว่างโปรเซสเหล่านั้นได้
2. ส่วนที่เป็นข้อมูล (Data) ซึ่งประกอบไปด้วยข้อมูลที่กำหนดไว้ในโปรแกรม อาจแบ่งได้เป็น

- ข้อมูลที่สามารถอ่านได้อย่างเดียวแบบมีการกำหนดค่าเริ่มต้น ซึ่งเป็นข้อมูลที่โปรแกรมได้กำหนดค่าไว้ตั้งแต่ตอนต้นในการทำงานและในขณะที่ทำงานค่าของข้อมูลชนิดนี้จะไม่สามารถเปลี่ยนแปลงได้
- ข้อมูลที่สามารถอ่านและเขียนได้แบบมีการกำหนดค่าเริ่มต้น หมายถึงข้อมูลที่โปรแกรมกำหนดค่าเริ่มต้นไว้และค่าของข้อมูลดังกล่าวสามารถเปลี่ยนแปลงได้ขณะที่โปรแกรมทำงาน
- ข้อมูลที่ไม่มีการกำหนดค่าเริ่มต้น หมายถึงข้อมูลที่ไม่มีการกำหนดค่าเริ่มต้นแต่จะถูกกำหนดค่าเป็น 0 ในขณะเริ่มต้นการทำงานของโปรแกรม และค่าสามารถเปลี่ยนแปลงได้ในขณะที่โปรแกรมทำงาน ข้อดีของข้อมูลชนิดนี้คือระบบไม่จำเป็นต้องสร้างเนื้อที่ขึ้นมาสำหรับเก็บข้อมูลชนิดนี้ในโปรแกรมไฟล์ เพราะค่าของข้อมูลจะต้องถูกกำหนดเป็น 0 โดยระบบอยู่แล้วในขณะที่โปรแกรมเริ่มทำงาน

3. ฮีป (Heap) ซึ่งเป็นเนื้อที่ที่ถูกใช้งานในขณะที่โปรแกรมทำงานเพื่อสร้างเนื้อที่สำหรับข้อมูลแบบไดนามิกส์ให้กับโปรเซส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. แสตก (Stack) ใช้สำหรับการจัดการเกี่ยวกับทำงานพวกรูทีนหรือฟังก์ชันต่างๆ ในโปรแกรม ใช้ในขั้นตอนการเรียกฟังก์ชันและรีเทิร์นฟังก์ชันและเป็นเนื้อที่ที่ใช้สำหรับข้อมูลที่กำหนดขึ้นใช้ภายในฟังก์ชันต่างๆด้วย



รูป 2.13 แสดงส่วนประกอบของโปรเซส

<u>ค่าเฉพาะของโปรเซส (Process ID)</u> ทุกๆ โปรเซสจะมีค่าตัวเลขเฉพาะของแต่ละโปรเซสหรือเรียกย่อๆว่าพีไอดี (PID) พีไอดีเป็นตัวเลขซึ่งมีค่าอยู่ระหว่าง 0 ถึง 30,000 โดยค่านี้เคอเนลจะเป็นผู้กำหนดให้แต่ละโปรเซสเมื่อโปรเซสใหม่เกิดขึ้นในระบบ โดยแต่ละโปรเซสจะมีค่าพีไอดีไม่ซ้ำกัน โปรเซสที่มีค่าพีไอดี 0 , 1 , 2 จะหมายถึงโปรเซสที่เป็นโปรเซสที่สร้างขึ้นโดยระบบเมื่อระบบเริ่มทำงาน

ทุกๆโปรเซสจะมีค่าเฉพาะของโปรเซสผู้ปกครอง (Parent Process ID) ซึ่งค่านี้เคอเนลก็เป็นผู้กำหนดค่านี้ให้กับโปรเซสที่เกิดขึ้นใหม่ภายในระบบ โปรเซสหลายตัวในระบบอาจมีค่าเฉพาะของโปรเซสผู้ปกครองเหมือนกันก็ได้

2.3.1.4 คำสั่งระบบ (System Calls)

เคอเนล จะมีคำสั่งจำนวนหนึ่ง (โดยทั่วไป ประมาณ 60 ถึง 200 คำสั่ง) ซึ่งเป็นการใช้งานโดยตรงจากโปรเซสไปยังเคอเนล ซึ่งเรียกว่า คำสั่งระบบ ซึ่งการทำงานในระดับคำสั่งเครื่อง (Maching Instruction)ในระบบจริงๆนั้นจะต้องมีการจัดการเกี่ยวกับการส่งผ่านอาร์กิวเมนต์และผลลัพธ์ที่ได้ระหว่างโปรเซสและเคอเนลเมื่อมีคำสั่งระบบเกิดขึ้น แต่สำหรับผู้เขียนโปรแกรมไม่จำเป็นต้องทราบเกี่ยวกับขั้นตอนดังกล่าวเพราะในภาษาซีมาตรฐานบนยูนิกซ์ (Standatrd Unix C Library) ได้มีคำสั่งของภาษาที่สามารถเรียกใช้คำสั่งระบบทุกคำสั่งได้ ทำให้การใช้งานคำสั่งระบบภายในภาษาซีนั้นเหมือนกับการเรียกใช้ฟังก์ชันทั่วๆไปของภาษานั่นเอง และในบางครั้งจึงมักจะพบว่ามีการใช้คำว่าฟังก์ชันและคำสั่งระบบในความหมายเดียวกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับการใช้คำสั่งระบบเมื่อเกิดความผิดพลาดขึ้นจะรีเทิร์นค่า -1 และจะรีเทิร์นค่า 0 หรือมากกว่า 0 เมื่อสำเร็จ และเมื่อมีความผิดพลาดเกิดขึ้นระหว่างทำงานในคำสั่งระบบจะทำให้เกิดค่า errno ซึ่งเป็นตัวเลขที่ใช้ส่งผ่านค่าความผิดพลาดไปยังระบบ เมื่อเคอเนลรับทราบถึงความผิดพลาดนั้นก็จะจัดการส่งข้อความแจ้งมายังผู้ใช้และสิ้นสุดคำสั่งนั้นๆหรือไม่ก็แล้วแต่ในระบบจะกำหนดไว้

2.3.2 การสร้างโปรเซสใหม่ในระบบและการเรียกใช้โปรแกรมอื่นในโปรเซส

ดังที่ได้กล่าวถึงมาแล้วว่าในระบบยูนิกซ์นั้น การสร้างโปรเซสใหม่ขึ้นมาในระบบต้องใช้คำสั่งระบบ ฟอร์ค การทำงานของคำสั่งระบบนี้จะสร้างก็อปปีของโปรเซสที่กำลังทำงานอยู่ขึ้นมาอีกโปรเซสหนึ่ง และโปรเซสซึ่งเป็นผู้ใช้คำสั่ง ฟอร์ค นี้จะเป็นโปรเซสผู้ปกครองของโปรเซสใหม่ที่เกิดขึ้นจากคำสั่งนี้ และโปรเซสใหม่ก็จะมีฐานะเป็นโปรเซสลูกของโปรเซสที่สร้างมันขึ้นมา

โปรเซสลูกจะใช้ส่วนของคำสั่งร่วมกับโปรเซสผู้ปกครอง และจะก็อปปี้ส่วนของข้อมูลจากโปรเซสผู้ปกครองมาเป็นข้อมูลของโปรเซส และลักษณะที่สำคัญคือไฟล์ใดๆที่ถูกเปิดในโปรเซสผู้ปกครองแล้วโปรเซสลูกจะก็อปปี้ไฟล์เดสคริปเตอร์ (File descriptors) เหล่านั้นมาเป็นไฟล์เดสคริปเตอร์ของตนเอง

จุดประสงค์ในการสร้างโปรเซสขึ้นใหม่อาจแบ่งได้ 2 กรณีคือ

1. เมื่อโปรเซสต้องการที่จะทำงานที่แตกต่างกันในขณะเดียวกัน
2. เมื่อโปรเซสต้องการเรียกใช้งานโปรแกรมอื่นๆ เนื่องจากทางเดียวที่จะสามารถสร้างโปรเซสใหม่ขึ้นมาต้องใช้คำสั่งฟอร์คก่อนเพื่อก็อปปีโปรเซสเดิมเป็นอีกโปรเซสหนึ่ง แล้วจึงเรียกโปรแกรมอื่นขึ้นมาทำงานในโปรเซสใหม่

หลังจากที่เรียกใช้คำสั่งระบบฟอร์คเพื่อสร้างโปรเซสใหม่ขึ้นมา จะได้โปรเซสที่มีส่วนของคำสั่งที่ใช้งานร่วมกับโปรเซสผู้ปกครอง และส่วนข้อมูลที่ก็อปปี้จากโปรเซสผู้ปกครองมา เมื่อเราต้องการจะนำโปรแกรมอื่นมาทำงานแทนในโปรเซสที่มีอยู่สามารถทำได้โดยใช้คำสั่งระบบ เอกซ์เซค ผลของคำสั่งนี้ไม่ทำให้ค่าพีไอดีเปลี่ยนแปลงไป โปรเซสที่เรียกใช้งานเอกซ์เซคนี้เราจะเรียกว่า โปรเซสผู้เรียกใช้ และโปรแกรมที่ถูกเรียกใช้ในคำสั่งเอกซ์เซคเรียกว่าโปรแกรมใหม่ หลายคนเข้าใจว่าโปรแกรมใหม่คือโปรเซสใหม่ซึ่งเป็นการเข้าใจที่ผิดเนื่องจากโปรแกรมใหม่นั้นจะทำงานในโปรเซสที่เรียกใช้โปรแกรมใหม่เพียงแต่เปลี่ยนแปลงสภาพไปจากเดิมเท่านั้น นั่นคือส่วนของข้อมูลและคำสั่งจะถูกแทนที่ด้วยโปรแกรมใหม่

