การพัฒนาโปรแกรมรับส่งไฟล์ข้อมูล ส่วนไคลเอนท์

A DEVELOPMENT OF FTP CLIENT

ปรียาภรณ์ จันทรโชติ
สมพรลักษณ์ แก่นสาร
สุทธิลักษณ์ ศุภองค์ประภา

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

A Development of FTP Client

PREYAPORN CHANTARACHOT
SOMPORNLAK KANSARN
SUTTILAK SUPAONGPRAPAR

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCES
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 1999

หัวข้อปัญหาพิเศษ	การพัฒนาโปรแกรมรับส่งไฟล์ข้อมูลส่วนไคลเอนท์		
	A DEVELOPMENT OF FTP CLIENT		
ชื่อนักศึกษา	นางสาวปรียาภรณ์	จันทรโชติ	39054638
	นางสาวสมพรลักษณ์	แก่นสาร	39054668
	นางสาวสุทธิลักษณ์	ศุภองค์ประภา	39054675
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์		
สาขาวิชา	วิทยาการคอมพิวเตอร์		
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ		

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2542

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	อาจารย์นันทิกา เบญจเทพานันท์	
กรรมการ	อาจารย์ไพรบูลย์ พันธรักษ์พงษ์	
กรรมการและอาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

(อาจารย์ ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ	การพัฒนาโปรแกรมรับส่งไฟล์ข้อมูล ส่วนไคลเอนท์		
ชื่อนักศึกษา	นางสาวปรียาภรณ์	จันทรโชติ	39054638
	นางสาวสมพรลักษณ์	แก่นสาร	39054668
	นางสาวสุทธิลักษณ์	ศุภองค์ประภา	39054675
ปริญญา	วิทยาศาสตรบัณฑิต		
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์		
สาขาวิชา	วิทยาการคอมพิวเตอร์		
ปีการศึกษา	2542		
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ		

บทคัดย่อ

โครงงานพิเศษเรื่อง โปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ (File Transfer Protocol - FTP Client) นี้ เป็นการศึกษาการเขียนโปรแกรมสำหรับรับส่งไฟล์ข้อมูล ระหว่างคอมพิวเตอร์ 2 เครื่อง (Client - Server) ซึ่งจะทำการเขียนและพัฒนาโปรแกรมฝั่งไคลเอนท์ โดยใช้ Visual C++ Version 6.0 มาเป็นเครื่องมือ ในการเขียนโปรแกรม

โดยโครงงานพิเศษนี้ เป็นโปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ที่สามารถติดต่อกับโปรแกรมรับส่งไฟล์ข้อมูลฝั่งเซิฟเวอร์ที่เปิดให้บริการทั่วไปได้ โดยมีฟังก์ชันพื้นฐานให้ใช้อย่างครบถ้วนสมบูรณ์ และยังมีจุดเด่นในเรื่องความสามารถในการทำงานแบบ Multitasking คือ สามารถทำงานในหลายๆ ส่วนได้พร้อมๆ กัน เช่น สามารถเชื่อมต่อไปยังหลายเซิฟเวอร์พร้อมกัน (Multiple Host) สามารถดาวน์โหลด หรือ อัพโหลด หลายไฟล์ได้พร้อมๆ กัน และเมื่อเกิดปัญหาขาดการติดต่อจากเซิฟเวอร์ ตัวโปรแกรมก็จะเชื่อมต่อเข้าเซิฟเวอร์นั้นๆ ใหม่อีกรอบโดยอัตโนมัติ และทำการดาวน์โหลด หรือ อัพโหลดต่อจากเดิม (Auto resume) ซึ่งในขณะที่ทำการ Auto resume นั้น ก็ยังสามารถทำงานอื่นๆไปพร้อมๆกันได้ด้วย

ลักษณะรูปแบบโปรแกรมจะเน้นการใช้งานที่สะดวก ง่ายต่อความเข้าใจสำหรับผู้เริ่มต้นใช้โปรแกรม

Special Project Title	A Development of FTP Client		
Students	Miss. Preyaporn	Chantarachot	39054638
	Miss. Sompornlak	Kansarn	39054668
	Miss. Suttilak	Supaongprapar	39054675
Degree	Bachelor's Degree of Science		
Department	Mathematics and Computer Sciences, Faculty of Science		
Programme	Computer Sciences		
Academic Year	1999		
Special Project Advisor	Lecturer Wisan Tangwongcharoen		

ABSTRACT

Special problem subject is A Development of File Transfer Protocol (FTP) Client. This is aimed for studying the problem on sending and receiving file between two computers. By developing the program we use Visual C++ Version 6.0 as the important tools.

This special problem is the program for receiving and sending file information with general FTP Server that have providing in the network. The multi-tasking oriented was viewed as the outstanding capability. There are many functions like multiple host, multiple file upload, multiple file download, resume or automatic resume when connection fail. And doing difference function simultaneously.

This is very convenient to use and easy to understand for user.

กิตติกรรมประกาศ

ขอขอบพระคุณบิดา มารดา ที่ให้กำเนิดเลี้ยงดูพวกเราด้วยความรัก ความเอาใจใส่ คอยอบรมชี้แนะ สิ่งที่ถูกที่ควร จนกระทั่งพวกเราเติบใหญ่จนถึงทุกวันนี้

ขอขอบคุณ อาจารย์วิสันต์ ตั้งวงษ์เจริญ อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำให้ข้อคิดดี ๆ ในการทำปัญหาพิเศษนี้และเป็นที่ปรึกษาในการแก้ปัญหาต่าง ๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

ขอขอบคุณ อาจารย์นรฤทธิ์ สุนทรศารกุล ที่ช่วยให้คำแนะนำ คำปรึกษาและให้ข้อคิดดี ๆ ในการทำปัญหาพิเศษนี้

ขอขอบคุณ อาจารย์ ธีรวัฒน์ ประกอบผล ที่ให้คำแนะนำในการเขียนโปรแกรม สำหรับการทำโครงงานนี้

ขอขอบคุณอาจารย์ทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ทั้งในภาคทฤษฎี และภาคปฏิบัติ ตลอดระยะเวลา 4 ปี จนกระทั่งโครงงานสัมฤทธิ์ผลด้วยดี

ขอขอบคุณรุ่นพี่ และเพื่อนๆที่ให้คำแนะนำและคอยช่วยเหลือเป็นกำลังใจมาตลอด

คณะผู้จัดทำ

มีนาคม 2543

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาของปัญหาพิเศษ

ในปัจจุบันมีการใช้เทคโนโลยีของอินเตอร์เนต (Internet) กันอย่างแพร่หลาย ไม่ว่าจะเป็น World Wide Web , IRC , E – mail ฯลฯ ซึ่งมีพื้นฐานอยู่บนความต้องการในการติดต่อสื่อสาร โดยไม่จำกัดระยะทาง ระหว่างผู้ส่งสื่อ และผู้รับสื่อ บ่อยครั้งที่การติดต่อสื่อสาร มีความต้องการที่จะรับส่งข้อมูล ที่ถูกเก็บอยู่ในรูปแบบของไฟล์ข้อมูลในคอมพิวเตอร์ ซึ่งโปรแกรมที่จะใช้ในการรับส่งไฟล์ข้อมูลนั้น คือ โปรแกรมรับส่งไฟล์ข้อมูล (FTP : File Transfer Protocol) โดยจะมีส่วนไคลเอนท์ที่ทำหน้าที่ส่งไฟล์ข้อมูลไปยังส่วนเซิฟเวอร์หรือรับข้อมูลมาจากส่วนเซิฟเวอร์ ซึ่งส่วนเซิฟเวอร์จะคอยเก็บไฟล์ข้อมูลที่ได้รับมานั้น ไว้ในระบบฐานข้อมูล (Database) และนำไฟล์ข้อมูลที่ถูกเรียก ส่งไปยังส่วนไคลเอนท์

จากความสำคัญของการรับส่งไฟล์ข้อมูลนี้ เราจึงมีความเห็นว่า การศึกษาในเรื่องของโปรแกรมรับส่งไฟล์ข้อมูลมีประโยชน์อย่างมาก ที่จะทำให้เกิดความเข้าใจพื้นฐาน เพื่อนำไปประยุกต์ และสามารถใช้ในการพัฒนาต่อไปได้ในอนาคต

1.2 วัตถุประสงค์ของปัญหาพิเศษ

1. ศึกษาพื้นฐานของการติดต่อสื่อสารระหว่างคอมพิวเตอร์ โดยผ่านทางโพรโตคอล TCP / IP
2. ศึกษาและทำความเข้าใจ การทำงานของโพรโตคอลส่งผ่านข้อมูล (FTP Protocol) โดยละเอียด
3. สร้างโปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ที่มีฟังก์ชันมาตรฐานครบถ้วน ซึ่งสามารถนำไปใช้ในการติดต่อกับโปรแกรมรับส่งไฟล์ข้อมูลฝั่งเซิฟเวอร์ทั่วไปในท้องตลาดได้ และมีความโดดเด่นในเรื่องการทำงานหลายฟังก์ชันพร้อมๆกัน หรือลักษณะการทำงานแบบ multi tasking

1.3 ขอบเขตของระบบงาน

1. โปรแกรมจะถูกสร้างเพื่อใช้งานบนวินโดว์ ซึ่งเป็นระบบปฏิบัติการที่มีใช้กันอย่างแพร่หลายในปัจจุบันนี้
2. มีฟังก์ชันมาตรฐานที่สามารถใช้ในการรับส่งไฟล์ จากเครื่องไคลเอนท์ไปยังเครื่องเซิฟเวอร์ได้ และสามารถทำงานหลายฟังก์ชัน ได้พร้อมๆกัน
3. ผู้ใช้สามารถใช้งานได้ง่าย ไม่ซับซ้อนยุ่งยาก

1.4 ขั้นตอนการดำเนินงาน

1. ขั้นตอนการสำรวจถึงลักษณะการใช้งาน โปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ ที่มีการใช้งานกันอยู่ในปัจจุบัน และรวบรวมศึกษาข้อมูลต่างๆที่เกี่ยวข้อง เพื่อนำมาใช้ออกแบบ และพัฒนาโปรแกรม
2. ขั้นตอนการศึกษา ความรู้พื้นฐานที่เกี่ยวข้องกับการรับส่งไฟล์ข้อมูล เช่น โพรโตคอล TCP / IP , Socket
3. ขั้นตอนการศึกษาโพรโตคอลรับส่งไฟล์ข้อมูล เพื่อให้ทราบถึงหลักการ และกระบวนการทำงานในการรับส่งไฟล์ข้อมูล
4. ขั้นตอนการศึกษาเครื่องมือที่ใช้ในการเขียนโปรแกรม โดยเลือกใช้โปรแกรมภาษา Visual C++ Version 6.0 เป็นเครื่องมือที่ใช้ในการเขียนโปรแกรม
5. ขั้นตอนการรวบรวมเอกสารและข้อมูลต่างๆ ที่ได้ทำการศึกษามาทั้งหมด เพื่อนำมาใช้ประกอบการเขียนโปรแกรม
6. ขั้นตอนการออกแบบโปรแกรม เพื่อหาขอบเขตในการเขียนโปรแกรม
7. ขั้นตอนการเขียนโปรแกรม ตามที่ได้ออกแบบไว้
8. ขั้นตอนการทดสอบ , แก้ไขปรับปรุงตัวโปรแกรม และทำการตกแต่งรูปแบบ เพื่อให้ตรงกับความต้องการ , มีประสิทธิภาพ และมีความสวยงาม
9. ขั้นตอนการทำเอกสารประกอบ และเอกสารอ้างอิง ในการทำปัญหาพิเศษ

1.5 การวางแผนงาน

1 มิ.ย. - 31 ก.ค.	ศึกษาและรวบรวมข้อมูล
1 ส.ค. - 15 ส.ค.	วิเคราะห์ และออกแบบโปรแกรม
16 ส.ค. - 30 ส.ค.	ศึกษาและเลือก Software Application ที่จะนำมาใช้
1 ก.ย. - 10 ต.ค.	ศึกษาเครื่องมือที่ใช้ในการเขียนโปรแกรม และทดลองเขียนโปรแกรมรับส่งไฟล์ข้อมูลอย่างง่ายๆ
11 ต.ค. - 20 ต.ค.	รวมรวมข้อมูลจัดทำเอกสารประกอบโครงงานปัญหาพิเศษ
21 ต.ค. - 25 ต.ค.	เตรียมการนำเสนอความก้าวหน้าของโครงงานปัญหาพิเศษ
1 พ.ย. - 30 ธ.ค.	เขียนโปรแกรมรับส่งไฟล์ข้อมูลตามที่ได้ออกแบบไว้
2 ม.ค. - 15 ม.ค.	ทดสอบและแก้ไขโปรแกรม
16 ม.ค. - 31 ม.ค.	ตกแต่งรูปแบบโปรแกรมให้สวยงาม
1 ก.พ. - 15 ก.พ.	สรุปโครงงานปัญหาพิเศษ
16 ก.พ. - 28 ก.พ.	จัดทำเอกสารประกอบโครงงานปัญหาพิเศษ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้ ความเข้าใจในการรับส่งไฟล์ข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง
2. โปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ที่สามารถนำไปใช้งานได้จริง และมีฟังก์ชันมาตรฐาน ในการรับส่งไฟล์กับโปรแกรมรับส่งไฟล์ข้อมูลฝั่งเซิฟเวอร์
3. เป็นแนวทางที่จะสามารถนำไปพัฒนาให้เกิดประโยชน์ และเหมาะสมกับเทคโนโลยีใหม่ๆ ที่จะเกิดขึ้นในอนาคตได้

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 TCP/IP

2.1.1 ลักษณะโดยรวมของโพรโตคอล TCP/IP

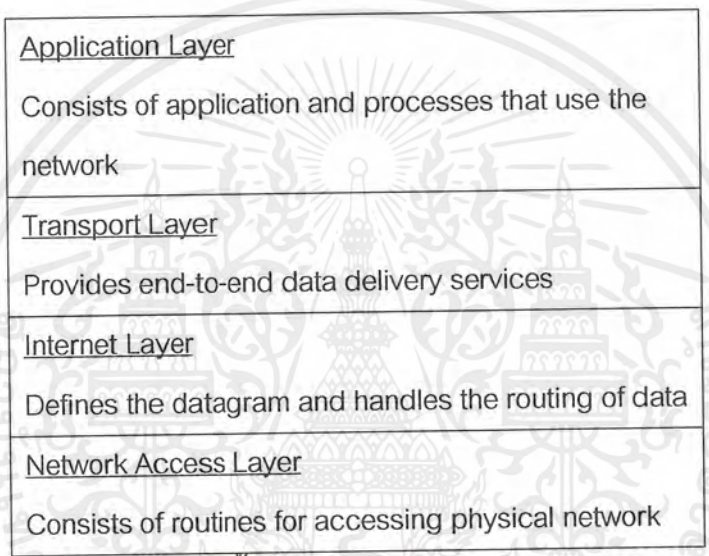
TCP/IP เป็นโพรโตคอลที่ใช้สื่อสารกันในระบบอินเตอร์เนต ซึ่งโพรโตคอลนี้ สามารถแบ่งได้เป็น 2 กลุ่มใหญ่ ได้แก่ Tranmission Control Protocal (TCP) และ Internet Protocol (IP) โดยโพรโตคอลTCP/IPนี้ เกิดจากการคิดค้นของกระทรวงกลาโหมของประเทศสหรัฐอเมริกา ที่ต้องการเชื่อมต่อเครื่องคอมพิวเตอร์ที่มีอยู่มากมายหลายยี่ห้อเข้าด้วยกันเป็นระบบเครือข่ายเดียว ปัจจุบันระบบสื่อสารข้อมูลโดยใช้ชุดโพรโตคอลTCP/IP เป็นที่ยอมรับแก่คนทั่วไป มีการใช้กันอย่างแพร่หลายจนกลายเป็นมาตรฐานสำหรับการติดต่อสื่อสารที่มีอยู่ในเครื่องคอมพิวเตอร์ทุกชนิด ทุกระบบ และทุกขนาด

Application Layer Consists of application programs that use the network
Presentation Layer Standardizes data presentation to the applications
Session Layer Manages sessions between application
Transport Layer Provide end-to-end error detection and correction
Network Layer Manages connections across the network for the upper layers
Data Link Layer Provides reliable data delivery across the physical link
Physical Layer Defines the physical characteristics of the network media

รูปที่ 2.1 การแบ่งชั้นการทำงานของ Network Protocol

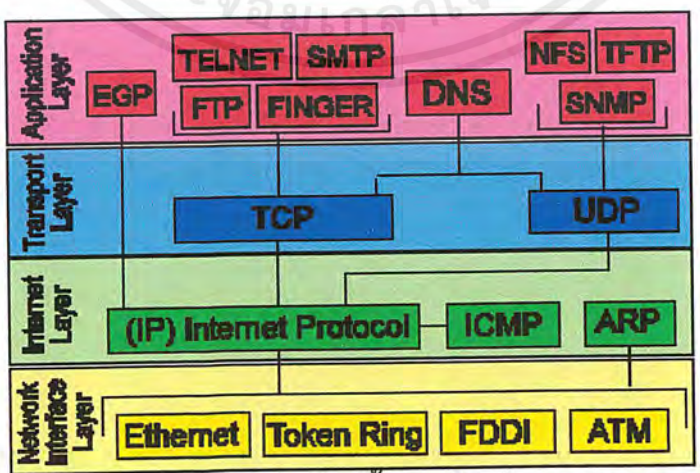
หลักการทำงานทั้งหมดของโพรโตคอลจะประกอบไปด้วยหลายๆ ชั้น ซึ่งนำมาวางซ้อนทับกันได้ออกมาในรูปที่เรียกว่าแสตกโพรโตคอล แต่ละชั้นจะมีหน้าที่การทำงานที่ชัดเจน และไม่เกี่ยวข้องกัน โดยแต่ละชั้นจะรู้เพียงวิธีการส่งข้อมูลไปยังชั้นอื่นๆ แต่ไม่รู้ถึงการทำงานข้างใน ได้มีการกำหนดมาตรฐานที่เป็นที่ยอมรับกันทั่วไป เรียกว่า Open System Interconnection (OSI) อ้างอิงโมเดลซึ่งทำการแบ่งการทำงานของ โพรโตคอลเครือข่ายออกเป็น 7 ชั้น ดังรูปที่ 2.1

แต่สำหรับโพรโตคอล TCP/IP จะมีการสร้างโครงสร้างขึ้นมาใหม่โดยแบ่งเป็น 4 ชั้น ดังรูปที่ 2.2



รูปที่ 2.2 การแบ่งชั้นการทำงานของโพรโตคอล TCP/IP

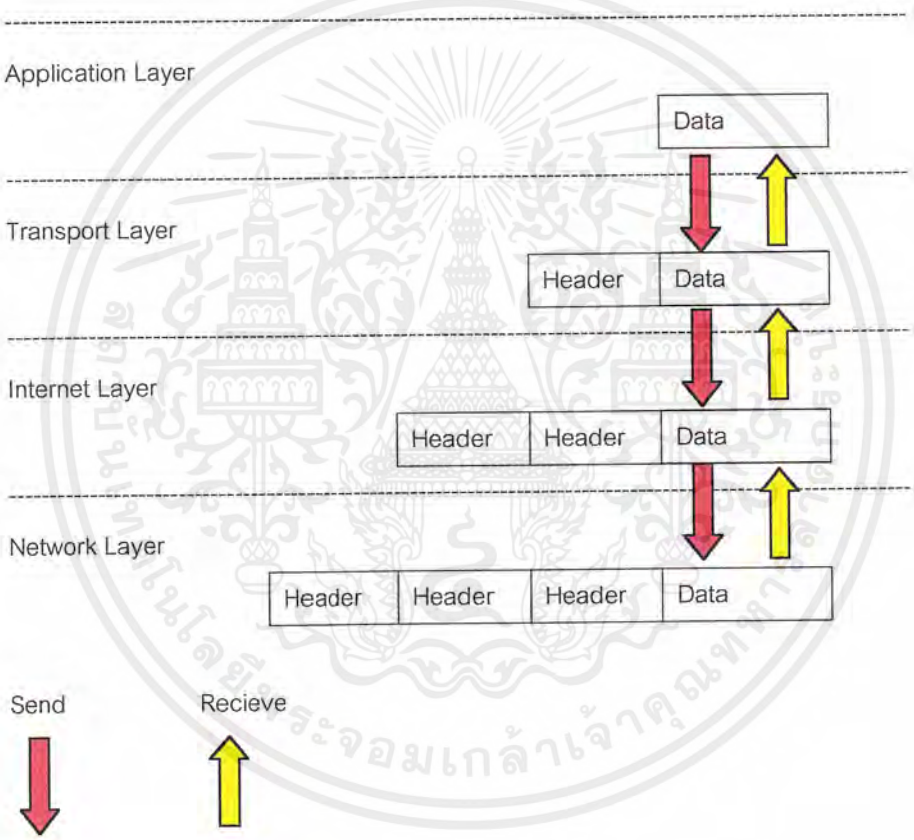
โดยในแต่ละชั้นจะประกอบด้วยการทำงานของส่วนต่างๆ ดังรูปที่ 2.3



รูปที่ 2.3 การทำงานในแต่ละชั้นของโพรโตคอล TCP/IP

ลักษณะการทำงานคือ ข้อมูลจะถูกส่งลงมาจากชั้นบนมายังชั้นล่าง ขณะที่ข้อมูลถูกส่งผ่านในแต่ละชั้นจะทำการเพิ่มข้อมูลควบคุมเข้าไปเพื่อให้การส่งข้อมูลถูกต้อง และเป็นการส่งพารามิเตอร์ที่จำเป็นไปให้กับชั้นของมันในเครื่องปลายทาง ข้อมูลเหล่านี้เรียกว่าส่วนเฮดเดอร์ ซึ่งแต่ละชั้นจะมีส่วนเฮดเดอร์ที่เป็นรูปแบบของตัวเอง การเพิ่มส่วนเฮดเดอร์เข้าไปรวมกับข้อมูลเรียกว่า Data Encapsulation โดยจะกระทำเช่นนี้ต่อไปจนกระทั่งส่งออกไปยังสายสื่อสาร เมื่อเครื่องปลายทางได้รับเฟรมข้อมูลก็จะถอดรหัสส่วนเฮดเดอร์ออก แล้วนำข้อมูลให้กับชั้นบนต่อไป กระทำเช่นนี้ไปเรื่อยๆ จนถึงชั้นแอพพลิเคชั่น

ลักษณะของ Data Encapsulation แสดงดังรูปที่ 2.4



รูปที่ 2.4 Data Encapsulation

2.1.2 ลักษณะการทำงานในแต่ละระดับชั้น

2.1.2.1 Network Access Layer ในชั้นนี้จะทำหน้าที่จัดส่งข้อมูลผ่านสายสัญญาณชนิดต่างๆ TCP/IP ไม่ได้กำหนดตัวกลาง หรือโพรโตคอลในระดับฟิสิกส์คัล คือ สามารถใช้ตัวกลางใดๆ สำหรับการส่งข้อมูลก็ได้ ถ้าอุปกรณ์ที่ทำการสื่อสารนั้นยังคงใช้ตัวกลางตัวเดียวกัน โดยการทำงานจะเป็นเช่นไรขึ้นอยู่กับชนิดของสายสัญญาณที่ใช้ สายแต่ละชนิดก็จะมีโพรโตคอล

ที่ใช้ในการควบคุมการทำงานที่แตกต่างกัน เช่น Ethernet ,FDDI ,Token ring และการเชื่อมโยงแบบอนุกรมหรือโมเด็ม ฯลฯ

2.1.2.2 Internet Layer ในชั้นนี้มีโพรโตคอลที่ทำงานอยู่ 2 ชนิด คือ Internet Protocol (IP) และ Internet Control Message Protocol (ICMP)

1) Internet Protocol (IP) เป็นโพรโตคอลชนิดคอนเน็คชันเลส คือ จะไม่ทำการส่งสัญญาณไปสถานีปลายทางก่อน แต่จะทำการส่งข้อมูลไปเลย โดยไม่คำนึงถึงว่ามันจะไปถึงปลายทางหรือไม่ และจะไม่มีการตรวจสอบความถูกต้องของข้อมูลที่ถูกส่งไปด้วย

โพรโตคอล IP นี้จะมีหน้าที่สำคัญ คือ

- สร้างเดต้าแกรม
- หาเส้นทาง เพื่อทำการส่งเดต้าแกรม
- แบ่ง และ ประกอบเดต้าแกรม

หน่วยที่ใช้ในการโอนย้ายข้อมูลนี้ เราเรียกว่า IP Datagram หรือ เรียกง่าย ๆ ว่า เดต้าแกรม ซึ่งจะคล้ายกับเฟรมที่อยู่ในระดับชั้น Network Access Layer แต่จะมีข้อแตกต่าง คือ ส่วนเฮดเดอร์ของเดต้าแกรม จะบรรจุตำแหน่ง IP Address แต่ส่วนเฮดเดอร์ของเฟรมจะบรรจุตำแหน่งฟิสิกส์คัล

2) internet Control Message Protocol (ICMP) ทำหน้าที่ในการส่งสัญญาณควบคุมต่างๆไปยังเครื่องปลายทาง โดยโพรโตคอลจะอาศัย IP Datagram ในการส่งสัญญาณควบคุมเหล่านั้น คำสั่งควบคุมที่สำคัญของ ICMP ได้แก่

- Flow Control
- Detecting Unreachable Destinations
- Redirecting Routes
- Checking Remote Hosts

2.1.2.3 Transport Layer ในชั้นนี้มีโพรโตคอลที่สำคัญอยู่ 2 ชนิด คือ

- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

ทั้ง 2 โพรโตคอลทำหน้าที่เหมือนกัน คือ เป็นตัวเชื่อมต่อการส่งข้อมูลระหว่าง ชั้นแอพพลิเคชั่น กับ ชั้นอินเตอร์เน็ต

1) User Datagram Protocol (UDP) เป็นโพรโตคอลแบบคอนเน็คชันเลส คือ ไม่มีการส่งสัญญาณกับเครื่องปลายทาง ไม่มีการตรวจสอบความถูกต้องของข้อมูลที่ได้รับมา ทำให้มีความเร็วในการทำงานสูง เหมาะที่จะใช้กับงานที่ส่งข้อมูลขนาดเล็กๆ และส่งบ่อยๆ

2) Transmission Control Protocol (TCP) เป็นโพรโตคอลแบบคอนเน็คชันโอเรียลเต็ด และมีการตรวจสอบความถูกต้องของข้อมูลที่ได้รับด้วย ทำให้การส่งเป็นไปอย่างน่าเชื่อถือ การส่งข้อมูลของ TCP จะถูกส่งทีละเซ็กเมนท์ โดยเครื่องปลายทางจะส่งสัญญาณตอบรับกลับมายังเครื่องต้นทางสำหรับทุกๆ เซ็กเมนท์ ที่มันได้รับ และตรวจสอบแล้วไม่พบข้อผิดพลาด ถ้าหากว่าเครื่องต้นทาง ไม่ได้รับสัญญาณตอบรับกลับมา ก็จะสันนิษฐานว่าเซ็กเมนท์ เกิดปัญหาและทำการส่งเซ็กเมนท์นั้นไปอีกครั้ง วิธีการนี้เราเรียกว่า Positive Acknowledgement with Re-transmission (PAR)

2.1.2.4 Application Layer เป็น Layer บนสุด ทำงานในส่วนแอพพลิเคชัน ซึ่งประกอบไปด้วยหลายโพรโตคอล เช่น ถ้าทำงานโดยใช้ TCP ได้แก่

- Network Terminal Protocol (Telnet)
- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)

ส่วน Protocol ที่ใช้ UDP ได้แก่

- Domain Nane Server (DNS)
- Routing Information Protocol (RIP)
- Network File System (NFS)

2.1.3 ความสัมพันธ์และการทำงานร่วมกันของโพรโตคอลในแต่ละชั้น

จากลำดับชั้นของชุดโพรโตคอล TCP/IP และแอพพลิเคชัน จะเห็นได้ว่าแอพพลิเคชัน แต่ละประเภทในชั้นแอพพลิเคชั่น เลือกใช้โพรโตคอลในชั้นทรานสปอร์ตแตกต่างกัน เช่น SMTP เลือกใช้ TCP ในขณะที่ BOOTP เลือกใช้ UDP สำหรับ TCP และ UDP จะเลือกใช้โพรโตคอล IP ในชั้นอินเตอร์เนต และจากนั้น ในชั้นเน็ตเวิรก์ ก็จะขึ้นกับว่าผู้ใช้งานเลือกใช้เครือข่ายประเภทใดในการส่งผ่านชุดโพรโตคอล TCP / IP

