ระบบบริหารเซิร์ฟเวอร์การพิมพ์

Print Server Management

นายธนาวุฒิ สมบูรณ์พงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ระบบบริหารเซิร์ฟเวอร์การพิมพ์

Print Server Management

โดย

นายธนาวุฒิ สมบูรณ์พงศ์

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบบริหารเซิร์ฟเวอร์การพิมพ์

PRINT SERVER MANAGEMENT

ผู้จัดทำ

1. นาย ธนาวุฒิ สมบูรณ์พงศ์ รหัสประจำตัว 42015307

_____ อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

ระบบบริหารเซิร์ฟเวอร์การพิมพ์

นายธนาวุฒิ สมบูรณ์พงศ์ 42015307
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

การจัดการบริหารการพิมพ์ของทางภาควิชาวิศวกรรมคอมพิวเตอร์นั้น ได้ใช้โปรแกรม PCOUNTER ในการจัดการบริหาร ซึ่งโปรแกรม PCOUNTER นี้ จำเป็นต้องทำงานร่วมกับโปรแกรมโนเวลเน็ตแวร์ ซึ่งทำให้การใช้งานขาดความยืดหยุ่น อันเนื่องมาจากจำเป็นต้องยึดติดกับโปรแกรมนี้

ทางภาควิชาจึงมีความประสงค์ที่จะพัฒนาโปรแกรมขึ้นมาเพื่อทดแทนโปรแกรม PCOUNTER เดิม โดยโปรแกรมที่พัฒนานี้ จะต้องใกล้เคียงกับโปรแกรมเดิม และสามารถทำงานได้บนระบบปฏิบัติการวินโดวส์

ปริญญานิพนธ์ฉบับนี้จึงได้เป็นการพัฒนาโปรแกรมขึ้นมาโดยใช้ Visual C++ 6.0 ในการพัฒนาโปรแกรม เพื่อสร้างโปรแกรมที่มีคุณสมบัติตรงกับความต้องการของผู้ใช้งานมากที่สุด โดยโปรแกรมที่พัฒนาขึ้นมานี้ เป็นโปรแกรมในแบบไดอะล็อก-เบส (Dialog-Based) คือโปรแกรมที่มีไดอะล็อกเป็นหน้าต่างหลักของโปรแกรม และได้ใช้เอ็มเอฟซี (MFC : Microsoft Foundation Class) มาช่วยในการพัฒนาโปรแกรม โดยโปรแกรมที่พัฒนาขึ้นนี้มีลักษณะคล้ายกับโปรแกรมเดิมที่ใช้ แต่สามารถทำงานได้บนระบบปฏิบัติการวินโดวส์

PRINT SERVER MANAGEMENT

Thanawoot Somboonpong 42015307
Thana Hongsuwan Advisor
Akkradach Watcharapupong Advisor

ABSTRACT

The printing management of computer engineering department used PCOUNTER program to manage all printing. PCOUNTER program has to work with NOVELL NETWRE program that unflexible.

Computer engineering department requires to develop the program to compensate the old program and compatible on Windows platform.

This thesis to develop the program by Visual C++ 6.0 for develop the program that meet most requirement. Type of program is Dialog-Based and used MFC (Microsoft Foundation Class). This program is similar to the old program but it can work with Microsoft Windows.

กิตติกรรมประกาศ

โครงงานและวิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี เนื่องจากทางผู้จัดทำได้รับความร่วมมือ และความอนุเคราะห์จากบุคคลต่างๆหลายฝ่าย ทั้งทางด้านการทำงาน กำลังใจ ทางผู้จัดทำขอขอบพระคุณเป็นอย่างยิ่ง

ขอบพระคุณ อาจารย์ที่ปรึกษา อาจารย์ธนา หงษ์สุวรรณ และ อาจารย์อัครเดช วัชระภูพงษ์ ที่ได้ให้ความช่วยเหลือเป็นอย่างมากและสละเวลาดูแลและเอาใจใส่การทำงานมาโดยตลอด

ขอบพระคุณ เหล่าคณาจารย์ ภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้ความรู้ทั้งทางด้านวิชาการและวิชาชีวิต

ขอบพระคุณ ภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้พึ่งพาอาศัยและความสะดวกสบายทุกๆด้านจนเปรียบเสมือนบ้านหลังที่สอง

ขอบคุณ เพื่อนๆ พี่ๆ น้องๆ ทุกคนสำหรับรอยยิ้ม และเสียงหัวเราะ กับสิ่งดีๆที่หยิบยื่นให้ตลอดระยะเวลาที่ทำการศึกษา

ขอบพระคุณบิดา มารดาอันเป็นที่รักยิ่ง ที่ได้ให้ทุกสิ่งทุกอย่างในชีวิตจนมีวันนี้

ขอบคุณเธอผู้เป็นกำลังใจ สำหรับทุกความอบอุ่น ทุกความรู้สึกและทุกสิ่งดีๆที่มอบให้ ขอบคุณ ขอบคุณ และขอบคุณ

จากหัวใจ

นายธนาวุฒิ สมบูรณ์พงศ์

สารบัญ

ชื่อเรื่อง	หน้า
บทที่ 1 บทนำ	**1**
1.1 วัตถุประสงค์	1
1.2 ขอบเขตของงาน	1
1.3 วิธีดำเนินงาน	1
บทที่ 2 win 32 programming	**2**
2.1 การเขียนโปรแกรมภาษา C บนวินโดวส์เบื้องต้น	2
2.1.1 WIN 32 API	2
2.1.2 Dynamic Link Library	3
2.1.3 Import Library	4
2.1.4 ชนิดของฟังก์ชั่นใน WIN 32 API	4
2.2 MFC(Microsoft Foundation Class)	4
2.2.1 การเขียนโปรแกรมบนวินโดวส์	5
2.2.2 การจัดการกับ message	8
2.2.3 การประกาศ message map	8
2.2.4 การเพิ่มมาโครที่สำคัญลงไปใน message map	9
2.2.5 การประกาศฟังก์ชั่นที่เป็นตัวจัดการกับ message map นั้น	9
2.2.6 เขียนรายละเอียดฟังก์ชั่นที่เป็นตัวจัดการกับ message อื่นๆ	10
2.3 Graphical Device Interface และ Device Context	12
2.3.1 Device Context	14
บทที่ 3 Printing	**14**
3.1 Introduction to Printing	14
3.2 Introduction to Print Provider	14
3.3 Print Provider	15
3.3.1 ความสามารถของ Print Provider	18
บทที่ 4 การจัดระบบการใช้เครื่องพิมพ์บนระบบเครือข่าย	**19**
4.1 Windows NT จัดการกับงานพิมพ์อย่างไร?	19
4.1.1 Windows NT server กับ Windows 95/98 Client	20
บทที่ 5 คุณสมบัติของโปรแกรม	**21**
5.1 คุณสมบัติหลักของโปรแกรม	21
5.1.1 การเพิ่มยอดเงินคงเหลือ	21
5.1.2 การลดยอดเงินคงเหลือ	22
5.1.3 การกำหนดอัตราค่าบริการงานพิมพ์	22
5.1.4 การกำหนดจำนวนเงินต่ำสุดที่จะอนุญาตให้ทำการพิมพ์ได้	23

สารบัญ

ชื่อเรื่อง	หน้า
5.1.5 การกำหนดจำนวนเงินสูงสุด	24
5.1.6 การแสดงรายละเอียดการพิมพ์ของผู้ใช้แต่ละราย	24
5.1.7 การแสดงยอดเงินคงเหลือ	24
5.2 คุณสมบัติอื่นๆของโปรแกรม	25
5.2.1 หน้าจอหลักของโปรแกรม	25
5.2.2 เมนูบาร์	26
5.2.3 ลิสต์บ็อกซ์	26
5.2.4 สแตติกเท็กซ์	27
บทที่ 6 รายละเอียดต่างๆของโปรแกรม	**28**
6.1 รูปแบบของโปรแกรม	28
6.2 รายละเอียดของโปรแกรม	28
6.2.1 ไดอะล็อกหลัก	28
6.2.2 ไดอะล็อก Set Cost	29
6.2.3 ไดอะล็อก Add Credit	30
6.2.4 ไดอะล็อก Set Credit Limit	30
6.2.5 ไดอะล็อก Set Balance	31
6.2.6 ไดอะล็อก Charge	32
6.2.7 ลิสต์บ็อกซ์ แสดงรายนามผู้ใช้ระบบ	32

คำนำ

ในการให้บริการเครื่องพิมพ์แก่นักศึกษาของทางภาควิชาวิศวกรรมคอมพิวเตอร์ เป็นที่นิยมอย่างยิ่งจากนักศึกษาภายในภาควิชา ทำให้มีผู้ใช้เป็นจำนวนมาก จึงทำให้ทางภาควิชาจำเป็นต้องมีการจัดการที่เหมาะสม และต้องง่ายต่อการจัดการ ซึ่งในปัจจุบัน ทางภาควิชานั้นได้ใช้โปรแกรม Pcounter ที่ทำงานบนโปรแกรมโนเวลเน็ตแวร์ ซึ่งได้ทำให้ขาดความยืดหยุ่นในการใช้งานโปรแกรมดังกล่าว เพราะจำเป็นต้องใช้โปรแกรมโนเวลเน็ตแวร์ด้วย ซึ่งในบางกรณีนั้นอาจไม่มีความสะดวกในการใช้งานและการจัดการที่ดี