2.3.3 สัญญาณ (Signal)

สัญญาณสามารถใช้เพื่อติดต่อระหว่างโปรเซสได้ สัญญาณบางครั้งอาจเรียกว่าซอฟต์แวร์อินเทอร์รัป (Software Interrupt) ก็ได้ การเกิดของสัญญาณไม่จำเป็นต้องเกิดขึ้นด้วยความถี่คงที่ จะเกิดเมื่อใดก็ได้ (Asynchronous) ดังนั้นโปรเซสจะไม่สามารถทราบล่วงหน้าว่าจะเกิดสัญญาณชนิดใดขึ้นเมื่อไร สัญญาณสามารถถูกส่งได้จากโปรเซสหนึ่งไปยังโปรเซสอื่นๆ หรืออาจส่งให้กับโปรเซสตัวเองก็ได้ หรือสัญญาณอาจถูกส่งจากเคอเนลไปยังโปรเซสก็ได้ สัญญาณในระบบทั้งหมดสามารถดูรายละเอียดได้จากไฟล์ signal.h โดยทั่วไปสัญญาณจะประกอบไปด้วยชนิดสัญญาณต่างๆดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Name	Value	Action	Comment
SIGHUP	1	A	Hangup detected
SIGINT	2	A	Interrupt from keyboard
SIGQUIT	3	A	Quit from keyboard
SIGILL	4	A	Illegal Instruction
SIGTRAP	5	CG	Trace / breakpoint trap
SIGABRT	6	C	Abort
SIGUNUSED	7	AG	Unused signal
SIGFPE	8	C	Floating point exception
SIGKILL	9	AEF	Termination signal
SIGUSR1	10	A	User-defined signal 1
SIGSEGV	11	C	Invalid memory reference
SIGUSR2	12	A	User-defined signal 2
SIGPIPE	13	A	Write to pipe with no readers
SIGALRM	14	A	Timer signal from alarm(1).
SIGTERM	15	A	Termination signal
SIGSTKFLT	16	AG	Stack fault on coprocessor
SIGCHLD	17	B	Child terminated
SIGCONT	18	B	Continue if stopped
SIGTSTOP	19	DEF	Stop process
SIGTSTP	20	D	Stop typed at tty
SIGTTIN	21	D	tty input for background process
SIGTTOU	22	D	tty output for background process
SIGIO	23	AG	I/O error
SIGXCPU	24	AG	CPU time limit exceeded
SIGXFSZ	25	AG	File size limit exceeded
SIGVTALRM	26	AG	Virtual time alarm (???)
SIGPROF	27	AG	Profile signal
SIGWINCH	29	BG	Window resize signal

ตาราง 2.5 ชนิดของสัญญาณ

ค่าในช่อง Action หมายถึงค่า Default Action ของสัญญาณ มีความหมายดังนี้

A Default action is to terminate the process.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

B Default action is to ignore the signal.

C Default action is to dump core.

D Default action is to stop the process.

E Signal cannot be caught.

F Signal cannot be ignored.

G Not a POSIX.1 conformant signal.

วิธีต่างๆในการส่งสัญญาณ อาจจำแนกได้ดังนี้

1. โดยใช้คำสั่งระบบ คิล (Kill) คำสั่งนี้อนุญาตให้โปรเซสส่งสัญญาณไปยังโปรเซสอื่นๆหรือตนเองได้ โดยทั่วไปแล้วโปรเซสจะไม่สามารถส่งสัญญาณไปยังโปรเซสโดยทั่วไปได้ โปรเซสที่ส่งสัญญาณและโปรเซสที่จะรับสัญญาณจะต้องมี effective user ID ให้มีอันกัน หรือโปรเซสที่ส่งสัญญาณอาจเป็นโปรเซสของซุปเปอร์ยูสเซอร์ก็จะสามารถส่งสัญญาณไปยังโปรเซสใดๆได้ โดยในคำสั่งคิลจะให้ระบุอาร์กิวเมนต์ คือค่าพีไอดีของโปรเซสที่ต้องการจะส่งสัญญาณไปให้ และค่าสัญญาณที่ต้องการส่งไปให้

คำสั่งคิลนี้อาจมีความหมายในการส่งสัญญาณได้หลายความหมายตามค่าอาร์กิวเมนต์ที่ส่ง คือ

- ถ้าค่าของ พีไอดี ที่ระบุไว้ในการส่งสัญญาณมีค่าเท่ากับ 0 สัญญาณจะถูกส่งไปยังทุกๆโปรเซสที่อยู่ในกรุปเดียวกันกับโปรเซสผู้ส่งสัญญาณ

- ถ้าค่า พีไอดี เท่ากับ -1 และโปรเซสผู้ส่งสัญญาณไม่ใช่ซุปเปอร์ยูสเซอร์ สัญญาณจะถูกส่งไปยังทุกโปรเซสที่มี ค่า เรียลยูสเซอร์ไอดี เท่ากับ เอฟเฟคทีฟยูสเซอร์ไอดี ของโปรเซสที่ส่งสัญญาณ

- ถ้าค่า พีไอดี เท่ากับ -1 และโปรเซสที่ส่งเป็นซุปเปอร์ยูสเซอร์ สัญญาณจะส่งไปยังทุกโปรเซสยกเว้นโปรเซสที่เป็นของระบบ ซึ่งมีค่าพีไอดี เป็น 0 , 1 , 2

- ถ้าค่าพีไอดีเป็นค่าลบแต่ไม่เท่ากับ -1 จะส่งสัญญาณไปยังทุกโปรเซสที่มี ค่ากรุปไอดีเท่ากับค่าสมบูรณ์ของพีไอดีของโปรเซสที่ส่งสัญญาณ

- ถ้าค่าของสัญญาณที่ส่งเท่ากับ 0 จะเป็นการตรวจสอบว่ามีโปรเซสตามที่ พีไอดีระบุไว้อยู่ในระบบหรือไม่ ไม่ได้เป็นการส่งสัญญาณจริงๆ

2. นอกจากการใช้คำสั่งระบบคิลนี้ในระบบแล้ว ยังสามารถใช้คำสั่งคิลนี้ในโปรแกรมเพื่อส่งสัญญาณได้เหมือนกัน

3. การส่งสัญญาณโดยใช้อักขระพิเศษที่ได้จากแป้นพิมพ์ไปให้กับโปรเซสที่กำลังทำงานอยู่ในขณะนั้น เช่น การกดแป้น คอนโทรล-ซี หรือ คอนโทรล-ดี เพื่อใช้ในการส่งสัญญาณ SIGINT ให้สิ้นสุดการทำงานของโปรเซส

4. บางกรณีฮาร์ดแวร์ก็อาจก่อให้เกิดการส่งสัญญาณไปให้แก่โปรเซสได้ เช่น เมื่อเกิดการโอเวอร์โฟลวของการคำนวณเลขทศนิยมเกิดขึ้น เคอเนลจะส่งสัญญาณ SIGFPE ไปยังโปรเซสที่ทำให้เกิดเหตุการณ์ดังกล่าวขึ้น

5. อาจมีกรณีที่การทำงานของโปรเซสอาจทำให้เคอเนลส่งสัญญาณออกมาก็ได้ เช่น สัญญาณ SIGURG จะถูกส่งออกมาเมื่อเกิดกรณีความถี่ของข้อมูลที่ซอคเก็ตไม่ตรงกับการทำงานของเครื่อง เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.4 การจับสัญญาณในโปรเซส (Catch Signal) และนำมาใช้งาน

เมื่อเราสามารถส่งสัญญาณไปยังโปรเซสต่างๆได้แล้ว โปรเซสที่ได้รับสัญญาณอาจเลือกวิธีที่จะจัดการกับสัญญาณต่างๆที่ได้รับดังนี้

1. โปรเซสอาจสร้างฟังก์ชันขึ้นมาเพื่อให้ทำงานเมื่อมีสัญญาณที่ต้องการส่งมาถึง ซึ่งฟังก์ชันนี้เราเรียกว่าเป็นตัวจัดการสัญญาณ และวิธีนี้เรียกว่าการจับสัญญาณในโปรเซส

2. โปรเซสอาจเลือกที่จะไม่สนใจกับสัญญาณที่ถูกส่งมาให้ก็ได้ ทุกสัญญาณยกเว้น SIGKILL และ SIGSTOP โปรเซสสามารถที่จะไม่สนใจได้

3. โปรเซสอาจเลือกที่จะกระทำเมื่อสัญญาณนั้นๆเกิดขึ้นด้วยการกระทำที่กำหนดไว้ก่อนแล้ว (Default Action) ซึ่งสิ่งที่กำหนดไว้ก่อนนั้นส่วนมากจะเป็นการสิ้นสุดการทำงานของโปรเซส

วิธีจัดการต่างๆที่กล่าวมานี้สามารถทำได้โดยใช้คำสั่งระบบ ซิกเนล

```
signal ( int sig , func);
```

sig หมายถึงค่าของสัญญาณที่ต้องการจะจับ

func หมายถึง ชื่อของฟังก์ชันที่ต้องการให้ทำงานเมื่อได้รับสัญญาณดังกล่าว ซึ่งอาจเป็นชื่อฟังก์ชันจริงๆ หรือ SIG_IGN (หมายถึงไม่ต้องสนใจเมื่อมีสัญญาณดังกล่าวส่งมา) หรือ SIG_DFL (หมายถึงให้ใช้สิ่งที่กำหนดไว้ก่อนในระบบสำหรับสัญญาณดังกล่าวเมื่อได้รับ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การพัฒนาระบบ