ในการส่งข้อมูลของแอพพลิเคชันต่าง ๆ ที่ทำงานอยู่ไปยังเครื่องปลายทางนั้น จะอาศัย TCP หรือไม่ก็ UDP ในการส่งข้อมูลไปให้ IP เพื่อทำการส่งไป นั่นคือ ข้อมูลของทุกๆโปรแกรมจะต้องส่งผ่าน IP ทั้งนั้น เราเรียกลักษณะเช่นนี้ว่าการ Multiplexing ส่วนกระบวนการย้อนกลับนั้น เราเรียกว่า Demultiplexing คือ ข้อมูลที่รับเข้ามาจะเข้าไปยัง IP แล้ว IP ก็จะแจกจ่ายข้อมูลเหล่านั้นขึ้นไปยังแอพพลิเคชัน ที่เป็นเจ้าของให้ถูกต้อง

ข้อมูลที่มาถึงจะต้องผ่านเลเยอร์ชั้นล่างขึ้นไปถึงชั้นบน ในชั้นอินเตอร์เนต นั้น IP จะใช้หมายเลขโพรโตคอล ในการส่งข้อมูลขึ้นไปยังชั้นทรานสปอร์ต และในทำนองเดียวกันชั้นทรานส-ปอร์ต ก็จะใช้หมายเลขพอร์ตในการส่งข้อมูลขึ้นไปยังชั้นแอพพลิเคชั่น

จากหลักการใช้ IP address โพรโตคอล และหมายเลขพอร์ต ข้างต้น จะเห็นได้ว่าต้องกำหนดให้ทุก ๆ เดต้าแกรมที่ส่งไปมาในอินเตอร์เนตนั้นมีปลายทางที่ไม่ซ้ำกัน แต่ละเดต้าแกรม จะมีปลายทางชี้ไปที่แอพพลิเคชันที่แน่นอนในเครื่องปลายทาง แต่วิธีการข้างต้นนี้ยังไม่เพียงพอ เนื่องจากเครื่องคอมพิวเตอร์เครื่องหนึ่งสามารถที่จะรันโปรแกรมเดียวกันพร้อม ๆ กันได้ เช่น ในเครื่องเซิฟเวอร์ระบบปฏิบัติการยูนิกซ์ ตัวหนึ่งอาจมีผู้ใช้หลายคนกำลังใช้โปรแกรมพูดคุยอยู่ ในกรณีอย่างนี้เราจะทราบได้อย่างไรว่าเดต้าแกรมที่เข้ามาเป็นของโปรแกรมพูดคุยตัวใด เพราะหมายเลข IP Address, โพรโตคอล และพอร์ตเหมือนกันหมด ปัญหานี้สามารถแก้ไขได้โดยใช้วิธีที่เรียกว่า Dynamically Allocated Port คือ ระบบจะไม่กำหนดเลขพอร์ตไว้ตายตัว แต่จะทำการกำหนดเลขพอร์ตให้กับโพรเซส ในขณะที่มันเริ่มต้นทำงาน ซึ่งเลขนี้จะต้องไม่ซ้ำกับเลขพอร์ตของงานอื่น ตัว-อย่างเช่น ถ้าเครื่องปลายทางเครื่องหนึ่งมีผู้ใช้ A และ B กำลังใช้โปรแกรม TELNET อยู่ เครื่องต้นทาง ซึ่งไม่จำเป็นจะต้องเป็นเครื่องเดียวกัน จะกำหนดหมายเลขพอร์ต ต้นทาง ให้เป็นค่าสุ่มค่าหนึ่งและเลขพอร์ตปลายทางให้เป็นหมายเลขพอร์ตมาตรฐาน ในตัวอย่างนี้ TELNET จะเท่ากับ 23 ด้วยวิธีการนี้เดต้าแกรม ของ A และ B จะมีเลขคู่ที่แตกต่างกัน เช่น ของ A อาจเป็น 3044,23 ส่วน B อาจจะเป็น 1027,23 ทำให้เดต้าแกรม สามารถเดินทางไปยังแอพพลิเคชันที่ถูก-ต้องได้

2.2 ซ็อกเก็ต

ซ็อกเก็ต คือ วิธีการสำหรับการสื่อสารกันระหว่างโปรแกรมไคลเอนท์กับ โปรแกรมเซิฟเวอร์ ในเครือข่าย ซ็อกเก็ตถูกกำหนดให้อยู่ในฐานะเอ็นท์พอยน์ของการคอนเน็คชั่น ซ็อกเก็ตถูกสร้างและใช้กับชุดของโปรแกรมร้องขอ หรือ socket API ซ็อกเก็ต สามารถถูกใช้เพื่อสื่อสารกันระหว่างโพรเซสที่อยู่ภายในเครื่องคอมพิวเตอร์เดียวกัน ในระบบแบบคอนเน็คชันเลส ของอินเตอร์เน็ต ที่เซิฟเวอร์ต้องจัดการกับการร้องขอจากไคลเอนท์หลาย ๆ ที่ และไม่มีการรักษาการเชื่อมต่อไว้นาน โดยทั่วไปซ็อกเก็ตจะสื่อสารข้อมูลกับซ็อกเก็ตตัวอื่น ๆ ที่กำลังทำงานอยู่ ในโดเมนของการติดต่อสื่อสารเดียวกันโดยใช้ IP ช่วย

2.2.1 **ชนิดของซ็อกเก็ต** ซ็อกเก็ตสามารถแบ่งได้เป็น 2 ชนิด คือ

1) **ซ็อกเก็ตแบบสตรีม** (stream) ทำงานกับข้อมูลที่จะส่งไปโดยไม่กำหนดขอบเขตของเรคคอร์ด หรือไม่เป็นเรคคอร์ด และข้อมูลที่ส่งเรียกว่า a stream of bytes ซึ่งสามารถส่งได้ทั้ง 2 ทาง นั่นคือ ส่งและรับในซ็อกเก็ตเดียวกัน และสตรีมจะรับประกันว่าข้อมูลถูกส่งถึงปลายทางอย่างถูกต้องโดยข้อมูลไม่สลับกันและไม่มีการส่งซ้ำซ้อน โดยสตรีมยังสามารถจัดการส่งและรับข้อมูลขนาดใหญ่ได้ดีอีกด้วยเพราะโพรโตคอลในชั้นเน็ตเวิร์ก อาจแตกข้อมูลให้อยู่ในขนาดที่สมควร

การสื่อสารข้อมูลแบบสตรีมนั้นเป็นการสื่อสารข้อมูลแบบคอนเน็คชันโอเรียลเด็ด สตรีมนั้นจะต้องมีการเชื่อมต่อกันอย่างชัดเจน ถ้า ซ็อกเก็ต A ต้องการเชื่อมต่อกับซ็อกเก็ต B แล้ว ซ็อกเก็ต A จะต้องมีการร้องขอเชื่อมต่อไปยัง ซ็อกเก็ต B แล้ว ซ็อกเก็ต B จะต้องยอมรับหรือปฏิเสธการร้องขอนั้น ๆ เหมือนกับโทรศัพท์ ซึ่งจะต้องมีผู้เรียกแล้วผู้ที่รับสายจึงจะคุยกันได้ และการสนทนาจะไม่มีเสียงซ้ำกันถ้าผู้พูดไม่ได้พูดซ้ำ ลำดับของเสียงที่พูดจะตรงกับลำดับของเสียงที่ผู้ฟังได้ยิน และที่สำคัญที่สุดคือเสียงที่ผู้พูดพูดไปนั้นจะถูกส่งไปถึงผู้รับแน่นอน ด้วยเหตุนี้จึงทำให้ซ็อกเก็ตแบบสตรีม นี้ เป็นที่นิยมมากกว่าซ็อกเก็ตแบบเดต้าแกรม

2) **ซ็อกเก็ตแบบดาต้าแกรม** ซ็อกเก็ตแบบนี้สนับสนุนการส่งข้อมูลสองทาง โดยไม่รับประกันว่าข้อมูลจะถูกส่งไปถึงอย่างถูกต้องหรือไม่และข้อมูลอาจมีการสลับกันหรือซ้ำซ้อนแต่ขนาดของเรคอร์ดจะได้รับการปกป้องรักษา แม้ว่าขนาดของเรคคอร์ดจะใหญ่กว่าขนาดที่ผู้รับกำหนด

การสื่อสารข้อมูลแบบดาต้าแกรม เป็นการสื่อสารข้อมูลแบบคอนเน็คชั่นเลส ไม่จำเป็นต้องสร้างการเชื่อมต่อโดยชัดเจน เพียงส่งแมสเสสเดต้าแกรมไปยังซ็อกเก็ต ที่กำหนดก็จะสามารถสื่อสารข้อมูลกันได้แล้ว และซ็อกเก็ตแบบเดต้าแกรมยังมีความสามารถในการส่งข้อมูลแบบกระจาย (Broadcasting Message)

2.2.2 **พอร์ตและแอดเดรสของซ็อกเก็ต**

1) **พอร์ต** เป็นเลขที่ใช้กำหนดค่าให้กับซ็อกเก็ตแต่ละตัว โดยจะไม่มีเลขที่ซ้ำกันเกิดขึ้นพร้อมกัน ดังนั้นซ็อกเก็ตของแอปพลิเคชั่นจะมีค่าของพอร์ตที่ไม่ซ้ำกันเลย ทำให้สามารถมีแอปพลิเคชั่นที่ทำงานกับซ็อกเก็ตได้หลายตัวพร้อมกัน

2) **แอดเดรสของซ็อกเก็ต** ซ็อกเก็ตแต่ละตัวจะติดต่อกับ IP Address ในเครือข่ายโดยทั่วไปแอดเดรสจะเปรียบเสมือนชื่อเครื่องเช่น kmitl.ac.th หรือ "161.246.10.21"

2.2.3 **การทำงานระหว่างซ็อกเก็ต และ ชั้นแอพพลิเคชั่น**

ดังที่ได้อธิบายไปแล้วว่า สตรีมของเดต้าถูกแบ่งเป็นเดต้าแกรมอย่างไร ส่งไปยังคอมพิวเตอร์อีกเครื่องอย่างไร และเดต้าแกรมถูกนำไปรวมกันอย่างไร อย่างไรก็ตามต้องมีวิธีที่จะเปิดการเชื่อมต่อไปยังคอมพิวเตอร์ที่ระบุ และบอกคอมพิวเตอร์ว่า คุณต้องการไฟล์ใด และ ควบคุมการถ่ายโอนของไฟล์ งานเหล่านี้จะถูกทำโดย แอพพลิเคชันโพรโตคอล ซึ่งรันอยู่บนโพรโตคอล TCP/IP นั่นก็คือ เมื่อแอพพลิเคชันโพรโตคอลต้องการส่งข้อความ ก็จะให้ข้อความนั้นแก่ TCP เพื่อให้ TCP ตรวจสอบให้แน่ใจว่าข้อความนั้นถึงจุดหมายปลายทาง เนื่องจาก TCP และ IP ดูแลในเรื่องรายละเอียดทางด้านเครือข่ายทั้งหมด แอพพลิเคชันโพรโตคอลสามารถปฏิบัติต่อเครือข่ายการเชื่อมต่อ(network connection) เสมือนว่าเป็นเพียงไบต์สตรีม ธรรมดา ๆ เหมือนกับเทอร์มินัลหรือสายโทรศัพท์ ก่อนที่จะลงลึกรายละเอียดเกี่ยวกับ แอพพลิเคชันโปรแกรมจะต้องอธิบายก่อนว่า จะหาแอพพลิเคชันได้อย่างไร สมมติว่าต้องการจะส่งไฟล์ไปที่คอมพิวเตอร์ที่มี Internet Address 128.6.4.7 จะไม่ใช้เพียงแค่ Internet Address อย่างเดียวเท่านั้นในการติดต่อไปยังเครื่องFTPเซิฟเวอร์ที่ปลายอีกข้างหนึ่ง ตามปกติแล้วโปรแกรมเน็ตเวิร์กจะถูกทำขึ้นมาพิเศษเพื่องานนั้นๆ โดยเฉพาะ ระบบส่วนใหญ่มีโปรแกรมแยกต่างหากไว้สำหรับจัดการกับการโอนถ่ายไฟล์ จัดการกับการล็อกอินจากเครื่องเทอร์มินัลระยะไกล และสำหรับจัดการกับเมล์ ฯลฯ เมื่อทำการเชื่อมต่อไปที่ 128.6.4.7 จะต้องระบุว่าต้องการจะติดต่อกับ FTP เซิฟเวอร์ใด โดยแต่ละเซิฟเวอร์จะมีซ็อกเก็ตที่ทำหน้าที่ทวนอีกครั้งว่า TCP ใช้หมายเลขพอร์ตใดในการเก็บรายละเอียดต่าง ๆ ของแต่ละครั้งในการสนทนา หมายเลขพอร์ตที่เฉพาะเจาะจงจะถูกกำหนดให้กับโปรแกรมที่นั่งรอการร้องขออยู่ ตัวอย่างเช่น ถ้าต้องการส่งไฟล์ เมื่อเริ่มต้นโปรแกรมรับส่งไฟล์ข้อมูลทางฝั่งไคลเอนท์ โปรแกรมจะเปิดคอนเน็คชันโดยใช้เลขสุ่ม เช่น 1234 สำหรับหมายเลขพอร์ตที่ฝั่งไคลเอนท์ และกำหนดพอร์ตหมายเลข 21 ซึ่งเป็นพอร์ตมาตรฐานสำหรับเครื่อง FTP เซิฟเวอร์ ให้กับปลายทางอีกฝั่งหนึ่ง ในที่นี้จะมีสองโปรแกรมที่ต่างกันมาเกี่ยวข้อง โปรแกรม FTP ไคลเอนท์จะถูกรันบนฝั่งต้นทาง และโปรแกรม FTP เซิฟเวอร์ จะถูกรันบนเครื่องฝั่งปลายทาง

โปรแกรมที่ออกแบบมาเพื่อตอบรับคำสั่งจากเทอร์มินอลเครื่องต้นทางและส่งคำสั่งต่อไปยังปลายทางอื่น โปรแกรมที่ติดต่อด้วยบนอีกเครื่องหนึ่งก็คือ FTPเซิฟเวอร์ ซึ่งถูกออกแบบมาเพื่อ

ตอบรับคำสั่งจากการเชื่อมต่อเครือข่าย แทนที่จะตอบรับจากเครื่องเทอร์มินัลที่ปฏิบัติต่อกันอยู่ หมายเลขพอร์ตอย่างเป็นทางการสำหรับแต่ละโปรแกรมจะถูกอธิบายด้วย เซตของเลข 4 จำนวน คือ Internet Address ของแต่ละเครื่องต้นทางและปลายทาง และ หมายเลข TCP พอร์ต ของแต่ละเครื่องต้นทางและปลายทาง แต่ละเดต้าแกรมจะมีเลขทั้ง 4 นี้ ซึ่ง Internet Address อยู่ในส่วนเฮดเดอร์ของ IP และ หมายเลข TCP พอร์ต อยู่ในส่วนเฮดเดอร์ของ TCP ไม่มี 2 คอนเนิกชันใด ๆ ที่สามารถมีเซตของเลขเหมือนกันได้ อย่างไรก็ตาม มีแค่เพียงตัวเลข ตัวหนึ่งที่แตกต่างกันก็พอแล้ว ตัวอย่างเช่น เป็นไปได้ที่จะมีผู้ใช้ 2 คนที่แตกต่างกันบนคอมพิวเตอร์เครื่องเดียวกันที่กำลังส่งไฟล์ไปยังเครื่องเดียวกัน คอนเน็กชันจะมีพารามิเตอร์ดังนี้ คอนเน็กชัน ที่ 1 คือ 128.6.4.194, 128.6.4.7, 1234, 21 และพารามิเตอร์ของคอนเน็กชัน ที่ 2 คือ 128.6.4.194, 128.6.4.7, 1235 ,21 เนื่องจากใช้เครื่องคอมพิวเตอร์เครื่องเดียวกัน Internet Address จึงเหมือนกัน และเนื่องจากผู้ใช้ทั้งสองกำลังถ่ายโอนไฟล์ปลายทางข้างหนึ่งของคอนเน็กชันจึงเกี่ยวข้องกับหมายเลขพอร์ตที่เป็นมาตรฐานสำหรับ FTP สิ่งเดียวที่แตกต่างก็คือ หมายเลขพอร์ตสำหรับโปรแกรมที่ผู้ใช้รันอยู่ โดยปกติแล้ว อย่างน้อยฝั่งหนึ่งฝั่งใดของการเชื่อมต่อ จะขอให้เน็ตเวิร์กซอฟท์แวร์ กำหนดหมายเลขพอร์ตที่รับประกันได้ว่าจะไม่เหมือนใคร ซึ่งตามธรรมดาจะเป็นฝั่งผู้ใช้เนื่องจากเครื่องเซิฟเวอร์ต้องใช้ เลขที่เป็นมาตรฐานและรู้จักกันดี เมื่อได้ทราบถึงวิธีการเปิดการเชื่อมต่อแล้ว ต่อไปนี้จะกล่าวถึงโปรแกรมแอพพลิเคชัน ดังที่ได้กล่าวไปข้างต้น เมื่อ TCP ได้เปิดการเชื่อมต่อแล้ว ก็จะเริ่มมีสิ่งที่เป็นเสมือนสายไว้คอยติดต่อ ส่วนที่ยุ่งยากก็เป็นหน้าที่ของ TCP และ IP อย่างไรก็ตามเราก็ยังจำเป็นต้องมีการตกลงกันเรื่องสิ่งที่จะส่งผ่านคอนเน็กชันซึ่งก็คือ ข้อตกลงในเรื่องเซตของคำสั่งที่แอพพลิเคชันสามารถเข้าใจ และรูปแบบว่าจะส่งสิ่งเหล่านั้นอย่างไร โดยทั่วไปสิ่งที่จะส่งก็คือคำสั่งและเดต้ารวมกัน ใช้อรรถอธิบายเพื่อให้แตกต่าง ตัวอย่างเช่น เมล์โพรโตคอลที่ทำงานดังนี้ โปรแกรมเมล์ฝั่งไคลเอนท์เปิดการเชื่อมต่อไปยังเมล์เซิฟเวอร์ที่ปลายทางอีกฝั่งหนึ่ง โปรแกรมฝั่งไคลเอนท์ให้ชื่อเครื่อง , ผู้ส่งข้อความ และ ผู้รับ จากนั้นโปรแกรมจึงส่งคำสั่งที่บอกว่ากำลังจะเริ่มส่วนที่เป็นข้อความ ณ จุดนั้น ปลายทางอีกฝั่งหนึ่งจะหยุดปฏิบัติและเข้าใจว่าสิ่งที่ได้รับว่าเป็นคำสั่ง เมื่อถึงจุดสิ้นสุดของข้อความเครื่องหมายพิเศษ เครื่องหมาย จุด "." ในแถวแรกจะถูกส่งไปยังเมล์เซิฟเวอร์ หลังจากนั้นปลายทางทั้งสองจะเข้าใจว่าโปรแกรมกำลังจะส่งคำสั่งต่อ วิธีนี้เป็นวิธีที่ง่ายที่สุด และเป็นวิธีที่แอพพลิเคชัน ส่วนใหญ่ใช้กัน

การรับส่งไฟล์ข้อมูลค่อนข้างจะยุ่งยากกว่าเมล์ การรับส่งไฟล์ข้อมูลจะเกี่ยวข้องกับ 2 การเชื่อมต่อที่แตกต่างกัน โดยการเชื่อมต่อแรกนั้นเหมือนกับเมล์ นั่นคือ โปรแกรมของผู้ใช้ส่งคำสั่งไป แต่เมื่อคำสั่งที่บอกว่าต้องการจะส่งเดต้าถูกส่งไป การเชื่อมต่อที่สองจะถูกสร้างขึ้นเพื่อใช้สำหรับรับส่งเดต้าโดยเฉพาะ สามารถเป็นไปได้ที่จะส่งเดต้าบนการเชื่อมต่อเดียวกับที่ใช้รับส่งคำสั่งอย่างที่เมล์ทำ แต่ในการรับส่งไฟล์บ่อยครั้งที่ใช้เวลานาน ผู้ออกแบบโปรแกรมรับส่งไฟล์ข้อมูลต้องการ

ปล่อยให้ผู้ใช้สามารถส่งคำสั่งต่อได้เลยในขณะที่กำลังมีการถ่ายโอนไฟล์อยู่ ตัวอย่างเช่น ผู้ใช้อาจจะมีข้อซักถาม หรือต้องการยกเลิกการถ่ายโอน ด้วยเหตุนี้ผู้ออกแบบรู้สึกว่าเป็นการดีที่สุดที่จะใช้อีกการเชื่อมต่อสำหรับรับส่งเดต้าต่างหาก และปล่อยการเชื่อมต่อเดิมไว้สำหรับรับส่งคำสั่ง และยังสามารถเป็นไปได้ที่จะเปิดการเชื่อมต่อสำหรับส่งคำสั่งไปยังคอมพิวเตอร์สองเครื่องที่ต่างกัน และบอกทั้งสองเครื่องให้ส่งไฟล์จากเครื่องหนึ่งไปยังอีกเครื่องหนึ่ง

ระบบการเชื่อมต่อเทอร์มินัลระยะไกลนั้นใช้อีกกลไกหนึ่ง สำหรับการล็อกอินระยะไกล มีเพียงแค่หนึ่งการเชื่อมต่อเท่านั้น โดยปกติจะส่งเดต้าเมื่อจำเป็น เช่น เมื่อส่งคำสั่งเพื่อที่จะ terminal type หรือเปลี่ยนโหมด จะใช้อักษรพิเศษเพื่อบ่งชี้ว่าตัวอักษรต่อไปคือคำสั่ง ถ้าผู้ใช้ต้องการจะพิมพ์อักษรพิเศษนั้นเป็นเดต้าต้องพิมพ์สองตัวส่งไป

อย่างไรก็ตาม มีระเบียบแบบแผนที่แอพพลิเคชันใช้เหมือนกันอยู่สองอย่าง ที่จะกล่าวถึงคือ ข้อแรก TCP/IP ตั้งใจให้สามารถใช้ได้บนคอมพิวเตอร์ใดๆ จากปัญหาที่พบว่าทุกคอมพิวเตอร์จะต้องตกลงในเรื่องว่าจะแสดงเดต้าอย่างไร เรื่องการมีความแตกต่างในเรื่องรหัสตัวอักษร เช่น ASCII กับ EBCDIC เรื่องจุดสิ้นสุดของบรรทัด และข้อแตกต่างในเรื่องที่ว่าเครื่องเทอร์มินัลคาดหวังว่าตัวอักษรจะถูกส่งมาแยกเป็นตัวหรือเป็นบรรทัดต่อครั้ง แต่โพรโตคอล TCP/IP ไม่สนใจเกี่ยวกับรูปแบบการนำเสนอ โพรโตคอลTCP/IPจะส่งเดต้าในรูปของเลขฐานแปด อย่างไรก็ตามโปรแกรมทั้งสองฝั่งจะต้องทำการตกลงว่าจะแปลงเลขฐานแปดอย่างไร จึงมีการกำหนด RFC สำหรับแต่ละแอพพลิเคชันเพื่อระบุรูปแบบการนำเสนอมาตรฐาน สำหรับแอพพลิเคชันนั้น ๆ ปกติจะเป็น net ASCII ซึ่งใช้ ตัวอักษร ASCII จุดสิ้นสุดของบรรทัดใช้แบบ carriage return ตามด้วย line feed สำหรับการล็อกอินจากระยะไกลก็มีนิยามของเทอร์มินัลมาตรฐานเช่นกัน ซึ่งก็คือ half-duplex terminal ที่มีการแอ็คโค่บนเครื่องท้องถิ่น แอพพลิเคชันส่วนใหญ่ก็เช่นกันจะกำหนดบทบัญญัติสำหรับคอมพิวเตอร์สองเครื่องในข้อตกลงเรื่องรูปแบบการนำเสนอแบบอื่น ๆ ที่อาจพบว่าสะดวก ตัวอย่างเช่น PDP-10 เป็น แบบ 36-bit words ก็มีทางที่ PDP-10's สองเครื่องจะตกลงที่จะส่ง แบบ 36-bit binary file คล้าย ๆ กัน สองระบบที่ชอบแบบ full-duplex ก็สามารถตกลงกันเองได้ อย่างไรก็ตามแต่ละแอพพลิเคชันควรมีรูปแบบการนำเสนอมาตรฐานที่ทุกเครื่องสามารถรองรับได้

2.3 Multithread

2.3.1 **โพรเซส** คือ โปรแกรมที่กำลังทำงานอยู่ (Program → Execution → Process) ซึ่งทุกๆโพรเซสมีองค์ประกอบหลักดังนี้

1) Data ข้อมูลที่โปรแกรมต้องการซึ่งอยู่บนหน่วยความจำ ข้อมูลนี้อาจอยู่ในโพรเซสใดโพรเซสหนึ่งหรืออาจใช้ร่วมกันกับโพรเซสอื่นๆก็ได้
2) Program Counter (PC) ทำหน้าที่เก็บแอดเดรสของ machine instruction ที่จะถูกประมวลผล(execute) เป็นคำสั่งถัดไป
3) Stack เนื้อที่หน่วยความจำที่ถูกใช้งานโดยโพรเซส

โดยหน้าที่ของซีพียูคือทำการอ่าน machine instruction จากหน่วยความจำแล้วทำการตีความ และสุดท้ายทำการประมวลผล โดยอาจเรียกซีพียูนี้ว่าเป็น hardware interpreter ได้

2.3.1.1 Process Control Block (PCB) เป็นเนื้อที่ในหน่วยความจำที่ระบบปฏิบัติการกำหนดขึ้นเพื่อใช้เก็บข้อมูลที่สำคัญของโพรเซสหนึ่งๆ โดยข้อมูลเหล่านี้ได้แก่

- สถานะของโพรเซสที่เป็นอยู่ในปัจจุบัน
- หมายเลขประจำตัวของโพรเซส
- ลำดับความสำคัญของโพรเซส
- พอยน์เตอร์ชี้ไปยังตำแหน่งที่อยู่ของโพรเซสในหน่วยความจำ
- พอยน์เตอร์ชี้ไปยังทรัพยากรต่างๆที่โพรเซสครอบครองอยู่
- พื้นที่ที่เก็บค่าของรีจีสเตอร์ (register save area)

PCB เป็นศูนย์กลางในการเก็บข้อมูลที่สำคัญของโพรเซสที่ระบบปฏิบัติการต้องการ เมื่อระบบปฏิบัติการต้องการให้โพรเซสอื่นได้เข้าใช้งานซีพียูบ้าง ระบบปฏิบัติการจะเก็บข้อมูลต่างๆของโพรเซสเดิมไว้ใน PCB เพื่อที่จะนำกลับมาใช้ใหม่อีกครั้ง เมื่อโพรเซสนี้ได้กลับมาใช้หรือครอบครอง ซีพียู

2.3.1.2 **สถานะของโพรเซส**

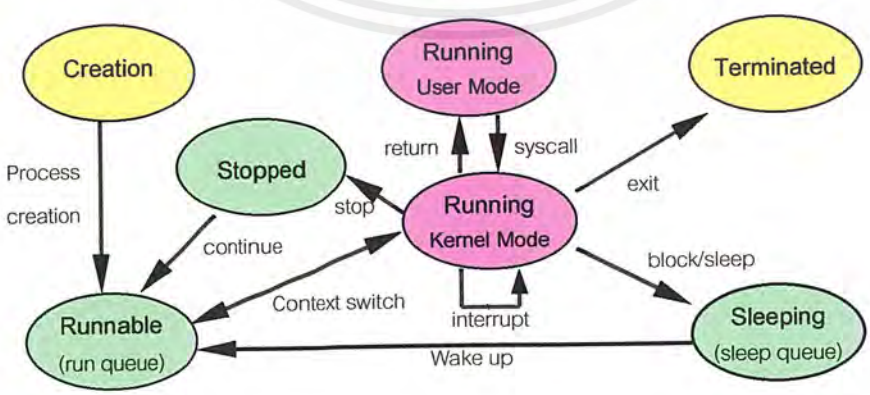
โพรเซสต่างๆในระบบจะต้องอยู่ในสถานะการทำงานสถานะใดสถานะหนึ่งดังต่อไปนี้

