ระบบส่งข่าวสารในรูปแบบของเสียงผ่านระบบอินเตอร์เน็ต

Internetworking Speech Messaging System

โดย

นายจิตพล จุลศรีไกวัล 36014076

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

วัน เดือน ปี 1 ต.ค. 2541
เลขทะเบียน 038323
เลขเรียกหนังสือ T 31943 1999 ร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ปีการศึกษา 2539

ระบบส่งข่าวสารในรูปแบบของเสียงผ่านระบบอินเตอร์เน็ต

Internetworking Speech Messaging System

โดย

นายจิตพล จุลศรีไกวัล 36014076

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบการส่งข่าวสารในรูปแบบเสียงผ่านระบบอินเตอร์เน็ต

Internetworking Speech Messaging System

ผู้จัดทำ

นายจิตพล จุลศรีไกวัล รหัสประจำตัวนักศึกษา 36014076

.....อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบส่งข่าวสารในรูปแบบของเสียงผ่านระบบอินเตอร์เน็ต

นายจิตพล จุลศรีไกวัล 36014076
อาจารย์ที่ปรึกษา
อาจารย์ ธนา หงษ์สุวรรณ

บทคัดย่อ

เนื่องจากในปัจจุบันระบบอินเตอร์เน็ตเข้ามามีบทบาทในชีวิตประจำวันมากขึ้นทั้งในด้านของธุรกิจ การศึกษา บันเทิง ศิลปวัฒนธรรม และการติดต่อสื่อสารในระบบจดหมายอิเล็กตรอนิกส์หรือ E-Mail ซึ่งเป็นวิธีการติดต่อสื่อสารที่มีราคาถูก สะดวก รวดเร็ว และสามารถติดต่อได้ผ่านระบบเครือข่ายอินเตอร์เน็ตซึ่งมีอยู่ทั่วโลก นอกจากนี้ยังสามารถส่งข้อมูลชนิดอื่น ๆ ที่ไม่เป็นตัวอักษรได้เช่น โปรแกรมคอมพิวเตอร์ รูปภาพ รวมไปถึงเสียง

อย่างไรก็ตามการส่งจดหมายเป็นข้อความธรรมดา อาจเกิดความผิดพลาดในการตีความได้ แต่ถ้าเป็นการส่งในลักษณะข้อมูลที่เป็นเสียงก็จะช่วยลดจุดด้อยนี้ได้ แต่อาจต้องใช้โปรแกรมอื่น ๆ เข้ามาช่วยในการบันทึกและแสดงข้อมูลเสียง อีกทั้งต้องปรับแต่งรูปแบบข้อมูลเสียงให้มีขนาดกะทัดรัดเพื่อส่งในเครือข่ายอินเตอร์เน็ต ทำให้เสียค่าใช้จ่ายในการจัดหาโปรแกรมดังกล่าว เกิดความยุ่งยากและเสียเวลาในการรับส่งข้อมูลเสียงแต่ละครั้ง โครงงานนี้มีจุดประสงค์จะสร้างโปรแกรมที่สามารถรับและส่งข่าวสารผ่านเครือข่ายอินเตอร์เน็ต โดยที่สามารถรับและส่งได้ทั้งข้อความ เสียง หรืออาจรวมถึงภาพด้วย

ทั้งนี้ระบบจะต้องเป็นไปตามข้อกำหนดของ SMTP และ POP3 โดยสื่อสารผ่านโปรโตคอล TCP/IP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Internetworking Speech Messaging System

Jittapol Julasrikaiwan 36014076

Advisor

Mr. Thana Hongsuwan .

ABSTACT

Because of internet now take part in our life all together in business,education,entertainment and culture.And communication with electronic mail or E-mail is cheap,fast and conventttient as well as it can contact with other internet system around the world,beside,it still can send other kinds of data except alphabet,like computer programs,picture and sound

However sending mail with normal message may cause a mistake in interpretation but if we send data in sound,it will case this problem.But to use thihs way we must have other programs to recording and annouce sound data and compact the format of sound data for sending in internet.These cause expense in providing programs,complication and take time in transporting sound data.The propose of this project is to make the program which can send and receive data which can be message,sound and picture pass internet.

All of this system must follow the regulation of SMTP and POP3 which communicate pass TCP/IP protocal.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1 บทนำ

1.1 ระบบ E-mail

ข้อความของ E-mail จะถูกขนส่งโดยระบบขนส่งข้อความ (Massage Transfer System : MTS) ซึ่งประกอบด้วยตัวแทนขนส่งข้อความ (Massage Transfer Agent : MTA) อยู่จำนวนหนึ่ง MTS จะถูกแบ่งแยกเป็นหลายรูปแบบและไม่อยู่ภายใต้การควบคุมจากหน่วยงานใดหน่วยงานหนึ่ง ในทางกลับกัน MTA จะถูกควบคุมโดยหน่วยงานที่แน่นอนหน่วยงานใดหน่วยงานหนึ่งในแต่ละระบบ ในส่วนของตัวแทนผู้ใช้(User Agent : UA) จะประกอบด้วย ผู้ใช้และส่วนที่ติดต่อกับ MTA ของเครือข่ายของตนเอง

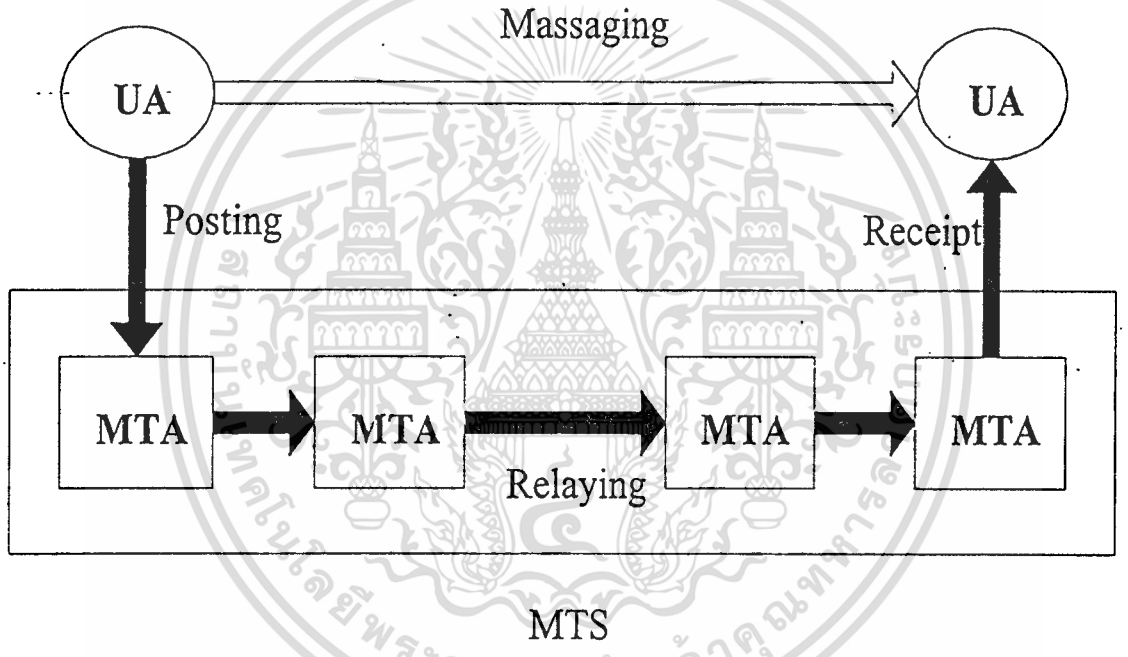
เมื่อข้อความของ E-mail ถูกส่งจากผู้ใช้คนหนึ่งไปยังผู้ใช้อีกคนหนึ่งจะเกิดการทำงานดังนี้ : ฝ่ายส่งจะแสดงที่อยู่ของผู้ส่งและที่อยู่ปลายทางของผู้รับ และหลังจากนั้นจะส่งข้อความเข้าสู่ MTA ซึ่งจะมีการพิจารณาความถูกต้องของที่อยู่และวากยสัมพันธ์(Syntax)ของข้อความที่จะส่ง เมื่อเสร็จสิ้นในการติดต่อแล้ว MTA จะส่งผลสนองกลับไปเพื่อให้ทราบว่าจะทำการส่งข้อความต่อไปหรือไม่ หรือการส่งข้อความล้มเหลวก็จะแจ้งให้ผู้ส่งทราบโดยทำเป็นรายงานความผิดพลาด(Error Report) หลังจากส่งผลตอบสนองกลับไปแล้ว MTA จะทำการตัดสินใจว่าสามารถส่งข้อความไปให้ผู้รับโดยตรงได้หรือไม่ ถ้าได้จะทำการส่งไปยัง MTA ของผู้รับทันที ถ้าไม่ได้จะติดต่อกับ MTA ที่อยู่ติดกันและอยู่ใกล้ผู้รับมากกว่า และส่งผ่านข้อความไปเรื่อย ๆ จนกระทั่งข้อความส่งถึง MTA ของผู้รับ

โดยสรุปแล้ว มีโปรโตคอลอยู่ 3 โปรโตคอลที่เกี่ยวข้องกับระบบ E-mail ได้แก่

1. โปรโตคอล "Massaging" ใช้ระหว่าง UA 2 ที่
2. โปรโตคอล "Relaying" ใช้ระหว่าง MTA
3. โปรโตคอล "Submission/Delivery" ใช้ระหว่าง MTA และ UA

ซึ่งโปรโตคอลเหล่านี้สามารถใช้โปรโตคอล SMTP(Simple Mail Transfer Protocol) และ POP3(Post Office Protocol version 3) ในการทำงานของ E-mail ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.1 ระบบของอิเลคทรอนิคเมล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2 ทฤษฎีและหลักการ

2.1 Simple Mail Transfer Protocol (SMTP)

2.1.1 บทนำ

จุดประสงค์ของซิมเปิ้ลเมล์ทรานเฟอร์โปรโตคอล (Simple Mail Trasnfer Protocol) หรือ SMTP คือการส่งเมล์ได้อย่างถูกต้องและมีประสิทธิภาพ โดยที่ SMTP จะไม่สนใจระบบของการสื่อสารแต่จะต้องการเพียงลำดับของข้อมูลที่ถูกต้อง

ลักษณะสำคัญของ SMTP คือความสามารถในการถ่ายทอดเมล์ (mail) ข้ามระบบของการ สื่อสาร ซึ่งระบบสื่อสารต่าง ๆ นั้นจะทำการเตรียม "อินเตอร์โพรเซสคอมมิวนิเคชั่นเอนไวรอนเมนท์(Interprocess Communication Environment) หรือ "IPCE" ซึ่งอาจจะครอบคลุมเครือข่ายเดียว , หลายเครือข่ายหรืออาจจะเป็นเครือข่ายย่อย ๆ ในเครือข่ายใด ๆ และที่สำคัญที่สุด คือระบบการติดต่อจะไม่เป็น 1 ต่อ 1 กระบวนการต่าง ๆ สามารถติดต่อโดยตรงกับกระบวนการอื่น ๆ ผ่านทาง IPCE ซึ่ง เมล์ก็เป็นกระบวนการตัวอย่างอันหนึ่ง เมล์สามารถติดต่อระหว่างกระบวนการใน IPCE ที่แตกต่างกัน โดยทำการถ่ายทอดผ่านกระบวนการซึ่งถูกติดต่ออยู่กับหลาย ๆ IPCE เมล์สามารถถูกถ่ายทอดระหว่าง โฮสท์ กับ โฮสท์ ในระบบการติดต่อที่ต่างกันได้

2.1.2 โครงสร้างของ SMTP

การออกแบบ SMTP อยู่บนพื้นฐานของการติดต่อสื่อสาร ความต้องการของผู้ใช้ , SMTP ตัวส่งจะสร้างการติดต่อ 2 ทิศทางไปยัง SMTP ตัวรับ ซึ่งอาจจะเป็นตัวรับสุดท้ายหรือเป็นตัวที่อยู่ระหว่างทางก็ได้ คำสั่งของ SMTP จะถูกสร้างจากตัวส่งและถูกส่งไปยังตัวรับ จากนั้นตัวรับจะส่งผลของการรับคำสั่งมาเพื่อตอบสนองกลับไปยังตัวส่ง

เมื่อช่องทางการติดต่อถูกสร้างขึ้น ตัวส่งจะส่งคำสั่ง "MAIL" เพื่อแสดงว่าจะมีการส่งเมล์ต่อไป ถ้าตัวรับพร้อมที่จะรับเมล์ได้จะส่งคำตอบ "OK" หลังจากนั้นตัวส่งจะส่งคำสั่ง "RCPT" เพื่อแสดงว่าได้รับผลตอบสนองแล้ว ถ้าตัวรับได้รับคำสั่งนี้และยอมรับก็จะส่ง "OK" กลับมา ถ้าไม่ได้จะตอบปฏิเสธการได้รับ ซึ่งทั้งตัวส่งและตัวรับอาจมีการติดต่อกันหลายครั้ง เมื่อมีการตอบรับการติดต่อตัวส่งจะทำการส่งข้อมูลของเมล์ ถ้าตัวรับได้รับเมล์ทั้งหมดจะตอบรับด้วย "OK"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SMTP จะมีการเตรียมกลไกของการติดต่อสื่อสารของเมล์จากตัวส่งไปยังตัวรับเมื่อทั้งสองฝ่ายติดต่ออยู่ภายในระบบเดียวกัน หรือทำการถ่ายทอดไปยังระบบอื่นถ้าทั้งลองติดต่อคนละระบบ

เพื่อที่จะสามารถเตรียมการถ่ายทอดได้ SMTP เซิร์ฟเวอร์จะต้องถูกกำหนดชื่อของจุดหมายปลายทางหรือที่รู้จักกันดีว่า "Destination Mailbox Name" สิ่งที่ต้องระบุในคำสั่ง "MAIL" คือ รีเวิร์สพาธ (reverse-path) ซึ่งแสดงว่าใครเป็นคนสร้างเมล์และสิ่งที่ต้องระบุลงในคำสั่ง "RCPT" คือ ฟอร์เวิร์ดพาธ(forward-path) ซึ่งแสดงว่าเมล์ถูกส่งไปให้ใคร

เมื่อข่าวสารเดียวกันถูกส่งไปยังตัวรับหลาย ๆ ที่ SMTP จะช่วยในการติดต่อโดยจะทำการขนส่งข้อมูลชุดเดียวไปยังตัวรับทั้งหมดในจุดหมายปลายทางเดียวกัน

คำสั่งและคำตอบของ SMTP จะมีวากยสัมพันธ์ (syntax) ตายตัว ซึ่งคำตอบจะอยู่ในรูปแบบของโค้ดตัวเลข ซึ่งทั้งคำสั่งและตำตอบจะไม่สนใจในลักษณะของการเขียน อาจเป็นตัวอักษรตัวใหญ่ , ตัวเล็กหรืออาจจะปนกันก็ได้ ซึ่งจะแตกต่างจากชื่อของเมล์บ็อกซ์ (mailbox) ของโฮสท์ซึ่งจะสนใจในเรื่องของขนาดตัวอักษร

2.1.3 SMTP โพรซีเยอร์

จะประกอบด้วยส่วนต่าง ๆ ดังนี้

2.1.3.1 MAIL

จะมีรายละเอียดทั้งหมด 3 ขั้นตอน โดยขั้นแรกจะเริ่มจากคำสั่ง "MAIL" ซึ่งให้ตัวส่งแจ้งให้ทราบว่าจะมีการติดต่อ ต่อมาคือคำสั่ง "RCPT" เพื่อยืนยันความพร้อมของตัวรับ และสุดท้ายคือคำสั่ง "DATA" ซึ่งเป็นการส่งข้อมูลและแจ้งการสิ้นสุดของการส่งข้อมูล มีรายละเอียดดังนี้

1. คำสั่ง "MAIL"

- MAIL<SP>FROM:<รีเวิร์สพาธ><CRLF>

คำสั่งนี้จะแจ้งให้ตัวรับรับทราบว่าการส่งเมล์จะเริ่มต้นขึ้นและจะทำการเคลียสถานะต่าง ๆ ทางตัวรับ และจะมีการระบุชื่อของตัวส่งเพื่อที่ตัวรับจะสามารถส่งผลตอบสนองกลับมาได้ ซึ่งถ้าตัวรับตอบรับจะส่งค่า "250"

2. คำสั่ง "RCPT"

- RCPT<SP>TO:<ฟอร์เวิร์ดพาธ><CRLF>

คำสั่งนี้ เป็นการยืนยันความพร้อมของตัวรับ ถ้าตัวรับพร้อมจะส่งค่า " 250" และเก็บชื่อของตัวส่งเอาไว้ ถ้าไม่พร้อมจะส่งค่า "550" ซึ่งขั้นตอนนี้สามารถทำได้หลายครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. คำสั่ง "DATA"

ถ้าตัวรับตอบรับจะส่งค่า "354" และพิจารณาข้อความ เมื่อสิ้นลุดการรับส่งจะส่งค่า "250" เมื่อข้อมูลของเมล์ถูกส่งบนช่องทางการติดต่อจะมีการส่งข้อมูลที่ระบุว่าสิ้นสุดข้อความแล้วบางอย่างที่แตกต่างจากข้อความของข้อมูลในเมล์

2.1.3.2 FORWARDING

จะมีบางกรณีที่ข้อมูลปลายทางใน <ฟอร์เวิร์ดพาธ> ไม่ถูกต้องแต่ตัวรับรู้ปลายทางที่ถูกต้อง ในกรณีนี้หนึ่งในคำตอบต่อไปนี้จะตอบกลับมายังตัวส่งเพื่อแจ้งให้ทำการติดต่อใหม่

- 251 User not local ; will forward to <forward-path>

คำตอบนี้จะแจ้งให้ตัวส่งทราบว่าตัวรับทราบปลายทางที่ถูกต้องและตัวรับจะทำการติดต่อใหม่ต่อไป

- 551 User not local ; please try <forward-path>

จะต่างจากอันแรกคือตัวรับจะหยุดการติดต่อโดยจะแจ้งปลายทางกลับไปให้ตัวส่งรับทราบเพื่อให้ตัวส่งทำการติดต่อใหม่

2.1.3.3 VERIFYING AND EXPANDING

จะต้องมีคำสั่งที่ใช้ตรวจสอบชื่อของผู้ใช้หรือรายชื่อทั้งหมดของเมล์บ็อกซ์ คือคำสั่ง "VRFY" และ "EXPN" ตามด้วยชื่อที่ต้องการตรวจสอบ สำหรับคำสั่ง "VRFY" ชื่อที่จะตรวจสอบคือชื่อของผู้ใช้ และผลตอบสนองอาจเป็นชื่อเต็มของผู้ใช้ (ถ้ารู้) และชื่อเมล์บ็อกซ์ของผู้ใช้ สำหรับคำสั่ง "EXPN" ชื่อที่ใช้ตรวจสอบคือรายชื่อผู้ใช้ในเมล์บ็อกซ์ ซึ่งผลตอบสนองจะรวมชื่อเต็มของผู้ใช้ (ถ้ารู้) และเมล์บ็อกซ์ ของรายชื่อนั้น ซึ่งอาจมีคำตอบหลายบรรทัด

2.1.3.4 SENDING AND MAILING

การส่งเมล์ไปยังเมล์บ็อกซ์จะเรียกว่า "Mailing" การส่งเมล์ไปยังเทอร์มินอล (terminal) เรียกว่า "Sending" ซึ่ง SMTP มีคำสั่ง 3 คำสั่งที่สนับสนุน "Sending"

- SEND<SP>FROM:<รีเวิร์สพาธ><CRLF>

คำสั่ง "SEND" จะสำเร็จก็ต่อเมื่อข้อความถูกส่งถึงเทอร์มินอล

- SOML<SP>FROM:<รีเวิร์สพาธ><CRLF>

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Send Or MaiL คำสั่งนี้จะสำเร็จก็ต่อเมื่อข้อความถูกส่งไปยังเมล์บ็อกซ์ หรือ เทอร์มินอล อย่างใดอย่างหนึ่งก็ได้

- SAML<SP>FROM:<รีเวิร์สพาธ><CRLF>

Send And MaiL คำสั่งนี้จะสมบูรณ์ก็ต่อเมื่อข้อความถูกส่งไปจนถึงเมล์บ็อกซ์

2.1.3.5 OPENING AND CLOSING

การเริ่มและสิ้นสุดการติดต่อใช้คำสั่งต่อไป

- HELO<SP><โดเมน (domain)><CRLF>

ใช้เริ่มต้นการติดต่อ

- QUIT<CRLF>

ใช้ยกเลิกการติดต่อ

2.1.3.6 RELAYING

เมื่อตัวส่งส่งคำสั่งออกไปถึงตัวรับตัวใดตัวหนึ่งซึ่งยังไม่ใช่ปลายทางที่แท้จริง. ตัวรับนั้นจะต้องทำการส่งต่อไปยังตัวรับอื่น ๆ ต่อไปเรื่อยจนกว่าจะถึงปลายทางที่แท้จริงโดยการที่ตัวรับจะทำการเปลี่ยนแปลงคำสั่งโดยการย้ายฟอร์เวิร์ดพาธไปยังจุดเริ่มต้นของรีเวิร์สพาธ จากนั้นตัวรับกลายเป็นตัวส่งและทำการสร้างเส้นทางการติดต่อและทำการส่งเมล์ต่อไป

ถ้า SMTP เซิร์ฟเวอร์ตกลงที่จะถ่ายทอดเมล์ และต่อมาทราบว่าฟอร์เวิร์ดพาธไม่ถูกต้อง หรือว่า เมล์ไม่สามารถส่งต่อได้มันจะสร้างข้อความแจ้งให้ทราบว่าเมล์นั้นส่งไปไม่ได้กลับมายังแหล่งกำเนิดซึ่งจะต้องสร้างเซิร์ฟเวอร์ที่โฮสท์ที่เกิดขึ้น ซึ่งแน่นอนว่าเซิร์ฟเวอร์ไม่สามารถแจ้งข้อความเกี่ยวกับข้อที่ทำผิดพลาดได้ การแก้ปัญหานี้อาจใช้คำสั่ง "MAIL" ในรูปแบบต่อไปนี้

MAIL<SP>FROM:<><CRLF>

2.1.3.7 DOMAINS

ชื่อของโฮสท์จะถูกแทนด้วยโดเมนและแสดงเป็นข้อความหรือตัวเลขที่คั่นด้วยจุดโดยเรียงลำดับตามความสำคัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3.8 CHANGING ROLES

คำสั่ง "TURN" ใช้ในการเปลี่ยนสถานะของตัวรับและตัวส่ง

ถ้าโปรแกรม A เดิมเป็นตัวส่งเมื่อส่งคำสั่ง "TURN" และได้รับค่า "250" โปรแกรม A จะกลายเป็นตัวรับ

ถ้าโปรแกรม B เดิมเป็นตัวรับ เมื่อได้รับคำสั่ง "TURN" และส่งค่า "250" กลับไป โปรแกรม B จะกลายเป็นตัวส่ง

การตอบปฏิเสธคำสั่ง "TURN" ตัวรับจะส่งค่า "502" กลับไป

2.1.4 THE SMTP SPECIFICATIONS

2.1.4.1 คำสั่งของ SMTP

2.1.4.1.1 COMMAND SEMATICS

คำสั่งของ SMTP จะระบุการส่งเมล์ตามที่ผู้ใช้ต้องการ คำสั่งของ SMTP จะเป็นตัวอักษรมาเรียงกันเป็นข้อความและจบด้วย <CRLF> และในคำสั่งเดียวกันจะแยกจากกันโดย <SP> ถ้ามีส่วนที่ตามมาและจบด้วย <CRLF>

ระบบปฏิบัติการของเมล์จะนำไปสู่ข้อมูลหลายประเภทที่ติดต่อ ขึ้นอยู่กับความแตกต่างของคำสั่งรีเวิร์สพาธเป็นสิ่งที่ต้องมีตามหลังคำสั่ง "MAIL" และฟอร์เวิร์ดพาธก็เป็นสิ่งที่ต้องมีตามหลังคำสั่ง "RCPT" และข้อความของเมล์เป็นสิ่งที่ตามมาจากคำสั่ง "DATA" ส่วนต่าง ๆ เหล่านี้จะต้องถูกส่งต่อและต้องยืนยันการติดต่อจนกว่าจะประกาศหยุดการติดต่อ ซึ่งการที่จะทำเช่นนี้จำเป็นต้องใช้บัฟเฟอร์ (buffer) เพื่อเตรียมรับข้อมูลแต่ละประเภท ได้แก่ รีเวิร์สพาธบัฟเฟอร์ , ฟอร์เวิร์ดพาธบัฟเฟอร์ และ เมล์ดาต้าบัฟเฟอร์ (mail data buffer)

รายละเอียดของคำสั่งแต่ละคำสั่งและผลกระทบต่าง ๆ ที่เกิดขึ้นกับสถานะและบัฟเฟอร์มีดังนี้

HELLO (HELO)

คำสั่งนี้จะส่งจากตัวส่งไปยังตัวรับซึ่งจะมีชื่อของโฮสท์ของตัวส่งส่งไปด้วย เมื่อตัวรับได้รับคำสั่งนี้แล้วจะตอบสนองกลับไปยังตัวส่ง เมื่อทำการยืนยันแน่นอนแล้วจะแสดงว่าทั้งตัวส่งและตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รับพร้อมที่จะเริ่มต้นการติดต่อแต่จะยังไม่มีการทำงานใด ๆ ทั้งสิ้น สถานะและบัฟเฟอร์ต่าง ๆ จะถูกเคลียร์

MAIL (MAIL)

คำสั่งนี้ใช้ในการเริ่มต้นการติดต่อ ซึ่งข้อมูลเมล์จะถูกส่งไปยังเมล์บ็อกซ์อื่น ๆ คำสั่งนี้จะระบุถึง รีเวิร์สพาธ ซึ่งประกอบด้วยรายชื่อของโฮสท์ และ เมล์บ็อกซ์ของตัวส่ง เมื่อรายชื่อของโฮสท์ถูกแสดงมันจะกลายเป็นเส้นทางเริ่มต้นและระบุว่าเมล์จะถูกส่งผ่านไปยังทุก ๆ โฮสท์ที่อยู่ในรายการ รายการนี้จะถูกใช้เหมือนเป็นจุดเริ่มต้นเพื่อที่จะส่งย้อนกลับได้ เพื่อแจ้งให้ทราบถึงผลตอบสนอง ขณะที่โฮสท์ทุก โฮสท์ทำหน้าที่เปลี่ยนตัวเองเป็นตัวเริ่มต้นของรายการต่อไป มันจะต้องใช้ชื่อของตัวมันเองที่รู้จักใน IPCE จากเมล์ที่ส่งมา

คำสั่งนี้จะเคลียร์บัฟเฟอร์ทั้งหมดและใส่ข้อมูลของรีเวิร์สพาธจากคำสั่งนี้ลงในรีเวิร์สพาธบัฟเฟอร์

RECIPIENT (RCPT)

คำสั่งนี้จะใช้ระบุตัวรับข้อมูลเมล์แต่ละตัว ถ้ามีตัวรับหลายตัวก็จะใช้คำสั่งนี้หลาย ๆ ครั้งเท่ากับจำนวนตัวรับทั้งหมด ในคำสั่งนี้จะระบุถึงฟอร์เวิร์ดพาธซึ่งจะประกอบด้วยรายการของโฮสท์และ เมล์บ็อกซ์ปลายทาง เมื่อรายการโฮสท์ถูกแสดงมันจะกลายเป็นจุดเริ่มต้นของเส้นทางและระบุว่าเมล์ต้องถูกถ่ายทอดไปยังโฮสท์ถัดไปในรายการ

เมื่อเมล์ถูกถ่ายทอดโฮสท์ที่ทำการถ่ายทอดจะย้ายตัวเองจากจุดเริ่มต้นของฟอร์เวิร์ดพาธไปยังจุดเริ่มต้นของรีเวิร์สพาธ เมื่อโฮสท์ถึงปลายทางแล้วตัวรับจะบรรจุลงในเมล์บ็อกซ์ที่เหมาะสม

คำสั่งนี้จะใส่ข้อมูลของฟอร์เวิร์ดพาธจากคำสั่งนี้ลงในฟอร์เวิร์ดพาธบัฟเฟอร์

DATA (DATA)

หลังจากสิ้นสุดคำสั่งนี้แล้ว ข้อมูลจะถูกส่งมายังตัวรับโดยผ่านทางเมล์ดาต้าบัฟเฟอร์ ซึ่งข้อมูลจะสิ้นสุดเมื่อพบลำดับของตัวอักษรดังนี้ <CRLF>.<CRLF>

การสิ้นสุดของข้อมูลเมล์จะแสดงความต้องการว่าตัวรับในขณะนี้ต้องกระทำการบรรจุข้อมูลข่าวสารลงในรีเวิร์สพาธบัฟเฟอร์ , ฟอร์เวิร์ดพาธบัฟเฟอร์ และ เมล์ดาต้าบัฟเฟอร์ และเมื่อสิ้นสุดคำสั่งนี้แล้วบัฟเฟอร์ทั้งหมดจะถูกเคลียร์ ถ้ากระบวนการทั้งหมดเสร็จสมบูรณ์ตัวรับจะส่งคำตอบ "OK" ถ้าไม่สมบูรณ์จะส่งข้อความแสดงความผิดพลาดกลับมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อตัวรับยอมรับข้อความไม่ว่าจะเป็นการส่งต่อหรือว่าสิ้นสุดมันจะแทรกไทม์สแตมป์ไลน์ (Time Stamp Line) ลงไปที่ส่วนเริ่มต้นของข้อมูลซึ่งระบุโฮสท์ที่ส่งและรับข้อมูลรวมถึงวัน เวลาที่ได้รับข้อมูลลงไปด้วย