ในบทนี้จะกล่าวถึงการนำวัตถุประสงค์ของโครงงาน หลักการและความรู้เพื่อใช้ในการสร้างระบบจริงๆ โดยจะกล่าวถึงความต้องการของระบบที่จะสามารถตอบสนองต่อวัตถุประสงค์ที่ตั้งขึ้นไว้ในตอนต้น จากนั้นเป็นการนำความต้องการของระบบไปพัฒนาในส่วนของโปรแกรมเพื่อสอดคล้องกับความต้องการ โดยอธิบายการเขียนโปรแกรมและการทำงานของระบบโดยแยกเป็นแต่ละส่วน

3.1 ความต้องการของระบบ

จากวัตถุประสงค์ที่ตั้งไว้นำมาพิจารณาสามารถแบ่งความต้องการของระบบโดยรวมได้เป็นส่วนๆ ดังนี้

3.1.1 สามารถเลือกวิธีการย่อข้อมูลจากหลายวิธีได้เองตามตัวเลือกที่ต้องการ

จากวัตถุประสงค์ของโครงงานและที่มาของปัญหาคือผู้ใช้อาจไม่รู้จักวิธีต่างๆ ในการย่อข้อมูล ซึ่งในปัจจุบันมีอยู่มากมายหลายวิธี แต่ละวิธีก็ให้ผลให้การย่อข้อมูลที่แตกต่างกัน เช่น บางวิธีอาจต้องใช้เวลาในการย่อข้อมูลมาก บางวิธีใช้เวลาน้อย และขนาดของข้อมูลหลังจากย่อโดยวิธีที่ต่างกันก็ต่างกันด้วย ดังนั้นระบบที่สร้างขึ้นจึงควรมีตัวเลือกให้ผู้ใช้ระบุความต้องการในการย่อข้อมูล เช่น เวลาที่ใช้ในการย่อ เวลาที่ใช้ในการขยาย เนื้อที่ที่ใช้ เป็นต้น แล้วระบบจึงนำค่าต่างๆที่ผู้ใช้ระบุไปทำการย่อข้อมูลจากวิธีต่างๆที่มีอยู่ในระบบแล้วเปรียบเทียบเลือกวิธีในการย่อเพียงวิธีเดียวที่เหมาะสมกับค่าที่ผู้ใช้ระบุเพียงวิธีเดียว แล้วใช้วิธีดังกล่าวในการย่อไฟล์ที่ผู้ใช้ต้องการ

3.1.2 สามารถขยายข้อมูลตามวิธีที่ถูกย่อได้เอง

เมื่อผู้ใช้ต้องการขยายไฟล์ใดๆ ที่ได้ผ่านการย่อแล้ว ผู้ใช้ไม่จำเป็นต้องทราบว่าไฟล์นั้นได้ถูกย่อด้วยวิธีใด แต่ระบบต้องสามารถตรวจสอบวิธีในการย่อไฟล์นั้นและเลือกวิธีที่ขยายไฟล์อย่างถูกต้องได้เอง

3.1.3 สามารถแสดงรูปภาพที่ผู้ใช้ต้องการได้

เพื่อตอบสนองการใช้งานไฟล์รูปภาพ ผู้ใช้ส่วนมากต้องการการแสดงภาพ หรือไฟล์ใดๆ ระบบควรมีความสามารถนำไฟล์ดังกล่าวมาแสดงให้ผู้ใช้ได้เห็นในฟอร์แมทที่ถูกต้อง เช่น ถ้าเป็นเทกซ์ไฟล์ระบบก็

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ควรเปิดไฟล์และแสดงไฟล์นั้นในรูปแบบของเทกซ์ไฟล์ได้ ถ้าเป็นไฟล์รูปภาพระบบต้องสามารถตรวจสอบว่าเป็นไฟล์รูปภาพในฟอร์เมทใดและแสดงภาพให้ถูกต้องในแต่ละฟอร์เมท

3.1.4 สามารถทำงานในส่วนที่ไม่ต้องติดต่อกับผู้ใช้เป็นแบคกราวน์โปรเซสได้

เพื่อช่วยในการประหยัดเวลาในการทำงาน ได้แก่ การทำงานในการเลือกวิธีการย่อไฟล์และการย่อไฟล์ การแสดงผลไฟล์ เป็นต้น

3.1.5 วิธีเพื่อให้ผู้ใช้ได้ทราบสถานะการทำงานของระบบ

เนื่องจากระบบต้องทำงานในลักษณะแบคกราวน์โปรเซสได้ จึงทำให้ในเวลาหนึ่งๆระบบอาจมีโปรเซสที่เกิดขึ้นจากการทำงานของระบบอยู่หลายโปรเซส และการทำงานของโปรเซสส่วนหนึ่งเสร็จสิ้นลงไปแล้ว ซึ่งระบบควรมีวิธีที่สามารถให้ผู้ใช้รับทราบจำนวนโปรเซสที่กำลังทำงาน และโปรเซสใดทำงานเสร็จเรียบร้อย หรือเกิดความผิดพลาดขึ้นในโปรเซสใดบ้างหรือไม่

3.1.6 สามารถให้ผู้ใช้ใช้งานได้อย่างสะดวก

โปรแกรมที่ดี ควรจะทำให้ผู้ใช้รู้สึกสะดวกสบาย และเข้าใจง่าย ในการใช้งาน ถ้าหากโปรแกรมมีความสามารถทำงานได้ดี แต่ขาดการพัฒนาเพื่อให้ผู้ใช้ได้ใช้งานอย่างสะดวกย่อมทำให้โปรแกรมดังกล่าวไม่สามารถใช้งานได้บรรลุจุดประสงค์ตามต้องการได้

3.2 ระบบที่ได้พัฒนาขึ้นและการทำงานในส่วนต่างๆ ของระบบ

จากความต้องการของระบบที่ได้กล่าวถึงความสามารถที่จะต้องพัฒนาขึ้น หลังจากกระบวนการพัฒนาระบบได้เสร็จสิ้นลง ระบบที่ได้ถูกสร้างขึ้นนี้เป็นระบบที่สามารถทำงานเพื่อตอบสนองความต้องการดังกล่าวได้ส่วนหนึ่งและยังต้องการพัฒนาเพื่อบรรลุวัตถุประสงค์ที่ต้องการในอนาคต ระบบที่ได้พัฒนาขึ้นประกอบไปด้วยโปรแกรม 3 โปรแกรมที่สามารถทำงานแยกกันได้ คือ โปรแกรมการทำงานหลัก โปรแกรมการย่อข้อมูล โปรแกรมแสดงผลรูปภาพ ซึ่งมีรายละเอียดในการทำงานดังต่อไปนี้

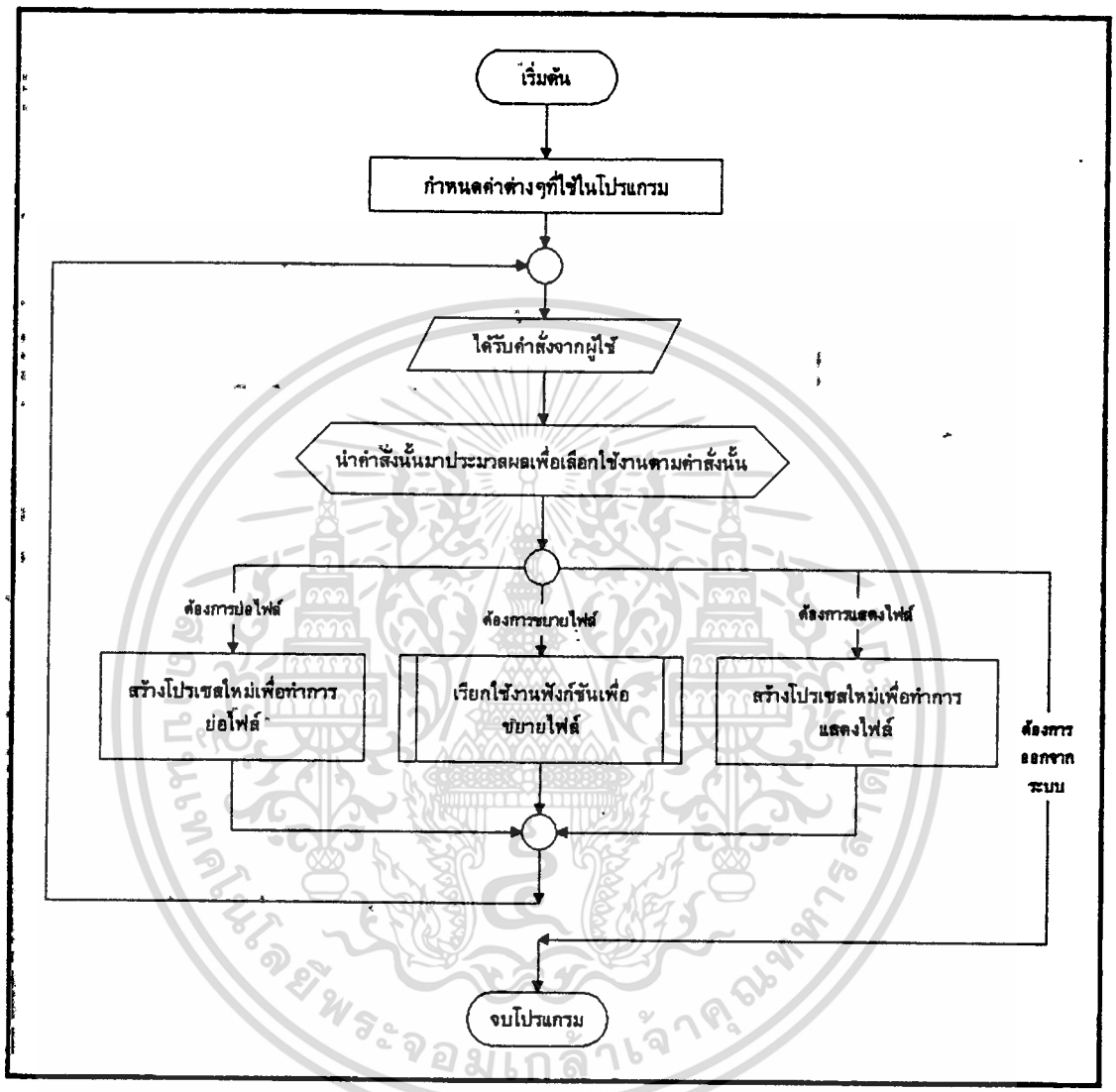
3.2.1 โปรแกรมการทำงานหลัก

การทำงานในส่วนของโปรแกรมหลักประกอบไปด้วย

1. ส่วนที่สร้างหน้าจอการติดต่อกับผู้ใช้สำหรับโปรแกรม
2. การจัดการในการติดต่อระหว่างโปรแกรมหลักและโปรแกรมอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. การใช้ข้อความเพื่อแสดงผลการทำงานให้ผู้ใช้ทราบ
4. ฟังก์ชันในการขยายไฟล์ที่ถูกย่อไว้