โครงงานนี้จึงจัดทำขึ้นเพื่อขจัดปัญหาดังกล่าว โดยโปรแกรมที่จะพัฒนาขึ้นมานั้น จะสามารถทำงานได้บนโปรแกรมวินโดวส์ได้ ทำให้มีความยืดหยุ่นในการใช้งานมากยิ่งขึ้น อีกทั้งยังสมารถที่จะระบุคุณสมบัติของโปรแกรมให้มีความสามารถตรงกับความต้องการของทางภาควิชามากยิ่งขึ้น ซึ่งจะเพิ่มความสามารถให้กับการจัดการการให้บริการเครื่องพิมพ์ของทางภาควิชาได้ไม่มากก็น้อย

ทางผู้พัฒนาโปรแกรมในโครงงานนี้ ได้ทำการศึกษาถึงสิ่งต่างๆที่จำเป็นต่อการพัฒนาอย่างเต็มกำลังความสามารถ หากมีข้อบกพร่องประการใด ทางผู้จัดทำจักต้องขออภัยมา ณ โอกาสนี้ด้วย

ผู้จัดทำ

บทที่ 1

บทนำ

ในปัจจุบันทางภาควิชาวิศวกรรมคอมพิวเตอร์ ได้มีการให้บริการเครื่องพิมพ์แก่นักศึกษาในภาควิชา ซึ่งบริการนี้มีได้รับความนิยมและมีความจำเป็นต่อนักศึกษาภายในภาควิชา จึงทำให้มีผู้ใช้บริการเป็นจำนวนมาก อันจะมีผลทำให้มีความยุ่งยากซับซ้อนในการจัดการ จึงจำเป็นต้องมีการจัดการเรื่องบริหารระบบให้บริการเครื่องพิมพ์นั้นอย่างมีประสิทธิภาพ และต้องง่ายต่อการจัดการ ซึ่งจะสามรถลดปัญหาเกี่ยวกับการให้บริการในส่วนนี้ได้มาก

1.1 วัตถุประสงค์

โครงงานนี้จัดทำขึ้นมาเพื่อทดแทนโปรแกรมเดิมที่ทางภาควิชาใช้บริหารงานพิมพ์อยู่ในปัจจุบัน ซึ่งก็คือโปรแกรม Pcounter ทำงานบนโปรแกรมโนเวลเน็ตแวร์ อันเนื่องมาจากตัวโปรแกรมเดิมนั้นได้ยึดติดอยู่กับตัวโปรแกรมโนเวลเน็ตแวร์ซึ่งทำให้เราไม่สามารถไม่ใช้โปรแกรมเดิมนี้ได้ หากว่าเราไม่ได้ติดตั้งโปรแกรมโนเวลเน็ตแวร์ทำให้การใช้งานต้องขึ้นองค์ประกอบที่ไม่เหมาะสม

ดังนั้นจึงต้องทำการพัฒนาโปรแกรมขึ้นมาใหม่ โดยไม่ต้องยึดติดกับตัวโปรแกรมโนเวลเน็ตแวร์อีกต่อไป เพื่อทดแทนโปรแกรมเดิมที่มีข้อจำกัดในการทำงาน โดยโปรแกรมที่พัฒนาขึ้นมาใหม่นี้ จะทำงานบนแพลตฟอร์มวินโดวส์ได้เลย โดยไม่จำเป็นต้องใช้ร่วมกับโปรแกรมอื่นใดอีก อันจะมีผลทำให้มีความยืดหยุ่นในการใช้งานมากขึ้น และสามารถทำให้มีคุณสมบัติตรงตามความต้องการในการใช้งานจริงยิ่งขึ้น

1.2 ขอบเขตของงาน

โครงงานนี้เป็นการเขียนโปรแกรมบนวินโดวส์ หรือที่เรียกว่า ***Win32 programming*** ซึ่งได้ใช้ Microsoft Visual C++ ในการพัฒนาโปรแกรม ซึ่งโปรแกรมจะต้องมีความสามารถในการจัดการระบบเซิร์ฟเวอร์การพิมพ์ได้ เช่น เพิ่มเครดิต , ลดเครดิต , ดูรายละเอียดของงานพิมพ์ของแต่ละ user ได้ เป็นต้น

1.3 วิธีการดำเนิน

การดำเนินงานโครงงานนี้ เริ่มจากการศึกษาการเขียนโปรแกรมบนวินโดวส์ หรือที่เรียกว่า Win32 programming ว่าต้องมีวิธีการอย่างไรบ้าง ต่อจากนั้นก็เริ่มที่จะต้องศึกษาการเขียนโปรแกรม โดยใช้ Microsoft Visual C++ และศึกษาเครื่องมือต่างๆ ที่ใช้สำหรับช่วยในการพัฒนาโปรแกรม เช่น การใช้ MFC เป็นต้น

ศึกษาถึงกระบวนการการจัดการงานพิมพ์ของวินโดวส์ ว่ามีกระบวนการจัดการเช่นไร ประกอบไปด้วยการทำงานในส่วนใดบ้าง จากนั้นต้องมาทำการศึกษาส่วนที่เกี่ยวข้องกับการพิมพ์ของโปรแกรมวินโดวส์ อันได้แก่ Spooler ว่ามีโครงสร้างอย่างไร สามารถใช้งานแบบใดได้บ้าง และมีโครงสร้างข้อมูลใดบ้างที่จำเป็นต่อการพัฒนาโปรแกรม

บทที่ 2

Win 32 Programming

ในการเขียนโปรแกรมบนวินโดวส์นั้นสามารถที่จะเขียนได้โดยใช้ ***Win32 API*** **(Application Programming Interface)** ซึ่งเป็นกลุ่มของฟังก์ชันพื้นฐานสำหรับ Windows Programming ซึ่งให้ service ต่างๆของไมโครซอฟท์วินโดวส์โดยตรง หรืออาจใช้เครื่องมือที่เรียกว่า ***MFC*** **(Microsoft Foundation Class Library)** ซึ่งเป็นของทางบริษัทไมโครซอฟท์เองมาช่วยก็ได้เช่นกัน ซึ่งรายละเอียดต่างๆ จะได้กล่าวในหัวข้อต่อๆไป

ก่อนที่จะเข้าสู่ การเขียนโปรแกรม เราจะต้อง มาทำความเข้าใจ เกี่ยวกับส่วนประกอบพื้นฐาน บางส่วนที่สำคัญ ดังต่อไปนี้ (ในโปรเจ็คต์นี้ เราใช้ Microsoft Visual C++ ในการพัฒนาโปรแกรม)

- ***WinMain( )***

  โปรแกรมในวินโดวส์จะเริ่มทำงาน (execute) ได้โดยการเรียก ฟังก์ชั่น WinMain( ) ขึ้นมาทำงานก่อน ซึ่งก็คล้ายกับ การเรียกฟังก์ชั่น main( ) ในภาษา C ที่ใช้งานบน DOS นั่นเอง

- ***Window Function***

  โปรแกรมทุกโปรแกรม ในระบบวินโดวส์ จะมีฟังก์ชั่นพิเศษ อยู่ฟังก์ชั่นหนึ่ง ซึ่งไม่ถูกเรียกใช้ จากตัวโปรแกรม ที่เราเขียนขึ้น แต่จะถูกเรียกใช้ จากตัวระบบปฏิบัติการวินโดวส์เอง ฟังก์ชั่นนี้จะเรียกว่า window function หรือ window procedure ซึ่ง window function นี้จะถูกเรียกใช้จาก ตัวระบบปฏิบัติการวินโดวส์ เมื่อวินโดวส์ต้องการ ติดต่อกับ ตัวโปรแกรมที่เราเขียนขึ้น (กล่าวง่าย ๆ ว่าระบบปฏิบัติการวินโดวส์ จะติดต่อกับโปรแกรม ที่เราเขียนขึ้น ผ่านทาง window function นี่เอง) นอกจาก window function จะทำหน้าที่ในการรับข่าวสาร (message) ที่ส่งมาจากระบบปฏิบัติการ window แล้ว window function นี้ยังต้องมี ส่วนของชุดคำสั่ง ที่ใช้ตอบสนอง ต่อข่าวสาร (message) ที่ได้รับจากระบบปฏิบัติการ window ด้วย โดยปกติแล้ว ภายในฟังก์ชั่น window function นี้จะต้อง มีรหัสคำสั่ง switch( ) อยู่ภายในเสมอ ซึ่งใช้ในการเลือก ชุดคำสั่งที่จะตอบสนอง ต่อข่าวสาร (message) แต่ละข่าวสาร ที่ส่งมาจากระบบปฏิบัติการวินโดวส์นั่นเอง