ตารางที่ 2.1 แสดงสถานะการทำงานของโพรเซส

สถานะ	คำอธิบาย
New	เป็นสถานะขณะโพรเซสถูกสร้างขึ้นใหม่
Ready	เป็นสถานะที่โพรเซสพร้อมที่จะใช้งานซีพียูทันทีที่ระบบปฏิบัติการ มอบหมายให้ ในสถานะนี้ไม่มีการรันของโพรเซส (โพรเซสหยุดนิ่ง) ซึ่งภายใน Ready State จะมีคิวของโพรเซสอยู่เพื่อใช้จัดลำดับของการเข้าใช้ซีพียูของโพรเซสต่างๆ
Running	เป็นสถานะที่โพรเซสกำลังครอบครองซีพียูอยู่มีการรันของโพรเซสจริงๆ เพราะโพรเซสใช้ซีพียูในการประมวลผลคำสั่งหรือโค้ดโปรแกรมของโพรเซสนั้น
Waiting, Suspended	เป็นสถานะที่โพรเซสหยุดรอเหตุการณ์ใดเหตุการณ์หนึ่งให้เกิดขึ้น เช่น การรอการทำงานของ I/O เป็นต้น ซึ่งโพรเซสไม่จำเป็นต้องใช้ซีพียูและยังไม่พร้อมที่จะครอบครองซีพียู
Terminate	เป็นสถานะที่โพรเซสไม่มีการทำงานใดๆเกิดขึ้น ไม่มีการรอใช้ซีพียูหรือเหตุการณ์ใดๆให้เกิดขึ้น โพรเซสจบการทำงานอย่างสมบูรณ์

2.3.1.3 **วงจรชีวิตของโพรเซส** (Process Life Cycle)

โพรเซสมีวงจรชีวิตโดยการเปลี่ยนแปลงไปยังสถานะต่างๆ ในแต่ละเวลาโพรเซสจะถูกกำหนดให้อยู่ในสถานะใดสถานะหนึ่งเพียงสถานะเดียว ซึ่งโพรเซสสามารถถูกสร้างขึ้นและจบการทำงานเพียงครั้งเดียว แต่สามารถที่จะเปลี่ยนไปในสถานะอื่นๆ กี่ครั้งก็ได้ในช่วงวงจรชีวิต



รูปที่ 2.5 วงจรชีวิตของโพรเซส

2.3.1.4 Context Switching

จากการศึกษาในเรื่องสถานะของโพรเซส ถ้ามีหลายโพรเซสอยู่ในสถานะ Ready เพื่อรอใช้ซีพียูเมื่อซีพียูว่างโพรเซสที่อยู่ต้นคิวในสถานะ Ready จะได้ครอบครองซีพียูแล้วเปลี่ยนไปสู่สถานะ Running โพรเซสอยู่ในสถานะนี้ได้ไม่เกินเวลาที่ระบบปฏิบัติการกำหนด (quantum time) ถ้าโพรเซสต้องรอเหตุการณ์บางอย่างโดยไม่มีความจำเป็นต้องใช้ซีพียูโพรเซสนั้นจะถูกย้ายไปอยู่ในสถานะ Waiting แต่ถ้าโพรเซสอยู่ในสถานะ Running จนครบกำหนดเวลา โพรเซสนั้นจะต้องไปรอใช้งานซีพียูที่สถานะ Ready เพื่อเปิดโอกาสให้โพรเซสอื่นได้มีโอกาสใช้งานซีพียูบ้าง ไม่ว่าโพรเซสจะปลดปล่อยซีพียูเนื่องจากหมดเวลา หรือมีการรอเหตุการณ์บางอย่างซีพียูจะต้องถูกครอบครองโดยโพรเซสอื่นที่อยู่ในสถานะ Ready นั่นหมายความว่าซีพียูต้องเปลี่ยนการทำงานจากงานหนึ่งไปเป็นอีกงานหนึ่ง เหตุการณ์นี้เรียกว่าการเปลี่ยนสถานะของซีพียู(Context Switching)

การเปลี่ยนสถานะของซีพียูถือว่าเป็นโอเวอร์เฮดของระบบ โอเวอร์เฮดในที่นี้หมายถึง งานที่ระบบจำเป็นต้องกระทำเพื่อจุดประสงค์บางอย่าง ระบบต้องใช้เวลาในการทำงานเหล่านี้ ทำให้เสียเวลาในการทำงานของโพรเซสของผู้ใช้ไปด้วย ดังนั้นในระบบจึงไม่ควรมีโอเวอร์เฮดของระบบมากเกินไป สำหรับการทำการเปลี่ยนสถานะของซีพียูจะเสียโอเวอร์เฮดสำหรับงานหลัก 3 งานดังนี้

1) เก็บค่ารีจิสเตอร์และสถานะของโพรเซสไว้ใน PCB ของโพรเซสที่กำลังปล่อยซีพียู
2) คัดเลือกโพรเซสที่อยู่ในสถานะ Ready ให้เข้าใช้งานซีพียู
3) โหลดค่ารีจิสเตอร์และสถานะของโพรเซสจาก PCB ของโพรเซสใหม่

2.3.2 Thread จากที่ทราบแล้วว่าการทำงานของโพรเซสนั้นมีโอเวอร์เฮดมาก เช่น

- ต้องมีแอดเดรสสเปซ หรือเนื้อที่หน่วยความจำเป็นของตนเอง
- ต้องใช้ IPC (Interprocess Communication) เมื่อต้องการติดต่อกันระหว่างโพรเซส
- ต้องทำงานผ่านการเรียกผ่านระบบปฏิบัติการ (system call) ซึ่งไม่ให้มีการทำงานข้ามขอบเขตการป้องกันหรือกล่าวอีกอย่างว่าไม่ให้โพรเซสทำงานก้าวก่ายการทำงานซึ่งกันและกัน

โพรเซสเป็น Compound Entity ซึ่งสามารถแบ่งได้เป็น 2 ส่วนใหญ่คือ กลุ่มของ Thread และกลุ่มของ Resource ในแต่ละ Thread จะมีองค์ประกอบส่วนตัวเช่น program counter (PC)

สแตก และค่ารีจิสเตอร์ เป็นของตนเอง Thread มีหลายประเภท ซึ่งแต่ละประเภทก็จะมีคุณสมบัติและวิธีการใช้งานที่แตกต่างกัน ในหัวข้อนี้จะอธิบายถึง Thread ใน 3 ประเภทที่สำคัญ

1) Kernel Threads เป็น Thread ของตัว kernel เองซึ่งไม่เกี่ยวข้องกับโพรเซสของผู้ใช้ จะถูกสร้างและถูกทำลายโดยตัว kernel เอง
2) Lightweight Processes (LWP) เป็น thread ของโพรเซสที่มี kernel ช่วยในการทำงาน โดยทำงานในระดับที่สูงกว่า kernel thread ดังนั้นระบบที่รองรับ LWP ต้องรองรับการทำงานของ kernel thread เสียก่อน ทุกๆโพรเซสอาจมี LWP มากกว่า 1 ตัวก็ได้ โดยแต่ละ LWP จะมี kernel thread ช่วยในการทำงาน
3) User Threads เป็น thread ที่อยู่ในระดับผู้ใช้ทั้งหมด ตัว kernel จะไม่รับรู้เกี่ยวกับ user thread ใดๆทั้งสิ้น ซึ่งการทำงานของ thread ในระดับนี้ต้องมี thread library เข้ามาเกี่ยวข้อง โดยใน library จะเตรียมฟังก์ชั่นสำหรับการสร้าง การเข้าจังหวะการทำงาน การจัดตารางเวลาการทำงานของ thread และการจัดการอื่นๆเกี่ยวกับ Thread โดยปราศจากการช่วยเหลือใดๆจาก kernel การโต้ตอบกันระหว่าง thread จะไม่เกี่ยวข้องกับ kernel ดังนั้น user thread จึงทำงานได้เร็วมาก ซึ่งการทำ thread นี้เรียกว่า synchronous programming paradigm นั่นคือ เมื่อถึงงานที่ต้องการแล้วค่อยสร้าง thread ขึ้นมา

ไมโครซอฟท์วินโดว์เป็นระบบปฏิบัติการตัวแรกๆที่ใช้หลักการของ multithread โดยอาจถือว่า Message Handler หรือ Driven Function นั้นเป็น Thread หนึ่งๆ โดยส่วนใหญ่ระบบปฏิบัติการที่รองรับการทำงานของ Thread จะมี Thread Library ซึ่งภายในจะมีโค้ดในส่วนของการทำ context switching ไว้ โดยมีการทำงานที่คล้ายกับ context switching ในระบบปฏิบัติการ

2.3.3 Multithreading บน MS-Windows

ความสามารถประการหนึ่งของ Windows คือสามารถจัดการกับหลายๆ โพรเซสที่ทำงานอยู่พร้อมๆ กันได้อย่างมีประสิทธิภาพ Windows จัดการกับการทำงานของหลายโพรเซสด้วยวิธี preemptive-multitasking โดยจะแตกโพรเซสใหญ่ให้เป็นโพรเซสย่อยที่เรียกว่า thread และให้แต่ละ thread แบ่งกันใช้ซีพียูtime ซึ่งขึ้นอยู่กับค่า priortity ส่วนภายในตัว Windows จะมีซอฟท์แวร์ของระบบที่ชื่อ ตัวจัดการตารางการทำงานของระบบ (system-scheduler) ซึ่งทำหน้าที่จัดสรรให้แต่ละ thread เข้าใช้งานซีพียูและจะเป็นผู้คัดเลือกว่า thread ใดและเมื่อใดที่ควรเข้าใช้ซีพียูการเข้าใช้ซีพียูของ thread ถูกกำหนดโดยค่า priority โดย thread ที่มี priority ต่ำจะคอยจนกระทั่ง thread ที่มี priority สูงทำงานเสร็จ ถ้าระบบเป็นแบบมัลติโพรเซสเซอร์ ตัวจัดการตารางการทำงานของระบบจะย้าย thread ที่กำลังคอยการทำงานอยู่ไปยังซีพียูตัวอื่นที่ว่างอยู่เพื่อให้การทำงานของซีพียูต่างๆมีความสมดุล

การเขียนโปรแกรมแบบ Multithread บน Windows NT สามารถทำได้ 2 วิธี

1. Win32 APIs
2. Microsoft Foundation Class Library (MFC)

เหตุผลใหญ่สำหรับการใช้ multithread คือยอมให้โพรเซสทำหลายๆงาน (multiple task) ในเวลาเดียวกัน ตัวอย่างเช่น โปรแกรมอาจใช้ 1 thread เพื่อให้จัดการ I/O แบบเรียลไทม์ เช่น คีย์บอร์ดหรือเม้าส์จากผู้ใช้ ในขณะที่ thread อื่นๆ ที่มี priority ต่ำกว่าจัดการกับงานที่ไม่ต้องทำให้เสร็จในทันทีคือเป็นงานที่ไม่เรียลไทม์ เช่น การคำนวณค่า หรือ การพิมพ์งานออกเครื่องพิมพ์ เป็นต้น

การเขียนโปรแกรมแบบ Multithread ยังมีประโยชน์กับโปรแกรมที่ทำงานในแบบเครือข่าย หรือการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ด้วย เช่นโปรแกรมอาจสร้างแต่ละ thread จัดการกับแต่ละไคลเอนท์ที่ต่อเข้ามา หรือแต่ละการติดต่อสื่อสารของอุปกรณ์ เป็นต้น

2.3.4 โปรแกรมที่ควรใช้ Multithread

Thread จะมีประโยชน์เมื่อโปรแกรมเมอร์ต้องการให้โปรแกรมของตนสามารถทำงานได้หลายๆงานพร้อมๆกัน โดยปกติแล้ว thread จะถูกใช้ในจุดประสงค์ดังต่อไปนี้

- เมื่อต้องการแบ่งงาน (task) ให้มีประสิทธิภาพ กล่าวคือโปรแกรมหนึ่งๆต้องการให้งานที่แตกต่างกันสามารถทำงานได้พร้อมๆกัน ซึ่งจะมีประสิทธิภาพมากกว่าถ้าให้ thread หนึ่งแทนงานๆหนึ่งมากกว่าที่จะให้โพรเซส หนึ่งแทนงานๆหนึ่ง
- เมื่อต้องการกำหนดค่า priority ให้กับงานต่างๆ งานที่ทำเป็น thread แล้วนั้นสามารถถูกกำหนดให้มีค่า priority ที่ต่างกันได้ตามความต้องการ
- สำหรับการประมวลผลแบบbackground การทำ thread เป็นแนวทางที่ดีสำหรับการจัดการกับโพรเซสแบบ background เช่นงานพิมพ์ออกเครื่องพิมพ์

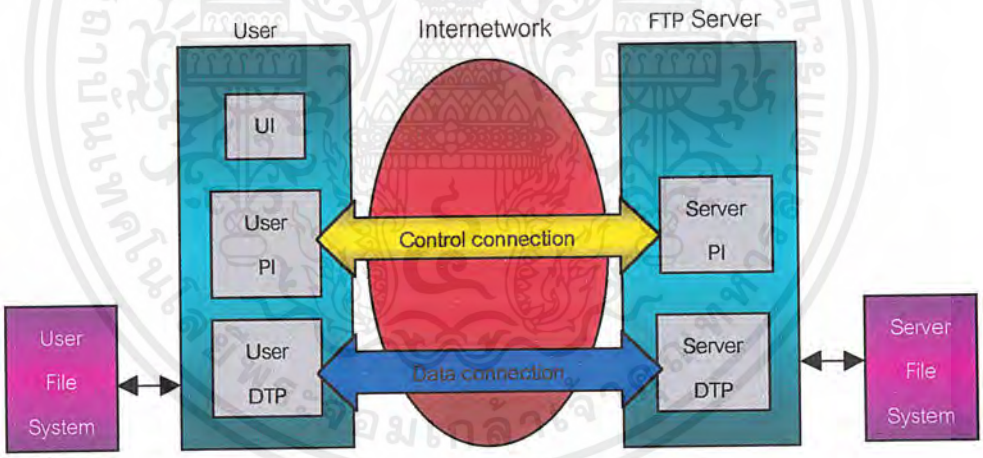
เมื่อต้องการให้งานเดียวกันทำไปพร้อมๆกัน ในบางครั้งโปรแกรมสามารถสร้างหลายๆ instance ซึ่งทุก instance ทำงานอย่างเดียวกัน

2.4 โพรโตคอลส่งผ่านข้อมูล (File Transfer Protocol)

เป็นโพรโตคอล(protocol) ที่ใช้ในการรับส่งไฟล์ระหว่างคอมพิวเตอร์สองเครื่องไคลเอนท์กับเซิฟเวอร์ โดยอาศัยโพรโตคอล TCP/IP โดย โปรแกรมรับส่งไฟล์ข้อมูลจะให้ความสามารถผู้ใช้ในการทำสำเนาไฟล์ระหว่างระบบ ดูรายการไดเร็กทอรี และทำงานทั่วไปที่พบได้บ่อย เช่น การเปลี่ยนชื่อใหม่หรือการลบไฟล์ ซึ่งฟังก์ชันเหล่านี้เป็นส่วนหนึ่งของมาตรฐานในชุดโพรโตคอล TCP/IP

โพรโตคอลส่งผ่านข้อมูลแตกต่างจากโพรโตคอลแอพพลิเคชันส่วนใหญ่ตรงที่โพรโตคอลส่งผ่านข้อมูลจะใช้โพรโตคอล TCP สองตัวแยกจากกันในการเชื่อมต่อระหว่างโพรโตคอลส่งผ่านข้อมูลฝั่งไคลเอนท์ และ โพรโตคอลส่งผ่านข้อมูลฝั่งเซิฟเวอร์ โดยโพรโตคอล TCP แรกที่ใช้เมื่อเกิดการเชื่อมต่อครั้งแรกเป็นการเชื่อมต่อการควบคุม (Control connection) ใช้สำหรับไคลเอนท์ ในการรับส่งคำสั่งคอมมาน และสำหรับเซิฟเวอร์ในการส่งการตอบรับ ส่วนโพรโตคอล TCP ตัวที่สองเป็นการเชื่อมต่อข้อมูล (Data connection) ใช้เพื่อการรับส่งข้อมูล

2.4.1 FTP Model



รูปที่ 2.6 FTP Model

ส่วนของโพรโตคอลส่งผ่านข้อมูลฝั่งไคลเอนท์ ประกอบด้วยส่วนติดต่อกับผู้ใช้ (User Interface(UI)) ซึ่งอาจเป็นแบบบรรทัดคำสั่งคอมมาน (command line driven) หรือ หน้าต่างติดต่อกับผู้ใช้ (Window User Interface) และอีกสองส่วนประกอบ คือ User Protocol Interpreter (PI) และ User Data Transfer Process (DTP) ในส่วนของโพรโตคอลฝั่งเซิฟเวอร์ ประกอบด้วย 2 ส่วน คือ Server Protocol Interpreter (PI) และ User Data Transfer Process (DTP)

เมื่อเริ่มทำงานโพรโตคอลส่งผ่านข้อมูลฝั่งไคลเอนท์จะ รับชื่อและตำแหน่ง IP ของโพรโตคอลส่งผ่านข้อมูลฝั่งเซิฟเวอร์จากผู้ใช้ แล้ว PI ฝั่งไคลเอนท์ จะสร้างการเชื่อมต่อโดยใช้โพรโตคอล TCP ไปยังพอร์ตของโพรโตคอลส่งผ่านข้อมูลฝั่งเซิฟเวอร์ ซึ่งเชื่อมต่อกับ PI ฝั่งเซิฟเวอร์ เป็นชนิดการเชื่อมต่อการควบคุม ในส่วนของโพรโตคอลส่งผ่านข้อมูลฝั่งเซิฟเวอร์เมื่อเริ่มทำงานจะเปิดพอร์ต TCP พอร์ต 21 ไว้เพื่อรอการเชื่อมต่อควบคุม จาก PI ผู้ใช้ ซึ่งจะส่งคำสั่งมา และ PI เซิฟเวอร์ จะส่งคำตอบรับกลับมาให้ผู้ใช้ สถานะต่างๆ จะถูกแสดงออกมาทาง UI

ในการส่งผ่านข้อมูลจะทำโดยโปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ จะเปิดพอร์ตชั่วคราว (local port) และผ่านค่า IP address และ หมายเลขพอร์ต TCP ไปให้โปรแกรมรับส่งข้อมูลฝั่งเซิฟเวอร์ และใช้พอร์ตนี้ในการส่งผ่านข้อมูลดังนั้นก่อนที่ข้อมูลจะถูกส่งผ่านระหว่างโปรแกรมรับส่งข้อมูลฝั่งไคลเอนท์/เซิฟเวอร์ จำเป็นต้องมีการสร้างการเชื่อมต่อ TCP ตัวที่สองระหว่าง DTP ฝั่งไคลเอนท์ กับ DTP ฝั่งเซิฟเวอร์โดยเมื่อ PI ไคลเอนท์ ส่งหมายเลขพอร์ตมายัง PI เซิฟเวอร์ ผ่านทางการเชื่อมต่อควบคุม แล้ว DTP ฝั่งเซิฟเวอร์ จะแอคทีฟพอร์ต 20 บนเซิฟเวอร์ สำหรับการเชื่อมต่อข้อมูล เมื่อการเชื่อมต่อถูกสร้างขึ้น PI ไคลเอนท์จะส่งคำสั่งคอมมานส่งผ่านข้อมูลมา แล้วข้อมูลจะถูกส่งผ่านส่วนการเชื่อมต่อข้อมูล เมื่อทำการส่งข้อมูลเรียบร้อยแล้วเซิฟเวอร์จะทำการปิดส่วนการเชื่อมต่อข้อมูลลง เป็นผลให้การเชื่อมต่อในด้านไคลเอนท์ถูกปิดลงด้วย

2.4.2 **ฟังชันก์ในการส่งผ่านข้อมูล** (DATA TRANSFER FUNCTION)

มีหลายคำสั่งที่เกี่ยวข้องกับการส่งผ่านข้อมูลระหว่างเซิฟเวอร์ คำสั่งของการส่งผ่านข้อมูลเหล่านี้จะรวมถึงโหมดของคำสั่งคอมมาน ซึ่งใช้ระบุว่าบิตของข้อมูลจะถูกส่งผ่านอย่างไร และโครงสร้างและ ประเภทของคำสั่งคอมมาน ซึ่งถูกใช้เพื่อกำหนดวิธีที่ข้อมูลจะถูกนำเสนอ

2.4.2.1 **ชนิดของข้อมูล** (DATA TYPE) ผู้ใช้เป็นผู้ระบุชนิดซึ่งอาจจะเป็นได้ทั้งแบบแน่นอน เช่น ASCII หรือ EBCDIC และแบบโลคัล การกำหนดขนาดไบต์เพื่อทำการแปลง จะอ้างเป็นไบต์มีขนาดตามสมควร ซึ่งการส่งทางการเชื่อมต่อข้อมูลจะเรียกได้ว่าเป็นการส่งผ่านขนาดเป็นไบต์

การนำเสนอรูปแบบที่ใช้มีดังนี้

- ASCII TYPE เป็น default type ถูกใช้เป็นค่าเริ่มต้นสำหรับการส่งผ่านเท็กซ์ไฟล์ ข้อมูลที่ส่งจะถูกส่งในรูปของมาตรฐาน NVT-ASCII 8 บิต
- EBCDIC TYPE ใช้สำหรับการส่งระหว่างเซิฟเวอร์ ซึ่งใช้ EBCDIC ของอักขระภายในสำหรับส่ง ข้อมูลที่ถูกส่งจะมีขนาดอักขระ EBCDIC 8 บิต ซึ่ง ASCII และ EBCDIC จะต่างกันที่การระบุหน้าที่

- IMAGE TYPE การส่งจะส่งเป็น แพ็ค แพ็ค ละ 8 บิต การรับจะรับข้อมูลเป็นแพ็คเช่นกัน ชนิดที่เป็น image type จะใช้สำหรับการเก็บหรือการกู้ข้อมูลกลับมา และใช้สำหรับการส่งไฟล์ข้อมูลประเภทไบนารี
- LOCAL TYPE สำหรับคอมพิวเตอร์ที่มีขนาดของไบต์แตกต่างกับเครื่องอื่น เช่น 11-บิต ไบต์ ไบต์ของข้อมูลที่ส่งจะมีขนาดเป็นไบต์ตามสมควรระบุโดยพารามิเตอร์ตัวที่2 คือ ขนาดไบต์ ค่าของขนาดไบต์ ต้องเป็นเลขฐาน 10 จะไม่มีค่า defualt ขนาดของไบต์ที่เหมาะสมไม่จำเป็นต้องเท่ากับขนาดของ ไบต์การส่งผ่าน 8 บิตต่อไบต์ ถ้ามีความแตกต่างในขนาดของไบต์ ควรจะแพ็คติดกัน และการแปลงนี้ต้องสามารถแปลงกลับ ได้

2.4.2.2 **จัดรูปการควบคุม** (FORMAT CONTROL)

ไฟล์ของอักขระอาจถูกส่งไปยังเซิฟเวอร์เนื่องด้วย 1 ใน 3 จุดมุ่งหมายดังนี้

- เพื่อการพิมพ์
- เพื่อการเก็บ
- เพื่อการประมวลผล

ในกรณีแลกถ้าไฟล์ถูกส่งไปเพื่อการพิมพ์ เซิฟเวอร์ที่รับต้องรู้ด้วยว่าจะจัดรูปแนวตั้งอย่างไร ในกรณีที่สองจะต้องสามารถเก็บไฟล์ที่เซิฟเวอร์และการกู้ไฟล์ได้เหมือนเดิม ส่วนในกรณีสุดท้ายควรจะสามารถย้ายไฟล์จากเซิฟเวอร์หนึ่งไปยังเซิฟเวอร์หนึ่งและประมวลผลไฟล์ที่เซิฟเวอร์ที่สองได้ โดยปราศจากปัญหาที่จะเกิดขึ้น

ซึ่งจัดรูปฟอร์แมต ASCII หรือ EBCDIC เพียงอย่างเดียวไม่เพียงพอกับความต้องการของเงื่อนไขเหล่านี้ทั้งหมด ดังนั้นชนิดเหล่านี้จึงมีพารามิเตอร์ตัวที่สองที่บ่งบอกถึง 1 ใน 3 รูปแบบ ต่อไปนี้

- NON PRINT เป็นฟอร์แมตพื้นฐาน ไฟล์ไม่มีการกำหนด vertical format ถ้ามีการส่งไปพิมพ์ จะใช้ค่ามาตรฐานที่กำหนดอยู่ในเครื่องพิมพ์ โดยปกติแล้วฟอร์แมตรูปแบบนี้จะใช้สำหรับไฟล์เพื่อการประมวลผลและเก็บลงที่เก็บข้อมูล
- TELNET FORMAT CONTROL ไฟล์จะเก็บรูปแบบของการควบคุมจัดเรียงแนวตั้ง เช่น <CR>, <LF>, <NL>, <VT>, <FF>
- CARRIAGE CONTROL (ASA) ไฟล์จะเก็บรูปแบบของ ASA (FORTAN) การควบคุมจัดเรียงแนวตั้ง

2.4.3 **โครงสร้างข้อมูล** (DATA STRUCTURES)

โพรโตคอลส่งผ่านข้อมูลยอมให้โครงสร้างของไฟล์สามารถถูกกำหนดได้ ซึ่งมีอยู่ 3 ประเภท คือ

- **โครงสร้างไฟล์** (File structure) เป็นdefault เมื่อโครงสร้างของคำสั่งคอมมานไม่ได้ถูกเรียกใช้ โครงสร้างของไฟล์จะมีผลกระทบกับทั้งโหมดการส่งผ่าน อินเตอร์พรีเตอร์ ที่เก็บของไฟล์ ในโครงสร้างไฟล์แบบนี้จะไม่มีโครงสร้างภายในและไฟล์จะถูกพิจารณาให้เป็นลำดับต่อเนื่องของข้อมูลแบบไบต์
- **โครงสร้างเรคคอร์ด** (Record structure) ไฟล์จะถูกสร้างขึ้นในลักษณะของเรคคอร์ดเรียงลำดับ (sequential records)
- **โครงสร้างเพจ** (Page structure) ใช้สำหรับส่งไฟล์ที่ไม่มีความต่อเนื่อง ซึ่งในบางครั้งจะเรียกว่า การเข้าถึงไฟล์อย่างสุ่ม (random access file) ไฟล์เหล่านี้จะอยู่ในรูปของเพจ

2.4.4 **การสร้างการเชื่อมต่อข้อมูล** (ESTABLISHING DATA CONNECTION)

กลไกของการส่งข้อมูลจะประกอบด้วย การเซตอัพการเชื่อมต่อข้อมูลให้มีพอร์ตที่เหมาะสม และเลือกพารามิเตอร์ในการส่ง ทั้งผู้ใช้ และ DTPS ฝั่งเซิฟเวอร์ จะต้องมีพอร์ตข้อมูลเป็นหมายเลขพอร์ตพื้นฐาน ในการประมวลผลฝั่งผู้ใช้จะต้องระบุพอร์ตของข้อมูล ให้ตรงกับพอร์ตของการเชื่อมต่อข้อมูล ส่วนการประมวลผลฝั่งเซิฟเวอร์จะระบุพอร์ตของข้อมูลเป็นพอร์ตที่ติดต่อกับพอร์ตของการเชื่อมต่อการควบคุม ขนาดไบต์ของการส่งผ่านจะเท่ากับ 8 บิตต่อไบต์ ขนาดไบต์นี้มีความสัมพันธ์กับการส่งผ่านข้อมูล แต่ไม่สัมพันธ์กับที่แสดงเป็นตัวอย่างของข้อมูลภายในระบบไฟล์ของเซิฟเวอร์หนึ่ง ๆ

สำหรับกระบวนการส่งผ่านข้อมูล จะรับพอร์ตข้อมูลก่อนที่จะส่งคำสั่งคอมมาน โดยคำสั่งคอมมานของโพรโตคอลส่งผ่านข้อมูลของความต้องการจะเป็นตัวกำหนดทิศทางการส่งข้อมูล ที่เซิฟเวอร์จะรับ ความต้องการจะเริ่มต้นส่งการเชื่อมต่อข้อมูลไปที่พอร์ต เมื่อทำการเชื่อมต่อแล้วการส่งผ่านข้อมูลจะเริ่มระหว่าง DTP และ จากนั้น PI ฝั่งเซิฟเวอร์ จะตอบไปยัง PI ฝั่งผู้ใช้ ทุก ๆ โพรโตคอลส่งผ่านข้อมูลจะต้องมีการใช้พอร์ตข้อมูลเป็น defualt ซึ่ง PI ฝั่งผู้ใช้ เท่านั้นที่สามารถเปลี่ยนเป็น non defualt สามารถเปลี่ยนเป็นพอร์ตของข้อมูล ได้โดยใช้พอร์ตของคำสั่งคอมมาน

โดยทั่วไปเซิฟเวอร์จะมีหน้าที่รับผิดชอบในส่วนของการเก็บรักษาการเชื่อมต่อข้อมูล ทั้งตอนเริ่มต้นและการปิด เซิฟเวอร์จะต้องปิดการเชื่อมต่อข้อมูลภายใต้เงื่อนไขดังนี้ :