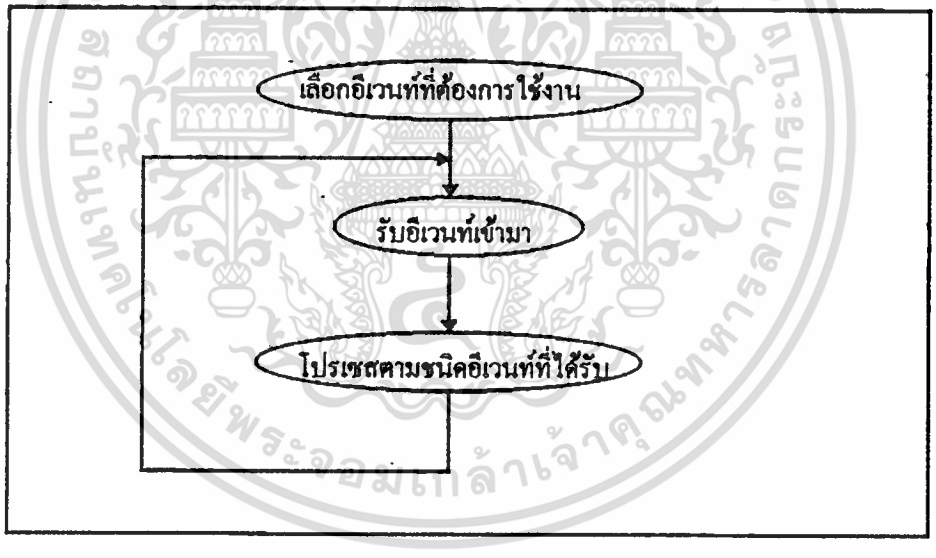
รูป 3.1 การทำงานของโปรแกรมหลัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- วินโดว์หลัก ไม่รับอีเวนท์ใดๆ
- ไทเทิลวินโดว์ รับอีเวนท์เอกโพส เพื่อใช้ในการวาดหน้าจอใหม่
- เมนูวินโดว์ทั้ง 4 อันจะรับเอกโพสอีเวนท์ และการคลิกของตัวชี้ที่วินโดว์ เพื่อใช้เป็นคำสั่งให้แก่ระบบ คือ เมื่อผู้ใช้คลิกตัวชี้ที่เมนูวินโดว์ที่หนึ่งหมายถึงผู้ใช้ต้องการจะย่อไฟล์ เมื่อผู้ใช้คลิกตัวชี้ที่เมนูวินโดว์ที่สองหมายถึงผู้ใช้ต้องการจะขยายไฟล์ เมื่อผู้ใช้คลิกตัวชี้ที่เมนูวินโดว์ที่สามหมายถึงผู้ใช้ต้องการจะแสดงไฟล์รูปภาพ เมื่อผู้ใช้คลิกตัวชี้ที่เมนูวินโดว์ที่สี่หมายถึงผู้ใช้ต้องการจะออกจากโปรแกรมนี้
- อินพุทไฟล์วินโดว์ เมื่อได้รับที่เป็นอินพุทจากเมนูว่าผู้ใช้ต้องการย่อ หรือขยาย หรือแสดงไฟล์รูปภาพ ต่อไประบบจะให้ผู้ใช้ใส่ชื่อไฟล์นั้นลงในอินพุทไฟล์วินโดว์นี้ ดังนั้นวินโดว์นี้จะรับอีเวนท์เป็นการกดแป้นพิมพ์ และเอกโพสอีเวนท์
- แมสเซสวินโดว์ จะรับเอกโพสอีเวนท์เพียงอย่างเดียว

3. <u>การรับอินพุทจากผู้ใช้</u>

เนื่องจากการเขียนโปรแกรมโดยใช้เอกซ์ลิปฟังก์ชันเป็นการเขียนโปรแกรมที่จะต้องติดต่อโดยตรงกับเอกซ์โปรโตคอลและมีรูปแบบการทำงานของโปรแกรมแบบอีเวนท์ไดรเวน (Event Driven)



ซึ่งเป็นการทำงานในส่วนการรับอินพุทเมนูทั้ง 4 อัน อินพุทที่ใช้ในการรับอินพุทชื่อไฟล์มีขั้นตอนในการทำงานดังนี้

- แมปอินพุทไฟล์วินโดว์ เพื่อรอรับการพิมพ์ชื่อไฟล์จากผู้ใช้
  - WHILE มีการกดแป้นพิมพ์
    - นำรหัสของแป้นพิมพ์ที่ได้รับแปลงเป็นชื่อของแป้นพิมพ์นั้น
    - IF ชื่อของแป้นพิมพ์เป็น XK_Return หมายถึงผู้ใช้กดปุ่ม Enter
      - เก็บ Null สตริงเป็นตัวอักษรปิดท้ายชื่อไฟล์นั้น
      - อันแมปอินพุทไฟล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

□สิ้นสุดการรับชื่อไฟล์

□ELSE

□นำรหัสของปุ่มที่กดแปลงเป็นตัวอักษร

□เก็บตัวอักษรต่อท้ายในสตริง

□พิมพ์ตัวอักษรดังกล่าวในอินพุทไฟล์วินโดว์และพิดแมป

3.2.1.2 การติดต่อระหว่างโปรแกรมหลักและแบคกราวน์โปรแกรม

การจัดการเกี่ยวกับสัญญาณที่ได้รับจากโปรเซสลูก โดยในโปรแกรมหลักจะต้องประกอบด้วย

1. เรียกใช้ฟังก์ชันในการจับสัญญาณเมื่อโปรเซสลูกสิ้นสุดการทำงาน

□ Signal (SIGCHLD , handler);

หมายถึงเมื่อมีสัญญาณดังกล่าวเข้ามาให้มีการเรียกใช้ฟังก์ชัน handler

2. สร้างฟังก์ชันที่จัดการกับสัญญาณดังกล่าว โดยในฟังก์ชันดังกล่าวมีการทำงานดังต่อไปนี้

□อ่านค่า PID และ status ในการสิ้นสุดการทำงานของโปรเซสลูก

□ค่า status ที่ได้จะประกอบไปด้วย 16 บิท โดย 8 บิทแรกเป็นค่าที่โปรเซสลูกใช้ในการสิ้นสุดการทำงานซึ่งเราต้องการค่านี้ ส่วน 8 บิทหลังจะมีค่าเป็น 0 ดังนั้นเราจะต้องทำการเลื่อนค่า status ที่ได้มาด้านขวา 8 บิท เพื่อให้ได้ค่า status ที่ต้องการ

□นำค่าที่ได้มาใช้เป็นตัวชี้ตำแหน่ง (Index) เพื่อเขียนข้อความให้ผู้ใช้ได้รับทราบถึงการสิ้นสุดของโปรเซสดังกล่าวและการทำงานของโปรเซสนั้นสำเร็จหรือมีข้อผิดพลาดใดเกิดขึ้น

3.2.1.3 การใช้ข้อความสื่อสารให้ผู้ใช้ทราบถึงการทำงานของโปรแกรม

เมื่อระบบมีข้อความเพื่อใช้เป็นแนวทางในการใช้งานให้ผู้ใช้ได้รับทราบ หรือข้อความเพื่อแจ้งถึงผลการทำงานในแต่ละขั้นตอน หรือโปรเซสที่ได้สิ้นสุดการทำงานลงแล้ว จะใช้แมสเซสวินโดว์เพื่อการส่งข้อความต่างๆ เหล่านี้ถึงผู้ใช้ ข้อความที่ใช้ในการส่งจะประกอบไปด้วย

1. ชื่อฟังก์ชันหรือโปรเซสที่ต้องการแจ้งข้อความให้ผู้ใช้ทราบ (ถ้ามี)
2. ข้อความที่ต้องการจะส่ง
3. ชื่ออินพุทไฟล์ที่เกี่ยวข้องกับการทำงานในส่วนดังกล่าว (ถ้ามี)