- ***Window Classes***

  คำว่าคลาสในที่นี้ไม่ได้หมายถึง class ในการเขียนโปรแกรมแบบ C++ แต่หมายถึง รูปแบบ (style) ของหน้าต่าง ของโปรแกรมที่เราจะสร้างขึ้น โดยก่อนที่โปรแกรม ที่เราเขียนขึ้น จะสามารถทำงานได้ จะต้องมีการกำหนด (define) และลงทะเบียน (register) window class นี้เสียก่อน ซึ่งเป็นการกำหนดว่า หน้าต่างโปรแกรมที่เราเขียนขึ้นนี้ มีรูปแบบเป็นอย่างไร มีการทำงานแบบไหน แต่การลงทะเบียน window class นี้ยังไม่ทำให้เกิด หน้าต่างโปรแกรมของเราออกมาได้ ยังต้องมีขั้นตอน การสร้างหน้าต่างโปรแกรม ของเรา เพิ่มเติมขึ้นมาอีก

- ***Message Loop***

  โปรแกรมที่เขียนขึ้น ภายใต้ระบบปฏิบัติการวินโดวส์ ทุกโปรแกรม จะต้องมี message loop อยู่ภายใน ฟังก์ชั่น WinMain() เสมอ เพื่อ ทำการอ่านข่าวสาร (message) ใด ๆ ที่ค้างอยู่ในคิวข่าวสารของโปรแกรม (application's message queue) แล้วจึงส่ง message นั้น ๆ กลับไปให้ ระบบปฏิบัติการวินโดวส์ แล้วระบบปฏิบัติการนี้ ก็จะเรียกใช้ window function ของโปรแกรมของคุณ โดยส่ง message นี้เป็นตัวแปร (parameter) ไปให้ window function นั้นด้วย

- *Windows Data Types*

ชนิดของข้อมูล ในโปรแกรมที่เขียนบนวินโดวส์ นอกจากจะมีชนิดที่ใช้ ในโปรแกรมภาษา C ทั่วไป เช่น **int** หรือ **char** แล้ว ก็ยังมีข้อมูลชนิดอื่น ๆ เพิ่มเติมขึ้นมาอีก ก็คือ

- **HANDLE** เป็นข้อมูล int ขนาด 32 บิต ที่ใช้เป็นตัว บอกถึงทรัพยากร (resource) บางตัว ของวินโดวส์ ซึ่งข้อมูลชนิด handle นี้มีหลายแบบ แต่ข้อมูลชนิด handle ทุกแบบ จะต้องขึ้นต้นด้วยตัว H เสมอ เช่น HWND ก็หมายถึง handle ของหน้าต่าง (window) ของโปรแกรมนั่นเอง
- **BYTE** เป็นตัวแปรชนิด unsigned character ขนาด 8 บิต
- **WORD** เป็นตัวแปร unsigned short integer ขนาด 16 บิต
- **DWORD** เป็นตัวแปร unsigned long integer ขนาด 32 บิต
- **UINT** เป็นตัวแปร unsigned integer ขนาด 32 บิต
- **LONG** เป็นตัวแปรชนิด long
- **BOOL** เป็นตัวแปร integer ที่มีค่าได้ 2 ค่าคือ ถูก (1) หรือผิด (0)
- **LPSTR** เป็นตัวแปรที่ใช้ชี้ ไปยังข้อความ (pointer to string)
- **LPCSTR** เป็นตัวแปรแบบ const pointer to string

- *นอกจากข้อมูลที่ใช้ในวินโดวส์ทั่ว ๆ ไปแล้ว ยังมี ตัวแปรโครงสร้าง ที่สำคัญอีก 2-3 ตัว ก็คือ*
  - **MSG** เป็นตัวแปร โครงสร้าง ที่ใช้เก็บข่าวสาร (message) ของวิน โดวส์
  - **WNDCLASS** เป็นตัวแปร โครงสร้าง ที่ใช้กำหนดรูปแบบ หน้าต่าง (window class) ของโปรแกรมเรานั่นเอง

2.1 การเขียนโปรแกรมภาษา C บนวินโดวส์ เบื้องต้น

โปรแกรมบนวินโดวส์ ที่เขียนด้วยภาษา C จะมีฟังก์ชั่นที่สำคัญอยู่ 2 ฟังก์ชั่น คือ WinMain( ) และ WindowFunc( ) โดยในโปรแกรม WinMain( ) จะต้องมีการกระทำการ ขั้นตอนที่สำคัญดังต่อไปนี้

1. กำหนดรูปแบบของหน้าต่างโปรแกรมเรา (define a window class)
2. ลงทะเบียน window class นั้นกับระบบปฏิบัติการวินโดวส์
3. สร้างหน้าต่าง ของ window class ที่ลงทะเบียนไปแล้วนั้น
4. แสดงหน้าต่าง (window) ของโปรแกรมเรา ออกมาบนจอภาพ
5. เข้าสู่วงรอบการรับข่าวสาร จากระบบปฏิบัติการวินโดวส์ (running the message loop)

2.1.1 Win32 API

ในการโปรแกรมกับวินโดวส์นั้นในที่นี้จะใช้ system call ของวินโดวส์ซึ่งก็คือ Windows API (Application Programming Interface) ทั้งนี้ก็เพราะว่าตัววินโดวส์เองนั้นมีฟังก์ชั่นอยู่มากมายซึ่งพร้อมที่จะให้เราใช้ในการเขียนโปรแกรม ซึ่งเราไม่จำเป็นที่จะต้องเขียนมันขึ้นมาใหม่ ให้เกิดความซ้ำซ้อนกับวินโดวส์หรือถึงแม้ว่าเราจะสามารถเขียนฟังก์ชั่นเหล่านี้ขึ้นมาเองก็ใช่ว่าจะทำงานได้ดีกว่าฟังก์ชั่น ที่ได้จัดเตรียมไว้ให้แล้วภายในวินโดวส์และฟังก์ชั่นเหล่านี้ก็ได้รับการทดสอบแล้วว่าเป็นที่น่าเชื่อถือ ดังนั้นถ้าไม่มีเหตุผล ที่จะต้องสร้างฟังก์ชั่นที่ซ้ำซ้อนกับวินโดวส์ขึ้นมาเราก็ควรที่จะใช้งานฟังก์ชั่น
ของวินโดวส์จะเป็นการดีกว่า

สำหรับท่านที่เคยโปรแกรมกับดอสและเคยเรียกใช้ Interrupt แล้ว เจ้า Windows API ก็คล้ายคลึงกันนั่นแหละครับ แต่จะไม่มีเบอร์ Interrupt แต่จะเป็นชื่อเรียกฟังก์ชั่นสื่อความหมายได้มากกว่า

ดังที่เราทราบแล้วว่าวินโดวส์ได้พัฒนามาได้หลายรุ่นแล้วจนมาถึงรุ่น 32 bits ในปัจจุบันอันได้แก่ Windows 95, Windows 98, และ Windows 2000 ดังนั้น Windows API จึงได้รับการปรับปรุงและเปลี่ยนแปลง ด้วยเช่นกัน โดยครั้งที่ Windows 3.1 ได้รับความนิยมนั้นโปรแกรมที่พัฒนาขึ้นจะมีขนาด 16 bits และเมื่อ Windows 32 bits ออกมาก็มีความพยายามทำให้โปรแกรมที่ทำงานกับ Windows 3.1 ทำงานได้ด้วย ดังนั้น Windows API จึงยังคงรักษาคุณลักษณะเดิมที่มีใช้ใน Windows 16 bits เป็นส่วนใหญ่ และในทางกลับกันก็ได้มีความพยายามที่จะให้โปรแกรมที่เขียนแบบ 32 bits สามารถทำงานได้ บน Windows 16 bits จึงได้มี Win32s ขึ้นมาเพื่อช่วยให้ โปรแกรมแบบ 32 bits สามารถทำงานได้กับ Windows 16 bits และ Windows API นี้จะเรียกว่า Win32s API ส่วน Windows API แบบ 32 bits แท้นั้นจะเรียกว่า Win32 API (ไม่มี s) และในบทความนี้จะอ้างอิงตาม Win32 API

2.1.2 Dynamic Link Library

ฟังก์ชั่นเหล่านี้จะอยู่ในรูปของ DLL files (Dynamic Linked Libraries) ที่ใช้ภายในวินโดวส์เช่น user32.dll gdi32.dll เป็นต้น ดังนั้นเราก็พร้อม ใช้งานฟังก์ชั่นเหล่านี้ได้ทันทีภายในโปรแกรมของเรา ไฟล์ DLL แต่ละไฟล์ จะเก็บฟังก์ชั่นต่างๆของ Windows ที่อยู่ในหมวดหมู่เดียวกันไว้ด้วยกัน เช่น user32.dll จะ เก็บฟังก์ชั่นเกี่ยวกับ User Interface ไฟล์ gdi32.dll จะเก็บฟังก์ชั่นเกี่ยวกับ graphics ต่างๆเช่น การลงจุด, การวาดเส้น เป็นต้น

เมื่อเป็นเช่นนี้แล้ว ดังนั้นโปรแกรมที่เราเขียนขึ้นจึงไม่เก็บฟังก์ชั่นเหล่านี้ไว้ในโปรแกรมแต่จะทำการ เชื่อมโยง กับ ฟังก์ชั่นเหล่านี้เมื่อโปรแกรมของเราทำงานเท่านั้น ดังนั้นจึงทำให้โปรแกรมที่เราเขียนขึ้นมีขนาดเล็ก กระทัดรัดอีกด้วย