1. เซิฟเวอร์มีการส่งไฟล์ที่สมบูรณ์ ในโหมดของการส่งผ่านที่เรียกให้ปิดการเชื่อมต่อโดย EOF
2. เซิฟเวอร์รับคำสั่ง ABORT จากผู้ใช้
3. พอร์ตถูกเปลี่ยนโดยคำสั่งจากผู้ใช้
4. การเชื่อมต่อการควบคุมปิดเองโดยบังเอิญ หรือกรณีอื่น ๆ
5. เกิดข้อผิดพลาดขึ้น

2.4.5 **การจัดการการเชื่อมต่อข้อมูล** (DATA CONNECTION MANAGEMENT) จะถูกกำหนดเป็นแบบใดแบบหนึ่งใน 3 แบบต่อไปนี้

1) Default Data Connection Ports การใช้งานจริงๆของ FTP ทั้งหมดจะใช้พอร์ตเป็น default มีเพียง PI ฝั่งผู้ใช้ที่สามารถกำหนดค่าเริ่มต้นการใช้พอร์ตเป็น non-default

2) Negotiating Non-Default Data Ports PI ฝั่งผู้ใช้อาจจะระบุพอร์ตเป็น non-default ในด้านไคลเอนท์หรือ ร้องขอไปยังด้านเซิฟเวอร์เพื่อให้ระบุพอร์ตเป็น non-default

3) Reuse of the Data Connection ในการใช้โหมดของการส่งผ่านข้อมูลแบบโหมดสตรีม (stream mode) เมื่อถึงจุดสิ้นสุดของไฟล์แล้ว การเชื่อมต่อจะถูกปิดลง เป็นเหตุให้เกิดปัญหาเมื่อมีหลายๆ ไฟล์ต้องการการส่งผ่าน มี 2 วิธีที่ใช้ในการแก้ปัญหาคือ หนึ่งทำให้พอร์ตเป็น non- default และ สองใช้โหมดของการส่งผ่านโหมดอื่น

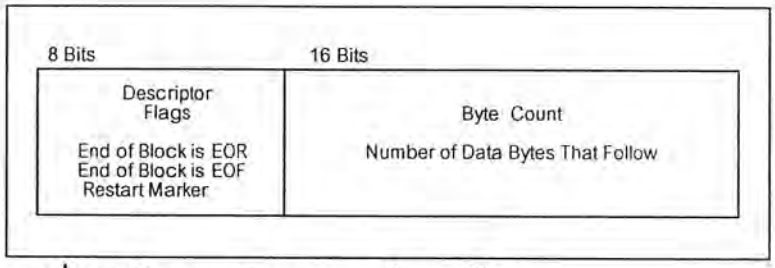
2.4.6 **โหมดการสื่อสาร** (TRANSMISSION MODE)

โหมดการสื่อสารจะถูกใช้ร่วมกับโครงสร้างไฟล์เพื่อกำหนดว่าข้อมูลจะถูกจัดรูปฟอร์แมตสำหรับการส่งผ่านอย่างไร ซึ่งมีอยู่ 3 ประเภท คือ

1) Stream mode

- สำหรับโครงสร้างไฟล์ (file-structure) ไฟล์จะถูกส่งในลักษณะชุดการไหล (stream)ของไบต์ โพรโตคอลส่งผ่านข้อมูลจะอาศัยโพรโตคอล TCP เพื่อตรวจสอบความถูกต้องของข้อมูลและไม่ต้องมีการเพิ่มส่วนเฮดเดอร์ หรือตัวกำหนดขอบเขต (delimiter) เข้ามาในข้อมูล วิธีการเดียวที่จะส่งสัญญาณเมื่อพบว่าถึงจุดสิ้นสุดของไฟล์แล้ว โดยการปิดการเชื่อมต่อข้อมูล
- สำหรับโครงสร้างเรคคอร์ด (record-structure) แต่ละเรคคอร์ดจะถูกกำหนดขอบเขตโดยโค้ดคอนโทรล 2 ไบต์ สำหรับ End Of Record (EOR) และโค้ด อีก 2 ไบต์ ถูกใช้สำหรับ End Of File (EOF)

2) Block mode ไฟล์จะถูกส่งผ่านเป็นลำดับชุดของบล็อคข้อมูล แต่ละบล็อคเริ่มต้นด้วย 3-ไบต์สำหรับส่วนเฮดเดอร์ ซึ่งส่วนเฮดเดอร์จะมีรูปแบบดังรูปที่ 2.7



รูปที่ 2.7 ลักษณะส่วนเฮดเดอร์ของบล็อก

สังเกตได้ว่า Descriptor เก็บค่า EOR flag ซึ่งใช้สำหรับกำหนดขอบเขตของเรคคอร์ด และ EOF flag ใช้สำหรับกำหนดว่าเป็นบล็อกสุดท้ายของไฟล์

3) Compress mode นำเสนอรูปแบบที่เป็นพื้นฐานของการบีบอัดข้อมูล เมื่อไฟล์ถูกส่งในโหมดนี้ อักขระจะถูกส่งซ้ำอย่างมีประสิทธิภาพ การเข้ารหัสเป็นพิเศษจะถูกใช้เพื่อส่งสัญญาณเป็นการบอกว่าเป็นรายการอักขระพิเศษควรถูกขยาย

ข้อผิดพลาดในการกู้และการเริ่มต้นใหม่ (ERROR RECOVERY AND RESTART)

ในปัจจุบันหลายๆ องค์กรมีความจำเป็นต้องการจะส่งไฟล์ที่มีขนาดใหญ่มาก สมมติว่าในระหว่างการส่งผ่านไฟล์ที่ใหญ่นั้นเกิดความผิดพลาดขึ้นทำให้ส่งไฟล์ไม่สำเร็จ ถ้าเป็นการส่งผ่านโพรโตคอลส่งผ่านข้อมูลทั่วไป ก็อาจจะต้องเริ่มต้นทั้งหมดใหม่อีกครั้งหรือไม่ผู้ใช้ก็จะต้องทำการบรรจุไฟล์ขึ้นมาใหม่ด้วยตัวเอง

การให้บริการเริ่มต้นการทำงานใหม่(restart) ถูกออกแบบขึ้นมาเพื่อใช้แก้ปัญหานี้ ถ้าบล็อคหรือการส่งผ่านข้อมูลอยู่ในโหมดการบีบอัด (compress mode) และการบริการเริ่มต้นทำงานใหม่ถูกนำมาใช้แล้ว การส่งโพรโตคอลส่งผ่านข้อมูลจะมีความสามารถในการแทรกตัวเครื่องหมายการเริ่มต้นทำงานใหม่ลงไปในชุดการไหลของข้อมูล เมื่อไหร่ก็ตามที่ผู้รับได้รับตัวเครื่องหมาย จะเขียนข้อมูลไปบนที่เก็บที่ไม่สามารถเปลี่ยนแปลงได้ง่ายๆ และเก็บแทร็คของตำแหน่งของตัวเครื่องหมายในข้อมูล

ถ้าระบบผู้รับของผู้รับเป็นแบบจำกัดเฉพาะแห่ง ผู้ใช้สุดท้ายจะถูกแจ้งถึงตัวเครื่องหมายแต่ละตัวทันทีที่ข้อมูลถูกเก็บ ถ้าระบบผู้รับอยู่ห่างไกล แมสเสสจะถูกส่งกลับไปที่ผู้ใช้ผ่านการเชื่อมต่อการควบคุมเพื่อบ่งชี้ว่าข้อมูลถูกเก็บอย่างปลอดภัย

หลังจากระบบเกิดความผิดพลาดผู้ใช้สามารถใช้คำสั่งคอมมานในการเริ่มต้นทำงานใหม่โดยใช้ค่าของตัวเครื่องหมาย เป็นตัวอาร์กิวเมนต์

การให้บริการการเริ่มต้นทำงานใหม่นี้เป็นเพียงทางเลือก มีเพียงไม่กี่โพรโตคอลแอพพลิเคชันของ TCP/IP ที่มีการให้บริการนี้อยู่

2.4.7 ฟังก์ชันในการส่งผ่านไฟล์ (FILE TRANSFER FUNCTIONS)

2.4.7.1 คำสั่งคอมมานโพรโตคอลส่งผ่านข้อมูล(FTP COMMAND) คำสั่งจะถูกส่งจากไฟล์รับส่งข้อมูลฝั่งไคลเอนท์ไปยังฝั่งเซิฟเวอร์ มีความยาว 3-4 ตัวอักษรโดยบางคำสั่งจะต้องมีพารามิเตอร์ด้วย ต่อไปนี้คือตัวอย่างของคำสั่ง

1) Access Control Commands

ตารางที่ 2.2 แสดง FTP Command ที่ใช้เกี่ยวกับการส่งผ่านการควบคุม

Command	Parameter(s)	Effect
USER (User Name)	<username>	This command identifies the user of the FTP session
PASS (Password)	<password>	The password associated with the user, specified in the USER command.
CWD (Change Working Directory)	<directory>	Changes the directory on the server to that specified in the <directory> parameter
CDUP (Change to Parent Directory)	-	A special case of the CWD command; moves the directory tree one level up. Equivalent to CD.. in Dos, Windows 95/NT and UNIX.
QUIT (log out)	-	Terminates a user and, if a file transfer is in progress, aborts the file transfer.

2) Data Transfer Commands

ตารางที่ 2.3 แสดง FTP Command ที่ใช้เกี่ยวกับการส่งผ่านเดต้า

Command	Parameter(s)	Effect
PORT (Data Port)	h1,h2,h3,h4,p1,p2	This command tells the FTP Server PI which port on the FTP client will be used to receive of send data
RETR (Retrieve)	<filename>	Requests the FTP server to send the FTP client the specified file via the port specified in the PORT command
STOR (Store)	<filename>	Tells the FTP server to get a file from the FTP client and store it in the filename specified.
RNFR (Rename From) RNTO (Rename To)	<old name> <new name>	These commands, which follow each other request the FTP server to rename the file <old name> to <new name>
ABOR	-	Tells the file server to abort the file transfer in progress.
DELE (Delete)	<filename>	Requests the FTP server to delete the file, <filename>
MKD (Make Directory)	<directory>	Asks the FTP server to create a new directory,<directory>.
RMD (Remove Directory)	<directory>	Requests the FTP server to delete the directory, <directory>

3) Other FTP command

ตารางที่ 2.4 แสดง FTP Command ที่ใช้เกี่ยวกับการส่งผ่านข้อมูลแบบอื่น ๆ

Command	Parameter(s)	Effect
PWD (Print Working Directory)	-	Asks the FTP server to list the files in the current working directory; the list is sent to the port specified with the PORT command.
SITE (Site Parameter)	<parameters>	Used to provide server– or site-specific functions. Can have multiple parameters specified.
STAT (Status)	-	Asks the FTP server to send a status report over the control connection.
HELP (Help)	-	Used to obtain a list of the PI commands supported by the server.
NOOP (Noop)	-	This command does not affect any parameters of previously entered commands. It specifies no action other than that the server send an OK reply.
SYST (System)	-	This command is used to find out the type of operating system at the server

2.4.7.2 **การตอบกลับหรือข้อความตอบรับ** (FTP REPLIES or FTP RESPONSE MESSAGE)

เมื่อคำสั่งคอมมานถูกส่งมายังโพรโตคอลส่งผ่านข้อมูลฝั่งเซิฟเวอร์ PI ฝั่งเซิฟเวอร์ จะแปลและนำการกระทำ ที่เกี่ยวข้องกับคอมมานนั้นออกมา แล้วส่งกลับข้อความตอบรับไปให้ไคลเอนท์ผ่านทางการเชื่อมต่อการควบคุม เพื่อบ่งชี้ถึงความสำเร็จหรือล้มเหลวของการกระทำที่ร้องขอมา

ข้อความตอบรับทั้งหมดจะอยู่ในรูป XYZ ค่า โดย

X : กำหนดการตอบกลับเป็นชนิดทั่วไป

Y : เป็นเครื่องแสดงว่าเป็นการตอบกลับชนิดอะไร

Z : ให้รายละเอียดเพิ่มเติม

ซึ่งโค้ด XYZ นี้จะถูกเข้าใจโดย PI เพื่อเป็นการง่ายในการใช้ จึงมีการเพิ่มข้อความที่อ่านได้ลงไปด้วย โค้ดการตอบรับพร้อมข้อความอธิบายแสดงดังตารางต่อไปนี้

ตารางที่ 2.5 โค้ดการตอบรับของ FTP ส่วนไคลเอนท์

Reply Codes	Text Details
110	Restart marker reply.
120	Service ready in nnn minute.
125	Data connection already open ; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented. Superfluous at this site.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message.
215	NAME system type. Where NAME is an official system name from the list in the Assigned Numbers document.
220	Service ready for new user.
221	Service closing control connection. Logged out if appropriate.
225	Data connection open; no transfer in progress.

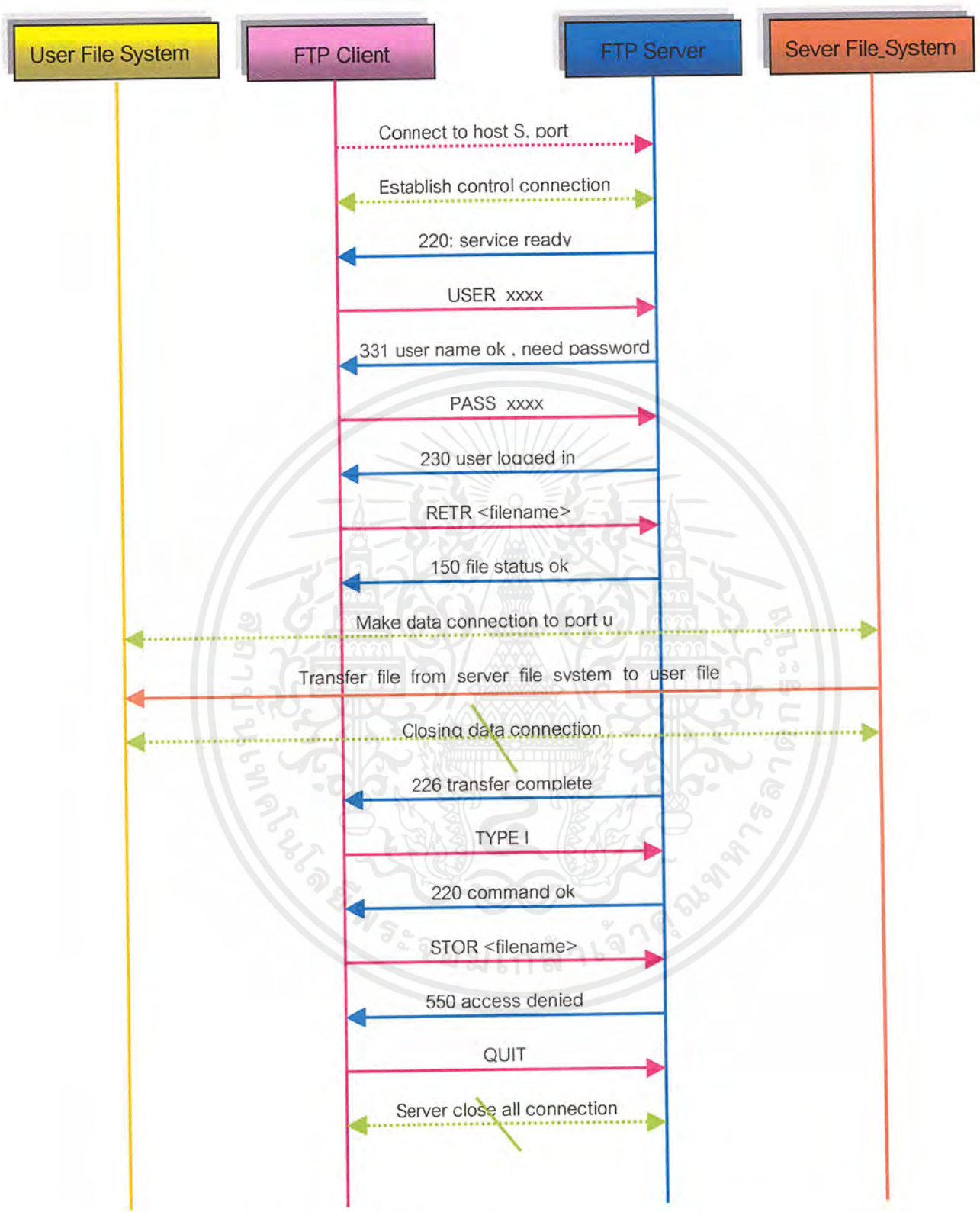
ตารางที่ 2.5 (ต่อ) โค้ดการตอบรับของ FTP ส่วนไคลเอนท์

Reply Codes	Text Details
226	Closing data connection. Request file action successful (for example file transfer or file abort).
227	Entering Passive Mode (h1, h2, h3, h4, p1, p2).
230	User logged in, proceed.
250	Requested file action okay, completed
257	"PATHNAME" created.
331	User name okay, need password.
332	Need account for login.
350	Requested file action pending further information.
421	Service not available, closing control connection.
425	Can't open data connection
426	Connection closed; transfer aborted
450	Requested file action not taken. File unavailable (e.g. , file busy)
451	Requested action aborted: local error in processing.
452	Requested action not taken. Insufficient storage space in system.
500	Syntax error, command unrecognized. This may include errors such as command line too long.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands
504	Command not implemented for that parameter.
530	Not logged in
532	Need account for storing files
550	Requested action not taken. File unavailable (e.g., file not found, no access).
551	Requested action aborted: page type unknown.
552	Requested file action aborted. Exceeded storage allocation (for current directory or data set.).
553	Requested action not taken. File name not allowed.

2.4.8 ลักษณะการรับส่งคำสั่งในโพรโตคอลส่งผ่านข้อมูลและโค๊ดของการตอบกลับ (TYPICAL FTP SCENARIO)

ต่อไปนี้คือลักษณะการรับส่งคำสั่งและคำตอบรับระหว่างส่วนไคลเอนท์กับส่วนเซิฟเวอร์ โดยกำหนดให้ผู้ใช้ซึ่งอยู่ที่ host U จะทำการรับส่งข้อมูลกับ host S สัญลักษณ์ ----> แทนคำสั่งที่ส่งจากไคลเอนท์ (host U) ไปยังส่วนเซิฟเวอร์ (host S) และสัญลักษณ์ <---- แทนการตอบรับที่ถูกส่งจากเซิฟเวอร์ (host S) ไปยังส่วนไคลเอนท์ (host U)

คำสั่งจากผู้ใช้	การกระทำที่เกิดขึ้น
ftp (host S) multics <CR>	Connect to host S, port L,
	establishing control connections.
	<---- 220 Service ready <CRLF>.
username Doe <CR>	USER Doe <CRLF> ---->
	<---- 331 User name ok,
	need password <CRLF>.
password mumble <CR>	PASS mumble <CRLF> ---->
	<---- 230 User logged in <CRLF>.
retrieve (local type) ASCII <CR>	User-FTP opens local file in ASCII.
(local pathname) test 1 <CR>	RETR test.p11 <CRLF> ---->
(for.pathname) tes.p11 <CR>	<---- 150 File status okay ; about to open
	data connection <CRLF>.
	Server makes data connection to port U.
	<---- 226 Closing data connection,
	file transfer successful <CRLF>.
type Image <CR>	TYPE I <CRLF>
	<---- 200 Command OK <CRLF>.
store (local type) image <CR>	User-FTP opens local file in Image.
(local pathname) file dump <CR>	STORE >udd>cn>fd <CRLF> ---->
(for.pathname) >udd>cn>fd <CR>	
	<---- 550 Access denied <CRLF>.
terminate	QUIT <CRLF> ---->
	Server closes all connections.



รูปที่ 2.8 ไดอะแกรมการรับส่งคำสั่งโพรโตคอลส่งผ่านข้อมูลและการตอบกลับ

2.4.9 ลำดับการรับส่งคำสั่งจากส่วนไคลเอนท์และโค้ดตอบกลับจากส่วนเซิฟเวอร์ (SQUENCING OF COMMANDS AND REPLIES)

การติดต่อกันระหว่างผู้ใช้กับเซิฟเวอร์จะเป็นไปตามลำดับ คือ เมื่อผู้ใช้ส่งคำสั่งไปต้องรอการตอบรับจากฝั่งเซิฟเวอร์เสียก่อน จึงควรจะส่งคำสั่งต่อไป และในบางการตอบรับ ผู้ใช้จะต้องรอการตอบรับที่สองก่อนที่จะสามารถส่งคำสั่งต่อไป เช่น เมื่อเซิฟเวอร์ส่ง โค้ดการตอบรับเป็น 120 "service ready in nnn minute" ผู้ใช้ควรรอให้เซิฟเวอร์ส่งคำตอบรับถัดไป คือ 220 "Service ready for new user" จึงจะส่งคำสั่งต่อไป

ตารางที่ 2.6 Sequencing of Commands and Replies

Type of commands	FTP commands	FTP replies	Next FTP replies
Connection Establishment	-	120 220 421	220 - -
Login	USER	230 530 500, 501, 421 331, 332	- - - -
	PASS	230 202 530 500, 501, 503, 421 332	- - - - -
	CWD	250 500, 501, 502, 421, 530, 550	- -
	CDUP	200 500, 501, 502, 421, 530, 550	- -
Logout	REIN	120 220 421 500, 502	220 - - -
	QUIT	221 500	- -

ตารางที่ 2.6 (ต่อ) Sequencing of Commands and Replies

Type of commands	FTP commands	FTP replies	Next FTP replies
Transfer parameters	PORT	200 500, 501, 502, 421, 530	- -
	MODE	200 500, 501, 504, 421, 530	- -
	TYPE	200 500, 501, 504, 421, 530	- -
File action commands	STOR	125, 150 532, 450, 452, 553 500, 501, 421, 530	(110) 226, 250 425, 426, 451, 551, 552 - -
	RETR	125, 150 532, 450, 452, 553 500, 501, 421, 530	(110) 226, 250 425, 426, 451 - -
	LIST	125, 150 450 500, 501, 502, 421, 530	226, 250 425, 426, 451 - -
	RNFR	450, 550 500, 501, 502, 421, 530	- -
	RNTO	250 532, 553 500, 501, 502, 503, 421, 530	- - -
	DELE	250 450, 550 500, 501, 502, 421, 530	- - -
	RMD	250 500, 501, 502, 421, 530, 550	- -

ตารางที่ 2.6 (ต่อ) Sequencing of Commands and Replies

Type of commands	FTP commands	FTP replies	Next FTP replies
File action commands (Cont')	MKD	257	-
		500, 501, 502, 421, 530, 550	-
	PWD	257	-
		500, 501, 502, 421, 550	-
	ABOR	225, 226	-
		500, 501, 502, 421	-
Information commands	SYST	215	-
		500, 501, 502, 421	-
	STAT	211, 212, 213	-
		500, 501, 502, 421, 530	-
	HELP	211, 214	-
		500, 501, 502, 421	-
Miscellaneous commands	SITE	200	-
		202	-
		500, 501, 530	-
	NOOP	200	-
		500, 421	-

2.5 เครื่องมือที่ใช้ในการเขียนโปรแกรม

2.5.1 เกี่ยวกับVisual C++ 6.0

Visual C++ ได้รับการพัฒนาขึ้นมาจาก Microsoft C/C++ ให้เป็น IDE ที่ทำงานบนระบบปฏิบัติการวินโดว์ได้อย่างเต็มที่ รองรับพัฒนาโปรแกรมวินโดว์โดยมี MFC (Microsoft Foundation Class) เป็นไลบารีที่จะช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดว์แทนการใช้ SDK (Software Development Kit) แต่ก่อนเมื่อจะพัฒนาโปรแกรมบนวินโดว์จะต้องใช้ SDK และ คอมไพเลอร์ภาษา C เช่น ไมโครซอฟต์ C++, บอแลนด์ C++ ช่วยในการเขียนโปรแกรม โค้ดโปรแกรมที่เขียนโดย SDK นี้จะค่อนข้างซับซ้อนและสร้างความลำบากในการศึกษาทำความเข้าใจ จึงได้มีการสร้างคลาสขึ้นมาชุดหนึ่ง เขียนขึ้นโดยใช้โครงสร้างของ OOP (Objected Oriented Programming) ด้วยภาษา C++ ชื่อว่า MFC ใช้สำหรับอำนวยความสะดวกในการเขียนโปรแกรมบนวินโดว์โดยเฉพาะ โปรแกรมที่เขียนขึ้นโดยใช้ MFC นั้น จะมีขนาดเล็กและไม่ซับซ้อน ทำให้การพัฒนาโปรแกรมบนวินโดว์ทำได้ง่ายขึ้นกว่าการใช้ SDK ซึ่งทำให้ MFC ก็ได้รับการพัฒนาให้มีประสิทธิภาพสูงขึ้นไปพร้อมกับ Visual C++ จนถึงทุกวันนี้

ข้อดี ของการเขียนโปรแกรมด้วย Visual C++ และ MFC อีกข้อหนึ่งคือ ความสามารถในการพอร์ตเทเบิล (Portability) หมายความว่าหากเรามีซอร์สโค้ดที่เขียนด้วย Visual C++ ในเวอร์ชันที่ต่ำกว่า เราสามารถนำมาคอมไพล์และลิงค์ใหม่ได้ โดยใช้ Visual C++ 6 หรือที่สูงกว่านี้ได้

การที่ Visual C++ 6 ต้องติดตั้ง IE4 ลงไปนั้นก็เพราะว่า โปรแกรมจะใช้ความสามารถของ IE4 ในการแสดงระบบช่วยเหลือ ซึ่งอยู่ในรูป HTML Help แต่ในโปรเจ็กต์นี้ทำการพัฒนาโปแกรมบนระบบปฏิบัติการ Window 98 ก็สามารถติดตั้งโปรแกรม Visual C++ 6 ลงไปได้เลยโดยไม่ต้องติดตั้ง IE4 อีก (เพราะ IE4 รวมอยู่กับ Window 98)

สิ่งที่ควรศึกษาคือ ศึกษาว่าการเขียนโปรแกรมด้วย MFC นั้นทำอย่างไร และ MFC ให้อะไรกับเราบ้าง

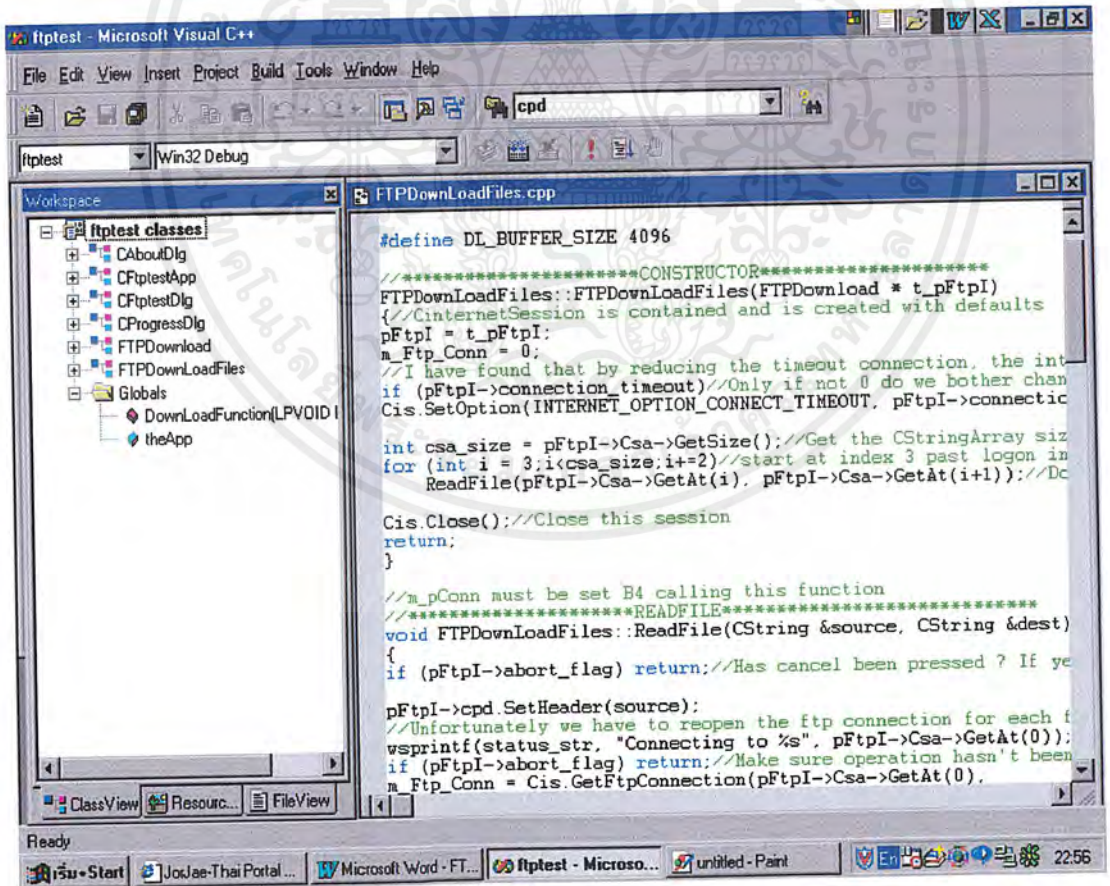
2.5.2 โปรเจ็กต์เวิร์กสเปซ (Project Workspace)