ตัวอย่างเช่น

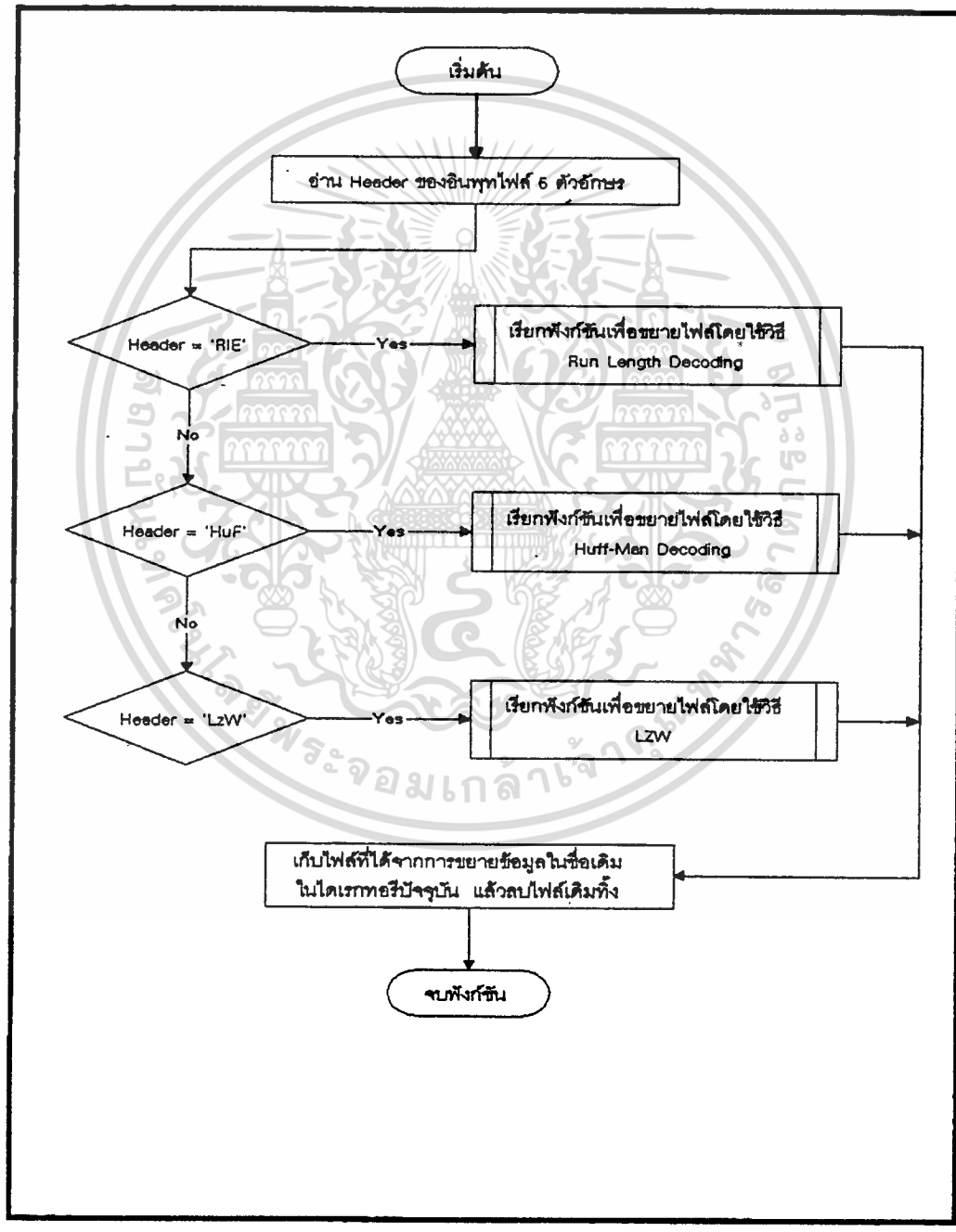
Compress : Successfull for execute
orange.tif

หมายถึง การทำงานในโปรเซสการย่อข้อมูลของไฟล์ orange.tif สำเร็จเรียบร้อยไม่มีข้อผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.1.4 การขยายไฟล์ข้อมูลตามวิธีที่ได้ถูกย่อไว้

เนื่องจากในระบบที่พัฒนาขึ้นนี้ในส่วนของการขยายไฟล์ได้สร้างขึ้นเป็นฟังก์ชันหนึ่งของโปรแกรมหลักเท่านั้น ไม่ได้สร้างขึ้นเป็นอีกโปรแกรมหนึ่ง ดังนั้นการขยายไฟล์จึงไม่สามารถทำงานเป็นแบคกราวน์โปรเซสได้ เมื่อผู้ใช้ต้องการที่จะขยายไฟล์ข้อมูลใดๆ ผู้ใช้จะต้องรอให้การขยายข้อมูลไฟล์นั้นเสร็จสิ้นเสียก่อนจึงจะทำงานอื่นต่อไปได้ การทำงานในฟังก์ชันของการขยายข้อมูลมีขั้นตอนดังต่อไปนี้



รูป 3.3 การทำงานของฟังก์ชันขยายข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

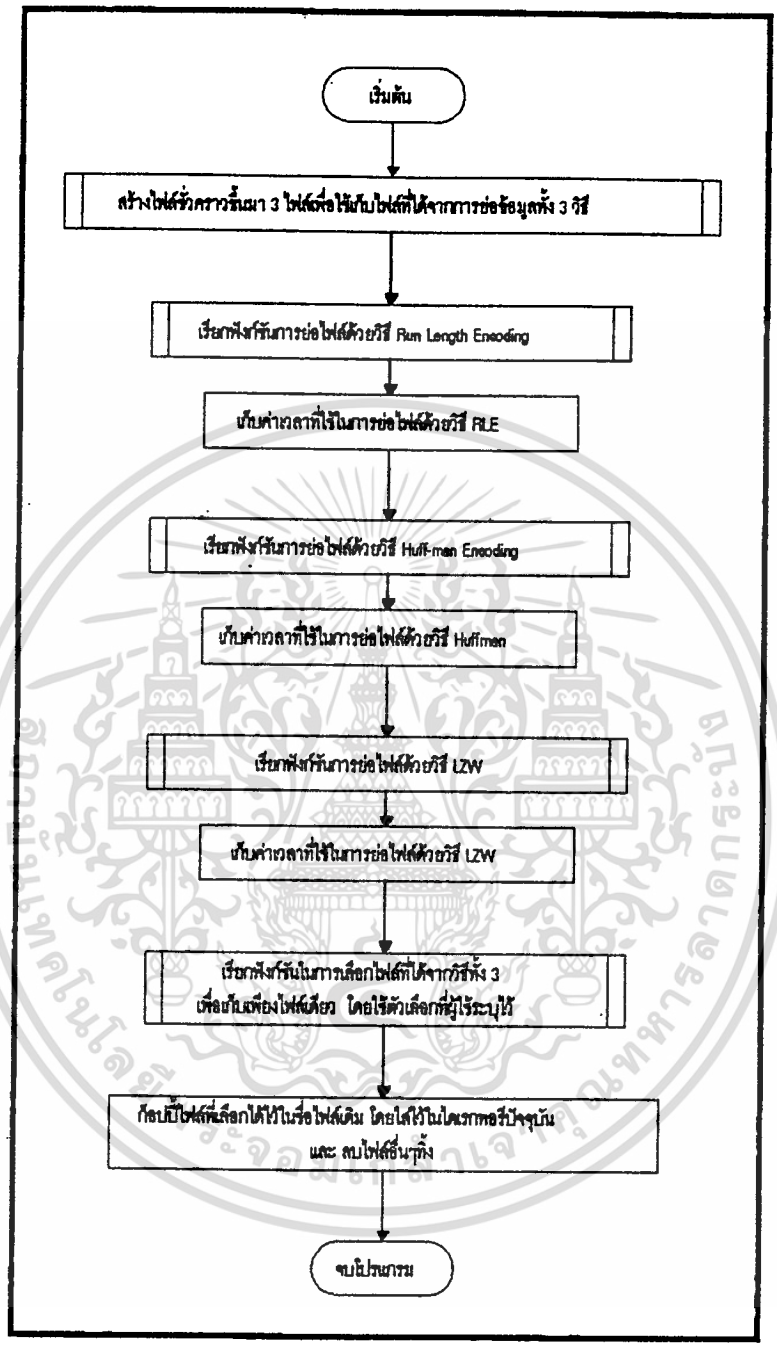
ไฟล์ที่ได้ย่อข้อมูลไว้จะเก็บส่วนต้นของไฟล์เป็นชื่อวิธีที่ใช้ในการย่อไฟล์เหล่านั้นเพื่อแยกแยะว่าไฟล์ดังกล่าวได้ถูกย่อข้อมูลไว้ด้วยวิธีใด ในระบบนี้ได้แบ่งการเก็บชนิดของการย่อข้อมูลไว้สามชนิดดังนั้นจะมี header ของไฟล์สามชนิด คือ 'RIE' สำหรับไฟล์ที่ย่อข้อมูลด้วยวิธี Run Length Encoding ,'HuF' สำหรับไฟล์ที่ย่อข้อมูลด้วยวิธี Huff-Man Encoding , 'LzW' สำหรับไฟล์ที่ย่อข้อมูลด้วยวิธี LZW

ในส่วนของขั้นตอนการขยายข้อมูลจะอธิบายไว้ในการอธิบายการย่อข้อมูลแต่ละวิธีเพื่อความเข้าใจในการทำงานได้ดียิ่งขึ้น

3.2.2 โปรแกรมการย่อข้อมูล

การทำงานในโปรแกรมการย่อข้อมูลมีขั้นตอนดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 3.4 การทำงานของโปรแกรมการย่อข้อมูล

เมื่อมีการสร้างโปรเซสสำหรับการย่อไฟล์ขึ้น และมีชื่อไฟล์อินพุท และตัวเลือกในการย่อข้อมูลที่ผู้ใช้เลือก (Minimize time , Minimize space) เป็นอาร์กิวเมนต์มายังโปรเซส โปรเซสจะสร้างไฟล์ชั่วคราวขึ้นมา 3 ไฟล์เพื่อเก็บไฟล์ที่ได้จากการย่อข้อมูลทั้งสามวิธี จากนั้นจะเรียกฟังก์ชันในการย่อข้อมูลแต่ละวิธีมาย่อไฟล์อินพุทแล้วเก็บผลที่ได้ไว้ในไฟล์ชั่วคราว พร้อมทั้งเก็บเวลาที่ใช้ในการย่อข้อมูลแต่ละวิธีด้วย จากนั้นเมื่อทำการย่อไฟล์ครบทั้งสามวิธีแล้วจะเรียกฟังก์ชันในการหาค่าประสิทธิภาพของไฟล์ที่ได้ทั้งสามว่ามีไฟล์ใดตรงกับค่าที่ผู้ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต้องการมากที่สุดจะเลือกไฟล์นั้นเก็บไว้ส่วนไฟล์อื่นๆจะลบทิ้งไป