2.1.3 Import Library

แต่เนื่องจากว่าโปรแกรมเราเรียกใช้งานฟังก์ชั่นที่เก็บอยู่ใน DLL files ดังนั้น โปรแกรมจึงจำเป็นจะต้องทราบว่าจะเชื่อมโยงกับ DLL files เหล่านี้อย่างไรซึ่งข้อมูลเหล่านี้จะถูกเก็บอยู่ในไฟล์ที่เรียกว่า import library ดังนั้นในการเขียนโปรแกรม โปรแกรมของเราจึงจำเป็น จะต้องเชื่อมโยงเข้ากับ import library เหล่านี้ด้วย ซึ่ง import library files เหล่านี้ก็มักจะทำมาสำหรับ DLL files แต่ละ ไฟล์อยู่แล้ว เช่น ในกรณีของ GNU C for Win32 , import library ชื่อ libuser32.a จะเก็บข้อมูลที่จำเป็นสำหรับการเชื่อมโยงกับ user32.dll ดังนั้น ถ้าโปรแกรมของเราเรียกใช้ฟังก์ชั่นที่อยู่ใน user32.dll โปรแกรมของเราจึงจำเป็นที่จะต้องแปลร่วมกับ libuser32.a ด้วย

2.1.4 ชนิดของฟังก์ชัน ใน Win32 API

Win32 API มีฟังก์ชั่นอยู่ 2 ประเภทคือ แบบหนึ่งสำหรับใช้กับระหัส ANSI ซึ่งใช้สำหรับ Win9x ส่วนอีกแบบหนึ่งจะใช้สำหรับระหัส UNICODE ซึ่งมีใช้ใน Windows NT เช่น ฟังก์ชั่น MessageBox ถ้าเป็นแบบ ANSI จะมีชื่อเป็น MessageBoxA แต่ถ้าเป็นแบบ Unicode จะเป็น MessageBoxW แต่ในทางปฏิบัตินั้นเราจะใช้เพียงชื่อฟังก์ชั่นเท่านั้นในการโปรแกรมในที่นี้คือ MessageBox ไม่ว่าจะเป็นแบบ ANSI หรือแบบ UNICODE ทั้งนี้เพราะตัวแปลภาษาของเราจะเลือกให้เราเองว่าเราทำงานอยู่กับ Platform ใด และจะเลือกฟังก์ชั่นที่เหมาะสมให้เราเอง

2.2 MFC (Microsoft foundation class)

- *MFC คืออะไร?*

กล่าวอย่างง่าย ๆ MFC ก็คือกลุ่มของ คลาส (class) ในภาษา C++ ที่ถูกออกแบบมา เพื่อช่วย ให้การเขียน โปรแกรมในวินโดวส์ ง่ายและ รวดเร็วขึ้นนั่นเอง โดย MFC จะประกอบไปด้วย คลาสที่มี การสืบทอด (hierarchy) กันมา เป็นชั้น ๆ รวมแล้วประมาณ 200 คลาส ซึ่งครอบคลุม การทำงานพื้นฐาน ของการเขียนโปรแกรม ในวินโดวส์ ได้เกือบทั้งหมด และเนื่องจาก MFC เป็นกลุ่มของคลาส ที่เขียนขึ้น โดยใช้ภาษา C++ ดังนั้น เราจะต้องมี ความรู้ในภาษา C++ ดีพอสมควร เพื่อที่จะสามารถ ใช้งาน MFC ได้อย่างมีประสิทธิภาพ

- *โครงสร้างพื้นฐานของโปรแกรมที่เขียนด้วย MFC*

โปรแกรม ที่เขียนขึ้นโดยใช้ MFC จะประกอบไปด้วย คลาสอย่างน้อย 2 คลาส คือ

- คลาสของโปรแกรม (Application Class) เป็นคลาสที่ กำหนดลักษณะ ของโปรแกรม ที่คุณเขียนขึ้น
- คลาสของวินโดวส์ (Window Class) เป็นคลาสที่กำหนด หน้าต่างหลัก ของโปรแกรมของคุณ

ซึ่งในที่นี้ Window Class ของเรา จะสืบทอดมาจาก คลาสที่ชื่อ ***CFrameWnd*** ส่วน Application Class จะสืบทอดมาจาก คลาส *CWinApp*

- *ขั้นตอนของการเขียนโปรแกรมโดยใช้ MFC*

1. สืบทอด (derive) คลาส ของหน้าต่างโปรแกรม ของเรา มาจากคลาส CFrameWnd
2. สืบทอดคลาส ของโปรแกรม (application) ของเรา มาจาก คลาส CWinApp
3. กำหนด message map (จะได้กล่าวถึงต่อไปในตอนหลัง)
4. เขียน ชุดคำสั่ง ที่ใช้ในฟังก์ชั่น InitInstance ของคลาส CWinApp ขึ้นมา
5. สร้างวัตถุ (instance) ของคลาส application ของเราขึ้นมา

2.2.1 การเขียนโปรแกรมบนวินโดว์

รูปแบบการเขียนโปรแกรมบนวินโดว์ตั้งแต่อดีตจนถึงปัจจุบัน จาก Win16 จนถึง Win32 การเขียนโปรแกรมโดยใช้ APIs โดยตรงเหล่านี้ยังคงมีโครงสร้างพื้นฐานหลักที่มีรูปแบบการเขียนที่ไม่ต่างจากเดิมคือ

```
WinMain( )
{
// Create Main Window
RegisterClass( );
CreateWindowEx( );
// Message Loop
while (GetMessage( )) {
TranslateMessage( );
```

```
DispatchMessage( );
}
// Destroy Main Window
DestroyWindow( );
}
```

โดยทั่วไปโปรแกรมที่ทำงานบนไมโครซอฟวินโดวส์ท์จะมีฟังก์ชันเริ่มต้นทำงานคือฟังก์ชัน WinMain( ) ซึ่งเป็นจุดเริ่มต้นในการทำงานคล้ายกับฟังก์ชัน main( ) ในภาษา C โดยในฟังก์ชัน WinMain นี้โดยส่วนใหญ่จะมีการทำงาน 3 ขั้นตอนใหญ่ๆ คือ

1. สร้างวินโดวส์หลัก (Main Window) ให้แก่โปรแกรม
2. สร้าง Message Loop สำหรับโปรแกรมเพื่อรับอีเว้นต์ (วินโดวส์เมสเสจ) ต่างๆจากไมโครซอฟท์วินโดวส์ (OS)
3. ทำลายวินโดวส์หลักเมื่อจบโปรแกรม

จากที่กล่าวแล้วว่าทุกๆ Windows GUI Program จะต้องมีฟังก์ชัน WinMain( ) ปรากฏอยู่ โดยจะถูกประกาศในรูปแบบ

int CALLBACK WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd);

พารามิเตอร์ของฟังก์ชัน WinMain( ) มี 4 ตัว โดยที่ 2 ตัวแรกเป็น instance handle ของโปรแกรม โดยที่ hInstance จะเป็นค่า Unique Application Identifier กล่าวคือจะเป็นค่าที่ Unique ของโปรแกรมในระบบเลยทีเดียว ในระบบปฏิบัติวินโดวส์จะกำหนดค่านี้ให้แก่โปรแกรมที่รันขึ้นมาโดยที่ไม่ให้ซ้ำกัน พารามิเตอร์ถัดไปเป็น hPrevInstance ค่านี้แต่เดิมโปรแกรมที่เขียนด้วย Win16 API จะมีการใช้งานค่านี้เพื่อตรวจสอบว่าโปรแกรมนี้กำลังรันอยู่ในระบบหรือไม่ ซึ่งโดยปกติแล้วใน Windows 16 บิตจะไม่ยอมให้โปรแกรมหนึ่งๆสามารถรันได้มากกว่า 1 instance ในช่วงเวลาหนึ่ง (แต่ถ้าต้องการให้โปรแกรมที่เขียนด้วย Win16 สามารถรันได้หลายๆ instance ก็สามารถทำได้ โดยเพิ่มโค้ดในส่วนการจัดการหน่วยความจำเล็กน้อย เพื่อแก้ปัญหาเรื่องการจัดการหน่วยความจำของไมโครซอฟท์วินโดวส์) ต่อมาเมื่อก้าวเข้าสู่ Win32 อย่างเช่น Windows 95/98 หรือ NT มีแนวทางการจัดการหน่วยความจำที่ดียิ่งขึ้น ดังนั้นจะเห็นว่าหลายๆโปรแกรมที่ทำงานบนไมโครซอฟท์วินโดวส์สามารถรันได้หลายๆ instance ในเวลาเดียวกัน ดังนั้นตัวแปร hPrevInstance จึงไม่ได้ถูกใช้งานอีกต่อไป ค่านี้จะเป็น 0 เสมอตัวถัดไปเป็นพอยต์เตอร์ที่ชี้ไปยังสตริง ซึ่งเก็บ command-line ที่ส่งผ่านเข้าสู่โปรแกรมเมื่อเริ่มต้นรัน และสุดท้ายเป็นค่าเลขจำนวนเต็มแสดงรูปแบบการปรากฏของวินโดวส์หลัก โดยจะถูกใช้เป็นพารามิเตอร์ของฟังก์ชัน ShowWindow( ) ซึ่งลักษณะการปรากฏเริ่มแรกของวินโดวส์หลักอย่างเช่น minimize maximize หรือ normal