เมื่อสร้างโปรแกรมโดยใช้ Visual C++ สิ่งแรกที่จะต้องทำคือ การสร้างโปรเจ็กต์เวิร์สเปซใหม่ซึ่งเป็นการกำหนดพื้นที่ในการเก็บโปรเจ็กต์ที่ต้องการสร้างและกำหนดตัวเลือกต่างๆ การใช้งานโปรเจ็กต์ของ Visual C++ จะมีลักษณะการทำงานเหมือนโปรเจ็กต์ไฟล์ทั่วไป จะเห็นได้ว่าในการเขียนโปรแกรมบนระบบดอส สามารถเขียนโปรแกรมใหญ่ๆ ได้โดยใช้ไฟล์เพียงไฟล์เดียว แต่ในการเขียนโปรแกรมบนวินโดว์นั้น จะต้องใช้ส่วนประกอบต่างๆและใช้ไฟล์ร่วมกันหลายๆไฟล์เพื่อให้เป็นสัดส่วนและแก้ไขโปรแกรมได้ง่ายขึ้น

โปรเจ็กต์ไฟล์ของ Visual C++ 6 จะเรียกว่า "เวิร์กสเปซ" โดยจะใช้นามสกุล .dsw เป็นไฟล์ที่เก็บตัวเลือกต่างๆของโปรเจ็กต์ และก็สามารถโหลดโปรเจ็กต์ที่เขียนด้วย Visual C++ เวอร์ชันที่ต่ำกว่านี้ได้ เช่น .mak หรือ .mdp เป็นต้น

ประโยชน์ของโปรเจ็กต์ไฟล์ สรุปได้ดังนี้

1. โปรเจ็กต์ไฟล์จะเก็บรายชื่อของไฟล์ที่เป็นซอร์สโค้ดโปรแกรมทั้งหมด ที่ใช้ร่วมกันในโปรเจ็กต์ เช่น ซอร์สโปรแกรม .h, .cpp รวมทั้งไฟล์ฐานข้อมูลโปรแกรมที่ใช้ใน ClassWizard เป็นต้น
2. โปรเจ็กต์ไฟล์จะเก็บค่าตัวเลือกสำหรับการคอมไพล์และการลิงค์เอาไว้ เพื่อบอกให้รู้ว่าโปรเจ็กต์นี้จะทำการคอมไพล์และลิงค์กับไลบารีใด หรือมีการสร้างส่วนประกอบอื่นๆอีกหรือไม่ เช่น ส่วนประกอบในการดีบัก
3. โปรเจ็กต์ไฟล์จะเก็บค่าตัวเลือกที่แสดงว่าโปรเจ็กต์นี้เป็นโปรเจ็กต์แบบใดเมื่อทำการคอมไพล์ เช่น Windows Application (.EXE) หรือ Dynamic-Link Library (.DLL) เป็นต้น



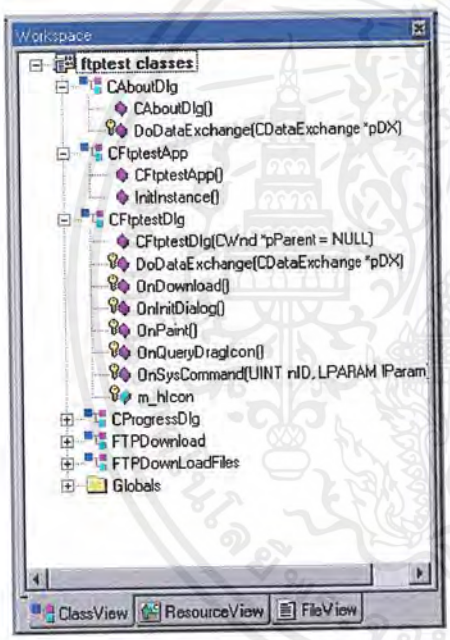
รูปที่ 2.9 หน้าจอ VC++ เมื่อเปิดเวริก์สเปซ

2.5.3 การวิเคราะห์ภาพรวมของโปรแกรม

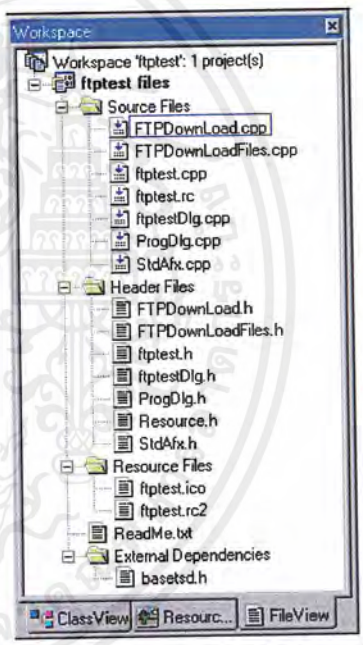
ข้อดีอีกข้อ คือ สามารถวิเคราะห์ภาพรวมของโปรแกรมต่างๆ ได้ โดยจะใช้ Class View และ File View ของโปรเจ็กต์เวิร์กสเปซเพื่อแสดงภาพรวมทั้งหมดของโปรแกรม โดยให้เราคลิกที่แท็บ Class View เพื่อดูรายละเอียดของคลาสก่อน ดังรูป 2.8

จะเห็นได้ว่า Visual C++ ได้แยกโค้ดโปรแกรมออกเป็นส่วนๆ อย่างชัดเจน และแยกให้เห็นว่ามีคลาสอะไรบ้างในโปรเจ็กต์ และสมาชิกของคลาสแต่ละตัวเป็นแบบ Protect หรือ Plublic และตัวแปรโกลบอลมีอะไรบ้าง ดังรูป 2.10 โดยใช้สัญลักษณ์ต่อไปนี้ ในการบอกระดับของสมาชิก

- รูปแม่กุญแจ หมายถึง สมาชิกคลาสที่เป็นแบบ private
- รูปดอกกุญแจ หมายถึงสมาชิกคลาสที่เป็นแบบ protect
- รูปกล่อง หมายถึงสมาชิกคลาสที่เป็นแบบ public



รูป 2.10 Class view ในเวิร์กสเปซ



รูป 2.11 File view ในเวิร์กสเปซ

File View

ได้มีการแยกไฟล์ออกเป็นส่วนๆ คือ Source Files ซึ่งจะเก็บไฟล์ .cpp ไว้ เฮดเดอร์ไฟล์จะเก็บไฟล์ส่วนเฮดเดอร์ .h และ Resource File จะเก็บไฟล์ที่เกี่ยวกับ resource เช่น ไอคอน รูปภาพต่างๆ ดังรูปที่ 2.11

2.5.4 **ความต้องการทางด้านฮาร์ดแวร์**

1. เครื่องคอมพิวเตอร์รุ่น Pentium 100 ขึ้นไป
2. หน่วยความจำไม่ควรต่ำกว่า 16 MB เป็นอย่างน้อย
3. Mouse และจอภาพ VGA หรือที่สูงกว่า
4. เนื้อที่ว่างในฮาร์ดดิสก์ประมาณ 100 MB ขึ้นไป
5. ไดร์ฟซีดีรอมสำหรับการติดตั้ง

ถ้ามีระบบฮาร์ดแวร์ไม่เหมาะสม อาจจะพบความล่าช้าบ้างในการพัฒนาโปรแกรม

2.5.5 **ความต้องการทางด้านซอร์ฟแวร์**

1. ระบบปฏิบัติการ Window95, 98 หรือ NT ซึ่งโปรเจ็กต์นี้ทำการพัฒนาโปรแกรมบน Window 98
2. โปรแกรม Microsoft Internet Explorer 4.01 หรือที่สูงกว่า
3. โปรแกรมติดตั้ง Microsoft Visual C++ เวอร์ชัน 6 0

2.6 Microsoft Foundation Class Library และ Application Framework ใน Visual C++

ในปัจจุบันการเขียนโปรแกรมบนวินโดว์ มี 5 วิธี

1. เขียนด้วยภาษา C และใช้ Win32 APIs ในการทำงานกับวินโดว์ ซึ่งเรียกกันว่าเขียนโดยใช้ SDK (Software Development Kit)
2. เขียนคลาสไลบรารีของวินโดว์ (Windows Class Library) ขึ้นมาใช้งานเอง โดยในคลาสครอบ Win32 APIs เอาไว้ภายใน
3. เขียนโดยใช้ MFC (Microsoft Foundation Class) Library
4. เขียนโดยใช้คลาสไลบรารีอื่นๆเช่น Object Windows Library ของบริษัท Borland
5. เขียนโดยใช้เครื่องมือที่เรียกว่า Rapid Development Tools (RAD) เช่น MS-Visual Basic หรือ Delphi เป็นต้น

แต่ละแบบก็จะมีข้อดีและข้อเสียแตกต่างกันออกไป ซึ่งควรใช้วิธีใดก็ขึ้นอยู่ชนิดของงาน เช่นถ้าเป็นงานเขียนเกมเรื่องของความเร็วเป็นสิ่งสำคัญมากดังนั้นจึงควรใช้วิธีที่ 1 หรือ 2 คือใช้ SDK หรือคลาสไลบรารีที่เขียนขึ้นเอง เพราะวิธีทั้งสองจะตัดสิ่งที่ไม่จำเป็นทั้งหลายออกไป ถ้าต้องการโปรแกรมใช้งานทั่วไปไม่เน้นความเร็วสูงในการประมวลผลและใช้เวลาในการเขียนโปรแกรมน้อยก็ควรใช้วิธีที่ 5 คือใช้เครื่องมืออย่าง Visual Basic หรือ Delphi ส่วนวิธีที่ 3 และ 4 นั้นเหมาะกับงานที่เน้นความเร็วในการประมวลผล มีเวลาในการพัฒนาโปรแกรมพอสมควร ส่วนตัวโปรแกรมเมอร์ต้องการเครื่องมือที่ช่วยในการพัฒนามากและเครื่องมือเหล่านั้นต้องยอมให้โปรแกรมเมอร์เข้าไปขยายความสามารถหรือแก้ไขบางสิ่งให้ตรงกับงานที่ตนต้องการได้ง่าย

ในตอนนี้สนใจเฉพาะวิธีที่ 3 คือการใช้ MFC Library เท่านั้นซึ่งโปรแกรมเมอร์ต้องใช้ภาษา C++ เป็นพื้นฐานในการทำงานร่วมกับ MFC สิ่งหนึ่งที่ทำให้ภาษา C++ เป็นที่นิยมคือมันสามารถพัฒนาคลาสไลบรารีเพื่อใช้งานในอนาคตได้ คลาสไลบรารีบางส่วนก็อาจไม่จำเป็นที่ต้องพัฒนาขึ้นเอง อาจได้มาจากคอมไพเลอร์ของภาษา C++ หรือบางส่วนอาจซื้อมาจากบริษัทซอฟท์แวร์ที่สร้างไว้แล้วก็ได้

คลาสไลบรารีคือกลุ่มของคลาสในภาษา C++ ที่ถูกเขียนขึ้น โดยทำงานเกี่ยวข้องกัน และสามารถนำมาใช้งานในโปรแกรมได้ ในบางครั้งโปรแกรมเมอร์อาจสร้างออบเจ็คจากคลาสที่ให้มาโดยตรงหรือบางครั้งอาจต้องทำการ สืบทอดคลาสก่อนจะนำมาใช้งาน ซึ่งขึ้นอยู่กับลักษณะการออกแบบคลาสไลบรารีเหล่านั้น

แอพพลิเคชันเฟรมเวิร์ค (application framework) เป็น superset ของคลาสไลบรารี ซึ่งคลาสไลบรารีเป็นเพียงกลุ่มของคลาสที่ถูกออกแบบมาเพื่อใช้งานในโปรแกรม แอพพลิเคชันเฟรม-เวิร์ค เป็นมากกว่าคลาสไลบรารีคือ สามารถเป็นตัวกำหนดโครงสร้างของโปรแกรมด้วย

โปรแกรมเมอร์ที่ต้องการใช้ MFC Library ควรทำความรู้จักกับสิ่งเหล่านี้

- MFC Library เป็นคลาสไลบรารีที่เขียนด้วยภาษา C++ และนำฟังก์ชันต่างๆใน Win32 APIs มาเขียนขึ้นเป็นคลาสเพื่อให้ใช้งานได้ง่ายยิ่งขึ้น

- โปรแกรมที่พัฒนาด้วย application framework ใน MFC Library จะมีความเป็นมาตรฐาน จากเดิมโปรแกรมเมอร์แต่ละคนจะมีสไตล์การเขียนโปรแกรมที่ต่างกัน ดังนั้นการที่จะบังคับให้โปรแกรมเมอร์ในทีมทุกคนให้ใช้สไตล์การเขียนโปรแกรมให้เหมือนกันจึงเป็นสิ่งที่ทำได้ลำบาก application framework จะเป็นเครื่องมือเบื้องต้นที่กำหนดสไตล์การเขียนโปรแกรมให้มีลักษณะที่คล้ายคลึงกันได้ง่ายยิ่งขึ้น

- โปรแกรมที่พัฒนาด้วย application framework จะมีขนาดเล็กและทำงานได้อย่างรวดเร็ว โดยเหตุผลที่ทำให้โปรแกรมมีขนาดเล็กคือการทำงานร่วมกับ DLLs ไม่ว่าจะเป็น DLLs ของบริษัท Microsoft หรือ DLLs ที่เขียนขึ้นเอง และการทำงานที่รวดเร็วนั้นเกิดจากการที่คอมไพเลอร์ได้สร้าง machine code ที่มีประสิทธิภาพสูง

- เครื่องมือต่างๆที่ VC++ เตรียมไว้จะช่วยลดงานเขียนโปรแกรมให้น้อยลงได้พอสมควร เช่น

Graphic editor เป็นเครื่องมือสำหรับการสร้าง resouce ต่างๆที่ต้องการในโปรแกรม เช่น ภาพบิตแมป รูปแบบของ dialog และรูปเคอร์เซอร์ เป็นต้น

AppWizard เป็นเครื่องมือในการสร้างโครงหลักของโปรแกรมทั้งหมด รวมถึงการสร้างโค้ดที่จำเป็นสำหรับโปรแกรม ซึ่งเมื่อเริ่มต้นสร้างโปรแกรม AppWizard จะให้เลือกประเภทของโปรแกรมที่ต้องการเช่น โปรแกรมในแบบ Dialog Based โปรแกรมในแบบ Single Document Interface (SDI) หรือโปรแกรมในแบบ Multiple Document Interface (MDI) เป็นต้น

ClassWizard เป็นเครื่องมือสำหรับการสร้าง function prototype และ function body ของวินโดว์เมสเสจต่างๆ

- MFC library มีคลาสต่างๆที่ให้โปรแกรมเมอร์ใช้งานมากมายโดยที่โปรแกรมเมอร์ไม่จำเป็นต้องทราบโครงสร้างภายในของคลาส แต่จะสนใจเพียงวิธีการใช้งานคลาสเท่านั้น

โดยโปรแกรมตามปกติจะใช้ 4 คลาสหลัก

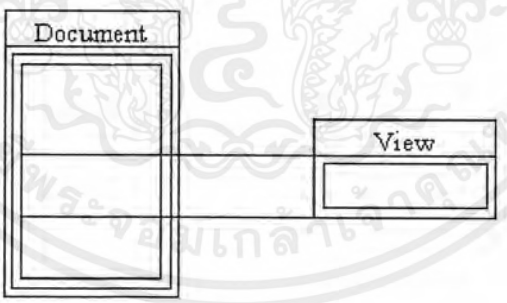
1) Application Class
2) Frame window Class
3) Document Class
4) View Class

โดยแต่ละคลาสนั้นจะมีลักษณะการทำงานดังตาราง 2.7

ตารางที่ 2.7 แสดงคลาสหลักที่มีอยู่ใน MFC

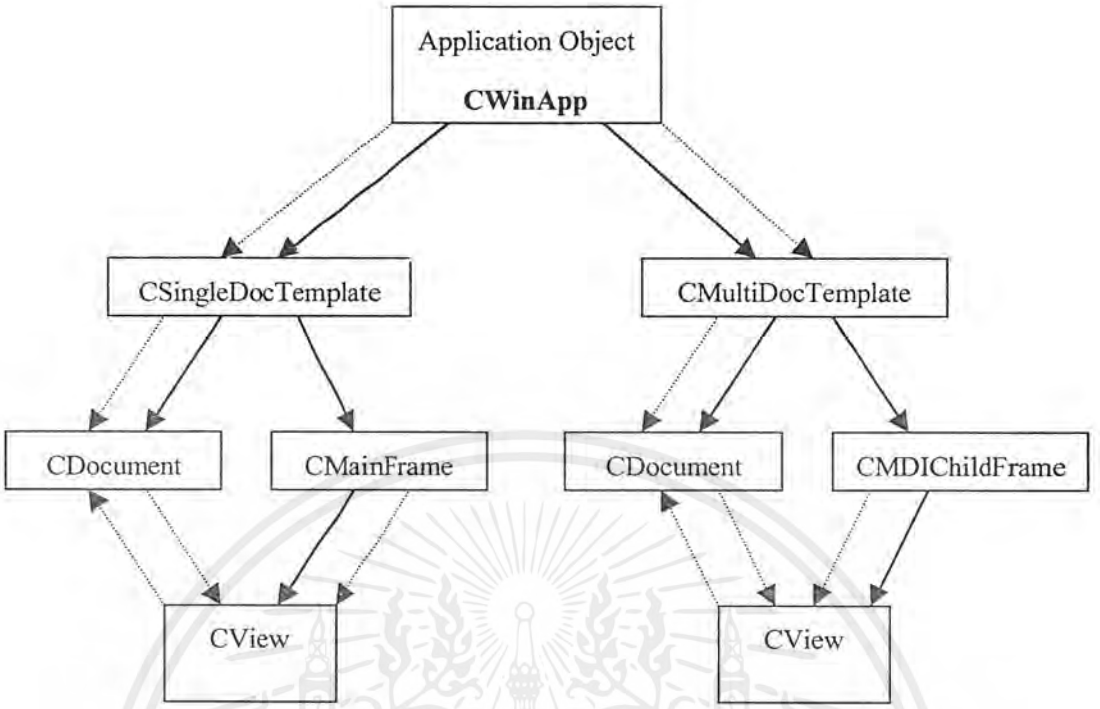
ชนิดของ class	คำอธิบาย
Application Class	แทนความเป็นแอพพริเคชันของโปรแกรม กล่าวคือเป็นคลาสหลักที่ทุกโปรแกรมที่เขียนด้วย MFC ใช้งาน
Frame window Class	แทนกรอบของวินโดว์หลักของโปรแกรม
Document Class	แทนความเป็น document หรือส่วนเก็บข้อมูลหลักของโปรแกรม การอ่าน/เขียนข้อมูลที่โปรแกรมใช้จะเกิดขึ้นที่คลาสนี้
View Class	แทนส่วนแสดงผลบนวินโดว์หรือเรียกว่าส่วน Client area ซึ่งเป็นส่วนที่อยู่ภายในกรอบของวินโดว์ (frame window) อีกที

Document-View Architecture เป็นสถาปัตยกรรมหลักของการเขียนโปรแกรมในรูปแบบของ MFC โดยหลักการแล้ว Document-View Architecture จะแยกส่วนของข้อมูลกับส่วนการแสดงผลข้อมูลออกจากกัน



รูปที่ 2.12 สถาปัตยกรรมของ Document-View Architecture

Document เป็นส่วนที่ใช้เก็บข้อมูลและ View เป็นวินโดว์ที่ทำหน้าที่แสดงข้อมูลที่อยู่ใน Document โดยอาจเป็นการแสดงข้อมูลเพียงบางส่วนใน Document ก็ได้ ซึ่ง Document อาจจะเป็นฐานข้อมูล หรือไฟล์เก็บข้อมูล เป็นต้น สำหรับ Document ใน Doc/View Architecture ไม่ใช่แค่ทำหน้าที่เก็บข้อมูลเท่านั้นแต่ภายใน Document Class ยังมีฟังก์ชัน interface ต่างๆ ที่จำเป็นสำหรับ View เพื่อใช้ในการแสดงข้อมูลที่ตนเก็บอยู่



รูปที่ 2.13 การสร้าง Document ผ่านทาง Document Template Class

Doc/View Architecture สามารถแบ่งเบื้องต้นได้เป็น 2 ลักษณะคือ Single Document Interface (SDI) และ Multiple Document Interface (MDI) สำหรับโปรแกรมที่เขียนในแบบ SDI นั้นจะใช้เพียง 1 Document Class ในการจัดการกับข้อมูล ส่วนโปรแกรมที่เขียนในแบบ MDI จะมี Document Class ได้หลายคลาสซึ่งหมายถึงสามารถจัดการกับข้อมูลได้หลายประเภท (Document Type) พิจารณารูปที่ 2.13 แสดงถึงการสร้าง Document ผ่านทาง Document Template Class

เส้นลูกศรทึบแสดงเส้นทางของการสร้าง View ใน Doc/View Architecture และเส้นลูกศรปะแสดงพอยต์เตอร์ที่สามารถใช้อ้างในโปรแกรมได้ ใช้สำหรับอ้างอิงไปมาระหว่างออบเจ็คในโปรแกรม

ประโยชน์อีกข้อที่ได้รับ คือ ข้อมูลเดียวกันสามารถนำมาแสดงผลได้หลายรูปแบบ

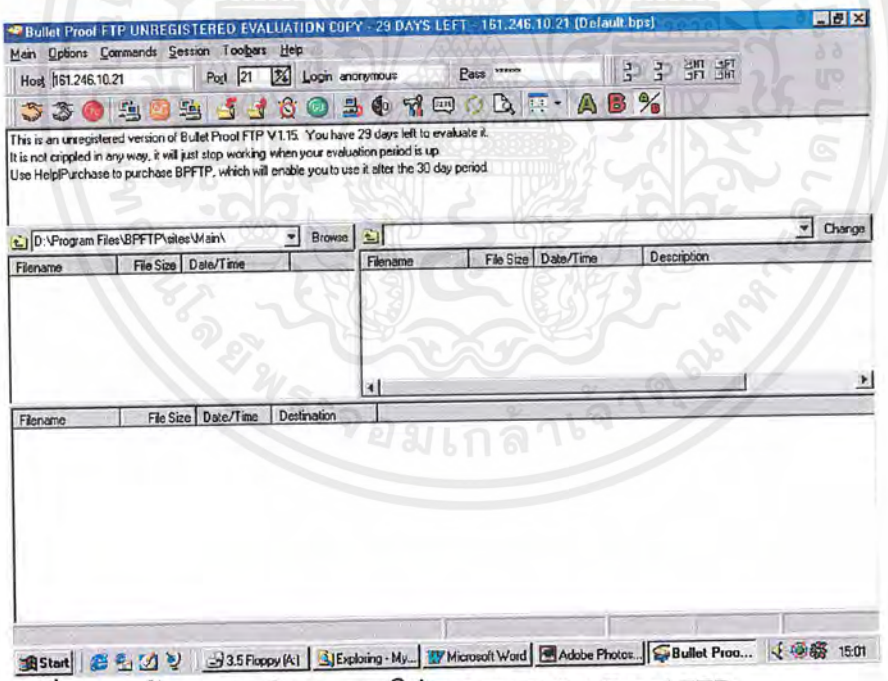
บทที่ 3
ตัวอย่างโปรแกรมส่งผ่านไฟล์ข้อมูลฝั่งไคลเอนท์

เพื่อทำความเข้าใจ และศึกษาฟังก์ชันการทำงานหลักของโปรแกรมส่งผ่านไฟล์ข้อมูล จึงได้ทำการทดสอบใช้โปรแกรมรับส่งไฟล์ข้อมูลฝั่งไคลเอนท์ที่มีผู้พัฒนาและมีผู้ใช้แล้ว โดยทำการทดสอบทั้งจากโปรแกรมที่เป็นเดโม่ และโปรแกรมที่แจกให้ใช้ฟรี

จากการทดสอบใช้โปรแกรม พบว่าฟังก์ชันการทำงานหลัก เช่น การทำการเชื่อมต่อเข้าเซิฟ-เวอร์ การดาวน์โหลดไฟล์ อัพโหลดไฟล์ การเปลี่ยนไดเรกทอรี่ ฯลฯ ทุกโปรแกรมจะมีการรองรับฟังก์ชันเหล่านี้ไว้ได้ดีเหมือนกันหมด ดังนั้นแต่ละโปรแกรมได้พยายามหาจุดเด่นและลูกเล่นต่างๆ เพื่อเพิ่มขีดความสามารถของโปรแกรมและทำให้น่าสนใจมากยิ่งขึ้น

ต่อไปนี้คือตัวอย่างของโปรแกรมส่งผ่านไฟล์ข้อมูลฝั่งไคลเอนท์ทั้งหมดที่ได้ทดลองใช้ พร้อมทั้งคำอธิบายฟังก์ชันการทำงานหลักๆ และจุดเด่นต่างๆ ของแต่ละโปรแกรม

3.1 Bullet Proof FTP (BPFTP)



รูปที่ 3.1 หน้าจอการทำงานของโปรแกรม Bullet Proof FTP

เวอร์ชัน	1.11
วันที่ออก	25 กุมภาพันธ์ 2542
ขนาดของโปรแกรม	579 KB.