3.2.2.1 การย่อและขยายข้อมูลโดยใช้วิธี Run Length Encoding

ในการย่อข้อมูลโดยวิธีนี้มีขั้นตอนการทำงานดังนี้

- กำหนด header ของข้อมูลเป็นสองชนิดคือ ส่วนที่ทำการย่อ ใช้ header (RUN) = 80h และส่วนที่ไม่ได้ย่อ header (SEQ) = 00h
- กำหนดจำนวนข้อมูลสูงสุดในการย่อแต่ละตัวอักษร (MAX_LEN) = 7fh
- สร้างบัฟเฟอร์เพื่อเก็บข้อมูลส่วนที่ไม่ได้ย่อ seq[MAX_LEN]
- เขียน header ลงไฟล์เอาท์พุท = 'RIE'
- กำหนด cur_char = run_char = อักษรจากไฟล์อินพุทตัวแรก
- ให้ตัวนับจำนวนตัวอักษรที่ถูกย่อ run_len = 1 ตัวนับอักษรในบัฟเฟอร์ seq_len = 0
- WHILE ยังไม่สิ้นสุดไฟล์
  - cur_char = ตัวอักษรตัวถัดไปในไฟล์
  - IF cur_char ไม่เท่ากับ run_char
    - IF run_len > 2
      - เขียนข้อมูลที่อยู่ในบัฟเฟอร์ seq ลงไฟล์เอาท์พุท แล้ว ตั้งค่า seq_len = 0
      - เขียนจำนวนและอักษรที่ย่อไว้ลงไฟล์เอาท์พุท
    - ELSE
      - บวกค่าตัวนับอักษรในบัฟเฟอร์ seq_len แล้ว เก็บตัวอักษร run_char ลงในบัฟเฟอร์
    - ตั้งค่า run_len = 1 และ run_char = cur_char
  - ELSE
    - บวกค่าตัวนับจำนวนอักษรที่ย่อได้ run_len ขึ้นหนึ่งค่า
      - IF run_len มีค่าครบ MAX_LEN
        - เขียนข้อมูลที่อยู่ในบัฟเฟอร์ seq ลงไฟล์เอาท์พุท แล้ว ตั้งค่า seq_len = 0
        - เขียนจำนวนและอักษรที่ย่อไว้ลงไฟล์เอาท์พุท
- END_WHILE
- IF ถ้ามีอักษรเหลือในบัฟเฟอร์
  - เขียนจากบัฟเฟอร์ลงเอาท์พุทไฟล์
- IF ถ้ามีอักษรที่ย่อไว้เหลือ
  - เขียนจำนวนและตัวอักษรที่ย่อไว้ลงไฟล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อต้องการขยายข้อมูลที่ได้จากไฟล์การย่อข้อมูลโดยวิธีที่อธิบายมาข้างต้นมีขั้นตอนดังนี้

- WHILE ยังไม่สิ้นสุดไฟล์
  - อ่านข้อมูลเข้ามาหนึ่งตัวเก็บไว้ในตัวแปร packet_hdr
  - นำ packet_hdr มาถอดความหมายว่าจำนวนตัวอักษรในชุดนี้มีกี่ตัวอักษร เก็บไว้ในตัวแปรชื่อ length
  - IF packet_hdr & RUN ซึ่งแสดงว่าข้อมูลชุดต่อไปนี้เป็นตัวอักษรที่ถูกย่อไว้จำนวนเท่ากับ length.
    - อ่านตัวอักษรตัวถัดไปเข้ามาเก็บชื่อว่า byte ซึ่งเป็นตัวอักษรที่ถูกย่อไว้นั่นเอง
    - FOR 1 TO length DO
      - เขียนตัวอักษรที่เก็บไว้ใน byte ลงเอาท์พุทไฟล์
  - ELSE แสดงว่าข้อมูลชุดนี้เป็นข้อมูลที่ไม่ได้ย่อจำนวนเท่ากับ length ตัวอักษร
    - FOR 1 TO length DO
      - อ่านตัวอักษรจากอินพุทไฟล์เข้ามา
      - เขียนตัวอักษรที่ได้จากอินพุทไฟล์ ลงในเอาท์พุทไฟล์
- END_WHILE

3.2.2.2 การย่อและขยายข้อมูลโดยใช้วิธี Huff - Man Encoding

การย่อข้อมูลโดยวิธีนี้อาจแบ่งขั้นตอนการทำงานได้เป็นสี่ขั้นตอนดังนี้คือ

1. การวิเคราะห์เพื่อเก็บความถี่ของการใช้ตัวอักษรทั้ง 256 ตัวอักษรในไฟล์อินพุท ด้วยหลักการย่อข้อมูลวิธีนี้เป็นการแทนอักขระที่มีความถี่ในไฟล์สูงด้วยบิทซีเควนท์ที่สั้น และอักขระที่มีความถี่ต่ำในไฟล์ด้วยบิทซีเควนท์ที่ยาวดังนั้นขั้นตอนเริ่มต้น จึงเป็นการอ่านอักขระในไฟล์ข้อมูลทั้งหมดเพื่อเก็บค่าความถี่ของอักขระที่ใช้แต่ละตัว

- สร้างอาร์เรย์เพื่อเก็บความถี่ของตัวอักษรและใช้ในการสร้างตารางความถี่และ decode table ต่อไป โดยสร้างเป็นอาร์เรย์ของข้อมูลแบบโครงสร้างมีสมาชิกดังนี้

  character คือรหัสแทนอักขระแต่ละตัว

  total คือจำนวนอักขระที่มีอยู่ในไฟล์อินพุท

  frequency คือความถี่ของอักขระแต่ละตัว

  seq_len จำนวนบิทซีเควนท์ที่ใช้แทนอักขระ

  bit_sequent หมายถึงบิทซีเควนท์ที่ใช้แทนอักขระในไฟล์เอาท์พุท

  พอยเตอร์ไปยังสมาชิกตัวบน ซ้าย และขวา ซึ่งในขั้นตอนนี้ให้ค่าเป็น Null

- WHILE ยังไม่สิ้นสุดไฟล์อินพุท
  - อ่านข้อมูลจากไฟล์อินพุทมาทีละหนึ่งตัว
  - บวกค่าเพิ่มใน total ของอักขระตัวที่อ่านมา
- END_WHILE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การนำความถี่ที่ได้มาสร้างเป็นโครงสร้างต้นไม้เพื่อใช้ในการกำหนดค่าบิทซีเควนท์ให้กับอักขระแต่ละตัว

- นำอาร์เรย์ที่ได้มาเรียงตามจำนวน total ของแต่ละอักขระจากมากไปหาน้อย
- หาค่าความถี่ของอักขระแต่ละตัวเก็บไว้ในสมาชิก frequency
- WHILE ค่าความถี่ของโหนดบนสุด น้อยกว่า 0.9999
  - หาโหนดที่มีค่าความถี่ต่ำที่สุด 2 โหนดและค่าพอยเตอร์ up = Null โดยสมาชิกอาร์เรย์ทุกตัวจะถูกมองว่าเป็น โหนด แต่ละโหนด
  - สร้างโหนดขึ้นมาหนึ่งโหนด

    ให้ค่า charnum = -1

    up = Null

    frequency = ค่า frequency ของโหนดทั้งสองรวมกัน

    right ชี้ไปยังโหนดที่มีค่า frequency ต่ำกว่า

    left ชี้ไปยังโหนดที่มีค่า frequency สูงกว่า
  - ให้โหนดที่มีความถี่ต่ำที่สุดทั้งสองมีค่าพอยเตอร์ up ชี้ไปที่โหนดที่สร้างใหม่
- END_WHILE

หลังจากขั้นตอนนี้จะได้โครงสร้างต้นไม้ที่มี Root เพียงหนึ่งโหนดเท่านั้น และมี Leaves เป็นโหนดที่หมายถึงอักษรแต่ละตัวในไฟล์อินพุท ดังนั้นอักษรที่มีความถี่น้อยย่อมจะมีความลึกในโครงสร้างต้นไม้ (Depth) มากกว่าอักษรที่มีความถี่สูง และ ค่าความลึกนี้ยังกำหนดจำนวนบิทซีเควนท์ที่ใช้สำหรับแทนตัวอักษรแต่ละตัวด้วย

3. กำหนดค่าบิทซีเควนท์สำหรับแต่ละตัวอักษร และสร้าง (ตารางเพื่อใช้ในการถอดรหัส) Decode table เพื่อเก็บไว้ในไฟล์เอาท์พุทใช้ในการขยายข้อมูล ในการกำหนดค่าบิทซีเควนท์สำหรับตัวอักษรที่ได้สร้างโครงสร้างต้นไม้ขึ้นมาแล้วนั้นอาศัยการเขียนฟังก์ชันแบบเรียกตัวเองจะได้ค่าบิทซีเควนท์และจำนวนบิทซีเควนท์สำหรับแต่ละอักษรซึ่งเป็น Leaves ของโครงสร้างต้นไม้