จากที่ทราบแล้วว่าเมื่อโปรแกรมเริ่มต้นทำงาน โปรแกรมที่เขียนขึ้นต้องมีการกำหนดค่าเริ่มต้นต่างๆมากมายเพื่อให้โปรแกรมทำงานได้อย่างถูกต้อง อีกทั้งพิจารณาแล้วว่าส่วนนี้ทุกๆโปรแกรมที่ทำงานบนไมโครซอฟท์วินโดวส์จะต้องทำ ดังนั้นเพื่อลดงานเหล่านี้ให้แก่โปรแกรมเมอร์ MFC จึงได้สร้างฟังก์ชัน WinMain( ) สำหรับทุกโปรแกรมที่เขียนด้วย MFC โดยจะใช้ชื่อเป็น AfxWinMain ซึ่งภายในฟังก์ชันนี้จะสร้างและจัดการกับ application object (เป็นออบเจ็คหนึ่งที่มีความสำคัญมากสำหรับการเขียนโปรแกรมด้วย MFC) ที่สืบทอดมาจากคลาส CwinApp และถ้าโปรแกรมเมอร์ต้องการกำหนดค่าเริ่มต้นใดๆเพิ่มเติมในโปรแกรมก็สามารถทำได้โดยการ override ฟังก์ชัน CWinApp::InitInstance( ) หรือถ้าต้องการเข้าไปแก้ไขหรือเพิ่มเติมขั้นตอนการรันก็สามารถทำได้โดยการ override ฟังก์ชัน CWinApp::Run( ) หรือต้องการเคลียร์สิ่งใดๆก่อนการจบโปรแกรมก็สามารถทำได้โดยการ override ฟังก์ชัน CWinApp::ExitInstance( )

ถ้าในโปรแกรมที่เขียนไม่มีวินโดวส์หลักใดๆอยู่เลยไม่ว่าจะมองเห็น (Visible) หรือไม่ก็ตาม นั่นคือโปรแกรมจะไม่มี Graphical User Interface และสิ่งที่ตามมาคือเป็นการยากที่จะโต้ตอบกับอีเว้นต์ (วินโดวส์เมสเสจ) ต่างๆ ดังนั้นโดยปกติโปรแกรมจะต้องมีวินโดวส์อย่างน้อย 1 วินโดวส์เพื่อเป็นวินโดวส์หลักเพื่อโต้ตอบกับอีเว้นต์ต่างๆที่เกิดขึ้น

สำหรับวิธีการเขียนโปรแกรมโดยใช้ Win32 API โดยตรง โปรแกรมต้องมีการเรียกใช้ฟังก์ชัน RegisterClass( ) เพื่อรีจีสเตอร์ window class (window class ในที่นี้ไม่ใช่อยู่ในลักษณะของออบเจ็คโอเรียลเต็ดแต่อย่างใด แต่เป็น data structure หนึ่งที่ใช้เก็บข้อมูลของวินโดวส์นั่นเอง) จากนั้นจะทำการเรียกใช้ฟังก์ชัน CreateWindowEx( ) เพื่อทำการสร้างวินโดวส์หลักของโปรแกรม โดยที่ใน MFC โค้ดที่จัดการกับสิ่งเหล่านี้จะถูกสร้างให้อัตโนมัติโดย Visual Studio อยู่แล้ว ซึ่งโค้ดที่ทำการสร้างวินโดวส์หลักจะอยู่ใน CWinApp::InitInstance( ) อยู่แล้ว สิ่งถัดไปที่ทุกๆโปรแกรมบนไมโครซอฟท์วินโดวส์ต้องมีคือ Message Loop โดยปกติโค้ดส่วนนี้จะมีลักษณะดังนี้

```
MSG msg;
  While (GetMessage(&msg, 0, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
```

loop นี้จะทำให้โปรแกรมนั้นสามารถรับอีเว้นต์ (วินโดวส์เมสเสจ) ได้ ซึ่งฟังก์ชัน GetMessage( ) จะส่งค่า Boolean ที่เป็น TRUE สำหรับทุกๆเมสเสจยกเว้น WM_QUIT ที่จะส่งค่า FALSE ออกมาซึ่งทำให้จบ while loop นี้ และทำให้โปรแกรมสิ้นสุดการทำงาน หลังจากการเรียกใช้ฟังก์ชัน GetMessage( ) จะได้รับข้อมูลของวินโดวส์เมสเสจ 1 เมสเสจกลับมา (อยู่ใน struct MSG นั่นเอง) จากนั้นจะมีการเรียกใช้ฟังก์ชัน TranslateMessage( ) เพื่อจัดการกับ accelerator key และ menu hot key ต่างๆและสุดท้ายจะเรียกใช้ฟังก์ชัน DispatchMessage( ) เพื่อส่งต่อวินโดวส์เมสเสจนี้ไปสู่วินโดวส์ที่เป็นเจ้าของเมสเสจ

สำหรับโปรแกรมที่เขียนด้วย MFC เช่นกันเหมือนกับโค้ดในส่วนสร้างวินโดวส์หลัก ส่วนโค้ดของ message loop นี้จะถูกสร้างให้โดยอัตโนมัติโดย Visual Studio ซึ่งจะอยู่เป็นส่วนหนึ่งในโค้ดของคุณ โดยที่ถ้าโปรแกรมของคุณเป็น Single Document Interface (SDI) หรือ Multiple Document Interface (MDI) ส่วนของ message loop นี้จะอยู่ภายในฟังก์ชัน CWinApp::Run( ) (แต่ลึกๆแล้วจะอยู่ใน CWinThread::Run( ) ซึ่งเป็นเบสคลาสของ CWinApp) และถ้าเป็น Dialog-based Application แล้ว message loop จะอยู่ในฟังก์ชัน CDialog::DoModal( ) ซึ่งจะถูกเรียกใช้ในระหว่างการทำงานของฟังก์ชัน CWinApp::InitInstance( ) เพื่อให้ปรากฏ Dialog หลักของโปรแกรมบนหน้าจอ และจากที่กล่าวว่าเมื่อ message loop มีการเรียกใช้ฟังก์ชัน DispatchMessage( ) เพื่อส่งต่อวินโดว์เมสเสจนี้ไปสู่วินโดวส์ที่เป็นเจ้าของเมสเสจ การทำงานเหล่านี้ผ่านทางฟังก์ชัน AfxWndProcBase และ AfxWndProc( ) ของ MFC และสุดท้ายจะถูกแมบ (mapped) ไปสู่ฟังก์ชันที่โต้ตอบกับวินโดวส์เมสเสจนี้ภายในคลาสของวินโดวส์หรือคลาสของ dialog ต่อไป

ส่วนสุดท้ายของงานในฟังก์ชัน WinMain( ) คือการทำลายวินโดวส์หลักเมื่อสิ้นสุดการทำงานของโปรแกรม โดยจะเรียกฟังก์ชัน DestroyWindow( ) ซึ่งเช่นกันโค้ดในส่วนนี้ก็ถูกรวมอยู่ใน MFC อยู่แล้วโปรแกรมเมอร์ไม่ต้องทำอะไรอีก แต่ถ้าต้องการเพิ่มเติมการทำงานเมื่อจะมีการจบการทำงานของโปรแกรมอย่างเช่นการปิดไฟล์หรือปิด socket ก็สามารถทำได้โดยการ override ฟังก์ชัน CWinApp::ExitInstance( ) ใน application class ของโปรแกรมของตน

2.2.3 การจัดการกับ message

- *Message คืออะไร?*

ดังที่ได้ กล่าวไปแล้ว ในบทก่อน ๆ ว่า วินโดวส์ จะติดต่อกับ โปรแกรม ที่เราเขียนขึ้น โดยการส่งข่าวสาร (message) ให้ ดังนั้น การจัดการกับ message นี้ จึงถือว่า เป็นหัวใจสำคัญ ของการเขียนโปรแกรม ในระบบวินโดวส์ เลยทีเดียว

message ในระบบวินโดวส์ มีจำนวนมากมาย ซึ่งจะใช้ค่าจำนวนเต็ม (interger value)
ค่า มาใช้แทน message หนึ่ง ๆ ตัวอย่างของ message ทั่ว ๆ ไป ที่พบได้บ่อย ก็เช่น

- WM_CHAR
- WM_PAINT
- WM_MOVE
- WM_CLOSE
- WM_LBUTTONUP
- WM_LBUTTONDOWN
- WM_COMMAND
- WM_SIZE

บางครั้ง message ที่วินโดวส์ ส่งมาให้กับโปรแกรมของเรา อาจจะมีค่า ข้อมูลอื่น ๆ เพิ่มเติมมาด้วย เช่น message WM_CHAR ที่จะถูกส่งมา เมื่อมีการกดแป้นพิมพ์ (keyboard) ก็จะมีข้อมูล เพิ่มเติม เป็น ค่าของคีย์ ที่ถูกกด ถูกส่งออกมาด้วย หรือ message WM_LBUTTONDOWN ที่จะถูกส่งมา เมื่อมีการ กดปุ่มซ้ายของเมาส์ลง ก็จะมีค่าตำแหน่ง โคออดิเนต ของเมาส์พอยน์เตอร์ ขณะนั้นถูกส่ง เพิ่มเติมกลับมาด้วย