ประเภท แชร์แวร์ (Shareware)

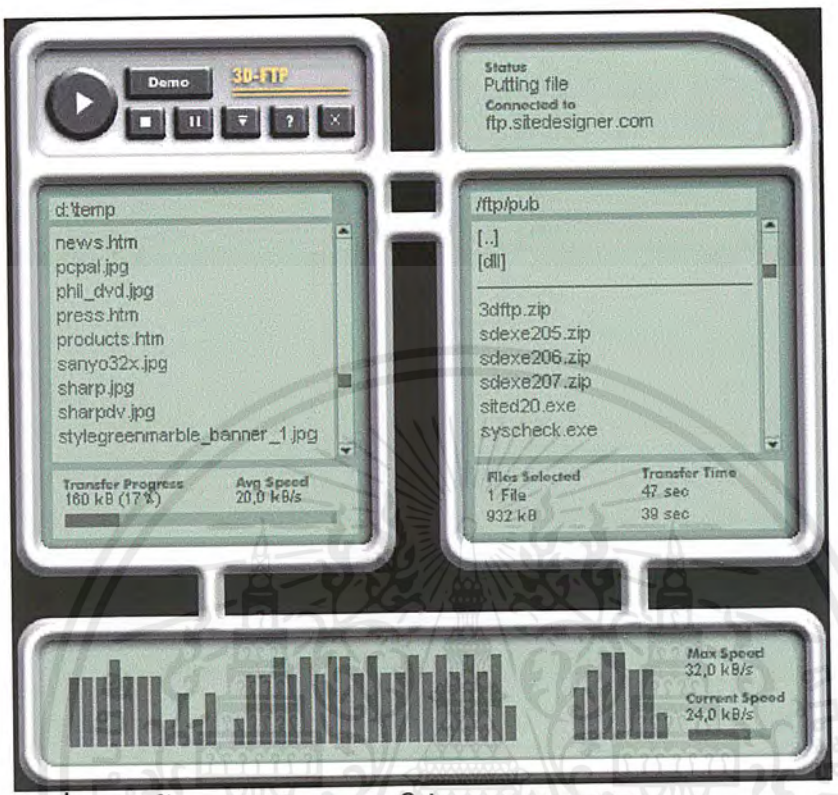
ระบบปฏิบัติการ วินโดว์ 95 / 98 / NT

BPFTP เป็นแชร์แวร์ที่จำกัดขอบเขตให้ทดลองใช้งานได้ฟรีเพียง 30 วัน หลังจากนั้นแล้ว ถ้าไม่ทำการจ่ายเงินสำหรับซื้อโปรแกรม ตัวโปรแกรม BPFTP นี้ก็จะใช้งานไม่ได้อีก

โดยฟังก์ชันต่างๆใน BPFTP นี้ จะมีมากมาย เช่น

- เมื่อเกิดปัญหา ขาดการติดต่อ (Disconnect) BPFTP จะทำการเชื่อมต่ออีกครั้ง และดาวน์โหลดต่อจากเดิม (Automatically reconnect และ resume downloading)
- มีฟังก์ชันที่จะทำการออกจากการเชื่อมต่อโดยอัตโนมัติ เมื่อดาวน์โหลดเสร็จแล้ว
- รองรับการใช้งานแบบคลิกแล้ววาง (drag & drop) ของวินโดว์
- มีไอคอนสำหรับแสดงสถานะในขณะนั้น ของการรับส่งไฟล์
- ถ้า FTP Server มีการจำกัดจำนวนของไฟล์ที่จะให้ดาวน์โหลด ตัวโปรแกรม BPFTP จะมีฟังก์ชันในการออกจากการเชื่อมต่อ และเข้าไปเชื่อมต่อใหม่จนกว่าไฟล์ที่เราต้องการจะได้รับการดาวน์โหลดจนหมด
- มีการแสดงข้อมูลพื้นฐานหลังจากการรับส่งไฟล์แต่ละไฟล์ เช่น ขนาดของไฟล์ อัตราความเร็วในการรับส่งไฟล์ เป็นต้น
- มีแฟ้มข้อมูลสำหรับเก็บรายชื่อเซิฟเวอร์
- ถ้ามีการขาดการติดต่อ ขณะที่ทำการอัพโหลดไฟล์ จะมีการเชื่อมต่อเข้าไปอีกครั้งโดยอัตโนมัติ และทำการอัพโหลดต่อจากเดิม

3.2 3D FTP



รูปที่ 3.2 หน้าจอการทำงานของโปรแกรม 3D FTP

เวอร์ชัน	2.0
วันที่ออก	19 กรกฎาคม 2542
ขนาดของโปรแกรม	1 MB
ประเภท	แชร์แวร์
ระบบปฏิบัติการ	วินโดว์ 95 / 98 / NT

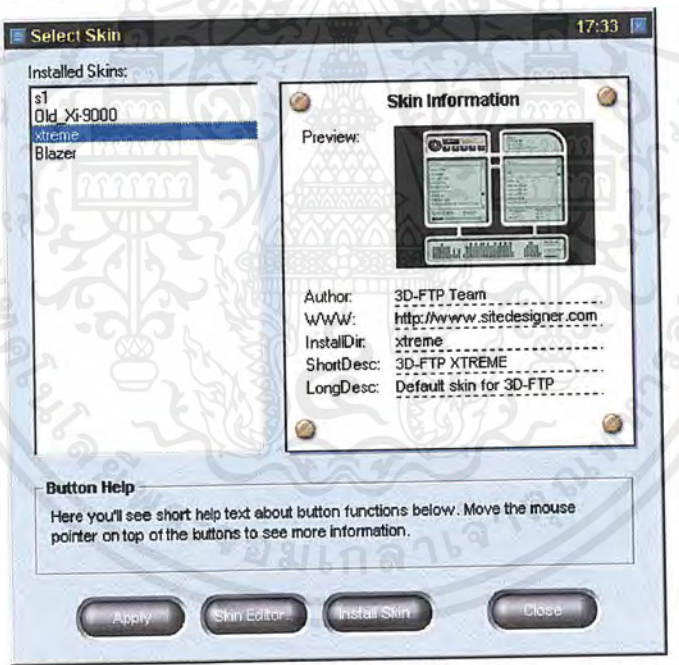
3D FTP เป็นโปรแกรมส่งผ่านไฟล์ข้อมูลฝั่งไคลเอนท์ที่มีหน้าจอสำหรับติดต่อกับผู้ใช้สวยงาม โดยผู้ใช้สามารถดาวน์โหลดรูปแบบของหน้าจอ (skin) เพิ่มเติม หรือ สามารถสร้างรูปแบบของหน้าจอสำหรับตนเองได้โดยใช้ skin editor ที่โปรแกรมให้มา

นอกจากความสวยงามแล้ว 3D FTP ก็ยังมีฟังก์ชันสำหรับการรับส่งไฟล์อย่างครบครัน และสะดวกในการใช้มาก เช่น

- มีหน้าจอแสดงอัตราความเร็วในการรับส่งข้อมูล และเวลาโดยประมาณที่ใช้ในการรับส่งข้อมูล

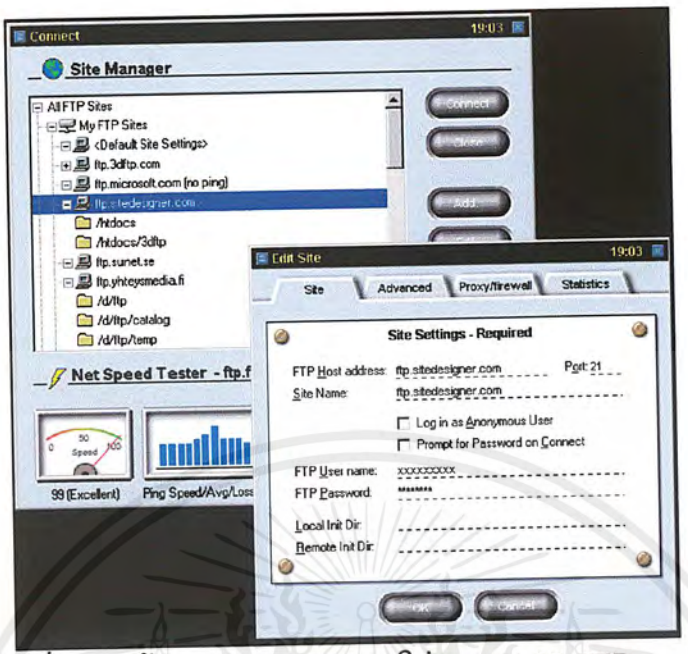
- ถ้ามีการขาดการติดต่อ จะมีฟังก์ชันในการเชื่อมต่อเข้าไปใหม่โดยอัตโนมัติ และสามารถอัพโหลด หรือ ดาวน์โหลด ต่อจากที่ทำไว้เดิมได้
- สามารถตรวจเช็ค อัตราความเร็วในเน็ตเวิร์ก ก่อนที่จะทำการอัพโหลด หรือ ดาวน์โหลดได้
- ถ้าไม่ต้องการใช้หน้าจอจากที่โปรแกรมให้มา ก็สามารถเปลี่ยนเป็นหน้าจอวินโดว์มาตรฐานได้
- รองรับการทำงานแบบคลิกแล้ววาง
- สามารถตั้งเวลาในการอัพโหลด หรือ ดาวน์โหลดได้
- สามารถอัพโหลด หรือ ดาวน์โหลดหลายไฟล์ จากหลายไดเร็กทอรี่ ได้พร้อมๆกัน
- มีกราฟแสดงอัตราการจราจรในเน็ตเวิร์ก เพื่อช่วยในการตรวจสอบปัญหาความคับคั่งของข้อมูลภายในเน็ตเวิร์ก

ตัวอย่างหน้าจอของการเลือก skin



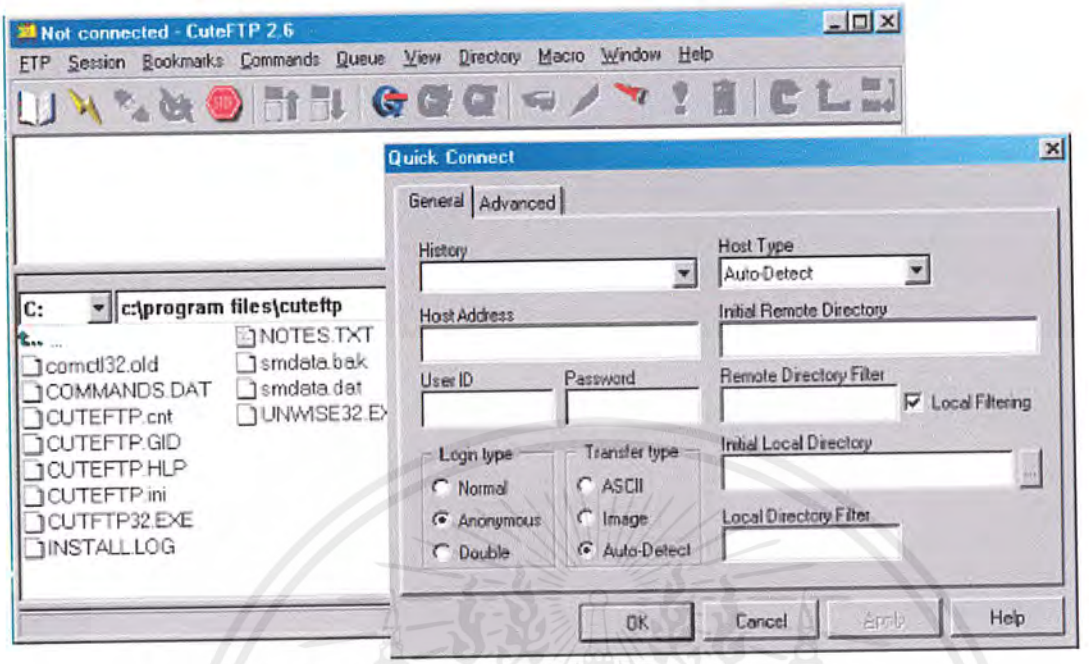
รูปที่ 3.3 หน้าจอการเลือกskinของโปรแกรม 3D FTP

ตัวอย่างหน้าจอของ Site Editor ซึ่งเราสามารถดูสถิติของการรับส่งข้อมูลในแต่ละ site ได้



รูปที่ 3.4 หน้าจอ Site Editor ของโปรแกรม 3D FTP

3.3 CuteFTP



รูปที่ 3.5 หน้าจอการทำงานของโปรแกรม CuteFTP

เวอร์ชัน	3.0.2
วันที่ออก	24 มิถุนายน 2542
ขนาดของโปรแกรม	1.5 MB
ประเภท	แชร์แวร์
ระบบปฏิบัติการ	วินโดว์ 95 / 98 / NT 4.0

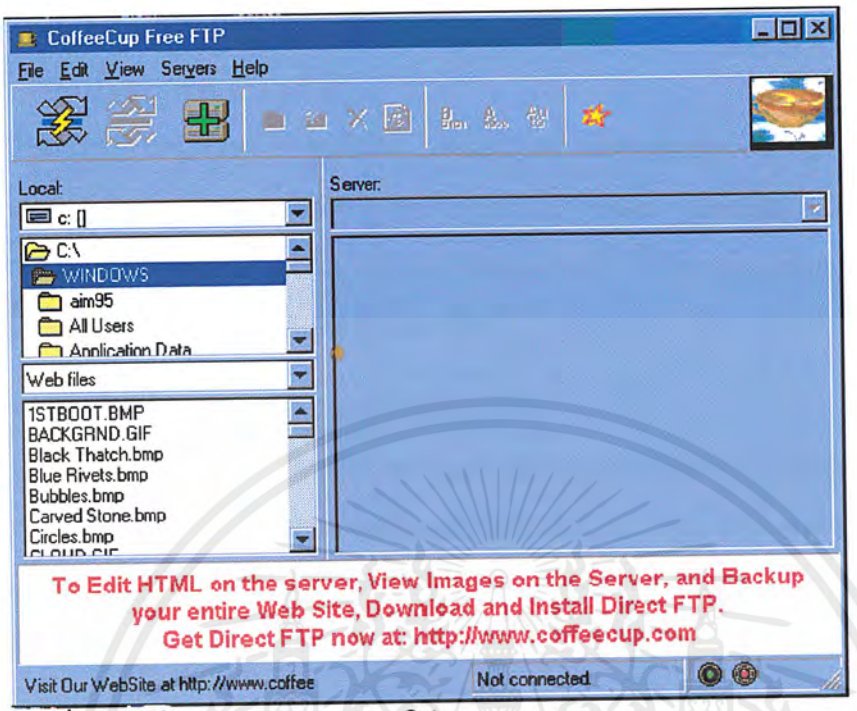
cuteFTP เป็นแอพพลิเคชั่นสำหรับรับส่งไฟล์ ที่ใช้ง่ายสำหรับผู้เริ่มต้น และมีความสามารถเพียงพอสำหรับผู้ใช้ที่ต้องการรับส่งไฟล์ที่มีประสิทธิภาพสูง

- ใช้การคลิกแล้ววาง มีบุ๊กมาร์คเปลี่ยนชื่อใหม่ได้โดยอัตโนมัติ และสามารถเริ่มต้นการดาวน์โหลดได้ใหม่
- มีมาโคร(macro)สำหรับบันทึกงานที่ถูกกระทำบ่อยๆ และสามารถเล่นกลับได้ในภายหลังด้วยไฟล์อื่นๆ
- สามารถทำการอัพโหลดไดเรกทอรี, เปรียบเทียบไดเรกทอรี่, จัดลำดับการอัพโหลดและดาว์นโหลด, สามารถเขียนทับหรือลบทิ้งไดเรกทอรี่ได้อย่างสมบูรณ์แบบ และแสดงการประเมินความก้าวหน้าของการดาวน์โหลดหรืออัพโหลดไฟล์

ในเวอร์ชั่น 3.0

- มีฟังก์ชันการค้นหาไฟล์ และ MP3
- สำหรับไฟล์ที่ได้พยายามทำการดาวน์โหลดหรืออัพโหลดไฟล์อยู่หลายครั้ง ในเวอร์ชั่นนี้จะมีความสามารถทำการเชื่อมต่อใหม่โดยอัตโนมัติและทำการส่งผ่านข้อมูลต่อจนกระทั่งไฟล์ถูกอัพโหลดหรือดาวน์โหลดเรียบร้อย
- มี Site Wizard สำหรับช่วยเหลือผู้ใช้
- สามารถคลิ้กขวาเพื่อทำการส่งไฟล์ไปยังที่ที่ถูกเลือกไว้โดยอัตโนมัติ
- มีการรวมเอาความสามารถของวินโดว์เอ็กซ์เพอร์เรอร์ เช่น การแก้ไขไฟล์ และลบไฟล์ใน recycle bin

3.4 CoffeeCup Free FTP



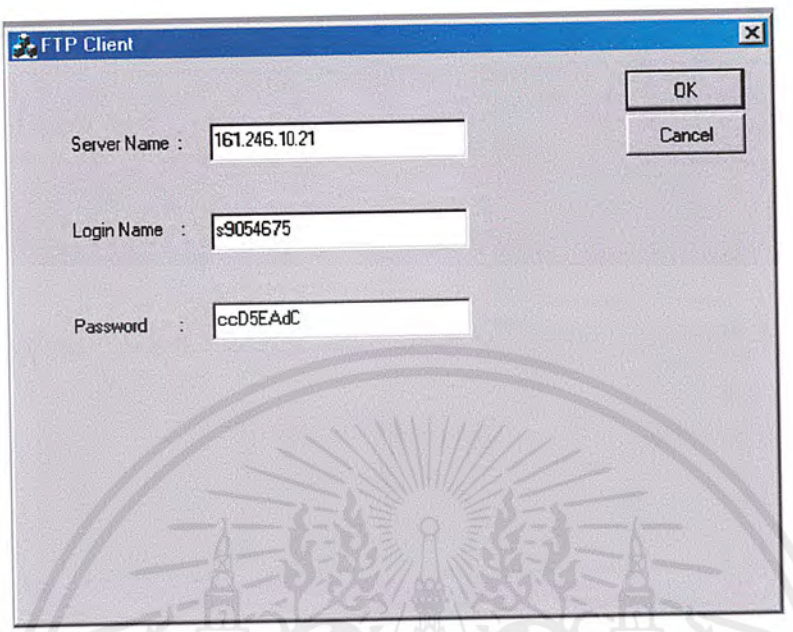
รูปที่ 3.6 หน้าจอการทำงานของโปรแกรม CoffeeCup Free FTP

เวอร์ชัน	1.0
วันที่ออก	10 กันยายน 2542
ขนาดของโปรแกรม	719 K
ประเภท	ฟรี
ระบบปฏิบัติการ	วินโดว์ 95 / 98 / NT 4.0

coffeeCup Free FTP เป็นโปรแกรมส่งผ่านไฟล์ข้อมูลฝั่งไคลเอนท์ที่ใช้ง่าย สำหรับการอัพโหลดและดาว์นโหลดไฟล์ ไม่จำกัดจำนวนหมายเลข FTP account

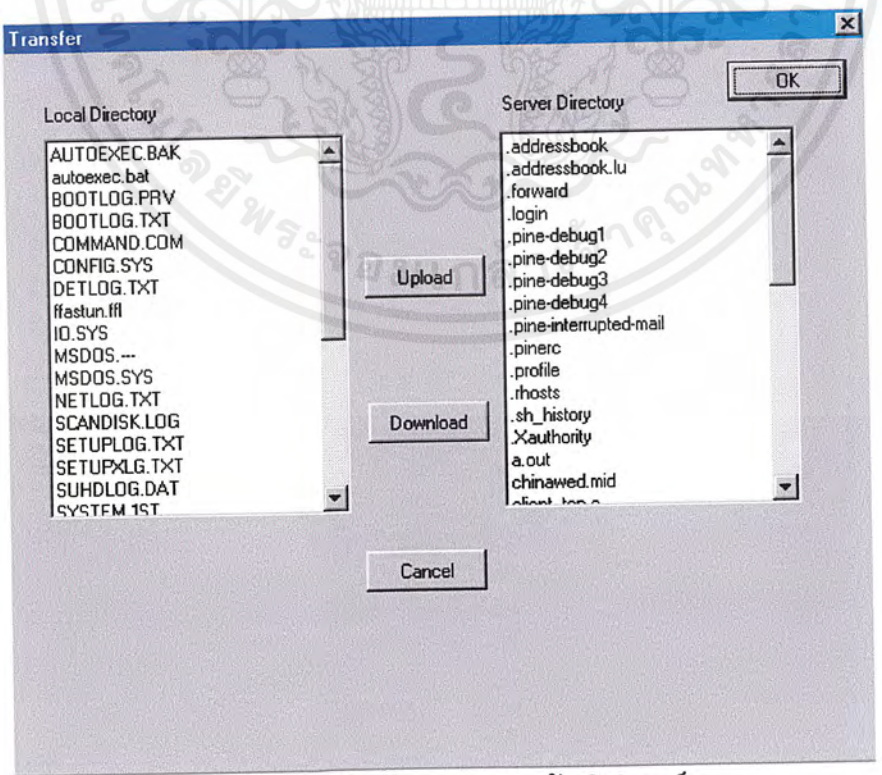
- ทำการดาวน์โหลดไฟล์ที่ขาดการติดต่อและส่งผ่านไฟล์ในรูปแบบของไบนารี ASCII หรือออโต้โหมด
- สามารถเลือกออโต้รีเฟรช (auto-refresh) ในแต่ละไฟล์ที่อัพโหลด

ซึ่งเมื่อทำการรันโปรแกรมที่ได้ทดลองเขียนจะปรากฏหน้าจอดังรูปที่ 4.1



รูปที่ 4.1 แสดงหน้าจอแรกเมื่อทำการรันโปรแกรม

เมื่อกดปุ่มก็ทำการเชื่อมต่อกับฝั่งเซิฟเวอร์ที่ต้องการแล้วจะปรากฏหน้าจอดังนี้



รูปที่ 4.2 แสดงหน้าจอหลังจาก connect เข้าเซิฟเวอร์

โปรแกรมที่เขียนขึ้นมาในการทดสอบขั้นต้นนี้สามารถทำการรับส่งไฟล์ข้อมูลได้ ต่อมาจึงทำการพัฒนาโปรแกรมโดยมีการใช้ Multithread ในการควบคุมการทำงานของโปรแกรมทำให้โปรแกรมสามารถทำงานได้หลายฟังก์ชันในเวลาเดียวกันได้ซึ่งเป็นขั้นตอนต่อไป

4.1.2 กำหนดหน้าที่และฟังก์ชันการทำงานหลักของโปรแกรม

โปรแกรมส่งผ่านไฟล์ข้อมูลฝั่งไคลเอนท์ จะทำหน้าที่รับส่งข้อมูลที่อยู่ในรูปของไฟล์ไปยังส่วนเซิฟเวอร์ โดยจะรองรับไฟล์ได้ทั้งประเภทASCII และ BINARY ตัวโปรแกรมจะมีฟังก์ชันต่างๆในการทำงานดังนี้

- สามารถรองรับคำสั่งที่เป็นมาตรฐานของ FTP Protocol ตามที่ระบุใน RFC959 ได้ครบถ้วน เช่น การรับไฟล์, ส่งไฟล์, การลบไฟล์, การเปลี่ยนชื่อไฟล์ เป็นต้น
- มีฟังก์ชันที่ช่วยในการรับส่งไฟล์ต่อในกรณีขาดการติดต่อ (connection fail) ก่อนที่จะทำการรับส่งไฟล์เสร็จ นั่นคือ การทำ resume downloading หรือ uploading
- มีฟังก์ชันทำงานสำหรับการเชื่อมต่อเข้าไปยังเซิฟเวอร์ใหม่ และดาว์นโหลดหรืออัพโหลดไฟล์ที่ทำค้างไว้อยู่โดยอัตโนมัติ (auto resume) เมื่อเกิดปัญหาขาดการติดต่อจากเซิฟเวอร์
- สามารถทำงานแบบ multitasking คือ ในขณะที่ทำการส่งผ่านไฟล์ ยังสามารถที่จะ browse site ที่เราเชื่อมต่ออยู่ และสามารถ browse local file system ของเราเองได้ด้วย ซึ่งในขณะเดียวกัน ก็สามารถที่จะเชื่อมต่อและรับส่งไฟล์กับเซิฟเวอร์อื่นได้พร้อมกัน
- มีส่วนติดต่อกับผู้ใช้ (User Interface) ที่สวยงาม ง่ายต่อการใช้งาน

4.2 เงื่อนไขบังคับการพัฒนาซอฟท์แวร์และโครงการ (Constraints)

4.2.1 สภาวะแวดล้อมของระบบ

1) โปรแกรมส่งผ่านไฟล์ข้อมูลฝั่งไคลเอนท์นี้ จะทำงานภายใต้ระบบปฏิบัติการ Windows NT ,Windows 98 หรือ Windows 95

2) การทำงานจะอยู่ภายใต้ระบบเครือข่าย (Network)

4.2.2 ผู้ใช้งานโปรแกรม

1) ผู้ใช้งานโปรแกรมเป็นผู้ที่มีความต้องการจะรับส่ง และจัดการงานต่างๆเกี่ยวกับไฟล์ ผ่านทางระบบเครือข่าย

2) ผู้ใช้งานโปรแกรมนี้ อาจจะไม่มีความชำนาญ หรือมีความรู้เกี่ยวกับระบบเครือข่าย และFTP Protocol มากนัก โปรแกรมจะต้องมีส่วนติดต่อกับผู้ใช้ (User Interface) ที่สวยงามและง่ายต่อการใช้งาน

4.2.3 ข้อจำกัด

เนื่องจากในท้องตลาดได้มีโปรแกรม FTP ส่วน Client ใช้กันอย่างแพร่หลาย จึงเกิดเป็นข้อจำกัดในการพัฒนาโปรแกรม ทำให้โปรแกรมที่พัฒนาขึ้นมาใหม่นี้ ต้องมีฟังก์ชันการทำงานพิเศษ ที่โปรแกรม FTP ในท้องตลาดทั่วไปยังไม่มี และฟังก์ชันที่พิเศษที่จะกำหนดขึ้นต้องสอดคล้องกับความเป็นไปได้ของโปรแกรม แต่เนื่องจากฟังก์ชันพิเศษนี้ต้องเป็นสิ่งใหม่ที่โปรแกรมทั่วไปยังไม่มี ดังนั้นแหล่งข้อมูลที่จะนำมาศึกษาค้นคว้าจึงหาได้ลำบาก

บทที่ 5
ผลการวิจัย

เมื่อได้ทำการทดลองเขียนโปรแกรมรับส่งไฟล์แบบพื้นฐาน คือ สามารถเชื่อมต่อไปยังเซิฟเวอร์ สามารถตรวจดูรายละเอียดภายในแต่ละไดเรกทอรี่ของเซิฟเวอร์ และสามารถทำการรับส่งไฟล์อย่างง่าย ทำให้มีความเข้าใจหลักการทำงาน เข้าใจวิธีใช้คลาสและฟังก์ชันหลักๆ มากขึ้น เพื่อนำมาเป็นพื้นฐานในการเขียนโปรแกรมรับส่งไฟล์ที่มีลักษณะซับซ้อนยิ่งขึ้นได้

ในโปรแกรมรับส่งไฟล์ที่ได้ทำการเขียนจะเพิ่มจุดเด่นขึ้นมา คือ สามารถทำการรับส่งไฟล์ได้หลายไฟล์ จากหลายเซิฟเวอร์พร้อมๆกัน โดยไม่จำเป็นต้องเปิดโปรแกรมขึ้นมาใหม่หลายหน้าจอ และจะมีการทำเชื่อมต่อกลับเข้าไปยังเซิฟเวอร์โดยอัตโนมัติ แล้วรับหรือส่งไฟล์ต่อจากที่ทำค้างไว้ เมื่อพบว่าขาดการติดต่อจากเซิฟเวอร์ในขณะที่กำลังรับหรือส่งไฟล์อยู่

ลักษณะของโปรแกรมส่งผ่านไฟล์ข้อมูลที่ได้ทำการเขียนและพัฒนาสำเร็จได้ตามจุดมุ่งหมายที่วางไว้เป็นดังนี้

5.1 ความต้องการพื้นฐานของคอมพิวเตอร์

5.1.1 ความต้องการทางด้านฮาร์ดแวร์

- คอมพิวเตอร์รุ่น Pentium100 ขึ้นไป
- หน่วยความจำไม่ต่ำกว่า 16 MB
- ฮาร์ดดิสก์ 10 MB ขึ้นไป
- เครื่องโมเด็มความเร็ว 14.4 mbps ขึ้นไป

5.1.2 ความต้องการทางด้านซอฟท์แวร์

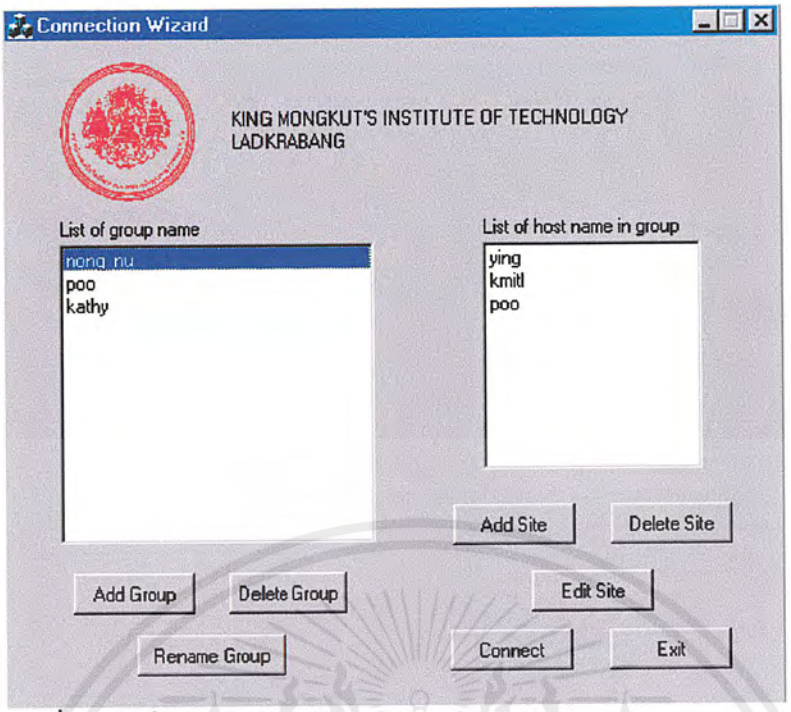
- ระบบปฏิบัติการ Window 95 , 98 หรือ NT

5.2 ขั้นตอนในการติดตั้งโปรแกรม

1) ลงโปรแกรมรับส่งไฟล์ฝั่งไคลเอนท์ในเครื่อง
2) ทำการรันโปรแกรม

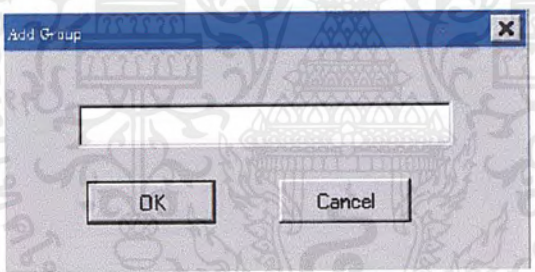
5.3 การทำงานของโปรแกรมส่งผ่านข้อมูล

- เมื่อรันโปรแกรมสำเร็จจะแสดงหน้าจอดังรูปที่ 5.1



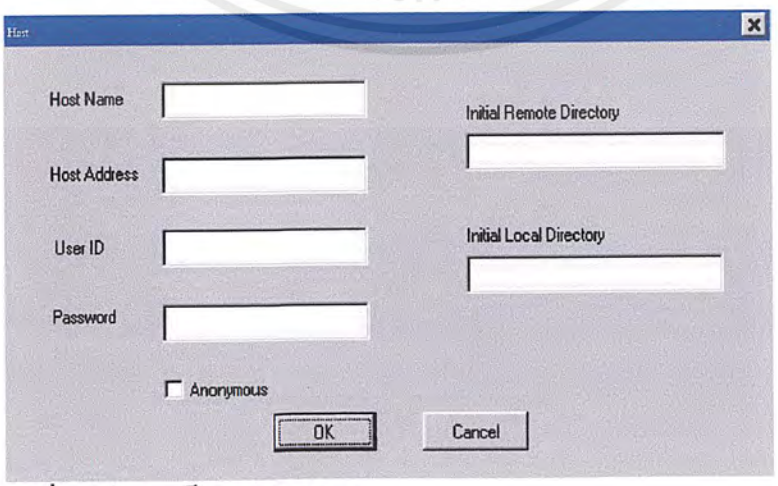
รูปที่ 5.1 หน้าจอแรกของโปรแกรม FTP Client