เมื่อข่าวสารส่งไปถึงปลายทางแล้ว จุดสิ้นสุดจะกลายเป็นรีเทรินพาธ (return-path) ซึ่งจะรักษาข่าวสารในรีเวิร์สพาธจากคำสั่ง "MAIL" การสิ้นสุดข้อมูลหมายความถึงข้อมูลจะหลุดจากระบบ SMTP

SEND (SEND)

คำสั่งนี้ใช้ในการเริ่มกระบวนการการส่งเมล์โดยที่ข้อมูลถูกส่งไปยังปลายทาง คำสั่งนี้จะมีรีเวิร์สพาธรวมอยู่ด้วยและคำสั่งนี้จะเสร็จสมบูรณ์เมื่อข่าวสารถูกส่งถึงเทอร์มินอล

รีเวิร์สพาธจะประกอบด้วยรายชื่อของโฮสท์ และ เมล์บ็อกซ์ เมื่อรายชื่อโฮสท์ถูกแสดงมันจะกลายเป็นเส้นทางเริ่มต้นและระบุว่าเมล์จะถูกส่งผ่านไปยังทุก ๆ โฮสท์ที่อยู่ในรายการ รายชื่อนี้ใช้เหมือนจุดเริ่มต้นเพื่อที่จะส่งย้อนกลับได้เพื่อแจ้งให้ทราบถึงผลตอบสนองได้ ขณะที่โฮสท์ทุกโฮสท์ทำหน้าที่เปลี่ยนตัวเองเป็นตัวเริ่มต้นของรายการต่อไป มันต้องใช้ชื่อของตัวมันเองที่รู้จักใน IPCE จากเมล์ ที่ส่งมา

คำสั่งนี้จะเคลียร์บัฟเฟอร์ทั้งหมดก่อน และบรรจุข้อมูลของรีเวิร์สพาธจากคำสั่งนี้ ลงในรีเวิร์สพาธบัฟเฟอร์

SEND OR MAIL (SOML)

คำสั่งนี้ใช้ในการเริ่มกระบวนการการส่งเมล์โดยที่ข้อมูลถูกส่งไปยังปลายทางหรือเมล์บ็อกซ์สำหรับแต่ละตัวรับ ข้อมูลจะถูกส่งไปยังเทอร์มินอลของผู้รับถ้าผู้รับใช้งานอยู่ แต่ถ้าไม่ทำงานจะถูกเก็บไว้ในเมล์บ็อกซ์ คำสั่งนี้จะมีรีเวิร์สพาธรวมอยู่ด้วยและคำสั่งนี้จะเสร็จสมบูรณ์เมื่อข่าวสารถูกส่งถึงเทอร์มินอล หรือ เมล์บ็อกซ์

รีเวิร์สพาธจะประกอบด้วยรายชื่อของโฮสท์ และเมล์บ็อกซ์ เมื่อรายชื่อโฮสท์ถูกแสดงมันจะกลายเป็นเส้นทางเริ่มต้นและระบุว่าเมล์จะถูกส่งผ่านไปยังทุก ๆ โฮสท์ที่อยู่ในรายการ รายชื่อนี้ใช้เหมือนจุดเริ่มต้นเพื่อที่จะส่งย้อนกลับได้เพื่อแจ้งให้ทราบถึงผลตอบสนองได้ ขณะที่โฮสท์ทุกโฮสท์ทำหน้าที่เปลี่ยนตัวเองเป็นตัวเริ่มต้นของรายการต่อไป มันต้องใช้ชื่อของตัวมันเองที่รู้จักใน IPCE จากเมล์ ที่ส่งมา

คำสั่งนี้จะเคลียร์บัฟเฟอร์ทั้งหมดก่อนและบรรจุข้อมูลของรีเวิร์สพาธจากคำสั่งนี้ลงในรีเวิร์สพาธบัฟเฟอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SEND AND MAIL (SAML)

คำสั่งนี้ใช้ในการเริ่มกระบวนการการส่งเมล์โดยที่ข้อมูลถูกส่งไปยังปลายทางหรือเมล์บ็อกซ์สำหรับแต่ละตัวรับ ข้อมูลจะถูกส่งไปยังเทอร์มินอลของผู้รับถ้าผู้รับใช้งานอยู่ และสำหรับทุกตัวรับจะส่งไปยังเมล์บ็อกซ์ คำสั่งนี้จะมีรีเวิร์สพาธรวมอยู่ด้วยและคำสั่งนี้จะเสร็จสมบูรณ์เมื่อข่าวสารถูกส่งถึง เทอร์มินอล หรือ เมล์บ็อกซ์

รีเวิร์สพาธจะประกอบด้วยรายชื่อของโฮสท์และเมล์บ็อกซ์ เมื่อรายชื่อโฮสท์ถูกแสดงมันจะกลายเป็นเส้นทางเริ่มต้นและระบุว่าเมล์จะถูกส่งผ่านไปยังทุก ๆ โฮสท์ที่อยู่ในรายการ รายชื่อนี้ใช้เหมือนจุดเริ่มต้นเพื่อที่จะส่งย้อนกลับได้เพื่อแจ้งให้ทราบถึงผลตอบสนองได้ ขณะที่โฮสท์ทุกโฮสท์ทำหน้าที่เปลี่ยนตัวเองเป็นตัวเริ่มต้นของรายการต่อไป มันต้องใช้ชื่อของตัวมันเองที่รู้จักใน IPCE จากเมล์ ที่ส่งมา

คำสั่งนี้จะเคลียร์บัฟเฟอร์ทั้งหมดก่อนและบรรจุข้อมูลของรีเวิร์สพาธจากคำสั่งนี้ลงในรีเวิร์สพาธบัฟเฟอร์

RESET (REST)

เป็นคำสั่งยกเลิกกระบวนการ สิ่งที่เก็บไว้โดยผู้รับ , ผู้ส่งและข้อมูลจะถูกยกเลิกทั้งหมด บัฟเฟอร์และสถานะต่าง ๆ จะถูกเคลียร์ทั้งหมดและผู้รับจะส่ง "OK" กลับมา

VERIFYING (VRFY)

คำสั่งนี้จะถามตัวรับเพื่อความแน่ใจว่าชื่อถูกต้องหรือไม่ ถ้าถูกต้องตัวรับจะส่งชื่อเต็ม (ถ้ารู้) และเมล์บ็อกซ์กลับมาซึ่งมาซึ่งในคำสั่งนี้ไม่มีผลกับบัฟเฟอร์

HELP (HELP)

คำสั่งนี้ช่วยให้ผู้รับส่งข้อมูลที่มีประโยชน์กลับมาสู่ผู้ส่ง ซึ่งคำสั่งนี้ไม่มีผลกับบัฟเฟอร์

NOOP (NOOP)

คำสั่งนี้ไม่มีผลอะไรในระบบนอกจากจะรอเวลาจนกว่าผู้รับส่ง "OK" กลับมาซึ่งจะไม่มีผลอะไรกับบัฟเฟอร์

QUIT (QUIT)

คำสั่งนี้ตัวรับจะส่ง "OK" กลับมาถ้าได้รับจากตัวส่งและจะสิ้นสุดเส้นทางการติดต่อ ซึ่งตัวรับจะไม่เลิกการติดต่อจนกว่าจะได้รับและตอบสนองต่อคำสั่ง "QUIT" และตัวส่งก็จะไม่ยกเลิกการติดต่อจนกว่าจะส่งคำสั่ง "QUIT" และได้รับการตอบสนอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TURN (TURN)

คำสั่งนี้จะทำให้ตัวรับเปลี่ยนสถานะไปซึ่งเป็นไปได้ 2 กรณีคือ

1. ส่งคำสั่ง "OK" กลับและกลายเป็นตัวส่ง

2. ส่งปฏิเสธและเป็นตัวรับเหมือนเดิม

สรุปการใช้คำสั่ง

- คำสั่ง "HELO" เป็นคำสั่งที่ใช้ในการเริ่มต้นการทำงานทั้งหมด

- คำสั่ง "NOOP" , "HELO" , "EXPN" และ "VRFY" สามารถใช้ได้ตลอดเวลา

- คำสั่ง "MAIL" , "SEND" , "SOML" และ "SAML" ใช้เป็นคำสั่งเริ่มการส่งข้อมูล หลังจากนั้นจะต้องมีการส่งคำสั่งต่าง ๆ อีกได้แก่ "RCPT" และ "DATA" และสามารถยกเลิกกระบวนการทั้งหมดโดยคำสั่ง "RSET"

- คำสั่ง "QUIT" ใช้ในการยกเลิกการทำงาน

2.1.4.1.2 COMMAND SYNTAX

โค้ดของคำสั่งจะมี 4 ตัวอักษรและไม่สนใจว่าจะเป็นตัวเล็กหรือตัวใหญ่ ในขณะที่โฮสท์จะสนใจอักษรตัวเล็กตัวใหญ่ด้วย โค้ดและพารามิเตอร์ต่าง ๆ จะถูกคั่นด้วย <SP> และสิ้นสุดคำสั่งด้วย <CRLF>

2.1.5 การตอบรับของ SMTP

2.1.5.1 ทฤษฎีของโค้ดคำตอบ (THEORY OF REPLY CODES)

ตัวเลข 3 ตัวที่เป็นโค้ดของคำตอบจะระบุถึงลักษณะเฉพาะ ตัวเลขตัวแรกจะแจ้งให้ทราบถึงผลตอบสนองว่าดี , ไม่ดีหรือไม่สมบูรณ์ ซึ่งตัวรับจะพิจารณาและทำงานต่อไปโดยดูจากตัวเลขตัวแรก ตัวส่งที่ต้องการจะรู้ว่าข้อผิดพลาดคืออะไรก็จะทำการตรวจสอบเลขตัวที่ 2 และ 3 เพื่อข้อมูลความผิดพลาดที่สมบูรณ์

จะมี 5 ค่าตัวเลขสำหรับคำตอบแรก

1yz - คำสั่งได้รับการยอมรับ แต่การกระทำที่ต้องการจะถูกระงับจนกว่าจะได้รับการยืนยันของคำตอบนี้และตัวส่งจะส่งคำสั่งอื่นเพื่อทำงานต่อไปหรือออกจากระบบ

2yz - การกระทำที่ต้องการเสร็จสมบูรณ์ทุกอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3yz - คำสั่งถูกยอมรับแต่การกระทำนั้นจะถูกระงับจนกว่าจะได้รับข่าวสารต่อไป ตัวส่งจะส่งคำสั่งอื่น ๆ ต่อไปคำตอบที่ใช้จะใช้ในกลุ่มลำดับคำสั่ง

4yz - คำสั่งไม่ยอมรับและการกระทำที่ต้องการไม่ปรากฏ แต่เงื่อนไขการผิดพลาดจะเป็นเพียงชั่วขณะและการกระทำอาจต้องการอีกครั้ง ตัวส่งควรกลับไปเริ่มต้นใหม่ตามลำดับของคำสั่ง

5yz - คำสั่งไม่ยอมรับและการกระทำจะไม่เกิด ตัวส่งจะถูกห้ามจากการทบทวนความต้องการความผิดพลาดบางอย่างสามารถแก้ไขได้ตัวส่งจะทำการส่งคำสั่งไปใหม่

ตัวเลขที่ 2 จะแสดงถึงสิ่งต่าง ๆ เหล่านี้

x0z - จะแจ้งถึงความผิดพลาดทาง syntax

x1z - จะแจ้งถึงความต้องการของข่าวสารเช่นสถานะหรือความช่วยเหลือ

x2z - จะอ้างอิงถึงเส้นทางการติดต่อ

x3z - ไม่ใช้

x4z - ไม่ใช้

x5z - จะแจ้งถึงระบบสื่อสาร

ตัวเลขที่ 3 จะช่วยอธิบายถึงความชัดเจนได้มากขึ้น รายการของคำตอบจะแสดงให้เห็นถึงสิ่งเหล่านั้น คำตอบแต่ละอันจะถูกเปลี่ยนไปตามคำสั่งซึ่งมีความสัมพันธ์กัน

คำตอบอาจมีมากกว่า 1 บรรทัด ในกรณีนี้ข้อความที่สมบูรณ์จะถูกทำอะไรบางอย่างเพื่อให้ตัวรับได้รู้ว่าสามารถหยุดรับคำตอบได้ ซึ่งต้องการรูปแบบพิเศษในแต่ละบรรทัดดังนี้

- รูปแบบสำหรับหลาย ๆ บรรทัด จะเริ่มต้นด้วยโค้ดตัวเลขที่เป็นคำตอบ ตามด้วยเครื่องหมาย "-" และตามด้วยข้อความในแต่ละบรรทัด และในบรรทัดสุดท้ายจะเริ่มด้วยโค้ดตัวเลขคำตอบ ตามด้วย <SP> และข้อความและจบด้วย <CRLF>

2.1.6 ลำดับของคำสั่งและคำตอบ

การติดต่อระหว่างตัวรับและตัวส่งจะถูกควบคุมด้วยตัวส่งโดยที่ตัวส่งจะส่งคำสั่งและตัวรับจะตอบสนองกลับมา ตัวส่งนั้นจะรอการตอบสนองแล้วค่อยส่งคำสั่งต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.7 รายละเอียดอื่น ๆ

2.1.7.1 MINIMUM IMPLEMENTATIONS

เพื่อการที่จะทำให้ SMTP ทำงานได้ดีจะต้องมีส่วนประกอบอย่างน้อยที่ต้องมีในตัวรับทุกตัวดังนี้

- HELO
- MAIL
- RCPT
- DATA
- RSET
- NOOP
- QUIT

2.1.7.2 TRANSPARENCY

ในการแสดงถึงการสิ้นสุดของข้อความจะใช้ชุดตัวอักษรดังนี้ <CRLF>.<CRLF>

ในการที่ผู้ใช้จะทำการติดต่อสื่อสารจำเป็นต้องใช้กระบวนการต่อไปนี้

1. ก่อนทำการส่งข้อความตัวส่งจะพิจารณาตัวอักษรตัวแรกของบรรทัด ถ้าเป็นจุดก็จะเพิ่มจุดเข้าไปอีก 1 จุดในบรรทัดนั้น

2. เมื่อข้อความได้รับโดยตัวรับแล้วจะทำการตรวจสอบ ถ้าในบรรทัดมีเพียงจุด 1 จุด แสดงว่าสิ้นสุดข้อความ ถ้าบรรทัดนั้นมีจุดและตามด้วยตัวอักษรอื่นจะทำการลบตัวอักษรตัวแรกทิ้ง

2.1.7.3 SIZES

ขนาดของตัวอักษรแต่ละประเภทกำหนดไว้สูงสุดดังนี้

user - ความยาวสูงสุด 64 ตัวอักษร

domain - ความยาวของชื่อหรือตัวเลขไม่เกิน 64 ตัวอักษร

path - ความยาวสูงสุดคือ 256 ตัวอักษร

command line - ในแต่ละบรรทัดจะไม่เกิน 512 ตัวอักษร

reply line - ในแต่ละบรรทัดจะไม่เกิน 512 ตัวอักษร

text line - ในแต่ละบรรทัดจะไม่เกิน 1000 ตัวอักษร

recipients buffer - จำนวนตัวรับสูงสุดคือ 100 ตัวอักษร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 Post Office Protocol - Version 3

2.2.1 บทนำ

โดยส่วนใหญ่แล้วเครื่องคอมพิวเตอร์ขนาดเล็กบนเครือข่ายอินเตอร์เน็ตไม่สามารถจัดการระบบส่งข่าวสาร (message transsport system : MTS) ได้ยกตัวอย่างเช่น เครื่องเวิร์กสเตชันอาจไม่มีทรัพยากรเพียงพอ (ความเร็ว,เนื้อที่ที่ใช้จัดเก็บ) ที่จะใช้เป็น SMTP เซิร์ฟเวอร์ [RFC821] และช่วยระบบจดหมายท้องถิ่น (local mail delivery system) ในการจัดเก็บและต้องทำงานอย่างต่อเนื่อง ในทำนองเดียวกันเราไม่สามารถนำเครื่องคอมพิวเตอร์ต่อเข้ากับเครือข่ายอินเตอร์เน็ตได้ตลอดเวลาด้วยเหตุผลด้านราคา

อย่างไรก็ตามหากสามารถสร้างระบบจัดการเมล์ด้วยเครื่องคอมพิวเตอร์ขนาดเล็กได้ก็จะเป็นประโยชน์อย่างมากและเครื่องคอมพิวเตอร์ขนาดเล็กมักจะสนับสนุนตัวแทนผู้ใช้ เพื่อช่วยจัดการเมล์ ด้วยเหตุนี้เครื่องคอมพิวเตอร์ขนาดใหญ่ที่สามารถจัดการส่งข่าวสารได้จึงมีบริการตู้รับจดหมาย (maildrop service) เพื่อเปิดให้คอมพิวเตอร์เข้าใช้บริการ และ Post Office Protocol - Version 3 (POP3) จึงถูกพัฒนาขึ้นเพื่อให้เครื่องเวิร์กสเตชันสามารถเข้าจัดการตู้รับจดหมายบนเซิร์ฟเวอร์ได้อย่างอิสระ (dynamically access) หรือเพื่อให้เครื่องเวิร์กสเตชันนำจดหมายออกจากเครื่องเซิร์ฟเวอร์ได้นั่นเอง

คำว่า ไคล์เอนท์โฮสท์ (client host) หมายถึง เครื่องคอมพิวเตอร์ที่ใช้บริการ POP3

คำว่า เซิร์ฟเวอร์โฮสท์ (server host) หมายถึง เครื่องคอมพิวเตอร์ที่เปิดให้บริการ POP3

2.2.2 ข้อกำหนด

กำหนดวิธีการที่ไคล์เอนท์โฮสท์จะนำจดหมายเข้าสู่ระบบการจัดส่ง (transport system) มีหลักการดังต่อไปนี้

"เมื่อตัวแทนผู้ใช้บนเครื่องไคล์เอนท์โฮสท์ต้องการนำจดหมายเข้าสู่ระบบการจัดส่ง จะต้องสร้างการติดต่อ SMTP เข้ากับรีเรย์โฮสท์ (relay host) ของมัน (ซึ่งรีเรย์โฮสท์นี้จะเป็นหรือไม่เป็น POP3 เซิร์ฟเวอร์ก็ได้)"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.3 คำสั่งพื้นฐาน

ในขั้นแรกเซิร์ฟเวอร์โฮสท์จะคอยสัญญาณที่ TCP พอร์ตที่ 110 เมื่อมีไคล์เอนท์โฮสท์ต้องการใช้บริการนี้ก็จะสร้างการเชื่อมต่อ TCP ที่พอร์ตนี้ เมื่อเกิดการเชื่อมต่อขึ้นเซิร์ฟเวอร์โฮสท์จะส่งสัญญาณ ทักทาย (greeting) จากนั้นทั้งเซิร์ฟเวอร์โฮสท์ และ ไคล์เอนท์โฮสท์จะโต้ตอบกันด้วยคำสั่ง (command) และการตอบรับ (response) ตามลำดับจนกระทั่งมีการปิดการเชื่อมต่อ (close connection) หรือยกเลิกการเชื่อมต่อ (abort connection)

คำสั่งใน POP3 ประกอบด้วย "คีย์เวิร์ด" (keyword) และมักตามด้วย "อะกิวเมนท์" (argument) หนึ่งตัวหรือมากกว่านั้น ทุก ๆ คำสั่งจะต้องลงท้ายด้วย CRLF (carriage return and line feed) คีย์เวิร์ด กับ อะกิวเมนท์ จะเป็นรหัสแอสกีที่สามารถพิมพ์ได้ (printable ASCII) ใช้การเว้นวรรค 1 ครั้ง (single space) แยกส่วนที่เป็น คีย์เวิร์ด กับ อะกิวเมนท์ออกจากกัน คีย์เวิร์ด มีความยาว 3-4 ตัวอักษร และแต่ละอะกิวเมนท์มีความยาวไม่เกิน 40 ตัวอักษร

การตอบรับใน POP3 ประกอบด้วย สถานะ (status indecator) กับ คีย์เวิร์ด และมักตามด้วยข้อมูลเพิ่มเติม ทุก ๆ การตอบรับจะต้องลงท้ายด้วย CRLF เสมอ มีสถานะอยู่ 2 สถานะคือ สถานะบวก "+OK" และสถานะลบ "-ERR"

การตอบรับส่วนใหญ่แล้วจะมีหลายบรรทัด (multi-line) ในกรณีนี้ทุกบรรทัดจะต้องลงท้ายด้วย CRLF และบรรทัดสุดท้ายจะต้องมี "." (termination octet : ASCII 046) เพียงตัวเดียวแล้วตามด้วย CRLF ดังนั้น การตอบสนองที่มีหลายบรรทัดจะต้องลงท้ายด้วยอักขระ 5 ตัวคือ "<CRLF>.<CRLF>" ทางด้านไคล์เอนท์โฮสท์จะคอยตรวจสอบว่าบรรทัดขึ้นต้นด้วย "." หรือไม่ ถ้าขึ้นต้นด้วย "." และตัวที่ตามหลังไม่ใช่ "<CRLF>" อักขระ "." จะถูกตัดออก แต่หากตัวที่ตามหลัง "." เป็น "<CRLF>" แล้วจะแสดงถึงการสิ้นสุดของการตอบรับและอักขระ ".<CRLF>" จะถูกตัดออก

ในระหว่างขั้นตอนการติดต่อแบบ POP3 นั้นสามารถแบ่งได้เป็นสภาวะ (State)ได้หลายสภาวะด้วยกัน เมื่อเกิดการเชื่อมต่อ TCP ขึ้นแล้วเซิร์ฟเวอร์ก็จะส่งสัญญาณทักทาย การติดต่อจะเข้าสู่สภาวะการให้สิทธิ์ (Authorization State) ในสภาวะนี้ไคล์เอนท์จะต้องแนะนำ (Identify) ตัวเองต่อ POP3 เซิร์ฟเวอร์ เมื่อแนะนำตัวเองเสร็จแล้วเซิร์ฟเวอร์จะเรียกทรัพยากรที่ใช้เก็บตู้จดหมายขึ้นมา จากนั้นจะเข้าสู่สภาวะการถ่ายโอน (Transaction State) ในสภาวะนี้ไคล์เอนท์จะขอกระทำการ (action) บน เซิร์ฟเวอร์ เมื่อไคล์เอนท์ส่งคำสั่ง QUIT ก็จะเข้าสู่สภาวะการปรับแต่ง (Update

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

State) ในสภาวะนี้ เซิร์ฟเวอร์จะปลดปล่อยทรัพยากรที่ถูกเรียกใช้ในสภาวะการถ่านโอนและกล่าวอำลา จากนั้นการเชื่อมต่อ TCP จะถูกปิด

เครื่องเซิร์ฟเวอร์อาจต้องมีการการตั้งเวลาปิดการเข้าใช้ (logout) อัตโนมัติเมื่อไม่มีคำสั่งเข้ามาเป็นเวลานาน และต้องตั้งเวลาไว้ไม่น้อยกว่า 10 นาที และเวลาที่นับจะถูกลบทิ้งเมื่อมีคำสั่งเข้ามาตามเวลาที่กำหนด เมื่อเวลาเกินไปจากที่กำหนดไว้การติดต่อนั้นจะถือเป็นโมฆะ นั่นหมายถึงจะไม่มีการเข้าสู่สภาวะการปรับแต่ง เซิร์ฟเวอร์จะปิดการเชื่อมต่อ TCP โดยไม่มีการลบจดหมายทิ้งหรือส่งสัญญาณตอบรับใด ๆ อีก

2.2.4 สภาวะการให้สิทธิ์

เมื่อการเชื่อมต่อแบบ TCP ถูกเปิดโดยไคล์เอนท์ จากนั้นเซิร์ฟเวอร์จะส่งคำทักทาย ซึ่งอาจอยู่ในรูปของข้อความใด ๆ ที่ลงท้ายด้วย <CRLF> เช่น

S : +OK POP3 server reply

สังเกตว่าการทักทายนี้ก็เป็นการตอบรับอย่างหนึ่งเซิร์ฟเวอร์ควรทักทายด้วยการตอบรับแง่บวกเสมอ

ขณะนี้การติดต่อเข้าสู่สภาวะการให้สิทธ์แล้วไคล์เอนท์ต้องแนะนำตัวต่อเซิร์ฟเวอร์ การแนะนำตัวนั้นจะมีอยู่ 2 วิธีที่อธิบายในเอกสารฉบับนี้คือ วิธีที่ใช้คำสั่ง USER และ PASS และวิธีที่ใช้คำสั่ง APOP

เมื่อใช้วิธี USER and PASS ทางไคล์เอนท์จะทำการส่งคำสั่ง USER ถ้าทางเซิร์ฟเวอร์ตอบรับด้วยเครื่องหมายตอบรับแง่บวกแล้วไคล์เอนท์อาจทำการส่งคำสั่ง PASS เพื่อดำเนินการต่อ หรือ QUIT เพื่อปิดการติดต่อก็ได้ ถ้าเซิร์ฟเวอร์ตอบรับคำสั่ง USER ด้วยเครื่องหมายตอบรับแง่ลบแล้วไคล์เอนท์ อาจเริ่มทำการขอสิทธิ์ใหม่หรือ QUIT ก็ได้

เมื่อไคล์เอนท์ทำการส่งคำสั่ง PASS ทางด้านเซิร์ฟเวอร์จะใช้อะกิวเมนท์ของ USER และ PASS เป็นเครื่องพิจารณาหาตู้จดหมายที่ถูกต้อง

เมื่อเซิร์ฟเวอร์กำหนดสิทธิ์ที่ไคล์เอนท์สามารถกระทำต่อตู้จดหมายดังกล่าวแล้วเซิร์ฟเวอร์จะล็อกตู้จดหมายนั้นทั้งนี้เพื่อป้องกันมิให้เกิดการแก้ไขหรือลบจากการติดต่ออื่น ๆ ก่อนที่จะเข้าสู่สถานะปรับแต่ง หากการล็อกสามารถกระทำได้แล้วเซิร์ฟเวอร์จะตอบรับด้วยเครื่องหมายสถานะแง่บวก จากนั้นเข้าสู่สภาวะถ่ายโอนพร้อมกับยังไม่มีการทำเครื่องหมายการลบที่จดหมายทุกฉบับ แต่ถ้าตู้จดหมายไม่สามารถเปิดไม่ว่าด้วยเหตุผลใดก็ตาม (เช่น ไม่สามารถทำการล็อกได้ , ไคล์เอนท์ถูกปฏิเสธในการใช้ตู้จดหมายที่ขอใช้ หรือไม่มีตู้จดหมายที่ขอใช้) เซิร์ฟเวอร์จะตอบรับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ด้วยเครื่องหมายสถานะแง่ลบ หลังจากนั้นเซิร์ฟเวอร์อาจปิดการเชื่อมต่อ หรือถ้าไม่ปิดไคล์เอนท์อาจทำคำสั่งขอใช้สิทธิ์ใหม่หรือใช้คำสั่ง QUIT

หลังจากเซิร์ฟเวอร์เปิดตู้จดหมายแล้ว จดหมายทุกฉบับจะถูกกำหนดตัวเลขโดยฉบับแรกเป็นเลขหนึ่ง ฉบับที่สองเป็นเลขสอง อย่างนี้ไปเรื่อย ๆ ใน POP3 ทั้งคำสั่งและการตอบรับจะแสดงตัวเลขประจำจดหมายและขนาดจดหมายเป็นเลขฐานสิบ

สรุปคำสั่งในสถานะการให้สิทธิ์

หมายเหตุ C หมายถึงไคล์เอนท์ S หมายถึง เซิร์ฟเวอร์

USER ชื่อที่ใช้ติดต่อกับ POP3

อะกิวเมนท์ :

ข้อความบ่งชี้ตู้จดหมาย (ต้องระบุ) ที่เซิร์ฟเวอร์รู้จัก

ระเบียบการใช้:

อาจใช้ในสภาวะการให้สิทธิ์เท่านั้นหลังจากเซิร์ฟเวอร์ทักทาย หรือหลังจากการขอใช้คำสั่ง USER และ PASS ที่ล้มเหลว

การตอบรับ:

+OK name is a valid mailbox

-ERR never heard of mailbox name

ตัวอย่างการใช้:

C : USER mrose

S : +OK mrose is a real hoopy frood

...

C : USER frated

S : -ERR sorry, no mailobox for frated here

PASS รหัสผ่าน

อะกิวเมนท์ :

รหัสผ่านของตู้จดหมายนั้น

ระเบียบการใช้:

อาจใช้ในสภาวะการให้สิทธิ์เท่านั้นหลังจากคำสั่ง USER ที่ถูกตอบรับแง่บวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้ออภิปราย:

เมื่อสิ่งที่ตามหลัง PASS คือรหัสผ่านเพียงอย่างเดียวก็น่าจะมองให้การเว้นวรรคหลัง PASS เป็นส่วนหนึ่งของรหัสผ่านด้วย

การตอบรับที่เป็นไปได้:

+OK maildrop locked and ready

-ERR invalid password

-ERR unable to lock maildrop

ตัวอย่างการใช้:

C : USER mrose

S : +OK mrose is a real hoopy frood

C : PASS secret

S : +OK mrose's maildrop has 2 messages (320 octets)

....