แต่เนื่องจาก message ที่วินโดวส์ ส่งมาให้กับ โปรแกรมของเรานั้น มีจำนวนมากมายมหาศาล เรา ไม่จำเป็นต้อง ตอบสนองกับทุก message ที่ส่งมาก็ได้ เราจะเลือกตอบสนอง ต่อ message ที่เรา สนใจเท่านั้น ส่วน message ที่เราไม่ตอบสนองนั้น MFC จะเป็นตัวจัดการให้เอง

- *การจัดการกับ Message ในแบบของ MFC*

การจัดการกับ messageในโปรแกรมของคุณ จะต้องมีขั้นตอน ดังต่อไปนี้

1. ประกาศ message map ลงบนคลาสที่สืบทอดมาจากคลาสของหน้าต่างวินโดวส์ของเรา
2. มีการเพิ่มบรรทัด ของ มาโครที่สัมพันธ์กับ message ที่เราสนใจ ลงไปใน message map
3. มีการประกาศ ฟังก์ชั่นที่จะดำเนินการกับ message นั้น ๆ ไว้ใน class ของหน้าต่างโปรแกรม ของเรา
4. มีการ เขียนรายละเอียด ของฟังก์ชั่น ที่จะดำเนินการ กับ message นั้นลงไปใน โปรแกรมของเรา

2.2.4 การประกาศ message map

จะทำได้โดยการพิมพ์บรรทัด DECLARE_MESSAGE_MAP( ) ลงไปในการประกาศคลาสที่สืบทอดมาจากคลาสหน้าต่าง ของวินโดวส์ของเรา
เช่น

/ Class ของหน้าต่างโปรแกรมของเรา ที่สืบทอดมาจากคลาส CFrameWnd */*

```
class CMainWin : public CFrameWnd
{
public:
CMainWin( );
DECLARE_MESSAGE_MAP(   ) /* ประกาศ message map ตรงท้ายสุดของการประกาศคลาส */
};
```

2.2.5 การเพิ่มมาโครที่สัมพันธ์กับ message ลงไปใน message map

การที่โปรแกรมของเราจะตอบสนองกับ message ที่วินโดวส์ ส่งมาให้ได้นั้น จะต้องมีการเพิ่มมาโครที่สัมพันธ์กับ message ที่เราต้องการลงไปใน message map ของโปรแกรมที่เราเขียนขึ้นนั้นเสียก่อน โดยมาโครของ message ในแบบของ MFC นั้น จะมีชื่อเดียวกันกับ message มาตรฐานของวินโดวส์ (standard Windows message) แต่จะขึ้นต้นด้วยคำว่า ON_ และปิดท้ายด้วยเครื่องหมายวงเล็บ() เช่น มาโครของ message WM_LBUTTONUP ก็คือ ON_WM_LBUTTONUP( ) , มาโครของ WM_PAINT ก็คือ ON_WM_PAINT( ) แต่มี*ข้อยกเว้นอยู่อย่างหนึ่ง คือ มาโครของ WM_COMMAND จะเป็น ON_COMMAND( )*

การเพิ่มมาโครเข้าไปใน message map จะพิมพ์แทรกลงไประหว่างคำสั่ง BEGIN_MESSAGE_MAP และ END_MESSAGE_MAP ยกตัวอย่างเช่น ถ้าในโปรแกรม ของคุณ ต้องการจัดการกับ message WM_CHAR จะต้องพิมพ์บรรทัดต่อไปนี้ลงไปใน message map

```
BEGIN_MESSAGE_MAP(CMainWin,CFrameWnd)
 ON_WM_CHAR( )
END_MESSAGE_MAP( )
```

เนื่องจากในโปรแกรม ที่เราเขียนขึ้น สามารถมีหน้าต่างวินโดวส์ ได้หลายหน้าต่าง และแต่ละหน้าต่างก็ สามารถมี message map ของตนเอง ทำให้เราสามารถ กำหนดได้ว่า message map ใดเป็นของวินโดวส์ใดได้ โดยในคำสั่ง BEGIN_MESSAGE_MAP จะมีพารามิเตอร์ 2 ตัวคือ Owner กับ Base ดังแสดงตัวอย่าง

```
BEGIN_MESSAGE_MAP(Owner, Base)
/* เว้นไว้เติมมาโครของ message ที่ต้องการจัดการ */
END_MESSAGE_MAP( )
```

Owner ก็คือ ชื่อคลาสของ หน้าต่าง ที่จะจัดการกับ message นั้น

ส่วน ***Base*** ก็คือชื่อคลาสแม่ (Base Class) ของคลาสที่จะจัดการกับ message นั้น

2.2.6 การประกาศฟังก์ชั่นที่เป็นตัวจัดการกับ message นั้น ๆ ลงในการประกาศคลาส ของหน้าต่างวินโดวส์ของเรา

ฟังก์ชั่นที่เป็นตัวจัดการกับ message เราเรียกอีกอย่างว่า message handler จะมีชื่อเดียวกับชื่อของ message ของเรา แต่จะขึ้นต้นด้วย คำว่า On แทน เช่น ตัวจัดการกับ message WM_CHAR ก็คือฟังก์ชั่นที่มีชื่อว่า OnChar( ) นั่นเอง หรือตัวจัดการกับ message WM_LBUTTONDOWN ก็คือฟังก์ชั่น OnLButtonDown() นั่นเอง ดังนั้น จากการประกาศคลาสในข้อที่ 1 ก็จะต้องมีการประกาศ ฟังก์ชั่น ที่เป็นตัว จัดการกับ message นั้นลงไปด้วย ดังนี้

```
/* Class   ของหน้าต่างโปรแกรมของเรา ที่สืบทอดมาจากคลาส CFrameWnd */
class   CMainWin : public CFrameWnd
{
 public: CMainWin( );
afx_msg void OnChar(UINT ch, UINT count, UINT flags ); /* ประกาศฟังก์ชั่นที่เป็นตัวจัดการกับ message WM_CHAR */
DECLARE_MESSAGE_MAP(   ) /* ประกาศ message map ตรงท้ายสุดของการประกาศคลาส */
};
```

จะสังเกตว่าฟังก์ชั่น OnChar( ) จะมีคำว่า afx_msg ขึ้นต้นด้วย เพราะฟังก์ชั่นที่เป็นตัวจัดการ message ในแบบ MFC นี้จะต้อง ใช้คำว่า afx_msg เป็นตัวกำหนด ชนิดของฟังก์ชั่นเสมอ

2.2.7 เขียนรายละเอียดของฟังก์ชั่นที่เป็นตัวจัดการกับ message นั้น ๆ ลงในโปรแกรมของเรา

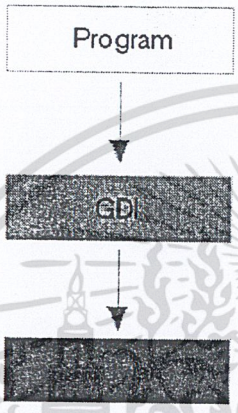
เช่นในการจัดการกับ message WM_CHAR โดยทุกครั้งเมื่อมีการกดปุ่มบนคีย์บอร์ด ในขณะที่หน้าต่างวินโดวส์ของโปรแกรมเรา active อยู่ จะมีตัวอักษรที่เรากด ไปปรากฏบน พื้นที่ทำงาน (client area) ของหน้าต่างวินโดวส์เรา ตรงตำแหน่งมุมบนซ้ายมือสุด เราจะเขียน รายละเอียด ของฟังก์ชั่นตัวจัดการ message WM_CHAR ดังต่อไปนี้

```
char str[80]; /* กำหนดตัวแปร เป็นตัวเก็บ string ที่จะนำไปแสดง */
/* รายละเอียดของฟังก์ชั่นที่ใช้จัดการกับ message WM_CHAR */
afx_msg void CMainWin::OnChar(UINT ch, UINT count, UINT flags)
{
 CClientDC dc(this);
 dc.TextOut(1, 1," ", 3); /* ลบตัวอักษรก่อนหน้านี้ */
wsprintf(str, "%c", ch); /* แปลงตัวอักษรที่ได้รับ ไปเป็น string */
dc.TextOut(1, 1, str, strlen(str)); /* ส่ง string ที่ได้ออกวินโดวส์ */ }
```

2.3 Graphical Device Interface และ Device Context

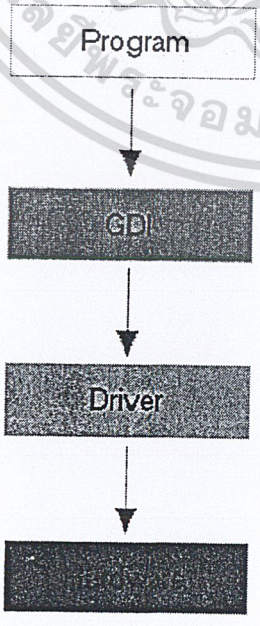
Windows เป็นระบบปฏิบัติการแบบ graphical user interface ดังนั้น Windows จึงจำเป็นที่จะมีมาตราการในการจัดการอุปกรณ์แสดงผล และในบทนี้เราจะมาเรียนรู้วิธีการที่ Windows ใช้กัน