หลังจากได้ค่าบิทซีเควนท์แล้วต่อไปจะเป็นการสร้างตารางเพื่อใช้ในการขยายไฟล์ข้อมูลต่อไป ตารางดังกล่าวนี้เป็นแทนโครงสร้างต้นไม้ด้วยอาร์เรย์เพื่อประหยัดเนื้อที่ที่ใช้ในการเก็บตารางนี้ในไฟล์เอาท์พุท โดยสมาชิกอาร์เรย์เป็นโครงสร้างข้อมูลที่ประกอบไปด้วย

charnum	เพื่อเก็บค่ารหัสของตัวอักษร
left	เพื่อเก็บค่าที่ใช้ชี้ไปยังสมาชิกอาร์เรย์ตัวทางซ้ายในโครงสร้างต้นไม้
right	เพื่อเก็บค่าที่ใช้ชี้ไปยังสมาชิกอาร์เรย์ตัวทางด้านขวาในโครงสร้างต้นไม้

โดยใช้การเขียนฟังก์ชันแบบเรียกตัวเองเพื่อการสร้างตารางนี้เช่นกัน

4. การแทนไฟล์อินพุทด้วยบิทซีเควนท์ที่ได้ลงในเอาท์พุทไฟล์ ประกอบไปด้วยขั้นตอนดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เขียนชนิดของการย่อข้อมูล 'HuF' ลงในเอาท์พุทไฟล์
- เขียนจำนวนของอาร์เรย์ที่ใช้ในตารางการถอดรหัสลงในเอาท์พุทไฟล์
- เก็บขนาดของไฟล์ข้อมูลเดิมเป็นไบต์เพื่อใช้ในการขยายข้อมูล
- เก็บตารางสำหรับการถอดรหัสลงไฟล์เอาท์พุท
- FOR 1 TO ขนาดของไฟล์ข้อมูลเดิมเป็นไบต์ DO
    - อ่านตัวอักษรจากไฟล์อินพุท
    - หาบิทซีเควนท์ที่แทนตัวอักษรนั้นจากตารางการถอดรหัสโดยใช้รหัสตัวอักษรนั้นค้นหา
    - เก็บบิทซีเควนท์นั้นลงในบัฟเฟอร์
        - IF บัฟเฟอร์นั้นครบ 8 บิท
            - เขียนบัฟเฟอร์ลงในเอาท์พุทไฟล์
- เขียนบัฟเฟอร์ที่เหลืออยู่ลงในไฟล์เอาท์พุท

การขยายข้อมูล เมื่ออ่านค่าที่เก็บไว้ในส่วนต้นของไฟล์อินพุทที่ต้องการขยายเข้ามาแล้ว ซึ่งได้แก่ค่าดังต่อไปนี้ ขนาดของไฟล์เดิม จำนวนสมาชิกอาร์เรย์ในตารางถอดรหัส ตารางถอดรหัส ต่อไปก็จะเป็นการขยายไฟล์อินพุท มีขั้นตอนดังนี้

- กำหนดค่าเริ่มต้น ให้ fastleft0 = decode_table[0].left , fastleft = fastleft0 , index = 0
- WHILE
    - อ่านตัวอักษรเข้ามาในตัวแปร buffer
    - FOR 1 TO 8 DO
        - IF buffer & 80h (เพื่อตรวจค่าบิทซีเควนท์ ใช้ในการ branch ไปตามตารางถอดรหัส)
            - ให้ค่า index = fastleft
        - ELSE
            - index = decode_table[index].right
        - ทำการชิฟ (Shift) ค่าใน buffer ไปทางซ้าย 1 ครั้ง เพื่อใช้ในตรวจสอบค่าบิทซีเควนท์ครั้งต่อไป
        - ให้ fastleft = decode_table[index].left เพื่อตรวจว่าพบตัวอักษรหรือยัง
        - IF fastleft = 0 (พบตัวอักษรในตารางถอดรหัสแล้ว)
            - เขียนค่ารหัสตัวอักษรที่พบลงในเอาท์พุทไฟล์
            - กำหนดค่าเริ่มต้นในการวนลูป คือ index = 0 , fastleft = fastleft0
            - IF จำนวนตัวอักษรที่เขียนลงในเอาท์พุทไฟล์เท่ากับจำนวนไบต์ของไฟล์เดิม
                - กระโดดออกจาก WHILE ลูป
- END_WHILE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2.3 การย่อและขยายข้อมูลโดยใช้วิธี LZW

หลักการที่ใช้ในที่นี้ใช้ขนาดของดิคชันนารีหรือตารางสตริงเท่ากับ 4096 สตริง และหลักการในการเลือกที่เก็บสตริงลงในตารางสตริงใช้การ Hashing เพื่อหาตำแหน่งในตารางสตริงซึ่งรายละเอียดในส่วนนี้จะไม่กล่าวถึงเพราะไม่เกี่ยวกับการย่อ-ขยายข้อมูล เพียงแต่นำมาประยุกต์ใช้งานเท่านั้น ขั้นตอนในการย่อข้อมูลมีดังนี้

- สร้างตารางสตริงเป็นอาร์เรย์ของโครงสร้างที่มีขนาด 4096 โดยโครงสร้างที่เป็นอาร์เรย์มีส่วนประกอบคือ
  - used เพื่อบอกว่าตำแหน่งนี้มีข้อมูลเก็บอยู่หรือไม่
  - next เพื่อใช้ในการอ้างถึงตำแหน่งอื่นในอาร์เรย์เมื่อเกิดการชนกันจาก Hashing
  - predecessor รหัสของสตริงที่เป็นส่วนหน้าของสตริงนี้
  - follower ตัวอักษรที่เป็นตัวสุดท้ายของสตริงนี้
- อ่านอินพุทไฟล์มาหนึ่งตัวอักษรเก็บใน c
- หาค่ารหัสของตัวอักษรนั้นจาก Hashing เก็บไว้ในตัวแปร code
- WHILE อ่านค่าอินพุทไฟล์มาอีกหนึ่งตัวอักษรเก็บใน c โดยไม่สิ้นสุดไฟล์
  - IF ให้ค่ารหัสของสตริงที่มี predecessor = code ,follower=c เป็น localcode พบในตารางสตริง แสดงว่าสตริงที่ประกอบด้วยสตริงรหัสที่ตรงกับ code และตามด้วยอักษรใน c นั้นมีเก็บอยู่ในตารางสตริงอยู่แล้ว
    - code = localcode
    - ไปเริ่มต้นลูปใหม่
  - ELSE พบสตริงที่ยังไม่เคยถูกเก็บในตารางสตริงแล้ว
    - เขียน code ลงในไฟล์เอาท์พุท
    - IF จำนวนสตริงที่เก็บในตารางสตริงยังไม่เต็ม
      - ใส่สตริงใหม่นี้ลงในตารางสตริง
    - code = ค่ารหัสของสตริงใหม่ที่ได้จากการเก็บลงในตารางสตริง
- END_WHILE
- เขียนรหัสที่เหลือลงในเอาท์พุทไฟล์

การขยายไฟล์ข้อมูลโดยใช้วิธี LZW นี้ เนื่องจากตารางสตริงไม่ได้ถูกเก็บลงมาในไฟล์ที่ถูกย่อไว้แล้ว ในส่วนการขยายไฟล์ต้องสร้างตารางสตริงในขณะที่ทำการขยายไฟล์ไปด้วย ด้วยหลักการที่ใช้คือสตริงจะถูกแทนที่ด้วยรหัสเมื่อตัวอักษรในสตริงนั้นประกอบไปด้วยสตริงที่เคยมีอยู่ในตารางแล้วและตัวอักษรเพิ่มอีกหนึ่งตัวและต้องเป็นสตริงที่ไม่เคยถูกเก็บในตารางสตริงมาก่อน ดังนั้นการหาสตริงเพื่อมาแทนรหัสที่ใช้ต้องอาศัยการทำงานของแสตกเพื่อให้ได้สตริงที่ต้องการ การขยายข้อมูลมีขั้นตอนดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

□code = oldcode = c = รหัสเริ่มต้นจากอินพุทไฟล์

□เพราะเป็นรหัสตัวแรกยังไม่มีการเข้ารหัสจึงเขียนรหัสตัวแรกลงไฟล์เอาท์พุทได้เลย

□กำหนดค่าเริ่มต้น finchar = c

□WHILE code = incode = อ่านรหัสตัวต่อไปจากอินพุทไฟล์ ยังไม่สิ้นสุดไฟล์

□กำหนดตำแหน่งตัวชี้ไปยังตารางสตริงเป็นตำแหน่งของ code

□IF ณ ตำแหน่งที่ชี้ไปนั้นว่าง แสดงว่ารหัสที่ได้นั้นเป็นรหัสที่ไม่มีอยู่ในตารางสตริงมาก่อน

□lastchar = finchar , code = oldcode , unknow = TRUE

□กำหนดตำแหน่งตัวชี้ไปยังตารางสตริงเป็นตำแหน่งของ code

□WHILE predecessor ณ ตำแหน่งของตัวชี้ตารางสตริงยังมีรหัสใดๆอยู่

□เก็บตัวอักษร follower ลงในแสตก

□code = predecessor ณ ตำแหน่งนั้น

□กำหนดตำแหน่งตัวชี้ไปยังตารางสตริงเป็นตำแหน่งของ code

□เขียน follower ณ ตำแหน่งตัวชี้ลงในเอาท์พุทไฟล์ เพราะลูป WHILE ข้างบนนี้จะสิ้นสุดเมื่อ follower = อักษรตัวแรกของสตริงที่เราต้องการถอดรหัส

□WHILE แสตกไม่ว่าง

□ป๊อปอักษรในแสตกออกมา

□เขียนอักษรที่ได้จากแสตกลงเอาท์พุทไฟล์

□IF unkwon (เป็นรหัสที่ไม่รู้จักมาก่อน แสดงว่าอักษรก่อนหน้านั้นเป็นอักษรตัวสุดท้ายในรหัสนั้น)