C: USER mrose

S : +OK mrose is a real hoopy frood

C : PASS secrect

S : -ERR maildrop already locked

QUIT

อะกิวเมนท์ :

ไม่มี

ระเบียบการใช้ :

ไม่มี

การตอบรับ:

+OK

ตัวอย่างการใช้:

C : QUIT

S : +OK deway POP3 server signing off

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.5 สภาวะการถ่ายโอน

เมื่อไคล์เอนท์แนะนำตัวเองแล้วเซิร์ฟเวอร์ล็อกและเปิดตู้จดหมายแล้ว เข้าสู่ สภาวะการถ่ายโอน สภาวะนี้ไคล์เอนท์อาจใช้คำสั่ง POP3 ซ้ำ ๆ กันโดยหลังแต่ละคำสั่งตามด้วยการตอบรับของเซิร์ฟเวอร์จนการทั่งไคล์เอนท์สั่ง QUIT จากนั้นจะเข้าสู่สภาวะปรับแต่ง

สรุปคำสั่งในสภาวะการถ่ายโอน

STAT

อะกิวเมนท์:

ไม่มี

ระเบียบการใช้:

อาจถูกใช้ในสภาวะการถ่ายโอนเท่านั้น

ข้ออภิปราย:

เซิร์ฟเวอร์ตอบรับด้วยการตอบรับแง่บวกและข้อมูลของตู้รับจดหมาย ซึ่งเรียกว่า "ดรอปลิสซิ่ง" (Drop Listing)

เพื่อให้ง่ายเข้าเซิร์ฟเวอร์ทุกตัวต้องการมีรูปแบบสำหรับดรอปลิสซิ่ง การตอบรับแง่บวกตามด้วยเว้นวรรค ขนาดของจดหมายในตู้เป็นออคเตต (octet)แต่เอกสารฉบับนี้ไม่กำหนดสิ่งที่ตามหลังขนาดของจดหมายในตู้ แต่อย่างน้อยควรลงท้ายการตอบรับนี้ด้วย <CRLF>

หมายเหตุ: เอกสารนี้ไม่สนับสนุนอย่างยิ่งที่จะมีการเพิ่มข้อมูลในดรอปลิสซิ่ง ส่วนเรื่องอื่นเช่นตัวเลือกคุณสมบัติจะอภิปรายต่อไป

สังเกตว่าจดหมายที่ถูกทำเครื่องหมายการลบไว้จะไม่ถูกนับ

การตอบรับ:

+OK nn mm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้ง

038323

ตัวอย่างการใช้:

C : STAT

S : +OK 2 320

LIST [msg]

อะกิวเมนท์:

เป็นหมายเลขจดหมาย(ระบุหรือไม่ระบุก็ได้) ซึ่ง (ปัจจุบันนี้อาจ)ไม่นับจดหมายที่ถูกทำเครื่องหมายถูกลบ

ระเบียบการใช้:

อาจใช้ในสภาวะการถ่ายโอนเท่านั้น

ข้ออภิปราย:

ถ้าอะกิวเมนท์ถูกระบุ เซิร์ฟเวอร์จะตอบด้วยการตอบรับแง่บวกและข้อมูลเพิ่มเติมเรียก "สแกนลิสซิ่ง" (scan listing)

ถ้าไม่ระบุอะกิวเมนท์ ทางเซิร์ฟเวอร์จะตอบด้วยการตอบรับแง่บวกและข้อมูลแบบหลายบรรทัด หลังจาก +OK แต่ละบรรทัดคือข้อมูลของแต่ละฉบับ

เพื่อให้ง่ายเข้าเซิร์ฟเวอร์ต้องการรูปแบบที่แน่นอนของสแกนลิสซิ่ง ซึ่งประกอบด้วยหมายเลขจดหมายตามด้วยเว้นวรรคและขนาดของจดหมายในหน่วยออคเตต แต่เอกสารฉบับนี้ไม่กำหนดสิ่งที่ตามหลังขนาดของจดหมาย แต่อย่างน้อยควรลงท้ายการตอบรับนี้ด้วย <CRLF>

หมายเหตุ: เอกสารนี้ไม่สนับสนุนอย่างยิ่งที่จะมีการเพิ่มข้อมูลในดรอปลิสซิ่ง ส่วนเรื่องอื่นเช่นตัวเลือกคุณสมบัติจะอภิปรายต่อไป

สังเกตว่าจดหมายที่ถูกทำเครื่องหมายการลบไว้จะไม่ถูกนับ

การตอบรับ:

+OK scan listing follows

-ERR no such message

ตัวอย่างการใช้:

C : LIST

S : +OK 2 messages (320 octets)

S : 1 120

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

S : 2 200

S : .

....

C : LIST 2

S : +OK 2 200

....

C : LIST 3

S : -ERR no such message, only 2 mesage in maildrop

RETR msg

อะกิวเมนท์:

หมายเลขจดหมาย (ต้องระบุ) ซึ่งอาจไม่รวมถึงจดหมายที่มีเครื่องหมายถูกลบ

ระเบียบการใช้:

อาจถูกใช้ในสภาวะการถ่ายโอนเท่านั้น

ข้ออภิปราย:

ถ้าเซิร์ฟเวอร์ตอบรับแง่บวกแล้วตามด้วยการตอบรับแบบหลายบรรทัด หลังจาก +OK เซิร์ฟเวอร์จะทำการส่งตัวจดหมายที่ระบุมาแล้วจบด้วยเทอร์มินอลคาร์แลคเตอร์ (termination charactor) (เช่นเดียวกับการตอบรับแบบหลายบรรทัดอื่น)

การตอบรับ:

+OK message follows

-ERR no such message

ตัวอย่างการใช้:

C : RETR 1

S : +OK 120 octets

S : <the POP3 server sends the entire message here>

S : .

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DELE msg

อะกิวเมนท์:

หมายเลขจดหมาย (ต้องระบุ) ซึ่งอาจไม่รวมถึงจดหมายที่ถูกทำเครื่องหมายถูกลบ

ระเบียบการใช้:

อาจใช้ในสภาวะการถ่ายโอนเท่านั้น

ข้ออภิปราย:

เซิร์ฟเวอร์ทำเครื่องหมายถูกลบที่จดหมาย หากมีการอ้างถึงอีกในอนาคตจะรายงานความผิดพลาด และเซิร์ฟเวอร์จะไม่ลบจดหมายจริง ๆ จนกว่าจะเข้าสู่สภาวะปรับแต่ง

การตอบรับ:

+OK message deleted

-ERR no such message

ตัวอย่างการใช้:

C : DELE 1

S : +OK message 1 deleted

....

C : DELE 2

S : -ERR message 2 already deleted

NOOP

อะกิวเมนท์:

ไม่มี

ระเบียบการใช้ :

อาจใช้ในสภาวะการถ่ายโอนเท่านั้น

ห้วข้ออภิปราย่ :

เซิร์ฟเวอร์เพียงแค่ตอบรับแง่บวก

การตอบรับ :

+OK

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างการใช้งาน :

C : NOOP

S : +OK

RSET

อะกิวเมนท์ :

ไม่มี

ระเบียบการใช้:

อาจใช้ในสภาวะการถ่ายโอนเท่านั้น

ข้ออภิปราย :

ยกเลิกการทำเครื่องหมายลบทุกเครื่องหมายและตอบรับแง่บวก

การตอบรับ:

+OK

ตัวอย่างการใช้:

C : RSET

S : +OK maildrop has 2 messages (320 octets)

2.2.6 สภาวะการปรับแต่ง

เมื่อไคล์เอนท์ทำการส่งคำสั่ง QUIT จะเข้าสู่สภาวะการปรับแต่ง (สังเกตว่าถ้า QUIT จากสภาวะให้สิทธิ์จะไม่เข้าสู่สภาวะการปรับแต่ง)

ถ้าการติดต่อหยุดลงโดยเหตุผลอื่นนอกจากคำสั่งของไคล์เอนท์แล้ว จะไม่เข้าสู่สภาวะปรับแต่งและต้องไม่ลบจดหมายออก

QUIT

อะกิวเมนท์:

ไม่มี

ระเบียบการใช้:

ไม่มี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้ออภิปราย:

เซิร์ฟเวอร์จะลบจดหมายทุกฉบับที่ถูกทำเครื่องหมายไว้ออกจากตู้จดหมาย แล้วปลดล็อกจากนั้นตอบรับแล้วปิดการเชื่อมต่อ TCP

การตอบรับ:

+OK

ตัวอย่างการใช้:

C : QUIT

S : +OK deway POP3 server signing off (maildrop empty)

....

C : QUIT

S : +OK deway POP3 server signing off (2 message left)

....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 วินโดวส์ ซ้อกเก็ต

Windows Sockets

กุญแจที่จะนำไปสู่การเขียนโปรแกรมบนอินเทอร์เนต (Internet Programming) โดยใช้วินโดวส์คือ "ซ้อกเก็ต" (Sockets) โดยทั่วไปแล้วซ้อกเก็ต หมายถึงเซต เซตหนึ่ง (a set) ของฟังก์ชันที่ใช้ทำการติดต่อแบบสองทาง (Two-way Communication) โดยใช้โปรโตคอล TCP/IP หรือโปรโตคอลอื่น ๆ

ส่วนถ้าหากเราพิจารณาคำว่าวินโดวส์ ซ้อกเก็ต (Windows Sockets) แล้วจะหมายความถึง API มาตรฐานที่ถูกพัฒนาโดยกลุ่มบริษัทกว่า 20 บริษัทและถูกใช้เป็นมาตรฐานการเปิดเครือข่ายเพื่อจัดการการติดต่อโดยใช้โปรโตคอล TCP/IP (ถึงแม้ว่าจะมีการเผื่อไว้สำหรับใช้กับโปรโตคอลอื่นภายหลังก็ตาม) ถ้าหากเราต้องการเขียนโปรแกรมบนอินเทอร์เนตสำหรับวินโดวส์แล้ว เราต้องรู้จักซ้อกเก็ตและวินโดวส์ซ้อกเก็ต

2.3.1 ซ้อกเก็ตคืออะไร

ซ้อกเก็ตมาจากระบบปฏิบัติการยูนิกซ์ (UNIX) และโดยเฉพาะอย่างยิ่งมาจากเบิร์กเลย์-ยูนิกส์ (Berkeley UNIX: BSD) ช่วงประมาณปีคริสตศักราช 1980 ซ้อกเก็ตหนึ่งซ้อกเก็ตคือชิ้นส่วนหนึ่ง (a piece) ของซอฟแวร์ (Software) ซึ่งสามารถรับและส่งข้อมูลบนเครือข่าย TCP/IP ได้ ส่วนของข่าวสารข้อมูล (information) เกี่ยวกับซ้อกเก็ตที่สำคัญมีอยู่ 3 ส่วนคือ

1. IP Address ซ้อกเก็ตกำลังติดต่ออยู่
2. พอร์ต ที่ซ้อกเก็ตใช้ติดต่อกับแอดเดรสนั้น
3. ชนิดของซ้อกเก็ต

สิ่งหนึ่งที่เกี่ยวกับพอร์ตที่จำเป็นต้องทำความเข้าใจให้ถูกต้องนั่นคือ พอร์ตในที่นี้ไม่เหมือนพอร์ตขนานหรือพอร์ตอนุกรม พอร์ตเหล่านี้ไม่ได้เป็นฮาร์ดแวร์ (Hardware) ใด ๆ ทั้งสิ้น พอร์ตในที่นี้เป็นเพียงสภาวะ (convention) ที่ถูกใช้โดยโปรแกรมที่รันอยู่บนเครื่องที่กำลังทำการติดต่อสื่อสาร โปรโตคอล TCP/IP จะให้เครื่องหนึ่งสามารถติดต่อกับอีกหลาย ๆ เครื่องในขณะเดียวกันได้ ยกตัวอย่างเช่น เราสามารถเปิดเวบบราวเซอร์ให้โหลดเพจขนาดใหญ่ ๆ ขณะเดียวกันก็ให้นิวส์รีดเดอร์จัดเรียงและอินเดกซิ่งบทความขนาดใหญ่ และโปรแกรม FTP ก็กำลังดาวน์โหลดไฟล์ขนาดใหญ่ไฟล์หนึ่ง และทันใดนั้นก็มีอีเมล์ส่งมาถึงพอดี เราก็เลยหันไปอ่านอีเมล์โดยปล่อยให้มีการโหลด,อินเดกซิ่งและอื่น ๆ ต่อไป สภาวะต่าง ๆ ของการติดต่อบนอินเทอร์เนตระหว่างเครื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของเรากับเครื่องอื่น สภาวะเหล่านี้จะเกิดขึ้นบนพอร์ตที่แตกต่างกัน บริการมาตรฐานบนอินเทอร์เนตเช่น เวบนิวส์ FTP และเมล์ เหล่านี้จะมีพอร์ตมาตรฐานของแต่ละบริการ

ซ้อกเก็ตอยู่ 2 ชนิดได้แก่ สตรีมซ้อกเก็ต (stream socket) และดาต้าแกรมซ้อกเก็ต (datagram socket) ซึ่งทั้งสองชนิดใช้กับโปรโตคอลหลักสองประเภทของ TCP/IP คือ TCP และ UDP โปรโตคอล TCP มีกระบวนการคล้ายกับการที่เครื่องสองเครื่องสร้างการติดต่อหนึ่งช่องการติดต่อ (form a connection) จากนั้นข้อมูลจะถูกแลกเปลี่ยนผ่านช่องการติดต่อนี้ เมื่อเสร็จก็ทำการปิดช่องการติดต่อนั้น ในทางตรงกันข้าม โปรโตคอล UDP จะไม่เกี่ยวของกับการสร้างการติดต่อเครื่องหนึ่งส่งข้อมูลให้เครื่องอื่นโดยไม่สร้างช่องทางการติดต่อ สตรีมซ้อกเก็ตใช้สร้างโปรโตคอล TCP และถูกใช้กับงานที่จะต้องแลกเปลี่ยนข้อมูลจำนวนมาก ๆ หรืองานที่เน้นว่าลำดับก่อนหลังของข้อมูลที่เข้ามามีความสำคัญ ตัวอย่างของการใช้งาน TCP ที่เก่าแก่ซึ่งก็ใช้สตรีมซ้อกเก็ต นั่นคือ โปรโตคอล FTP แต่ดาต้าแกรมซ้อกเก็ตใช้สร้างโปรโตคอล UDP และถูกใช้กับงานที่มีการแลกเปลี่ยนข้อมูลน้อย ๆ หรืองานที่ไม่สนใจลำดับก่อนหลังของข้อมูล ตัวอย่างของการใช้งาน UDP ที่เก่าแก่เห็นจะเป็นคล็อกอัพ เดทเตอร์ (clock-updater) ซึ่งจะบรอดคาสต์เวลาของระบบไปสู่เครื่องอื่น ๆ

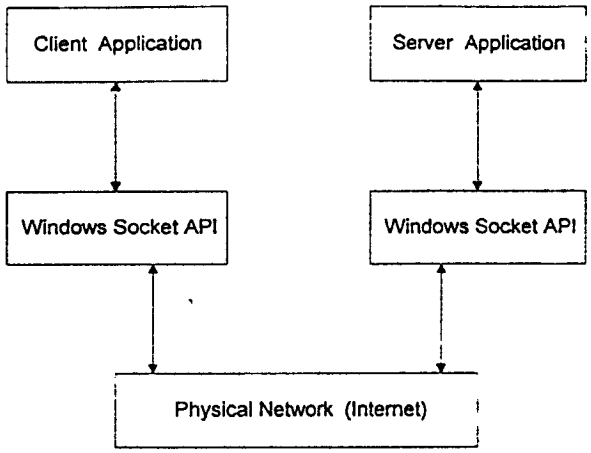
2.3.2 การได้รับ API ของวินโดวส์ ซ้อกเก็ต

เมื่อมีคนพูดถึง API ของวินโดว์ซ้อกเก็ต คน ๆ นั้นจะหมายถึงข้อตกลงที่เป็นที่ยอมรับของทุก ๆ บริษัท ซึ่งก็คือชื่อของฟังก์ชันต่าง ๆ และหน้าที่การทำงานของฟังก์ชันเหล่านั้นนั้น ยกตัวอย่างเช่นข้อตกลงระบุว่า มีฟังก์ชันหนึ่งชื่อ connect() ซึ่งทำหน้าที่ "สร้างการช่องการติดต่อระหว่างซ้อกเก็ตที่กำหนด"

อย่างไรก็ตามเราไม่สามารถที่จะเอ็กซิคิวต์ "ข้อตกลง" ได้ เราต้องมีซอสโค้ด หรือโค้ดที่คอมไพล์แล้ว หลายบริษัทจึงจัดหาไดนามิกไลบรารี่ที่เรียกว่า WINSOCK.DLL ซึ่งทำให้เราสามารถเรียกฟังก์ชันได้ทุกฟังก์ชันที่ระบุไว้ในข้อตกลงดังกล่าว ตัวอย่างเช่นชุดของโปรแกรมซึ่งโดยทั่วไปเรียกว่า ทรัมเป็ต (Trumpet) ของปีเตอร์ แทตแทม (Peter Tattam) จะมีไฟล์ WINSOCK.DLL ซึ่งจะมีฟังก์ชันเหล่านี้และฟังก์ชันอื่นที่ใช้ในการสร้างการเชื่อมต่อกับอินเทอร์เนตแบบ PPP (Point-to-Point Protocol) หรือ SLIP (Serial Line Internet Protocol) ผ่านสายโทรศัพท์ แต่ถ้าเป็นวินโดวส์ 95 และวินโดวส์ NT จะมี WINSOCK.DLL ต่างกันเล็กน้อย

API ของวินโดว์ซ้อกเก็ตจะเป็นตัวที่อยู่ระหว่างแอพพลิเคชันกับเน็ตเวิร์ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 การติดต่อของวินโดวส์แอพพลิเคชันผ่านเครือข่ายอินเทอร์เนต

รูปนี้แสดงการติดต่อของวินโดวส์แอพพลิเคชันสองแอพพลิเคชันคืออินเทอร์เนตไคลเอนต์กับอินเทอร์เนตเซิร์ฟเวอร์ ซึ่งทั้งสองติดต่อผ่านเครือข่ายอินเทอร์เนตทางวินโดว์ซ้อกเก็ต เป็นที่แน่นอนว่าโปรแกรมที่เราติดต่ออยู่อาจไม่เป็นวินโดวส์แอพพลิเคชัน ซ้อกเก็ตสร้างกระบวนการติดต่อแบบเดียวกันในหลายระบบปฏิบัติการ

2.3.3 การเรียกใช้ซ้อกเก็ตในโปรแกรม C++

ข้อกำหนด Winsock ระบุฟังก์ชันและพารามิเตอร์ของแต่ละฟังก์ชัน ส่วนไฟล์ WINSOCK.DLL ทำให้เราสามารถเรียกใช้ฟังก์ชันเหล่านั้นจากส่วนใดก็ได้ของโปรแกรมของเรา เราจะเรียนรู้เกี่ยวกับฟังก์ชันเหล่านี้ในความหมายโดยทั่วไป จากนั้นสร้างขึ้นเป็นคลาสของซ้อกเก็ตหนึ่งคลาสซึ่งมี API อยู่ข้างใน

2.3.4 ฟังก์ชันซ้อกเก็ตของเบิร์กเลย์

สำหรับโปรแกรมเมอร์ที่เคยเขียนซ้อกเก็ตของระบบปฏิบัติการอื่นมาแล้วจะคุ้นเคยกับฟังก์ชันเหล่านี้เป็นอย่างดี เพราะฟังก์ชันเหล่านี้มีชื่อเดียวกัน และมีการทำงานเหมือนกันกับฟังก์ชันในซ้อกเก็ตของระบบยูนิกซ์ดั้งเดิมซึ่งถูกสร้างที่เบิร์กเลย์ ฟังก์ชันเหล่านี้ได้แก่

- accept
- bind

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- closesocket
- connect
- getpeername
- getsockname
- getsockopt, setsockopt
- htonl, htons, ntohl, ntohs
- inet_addr, ihet_ntoa
- ioctlsocket
- listen
- recv, recvfrom
- select
- send, sendto
- shutdown
- socket

ถ้าหากเราไม่เคยมีประสบการณ์การเขียนโปรแกรมซ้อกเก็ตมาก่อน ฟังก์ชันทั้งสี่คือ htonl, htons, ntohl และ ntohs มีความพิเศษคือใช้สำหรับใช้แปลงตัวเลขซึ่งแทนอินเทอร์เนตแอดเดรส อินเทอร์เนตแอดเดรสสามารถแทนด้วยตัวเลขจำนวน 4 ชุดซึ่งคั่นด้วยเครื่องหมาย "." เช่น 198.53.145.3 ซึ่งตัวเลขแต่ละจำนวนมีค่าตั้งแต่ 0 ถึง 255 สามารถซึ่งแทนด้วยรหัส 8 บิท ดังนั้น เลขทั้งสี่จะแทนด้วยรหัส 32 บิท เป็นที่น่าเสียดายว่าการแทนเลข 32 บิทสามารถทำได้สองวิธี

วิธีแรกคือแทนตัวเลขซ้ายสุด (ในกรณีนี้คือ 198) แทนด้วย 8 บิทบนสุด (8 most significant bits) ตัวเลขถัดมาทางขวาก็แทนด้วย 8 บิทที่อยู่ถัดมาข้างล่าง ทำอย่างนี้จนครบ 32 บิท เป็นการใส่จากบิทสูงมาบิทต่ำเรียกว่า "การเรียงลำดับแบบบิกอินเดียน" (big-Endian order) อีกวิธีหนึ่งจะแทนตัวเลขซ้ายสุดด้วย 8 บิทที่อยู่ต่ำสุดแล้วไล่ขึ้นมาจนครบ 32 บิท วิธีนี้เรียก "การเรียงลำดับแบบลิตเติลอินเดียน" (little-Endian order) ยกตัวอย่างเช่น แอดเดรส 0.0.0.1 จะถูกแปลงเป็น 1 ในการเรียงแบบบิกอินเดียนหรือ 16,777,216 (เท่ากับ 2^{24}) ถ้าหากเครื่องสองเครื่องที่ใช้การเรียงลำดับต่างกันจำเป็นต้องแลกเปลี่ยนแอดเดรสกัน เครื่องหนึ่งส่งตัวเลขแบบบิกอินเดียน แต่อีกเครื่องส่งตัวเลขแบบลิตเติลอินเดียน และนี่เป็นปัญหาที่ชัดเจนที่สุดของการสร้างช่องทางการติดต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบอินเทอร์เนตใช้การเรียงลำดับแบบบิกอินเดียน แต่เครื่องของเราใช้การเรียงลำดับแบบลิตเติลอินเดียน (เครื่องที่ใช้ไมโครโปรเซสเซอร์ของอินเทลใช้การเรียงลำดับแบบลิตเติลอินเดียน ขณะที่เครื่องที่ใช้ไมโครโปรเซสเซอร์ของโมโตโรล่า รวมทั้งเครื่องแมกอินทอชใช้การเรียงลำดับแบบบิกอินเดียน) ซ้อกเก็ตฟังก์ชัน htonl() แปลง 32 บิทโฮสแอดเดรสไปเป็นเน็ตเวิร์กแอดเดรส (32 บิทเรียงลำดับแบบบิกอินเดียน) ทั้งนี้โฮสแอดเดรสเดิมจะเป็นการเรียงลำดับแบบใด ก็จะมี DLL ที่ออกแบบมาเหมาะสมเพื่อใช้ในการแปลงนี้ ส่วนฟังก์ชัน htons() ก็ทำแบบเดียวกันแต่กับ 16 บิทโฮสแอดเดรส และการแปลงกลับเป็นโฮสแอดเดรสก็ใช้ฟังก์ชัน ntonl() และ ntons()

โดยทั่วไปฟังก์ชันที่ใช้สำหรับแปลงค่าแอดเดรสเหล่านี้จะถูกนำไปใช้เมื่อมีเครื่องหนึ่งต้องการผ่านค่าอินเทอร์เนตแอดเดรสหรือค่าพอร์ตไปให้เครื่องอื่น และอาจถูกนำไปใช้ในกรณีอื่น (ซึ่งหายาก)

นอกจากนี้ฟังก์ชัน inet_addr() แปลงจากสตริงของตัวอักขระ เช่น "198.53.145.3" ไปเป็นเน็ตเวิร์กแอดเดรส และในทางกลับกันฟังชัน inet_ntoa() เปลี่ยนเน็ตเวิร์กแอดเดรสไปเป็นสตริงของตัวอักขระ ASCII

ส่วนฟังก์ชันของเบิร์กเลย์ที่เหลือจะได้กล่าวถึงต่อไป

2.3.5 ฟังก์ชันฐานข้อมูล

ถึงแม้ว่าฟังก์ชันเหล่านี้เรียกว่าเป็นฟังก์ชันฐานข้อมูล แต่ก็ไม่จำเป็นที่จะใช้หาข้อมูลในฐานข้อมูลเสมอไป ยกตัวอย่างเช่นอาจใช้ส่งรีเควสไปเครื่องอื่นที่อยู่บนอินเทอร์เนตเพื่อที่จะค้นหาเครื่องที่ต้องการ

- gethostbyaddr
- gethostname
- gethostbyname
- getprotobyname
- getservbyname
- getservbyport

ฟังก์ชันเหล่านี้สามารถแปลงกลับไปกลับมาได้ระหว่างตัวเลขที่เข้ารหัสไว้ (cryptic number) กับโค้ด (code) และชื่อเครื่อง ตัวอย่างเช่นฟังก์ชัน gethostbyname() จะให้ค่า 32

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บิทเน็ตเวิร์กแอดเดรสของเครื่องที่มีชื่อที่กำหนดให้ (เช่น ftp.microsoft.com เป็นต้น) และเพื่อหลีกเลี่ยงความยุ่งยากของการกำหนดพอร์ต เราสามารถเรียกฟังก์ชัน getservbyname() เพื่อแปลงจากชื่อของบริการอย่างเช่น "ftp" เป็นชื่อของพอร์ตมาตรฐานที่ใช้ ในที่นี้คือพอร์ต 21 ได้

2.3.6 ส่วนที่วินโดวส์เพิ่มเติมเข้ามา

เนื่องจากการเขียนโปรแกรมบนวินโดวส์เป็นการเขียนโดยใช้แมสเซส (messege-based) ซึ่งเป็นระบบที่เป็นแบบอซิงโครนัส นั่นคือสิ่งหนึ่งสิ่งใดไม่จำเป็นต้องเกิดตามลำดับงาน (task) หนึ่งไม่จำเป็นต้องเสร็จก่อนที่จะเริ่มต้นทำงานอื่นเสมอไป เมื่อวินโดว์ต้องการทำสิ่งหนึ่งสิ่งใด เช่น การรีดรอว์ (redraw) วินโดว์ วินโดว์หนึ่ง มันจะส่งแมสเซสไปวินโดว์นั้น ("จงรีดรอว์ ตัวเอง") แล้วหันไปทำสิ่งอื่น ๆ ไป เมื่อวินโดว์นั้นได้รับแมสเซสนี้มันก็จะปฏิบัติตามคำสั่งในแมสเซส

แต่โดยทั่วไปการเขียนโปรแกรมโดยใช้ซ้อกเก็ตจะเป็นระบบแบบซิงโครนัส โปรแกรมจะขอให้ตัวซ้อกเก็ตส่งข้อมูลแล้วจะรอคำคอบ ดังนั้นการนำซ้อกเก็ตไปใช้ในวินโดวส์ต้องการแนวทางการทำงานแบบใหม่ ในโลกของวินโดวส์ซ้อกเก็ตทำการส่งรีเควสและทำการเรียบเรียงสำหรับแมสเซสหนึ่งเพื่อถูกละทิ้งเมื่อคำตอบมาถึง ขณะที่ส่วนหนึ่งของโปรแกรมกำลังรอการตอบสนอง (response) แมสเซสอื่น ๆ สามารถถูกจัดการ (handle) ได้ วินโดวส์ซ้อกเก็ตได้กำหนดฟังก์ชันเพิ่มเติม ซึ่งทั้งหมดนี้ทำงานแบบอซิงโครนัส ซึ่งได้แก่ฟังก์ชัน

- WSAAsyncGetHostByAddr
- WSAAsyncGetHostByName
- WSAAsyncGetProtoByName
- WSAAsyncGetProtoByNumber
- WSAAsyncGetServByName
- WSAAsyncGetServByPort
- WSAStartup, WSACleanup

ฟังก์ชัน 6 ฟังก์ชันแรกเป็นอซิงโครนัสเวอร์ชันของฟังก์ชันฐานข้อมูลที่มีชื่อเดียวกันนั้น ฟังก์ชันเหล่านี้ส่งรีเควสหนึ่งเพื่อขอชื่อโฮสหรือหมายเลขบริการหรืออะไรอื่น ๆ แทนที่จะรอคำตอบที่จะตอบกลับมา ฟังก์ชันเหล่านี้ขึ้นกับแมสเซสที่ถูกส่งมาพร้อมกับการตอบสนอง โปรแกรมเมอร์จะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม่มีการ "รอ" ถึงแม้ว่าการทำ "บล็อกกิ้ง" เป็นศัพท์ทางเทคนิคสำหรับฟังก์ชันซ้อกเก็ตซึ่งอาจไม่คืนการตอบสนองตราบใดที่มันรอการตอบสนองจากเครือข่าย