ปกติในการจัดการจัดการอุปกรณ์ Input / Output ของ Windows นั้น Windows จะไม่เข้าจัดการอุปกรณ์เหล่านี้โดยตรงด้วยตนเอง แต่ Windows จะใช้วิธีการสร้าง function ทั่วไป สำหรับจัดการกับอุปกรณ์เหล่านี้ไว้เป็นส่วนหนึ่งของ Windows และเมื่อใดที่จะเข้าถึงอุปกรณ์ Input / Output เหล่านี้ (เช่น printer หรือ การ์ดแสดงผลเป็นต้น) Windows จะเรียกใช้ function ทั่วไปในกลุ่มนี้จัดการ ฟังก์ชั่นกลุ่มนี้จะถูกเก็บเป็นไฟล์ชนิด DLL ซึ่งเรียกว่า Graphical Device Interface (GDI) GDI มีหน้าที่เปลี่ยนคำสั่งของฟังก์ชั่นที่โปรแกรมเรียกใช้ ให้ไปอยู่ในรูปแบบคำสั่งที่ hardware เข้าใจได้ ดังนั้นถ้าโปรแกรมมีการเรียกใช้ฟังก์ชั่นของ GDI แล้ว ก็หมายความว่าโปรแกรมของเราจะสามารถที่จะทำงานได้บน hardware ที่แตกต่างกันได้นั่นเอง ซึ่งแสดงได้ดังรูปข้างล่าง



แต่สำหรับในกรณี hardware ใหม่ๆได้ออกมาภายหลัง Windows แล้วนั้น และไม่สนับสนุนโดย GDI แล้ว ผู้ผลิต hardware เหล่านี้จะต้องจัดหาสิ่งที่เรียกว่า *device driver* ซึ่งหมายถึงโปรแกรมที่ใช้ควบคุม hardware นั้นๆ

Device driver มีหน้าที่ๆ จะเปลี่ยนคำสั่งจากฟังก์ชั่น (คำสั่ง) ทั่วไปที่มีอยู่ใน GDI ให้เป็นคำสั่งที่ hardware ที่มันควบคุมอยู่เข้าใจได้ ดังนั้นถ้าเป็นในกรณี เช่นนี้แล้ว รูปภาพจะเปลี่ยนไปเป็นดังต่อไปนี้



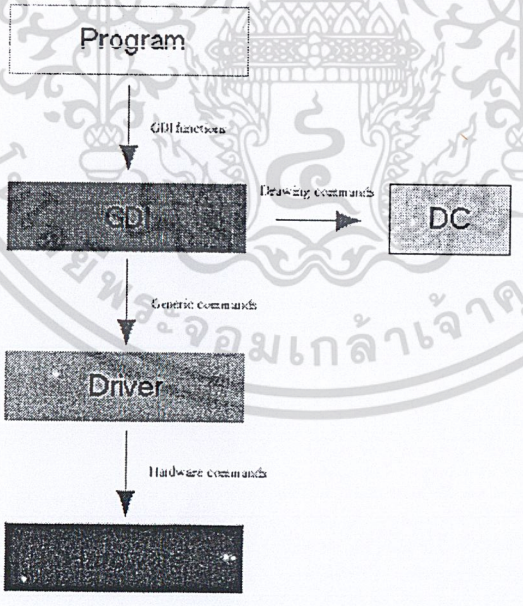
- *การแสดงผลของ Windows*

ภายใต้ dos นั้นตัวอักษรที่ใช้จะมีขนาดที่แน่นอน และโปรแกรมที่ทำงานแบบ graphics ภายใต้ dos จะมีวิธีแสดงผลอักษร หรือ รูปภาพตามแต่โปรแกรมนั้นๆ ที่ถูกโปรแกรมขึ้น และนอกจากนี้การเขียนโปรแกรมยังต้องขึ้นอยู่กับ graphics mode ที่แตกต่างกันด้วย ดังนั้นจึงมีโปรแกรมที่ทำหน้าที่ซ้ำๆ กันมาก แถมยัง ไม่สะดวกต่อการที่จะ port โปรแกรมที่เขียนอีกด้วย

จุดเด่นของ Windows อย่างหนึ่งที่ต่างจาก dos คือ การออกแบบให้ Windows เป็นแบบ Device Independence ซึ่งหมายความว่าโปรแกรมเดียวกัน สามารถที่จะทำงานได้โดยไม่ขึ้นกับอุปกรณ์ นั่นหมายความว่า เราไม่ต้องเปลี่ยนแปลงแก้ไขโปรแกรมของเราใหม่ทุกครั้งที่เราเปลี่ยนอุปกรณ์ใหม่ (เช่น Games บน Windows ของคุณยังคงทำงานได้อย่างถูกต้อง ถึงแม้ว่าคุณจะทำการเปลี่ยน การ์ดแสดงผลใหม่) และการที่ Windows กระทำเช่นนี้ได้ก็ดังได้กล่าวไปแล้วในหัวข้อข้างต้นว่า Windows จะไม่ส่งคำสั่งควบคุม hardware โดยตรงแต่มันจะเรียกใช้ function ทั่วไปที่เป็นส่วนของ GDI ในการจัดการแทน และการที่ function ใน GDI ทำงานแบบไม่ขึ้นกับอุปกรณ์ใดได้นั้นเพราะฟังก์ชั่นเหล่านี้จะทำงานกับ *Device Context (DC)*

- *Device Context (DC)*

Device Context อาจคิดได้ว่าเป็น logical hardware ซึ่งเป็นนามธรรมที่ได้ถูกสร้างขึ้นให้มีความสัมพันธ์กับ อุปกรณ์ที่มันเป็นตัวแทน เช่น Video card, Printer เป็นต้น โดยที่ function ใน GDI ทุกฟังก์ชั่นจะทำงานกับ DC (ไม่ทำงานโดยตรงกับ hardware) แล้ว GDI จะเปลี่ยนคำสั่งเหล่านี้ไปเป็น คำสั่งที่ hardware เข้าใจอีกที (อาจจะโดยการผ่าน device driver) และนี่ก็เป็นเหตุให้ Windows มีคุณสมบัติ Device Independence นั่นเองครับ ซึ่งสิ่งที่กล่าวมาสามารถแสดงได้ดังภาพต่อไปนี้



Device Context

เราจะพบว่า เมื่อต้องการ ส่งข้อความ หรือตัวอักษร หรือรูปกราฟใด ๆ ออกบนพื้นที่ทำงาน (client area) ของวินโดวส์หน้าต่างของโปรแกรมเรา จะต้องมีการ สร้างวัตถุเป้าหมาย (object) ของคลาส CClientDC ขึ้นมาเสียก่อน ซึ่งเมื่อมีการสร้าง วัตถุเป้าหมายขึ้นมา จะมีการรับค่าของ device context มาโดยอัตโนมัติ เมื่อได้ค่า device context มาเรียบร้อยแล้ว เราก็สามารถติดต่อ

กับพื้นที่ทำงาน ของหน้าต่างวินโดวส์ ของเราได้ทันที ทำให้สามารถใช้ ฟังก์ชั่น TextOut( ) ในการส่ง ข้อความ ออกไปยังตำแหน่ง ที่ต้องการได้ทันที

แล้ว device context คืออะไร? จริง ๆ แล้ว device context เป็นโครงสร้าง (structure) ที่เก็บค่า ตัวแปรต่าง ๆ ในการแสดง ผล ของวินโดวส์ไว้ เช่น ดีไวซ์ไดรเวอร์ (device driver), ชนิดของ ตัวอักษร ที่ใช้อยู่, และอื่น ๆ อีกมากมาย การที่จะแสดงผล ออก บนหน้าต่าง วินโดวส์ของโปรแกรมเรา จะต้องรู้จัก device context มาก่อนไม่ได้ก่อนอื่น

เราพบว่าโปรแกรม ที่เขียนขึ้นมีปัญหา อย่างหนึ่งคือ เมื่อหน้าต่างวินโดวส์ ของโปรแกรมเรา ถูกย่อลง หรือถูกบังโดย หน้าต่างวินโดวส์ ของโปรแกรมอื่น จะทำให้ข้อความ ที่แสดงอยู่ ในหน้าต่างวินโดวส์ ของ โปรแกรมเราหายไป เพราะวินโดวส์จะ ไม่ เก็บข้อมูลที่แสดง ในหน้าต่างไว้ เป็นหน้าที่ ของผู้เขียนโปรแกรม ที่จะต้อง จัดการในส่วนนี้เอง ซึ่งในตอนนี้ เราจะแก้ปัญหา โดยใช้วิธีการง่าย ๆ คือ มีเหตุการณ์อยู่ 3 อย่าง คือ

- เมื่อมีการแสดงหน้าต่างวินโดวส์ ของโปรแกรมขึ้นมาครั้งแรก
- เมื่อหน้าต่างวินโดวส์ ของโปรแกรมเรา ถูกทับโดยหน้าต่างวินโดวส์ ของโปรแกรมอื่น
- เมื่อหน้าต่างวินโดวส์ ของโปรแกรมเราถูก minimize

ซึ่งทั้ง 3 กรณีข้างต้น ระบบปฏิบัติการวินโดวส์ จะรู้ว่าจำเป็นต้อง มีการเขียนหน้าต่าง วินโดวส์ของ โปรแกรมเราขึ้นมาใหม่ เมื่อมีการคลิ๊กเรียก หน้าต่างของโปรแกรมเรา ออกมาอีกครั้ง วินโดวส์ก็จะส่ง message ที่ชื่อว่า WM_PAINT มายังโปรแกรมเรา เพื่อบอกให้โปรแกรม ของ เรารู้ว่า จะต้องมีการเขียน ข้อมูลในพื้นที่ทำงาน (client area) ของวินโดวส์ขึ้นใหม่