- เมื่อกดปุ่ม Add Group จะแสดงหน้าจอรับชื่อกลุ่มดังนี้



รูปที่ 5.2 ไดอะล็อก Add Group

- เมื่อกดปุ่ม Add site ,Edit Site โปรแกรมจะแสดงหน้าจอและรับข้อมูลของเซิฟเวอร์ดังนี้



รูปที่ 5.3 ไดอะล็อก Add Host

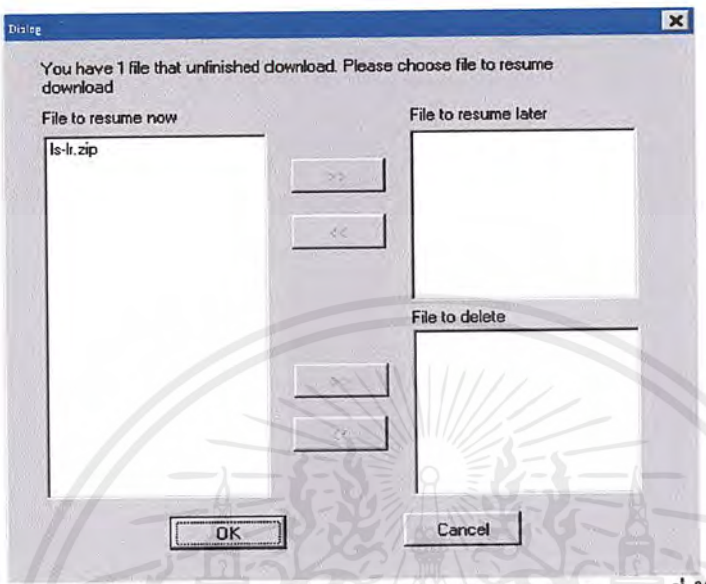
การทำงานของโปรแกรมส่งผ่านข้อมูลฝั่งไคลเอนท์ในหน้าจอแรกเป็นดังนี้

ตารางที่ 5.1 แสดงฟังก์ชันการทำงานของโปรแกรมในหน้าจอแรก

คำสั่งจากผู้ใช้	หน้าที่ของโปรแกรม	แสดงผล
กดปุ่ม Add Group เพิ่มกลุ่มของเซิฟเวอร์	เก็บค่ากลุ่มของเซิฟเวอร์ที่รับมาใส่ในบัฟเฟอร์	หน้าจอรับชื่อกลุ่มและกดปุ่ม OK แล้วจะแสดงชื่อบนหน้าจอเดิม
กดปุ่ม Delete Group เลือกกลุ่มเซิฟเวอร์ที่ต้องการลบ	ลบกลุ่มของเซิฟเวอร์	ไม่มีชื่อกลุ่มที่เลือก
กดปุ่ม Rename Group เลือกและใส่ชื่อกลุ่มใหม่	เปลี่ยนชื่อกลุ่มใหม่	แสดงชื่อกลุ่มใหม่
กดปุ่ม Add Site เพิ่มเซิฟเวอร์ในกลุ่ม โดยก่อนกดปุ่มนี้ ต้องเลือกกลุ่มของเซิฟเวอร์ก่อน	เก็บรายละเอียดของแต่ละเซิฟเวอร์	แสดงหน้าจอรับรายละเอียดของเซิฟเวอร์ เมื่อกดปุ่ม OK แล้วจะแสดงชื่อเซิฟเวอร์บนหน้าจอเดิม
กดปุ่ม Delete Site เลือกเซิฟเวอร์ที่ต้องการลบ	ลบข้อมูลของเซิฟเวอร์นั้นๆ ออก	ไม่มีชื่อเซิฟเวอร์ที่เลือก
กดปุ่ม Rename Site เลือกและใส่ชื่อเซิฟเวอร์ใหม่	เปลี่ยนชื่อเซิฟเวอร์ใหม่	แสดงชื่อเซิฟเวอร์ใหม่
กดปุ่ม Edit Site เลือกเซิฟเวอร์ที่ต้องการดูและแก้ไขข้อมูล	แสดงและแก้ไขข้อมูลของเซิฟเวอร์	แสดงข้อมูลของเซิฟเวอร์
กดปุ่ม Connect เลือกกลุ่มของเซิฟเวอร์ที่ต้องการเชื่อมต่อ	เปลี่ยนหน้าไปเป็นหน้าจอ FTP และแสดงชื่อเซิฟเวอร์ต่างๆที่อยู่ในกลุ่มที่เลือกไว้ในลิสต์ในหน้าจอ FTP	แสดงหน้าจอ FTP
กดปุ่ม Exit ออกจากโปรแกรม	ปิดโปรแกรม	ปิดโปรแกรม

กรณีที่ผู้ใช้ใส่ข้อมูลไม่เรียบร้อยหรือเกิดปัญหาโปรแกรมจะขึ้น Message Box แจ้งผู้ใช้ทุกครั้งว่าผิดอย่างไรผู้ใช้จะรู้ได้

- เมื่อกดปุ่ม CONNECT แล้วจะมีการแสดงหน้าจอขึ้นมาว่ามีไฟล์ไหน ที่มีการดาวน์โหลดและอัพโหลดค้างไว้ โดยให้เลือกว่าจะทำการดาวน์โหลด หรืออัพโหลดต่อจากเดิมในขณะนั้นเลย หรือจะให้ทำในคราวต่อไป หรือจะไม่ทำต่ออีกเลย



รูปที่ 5.4 **หน้าจอให้เลือกว่าจะดาวน์โหลด หรือ อัพโหลด ต่อจากที่ค้างไว้เลยหรือไม่**

การทำงานของปุ่มต่างๆ ในหน้าจอทำการดาวน์โหลดหรืออัพโหลดต่อจากที่ค้างไว้เป็นดังตาราง 5.2

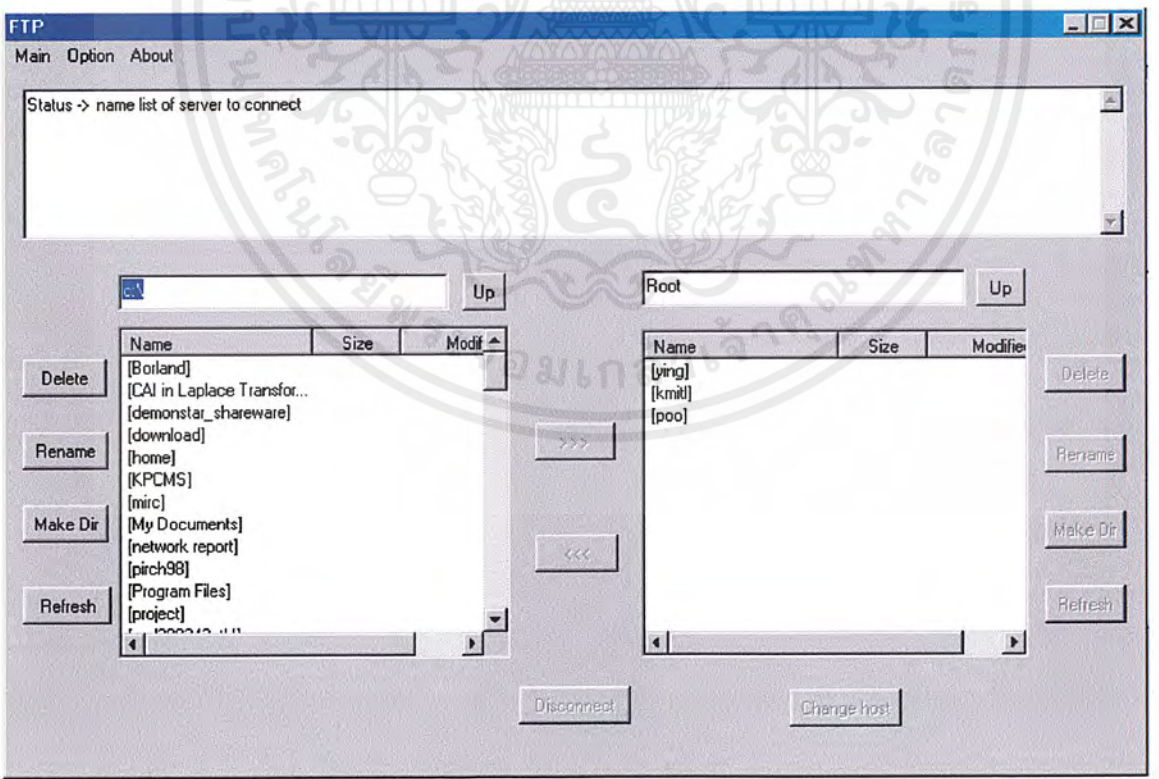
ตารางที่ 5.2 **แสดงฟังก์ชันการทำงานของโปรแกรมในหน้าจอ** resume download / upload

คำสั่งจากผู้ใช้	หน้าที่ของโปรแกรม	แสดงผล
กดปุ่ม >> อันบน เลือกไฟล์ที่จะให้ถามอีกทีเมื่อเปิดโปรแกรมครั้งหน้า	ย้ายชื่อไฟล์จากลิสต์ File to resume now ไปไว้ในลิสต์ File to resume later	ชื่อไฟล์อยู่ในลิสต์ File to resume later
กดปุ่ม << อันบน เลือกไฟล์ที่จะให้ดาวน์โหลดหรืออัพโหลดต่อจากเดิม ในตอนนี้เลย	ย้ายชื่อไฟล์จากลิสต์ File to resume later ไปไว้ในลิสต์ File to resume now	ชื่อไฟล์อยู่ในลิสต์ File to resume now
กดปุ่ม >> อันล่าง เลือกไฟล์ที่จะไม่ทำการดาวน์โหลด หรืออัพโหลดต่อจากเดิมอีกเลย	ย้ายชื่อไฟล์จากลิสต์ File to resume now ไปไว้ในลิสต์ File to delete	ชื่อไฟล์อยู่ในลิสต์ File to delete

ตารางที่ 5.2 (ต่อ) แสดงฟังก์ชันการทำงานของโปรแกรมในหน้าจอ resume download/upload

คำสั่งจากผู้ใช้	หน้าที่ของโปรแกรม	แสดงผล
กดปุ่ม << อันล่าง เลือกไฟล์ที่จะให้ดาวน์โหลดหรืออัพโหลดต่อจากเดิม ในตอนนี้เลย	ย้ายชื่อไฟล์จากลิสต์ File to delete ไปไว้ในลิสต์ File to resume now	ชื่อไฟล์อยู่ในลิสต์ File to resume now
กดปุ่ม OK ตกลง ตามที่เลือกไว้ในลิสต์ต่างๆ	ทำการอัพโหลดหรือดาวน์โหลดไฟล์ตามลิสต์ที่เลือกไว้	แสดงไดอะล็อกการอัพโหลดหรือดาวน์โหลดไฟล์ตามลิสต์ที่เลือกไว้
กดปุ่ม Cancel ยกเลิกที่เลือกไว้ทุกอย่างในลิสต์	ไม่ทำการใดๆเลย แต่จะเปลี่ยนไปยังหน้าจอ FTP เลย	แสดงหน้าจอ FTP

- เมื่อทำการเลือกไฟล์ที่ให้ทำการดาวน์โหลด/อัพโหลดต่อเสร็จแล้ว จะเกิดหน้าจอหลักของโปรแกรม ขึ้นดังรูป 5.5



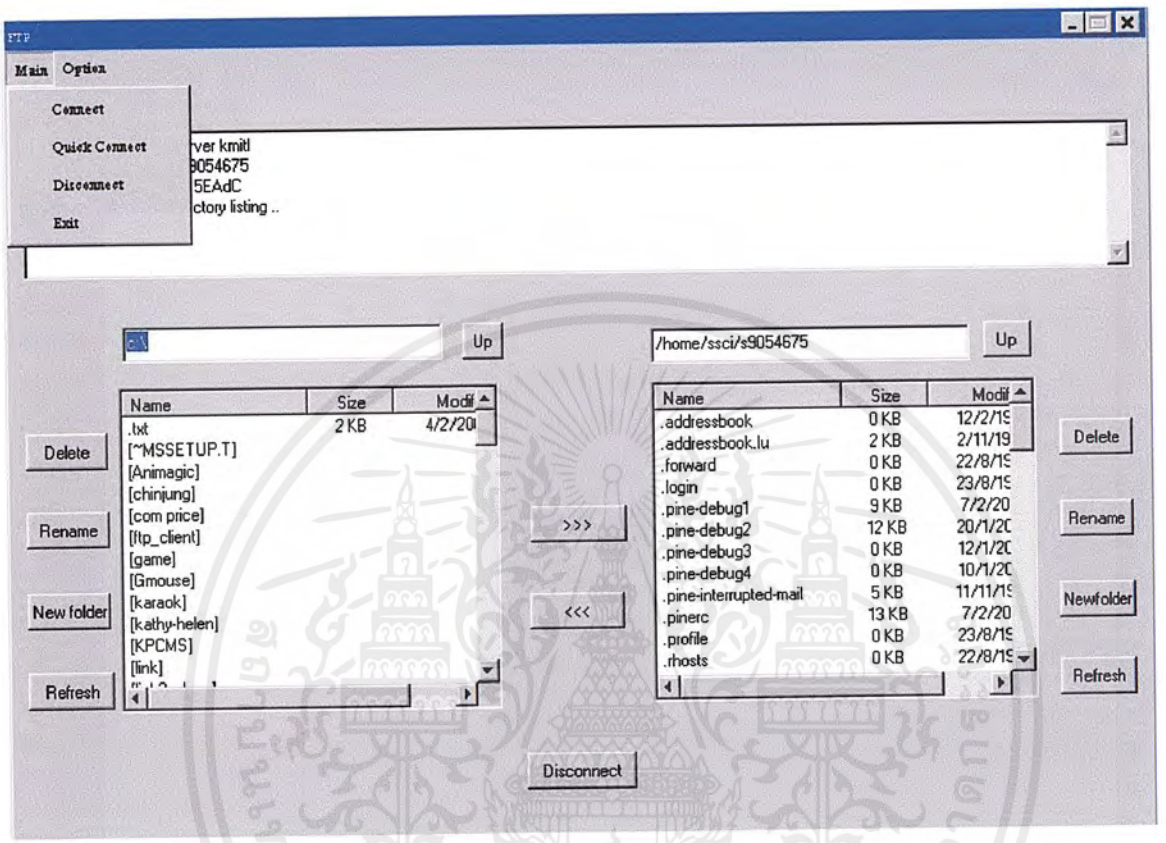
รูป 5.5 หน้าจอ FTP เป็นส่วนทำเกี่ยวกับการรับส่งไฟล์จริงๆ

การทำงานของปุ่มต่างๆ หน้าจอ FTP เป็นดังตาราง 5.3

ตารางที่ 5.3 แสดงฟังก์ชันการทำงานของปุ่มต่าง ๆในหน้าจอหลักของโปรแกรม

คำสั่งจากผู้ใช้	หน้าที่ของโปรแกรม	แสดงผล
กดปุ่ม Upload เลือกไฟล์ที่จะอัพโหลดและระบุไดเร็กทอรีปลายทางฝั่งเซิฟเวอร์ที่จะเก็บไฟล์	คัดลอกและส่งข้อมูลบันทึกไฟล์ลงบนเซิฟเวอร์	แสดงรายละเอียดของการอัพโหลดบนโพรเกสบาร์ไดอะล็อก
กดปุ่ม Download ชื่อไฟล์ที่ต้องการดาวน์โหลดและระบุไดเร็กทอรีปลายทางฝั่งไคลเอนท์ที่จะเก็บไฟล์	คัดลอกและส่งข้อมูลบันทึกกลับมาฝั่งไคลเอนท์	แสดงรายละเอียดของการอัพโหลดบนโพรเกสบาร์ไดอะล็อก
กดปุ่ม Delete เลือกไฟล์ที่ต้องการลบฝั่งไคลเอนท์หรือเซิฟเวอร์	ลบไฟล์ทิ้ง	ไม่มีไฟล์นั้น
กดปุ่ม Rename เลือกและใส่ชื่อไฟล์ที่ต้องการเปลี่ยนชื่อฝั่งไคลเอนท์หรือเซิฟเวอร์	เปลี่ยนชื่อไฟล์	แสดงชื่อไฟล์ใหม่
กดปุ่ม New Folder ใส่ชื่อไดเร็กโทรี่ใหม่	สร้างไดเร็กทอรี่ใหม่	แสดงไดเร็กทอรี่ใหม่
กดปุ่ม Refresh	ปรับหน้าจอให้แสดงข้อมูลปัจจุบัน	แสดงข้อมูลปัจจุบัน
กดปุ่ม Up	เข้าไปยังไดเร็กทอรี่ที่อยู่สูงกว่าเดิม 1 ขั้น	แสดงไฟล์ที่อยู่ในไดเร็กทอรี่ที่อยู่สูงกว่าเดิม 1 ขั้น
กดปุ่ม Disconnect	ยกเลิกการเชื่อมต่อ	กลับไปแสดงลิสต์ของชื่อเซิฟเวอร์ในกลุ่มที่เราเลือก
กดปุ่ม Change host	กลับไปแสดงลิสต์ของชื่อเซิฟเวอร์ในกลุ่มที่เราเลือก	กลับไปแสดงลิสต์ของชื่อเซิฟเวอร์ในกลุ่มที่เราเลือก
Double click List	เข้าไปยังไดเรกทอรี่ที่เลือก	แสดงไฟล์ในไดเรกทอรี่ที่เลือก

- เมื่อกดเลือก main menu ที่ menu bar ของหน้าจอหลัก จะแสดงคำสั่งเกี่ยวกับการ connect เข้าสูเซิฟเวอร์ดังรูปที่ 5.6 และคำอธิบายการทำงานของคำสั่งแสดงในตารางที่ 5.4

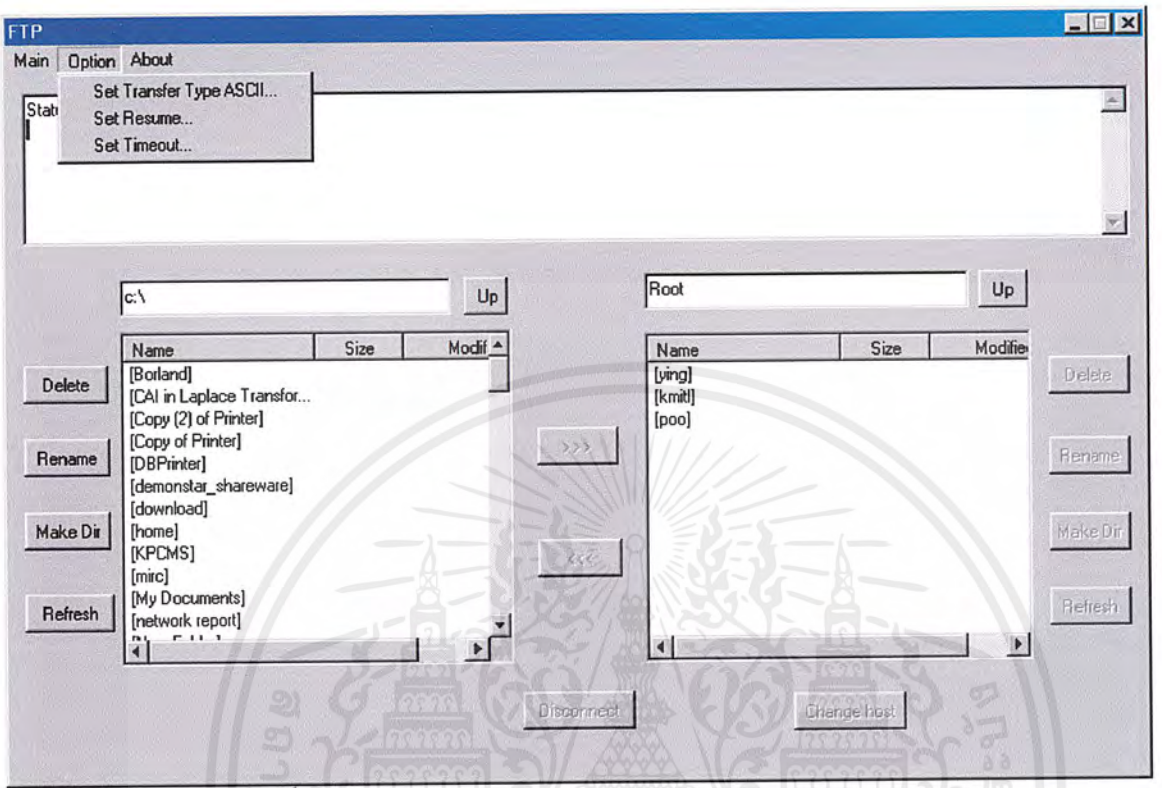


รูปที่ 5.6 **แสดงคำสั่งเมื่อคลิก** main menu

ตารางที่ 5.4 **ฟังก์ชันการทำงานของคำสั่งใน** main menu

คำสั่งจากผู้ใช้	หน้าที่ของโปรแกรม	แสดงผล
เลือกคำสั่ง Connect	ทำการเชื่อมต่อกับเซิฟเวอร์ที่ได้กำหนดไว้ในลิสต์	แสดงสถานะการเชื่อมต่อไปยังเซิฟเวอร์
เลือกคำสั่ง Quick Connect	ให้ผู้ใช้กำหนดการเชื่อมต่อขึ้นใหม่เอง	แสดงไดอะล็อก ให้ใส่แอดเดรสของเซิฟเวอร์ login และ password
เลือกคำสั่ง DisConnect	ปิดการเชื่อมต่อกับเซิฟเวอร์ที่เชื่อมต่ออยู่	แสดงสถานะปิดการเชื่อมต่อ
เลือกคำสั่ง Exit	ออกจากโปรแกรม	ปิดหน้าจอ

- และเมื่อคลิก option menu ใน menu bar จะแสดงคำสั่งเกี่ยวกับการ set option ของโปรแกรม แสดงดังรูป 5.7

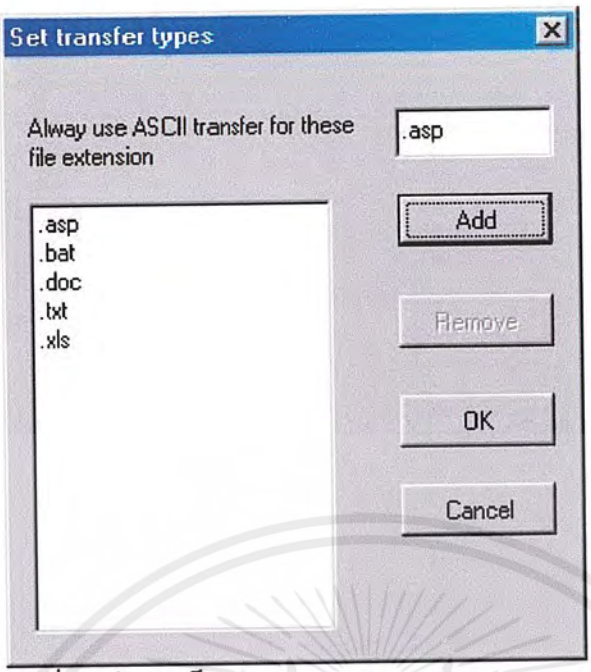


รูปที่ 5.7 แสดงคำสั่งเมื่อคลิก option menu

ตารางที่ 5.5 ฟังก์ชันการทำงานของคำสั่งใน option menu

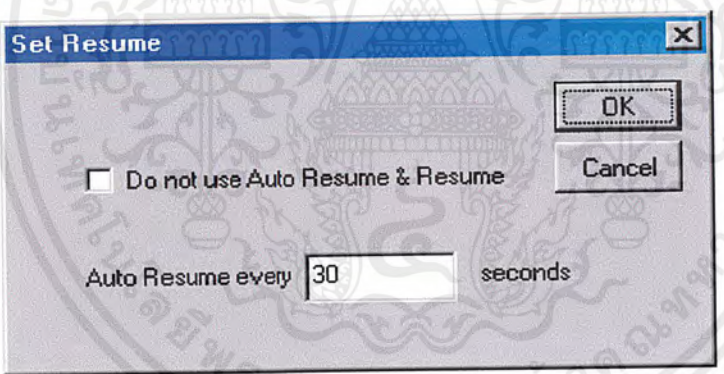
คำสั่งจากผู้ใช้	หน้าที่ของโปรแกรม	แสดงผล
เลือกคำสั่ง Set Transfer Type ASCII	กำหนดให้ชนิดของไฟล์ที่ผู้ใช้กำหนดรับส่งข้อมูลแบบ ASCII	แสดงไดอะล็อก ให้กำหนด นามสกุลของไฟล์ที่ต้องการโหลด แบบ ASCII
เลือกคำสั่ง Set Resume	กำหนดช่วงเวลาในการตรวจเช็คการติดต่อกับเซิฟเวอร์ auto resume และ resume	แสดงไดอะล็อก ให้ผู้ใช้กำหนดช่วงเวลาสำหรับ auto resume และ resume
เลือกคำสั่ง Set Timeout	กำหนดเวลาสำหรับการเชื่อมต่อแต่ละเซิฟเวอร์	แสดงไดอะล็อก ให้ผู้ใช้กำหนดเวลา timeout

- เมื่อเลือกเมนู Option เลือก Set Transfer Type ASCII จะแสดงไดอะล็อกเพื่อให้ผู้ใช้กำหนดชนิดของไฟล์ที่จะรับส่ง ให้เป็น ASCII ดังรูปที่ 5.8



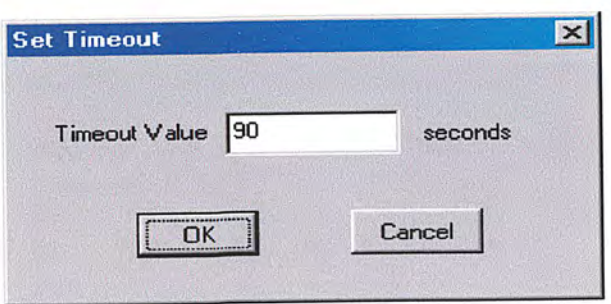
รูปที่ 5.8 **ไดอะล็อก** Set transfer types

- เมื่อเลือกเมนู Option เลือก Set Resume จะแสดงไดอะล็อกเพื่อให้ผู้ใช้กำหนดลักษณะของการ resume ดังรูปที่ 5.9



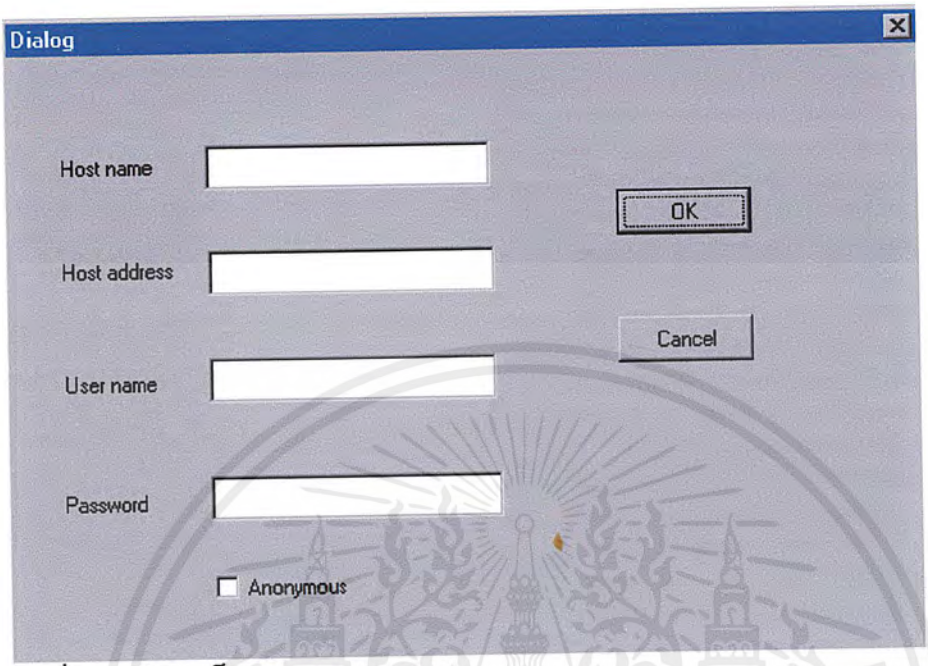
รูปที่ 5.9 **ไดอะล็อก** Set Resume

- เมื่อเลือกเมนู Option เลือก Set Time Out จะแสดงไดอะล็อกเพื่อให้ผู้ใช้กำหนดเวลา timeout สำหรับโปรแกรม ดังรูปที่ 5.10