เพื่อที่จะหลีกเลี่ยงการทำบล็อกกิ้งของซ้อกเก็ตฟังก์ชันใด ๆ ซึ่งเรียกใช้ฟังก์ชัน WSAAsyncSelect() ยกตัวอย่างเช่นฟังก์ชัน recv() ฟังก์ชันนี้ทำให้โปรซีดเจอร์ของเราสามารถที่จะ "ลงทะเบียนความสนใจ" (register interest) ในเหตุการณ์ที่เกิดขึ้นในเครือข่าย เช่น เมื่อซ้อกเก็ตนั้นพร้อมที่จะถูกอ่านถูกอ่านค่า ฟังก์ชัน WSAAsyncSelect() ส่งแมสเซสหนึ่งชื่อว่า WM_SOCKET_NOTIFY() ไปยังซ้อกเก็ตนั้นเมื่อมีข้อมูลที่จะถูกอ่านหรือการติดต่อเสร็จสมบูรณ์ (completed connection)

สองสิ่งที่ซ้อกเก็ตโค้ดสามารถทำได้เมื่อซ้อกเก็ตพร้อมสำหรับการถูกรีเควส (request) สิ่งแรกคือการจัดการแบบโลคอล (locally) ยกตัวอย่างเช่น การเติมตัวอักขระที่เข้ามาทางช่องทางการติดต่อลงในอาเรย์ (array) อาเรย์หนึ่ง และรอให้ออบเจกอื่นรีเควสข้อมูลจากอาเรย์ แบบที่สองคือเพื่อสร้างคอลแบ็กและทำเพียงแค่ให้บัฟเฟอร์ของข้อมูลแก่รูทีนที่จัดการแมสเซส (message handling routine) ของออบเจกอื่นซึ่งเป็นคนบอกซ้อกเก็ตให้ทำแอกชันที่ถูกรีเควส

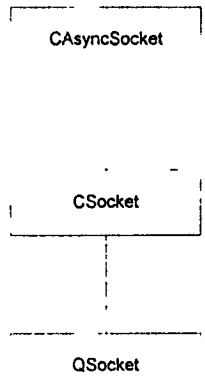
วิธีจัดการแบบโลคอลนั้นจะใช้ในดีกว่าถ้าหากโค้ดที่เรียกใช้ไม่สามารถทำสิ่งอื่นจนกว่ามันจะได้รับคำตอบจากเครือข่าย ยกตัวอย่างเช่นโค้ดที่เรียกใช้อาจส่งรหัสผู้ใช้ (userid) แล้วรอจนกระทั่งได้รับการตอบรับ จากนั้นส่งรหัสผ่าน (password) ส่วนวิธีสร้างคอลแบ็กจะใช้ได้ดีในกรณีที่เราต้องการทำมัลติทาสก์ระหว่างที่กำลังรับข้อมูล ยกตัวอย่างเช่น ถ้าเรากำลังเคลื่อนย้ายไฟล์ขนาดใหญ่ ๆ โดยทั่วไปเราจะใช้ทั้งสองวิธีแล้วแต่ว่าอยู่ในสถานะการณ์ใด

ฟังก์ชัน WSAStartup() และฟังก์ชัน WSACleanup() เป็นฟังก์ชันที่มีความสำคัญเป็นอย่างยิ่ง ชื่อฟังก์ชันก็บอกอยู่แล้วว่าเป็นฟังก์ชันที่ต้องเรียกก่อนที่จะเริ่มเรียกซ้อกเก็ตใด ๆ และต้องเรียกเมื่อจบโปรแกรมเพื่อทำการคลีนอัพ อย่างไรก็ตามโชคดีที่เราไม่ต้องเรียกฟังก์ชันเหล่านี้ตรง ๆ แต่มันจะถูกเรียกใช้อัตโนมัติโดยวินโดวส์

2.3.7 การรวมฟังก์ชันและดาต้าลงในซ้อกเก็ตคลาส

เมื่อข้อตกลงวินโดวส์ ซ้อกเก็ตได้ถูกนำออกเผยแพร่ บรรดาอินเทอร์เนตโปรแกรมเมอร์ซึ่งใช้ภาษา C++ ก็เริ่มที่จะเขียนคลาส "สำเร็จ" ของฟังก์ชันใน API ในคอมไพล์เลอร์วิชวล C++ 2.1 และ 1.52 มีสองคลาสคือ Csocket และ CAsyncSocket อย่างไรก็ตามคลาสเหล่านี้ยังมีจุดอ่อนอยู่ แต่เนื่องจากคุณสมบัติการทำอินเฮอริแทนต์ (inheritant) ของ C++ เราจะพัฒนาซ้อกเก็ตคลาสนี้ให้มีชื่อว่า QSocket ซึ่งมีต้นฉบับมาจาก CSocket ของไมโครซอฟต์ โดยพัฒนาส่วน CAsyncSocket ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 การพัฒนาของ QSocket

2.3.8 การใช้ซ้อกเก็ตคลาส

บางครั้งการแสดงตัวอย่างการใช้ซ้อกเก็ตคลาสอย่างง่ายที่สุดน่าจะเป็นการแสดงพับบลิกเมมเบอร์ฟังก์ชัน (public member function) ของ QSocket แล้วจากนั้นพิจารณาแต่ละฟังก์ชัน ฟังก์ชันในเฮดเดอร์ไฟล์มีดังต่อไปนี้

```
QSocket(BOOL create_socket = TRUE); // constructor
BOOL  Disconnect(void);
BOOL  SetReceiveTarget(CWnd *window, UINT message);
void    Send(const  Cstring& data);
void    SendRaw(const  void* data, const  int dataLen);
Cstring  GetLine(void);
BOOL  Listen(int  back_log  = 5);
QSocket  *Accept();
void    Linger();
SocketStatus   GetStatus(void)                { return  CurrentStatus; }
CString          GetErrorString(void)    { return  ErrorString; }
```

เพื่อทำความเข้าใจการทำงานของฟังก์ชันเหล่านี้ เราจะมาดูไพรเวทดาต้าเมมเบอร์ของ Qsocket

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SocketStatus    CurrentStatus;
CString         ErrorString;
CtringList      RecieveLines;
CString         RemainingRecieve;
CStringList     SendLines;
char            *RawSendData;
int             RawSendDataLength;
CWnd*           RecieveWindow;
UINT            RecieveMessege;
```

ตัวแปร CStringList เป็นลิงก์ลิสต์ของ Cstrings และเป็นข้อมูลที่กำลังถูกรับหรือส่ง แบ่งออกเป็นบรรทัดโดย CR-LF ส่วนตัวแปร RecieveWindow เป็นพ้อยเตอร์ของ CWnd และตัวแปร RecieveMessage เป็นจำนวนเต็มแบบไม่มีเครื่องหมาย ทั้ง RecieveWindow และ RecieveMessage ใช้แสดงวิธีการที่ซ้อกเก็ตคอลแบ็กฟังก์ชันซึ่งเป็นคนรีเควสให้ซ้อกเก็ตส่งหรือรับข้อมูลหรือทำอย่างอื่น ๆ เมื่อการตอบสนองนั้นพร้อม และแมสเซสเดียวกันนี้ก็จะสร้างในกรณีเมื่อใดก็ตามที่สถานะซ้อกเก็ตมีการเปลี่ยนแปลงรวมทั้งเมื่อเปลี่ยนเป็นสถานะบอกความผิดพลาด (error state) หรือยกเลิกการติดต่อ (disconnect)

2.3.9 โครงสร้างของ Qsocket (QSocket Constructor)

คอนสตักเตอร์มีโค้ดดังต่อไปนี้

```
QSocket::Qsocket(BOOL  create_socket)
 : CurrentStatus(UNINITIALIZED), RecieveWindow(0),
  RecieveMessage(0)
{
        if (create_socket)
        {
                if (Start())
                {
                        CurrentStatus = DISCONNECTED;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        }
    }
}
```

ฟังก์ชัน Start() เรียกฟังก์ชัน Create() และเซตค่าของสถานะเป็น ERRORSTATE กรณีฟังก์ชัน Create() เกิดล้มเหลว คอนสตักเตอร์นี้เปลี่ยนสถานะเป็น DISCONNECTED กรณีฟังก์ชัน Start() สำเร็จ ฟังก์ชัน Start() สามารถทำได้แค่เซตค่าสถานะอย่างในกรณีที่กล่าวมาแต่โค้ดเป็นแบบนี้ก็เพื่อ historical reason

2.3.10 Connect Helper

ฟังก์ชันต่อมาคือ Disconnect() แต่เราจะไม่เห็นฟังก์ชัน Connect() ในการประกาศ QSocket เนื่องจาก QSocket ใช้ฟังก์ชัน Connect() ของเดิมของ CSocket อย่างไรก็ดีเราจะเขียนโปรเทกฟังก์ชันชื่อ ConnectHelper() เสียใหม่ซึ่งฟังก์ชันนี้จะเป็นตัวสุดท้ายที่เรียกเพื่อสร้างช่องทางการติดต่อ ฟังก์ชัน Connect() ได้แบ่งเป็นสองส่วนคือส่วนแรกเป็นการหาแอดเดรส (determining address) ซึ่งจะไม่ถูกแก้ไขเปลี่ยนแปลงโดยคลาสที่นำไปดีไรว์ (derive) แต่ส่วนที่เหลือสามารถแก้ไขเปลี่ยนแปลงได้ โปรเทกฟังก์ชัน ConnectHelper() เป็นดังนี้

```
BOOL QSocket::ConnectHelper(const SOCKADDR* lpSockAddr,
 int nSockAddrLen)
{
    CurrentStatus = CONNECTING;
    if (!AsyncSelect(FD_CONNECT))
    {
        SET_ERROR_VARS();
        return FALSE;
    }
    if (connect(m_hsocket, lpSockAddr, nSockAddrLen)
     == SOCKET_ERROR)
    {
        int err = GetLastError();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                if (err != WSAEWOULDBLOCK)
                {
                        SET_ERROR_WARS();
                        return FALSE;
                }
        }
        else
        {
                if (!AsyncSelect(FD_READ | FD_WRITE | FD_CLOSE))
                {
                        SET_ERROR_WARS();
                        return FALSE;
                }
                CurrentStatus = CONNECTED;
                if (ReceiveWindow)
                {
                        RecieveWindow-SendMessage(RecieveMessage,
                                (WPARAM) SocketStatusChanged,
                                (LPARAM) 0);
                }
        }
        return FALSE;
}
```

ฟังก์ชัน AsyncSelect() เป็นฟังก์ชันของคลาส CSocket ซึ่งเรียกฟังก์ชัน WSAAsyncSelect() และเรียบเรียงแมสเซสมาตรฐานเพื่อส่งไปเมื่อซ้อกเก็ตพร้อมสำหรับคำสั่งที่รีเควสมาเมื่อซ้อกเก็ตพร้อมแมสเซส WM_SOCKET_NOTIFY ก็จะถูกส่งไปยังวินโดว์ที่ซ่อนอยู่ของซ้อกเก็ต (socket's hidden window) ซึ่งจากนั้นจะเรียกเวอร์ชวลฟังก์ชันที่เหมาะสมของซ้อกเก็ตคอลแบ็กที่สามารถแก้ไขใหม่ได้แก่ Onrecieve(), Onwrite(), OnOutofBandData(), OnAccept(), OnConnect() และ Onclose()

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เราเริ่มด้วยการเรียกฟังก์ชัน AsyncSelect() และบอกว่าเราต้องการใช้ซ้อกเก็ต ถ้าหากมันส่งค่า FALSE แสดงว่ามีสิ่งผิดปกติเกิดขึ้นและเราถูกยกเลิกการติดต่อ

ขั้นต่อมาคือเรียกฟังก์ชัน Connect() ซึ่งเป็น API ฟังก์ชันซึ่งทำหน้าที่เชื่อมต่อจริง ๆ ระหว่างซ้อกเก็ตกับเครื่องอื่นบนเครือข่ายอินเทอร์เนต ถ้าฟังก์ชันนี้ส่งค่าความผิดพลาดที่ไม่ใช่ค่า WSAEWOULDBLOCK มาแล้วแสดงว่าเรามีปัญหาและถูกยกเลิกการติดต่อ แต่ถ้าหากค่าความค่าความผิดพลาดเป็นค่า WSAEWOULDBLOCK แล้วแสดงว่าต้องใช้เวลาลักครู่ การติดต่อจึงจะสมบูรณ์ ดังนั้นเราก็ไม่ต้องทำอะไรเพิ่มเติม เมเธิร์ดฟังก์ชัน OnConnect() จะถูกเรียกใช้เมื่อซ้อกเก็ตได้รับการเชื่อมต่อเรียบร้อยแล้ว และหากไม่มีข้อผิดพลาดใด ๆ เราก็จะสร้างช่องทางการติดต่อได้สำเร็จ ต่อมาเราเรียกฟังก์ชัน AsyncSelect() อีกครั้งเพื่อบอกว่าเราสนใจที่จะอ่านค่าจากที่ใดจะเขียนค่าไปที่ใด รวมถึงจะปิดซ้อกเก็ตนี้ และทันทีที่มันพร้อมสำหรับการกระทำทั้งสามเราก็จะเรียก เมดเธิร์ดฟังก์ชัน OnRecieve(), OnWrite() หรือ OnClose() จะถูกเรียก

มาถึงจุดนี้ซ้อกเก็ตถูกติดต่อเรียบร้อยแล้ว ดังนั้นเราจะเซตค่าสถานะเป็น CONNECTED เนื่องจากเราทำให้เป็นไปได้ที่จะบอกซ้อกเก็ตที่ต้องการคอลแบ็กหลังจากการติดต่อได้ เราผ่านแมสเซสนั้นโดยเรียกเมเธิร์ดที่ใช้ส่งแมสเซสของ RecieveWindow

2.3.11 Disconnect

เนื่องด้วยการติดต่อของเราต้องสร้างช่องทางการติดต่อ ตอนนี้เราจะมาดูกันที่ฟังก์ชัน Disconnect() มันทำหน้าที่เดียวกับฟังชัน Close() ซึ่งอยู่ในเบสคลาสของเรา แต่เราดัดแปลงนิดหน่อยเพื่อให้สามารถใช้ฟังก์ชัน Linger() ได้ ซึ่งฟังก์ชัน Linger() นี้จะได้กล่าวถึงต่อไป ฟังก์ชัน Linger() มีรายละเอียดดังนี้

```
BOOL QSocket::Disconnect(void)
{
        // Allow socket related messages to be processed
        // before disconnect
        for (;;)
        {
                MSG msg;
                if (PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE)
    && msg.message != WM_QUIT)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                {
                        if    (PeekMessage(&msg,  NULL,  0,  0,
PM_REMOVE))
                        {
                                TranslateMessage(&msg);
                                DispatchMessage(&msg);
                        }
                }
                else
                {
                        break;
                }
        }
        // You don't just call CSocket::Close() because you want to
        // disable non-blocking mode before the closesocket() is
        // called in the Casyncsocket::Close().
        // You disable the non-blocking mode so that when you have
        // linger on you will block rather than return
        // a WSAEWOULDBLOCK.
        unsigned long nbIO = 0;
        CancelBlockingCall();
        AsyncSelect(0);
        // disable non-blocking mode
        ioctlsocket(m_hsocket, FIONBIO, &nbIO);
        CAsyncSocket::Close();
        m_hSocket = INVALID_SOCKET;
        return TRUE;
}
```

ฟังก์ชันเริ่มทำงานด้วยการปั๊มแมสเซส (message pump) ซึ่งจะเป็นการเคลียร์แมสเซสใด ๆ ที่มีผลต่อซ้อกเก็ตนี้ ยกตัวอย่างเช่นแมสเซสซึ่งบอกว่าซ้อกเก็ตพร้อมที่จะให้ข้อมูลที่เราต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแก่เรา แมสเซสนี้จะไปเรียกฟังก์ชัน OnRecieve เราต้องการให้แมสเซสเหล่านี้ถูกประมวลผลก่อนการปิดซ้อกเก็ต

ก่อนที่เราจะปิดซ้อกเก็ต เราต้องทำการยกเลิกโหมดการทำงานแบบนอน-บล็อกกิ้ง (non-blocking mode) ในโหมดการทำงานแบบนอน-บล็อกกิ้ง การเรียกฟังก์ชัน Close() จะให้ข้อผิดพลาดถ้าหากยังมีข้อมูลที่จะต้องถูกส่ง การแก้ไขคือใช้ฟังก์ชัน Linger() ดังนั้นจึงเหมือนว่าฟังก์ชัน Linger() เป็นตัวไปปิดโหมดการทำงานแบบนอน-บล็อกกิ้งก่อนการเรียกฟังก์ชัน Close() เราทำแบบนี้ด้วยการเรียกฟังก์ชัน ioctlsocket() แต่ถ้าหากมีการลงทะเบียนเหลืออยู่จากการเรียกฟังก์ชัน AsynSelect() ก่อนหน้านี้ก็จะทำให้การเรียกฟังก์ชัน ioctlsocket() ล้มเหลว ดังนั้นเราต้องเรียกฟังก์ชัน AsynSelect(0) ก่อนเพื่อยกเลิกการเรียบเรียงใด ๆ แต่ยังคงทำงานในทางกลับกันก่อนหน้าที่เราสามารถเรียกฟังก์ชัน AsynSelect(0) เราต้องเรียก CancelBlockingCall() เพื่อยกเลิกการเรียบเรียงใด ๆ ซึ่งเกิดขึ้นก่อนหน้า

ดังนั้นเราเขียนเรียบลำดับแบบกลับหลัง เราเรียกฟังก์ชัน CancelBlockingcall() เพื่อกำจัดการเรียกก่อนหน้า (previous calls) ฟังก์ชัน AsyancSelect() เพื่อยกเลิกการเรียบเรียง เพื่อเรียกฟังก์ชันของเราเมื่อซ้อกเก็ตพร้อมที่จะรับส่งข้อมูล ฟังก์ชัน ioctlsocket() ใช้ปิดโหมดการทำงานแบบนอน-บล็อกกิง และท้ายสุดฟังก์ชัน CSyncSocket::Close() เพื่อปิดซ้อกเก็ตจริง ๆ

เมื่อซ้อกเก็ตถูกปิด เซตค่าของแฮนเดิลเป็น INVALID_SOCKET เพื่อให้แน่ใจว่าไม่มีใครลามารถใช้มันอีก

2.3.12 SetRecieveTarget

ฟังก์ชัน SetRecieveTarget() เป็นวิธีที่เราให้ออบเจกอื่นเรียบเรียงเพื่อทำการคอลแบ็ก

```
BOOL QSocket::SetRecieveTarget(CWnd *window, UINT message)
{
    // Sets who to send all received data to.
    RecieveWindow = windows;
    RecieveMessage = message;
    // Clear receive buffer as the data mst be left over.
    RecieveLines.RemoveAll();
    RemainingRecieve = "";
    // Set up AsyncSelect.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Switch (CurrentStaus)
{
        case CONNECTED:
                if (!AsyncSelect(FD_READ | FD_WRITE |
                                                FD_CLOSE))
                {
                        SET_ERROR_VARS();
                        return FALSE;
                }
                break;
        case LISTENING:
                if (!AsyncSelect(FD_ACCEPT))
                {
                        SET_ERROR_VARS();
                        return FALSE;
                }
                break;
        default:
                if (!AsyncSelect(FD_CONNECT | FD_CLOSE))
                {
                        SET_ERROR_VARS();
                        return FALSE;
                }
                break;
}
return FALSE;
}
```

ฟังก์ชันนี้เซตค่าเมมเบอร์วาเรียเบิลซึ่งเก็บวินโดว์และแมสเซสเพื่อใช้คอลแบ็ก และเคลียร์สิ่งที่อยู่ใน CStringList RecieveLines **และ** CString RemainingRecieve **เนื่องจากฟังก์ชันคอลแบ็กตัวใหม่จะไม่ต้องการให้ข้อมูลคนอื่นส่งไปที่มัน**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขณะนี้เราจำเป็นต้องทำซ้ำฟังก์ชัน AsyncSelect() ถ้าซ้อกเก็ตนี้ถูกสร้างขึ้นแล้วโดยการตอบรับอันหนึ่งอันใด ถ้าหากไม่ ฟังก์ชัน AsyncSelect()จะถูกเรียกในฟังก์ชันนี้ ซ้อกเก็ตที่ติดต่อเรียบร้อยแล้ว (connected socket) จะบ่งชี้ว่าต้องการอ่าน เขียนหรือปิดซ้อกเก็ตที่รอการตอบรับ (listening socket) จะต้องการการเรียกและซ้อกเก็ตอื่น ๆ จะต้องการเปลี่ยนเป็นซ้อกเก็ตที่ได้รับการติดต่อเรียบร้อยหรือซ้อกเก็ตที่รอการตอบรับ

2.3.13 Send

ฟังก์ชัน Send() ส่ง CString ผ่านซ้อกเก็ตที่ติดต่อเรียบร้อยแล้ว ไปยังเครื่องอื่น

```
void QSocket::Send(const CString& data)
    {
        if (data.GetLength() == 0)
        {
            return;
        }
        BOOL send_buffer_empty = SendLine.IsEmpty();
        SendLines.AddTail(data);
        if (send_buffer_empty)
        {
            int amt = SendLine.GetHead().GetLength();
            amt = CAsyncSocket::Send(SendLines.GetHead(), amt,
0);
            if (amt == SOCKET_ERROR)
            {
                int error = GetLastError();
                if (error != WSAEWOULDBLOCK
                        && error != WSAEINPROGRESS)
                {
                    SET_ERROR_VARS();
                }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        }
        else
        {
            if (amt == SendLines.GetHead().GetLenght())
            {
                SendLines.RemoveHead();
            }
            else
            {
                // Only part of line was sent;
                // leave rest for later.
                const char * head=SendLines.GetHead();
                strcpy((char *)head, head+amt);
            }
        }
    }
}
```

ฟังก์ชันนี้ได้รับสตริงและเพิ่มสตริงนั้นลงใน CStringList SendList ถ้าลิสต์นั้นว่างอยู่แล้วไม่มีรีเควสเพนดิ้งส่งมาและเราต้องเรียกฟังก์ชัน Send() จาก CAsyncSocket ถ้าใน SendList มีข้อมูลอยู่แล้วเราทำเพียงนำข้อมูลลงไปต่อท้าย มันจะถูกส่งโดยฟังก์ชัน Onwrite() เมื่อซ้อกเก็ตพร้อม

ฟังก์ชัน CAsyncSocket()::Send() ให้ค่าของอักขระจำนวนหนึ่งซึ่งมันสามารถส่งได้ ซึ่งบางครั้งไม่เต็มบรรทัด ส่วนที่ไม่ได้ส่งจะหลงเหลืออยู่ในลิสต์ และส่วนที่เหลือนี้จะถูกส่งโดยฟังก์ชัน Onwrite() เมื่อซ้อกเก็ตพร้อม

2.3.14 SendRaw

ฟังก์ชันนี้ถูกใช้ในการส่งข้อมูลไบนารี่ ซึ่งต่างกับฟังก์ชัน Send() ซึ่งเป็นฟังก์ชันสำหรับส่งข้อมูลเป็นบรรทัด (line oriented)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void QSocket::SendRaw(const void *data, const int datalen)
{
        char *sdata = new char[RawSendDataLength + datalen];
        memory(sdata, RawSendData, RawSendDataLength);
        memcpy(sdata+RawSendDataLength, data, datalen);
        delete RawSendData;
        RawSendData = sdata;
        RawSendDataLength += datalen;
        while (RawSendDataLEngth > 0)
        {
                int amt_sent = CAsyncSocket::Send((char)
                                RawSendData, RawSendDataLength, 0);
                if (amt_sent == SOCKET_ERROR)
                {
                        int error = GetLastError();
                        if (error != WSAEWOULDBLOCK && error !=
                                                WSAEINPROGRESS)
                        {
                                SET_ERROR_VARS();
                        }
                        break;
                }
                else
                {
                        RawSendDataLength -=amt_sent;
                                memcpy(RawSendData,
                                RawSendData+amt_sent,
                                RawSendDataLength);
                }
        }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นแรกบัฟเฟอร์ของข้อมูลที่ data ชี้อยู่จะถูกต่อท้ายที่ RawSendData และ RawSendDataLenght จะถูกเพิ่มเท่ากับ datalen เรียกฟังก์ชัน CAsyncSocket::Send() เพื่อจะส่งจำนวนที่เราไม่ทราบค่าของอักขระ ดังนั้นเราต้องตรวจสอบ amt_sent เพื่อให้ทราบจำนวนที่แท้จริง ในการที่จะอ่านค่าอักขระออกจากในบัฟเฟอร์ RawSendData เพียงแค่คัดลอกบัฟเฟอร์จากหลังไปข้างหน้าและปรับค่าใน RawSendDataLength

2.3.15 Getline

ฟังก์ชันนี้เป็นการใช้อซิงโครนัสซ้อกเก็ตแบบซิงโครนัสเทียม (psudo-synchronous) การเรียกฟังก์ชัน GetLine() จะไม่กลับจนกว่าซ้อกเก็ตได้ส่งหนึ่งบรรทัดออกไป (หนึ่งบรรทัดคือ CString ซึ่งมี CR-LF) และมันจะคืนค่าเป็นบรรทัดนั้น อย่างไรก็ดีมันไม่ได้เป็นบล็อกกิ้งฟังชันอย่างแท้จริงเนื่องจากมันจะปั๊มวันโดว์แมสเซสขณะที่มันกำลังรอ

```
CString QSocket::GetLine(void)
{
    if (RecieveWindow)
    {
        // You are in the wrong mode to use GetLine().
        Return "";
    }
    while (CurrentStatus == CONNECTED &&
RecieveLines.IsEmpty())
    {
        char temp[SOCK_BLOCK_SIZE+1];
        int amt = Receive(temp, SOCK_BLOCK_SIZE, 0);
        if (amt == SOCKET_ERROR)
        {
            SET_ERROR_VARS();
        }
        else if (amt == 0)
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    OnClose(0);
                }
                else
                {
                    temp[amt] = 0;
                    // Add to buffer you are using GetLine() mode.
                    AddToRecieve(temp, amt);
                }
            }
            if (!RecieveLines.IsEmpty()) {
                return RecieveLines.RemoveHead();
            }
            else {
                return "";
            }
        }
```

เรียกฟังก์ชัน GetLine() เมื่อมีการเซต RecieveWindow อยู่เป็นสิ่งที่ไม่มีประโยชน์ ดังนั้นสิ่งนี้เป็นสิ่งแรกที่ต้องตรวจสอบ จากนั้นตราบใดที่ซ้อกเก็ตได้รับการติดต่อแต่ยังรับค่าครบหนึ่งบรรทัด เราเรียกฟังก์ชัน CSocket Receive() (ซึ่งจะคอยปั๊มแมสเซลขณะที่รอ) เพื่อรอการเติมบัฟเฟอร์ ถ้าฟังก์ชัน Receive() ให้ค่า 0 แสดงว่าไม่มีข้อมูลมากกว่านี้อีกแล้วดังนั้นจึงแยกฟังก์ชัน Close() ถ้าเราจะรับข้อมูลให้เรียกฟังก์ชัน AddToRecieve() เพื่อส่งค่าใน temp ไปเป็นบรรทัด ๆ ที่แยกกันด้วย CR-LF มันจะไม่เพิ่มข้อมูลที่ไม่เต็มบรรทัดลงใน RecieveLines แต่จะเก็บใน RemainingRecieve สำหรับครั้งต่อไป

เมื่อเราออกจากลูป while แล้วถ้าหากยังมีบรรทัดหนึ่ง (หรือมากกว่า) เหลืออยู่ใน RecieveLines เราก็คืนค่านั้นกลับมายังโปรแกรม สังเกตว่าถ้าเรามีสองบรรทัดในการคืนค่ากลับจะมีผลให้บรรทัดที่สองยังคงอยู่ใน RecieveList และเมื่อเรียกฟังก์ชัน GetLine() ครั้งต่อไปเราไม่ได้เข้าในลูป while แต่จะคืนค่าบรรทัดที่สองนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.16 Listen

ฟังก์ชัน Listen() ใช้เรียบเรียงซ้อกเก็ตเพื่อตอบสนองการขอสร้างช่องทางการติดต่อสำหรับพอร์ตที่กำหนด (ซอร์สโค้ดที่เรียกฟังก์ชัน Listen() จะสร้างและไบน์ซ้อกเก็ตไปที่พอร์ตพอร์ตหนึ่งก่อน) นี่เป็นวิธีที่เราสร้างช่องทางการติดต่อซึ่งถูกเริ่มโดยเครื่องอื่นแทนที่จะเป็นเครื่องของเรา ฟังก์ชัน Listen() ใช้เป็นพื้นฐานในการสร้างเซิร์ฟเวอร์แอพพลิเคชัน