□เขียนอักษรใน lastchar ลงในเอาท์พุทไฟล์

□IF ตารางสตริงยังไม่เต็ม

□เก็บรหัสที่ได้ลงในตารางสตริง

□oldcode = incode เพื่อเก็บค่ารหัสสุดท้ายที่รู้จักเอาไว้ใช้ในลูปต่อไป

□END_WHILE

3.2.3 โปรแกรมการแสดงผลรูปภาพ

สำหรับโปรแกรมในส่วนการแสดงผลรูปภาพเป็นโปรแกรมซึ่งสามารถแสดงผลรูปภาพใน TIFF Format ได้เท่านั้น โดยทำการเปิดไฟล์รูปภาพ อ่าน Header ของไฟล์ อ่านค่าที่เก็บในไดเรกทอรี และตั้งค่าทรัพยากรที่ต้องใช้ตามค่าที่อ่านได้ จากนั้นจะสร้างอิมเมจขึ้นจากข้อมูลไฟล์นั้นเพื่อใช้แสดงรูปภาพต่อไป และมีข้อจำกัดคือสามารถแสดงผลสำหรับ 5.0 TIFF Specification ใน คลาส B , G , P , R

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
สรุปผลการทำงาน

4.1 ผลที่ได้และแนวทางการพัฒนาระบบในอนาคต

ระบบจัดการรูปภาพนี้นับว่าเป็นระบบที่สร้างขึ้นด้วยหลักการง่ายๆ แต่ต้องใช้ความรู้ในหลายๆ ด้านจึงจะสามารถที่จะสร้างระบบที่สมบูรณ์ขึ้นได้ ผลการทำงานที่ได้รับจากระบบในด้านต่างๆ

1. การติดต่อกับผู้ใช้ เนื่องจากระบบไม่จำเป็นต้องติดต่อกับผู้ใช้มากนัก ดังนั้นการใช้เอกซ์ลิปฟังก์ชันก็เป็นการพอเพียงแล้วสำหรับการเขียนโปรแกรมในส่วนที่ติดต่อกับผู้ใช้ ซึ่งก็ให้ความสะดวกและใช้งานง่ายได้ดีพอสมควร เพื่อเพิ่มความสะดวกแก่ผู้ใช้อาจสร้างส่วนการโหลดไฟล์อินพุทให้สามารถแสดงไฟล์ในไดเรกทอรีต่างๆ เพื่อให้ผู้ใช้สะดวกในการเลือกไฟล์มาใช้งานมากขึ้น

2. การติดต่อระหว่างโปรเซส การใช้สัญญาณติดต่อระหว่างโปรเซสนี้เป็นการติดต่อขั้นพื้นฐานซึ่งให้รายละเอียดในการติดต่อได้น้อย ดังนั้นควรปรับปรุงโดยการใช้ลอกแมสเซสของระบบ หรือ ระบบ ไอพีซี (Interprocess Communication) อื่นๆในการแลกเปลี่ยนข้อมูลกันระหว่างโปรเซส

3. การแสดงผลรูปภาพสามารถแสดงผลรูปภาพได้ชัดเจนและไม่มีข้อผิดพลาด ในอนาคตควรจะมีการแสดงผลรูปภาพในฟอร์เมทอื่นๆ รวมทั้งเท็กซ์ไฟล์ได้ด้วย

4. การย่อและการขยายไฟล์ต้องมีการปรับปรุงทั้งประสิทธิภาพให้ดีกว่านี้ จากการทำงานจริงสามารถย่อไฟล์ได้ประมาณ 40 เปอร์เซ็นต์ของขนาดไฟล์เดิม ซึ่งยังเป็นอัตราการย่อข้อมูลที่ไม่ดีนัก และจำนวนวิธีของการย่อ-ขยายข้อมูลควรมีจำนวนมากกว่านี้ รวมทั้งตัวเลือกที่ผู้ใช้ต้องการก็ควรมีมากขึ้นเช่นกัน

5. การทำงานด้านแบคกราวน์โปรเซส สามารถทำงานได้อย่างดี ถ้าหากในอนาคตมีจำนวนวิธีในการย่อข้อมูลเพิ่มขึ้นควรจะสร้างโปรเซสสำหรับแต่ละวิธีขึ้นมาด้วยเพื่อจะได้ประหยัดเวลาในการทำงานลงอีก

6. เนื่องจากระบบนี้ได้พัฒนาบนยูนิกซ์ซึ่งสามารถใช้งานโปรแกรมได้พร้อมๆกันหลายคน ดังนั้นควรมีการพัฒนาในการใช้งานร่วมกันได้ด้วย ทั้งการใช้งานไฟล์รูปภาพร่วมกัน การล็อกไฟล์เมื่อต้องการย่อหรือขยายไฟล์ใดๆ

7. การเขียนโปรแกรมส่วนมากจะใช้ ตัวแปรแบบโกลบอลทำให้สิ้นเปลืองหน่วยความจำในระบบในการทำงาน หากจะต้องพัฒนาระบบใหญ่มากขึ้นกว่านี้ควรปรับปรุงการเขียนโปรแกรมให้มีการใช้ตัวแปรแบบโลคอล

สำหรับระบบที่ได้พัฒนาขึ้นมานี้เป็นระบบที่สามารถใช้งานได้จริงในระดับหนึ่งเท่านั้น เนื่องจากระบบที่ต้องการนั้นมีส่วนประกอบในการทำงานเป็นจำนวนมาก จึงไม่สามารถที่จะสร้างระบบให้ทุกส่วนทำงานได้ตามต้องการ ระบบที่ได้นี้จึงเหมาะสมกับการนำไปพัฒนาต่อไปในอนาคตเพื่อการใช้งานจริง ซึ่งเมื่อระบบได้พัฒนาจนบรรลุจุดประสงค์ที่กำหนดไว้แล้วจะเป็นระบบที่อำนวยความสะดวกให้แก่ผู้ใช้และประหยัดเวลาและเนื้อที่ในการจัดการรูปภาพลงไปได้มาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 ปัญหาที่พบในการพัฒนาระบบ

ในการพัฒนาระบบปัญหาที่พบได้แก่

1. รายละเอียดการเขียนโปรแกรมบนเอกซ์วินโดว์หาได้ยาก ทำให้การพัฒนาระบบเป็นไปด้วยความล่าช้า และไม่มีประสิทธิภาพ

2. การพัฒนาโปรแกรมบนเอกซ์วินโดว์โดยใช้เอกซ์ลิปฟังก์ชัน ซึ่งมีรายละเอียดปลีกย่อยในการเขียนโปรแกรมมาก ถ้าหากต้องการพัฒนาระบบให้มีขนาดใหญ่และมีความสะดวกรวดเร็วในการเขียนโปรแกรมควรใช้ทูลคิตท์และวิดเจทแทน

3. ระบบต้องการความรู้ในหลายๆด้าน ทำให้ต้องใช้เวลาในการศึกษาค้นคว้าเพื่อที่จะสร้างแต่ละส่วนของระบบเป็นเวลานาน เช่น การย่อ-ขยายข้อมูล การสร้างกราฟิคติดต่อกับผู้ใช้

4. เนื่องจากระบบนี้ได้สร้างขึ้นเพื่อใช้งานบนลีนุกซ์ ซึ่งหารายละเอียดในการใช้งาน การเขียนโปรแกรม การทำงานของระบบได้ยาก รวมทั้งระบบมีความไม่เข้ากันกับระบบยูนิกซ์ด้วย ทั้งรายละเอียดการทำงานของระบบ และด้านการเขียนโปรแกรมฟังก์ชันที่ใช้บางครั้งก็ไม่เหมือนกัน

5. การที่ขาดอุปกรณ์ในการพัฒนาระบบ เช่น เท็กซ์อิดิเตอร์ที่มีประสิทธิภาพและใช้งานสะดวกกว่าที่มีอยู่ เครื่องมือช่วยในการดีบักโปรแกรม ทำให้การพัฒนาระบบเป็นไปด้วยความล่าช้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ผู้จัดทำขอขอบพระคุณบุคคลต่างๆ ที่มีส่วนช่วยเหลือ สนับสนุน และให้กำลังใจให้โครงงานนี้สามารถสำเร็จลงได้ด้วยความเรียบร้อย ดังต่อไปนี้

1. อาจารย์วัชระ ฉัตรวิริยะ อาจารย์ที่ปรึกษา ผู้ให้คำปรึกษา ให้ข้อมูลที่ใช้ในการพัฒนาระบบ และชี้แนะแนวทางในการสร้างระบบนี้ขึ้นมา

2. โอ๊ต , นะ , ลี , ชู และเพื่อนทุกๆคนที่คอยให้กำลังใจ และความช่วยเหลือในการช่วยจัดพิมพ์ปริญญานิพนธ์นี้บางส่วน

3. พี่ศิริ ที่ให้ความช่วยเหลือชี้แนะในการพัฒนาโปรแกรมบางส่วน และให้ยืมหนังสือมาใช้ในการทำปริญญานิพนธ์นี้ด้วย

4. "J.R.Suckling" <bobs@cbda9.apgea.army.mil> ผู้ให้คำอธิบาย ให้คำปรึกษา และให้กำลังใจในการเขียนโปรแกรมมาโดยตลอด

5. Jason G Weinstein <weinstei@nettech.com> ผู้ให้คำอธิบาย ให้คำปรึกษา และให้กำลังใจในการเขียนโปรแกรมมาโดยตลอดเช่นกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

1. Hewlett-Packard Company, "X Window System Programming : Xlib Student Workbook", Hewlett-Packard Company, USA, 1989

2. Hewlett-Packard Company, "Programming with Xlib", Hewlett-Packard Company, USA, 1991

3. Bilbert Held and Thomas R. Marshall, "Data compression : techniques and applications : hardware and software considerations", John Wiley & Sons, 1991

4. Mark Nelson, "The Data Compression Book", M&T Publishing, 1992

5. W.Richard stevens, "Unix Network Programming", Prentice Hall, 1991

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้