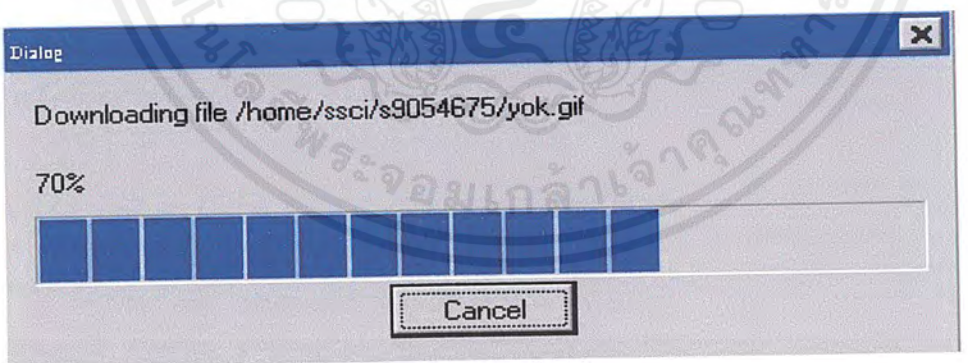
รูปที่ 5.10 **ไดอะล็อก** Set Timeout

- ถ้าต้องการที่จะ connect เข้าสู่เซิฟเวอร์โดยไม่กลับไปหน้าจอแรก สามารถทำได้โดยคลิกที่เมนู main เลือก Quick Connect จะแสดงไดอะล็อก ดังรูป 5.11



รูปที่ 5.11 ไดอะล็อก Quick Connect

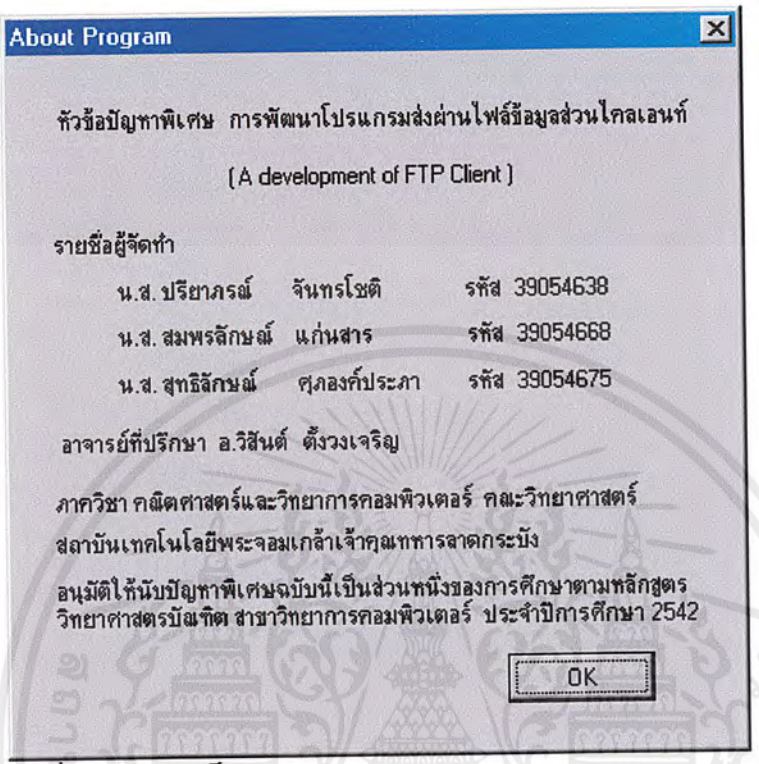
- เมื่อทำการดาว์นโหลดหรืออัพโหลดไฟล์ จะแสดงไดอะล็อกเพื่อแสดงสถานะของการดาวน์โหลดดังรูปที่ 5.12



รูป 5.12 ไดอะล็อก progress ของการโหลดไฟล์

ในไดอะล็อก Progress ของการโหลดไฟล์นี้ จะแสดงสถานะของการโหลดไฟล์ ชื่อไฟล์ และเปอร์เซนต์ในการโหลดไฟล์ โดยในขณะที่ทำการโหลดไฟล์จะสามารถยกเลิกการโหลดได้ตลอดเวลาโดยการกดปุ่ม Cancel

ถ้าต้องการทราบเกี่ยวกับตัวโปรแกรม เช่น ชื่อทีมงานที่จัดทำหรือลิขสิทธิ์ซึ่งเป็นของทางสถาบัน สามารถดูได้โดย ไปที่ about menu แล้วเลือก about program จะแสดงไดอะล็อกดังรูป



รูปที่ 5.13 ไดอะล็อก about program

หลังจากทำการดาว์นโหลดหรืออัพโหลดเสร็จผู้ใช้สามารถออกจากโปรแกรมโดยไปที่ main menu แล้วเลือก Exit โปรแกรมจะจบการทำงาน

บทที่ 6
สรุปผลและข้อเสนอแนะ

6.1 สรุปผล

การทำปัญหาพิเศษนี้ วัตถุประสงค์ของการเรียนรู้ของทีมงาน คือ เรียนรู้เกี่ยวกับโพรโตคอล TCP/IP, FTP และเครื่องมือหรือโปรแกรมภาษาที่ใช้ในการพัฒนา เพื่อนำความรู้มาสร้างเป็นโปรแกรมที่สามารถใช้งานได้จริง มีความโดดเด่นอยู่ในตัวงาน และเพื่อให้เป็นพื้นฐานความรู้ทางด้านระบบเครือข่าย โดยคาดหวังให้สามารถนำความรู้ไปประยุกต์ใช้ในภายภาคหน้าได้

การศึกษาเก็บรวบรวมข้อมูลเพื่อใช้เป็นแนวทางในการออกแบบและพัฒนาโปรแกรม ได้ทำการค้นคว้าจากหนังสือในห้องสมุด หรือหาเพิ่มเติมจากอินเตอร์เนตซึ่งเป็นแหล่งความรู้ที่มีประโยชน์มาก พร้อมทั้งทำการศึกษาทดลองใช้โปรแกรมรับส่งไฟล์ข้อมูลที่มีอยู่ในท้องตลาด เพื่อศึกษาฟังก์ชันการทำงานหลักๆ และค้นหาจุดบกพร่องหรือสิ่งที่ขาดหายในแต่ละโปรแกรม แล้วนำมาพัฒนาให้เป็นจุดเด่นในโปรแกรมรับส่งไฟล์ข้อมูลที่จะพัฒนาขึ้น

จุดเด่นที่นำมาพัฒนาคือ การทำงานแบบ multitasking ซึ่งเป็นคุณสมบัติที่ช่วยให้โปรแกรมสามารถทำงานหลายๆ งานในเวลาพร้อมๆ กันหรือใกล้เคียงกันได้อย่างมีประสิทธิภาพยิ่งขึ้น จากการศึกษาพบว่า thread สามารถจัดการกับการทำงานแบบ multitasking ได้เป็นอย่างดี จึงนำ thread เข้ามาช่วยในการจัดการงานต่างๆ ที่ต้องการการประมวลผลในเวลาเดียวกัน เช่น นำ thread มาใช้เป็นตัวจัดการการดาวน์โหลดหรืออัพโหลดไฟล์แต่ละไฟล์ ดังนั้นผู้ใช้สามารถทำการดาวน์โหลดหรืออัพโหลดไฟล์ได้หลายๆ ไฟล์ในเวลาเดียวกัน

และจุดเด่นอีกข้อที่นำมาพัฒนาคือ การทำงานแบบ multihosting หรือความสามารถในการ connect เข้าสู่หลายๆ เซิฟเวอร์ โดยปกติในโปรแกรมทั่วๆ ไปถ้าผู้ใช้ต้องการ connect เข้าอีกเซิฟเวอร์ จะต้องเปิดโปรแกรมขึ้นมาอีกหน้าจอ จึงถือเป็นการสิ้นเปลืองทรัพยากรและเสียเวลาในการกำหนดค่าใหม่ให้กับหน้าจอที่สอง ดังนั้นโปรแกรมที่ทำการพัฒนาขึ้นมา ผู้ใช้สามารถกำหนดเซิฟเวอร์ที่สองและที่สามได้ภายในกลุ่ม เมื่อทำการ connect เซิฟเวอร์ทั้งหมดที่ได้กำหนดไว้จะถูกแสดงภายในหน้าจอเดียวกัน และผู้ใช้สามารถคลิ้กเข้าสู่แต่ละเซิฟเวอร์ได้ตลอดเวลา

ปัญหาที่พบในระหว่างพัฒนาโปรแกรม เกิดเนื่องมาจากทางทีมงานไม่มีความรู้ความชำนาญในโปรแกรมภาษา Visual C++ รวมทั้งการสร้างฟังก์ชันพิเศษที่นำมาพัฒนาเป็นจุดเด่นของโปรแกรมไม่มีใช้งานอยู่ในโปรแกรมรับส่งไฟล์ข้อมูลที่มีอยู่ทั่วไป จึงทำให้การพัฒนาเป็นไปอย่างล่าช้า และอาจจะยังไม่สมบูรณ์แบบ

6.2 ข้อเสนอแนะ

โปรแกรมรับส่งไฟล์ข้อมูลส่วนไคลเอนท์นี้ยังสามารถพัฒนาให้มีความสามารถได้อีกหลายข้อ ทางทีมงานจึงมีข้อเสนอแนะเกี่ยวกับฟังก์ชันเพิ่มเติมสำหรับผู้ที่ต้องการจะนำไปพัฒนาต่อ ดังนี้

1. สร้างฟังก์ชันสำหรับการบีบอัดข้อมูล โดยการกำหนดให้ไฟล์ที่ดาวน์โหลดมานั้น ถูกเก็บในลักษณะของไฟล์ที่บีบอัดข้อมูล เช่น เป็นไฟล์ที่นามสกุล .zip เพื่อประหยัดเนื้อที่ของหน่วยความจำ
2. สร้างฟังก์ชัน virtual path โดยผู้ใช้สามารถกำหนด virtual path แทนไดเร็กทอรี่ที่ผู้ใช้มีความจำเป็นต้องเข้าไปดาวน์โหลดหรืออัพโหลดไฟล์บ่อยๆ ได้ ซึ่งถือเป็นการช่วยอำนวยความสะดวกแก่ผู้ใช้

นอกจากนี้ อาจทำการพัฒนาโปรแกรมรับส่งไฟล์ข้อมูลบนระบบปฏิบัติการอื่นๆ เช่น ระบบปฏิบัติการ UNIX ซึ่งนับวันจะยิ่งมีบทบาทและมีผู้นิยมใช้มากขึ้น รวมทั้งระบบปฏิบัติการ UNIX สามารถสนับสนุนโปรแกรมที่ทำงานแบบ multitasking ได้เป็นอย่างดี เนื่องจากมีฟังก์ชันช่วยในการจัดการงานของ thread ทำให้ thread ทำงานได้อย่างมีประสิทธิภาพยิ่งขึ้น

ภาคผนวก ก อธิบายคลาสและฟังก์ชันในโปรแกรม

ตารางที่ ก แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CAddGroup	CAddGroup	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	virtual void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnOk	virtual void	void	ยอมรับการเพิ่มกรุ๊ป แล้วเปลี่ยนกลับไปไดอะล็อกหลัก
CAddHost	CAddHost	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	virtual void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnAnonymous	afx_msg void	void	ตั้งค่ากรณีที่เป็น Anonymous
	OnOK	virtual void	void	ยอมรับการเพิ่มเซิฟเวอร์ แล้วเปลี่ยนกลับไปไดอะล็อกหลัก
CAskResume	CAskResume	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	OnBackL	afx_msg void	void	ย้ายชื่อไฟล์ที่อยู่ในรายการที่จะ resume ภายหลัง ให้กลับมา resume ในขณะนั้น
	OnBackN	afx_msg void	void	ย้ายไฟล์ที่อยู่ในรายการที่จะไม่ resume อีกเลย ให้กลับมา resume ในขณะนั้น
	OnOK	virtual void	void	รับค่าต่างๆเป็นชื่อไฟล์ต่างๆ เกี่ยวกับการ resume แล้วกลับไปยังไดอะล็อกหลัก
	OnResL	afx_msg void	void	ย้ายชื่อไฟล์ที่อยู่ในรายการที่จะ resume ในขณะนั้น ให้ไป resume ในภายหลัง
	OnResN	afx_msg void	void	ย้ายชื่อไฟล์ที่อยู่ในรายการที่จะ resume ในขณะนั้น ให้ไม่ resume อีกเลย
	OnSetfocusList1	afx_msg void	void	ทำการกำหนดว่าปุ่มไหนในไดอะล็อกจะใช้ได้บ้าง เมื่อทำการใช้เม้าส์คลิ๊กไปยังรายการที่จะ resume ในขณะนั้น
	OnSetfocusList2	afx_msg void	void	ทำการกำหนดว่าปุ่มไหนในไดอะล็อกจะใช้ได้บ้าง เมื่อทำการใช้เม้าส์คลิ๊กไปยังรายการที่จะ resume ในภายหลัง
	OnSetfocusList3	afx_msg void	void	ทำการกำหนดว่าปุ่มไหนในไดอะล็อกจะใช้ได้บ้าง เมื่อทำการใช้เม้าส์คลิ๊กไปยังรายการที่จะไม่ resume อีกเลย

ตารางที่ ก (ต่อ) แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CFileExist	CFileExist	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	OnInitDialog	virtual BOOL	void	กำหนดค่าต่างๆ เมื่อสร้างไดอะล็อก
	OnOK	virtual void	void	รับชื่อไฟล์ที่ต้องการเปลี่ยนใหม่ แล้วกลับไปยังไดอะล็อกหลัก
CFtp	CFtp	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	Connect	void	void	ทำการเชื่อมต่อไปยังเซิฟเวอร์
	GetFTPList	void	void	แสดงไฟล์ที่อยู่ในไดเร็กทอรี่ของเซิฟเวอร์ในขณะนั้น
	GetList	void	void	แสดงไฟล์ที่อยู่ในไดเร็กทอรี่ของไคลเอนท์ในขณะนั้น
	OnDblclkList2	afx_msg void	NMHDR* pNMHDR, LRESULT* pResult	เชื่อมต่อเข้าไปยังเซิฟเวอร์ที่ต้องการ หรือเข้าไปยังไดเร็กทอรี่ของเซิฟเวอร์ที่ทำการเชื่อมต่อแล้ว ทำเมื่อผู้ใช้โปรแกรม ดับเบิ้ลคลิ้กที่ชื่อไดเร็กทอรี่ ในลิสต์ของเซิฟเวอร์
	OnDisconnect	afx_msg void	void	ยกเลิกการเชื่อมต่อกับเซิฟเวอร์
	OnDownload	afx_msg void	void	ส่งค่าการดาวน์โหลดไฟล์ไปคลาส CsetDownload
	OnInitDialog	virtual BOOL	void	กำหนดค่าต่างๆ เมื่อสร้างไดอะล็อก
	OnLocalDelete	afx_msg void	void	ลบไฟล์ในส่วนไคลเอ็นท์
	OnLocalRename	afx_msg void	void	เปลี่ยนชื่อไฟล์ในส่วนไคลเอ็นท์
	OnLocalUp	afx_msg void	Void	อัพไดเร็กทอรี่ในส่วนไคลเอ็นท์ 1 ขั้น
	OnMainExit	afx_msg void	void	ออกจากโปรแกรม
	OnMainQuick connect	afx_msg void	void	แสดงไดอะล็อกสำหรับการเชื่อมต่อ 1 เซิฟเวอร์
	OnMake	afx_msg void	void	สร้างไดเรกโทรี่ใหม่ขึ้นในเซิฟเวอร์
	OnServerDelete	afx_msg void	void	ลบไฟล์ในเซิฟเวอร์
	OnServerRename	afx_msg void	void	เปลี่ยนชื่อไฟล์ในเซิฟเวอร์
	OnServerUp	afx_msg void	void	อัพไดเร็กทอรี่ในส่วนเซิฟเวอร์ 1 ขั้น
	OnUpload	afx_msg void	void	ส่งค่าการอัพโหลดไฟล์ไปคลาส CsetUpLoad

ตารางที่ ก (ต่อ) แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CMakeDir	CMakeDir	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
CProgressDlg	CheckCancelButton	BOOL	void	ตรวจดูว่าปุ่ม cancel ของไดอะล็อกที่แสดงโพรเกรสบาร์ มีการกดหรือไม่
	CProgressDlg	void	UINT nCaptionID	คอนตรักเตอร์ของคลาส CprogressDlg
	~CProgressDlg	void	void	เดสตรักเตอร์ของคลาส CprogressDlg
	Create	BOOL	CWnd *pParent	สร้างไดอะล็อกที่แสดงโพรเกรสบาร์
	DestroyWindow	virtual BOOL	void	ทำลายไดอะล็อกที่แสดงโพรเกรสบาร์
	DoDataExchange	virtual void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnCancel	virtual void	void	ตั้งค่าตัวแปร เมื่อมีการกดปุ่ม cancel ในไดอะล็อกโพรเกรสบาร์
	OnInitDialog	virtual BOOL	void	กำหนดค่าต่างๆ เมื่อสร้างไดอะล็อก
	PumpMessage	void	void	รับแมสเสจจากไดอะล็อก
	ReEnableParent	void	void	ทำให้ไดอะล็อกที่เรียกทำคลาสนี้สามารถทำงานได้
	SetPos	void	unsigned int so_far	ขยับแถบสีในโพรเกรสบาร์ตามที่ต้องการ
	SetStatus	void	LPCTSTR lpszMessage	แสดงรายละเอียดบนไดอะล็อกโพรเกรสบาร์ในขณะที่ทำการรับส่งไฟล์
	SetUpper	void	unsigned int nUpper	ตั้งค่าให้โพรเกรสบาร์ว่าจะแบ่งเป็นกี่ส่วน
	UpdatePercent	void	int nCurrent	แสดงว่าขณะนั้น รับหรือส่งไฟล์มากี่เปอร์เซนต์แล้ว
Cprogress Download	CProgressDownload	void	void	คอนตรักเตอร์ของคลาส CprogressDownload
	~CProgressDownload	virtual	void	เดสตรักเตอร์ของคลาส CprogressDownload
	DoDownload	void	void	ทำการดาวน์โหลดไฟล์จากเซิฟเวอร์
	Set	void	CSetdownload *thread	ตั้งค่าตัวแปรต่างๆที่ต้องใช้ในคลาส
	UpdateStatus	void	void	ส่งค่าให้คลาส CprogressDlg เพื่อแสดงรายละเอียดระหว่างดาวน์โหลดไฟล์

ตารางที่ ก (ต่อ) แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CProgressResume	CprogressResume	void	void	คอนตรักเตอร์ของคลาส CprogressResume
	~CProgressResume	virtual	void	เดสตรักเตอร์ของคลาส CprogressResume
	DoAutoResume	void	void	ทำการดาวน์โหลดต่อจากที่ทำค้างไว้
	Set	void	CSetAutoResume *thread1	ตั้งค่าตัวแปรต่างๆที่ต้องใช้ในคลาส
	UpdateStatus	void	void	ส่งค่าให้คลาส CprogressDlg เพื่อแสดงรายละเอียดระหว่างดาวน์โหลดไฟล์ที่ค้างไว้
CProgressResumeU	CprogressResumeU	void	void	คอนตรักเตอร์ของคลาส CprogressResumeU
	~CprogressResumeU	virtual	void	เดสตรักเตอร์ของคลาส CprogressResumeU
	DoAutoResumeU	void	void	ทำการอัพโหลดต่อจากที่ทำค้างไว้
	Set	void	CSetAutoResumeU *thread2	ตั้งค่าตัวแปรต่างๆที่ต้องใช้ในคลาส
	UpdateStatus	void	void	ส่งค่าให้คลาส CprogressDlg เพื่อแสดงรายละเอียดระหว่างอัพโหลดไฟล์ที่ค้างไว้
CProgressU	CprogressU	void	void	คอนตรักเตอร์ของคลาส CprogressU
	~CprogressU	virtual	void	เดสตรักเตอร์ของคลาส CprogressU
	DoUpload	void	void	ทำการอัพโหลดไฟล์จากเซิฟเวอร์
	Set	void	CSetUpload *thread	ตั้งค่าตัวแปรต่างๆที่ต้องใช้ในคลาส
	UpdateStatus	void	void	ส่งค่าให้คลาส CprogressDlg เพื่อแสดงรายละเอียดระหว่างอัพโหลดไฟล์
CRealApp	CRealApp	Void	void	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	InitInstance	virtual BOOL	void	กำหนดค่าของไดอะล็อก

ตารางที่ ก (ต่อ) แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CRealDlg	CRealDlg	Void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	Void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnAddGroup	afx_msg void	void	เพิ่มกรุ๊ปที่ต้องการ
	OnAddSite	afx_msg void	void	เพิ่มเซิฟเวอร์ที่ต้องการเชื่อมต่อในกรุ๊ป
	OnConnect	afx_msg void	void	เปลี่ยนหน้าไปยังไดอะล็อก FTP เพื่อให้เลือกชื่อเซิฟเวอร์ที่ต้องการเชื่อมต่อ
	OnDelGroup	afx_msg void	void	ลบกรุ๊ปที่ไม่ต้องการ
	OnDelSite	afx_msg void	void	ลบเซิฟเวอร์ที่ไม่ต้องการ
	OnEditSite	afx_msg void	void	แก้ไขข้อมูลของเซิฟเวอร์ที่เคยตั้งค่าไว้แล้ว
	OnExit	afx_msg void	void	ออกจากโปรแกรม
	OnInitDialog	virtual BOOL	void	กำหนดค่าต่างๆ เมื่อสร้างไดอะล็อก
	OnPaint	afx_msg void	void	การแสดงไดอะล็อก
	OnQueryDragIcon	afx_msg HCURSOR		คอนโทรลการคลิ้กลากไอคอน
	OnRenameGroup	afx_msg void	void	เปลี่ยนชื่อกรุ๊ป
	OnSelchange GroupList	afx_msg void	void	แสดงชื่อเซิฟเวอร์ต่างๆ ตามกรุ๊ปที่เลือก
	OnSysCommand	afx_msg void	UINT nID, LPARAM lParam	คอนโทรลปรับลดและขยายขนาดของไดอะล็อก
Crename	CRename	Void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	Void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnOK	virtual void	void	ยอมรับการเปลี่ยนชื่อ แล้วเปลี่ยนกลับไปไดอะล็อกหลัก

ตารางที่ ก (ต่อ) แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CsetAuto Resume	CsetAutoResume	Void	char newname[1000], CString oldname, char ClientPath1[2500], char ServerPath1[2500], char Server1[50], char Login1[50], char Password1[50]);	คอนตรักเตอร์ของคลาส ซึ่งจะรับค่าพารามิเตอร์ที่ส่งมาเพื่อนำมาใช้ในคลาส พร้อมประกาศเรียก เทรดฟังก์ชัน
	~CsetAutoResume	Virtual	void	เดสตรักเตอร์ของคลาส
	AutoResumeFunction	UINT	LPVOID lParam	เป็นเทรดฟังก์ชันเรียกทำคลาส และคอยดักจับความผิดพลาดที่เกิดขึ้นในการดาวน์โหลดในแต่ละเทรด
CsetAuto ResumeU	CsetAutoResumeU	void	char newnameU[1000], CString oldnameU, char ClientPath1U[2500], char ServerPath1U[2500], char Server1U[50], char Login1U[50], char Password1U[50]	คอนตรักเตอร์ของคลาส ซึ่งจะรับค่าพารามิเตอร์ที่ส่งมาเพื่อนำมาใช้ในคลาส พร้อมประกาศเรียก เทรดฟังก์ชัน
	~CsetAutoResumeU	virtual	void	เดสตรักเตอร์ของคลาส
Cset Download	CSetdownload	Void	CString Choose, char dir1[2500], char dir3[2500], char status[1000], char add[50], char log[50], char pas[50]	ตั้งค่าพารามิเตอร์ที่ส่งมาเพื่อนำมาใช้ในคลาส และ คอยดักจับความผิดพลาดที่เกิดขึ้นในการดาวน์โหลดในแต่ละเทรด
	~CSetdownload	Virtual	Void	เดสตรักเตอร์ของคลาส

ตารางที่ ก (ต่อ) แสดงคลาสและฟังก์ชันการทำงานในโปรแกรม

Class	Function	Return value	Parameter	หน้าที่การทำงาน
CSetTransfer	CSetTransfer	Void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	Virtual void	CDataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnAdd	Afx_msg void	Void	เพิ่มชนิดของไฟล์ที่จะรับ-ส่งแบบ ASCII
	OnChangename	Afx_msg void	Void	กำหนดสถานะให้ปุ่ม Add ว่าจะสามารถทำงานได้ก็ต่อเมื่อมีการพิมพ์ชนิดของไฟล์ลงไปในช่อง
	OnInitDialog	Virtual BOOL	Void	กำหนดค่าต่างๆ เมื่อเริ่มไดอะล็อก
	OnRemove	Afx_msg void	Void	ลบชนิดของไฟล์ที่จะรับ-ส่งแบบ ASCII
	OnSelchangeEx List	Afx_msg void	Void	กำหนดสถานะของปุ่ม remove ว่าสามารถทำงานได้ก็ต่อเมื่อมีการคลิ้กเลือกชนิดของไฟล์ไว้
CSetUpload	CSetUpload		char UpAfchoose[1000], CString Upchoose , char Updir1[2500] , char Updir3[2500] , char Upadd[50] , char Uplog[50] , char Uppas[50]	ตั้งค่าพารามิเตอร์ที่ส่งมาเพื่อนำมาใช้ในคลาส และ คอยดักจับความผิดพลาดที่เกิดขึ้นในการดาวน์โหลดในแต่ละเทรด
	~CSetUpload	Virtual	Void	เดสตรักเตอร์ของคลาส CsetDownload
QuickConnect	QuickConnect	void	CWnd* pParent	ตั้งค่าเริ่มต้นตัวแปรที่ใช้ในคลาส
	DoDataExchange	void	CdataExchange* pDX	จัดการการแลกเปลี่ยนข้อมูลในไดอะล็อก
	OnAnonymQ	afx_msg void	Void	เซตค่าล็อกอินเนมและพาสเวิร์ดสำหรับเชื่อมต่อเข้าเซิฟเวอร์แบบ Anonymous โดยเฉพาะ
	OnOK	virtual void	Void	รับค่าเซิฟเวอร์ที่ต้องการเชื่อมต่อ แล้วเปลี่ยนกลับไปไดอะล็อกหลัก

บรรณานุกรม

นิรุธ อำนวยศิลป์. 2542. **คู่มือการเขียนโปรแกรม** Microsoft Visual C++ version 6.0. กรุงเทพฯ : ซัคเซส มีเดีย.

Fiet, S. **TCP/IP Architecture, Protocols and Implementation.** 2nd ed. New York : Mc.Graw Hill International

Burk, R. et. al. **TCP/IP Blueprint.** New York : SAMS Publishing

Comer, D.E. **Internet Working with TCP/IP–Client/Server Programming and Application.** vol. 3. New Jersey : Prentice-Hall Inc.

Bonner, P. **Network Programming with Window Socket** . New Jersey : Prentic-Hall Inc.

Shepsherd, G. and Wingo, S. **MFC Internals Inside the Microsoft Foundation Class Architecture.** : Addison Wesley Developer press.

Richter, J. **Advanced Windows - The Developer Guide to the Win32 API for window NT 3.5 and Window 95.** Washington D.C. : Microsoft Press.

Young, M.J. **Mastering Visual C++ 6.0.** : BPB Publications.

Norton, S.J. and Pasquale, M.D. **The Multithread Programming Guide – Thread Time.**

Phan, T.Q. And Grag, Z.K. **Multithreaded Programming with Window NT.** New Jersey : Prentice-Hall.

Newmarch, J. 2000. **Client/Server Computing Semester1, 2000.** [online]. Available : http://pandonia.canberra.edu.au/ClientServer/index.html

Information Sciences Institute University of Southern California. September 1981. **Transmission Control Protocol Darpa Interner Program Protocol Specification (rfc793).** [online]. Available : ftp://ftp.isi.edu/in-notes/rfc793.txt

Postel J. Reynolds, J. october 1985. **File Transfer Protocol (rfc959).** [online] Available : http://www.w3.org/Protocols/rfc959/