```
BOOL QSocket::Listen(int back_log)
{
        if (RecieveWindow != NULL)
        {
                // Receive callbacks when status changes
                // so set up AsyncSelect
                if (!AsyncSelect(FD_ACCEPT))
                {
                        SET_ERROR_VARS();
                        return FALSE;
                }
        }
        if (!CAsyncSocket::Listen(Back_log))
        {
                SET_ERROR_VARS();
                return FALSE;
        }
        CurrentStatus = LISTENING;
        return TRUE;
}
```

ถ้าหากโค้ดที่เรียกฟังก์ชัน Listen() ถูกเตรียมพร้อมเพื่อรับคอลแบ็กเมื่อซ้อกเก็ตได้รับการติดต่อสมบูรณ์ แล้วเราเรียก AsyncSelect() เพื่อบอกว่าเราต้องการถูกเตือนหากมีการสร้างการติดต่อได้ จากนั้นเราเรียก CAsyncSocket::Listen() เพื่อส่งคำสั่งลิสเซิล (listen command)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.17 Accept

ฟังก์ชัน Accept() ถูกเรียกเมื่อซ้อกเก็ตกำลังรอการเรียก (listening) ฟังก์ชันนี้ใช้สร้างช่องการติดต่อใหม่ ๆ บนซ้อกเก็ตตัวหนึ่งซึ่งก็จะคืนค่าพ้อยเตอร์ของซ้อกเก็ตตัวนั้น

```
QSocket         *QSocket::Accept()
{
        QSock *return_socket = new QSocket(FALSE);
        // Do not get the address of remote end.
        if (!CSocket::Accept(*return_socket, NULL, NULL))
        {
                delete return_socket;
                return_socket = 0;
                SET_ERROR_VARS();
        }
        return_socket->CurrentStatus = CONNECTED;
        return return_socket;
}
```

เราสร้างซ้อกเก็ตใหม่แต่ผ่านค่า FALSE ไปในคอนสตรักเตอร์ทำให้ฟังก์ชัน Start() ไม่ถูกเรียก จากนั้นเราเรียก CSocket::Accept() ซึ่งจะให้ค่าซ้อกเก็ตจริง ๆ สำหรับช่องการติดต่อใหม่ เราเซตค่าสถานะให้เป็น CONNECTED แล้วคืนค่าพ้อยเตอร์

2.3.18 Linger

บางครั้งการเขียนโปรแกรมแบบอซิงโครนัสสามารถก่อให้เกิดปัญหาเล็กน้อยได้ ยกตัวอย่างเช่น โปรแกรมของเราอาจเรียก Send() หรือ SendRaw() เพื่อส่งข้อมูลบางอย่างผ่านซ้อกเก็ตจากนั้นเรียก Disconnect() เพื่อปิดซ้อกเก็ต ถึงแม้ว่าข้อมูลอาจยังไม่ถูกส่งไปทั้งหมด การแก้ไขปัญหานี้คือเรียกใช้ฟังก์ชัน linger() ก่อนเรียกฟังก์ชัน Close() การทำแบบนี้จะทำให้ซ้อกเก็ตจะถูกเปิดไว้จนกว่าข้อมูลสุดท้ายจะถูกส่งออกไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void QSocket::Linger(void)
{
        int what = 0;
        struct linger sL;
        sL.l_onoff = 1;
        sL.l_linger = 30;
        what = SetSockOpt(SO_LINGER, (char *)&sL, sizeof(sL));
        if (what == SOCKET_ERROR)
        {
                SET_ERROR_VARS();
        }
}
```

สิ่งที่ฟังก์ชันนี้ทำจริงก็คือเรียกฟังก์ชัน SetSockOpt() จาก Winsock API มันสร้างตัวแปรชนิด linger ชื่อว่า sL และเติมค่า l_onoff ด้วย 1 (จริง ๆ การให้ค่า linger เป็น on ให้ค่าอะไรก็ได้ที่ไม่เท่ากับ 0) และ l_linger เป็น 30 (ระยะเวลาเป็นวินาทีที่อนุญาตให้บิทสุดท้ายถูกรับส่ง) เมื่อฟังก์ชัน SetSockOpt() ถูกเรียกมีผลให้ฟังก์ชัน Disconnect() ที่ถูกเรียกหลังจากนั้นจะรอจนกระทั่งข้อมูลถูกรับส่งเรียบร้อยจึงปิดซ็อกเก็ต

2.3.19 Inherited Function

ฟังก์ชัน 3 ฟังก์ชันของ CSocket ซึ่งไม่ต้องปรับเปลี่ยนแก้ไข นำมาใช้ในคลาส QSock ได้ทันที

```
BOOL Connect(const CString& address, const int port);
BOOL GetSockName(SOCKADDR *sock_addr, int *addr_len);
BOOL Bind(const SOCKADDR *sock_addr, int *addr_len);
```

ฟังก์ชัน Connect() จัดการแอดเดรสซึ่งถูกผ่านเข้าไปในฟังก์ชันและเรียกฟังก์ชัน ConnectHelper() ที่เราได้กล่าวมาแล้ว ฟังก์ชัน GetSockName() นำมาจากฟังก์ชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

getsockname() ของเบิร์กเลย์ ฟังก์ชัน bind() ต้องถูกเรียกก่อน Listen() มันเตรียมซ้อกเก็ตว่าง สำหรับพอร์ตที่กำหนดบนเครื่องของเรา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 Format of a Mail Message
(รูปแบบของข้อความเมล์)

ข้อความของเมล์จะประกอบไปด้วยบรรทัดของตัวอักษร 2-3 บรรทัดแรกนั้นจะเรียกว่า "เฮดเดอร์" (header) ซึ่งจะมีรูปแบบที่แน่นอน บรรทัดต่อจากเฮดเดอร์จะเรียกว่า "บอดี้" (body) ซึ่งจะมีรูปแบบที่ไม่แน่นอน ในความเป็นจริงแล้วในบางกรณีข้อความของเมล์อาจจะไม่มีส่วนของบ่อดี้ก็ได้ ส่วนของบอดี้และเฮดเดอร์นั้นจะถูกแยกออกจากกันด้วยบรรทัดว่าง 1 บรรทัด

ส่วนเฮดเดอร์จะเริ่มต้นด้วยฟิลด์เนม (field name) (ซึ่งจะอธิบายต่อไป) และจะตามด้วยเครื่องหมาย " : " และจะมีส่วนของฟิลด์บอดี้ (field body) ตามมา ส่วนของเฮดเดอร์จะประกอบไปด้วยส่วนต่าง ๆ ดังนี้

* Date
* From,or Send and From
* To,or CC (Carbon Copy),or BCC (Blind Carbon Copy)

และอาจมีส่วนเพิ่มเติมได้อีกดังนี้

- Return-path
- Recieved
- Reply-To
- Message-ID
- In-Reply-To
- References
- Keywords
- Subject
- Comments
- Encrypted

โปรแกรมรับส่งเมล์บางโปรแกรมจะมีการสร้างส่วนของเฮดเดอร์ของตัวเองเพิ่มขึ้นมาเป็นพิเศษ เฮดเดอร์พิเศษจะเริ่มต้นด้วยตัวอักษร " X- " เพราะถ้าส่วนเฮดเดอร์พิเศษถูกเพิ่มใน RFC มันจะไม่เริ่มด้วยตัวอักษรนี้

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เฮดเดอร์อาจจะถูกแยกออกจากกันให้มีมากกว่า 2 บรรทัดก็ได้ แต่ต้องอยู่ภายใต้กฎเกณฑ์ดังต่อไปนี้

1. การแบ่งต้องถูกแทนที่ที่มีช่องว่าง จะเกิดขึ้นเป็นปกติ ตัวอย่างเช่น ตรงกลางระหว่างชื่อของผู้ใช้หรืออะไรที่ใกล้เคียง

2. บรรทัดที่ถูกแยกออกมานั้นจะต้องเริ่มต้นด้วยช่องว่างหรือแทป (tab) เสมอ

ดังนั้น ตัวอย่างของเฮดเดอร์อาจจะเป็น

From:s6014076@diamond (Jittapol Julasrikaiwan)

เฮดเดอร์นี้อาจจะถูกทำเป็น 2 บรรทัดได้ดังนี้

From:s6014076@diamond

(Jittapol Julasrikaiwan)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3 การคำนวณและการสร้าง

โดยปกติแล้วในการสร้างโปรแกรมเมล์ไคลเอนท์ (Mail Client) เราจะไม่ใช้โปรโตคอล SMTP สำหรับทั้งการรับและการส่งเมล์ เนื่องมาจากโปรโตคอล SMTP นั้นจะเหมาะสมในการส่งเมล์ เพราะผู้ใช้จะมีการสร้างการติดต่อ SLIP หรือ PPP ขึ้นมาก่อนจะใช้งานโปรแกรม แต่เราจะใช้โปรโตคอล POP3 ในการรับเมล์ ซึ่งไม่จำเป็นที่จะต้องมีเมล์บ็อกซ์บนเครื่องของผู้ใช้งาน แต่โปร่โตคอล POP3 จะทำการติดต่อกับเมล์บ็อกซ์ให้เองอย่างสมบูรณ์

ดังนั้นผู้ใช้จะต้องทำการติดต่อกับระบบอินเตอร์เน็ตเสียก่อน แล้วจึงเริ่มใช้งานโปรแกรมนี้ ขั้นตอนต่อไปคือเราจะต้องทำการติดต่อกับโฮสท์ของ POP3 (POP3 host) และทำการค้นหาตรวจสอบว่ามีเมล์ของผู้ใช้ที่อยู่ในโฮสท์นั้นหรือไม่ และทำการอ่านข้อความในเมล์ที่ผู้ใช้ต้องการจะอ่านและสามารถทำการลบเมล์บางเมล์ที่ต้องการจะลบออกไป และถ้าผู้ใช้ต้องการที่จะทำการส่งหรือตอบเมล์กลับไปก็จะใช้โปรโตคอล SMTP ทำการติดต่อกับโฮสท์ของ SMTP (SMTP host) และทำงานต่อไปซึ่งโฮสท์ทั้งสองนั้นจะเป็นโฮสท์เดียวกันหรือไม่ก็ได้

ต่อไปจะต้องพิจารณาถึงส่วนที่จะติดต่อกับผู้ใช้ เราจำเป็นจะต้องแสดงให้ผู้ใช้ได้เห็นถึงรายการของเมล์ทั้งหมดในโฮสท์ของ POP3 ที่ผู้ใช้สามาถทำการอ่านได้ เราต้องการส่วนที่แสดงข้อความทั้งหมดในเมล์ที่ผู้ใช้ต้องการจะอ่านโดยแยกจากกัน และสามารถทำการตอบกลับหรือส่งเมล์ออกไป เราจะต้องรู้ถึงแอดเดรส (address) ของโฮสท์ของ POP3 และของ SMTP , อีเมล์แอดเดรส (E-mail address) ของผู้ใช้งาน , รหัสผ่านของโฮสท์ POP3 ของผู้ใช้งานและชื่อจริงของผู้ใช้งาน

ทางหนึ่งในการออกแบบโปรแกรมรับส่งเมล์ก็คือจะต้องมีส่วนที่ใช้สำหรับเก็บบันทึกแอดเดรสของโฮสท์ทั้งของ POP3 และของ SMTP , ชื่อผู้ใช้งาน , รหัสผ่าน และรายละเอียดอื่น ๆ และจะมีส่วนต่าง ๆ ที่แยกออกจากกันเพื่อใช้ในการแสดงรายละเอียดต่าง ๆ ดังนี้ ส่วนที่ใช้ในการแสดงรายการของเมล์ของผู้ใช้ที่มีอยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 , ส่วนสำหรับใช้ในการแสดงข้อความที่มีอยู่ในเมล์ที่ผู้ใช้ต้องการจะอ่าน และสุดท้ายเป็นส่วนที่ใช้สำหรับการส่งหรือตอบเมล์กลับไป โดยในส่วนนั้นจะใช้โปรโตคอล SMTP

ดังนั้นในการออกแบบและสร้างโปรแกรมนี้นั้นจะใช้หลักการของ "มัลติเปิ้ลดอคคิวเมนต์อินเตอร์เฟส" หรือ "เอ็มดีไอ" (Multiple Document Interface,MDI) ซึ่งก็เป็นประเภทเดียวกันกับโปรแกรมอีดิเตอร์ (Editor) โดยทั่วไปกล่าวคือ จะมีหน้าต่างหลายหน้าต่างเพื่อใช้ในการแสดงส่วนที่ต้องการออกมา

แต่โดยทั่วไปแล้วมัลติเปิ้ลดอคคิวเมนต์อินเตอร์เฟสจะมีเมนู (menu) มาตราฐานอยู่ ดังนั้นเราจำเป็นจะต้องมีการสร้างเมนูขึ้นมาเพิ่มเติม เพื่อจะสามารถให้ผู้ใช้งานโปรแกรมนี้ใช้งานได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างเช่น ส่วนที่ต้องใช้ในการตั้งค่าเริ่มต้นต่าง ๆ ของผู้ใช้ก่อนใช้โปรแกรมทำงานต่อไป , ส่วนที่ใช้ในการตรวจสอบและค้นหาเมล์ที่มีอยู่ในเมล์บ็อกซ์ และแสดงออกมาให้ผู้ใช้ได้ทราบ , ส่วนที่ใช้ในการอ่านข้อความในเมล์ที่ผู้ใช้ต้องการจะอ่าน , ส่วนที่ใช้สำหรับทำการตอบเมล์ , ส่วนที่ใช้สำหรับทำการส่งเมล์ , และส่วนที่ใช้สำหรับทำการลบเมล์ที่ไม่ต้องการเก็บไว้ในเมล์บ็อกซ์ออกไป

เพราะฉะนั้นเราจะต้องทำการเพิ่มเมนูและตัวเลือกต่าง ๆ ในเมนูเข้าไปดังนี้ เพิ่มตัวเลือก Setup และตัวเลือก Check Mail ลงไปในเมนู File และสร้างเมนู Message เพิ่มเติมเข้าไปซึ่งในเมนู Message นี้จะประกอบไปด้วยตัวเลือกต่าง ๆ เหล่านี้ Display , Reply , New , Send และ Delete

นอกจากนี้ยังต้องมีการสร้างไดอะล็อกบล็อกขึ้นมาเพื่อใช้ในการกรอกข้อมูลเริ่มต้นต่าง ๆ ของผู้ใช้งานและเชื่อมโยงกับส่วนของ MDI และจะแสดงเมื่อทำการเรียกใช้เพื่อการกรอกข้อมูล

3.1 รายละเอียดตัวเลือกต่าง ๆ ที่เพิ่มเติมเข้ามา

3.1.1 ตัวเลือก Setup ในเมนู File

ขั้นตอนแรกของการใช้งานโปรแกรมนี้จะต้องเริ่มต้นจากการที่ผู้ใช้จะต้องตั้งค่าเริ่มต้นเพื่อนำไปใช้งานในส่วนอื่น ๆ ต่อไป โดยในส่วนนี้จะสร้างเป็นไดอะล็อกบ็อกซ์ (Dialog box) และมีที่สำหรับให้ผู้ใช้กรอกรายละเอียดต่าง ๆ สำหรับค่าเริ่มต้นที่ให้ผู้ใช้กรอกที่สำคัญและจำเป็นต้องใช้ในโปรแกรมนี้มีทั้งหมด 6 อย่าง ได้แก่

1. แอดเดรสของโฮสท์ของ SMTP
2. แอดเดรสของโฮสท์ของ POP3
3. แอดเดรสของเครื่องที่ใช้งาน
4. ชื่อที่ใช้ในการติดต่อกับโฮสท์ของ POP3
5. รหัสผ่านของโฮสท์ของ POP3
6. ชื่อจริงของผู้ใช้งาน

โดยจะทำการเก็บเป็นตัวแปรเพื่อที่จะนำไปใช้งานในโปรแกรมนี้ต่อไปในชื่อของ SMTP , POP3 , Site , Userid , Password และ Username ตามลำดับ

โดยที่ค่าเริ่มต้นทั้งหมดนั้นสามารถเก็บบันทึกเอาไว้ให้อยู่ในรูปแบบของไฟล์เพื่อเรียกใช้งานในครั้งต่อไปได้

ตัวเลือกตัวนี้สามารถเรียกใช้ได้ตลอดการใช้โปรแกรมตามที่ต้องการในกรณีที่มีผู้ใช้คนอื่นต้องการใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.2 ตัวเลือก Check Mail ในเมนู File

เป็นส่วนที่ใช้สำหรับทำการตรวจสอบเมล์ที่มีทั้งหมดของผู้ใช้ที่อยู่ในโฮสท์ของ POP3 และแสดงออกมาให้ผู้ใช้ทราบ

เมื่อผู้ใช้งานเลือกตัวเลือกนี้แล้วโปรแกรมจะต้องทำการติดต่อไปยังโฮสท์ของ POP3 และตรวจสอบว่ามีเมล์ของผู้ใช้ทั้งหมดจำนวนเท่าไหร่ หลังจากนั้นก็ทำการดึงมาและใส่ข้อความอธิบายลงไปและแสดงให้ผู้ใช้ได้เห็น และทำการยกเลิกการติดต่อกับโฮสท์ของ POP3 ซึ่งเราสามารถทำการติดต่อใหม่กับโฮสท์ของ POP3 อีกกี่ครั้งก็ได้ตามที่ผู้ใช้ต้องการ เพื่อที่จะแสดงหรือลบเมล์

สำหรับลำดับการทำงานของตัวเลือกนี้คือ ในขั้นตอนแรกโปรแกรมจะส่งคำสั่ง USER และPASS เพื่อทำการติดต่อกับโฮสท์ของ POP3 ซึ่งค่าที่ส่งไปพร้อมกับคำสั่งนี้คือ Userid และ Password ที่ได้มาจากค่าเริ่มต้นของผู้ใช้งานตามลำดับ แล้วทำการตรวจสอบความถูกต้องของ ผู้ใช้งาน หลังจากนั้นถ้าการติดต่อเสร็จสมบูรณ์ ก็จะทำการตรวจสอบจำนวนของเมล์ที่มีอยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 โดยจะส่งคำสั่ง STAT ออกไปตรวจสอบจำนวน หลังจากนั้นจะส่งคำสั่ง RETR ออกไป เพื่อแสดงรายละเอียดของเมล์ที่มีอยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 ให้ผู้ใช้งานได้ทราบ เพื่อที่จะได้ทำการเลือกเมล์ที่จะทำการอ่านต่อไป หลังจากนั้นจะทำการส่งคำสั่ง QUIT ออกไปเพื่อสิ้นสุดการติดต่อกับโฮสท์ของ POP3

สำหรับในขั้นตอนนี้จะทำการบันทึกเฮดเดอร์ในส่วนของฟิลด์บอดี้ของ "From:" และ "Subject:" เอาไว้เพื่อนำไปใช้ในส่วนของการตอบเมล์ต่อไป

ตัวเลือกนี้สามารถเรียกใช้งานได้ตลอดการใช้โปรแกรมเมื่อผู้ใช้ต้องการทำการตรวจสอบเมล์ที่มีอยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3

3.1.3 ตัวเลือก Display ในเมนู Message

เป็นส่วนที่ใช้ในการแสดงข้อความที่มีอยู่ในเมล์ที่ผู้ใช้งานต้องการอ่าน

เมื่อผู้ใช้งานเลือกตัวเลือกนี้แล้วโปรแกรมจะทำการสร้างหน้าต่างขึ้นมาใหม่เพื่อใช้ในการแสดงข้อความที่มีอยู่ในเมล์ที่ผู้ใช้ต้องการจะอ่านโดยอัตโนมัติ

ลำดับขั้นตอนของการทำงานของตัวเลือกนี้คือจะเริ่มจากการส่งคำสั่ง USER และ PASS และค่าที่ส่งไปด้วยคือตัวแปร Userid และ Pasword ซึ่งได้มาจากค่าเริ่มต้นที่ทำการเก็บบันทึกเอาไว้ เพื่อทำการติดต่อกับโอสท์ของ POP3 และตรวจสอบความถูกต้อง ซึ่งถ้าถูกต้องก็จะทำขั้นตอนต่อไปคือส่งคำสั่ง RETR พร้อมด้วยหมายเลขของข้อความที่ผู้ใช้งานต้องการอ่านออกไป หลังจาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นั้นที่หน้าต่างใหม่ที่สร้างขึ้นมาก็จะทำการแสดงข้อความที่มีทั้งหมดในเมล์นั้นออกมา ซึ่งในขั้นตอนนี้นั้นโปรแกรมจะทำการตรวจสอบข้อความและแสดงออกมาบนหน้าต่าง โดยที่ถ้าบรรทัดแรกของข้อความเป็น " ." โปรแกรมจะทำการตรวจสอบว่าตัวอักษรต่อไปเป็น " ." หรือไม่ ถ้าใช่ก็จะทำการลบ " . " นั้นทิ้งไปและทำการแสดงข้อความที่อยู่ตามมาออกมาเรื่อย ๆ จนกว่าจะตรวจพบตัวอักษร <CRLF>.<CRLF> ซึ่งเป็นสัญญลักษณ์แสดงถึงการที่จบข้อความทั้งหมดของเมล์แล้วก็จะหยุดการแสดงข้อความ หลังจากนั้นโปรแกรมจะทำการส่งคำสั่ง QUIT ออกไปเพื่อทำการหยุดการติดต่อกับโฮสท์ของ POP3

การแสดงข้อความในส่วนนี้ จะเป็นไปตามคุณสมบัติของโปรโตคอล POP3

ตัวเลือกนี้จะสามารถใช้งานได้ภายหลังจากการเลือกใช้ตัวเลือก Check Mail ในเมนู File และทำการเลือกเมล์ที่ต้องการแล้วเท่านั้นในกรณีอื่น ๆ ไม่สามารถใช้งานตัวเลือกนี้ได้

3.1.4 ตัวเลือก Delete ในเมนู Message

เป็นส่วนที่ใช้สำหรับทำการลบเมล์ที่ผู้ใช้งานไม่ต้องการที่จะเก็บไว้ในเมล์บ็อกซ์ของโฮสท์ของ POP3

หลักการทำงานของตัวเลือกนี้คือ จะทำการลบเมล์ที่ไม่ต้องการเก็บไว้ออก หลังจากนั้นก็จะทำการตรวจสอบเมล์ที่เหลืออยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 และแสดงเมล์ที่เหลือให้ผู้ใช้งานได้ทราบ

การทำงานของตัวเลือกนี้จะเริ่มจากทำการส่งคำสั่ง DELE พร้อมด้วยหมายเลขของเมล์ที่ต้องการจะลบออกไป หลังจากนั้นโปรแกรมก็จะทำการส่งคำสั่ง QUIT ออกไป เพื่อทำการยุติการติดต่อกับโฮสท์ของ POP3 สำหรับสาเหตุที่ต้องมีการส่งคำสั่ง QUIT ออกไปนั้นเนื่องมาจากว่าการทำงานของ โปรโตคอล POP3 นั้นเมื่อเราทำการลบเมล์ใด ๆ ก็ตาม โฮสท์ของ POP3 จะทำการเพียงมาร์ก (mark) เมล์นั้นเอาไว้ว่าจะทำการลบ แต่จะไม่ทำการลบในทันทีโฮสท์ของ POP3 จะทำการลบเมล์นั้นก็ต่อเมื่อสิ้นสุดการติดต่อกับโฮสท์ของ POP3 แล้วเท่านั้น ดังนั้นเราจึงจำเป็นต้องส่งคำสั่ง QUIT ออกไปด้วย หลังจากนั้นโปรแกรมก็จะการเรียกใช้ส่วนที่ทำการตรวจสอบเมล์ที่มีอยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 อีกครั้งโดยอัตโนมัติ และแสดงเมล์ที่มีเหลืออยู่ออกมาให้ผู้ใช้งานได้เห็น

ตัวเลือกนี้จะสามารถใช้งานได้ภายหลังจากการเลือกใช้ตัวเลือก Check Mail ในเมนู File และทำการเลือกเมล์ที่ต้องการแล้วเท่านั้นในกรณีอื่น ๆ ไม่สามารถใช้งานตัวเลือกนี้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.5 ตัวเลือก New ในเมนู Message

เป็นส่วนที่จะทำการสร้างหน้าต่างขึ้นมาใหม่ 1 หน้าต่างเพื่อใช้ในการส่งเมล์ โดยจะมีส่วนของเฮดเดอร์ของข้อความขึ้นมาให้โดยอัตโนมัติ

การทำงานในส่วนนี้นั้น เมื่อเราเลือกตัวเลือกนี้แล้วโปรแกรมจะทำการสร้างหน้าต่างขึ้นมาใหม่ 1 หน้าต่างพร้อมทั้งมีเฮดเดอร์มาให้ด้วย ซึ่งจะประกอบไปด้วย "To:" , "From:" , "Subject:" เพื่อให้ผู้ใช้งานได้พิมพ์รายละเอียดต่าง ๆ ลงไปภายหลัง ซึ่งในขั้นตอนนี้นั้นในส่วนของฟิลด์บอดี้ของเฮดเดอร์ "From:" นั้นโปรแกรมจะขึ้นอีเมล์แอดเดรสของผู้ใช้งานมาให้โดยอัตโนมัติ ซึ่งฟิลด์บอดี้ที่โปรแกรมขึ้นมาให้โดยอัตโนมัตินี้จะเป็นตัวแปรที่ได้มาจากค่าเริ่มต้นที่ผู้ใช้งานกรอกรายละเอียดเอาไว้ในตอนแรกเมื่อเริ่มต้นใช้โปรแกรม ซึ่งก็คือค่าตัวแปร Userid และ Pop3 นั่นเอง โดยในการที่ขึ้นเป็นอีเมล์แอดเดรสนั้นจะนำค่าตัวแปรทั้ง 2 มารวมกันในลักษณะต่อไปนี้

" ค่าตัวแปร Userid แล้วตามด้วยตัวอักษร "@" และตามด้วยค่าตัวแปร Pop3 "

สำหรับในส่วนของบอดี้ของข้อความของเมล์นั้นผู้ใช้งานจะต้องทำการเว้นบรรทัดว่าง 1 บรรทัดจากเฮดเดอร์ "Subject:" ตามรูปแบบของข้อความของเมล์โดยปกติ หลังจากที่ผู้ใช้งานทำการพิมพ์ข้อความที่ต้องการจะส่งเสร็จเรียบร้อยแล้ว และต้องการจะส่งเมล์นั้นออกไปผู้ใช้งานจะต้องทำการเลือกเมนู Message และเลือกไปที่ตัวเลือก Send ต่อไป

ตัวเลือกนี้จะสามารถใช้งานได้ หลังจากการเลือกใช้ตัวเลือก Set up ในเมนู File หรือทำการเปิดข้อมูลที่ทำการบันทึกค่าเริ่มต้นแล้วเท่านั้น ในกรณีอื่น ๆ ไม่สามารถเรียกใช้งานตัวเลือกนี้ได้

3.1.6 ตัวเลือก Reply ในเมนู Message

เป็นส่วนที่ใช้สำหรับทำการตอบเมล์ที่ผู้ใช้งานกำลังอ่านอยู่และต้องการตอบกลับไป

การออกแบบในส่วนนี้นั้นจะคล้ายกับการทำงานในส่วนของตัวเลือก New ในเมนู Message เพียงแต่มีความแตกต่างกันเล็กน้อย นั่นคือในส่วนของฟิลด์บอดี้ของเฮดเดอร์ "Subject:" นั้นในส่วนนี้โปรแกรมจะขึ้นให้มาโดยอัตโนมัติ โดยจะนำค่ามาจากฟิลด์บอดี้ของเฮดเดอร์ "Subject:" ในเมล์ที่เราอ่านแล้วต้องการจะตอบกลับไป และในส่วนของเฮดเดอร์ "To:" นั้นจะมีค่าของฟิลด์บอดี้เท่ากับค่าฟิลด์บอดี้ของเฮดเดอร์ "From:" ของเมล์ที่เราจะตอบ และส่วนของเฮดเดอร์ "From:" นั้นจะมีค่าของฟิลด์บอดี้เป็นอีเมล์แอดเดรสของผู้ใช้เอง โดยจะอยู่ในรูปแบบที่นำค่าเริ่มต้นมาต่อกันเหมือนกับค่าที่เป็นส่วนของเฮดเดอร์ "From:" ในตัวเลือก New ที่อยู่ในเมนู Message นั่นก็คือจะอยู่ในรูปแบบของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

" ค่าตัวแปร Userid แล้วตามด้วยตัวอักษร "@" และตามด้วยค่าตัวแปร Pop3 "

สำหรับรายละเอียดในการทำงานต่างนั้นจะเหมือนกับการทำงานของตัวเลือก New ในเมนู Message ตามที่ได้กล่าวไปแล้วข้างต้น

ตัวเลือกนี้จะสามารถเรียกใช้งานได้ภายหลังจากมีการเลือกใช้งานตัวเลือก Display ในเมนู Message แล้วเท่านั้น ในกรณีอื่น ๆ ไม่สามารถเรียกใช้งานตัวเลือกนี้ได้

3.1.7 ตัวเลือก Send ในเมนู Message

เป็นส่วนที่ใช้สำหรับทำการส่งเมล์ที่ผู้ใช้งานพิมพ์เสร็จแล้วและต้องการที่จะส่งเมล์นั้นออกไป

ลำดับขั้นตอนการทำงานของตัวเลือกนี้นั้นเมื่อผู้ใช้งานพิมพ์ข้อความของเมล์ที่ต้องการจะส่งเสร็จเรียบร้อยแล้ว ผู้ใช้งานจะต้องทำการเลือกเมนู Message และเลือกไปที่ตัวเลือก Send หลังจากนั้นโปรแกรมจะเริ่มทำงานโดยเริ่มต้นจากการส่งคำสั่ง HELO พร้อมด้วยค่าที่ต้องส่งออกไปด้วย ซึ่งนั่นก็คือค่าตัวแปร Smtp ที่ได้มาจากค่าเริ่มต้นที่เก็บไว้นั่นเอง เพื่อทำการติดต่อกับโฮสท์ของ SMTP ที่จะทำการส่งเมล์ออกไป หลังจากทำการติดต่อได้แล้วโปรแกรมก็จะทำการส่งคำสั่ง MAIL FROM พร้อมด้วยค่าที่ส่งไปด้วยคือค่าตัวแปรที่นำมาจากฟิลด์บอดี้ของ "From:" ที่ได้มาจากส่วนของ New หรือ Reply ที่พิมพ์เอาไว้เพื่อรอทำการส่งออกไป เพื่อแจ้งถึงรายละเอียดข้อมูลของผู้ส่ง หลังจากนั้นก็จะทำการส่งคำสั่ง RCPT TO พร้อมด้วยค่าที่ส่งไปด้วยคือค่าตัวแปรที่นำมาจากฟิลด์บอดี้ของ "Subject:" ที่ได้มาจากส่วนของ New หรือ Reply ที่พิมพ์เอาไว้เพื่อรอทำการส่งออกไป เพื่อแจ้งถึงรายละเอียดข้อมูลของผู้รับ ซึ่งในขั้นตอนนี้จะมีการตรวจสอบความถูกต้องของทั้งผู้รับและผู้ส่งด้วยว่าถูกต้องหรือไม่ ถ้าถูกต้องโปรแกรมก็จะทำการส่งคำสั่งต่อไปออกไปซึ่งได้แก่คำสั่ง DATA ซึ่งจะทำการส่งข้อความในส่วนของบอดี้ออกไปทีละบรรทัดไปยังโฮสท์ของ SMTP ถ้าบรรทัดแรกของข้อความเริ่มต้นด้วย " . " โปรแกรมก็จะทำการเพิ่มตัวอักษร " . " เข้าไปอีกหนึ่งตัว หลังจากนั้นก็ทำการส่งข้อความที่เหลือต่อไปจนหมด หลังจากนั้นโปรแกรมจะทำการส่งตัวอักษร " . " เพิ่มเข้าไปอีกหนึ่งตัว เพื่อแสดงให้โฮสท์ของ SMTP ทราบว่าข้อความที่ทำการส่งนั้นสิ้นสุดแล้ว หลังจากนั้นโปรแกรมก็จะทำการส่งคำสั่ง QUIT ออกไปเพื่อยกเลิกการติดต่อกับโฮสท์ของ SMTP

ตัวเลือกนี้จะสามารถเรียกใช้งานได้ภายหลังจากมีการเลือกใช้งานตัวเลือก Display ในเมนู Message แล้วเท่านั้น ในกรณีอื่น ๆ ไม่สามารถเรียกใช้งานตัวเลือกนี้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 การส่งเมล์ในรูปแบบของเสียง

ถ้าเราต้องการจะส่งเมล์ไปยังผู้รับในรูปแบบที่เป็นเสียงนั้นจะต้องเริ่มด้วยการทำการบันทึกเสียงที่ต้องการจะส่งเก็บไว้ในรูปแบบของ WAV ไฟล์ หลังจากนั้นทำการแก้ไขปรับปรุงการทำงานของโปรแกรมโดยจะมีรายละเอียดดังต่อไปนี้

ในขั้นแรกจะต้องมีการเพิ่มเฮดเดอร์เข้าไปอีกหนึ่งตัวในทุก ๆ ครั้งที่มีการเรียกใช้ตัวเลือก New และ Reply ในเมนู Message คือ "ATTACHMENT:" และฟิลด์บอดี้ของเฮดเดอร์นี้ก็จะเป็นชื่อไฟล์เสียงที่ผู้ใช้งานต้องการจะส่งออกไปให้ยังผู้รับปลายทาง และเมื่อมีการส่งเมล์นี้ออกไปแล้ว โปรแกรมจะต้องมีการตรวจสอบฟิลด์บอดี้ของเฮดเดอร์ "ATTACHMENT:" ว่ามีค่าที่พิมพ์ใส่ลงไปเพื่อแสดงว่าต้องการส่งไฟล์เสียงออกไปพร้อมกับเมล์นั้นด้วยหรือไม่ ถ้ามีโปรแกรมจะทำการตรวจสอบไฟล์ที่ต้องการจะส่งนั้นว่ามีอยู่จริงหรือไม่ ถ้ามีอยู่จริงโปรแกรมก็จะทำการแปลงไฟล์นั้นให้อยู่ในรูปแบบที่สามารถส่งออกไปผ่านระบบอินเตอร์เน็ตได้ ซึ่งในที่นี้เราจะใช้รูปแบบของ "Base-64" ซึ่งจะทำการแปลงไฟล์เสียงนั้นให้อยู่ในรูปแบบของตัวอักษร และส่งเพิ่มเติมต่อท้ายจากข้อความธรรมดาที่ต้องการส่งไปด้วยซึ่งข้อความธรรมดานั้นจะมีหรือไม่มีก็ได้ ซึ่งโปรแกรมจะทำการแยกทั้งสองส่วนคือ ข้อความธรรมดาและข้อความที่ได้มาจากการแปลงไฟล์เสียงมา และจะมีส่วนที่ระบุว่าส่วนไหนเป็นข้อความธรรมดา ส่วนไหนเป็นไฟล์เสียงที่แปลงมา แล้วจึงส่งเมล์นั้นออกไปยังโฮสท์ของ SMTP

เมื่อทางผู้รับได้รับเมล์นี้แล้วและต้องการที่จะอ่านเมล์นั้น โปรแกรมก็จะทำการตรวจสอบข้อความที่จะแสดงขึ้นมาให้ผู้ใช้งานได้อ่านนั้นว่าเป็นเพียงข้อความธรรมดาเพียงอย่างเดียว หรือว่ามีส่วนที่เป็นไฟล์เสียงที่ถูกแปลงมาด้วยหรือไม่ ถ้าพบว่ามีเพียงข้อความธรรมดาอย่างเดียวก็จะแสดงข้อความนั้นออกมาให้ผู้ใช้งานได้อ่านเลยทันที แต่ถ้าตรวจสอบแล้วพบว่ามีส่วนที่เป็นไฟล์เสียงแปลงมาด้วย ก็จะแสดงข้อความธรรมดาออกมาให้ผู้ใช้งานได้อ่านก่อนและจะมีการแจ้งให้ผู้ใช้งานได้ทราบว่ามีไฟล์เสียงส่งมาด้วย โดยจะเพิ่มบรรทัดข้อความขึ้นมาอีกหนึ่งบรรทัดเป็นข้อความที่จะแสดงให้ผู้ใช้งานได้รู้ และทำการแปลงส่วนที่เป็นไฟล์เสียงที่ถูกแปลงมาให้กลับมาอยู่ในรูปแบบของไฟล์เสียงตามเดิม และเก็บไว้ในไดเรคทอรี่ (Directory) ที่มีโปรแกรมนี้อยู่ ผู้ใช้ก็สามารถเรียกไฟล์เสียงนั้นมาฟังได้ในโอกาสต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดลองและผลการทดลอง

4.1 การสร้างเมล์ไคลเอนท์

เราจะทำการสร้างเมล์ไคลเอนท์โดยการใช้ AppWizard ของโปรแกรม Visual C++ เพื่อลร้างส่วนติดต่อกับผู้ใช้งาน ส่วนสำคัญสำหรับการสร้างนั้นมีอยู่สองส่วนได้แก่:แอพพลิเคชั่น (Aplication) ที่จะทำการสร้างนี้ต้องเป็น MDI แอพพลิเคชั่นและต้องมีแถบเครื่องมือ (Toolbar)

เริ่มต้นการสร้างแอพพลิเคชั่นดังนี้

1. เลือก File , New Project , OK จากโปรแกรม Visual C++ และตั้งชื่อชิ้นงานว่า "mail"

2. คลิ้ก Cretate และเลือก Multiple Document Interface , Next , None สำหรับฐานข้อมูล ; Next , None สำหรับ OLE , Next

3. เช็ค Status Bar และ Toolbar , ไม่เช็ค Ptrint Support , เช็ค Sockets Support

4. คลิ้ก Next , static library , Finish , OK

AppWizard จะทำการสร้างส่วนต่าง ๆ ของแอพพลิเคชั่นขึ้นมาเพื่อรอการเพิ่มเติม แก้ไขปรับปรุงต่อไปในภายหลัง

4.2 AppStudio

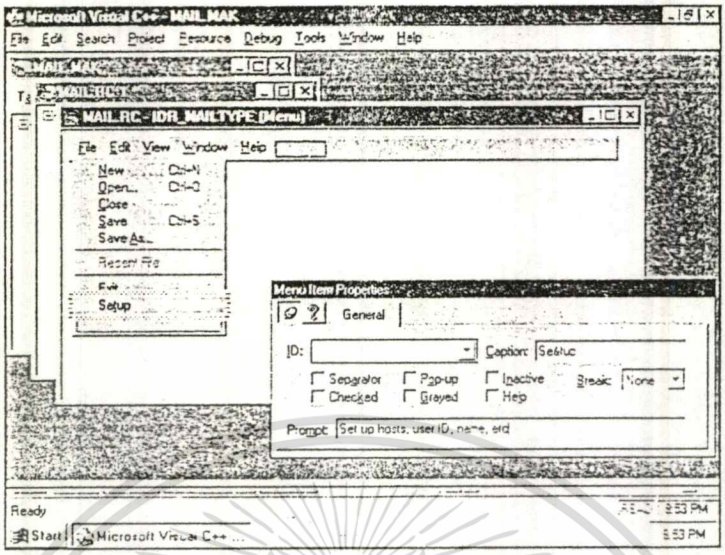
ต่อไปจะเป็นการสร้างและแก้ไขเมนู ซึ่งเราสามารถทำได้โดยการดับเบิ้ลคลิ้ก MAIL.RC ในโปรเจคลิสท์ (Project list) ต่อไปคลิ้กไปที่ Menu เราจะเห็นว่ามีเมนูอยู่สองเมนูเรียบร้อยแล้วคือ IDR_MAINFRAME และ IDR_MAILTYPE

IDR_MAINFRAME เป็นเมนูที่ถูกใช้งานเมื่อไม่มีการเปิดไฟล์ใด ๆ ขึ้นมา เราจะไม่มีการแก้ไข เพิ่มเติมเมนูในส่วนนี้ เนื่องจากว่าเราไม่จำเป็นต้องทำการใด ๆ ในขณะที่ไม่มีไฟล์ใด ๆ ถูกเปิดขึ้นมา

IDR_MAILTYPE เป็นเมนูที่ถูกใช้งานเมื่อไฟล์ใดไฟล์หนึ่งถูกเปิดขึ้นมา ซึ่งเราจะทำการแก้ไขเพิ่มเติมเมนู และตัวเลือกต่าง ๆ ในส่วนนี้

เริ่มโดยการดับเบิ้ลคลิ้ก IDR_MAILTYPE หลังจากนั้นคลิ้กไปที่ File จะเห็นส่วนที่ดร็อป (drop) ลงมา ซึ่งจะมีตัวเลือกว่าง (Blank Item) อยู่ที่ส่วนล่างสุด คลิ้กส่วนนั้นเพื่อทำการเลือก ใช้โพรเพอร์ตี้บ็อกซ์ (Properties box) เพื่อเปลี่ยนชื่อเรียก (Caption) เป็น Se&tup และที่พรอมท์ (Prompt) เป็น "Set up hosts,userid,name,etc." ซึ่งแสดงให้เห็นดังรูปที่ 4.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

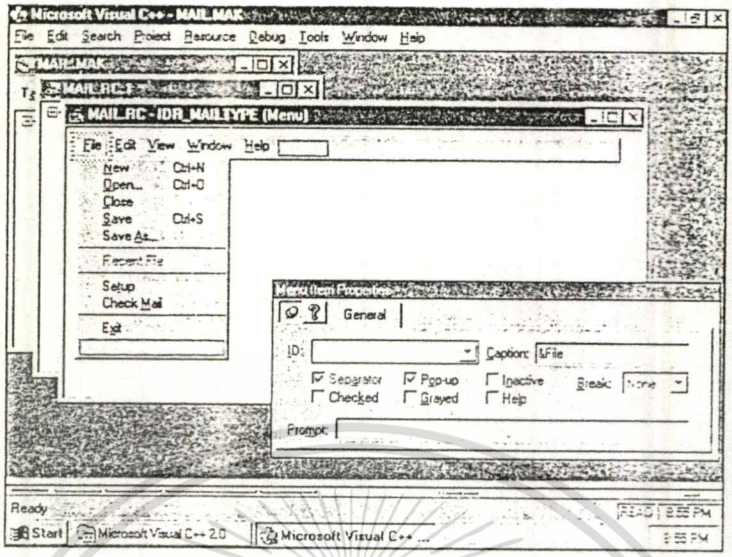


รูปที่ 4.1 การเพิ่มตัวเลือก Setup ในเมนู File

หลังจากนั้นคลิ้กที่ตัวเลือก Setup และย้ายไปอยู่เหนือตัวเลือก Exit

เมื่อเราเริ่มทำการแก้ไข เปลี่ยนแปลงตัวเลือกว่าง ก็จะมีการสร้างตัวเลือกว่างขึ้นมาใหม่อีกหนึ่งตัวเลือก คลิ้กที่ตัวเลือกว่างนั้น ทำการเปลี่ยนแปลงแก้ไขค่าต่าง ๆ ซึ่งได้แก่ ชื่อเรียกเป็น Check Mail และพรอมท์เป็น "Contact the POP3 host to get new mail" ย้ายตัวเลือกนี้ขึ้นมาอยู่ระหว่าง Setup และ Exit หลังจากนั้นย้ายตัวเลือกว่างใหม่ที่เกิดขึ้นมาให้อยู่ระหว่าง Check Mail และ Exit และคลิ้กที่ Separate ในโพรเพอร์ตี้บ็อกซ์ เพื่อการสร้างเส้นคั่นระหว่างตัวเลือก เมื่อทำการเปลี่ยนแปลงแก้ไขจนเสร็จแล้วจะเป็นดังรูปที่ 4.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



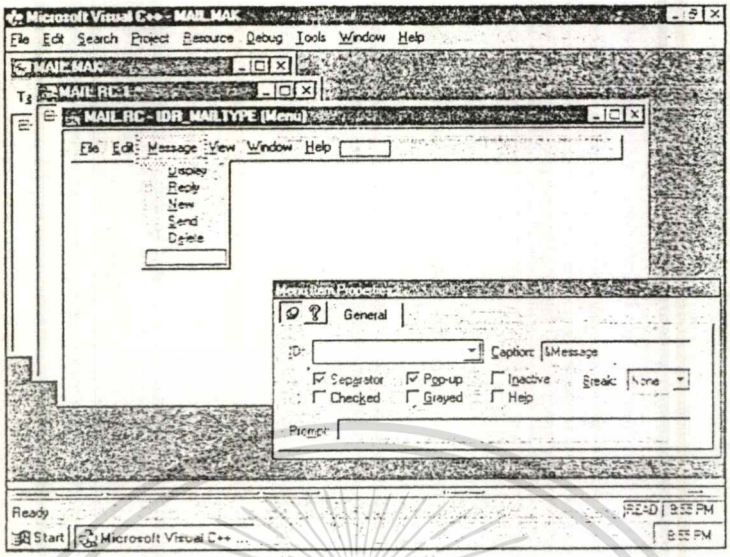
รูปที่ 4.2 เมนู File ที่ประกอบไปด้วย 2 ตัวเลือกใหม่ และเส้นคั่น

หลังจากสิ้นสุดการเปลี่ยนแปลงเมนู File แล้วต่อไปเราต้องทำการเพิ่มเมนูใหม่ขึ้นมาอีกหนึ่งเมนู คือเมนู Message ซึ่งเราจะสังเกตเห็นได้ว่าจะมีช่องว่างอยู่ทางขวาของเมนู Help คลิ้กที่นั่นและเปลี่ยนชื่อเรียกเป็น &Message ซึ่งจะปรากฏตัวเลือกว่างอยู่ข้างใต้ ทำการเพิ่มเติมตัวเลือกอีกห้าตัวเลือก ซึ่งมีรายละเอียดตามตาราง

ชื่อเรียก	พรอมท์
&Display	Display tthe highlighted message
&Reply	Compose a reply to this message
&New	Compose a new mail message
&Send	Send the complete message to the SMTP host
D&elete	Delette the highlighted message

ซึ่งหลังจากทำการแก้ไขเรียบร้อยแล้ว จะปรากฏดังรูปที่ 4.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 เมนู Message และส่วนประกอบที่สร้างขึ้นใหม่

4.3 การเพิ่มเติมส่วนที่มองเห็น (Adding New View)

เฟรมเวิร์ค (frame work) จะเก็บแทรค (track) ของส่วนที่มองเห็นเอาไว้ในสตริงเทเบิ้ล (string table) สำหรับแต่ละแอพพลิเคชั่น ในการสร้างส่วนที่มองเห็นขึ้นมาใหม่สิ่งแรกที่ต้องทำคือเพิ่มเอนทรี่ (entry) ลงไปในสตริงเทเบิ้ล เลือก AppStudio และเลือก String table จะมีส่วนเอนทรี่สำหรับ IDR_MAINFRAME และสำหรับ IDR_MAILTYTPE ซึ่งเป็นส่วนที่ AppWizard กำหนดไว้เรียบร้อยแล้ว

เอนทรี่ในสตริงจะบอกเฟรมเวิร์คว่าดอคคิวเมนท์ชนิดไหนที่จะทำงานสัมพันธ์กับส่วนที่มองเห็น , พรอมท์อะไรที่ใช้ในการบรรยายลักษณะของดอคคิวเมนท์ และอื่น ๆ เราสามารถทำการคัดลอกเอนทรี่จำนวนมากสำหรับ MAILTYPE ซึ่งต้องแน่ใจว่าโพรเพอร์ตี้บ็อกซ์ปรากฏขึ้นบนจอภาพ หลังจากนั้นคลิ้ก IDR_MAILTYPE เลือก File , Copy เพื่อใส่เอนทรี่ลงไปในคลิปบอร์ด Clipboard เลือก File , Paste และเปลี่ยนชื่อจาก IDR_MAILTYPE2 เป็น IDR_MESSAGE เลือก Paste อีกครั้งและเปลี่ยนชื่อเป็น IDR_COMPOSE ขณะนี้เราจะมี 3 สตริงเทเบิ้ลเอนทรี่ซึ่งทำงานตามชื่อของมัน

เมื่อผู้ใช้งานเลือก File ,New จากแอพพลิเคชั่น ระบบจะสร้างไดอะล็อกบ็อกซ์เพื่อให้ผู้ใช้งานได้เลือกชนิดของไฟล์ที่ต้องการ ซึ่งอาจจะมีส่วนที่มองเห็นทั้งหมดที่เป็นชนิดเดียวกันขึ้นมาให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือก เราจะต้องทำการป้องกันในส่วนนี้ กดไปที่หนึ่งในสามของสตริงและดูในโพรเพอร์ตี้บ็อกซ์ สังเกตุว่ามันจะแจ้งว่า "Mail Document" ครั้งแรกจะเป็นพรอมท์สำหรับ File , New และถ้าเราย้ายมันออก ดังนั้นสตริงจะเริ่มด้วย \nmail\n\n ดังนั้นไดอะล็อกบ็อกซ์นี้จะไม่ปรากฏขึ้นในส่วนของ File , New ดังนั้นเราจะเอาส่วน "Mail Document" ออกจาก IDR_MESSAGE และ IDR_COMPOSE แต่เหลือทิ้งไว้ใน IDR_MAILTYPE

ปิด MAIL.RC และเปิด MAIL.CPP ซึ่งจะบรรจุออปเจค (object) ของแอพพลิเคชั่นเอาไว้ในชื่อ CMailApp ในฟังก์ชั่น InitInstance() เราสามารถค้นหาโค้ดส่วนต่อไปนี้

```
// Register the application's document template.
// Document template
// Serve as the connection between document,
// frame windows and views.

CMultiDocTemplate* pDocTemplate;
pDocTemplate = new CMultiDocTemplate(
        IDR_MAILTYPE,
        RUNTIME_CLASS(CMailDoc),
        RUNTIME_CLASS(CMDIChildWnd),
        // Standard MDI child frame
        RUNTIME_CLASS(CMailView));
AddDocTemplate(pDocTemplate);
```

ซึ่งจะเป็นการประกาศเทมป์เพลท (template) ซึ่งรวมทรัพยากรสตริง IDR_MAITYPE และสาม C++ ออปเจคคลาส (object class) ซึ่งได้แก่ CMailDoc , CMDIChildWnd , CMailView

เราต้องการอีกสองเทมป์เพลท หนึ่งเทมป์เพลทสำหรับแต่ละส่วนที่มองเห็นที่เรากำลังจะเพิ่มขึ้น เราต้องมีเทมป์เพลททั้งสามเพื่อเป็นเมมเบอร์ (member) ของคลาส CMailApp ดังนั้นต้องทำการแก้ไขเพิ่มเติมใน MAIL.H ดังนี้

```
CMultiDocTemplate* pDocTemplate;
CMultiDocTemplate* pMessageTemplate;
CMultiDocTemplate* pComposeTemplate;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับตอนนี้นั้นเราจะตั้งชื่อสำหรับเฟรมและออปเจคของส่วนที่มองเห็น ซึ่งทำงานสัมพันธ์กับแต่ละส่วนที่มองเห็น IDR_MESSAGE จะยังคงสัมพันธ์กับ CMailDoc และ แต่ส่วนที่มองเห็นจะเป็น CMessageView , IDR_COMPOSE จะมีส่วนที่มองเห็นเป็น CComposeView , เปลี่ยนชื่อของเทมป์เพลทในบ็อกซ์ใหม่ของโค้ด pMessageTemplate และ pComposeTemplate ดังนี้

```
// Register the application's document templates.
// Document templates serve as the connect
// between documents,frame windows and views.

CMultiDocTemplate* pDocTemplate;
pDocTemplate = new CMultiDocTemplate(
        IDR_MAILTYPE,
        RUNTIME_CLASS(CMailDoc),
        RUNTIME_CLASS(CMDIChildWnd),
        // Standard MDI child frame
        RUNTIME_CLASS(CMailView));
AddDocTemplate(pDocTemplate);

CMultiDocTemplate* pDocTemplate;
pDocTemplate = new CMultiDocTemplate(
        IDR_MAILTYPE,
        RUNTIME_CLASS(CMailDoc),
        RUNTIME_CLASS(CMDIChildWnd),
        // Standard MDI child frame
        RUNTIME_CLASS(CMessageView));
AddDocTemplate(pMessageTemplate);

CMultiDocTemplate* pDocTemplate;
pDocTemplate = new CMultiDocTemplate(
        IDR_MAILTYPE,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        RUNTIME_CLASS(CMailDoc),
        RUNTIME_CLASS(CMDIChildWnd),
        // Standard MDI child frame
        RUNTIME_CLASS(CComposeView));
    AddDocTemplate(pComposeTemplate);
```

ใช้ ClassWizard ในการสร้างคลาสใหม่และเพิ่มเข้าไปในโปรเจค,เลือก ClassWizard คลิ้กที่ Add Class และเติมชื่อคลาสและชื่อไฟล์ เราสามารถใช้ MESSAGE.CPP และ MESSAGE.H สำหรับ CMessageView , COMPOSE.CPP และ COMPOSE.H สำหรับ CComposeView

หลังจากตั้งค่าต่าง ๆ ถูกต้องแล้ว ขณะนี้เมนู IDR_MAINFRAME จะถูกแสดงเมื่อไม่มีไฟล์ใด ๆ ถูกเปิดขึ้น เมนู IDR_MAILTYPE จะถูกแสดงเมื่อมีไฟล์ถูกเปิด และ CMailView จะถูกเลือก,เมนู IDR_MESSAGE เมื่อ CMessageView ถูกเลือก , เมนู IDR_COMPOSE เมื่อ CComposeView ถูกเลือก

แต่ละเทมป์เพลทจะรวบรวมข้อมูลต่าง ๆ.เกี่ยวกับดอคคิวเมนท์ , ส่วนที่มองเห็น , เฟรม และเมนู ชื่อของเมนูจะเป็นไปตามชื่อของทรัพยากรของสตริง บรรทัดต่าง ๆ ต่อไปนี้ของโค้ดจะจัดลำดับสำหรับทั้งสามส่วนที่มองเห็นเมื่อใช้เมนูและเทเบิ้ลเดียวกัน

```
pMessageTemplate -> m_hMenuShared = PDocTemplate -> m_hMenuShared;
pMessageTemplate -> m_hAccelTable = PDocTemplate -> m_AccelTable;
pComposeTemplate -> m_hMenuShared = PDocTemplate -> m_hMenuShared;
pComposeTemplate -> m_hAccelTable = PDocTemplate -> m_AccelTable;
```

4.4 คลาสวิสาสด์ (ClassWizard)

เมื่อ ผู้ใช้งานเลือกตัวเลือกในเมนู หน้าต่างข้อความจะถูกสร้างขึ้น หนึ่งในออปเจคสามารถจับ (catch) ข้อความและเรียกฟังก์ชั่นของตัวเองขึ้นมาเป็นผลลัพธ์ เมนูที่เราสร้างขึ้นมาใหม่นั้นประกอบไปด้วยตัวเลือก Setup และ Check Mail ในเมนู File และ Display , Reply , New , Send และ Delete ในเมนู Message เราต้องทำการตัดสินใจว่าออปเจคไหนจะต้องทำงานตามลำดับของคำสั่งและแจ้งผลลัพธ์อะไรให้ผู้ใช้ได้ทราบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

File , Setup เปลี่ยนข้อมูลของผู้ใช้งานซึ่งจะเก็บไว้ในดอคคิวเมนท์ ดังนั้นดอคคิวเมนท์ออปเจค CMailDoc ควรจะทำงานนี้ File , Check Mail ต้องการข้อมูลของผู้ใช้งานจากดอคคิวเมนท์ แต่มันจะเปลี่ยนส่วนประกอบของรายการใน CMailView ซึ่งก็คือรายการของเมล์ในเมล์บ็อกซ์ CMailView ทำงานกับส่วนนี้และถามดอคคิวเมนท์สำหรับข้อมูลที่ต้องการ

Message , Display จะมีประโยชน์ก็ต่อเมื่อ CMailView ถูกเลือก และข้อความที่ต้องการถูกเลือก ต้องเป็นที่รู้กันว่าจะไม่มีการเลือกข้อความนี้ขณะที่ทำการอ่านข้อความนั้นอยู่ (CMessageView ถูกเลือกใช้งาน) หรือขณะที่กำลังเขียนข้อความใหม่ (CComposeView ถูกเลือกใช้งาน) เพราะฉะนั้นตัวเลือก Display ควรจะถูกจับโดยออปเจค CMailView

สำหรับตัวเลือก Reply นั้นจะมีผลก็ต่อเมื่อ CMessageView ถูกเลือกเท่านั้น , New สามารถทำงานได้ทั้ง CMailView และ CMesageView แต่เพื่อความง่ายของโปรแกรม เราจะกำหนดให้ CMailView มีผลเพียงอย่างเดียว , Send จะเป็นของ CComposeView และ Delete จะมีผลกับ CMailView ซึ่งเป็นการแสดงรายการของข้อความ

เมื่อออกแบบได้แล้วก็ใช้ ClassWizard ในการสร้างการติดต่อ โดยเริ่มจากการเรียก ClassWizard และเลือก CMailDoc จากดร็อป-ดาวน์ลิสท์บ็อกซ์ (drop-down list box) เลือก ID_FILE_SETUP จาก Object IDs box ต่อไปเลือก COMMAND ใน Mmessage box คลิ้กปุ่ม Add Function และคลิ้กปุ่ม OK

จากนั้นเปลี่ยนตัวเลือกในดร็อป-ดาวน์ลิสท์บ็อกซ์เป็น CMailView และเลือก ID_FILE_CHECKMAIL , ID_MESSAGE_DISPLAY , ID_MESSAGE_NEW เปลี่ยนตัวเลือกเป็น CMessageView และเพิ่มฟังก์ชั่น ID_MESSAGE_REPLY และ ID_MESSAGE_DELETE และสุดท้ายเลือก CComposeView และ ID_MESSAGE_SEND จากนั้นทำการคอมไพล์ (compile) แอพพลิเคชั่น

เราต้องทำให้ CMailView ประกอบไปด้วยรายการของข้อความ ซึ่งทำได้โดยการเพิ่มโค้ดเข้าไปใน MAILVIEW.H เพื่อบอกว่าส่วนที่มองเห็นนี้มีบ็อกซ์ของรายการที่เรียกว่า MessageList :

```
CListBox MessageList;
```

ซึ่งบ็อกซ์นี้จะว่างเปล่าจนกว่าผู้ใช้งานจะเลือก Check Mail เพื่อเติมรายละเอียดเข้าไป แต่เราจำเป็นต้องสร้างบ็อกซ์เมื่อเราสร้างส่วนที่มองเห็น โดยการแก้ไขที่ OnInitailUpdate และใช้วิธี Create ซึ่งจะทำให้บล็อกรายการปรากฏขึ้นบนจอหน้าต่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

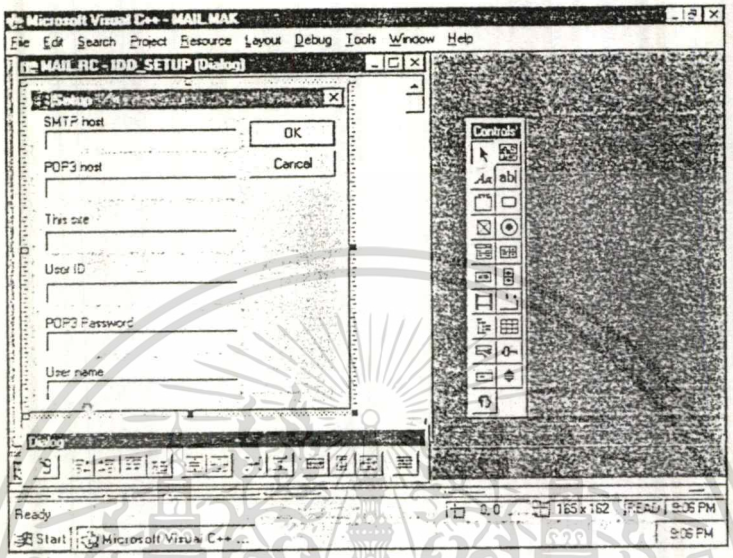
```
void CMailView::OnInitialUpdate()
{
        Crect rectangle;
        GetClientRect(rectangle);
        MessageList.Create(LBS_NOTIFY | LBS_USETABSTOPS |
                WS_CHILD | WS_VISIBLE | WS_VSCROLL |
                WS_HSCROLL,
                rectangle , this , IDC_MESSAGELISTBOX);
}
```

4.5 โค้ดสำหรับตัวเลือก Setup ในเมนู File

(Code for the File,Setup Menu Item)

ในขั้นแรกเราจำเป็นต้องมี 6 อีดิดบ็อกซ์ (edit box) ใน AppStudio สร้างไดอะล็อกบ็อกซ์และตั้งชื่อว่า IDD_SETUP ซึ่งประกอบไปด้วย 6 อีดิดบ็อกซ์ ดังแสดงในรูปที่ 4.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.4 การสร้าง Setup ไดอะล็อก

และเราตั้งชื่อแต่ละบ็อกซ์ดังนี้

- IDC_SMTP
- IDC_POP3
- IDC_SITE
- IDCUSERID
- IDC_PASSWORD
- IDC_USERNAME

ต่อไปใช้ ClassWizard เพื่อสร้างคลาสสำหรับไดอะล็อกนี้ และติดต่อการควบคุมไปยังค่าตัวแปร Cstrting 6 ตัว ตั้งชื่อคลาส CSTettupDialog และตัวแปร m_smtp , m_pop3 , m_site , m_userid , m_password และ m_username และรอทำการเพิ่มเติมต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชั่น CMailDoc::OnFileSetup() จะถูกเรียกใช้เมื่อผู้ใช้งานทำการเลือก File , Setup , ดังนั้นเราต้องเพิ่มเติมโค้ดเพื่อใช้เรียกไดอะล็อกนี้ขึ้นมาและใช้งานมัน ในการใช้งานนั้นจำเป็นต้องมีค่าตัวแปรในดอคคิวเมนท์ออปเจค ซึ่งจะเก็บค่าที่ผู้ใช้งานตั้งเก็บไว้ในรูปแบบของ Cstring เป็นค่า SMTP , POP3 , Site ,Userid , Password และ Usernameโดยทำการเพิ่มโค้ดใน MAILDOC.H ดังนี้

```
void CMailDoc::OnFileSetup()
{
        CSetupDialog dialog;
        dialog.m_smtp = SMTP;
        dialog.m_pop3 = POP3;
        dialog.m_site = Site;
        dialog.m_userid = Userid;
        dialog.m_password = Password:
        dialog.m_username = Username;

        if (dialog.DoModal() == IDOK)
        {
            SMTP = dialog.m_smtp;
            POP3 = dialog.m_pop3 ;
            Site = dialog.m_site;
            Userid = dialog.m_userid;
            Password = dialog.m_password;
            Username = dialog.m_username;
        }
}
```

โค้ดส่วนนี้จะสร้างไดอะล็อกเพื่อให้ผู้ใช้งานทำการแก้ไขเพิ่มเติมข้อมูล ซึ่งส่วนนี้เราสามารถเก็บบันทึกไว้ใช้งานในภายหน้าได้ โดยการเขียนโค้ดดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CMailDoc::Serialize(Cartchieve& ar)
{
        if (ar.IsStoring())
        {
                ar << SMTP;
                ar << POP3;
                ar << Site;
                ar << Userid;
                ar << Password;
                ar << Username;
        }
        else
        {
                ar >> SMTP;
                ar >> POP3;
                ar >> Site;
                ar >> Userid;
                ar >> Password;
                ar >> Username;
        }
}
```

4.6 โค้ดสำหรับตัวเลือก Setup ในเมนู File

(Code for the File,Check Mail Menu Item)

เราจำเป็นต้องเขียน CMailView::OnFileCheckmail() เราต้องทำการติดต่อกับโฮสท์ของ POP3 และถามถึงจำนวนของข้อความว่ามีอยู่เท่าไหร่ จากนั้นแสดงข้อความและรายละเอียดให้ผู้ใช้งานได้เห็น และทำการหยุดการติดต่อ ซึ่งเราสามารถทำการเปิดใหม่อีกกี่ครั้งก็ได้ตามต้องการ ดังนั้น CMailView::OnFileCheckmail() จะเป็นดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CMailView::OnFileCheckmail()
{
        delete pSocket;
        pSocket = new Qsocket();
        pSocket -> SetRecieveTargett(this , WM_SOCKET_RESPONSE);
        // Blank list box
        MetssageList.ResetContent();
        State = CheckMail;
        if (!pSocket -> Connect(GetDocument () - GetPop3 () , 110 ))
        {
                AfxMessageBox("Host is unreachable.");
                delete pSocket;
                pSocket = 0;
                return;
        }
}
```

ใน MAIL.H เพิ่มเติมโค้ด

```
enum CommandState{Idle,CheckMail};
```

ใน MAILVIEW.CPP เพิ่ม

```
State = Idle;
```

ต่อไปทำการเปลี่ยนแปลงแก้ไขในส่วนของ OnSocket() **ดังนี้**

```
LRESULT CMailView::OnSocket(WPARAM amount , LPARAM buffer)
{
        if ((int)amount > 0)
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
else
{
        // if amount < 0 it is a receive command
        Cstring response;
        CMailDoc* doc = GetDocument();
        switch((SocketRecieveCmd)amount)
        {
                case SocketStatusChanged:
                        switch (pSocket -> GetStatus())
                        {
                                case CONNECT :
                                // switch to getline
                                pSocketт -> SetRecieveTarget(NULL,0);
                                response = pSocket -t. GetLine();
                                pSocket -> Send("USER "
                                 + doc -> GetUserid() ++ "\r\n;
                                response = pSocket -> GetLine();
                                pSocket -> Send("PASS "
                                 + doc -> GetPassword() + "\r\n");
                                response = pSocket -> GetlLne();

                                if (Statte == CheckMail)
                                {
                                        FillListBox();
                                        State = Idle;
                                }
                                break;
                        case DISCONNECTED:
                                break;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    default:
                        break;
                }
                break;
            }
        }
        return 0;
}
```

ในส่วนที่รับรายการของข้อความและใส่มันลงไปในลิสท์บ็อกซ์ คือ FillListBox() FillListBox() จำเป็นต้องรู้ว่ามีข้อความอยู่เท่าไหร่ ซึ่งจะทำได้โดยการส่งคำสั่ง STAT

```
Cstring response;
pSocket -> Send("STAT \r\n");
response = response.Right(response.GetLength() - 4);
// Strip away '+OK'
int space = response.Find(' ')  ;
response = response.Left(space);
int msgs = atoi(response);
```

ต่อไปจะเป็นส่วนที่แสดงรายละเอียดต่าง ๆ ของข้อความที่จะแสดงให้ผู้ใช้งานได้เห็น

```
Cstring subject , from ,line;
for (int i=1;i<=msgs;I++)
{
        char command[20];
        // Unlikely to have a 13 digit number of message
        sprintf(command,"RETR %i \r\n",I);
        pSocket -> Send(command);
        response = pSocket -> Getline();
        int length;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if (response.Left(3) == "+OK")
{
        // header follow
        subject = " ";
        from = " ";
        while ((response = pSocket -> GetlLne()) != ".\r\n"
                && pSocket -> GetLine() == CONNECTED)
        {
                length = response.GetLength();
                if (response.Left(9) == "Subject: ")
                {
                        subject = response.Right(length - 9);
                        length = subject.GetLength();
                        if (length >= 2
                         && subject[length-2] == '\r'
                         && subject[length-1] == '\n')
                        {
                                subject = subject.Left(length-2);
                        }

                }
                if (response.Left(6) == "From: ")
                {
                        from = response.Right(length - 6);
                        length = from.GetLength();
                        if (length >= 2
                          && from[length-2] == '\r'
                          && from[length-1] == '\n')
                        {
                                from = from.Left(length-2);
                        }
```

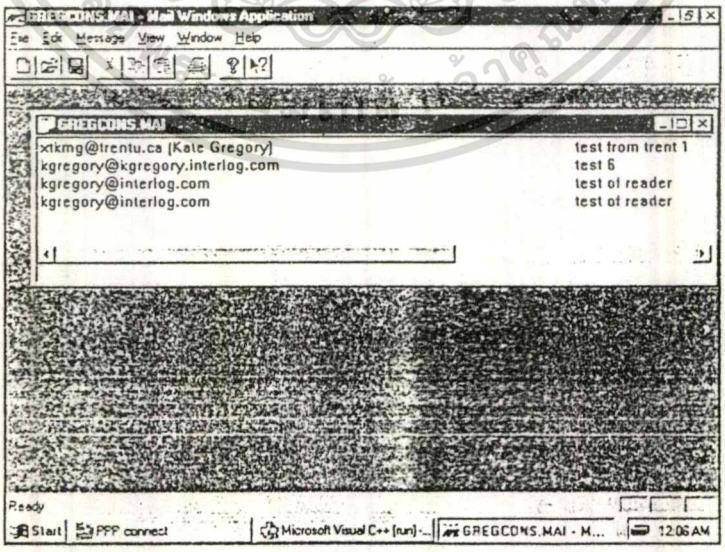
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    }
                }
                line = from.Left(tabwidth);
                // So as not to overshoot tabstop.
                line += '\t';  // tab
                line += subject;
                MessageList.AddString(line);
            } // if response = +OK
        } //for loop going through message
```

เราใช้ฟังก์ชั่น GetLine() ในลูป (loop) เพื่อทำการเก็บบันทึกในส่วนของ “Subject: “ หรือ “From: “เพื่อนำมาแสดงเป็นรายละเอียดให้ผู้ใช้งานได้รู้ว่าส่งมาจากใครและเรื่องอะไร จากนั้นทำการยกเลิกการติดต่อ

```
        pSocket -> Send("QUIT \r\n");
        response = pSocket -> GetLine();
        pSocket = NULL;
}
```

เมื่อเสร็จสมบูรณ์แล้วและทำการเลือกตัวเลือก Check Mail จะแสดงได้ดังรูปที่ 4.5



รูปที่ 4.5 แสดงผลเมื่อทำการเลือกตัวเลือก Check Mail

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.7 โค้ดสำหรับตัวเลือก Display ในเมนู Message

(Code for the Message,Display Menu Item)

เราต้องเขียนฟังก์ชั่นเพื่อใช้ในการกำหนดข้อความที่จะถูกแสดง , สร้างส่วนที่มองเห็นใหม่ และส่งค่าของตัวเลขของข้อความนั้นไปยังส่วนที่มองเห็นใหม่ที่สร้างขึ้น ดังนี้

```
void CMailView::OnMessageDisplay()
{
        int index = MessageList.GetCurSel();
        if (index < 0)
        {
                // User hasn't selected a message - - use the first one.
                Index = 0;
        }
        CMailApp* App = (CMailApp*)AfxGetApp();
        CMDIChildWnd* pNewFrame = (CMDIChildWnd*)App.->
                pMessageTemplate -> CreateNewFrame(m_pDocument,NULL);
        if (pNewFrame == NULL)
        {
                return;
        }
        App -> pMessageTemplate ->
                InitialUpdateFrame(pNewFrame,m_Document);
        CMessageView* pMessageView =
                (CMessageView*)pNewFrame -> GetActiveView();
        pMessageView -. SetMessageNumber(index);
}
```

ต่อไปจะเป็นฟังก์ชั่นที่ใช้สำหรับรับค่าของข้อความที่จะใช้แสดง ซึ่งได้แก่ SetMessageNumber ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CMessageView::SetMessageNumber(unsign int msg)
{
        MessageNumber = msg + 1;
        // Index in list box is zero based;mailbox is 1 based.

        Delete pSocket;
        pSocket = new Qsocket();
        pSocket -> SetRecieveTarget(this,WM_SOCKET_RESPONSE);
        if (!pSocket -> Connect(GetDocument() -> GetPop3(),110))
        {
                AfxMessageBox("Host is unreachable.");
                delete pSocket;
                pSocket = 0;
                return;
        }
}
```

ต่อไปเป็นการแก้ไขในฟังก์ชั่น OnSocket() ใน CMessageView ดังนี้

```
LRESULT CMessageView::OnSocket(WPARAM amount,LPARAM buffer)
{
        if ((int) amount > 0)
        {
                // Handle socket data
        }
        else
        {
                // If amount < 0 it is a receive command
                Cstring response;
                CMailDoc* doc = GetDocument();
                switch ((SocketRecieveCmd) amount)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
        case SocketStatusChanged:
        switch (pSocket -> GetStatus())
        {
                case CONNECTED:
                        // handle connection
                        break;
                case DISCONNECTED:
                        break;
                defualt:
                        break;
        }
        break;
    }
  }
  return 0;
}
```

และในกรณีของ CONNECTED นั้นจะเป็นดังนี้

```
// Switch to getline
pSocket -> SetRecieveTarget(NULL,0);
response = pSocket -> GetLine();
pSocket -> Send("USER " + GetDocument() -> GetUserid() + "\r\n");
response = pSocket -> GetLine();
pSocket -> Send("PASS " + GetDocument() -> GetPassword() + "\r\n");
response = pSocket -> GetLine();

pSocket -> SetRecieveTarget(this,WM_SOCKET_RESPONSE);
char command[20];
// Unlikely to have a 13 digit message number
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
sprintf(command,"RETR %l \r\n",MessageNumber);
pSocket -> Send(command);
```

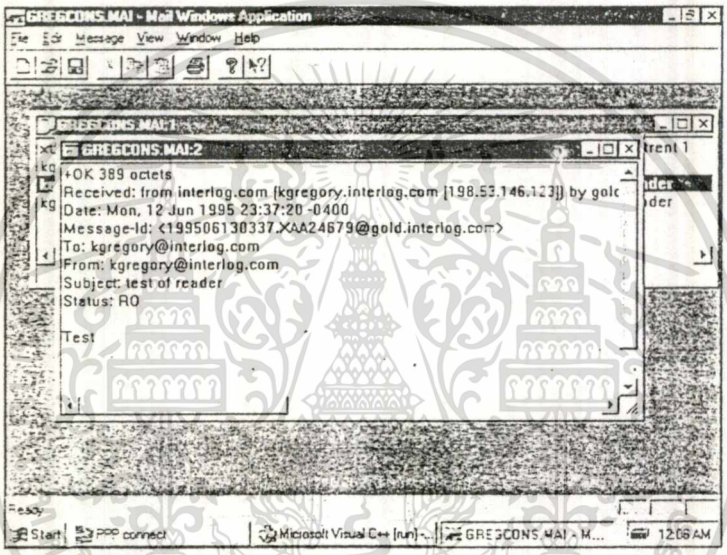
ในตอนแรกนั้นเราส่งคำสั่ง USER และ PASS แต่ขณะนี้เราเปลี่ยนมาอยู่ในเกทไลน์โหมด (getline mode) ก่อนการสร้างและการส่งคำสั่ง RETR เพื่อรับหมายเลขของแต่ละข้อความ ในตอนนี้เราจะมีบล็อกที่ชื่อ "Handle socket data" ซึ่งบล็อกนี้จะทำงานกับค่าตัวแปร CString ที่ชื่อ windowtext จนกว่าจะเจอ <CRLF>.<CRLF> ดังนั้นในส่วนของ "Handle socket data" จะเป็นดังนี้

```
windowtext += (char*) buffer;
if (windowtext.Right(4) == "\n.\r\n")
// End of transmission.
{
        // Find .. at start of line,and drop one.
        If (windowtext.Left(2) == "..")
        {
                windowtext = windowtext.Right windowtext.GetLength()-1);
        }
        char* position;
        char* windowbuffer = windowtext.GetBuffer(0);
        position = windowbuffer;
        while (position = strstr(position,"n.."))
        {
                int length = windowtext.GetLength();
                strcpy(position+1,position+2);
                // Skip second dot
                position += 2;
                // Get past the \n. left behind.
        }
        GetEditCtrl().SetWindowText(windowtext);
        // Switch to getline.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        pSocket -> SetRecieveTarget(NULL,0);
        pSocket -> Send("QUIT \r\n");
        CString response = pScoket -> GetLine();
        delete pSocket;
        pSocket = NULL;
}
```

เมื่อสมบูรณ์และทำการทดสอบถ้าถูกต้องจะได้ผลดังรูปที่ 4.6



รูปที่ 4.6 แสดงผลที่ได้จากการเลือกตัวเลือก Display

4.8 โค้ดสำหรับตัวเลือก Delete ในเมนู Message (Code for the Message,Delete Menu Item)

ฟังก์ชั่น OnMessageDelete() ใน CMessageView จะมีการทำงานคล้ายกับ Check Mail คือทำการติดต่อ หลังจากนั้นใน OnSocket จะทำการส่งคำสั่ง USER และ PASS

```
void CMessageView::OnMessageDelete()
{
        delete pSocket;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pScoket = new Qsocket();
pSocket -> SetRecieveTarget(this,WM_SOCKET_RESPONSE);
State = Delete;
if (!pSocket -> Connect(GetDocument() -. GetPop3(),110));
{
        AfxMessageBox("Host is unreachable.");
        delete pSocket;
        pSocket = 0;
        return;
}
}
```

หลังจากนั้นต้องทำการเพิ่มโค้ดใน MAILVIEW.H ในส่วนของ enum คือ Delete เมื่อสถานะเป็นการเช็คเมล์จะสร้างบ็อกซ์ใหม่ขึ้นมา

```
if (Status == Delete)
{
        int ins\dex = MessageList.GetCurSel():
        if (index >= 0)
        {
                MessageList.ResetContent();
                char command[20];
                // Unlikely to have a 13 digit message number.
                sprintf(command,"DELE %i \r\n",index+1);
                pSocket -> Send(command);
                response = pScoket -> GetLine();

                pSocket -> Send("QUIT \r\n");
                response = pSocket -> GetLine();
                delete pSocket;
                pSocket = NULL;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            OnFileCheckmail();
        }
        else
        {
            Status = Idle;
        }
    }
```

ในขั้นตอนแรกเราต้องกำหนดว่าข้อความไหนจะถูกลบ หลังจากนั้นจะทำการส่งคำสั่ง DELE และ QUIT หลังจากนั้นทำการเรียกฟังก์ชั่น OnFileCheckmail() เพื่อทำการติดต่อใหม่ และแสดงรายการของเมล์ที่เหลือขึ้นมา

4.9 โค้ดสำหรับตัวเลือก New ในเมนู Message

(Code foe the Message,New Menu Item)

ต่อไปจะเป็นส่วนของฟังก์ชั่น OnMessageNew() ใน CMessageView ซึ่งจะมีการทำงานคล้ายกับ OnMessageDisplay() คือมีการสร้างส่วนที่มองเห็นใหม่ขึ้นมา ซึ่งในกรณีนี้เราจะใช้เทมป์เพลท CComposeTemplate

```
void CMailView::OnMessageNew()
{
    CMailApp* App = (CMailApp*)AfxGetApp();
    CMDIChildWnd* pNewFrame = (CMDIChildWnd*)App ->
        pComposeTemplate -> CreateNewFrame(m_pDocument,NULL);
    if (pNewFrame == NULL)
    {
        return;
    }
    App -> pComposeTemplate -> InitialUpdateFrame(pNewFrame,
        m_Document);
    CComposeView* pComposeView =
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                (CComposeView*)pNewFrame -> GetActiveView();
        pComposeView -> Fill();
}
```

ต่อไปทำการเพิ่มเติมโค้ดในส่วนของ CComposeView:::OnInitialUpdate() ดังนี้

```
void CComposeView::OnInitialUpdate()
{
        GetEditCtrl().SetReadOnly(FALSE);
        CEditView::OnInitialUpdate();
}
```

ต่อไปแก้ไขโค้ดในส่วนของฟังก์ชั่น Fill()

```
void CComposeView::Fill()
{
        CString text = " ";
        text += "To: \r\n";
        text += "From: " + GetDocument() -> GetUserid()
                + "@" + GetDocument() ->GetSite() + "\r\n";
        text +="Subject: \r\n";
        text +="\r\n";
        GetEditCtrl().SetWindowText(text);
        int selchar = GetEditCtrl().LineIndex(GetEditCtrl().GetLineCount()-1);
        GetEditCtrl().SetSel(selchar,selchar);
}
```

ในส่วนนี้จะเป็นส่วนที่แสดงรายละเอียดบนหน้าจอใหม่ ดังนี้ : "To: " และตามด้วยช่องว่าง , บรรทัดต่อมาจะเป็น "From: " และตามด้วยข้อมูลของผู้ใช้งานซึ่งมีอยู่แล้ว , บรรทัดต่อมาคือ "Subject: " และช่องว่าง ซึ่งทั้งหมดที่กล่าวมานั้นเป็นส่วนของเฮดเดอร์ที่จำเป็นจะต้องมีบรรทัดต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไปจะเป็นบรรทัดว่าง 1 บรรทัด เพราะข้อความของเมล์นั้นจะต้องมีบรรทัดว่างระหว่างส่วนของเฮดเดอร์และส่วนของบอดี้

4.10 โค้ดสำหรับตัวเลือก Send ในเมนู Message

(Code for the Message,Send Menu Item)

ทำการแก้ไขส่วนของฟังก์ชั่น OnMessageSend() ใน CComposeView ดังนี้

```
void CComposeView::OnMessageSend()
{
        delete pSocket;
        pScoket = new Qsocket();
        pSocket -> SetRecieveTarget(this,WM_SOCKET_RESPONSE);
        if (!pSocket -> Connect(GetDocument() -> GetSMTP() , 25))
        {
                AfxMessageBoxt("Host is unreachable.");
                delete pScoket;
                pSocket = 0;
                return;
        }
}
```

สังเกตที่ฟังก์ชั่น Connect() จะเห็นว่าเรากำลังทำการติดต่อกับโฮสท์ของ SMTP ซึ่งจะใช้พอร์ต 25 แทนที่จะเป็นพอร์ต 110 สำหรับ POP3 ต่อไปทำการเปลี่ยนแปลงส่วนของฟังก์ชั่น OnSocekt()

```
LRESULT CComposeView::OnSocket(WPARAM amount , LPARAM buffer)
{
        if ((int)amount > 0)
        {
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else
        {
                // If amount > 0 it is a receive command
                CString response;
                unsigned int i;
                unsigned int lines;
                CString to;
                CMailDoc* doc = GetDocument();

                switch ((SocketRecieveCmd) amount)
                {
                        case SocketStatusChanged:
                                switch (pSocket -> GetSatatus())
                                {
                                        case CONNECTED:
                                                // Handle connection
                                                break;

                                        case DISCONNECTED:
                                                break;

                                        default:
                                                break;
                                }
                                break;
                }
        }
        return 0;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการเริ่มการติดต่อกับโฮสท์ของ SMTP นั้นจะเริ่มด้วยคำสั่ง HELO แทนที่จะเป็น USER และ PASS ของ POP3 ดังนั้นในส่วนของ "Handle socket" จะเป็นดังนี้

เริ่มจากการส่งคำสั่ง HELO

```
// Switch to getline.
pScoket -> SetRecieveTarget(NULL,0);
response = pSocket -> GetLine();
pSocket -> Send("HELO "
                + GetDocument() -> GetSite() + "\r\n");
response = pScoket -> GetLine();
```

ต่อไปส่งคำสั่ง MAIL FROM

```
pSocket -> Send("MAIL FROM: <"
                + GetDocument() -> GetUserid() + "@"
                + GetDocument() -> GetSite() + ">\r\n");
response = pSocket -> GetLine();

// Determine recipient - -only one at the moment.
// Should handle cc , bcc as well and for to , cc ,bcc
// should handle comma - separate names.
// Comma in the To field will probably blow this up

char buffer[5004];
// For \r , \n , zero term , and perhaps an extra.
lines = GetEditCtrl().GetLineCount();
unsigned int linelength;
for ( i=0;i<lines;i++)
{
        linelength = GetEditCtrl().GetLine(i,buffer,5000);
        if (strnicmp(buffer,"To: ",4) == 0)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        {
                buffer[linelength] = 0;
                to = buffer+4;
                break;
        }
}
```

ต่อไปส่งคำสั่ง RCPT TO

```
pSocket -> Send("RCPT TO: <" + to + ">\r\n");
response = pSocket -> GetLine();
```

ซึ่งในขั้นตอนนี้ที่โฮสท์ของ SMTP จะทำการตรวจสอบว่าถูกต้องหรือไม่ ถ้าใช่จะทำการส่งข้อความออกไป

```
pSocket -> Send("DATA \r\n");
response = pSocket -> GetLine();

for (i=0;i<lines;i++)
{
        linelength = GetEditCtrl().GetLine(i,buffer,5000);
        buffer[linelength] = 0;
        if (buffer[0] == ".")
        {
                pSocket -> Send('.' + (CString)buffer + "\r\n");
        }
        else
        {
                pSocket -> Send((CString)buffer + "\r\n");
        }
pSocket -> Send(".\r\n");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
response = pSocket t-> GetLine();
```

โค้ดในส่วนนี้จะทำการส่งข้อความแต่ละบรรทัดไปยังโฮสท์ของ SMTP ซึ่งถ้าบรรทัดแรกเริ่มต้นด้วย "." เราจะส่ง "." ออกไปอีก หลังจากนั้นทำการส่งข้อความออกไปเรื่อย ๆ จนหมดข้อความแล้วก็จะส่ง "." ออกไปเพื่อแจ้งให้โฮสท์ทราบว่าข้อความสมบูรณ์แล้ว หลังจากนั้นจะส่งคำสั่ง QUIT ออกไป

```
pSocket -> Send("QUIT \r\n");
response = pSocket -> GetLine();
```

เพื่อเป็นการง่ายที่จะทำให้ผู้ใช้งานได้ทราบว่าข้อความของเมล์นั้นถูกส่งออกไปเรียบร้อยแล้ว สามารถทำได้โดยการทำให้หน้าต่างที่เขียนข้อความและถูกส่งไปนั้นหายไปโดยอัตโนมัติเมื่อการส่งเสร็จสมบูรณ์ เราทำได้โดยการเพิ่มโค้ดเข้าไปดังนี้

```
CWnd *wnd = GetParent();
wnd -> DestroyWindow();
```

4.11 โค้ดสำหรับตัวเลือก Reply ในเมนู Message

(Code for the Message,Reply Menu Item)

การทำงานของการตอบข้อความนั้นคล้ายกับการสร้างส่วนที่จะใช้เขียนข้อความเพื่อทำการส่งออกไปใหม่ แต่จะมีส่วนที่แตกต่างกันบ้างเล็กน้อย คือในส่วน "Subject: " ควรจะเป็นของเมล์เดิมที่จะตอบ และส่วนของ "TO: " ควรจะมาจากส่วนของ "From: " ของเมล์เดิม และ "From: " จะเป็นข้อมูลของผู้ใช้งานเองเหมือนเดิม ทางที่ง่ายที่สุดคือการเพิ่มพารามิเตอร์ "parameter" ในฟังก์ชั่น Fill() เข้าไปสองตัวคือส่วนของ To และ From ดังนั้นส่วนของ CMessageView::OnMessageReply() จะเป็นดังนี้

```
void CMessageView::OnMessageReply()
{
        CMailApp* app =(CMailApp*)AfxGetApp();
        CMDIChildWnd* pNewFrame = (CMDIChildWnd*)App
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        - >     p C o m p o s e T e m p l a t e     - >
CreateNewFrame(m_pDocumen,NULL);
if (pNewFrame == NULL)
{
        return;
}
App -> pComposeTemplate -> InitialUpdateFrame(
        pNewFrame,m_pDocument);

CString from,subject;
char buffer[5004];
// For \r , \n , zero tterm , and perhaps an extra.
unsigned int lines = GetEditCtrl().GetLineCount();
unsigned int linelength;
for (unsigned int i= 0;i<lines;i++)
{
        linelength = GetEditCtrl().GetLine(i,butffer,5000);
        if (strnicmp(buffer,"From: ",6) == 0)
        {
                buffer[linelength] = 0;
                from = buffer + 6;
                break;
        }
}
for ( i=0;i<lines;i++)
{
        linelength = GetEditCtrl().GetLine(i,buffer,5000);
        if(strnicmp(buffer,"Subject: ",9) == 0)
        {
                buffer[linelength] = 0;
                subject = buffer + 9;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    break;
                }
            }
            CComposeView* pComposeView =
                (CComposeView*)pNewFrame -> GetActiveView();
            pComposeView -> Fill(from,subject);
}
```

ต่อไปทำการเปลี่ยนแปลงในส่วนของฟังก์ชั่น Fill() ซึ่งทำได้โดยการแก้ไขใน COMPOSE.H จาก

```
void Fill();
```

เป็น

```
void Fill(CString To=" ",CString Subject=" ");
```

และแก้ไขฟังก์ชั่นดังนี้

```
void CComposeView::Fill(CString To,CString Subject)
{
        CString text += " ";
        text += "To: " + To + "\r\n";
        text += "From: " + GetDocument() -> GetUserid()
                + "@" + GetDocument() -> GetSite() + "\r\n";
        text += "Subject: " + Subject + "\r\n";
        GetEditCtrl().SetWindowText(text);
        int selchar = GetEditCtrl().LineIndex(
                GetEditCtrl().GetLineCount()-1);
        GetEditCtrl().SetSel(selchar,selchar);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.12 ดับเบิ้ลคลิ้กบนบ็อกซ์รายการ (Double Clicking a List Box)

ขั้นตอนต่อไปจะเป็นส่วนที่ทำให้ผู้ใช้งานโปรแกรมนี้ได้ง่ายขึ้นในการอ่านข้อความของเมล์ที่ต้องการโดยการทำดับเบิ้ลคลิ้กที่ข้อความที่แสดงได้เลย ไม่ต้องใช้เมนูและตัวเลือก ซึ่งเราสามารถทำได้โดยการแก้ไขใน MAILVIEW.CPP ดังนี้

```
BEGIN_MESSAGE_MAP(CMailView , Cview)
        //{{AFX_MSG_MAP(CMailView)
        ON_COMMAND(ID_FILE_CHECKMAIL,OnFileCheckmail)
        ON_COMMAND(ID_MESSAGE_DISPLAY,OnMessageDisplay)
        ON_COMMAND(ID_MESSAGE_NEW,OnMessageNew)
        ON_COMMAND(ID_MESSAGE_DELETE,OnMessageDelete)
        ON_UPDATE_COMMAND_UI(ID_MESSAGE_DELETE,
            OnUpdateMessageDelete)
        // AFX_MSG_MAP
        ON_MESSAGE(WM_SOCKET_RESPONSE,OnSocket)
        ON_LBN_DBLCLK(IDC_MESSAGELISTBOX,OnMessageDisplay)
END_MESSAGE_MAP
```

4.13 การตรวจสอบความผิดพลาด (Error Checking)

เราจำเป็นต้องมีการตรวจสอบความถูกต้องเมื่อทำการติดต่อกับโฮสท์ของ SMTP และ POP3 และแจ้งให้ผู้ใช้งานได้ทราบว่าถูกต้องหรือไม่

ใน CMailView::Onsocket() เมื่อเราส่งคำสั่ง USER และ PASS ออกไป ถ้าผลตอบสนองที่กลับมาเป็น "-ERR" เราไม่สามารถทำการติดต่อกับเมล์บ็อกซ์ได้ และผู้ใช้งานควรจะถูกแจ้งให้ทำการตั้งค่าเริ่มต้นใหม่อีกครั้ง เราสามารถทำส่วนนี้ได้โดยการแก้ไขโค้ดส่วน if ที่ต่อจากการส่งคำสั่ง USER ออกไปดังนี้

```
if (response.Left(3) == "+OK")
```

และในส่วนของ else เพื่อแจ้งให้ผู้ใช้งานทราบเกี่ยวกับปัญหาและยกเลิกการติดต่อดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
else
{
        AfxMessageBox("Userid incorrect.Enter the
                correct userid with File Setup.");
        pSocket -> Send -> ("QUIT \r\n");
        response = pSocket -> GetLine();
        delete pSocket;
        pSocket = NULL;
}
```

และทำการตรวจสอบหลังจากส่งคำสั่ง PASS ออกไป ดังนั้นในส่วนของ CONNECTED จะเป็นดังนี้

```
case CONNECTED:
        // Switch togetline.
        pSocket -> SetRecieveTarget(NULL,0);
        response = pSocket -> GetLine();
        pSocket -> Send("USER " + doc -> tGetUserid() + "\r\n");
        response = pSocket -> GetLine()
        if (response.LEFT(3) == "+OK")
        {
                pSocket -> Send("PASS " + doc -> GetPassword() + "\r\n");
                response = pSocket -> GetLine();
                if (response.LEFT(3) == "+OK")
                {
                        if (State == CheckMail)
                        {
                                FillListBox();
                                Status = Idle;;;
                        }
                        if (State == Delete)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                {
                        int index = MessageList.GetCurSel();
                        MessageList.ResetContent();
                        char command[20];
                        // Unlikely to have a 13 digit message number
                        sprintf(command,"DELE %i \r\n",index+1);
                        pSocket -> Send(command);
                        response = pSocket -> GetLine();

                        pSocket -> ("QUIT \r\n");
                        response -> GetLine();
                        delete pSocket;
                        pSocket = NULL;

                        OnFileCheckmail();
                }
        }
        else
        {
                AfxMessageBox("Password incorrect.Enter
                        the correct password with File Setup.");
                pSocket -> Send("QUIT \r\n");
                response = pSocket -> GetLine();
                delete pSocket;
                pSocket = NULL;
        }
}
else
{
        AfxMessageBox("Userid incorrect.Enter
                the correct userid with File Setup.");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            pSocket -> Send("QUIT \r\n");
            response = pSocket -> GetLine();
            delete pSocket;
            pSocket = NULL;
        }
        break;
```

หลังจากนั้นทำการแก้ไขในส่วนของ CMessageView::OnSocket() และ CComposeView::OnSoket() ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5 บทวิจารณ์และสรุป

5.1 การใช้โปรแกรม

เนื่องมาจากโปรแกรมนี้ใช้โปรโตคอลที่แตกต่างกันสองชนิด คือ โปรโตคอล SMTP และโปรโตคอล POP3 แต่ผู้ใช้ไม่จำเป็นจะต้องมีความรู้ในส่วนของโปรโตคอลทั้งสองตัวนี้ก็สามารถใช้งานโปรแกรมนี้ได้

โดยปกติแล้วผู้บริการอินเตอร์เน็ตจะบอกให้ผู้ใช้บริการได้ทราบถึงโฮสท์ของ SMTP และโฮสท์ของ POP3 และชื่อที่ใช้ในการติดต่อกับโฮสท์ของ POP3 และรหัสผ่านของผู้ใช้ ผู้ให้บริการบางรายจะให้ข้อมูลที่เกี่ยวกับโฮสท์ของ POP3 มาในรูปแบบของ "แอคเคาทน์" (Account) โดยที่ส่วนที่อยู่หน้าตัวอักษร " @ " จะเป็นชื่อที่ใช้ในการติดต่อกับโฮสท์ของ POP3 และส่วนที่อยู่ต่อท้ายตัวอักษร " @ " จะเป็นชื่อแอดเดรสของโฮสท์ของ POP3

เมื่อเริ่มต้นใช้งานโปรแกรมนี้ผู้ใช้งานจะต้องทำการเลือกเมนู File หลังจากนั้นก็เลือกตัวเลือก Setup จะมีเป็นไดอะล็อกบล็อกขึ้นมาและมีที่ว่างให้ผู้ใช้งานได้กรอกรายละเอียดต่าง ๆ ลงไป ซึ่งได้แก่ ชื่อของโฮสท์ทั้งของ SMTP และ POP3 , ชื่อที่ใช้ในการติดต่อกับโฮสท์ของ POP3 , รหัสผ่านของผู้ใช้งาน และชื่อจริงของผู้ใช้งาน หลังจากนั้นกดปุ่ม "OK" แล้วเลือกไปที่เมนู File และตัวเลือก Save As และตั้งชื่อเก็บเอาไว้เพื่อที่การใช้งานครั้งต่อไปจะได้ไม่ต้องตั้งค่าเริ่มต้นใหม่ เมื่อต้องการใช้งานอีกครั้งก็เพียงเลือกเมนู File และตัวเอก Open และเรียกไฟล์ที่เราเก็บบันทึกไว้เอาขึ้นมาใช้งานได้ทันที

ต่อจากนั้นถ้าต้องการตรวจดูเมล์ของผู้ใช้งานที่อยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 ว่ามีอยู่หรือไม่และมีอะไรบ้าง ก็จะต้องทำการเลือกเมนู File และตัวเลือก Check Mail โปรแกรมก็จะทำการแสดงรายการของเมล์ที่มีอยู่ทั้งหมดในในเมล์บ็อกซ์ของโฮสท์ของ POP3 บนหน้าต่างที่สร้างขึ้นมา และถ้าผู้ใช้งานต้องการจะอ่านข้อความในเมล์ไหนก็ทำการดับเบิ้ลคลิ้กเมาส์ (Double Click mouse) หรือคลิ้กเพียงทีเดียวที่ลำดับของเมล์ที่ต้องการจะอ่าน และเลือกเมนู Message และตัวเลือก Display ก็จะมีหน้าต่างขึ้นมาใหม่ซึ่งจะแสดงข้อความที่มีอยู่ในเมล์นั้นขึ้นมาให้ผู้ใช้งานได้อ่าน ถ้าต้องการจะลบเมล์ใด ๆ ก็ตามที่ไม่ต้องการเก็บไว้ให้คลิ้กที่เมล์นั้นในหน้าต่างที่ได้จากการ Check Mail แล้วเลือกเมนู Message และตัวเลือก Delete ก็สามารถทำการลบเมล์นั้นได้โดยที่หน้าต่างนี้ก็จะแสดงเมล์ที่มีเหลืออยู่ในเมล์บ็อกซ์ของโฮสท์ของ POP3 ให้ผู้ใช้งานได้ทราบ ถ้าต้องการจะตอบเมล์ใด ๆ ก็จะต้องเปิดเมล์นั้นเพื่อทำการอ่านเสียก่อนหลังจากนั้นก็เลือกไปที่เมนู Message และตัวเลือก Reply ก็จะมีหน้าต่างขึ้นมาใหม่เพื่อให้ผู้ใช้ทำการพิมพ์ข้อความของเมล์ที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะตอบกลับไป ถ้าต้องการส่งเมล์ไปหาผู้ใดก็ตามให้เลือกไปที่เมนู Message และตัวเลือก New ก็จะมีหน้าต่างขึ้นมาใหม่เพื่อให้ผู้ใช้ทำการพิมพ์ข้อความของเมล์ที่จะส่งออกไป และหลังจากที่พิมพ์ข้อความของเมล์เสร็จเรียบร้อยแล้วทั้งในส่วนของตัวเลือก Reply หรือ New ในเมนู Message และจะทำการส่งเมล์นั้นออกไปผู้ใช้ก็จะต้องทำการเลือกไปที่เมนู Message และตัวเลือก Send ก็จะสามารถทำการส่งเมล์นั้นออกไปอย่างสมบูรณ์

5.2 บทวิจารณ์

ในโปรแกรมที่จัดทำขึ้นมานี้ยังมีจุดบกพร่องที่ต้องการการแก้ไขเพิ่มเติมในโอกาสต่อไปอีกหลายประการตัวอย่างเช่น

* ในการตั้งค่าเริ่มต้นนั้น ถ้าสมมติว่าผู้ใช้งานกรอกข้อมูลรายละเอียดผิดพลาด หรือกรอกไม่ครบถ้วนสมบูรณ์ เช่น ในส่วนของ UserID ไม่ได้มีตัวอักษร " @ " เป็นต้น ในโปรแกรมน่าจะมีการตรวจสอบถึงความถูกต้องของข้อมูลรายละเอียดต่าง ๆ หลังจากที่ผู้ใช้งานกดปุ่ม " OK " เพื่อยอมรับในสิ่งต่าง ๆ ที่ผู้ใช้งานกรอกลงไป โดยอาจจะแสดงเป็นข้อความขึ้นมาถามความแน่ใจอีกครั้งเพื่อให้ผู้ใช้งานได้มีโอกาสตรวจสอบข้อมูลรายละเอียดต่าง ๆ ที่ตนเองกรอกลงไปอีกครั้งว่าถูกต้องสมบูรณ์และครบถ้วนหรือไม่ ถ้าตรวจสอบแล้วพบว่าครบถ้วนสมบูรณ์แล้วจึงทำงานในขั้นตอนต่อไปได้

* ในการที่จะเขียนข้อความในเมล์ที่จะถูกส่งออกไปนั้นผู้ใช้งานจะต้องทำการเว้นบรรทัดว่างไว้หนึ่งบรรทัดระหว่างในส่วนของเฮดเดอร์และส่วนของบอดี้ ดังนั้นก่อนที่จะมีการทำการติดต่อกับระบบอินเตอร์เน็ตนั้นจึงควรจะมีการตรวจสอบว่ามีบรรทัดว่างที่ต้องการนี้หรือไม่ ดังนั้นควรมีการแจ้งให้ผู้ใช้งานได้ทราบในกรณีที่หลังจากทำการเลือกเมนู Message และตัวเลือก Send และตรวจสอบพบว่าผู้ใช้งานไม่เว้นบรรทัดว่างไว้หนึ่งบรรทัด เพื่อให้ผู้ใช้ได้ทำการเว้นบรรทัดให้เรียบร้อยเสียก่อน และตรวจสอบอีกครั้งถ้าถูกต้องจึงจะทำการส่งเมล์นั้นออกไป

* โปรแกรมน่าที่จะสามารถส่งเมล์ไปยังผู้รับได้ทีเดียวหลาย ๆ คน ตัวอย่างเช่นในบรรทัดของเฮดเดอร์ "TO:" นั้นฟิลด์บอดี้ของเฮดเดอร์นี้น่าจะเขียนได้เป็น

To:s6014076@diamond,s6014494@diamond

ในการออกแบบเพื่อที่จะสามารถทำงานเช่นนี้ได้นั้นอาจจะต้องใช้ลักษณะของ "อาร์เรย์" (array) โดยที่เมื่อพบบรรทัดที่ขึ้นต้นด้วยเฮดเดอร์ "To:" โปรแกรมควรจะตรวจสอบว่าในบรรทัดนั้นมีเครื่องหมาย " , " หรือไม่ ถ้ามีก็ทำการแยกฟิลด์บอดี้นั้นออกเป็นส่วน ๆ และทำการส่งเมล์ไปให้ยังผู้รับแต่ละคนต่อไป

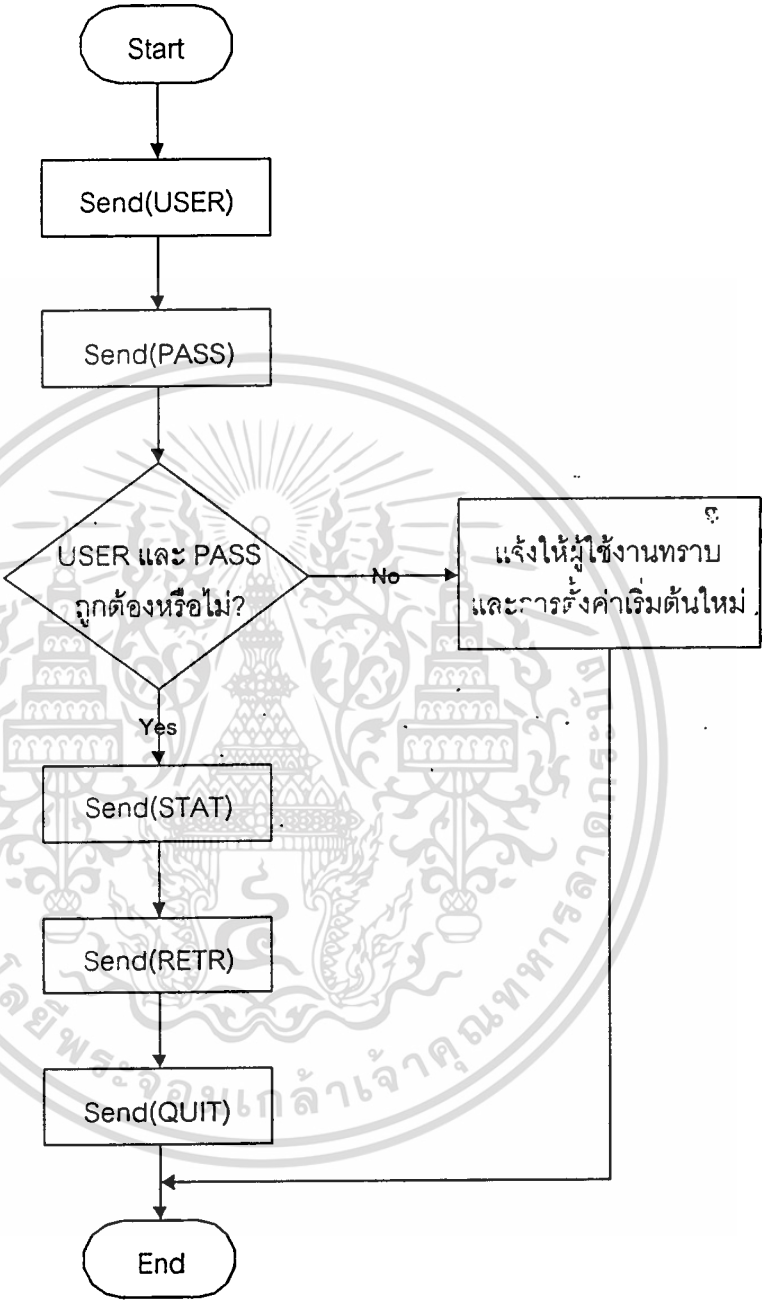
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

* การที่มีการตั้งค่าเริ่มต้นโดยการที่มีการใส่รหัสผ่านเอาไว้ด้วยนั้นอาจจะเป็นการที่ไม่ดี เพราะจะทำให้ผู้ใช้ใด ๆ สามารถตรวจสอบและกระทำการใด ๆ กับเมล์ของผู้ใช้งานคนอื่นได้ในกรณีที่เครื่องที่มีโปรแกรมนี้อยู่นั้นมีผู้ใช้งานหลายคน จึงเป็นการไม่ส่วนตัว ดังนั้นควรจะไม่มีในส่วนรหัสในการตั้งค่าเริ่มต้น แต่อาจทำเป็นไดอะล็อกบล็อกขึ้นมาเพื่อให้ผู้ไงานได้กรอกรหัสผ่านเมื่อผู้ใช้งานต้องการตรวจสอบและดูรายการเมล์ของตนเอง

โปรแกรมนี้เป็นเพียงโปรแกรมพื้นฐานเพื่อใช้ในการศึกษาและพัฒนาปรับปรุงต่อไปให้ดีขึ้นสามารถใช้งานได้มากขึ้นตามลำดับ

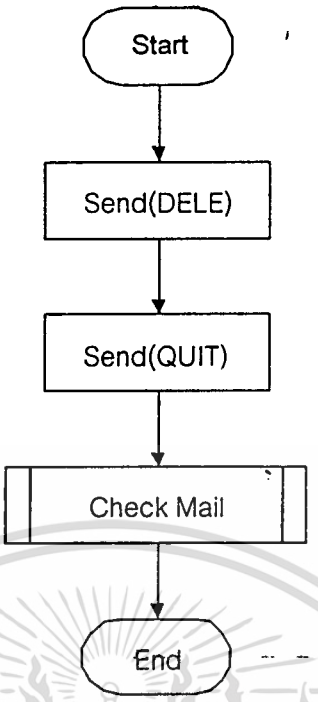
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก



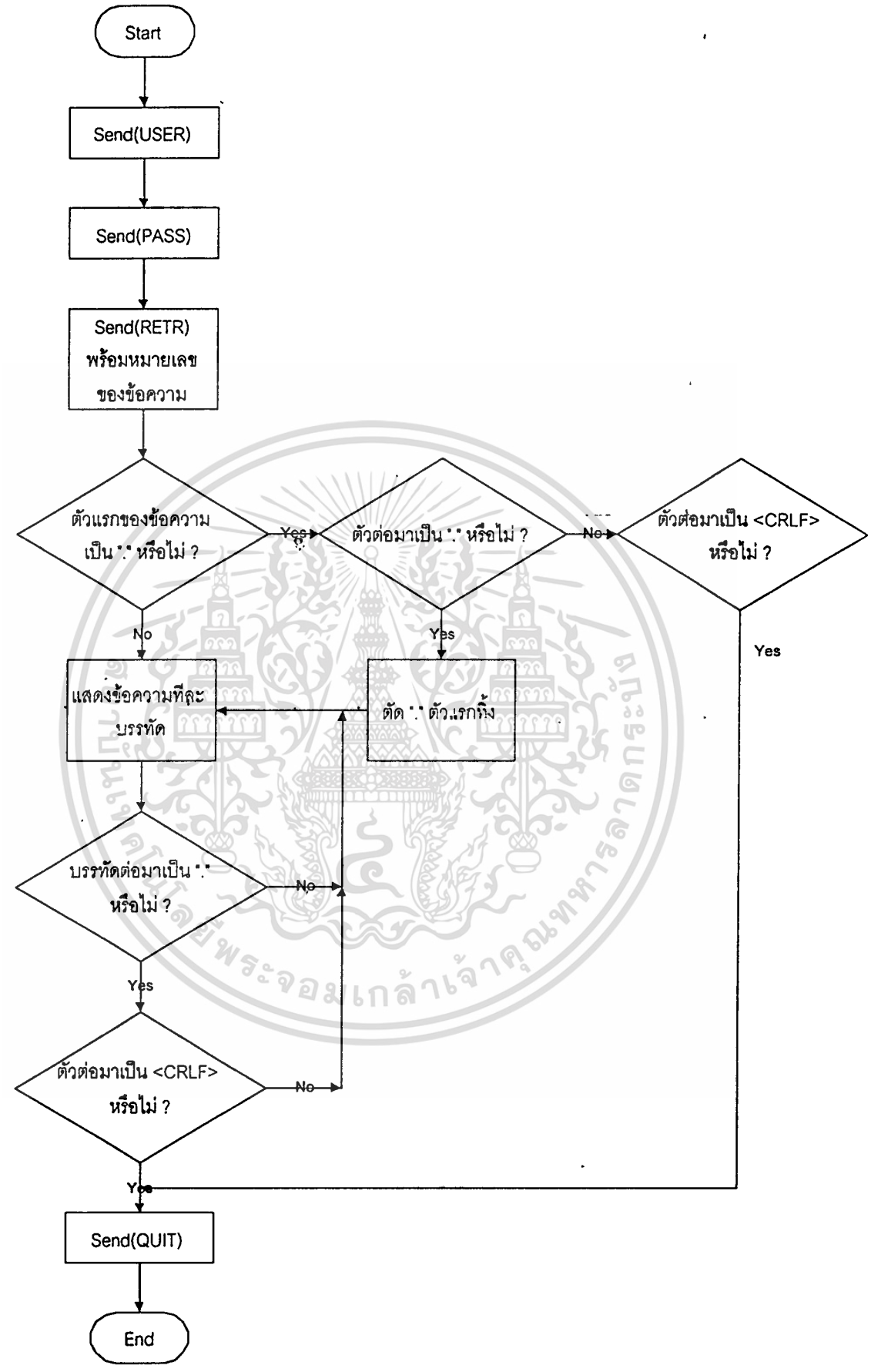
Flow Chart แสดงการทำงานของตัวเลือก Check Mail ในเมนู File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



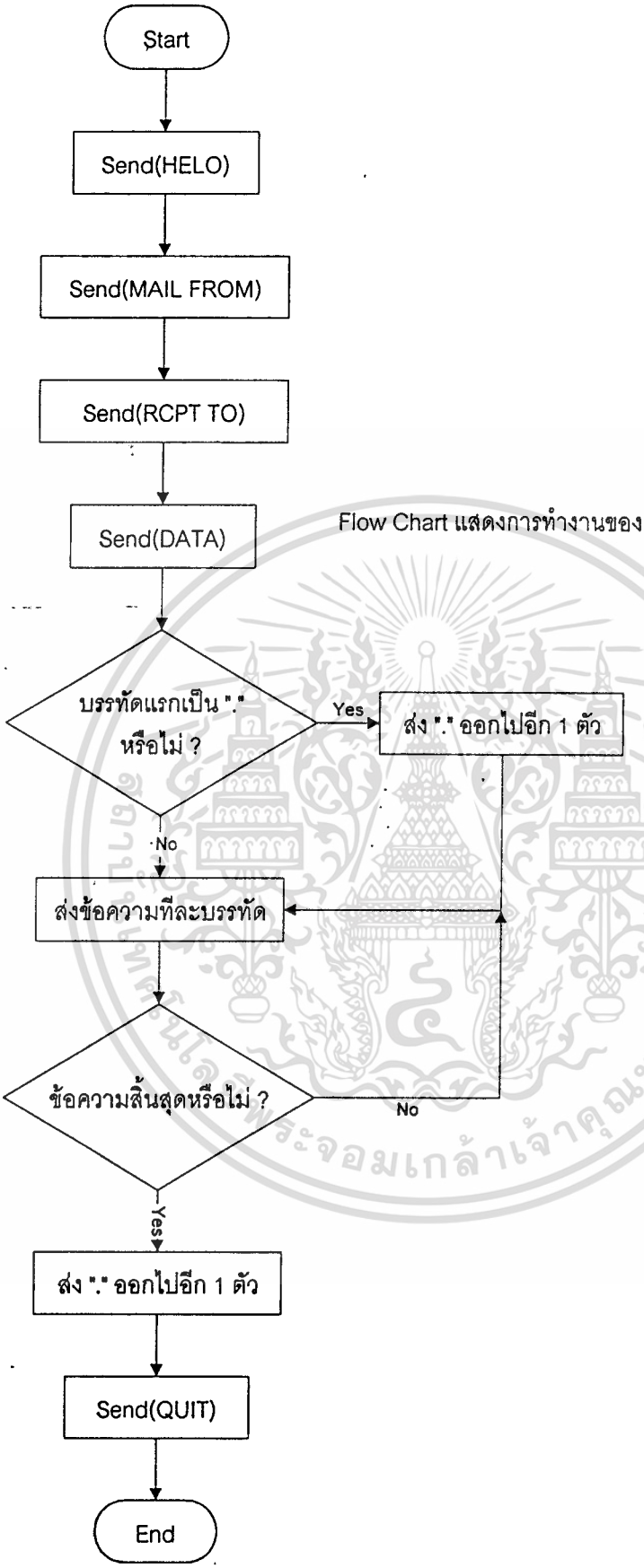
Flow Chart แสดงการทำงานของตัวเลือก Check Mail ในเมนู File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Flow Chart แสดงการทำงานของตัวเลือก Display ในเมนู Message

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Flow Chart แสดงการทำงานของตัวเลือก Send ในเมนู Message

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

เอกสารอ้างอิงที่เป็นหนังสือภาษาอังกฤษ

- Ori Gurewich and Natharn Gurewich,"Teach Yourself Visual c++ 4 in 21 days", SAMS Publishing,810 p.,1996.
- Kate Gregory,"Building Internet Application with Visual C++",Que Coperation, 460 p.,1995.
- Marshall T. Rose,"The Internet Message",P T R Prentice-Hall,370 p.,1993.

เอกสารอ้างอิงที่เป็นวารสารภาษาอังกฤษ

- Jon B. Postel,"Simple Mail Transfer Protocal",request for comments 821, USC/Information Sciences Institue,August 1982.
- Marshall T. Rose,"Post Office Protocal",Version 3.,request for comments 1225, Performance Systems International,Inc.,May 1991.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ขอกราบขอบพระคุณทุกท่านที่จะกล่าวถึงต่อไปนี้เพราะเป็นส่วนที่เป็นแรงผลักดันให้ข้าพเจ้ามีความตั้งใจและความพยายามในการทำโครงงานชิ้นนี้จนสมบูรณ์

- บิดา มารดาของข้าพเจ้า ที่คอยให้กำลังใจในการทำโครงงาน

- อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษาที่คอยให้การช่วยเหลือไม่ว่าจะเป็นคำแนะนำและการกระตุ้นข้าพเจ้าในการทำโปรเจคชิ้นนี้

- เพื่อน ๆ พี่ ๆ น้อง ๆ ชุมนุมวิชาการ วิศวะฯ ลาดกระบังฯ ทุกคนที่เป็นกำลังใจและคอยช่วยงานด้วยดี

- เพื่อน ๆ พี่ ๆ น้อง ๆ ชมรมบาสเกตบอล ลาดกระบังฯ ทุกคนที่เป็นกำลังใจและคอยช่วยงานด้วยดี

- เพื่อน ๆ ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคนที่คอยเป็นห่วงและให้กำลังใจด้วยดีเสมอมา

- และบุคคลอื่น ๆ ที่ไม่ได้กล่าวไว้ในที่นี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้