บทที่ 3

Printing

3.1 Introduction to Printing

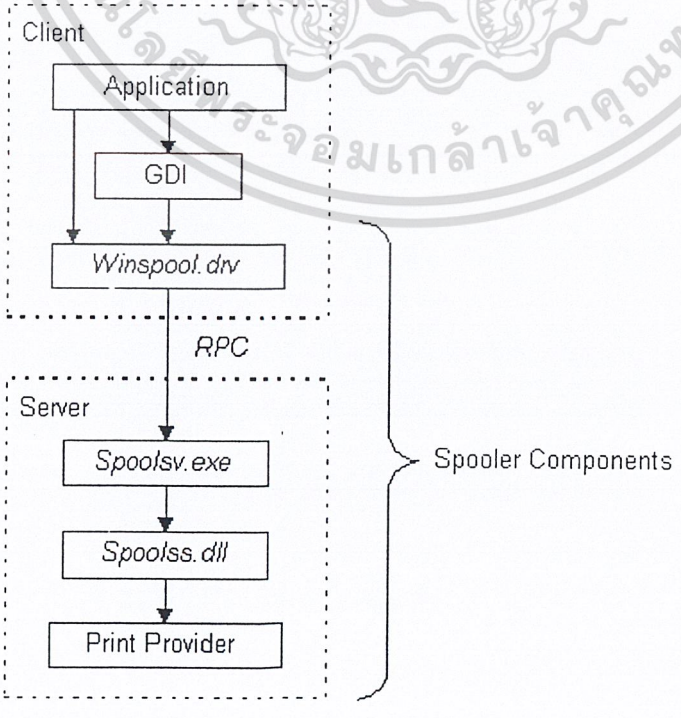
ไมโครซอฟท์วินโดวส์ 2000 และเวอร์ชั่นก่อนหน้านั้น จะมีสถาปัตยกรรมการพิมพ์ซึ่งประกอบด้วย print spooler และกลุ่มของ printer driver ซึ่งจะถูกเรียกใช้โดยอุปกรณ์การพิมพ์และฟังก์ชั่น GDI แอพพลิเคชั่นสามารถสร้างงานพิมพ์และส่งไปยังอุปกรณ์ต่างๆหลายรูปแบบได้ เช่น เลเซอร์พริ้นเตอร์ , เวคเตอร์พริ้นท์เตอร์ , ราสเตอร์พริ้นท์เตอร์ และเครื่องแฟกซ์ printer driver นั้นจะมี user interface ที่อนุญาตให้ผู้ใช้สามารถเลือก option ต่างๆ ที่มีอยู่ของเครื่องพิมพ์ได้

แอพพลิเคชั่นที่เรียก Win32 GDI ฟังก์ชั่น จะส่งค่าไปยัง GDI graphics engine ซึ่งค่าที่กล่าวถึงนี้คือ ค่าของ spooler ที่ใช้คำสั่งในการวาด เช่นเดียวกับ EMF ไฟล์ ในการเชื่อมต่อระหว่าง printer driver ภาพงานพิมพ์ที่แสดงนั้นจะถูกส่งไปยัง spooler , spooler component จะทำการแปล EMF ไฟล์ และยังสามารถใส่ข้อมูลเลย์เอาท์ของหน้ากระดาษได้อีกด้วย และคำสั่งควบคุมงานพิมพ์นี้จะถูกรวมเข้าไปในกระแสข้อมูล แล้วspooler จะส่งกระแสข้อมูลนี้ไปยังไดร์เวอร์พอร์ตอนุกรม , พอร์ตขนาน หรือ เนทเวิร์คพอร์ต ที่เชื่อมต่อกับ I/O พอร์ตของเครื่องพิมพ์

ส่วนประกอบของ spooler และ driver นั้น จะถูกออกแบบมาให้สามารถทดแทนกันได้ ดังนั้นผู้ผลิตฮาร์ดแวร์ก็จะง่ายต่อการออกผลิตภัณฑ์มาเพื่อสนับสนุน

3.2 Introduction to Spooler Components

ส่วนประกอบแรกสุดของไมโครซอฟท์วินโดวส์ 2000 และเวอร์ชั่นก่อนหน้านั้น ตัว print spooler จะถูกแสดงให้เห็นตามไดอะแกรมด้านล่าง



โดยรายละเอียดต่างๆ ของไดอะแกรม มีดังนี้

- ***Application***

  พริ้นท์แอพพลิเคชั่น จะสร้างงานพิมพ์โดยเรียกจาก GDI ฟังก์ชั่น

- ***GDI***

  Graphics device interface (รวมไปถึง user mode และ kernel mode ด้วย)

- ***Winspool.drv***

  ส่วนนี้เป็นส่วนติดต่อกับเครื่องไคล์เอนท์ภายในตัวspooler โดยจะส่งฟังก์ชั่นที่สำหรับสร้าง Win32 API ของตัว spooler และจัดสรร RPC เพื่อเข้าไปแอคเซสยังฝั่งเซิร์ฟเวอร์

- ***Spoolsv.dll***

  ส่วนนี้เป็นส่วนของ API เซิร์ฟเวอร์ของ spooler ซึ่งจะมีอยู่ในวินโดวส์ 2000 และเวอร์ชั่นก่อนหน้านั้น เซอร์วิสนี้จะเริ่มขึ้นเมื่อระบบปฏิบัติการเริ่มทำงาน โมดูลนี้จะส่ง RPC อินเทอร์เฟส ไปยังเซิร์ฟเวอร์ ในส่วนของ Win32 API ของ spooler spoolsv.exe ของไคล์เอนท์นั้น จะประกอบไปด้วย winspool.drv (แบบ Local) และ win32spl.dll(แบบ remote) โมดูลนี้จะมีอยู่ในหลาย API ฟังก์ชั่น แต่จริงแล้วฟังก์ชั่นโดยมากจะถูกเรียก และส่งผ่านไป print provider โดยจะหมายถึงเราท์เตอร์ (spoolss.dll)

- ***spoolss.dll***

  ส่วนนี้เป็นส่วนของเราท์เตอร์ ของ spooler ซึ่งถูกกำหนดให้ print provider เรียกใช้ โดยจะเป็นพื้นฐานของชื่อเครื่องพิมพ์หรือซัพพลายที่ทำงานอยู่กับแต่ละฟังก์ชั่นการทำงานที่เรียกใช้ และส่งผ่านฟังก์ชั่นเหล่านั้นไปยัง provider ที่ถูกต้อง

- ***Print provider***

  Print Provider นั้นจะสนับสนุนกับเครื่องพิมพ์ตามแต่ละชนิดต่างๆกัน

Spooler component ทั้งหมดจะถูกประมวลผลใน user mode

3.3 Print Providers

Introduction to Print Providers

Print Providers นั้นจะทำหน้าที่ส่งงานพิมพ์ตรงไปยังเครื่องพิมพ์ที่เป็น Local หรือ Remote รวมทั้งมีหน้าที่กระทำการจัดการลำดับของงานพิมพ์ด้วย เช่น เริ่ม , หยุด และระบุลำดับของงานพิมพ์ที่ตัว Server ด้วย Print Providers ได้ถูกกำหนดให้เป็น high – level , machine – independence , Operating system – independence โดย Print Server

โดยรวม Print Providers นั้นแสดงถึงเซตพื้นฐานของ Print Provider Capabilities โดยความสามารถนี้จะถูกกำหนดเซตของ API Function ซึ่งถูกเรียกโดยเราท์เตอร์ของตัว Spooler (spoolss.dll)

ฟังก์ชั่นการทำงานโดยมากจะถูกกำหนดโดย Print Providers ที่ต้องการเครื่องพิมพ์ที่มีอยู่เป็นเช่นอิน-พุต spooler ของเครื่อง client จะถูกเรียกโดยฟังก์ชั่น OpenPrinter ใน winspool.drv ซึ่งเรียกโดย API Server (spoolsv.exe) จากนั้นเราท์เตอร์ของ spooler (spoolss.dll) จะเรียกฟังก์ชั่น OpenPrinter ของแต่ละ Print Providers จนกระทั่งจะมีเครื่องพิมพ์เครื่องใดเครื่องหนึ่งที่มีอยู่ (ทำงานอยู่)จะส่งค่ากลับมา Print Provider ก็จะจดจำชื่อของเครื่องพิมพ์ที่ระบุนั้นไว้ ต่อจากนั้น ต่อจากนั้นเราท์เตอร์ก็จะส่งค่าต่างๆ ที่มีอยู่ไปยัง API Server ค่าที่เราท์เตอร์มีอยู่นั้นประกอบไปด้วย Printer Handle กับ Provider Handle ซึ่งค่าทั้งสองนี้จะถูกส่งกลับไปยังแอพพลิเคชั่น และภายหลังจากนั้นแอพพลิเคชั่นก็จะสามารถเชื่อมต่อโดยตรงกับ Provider และ Printer ที่ถูกต้องได้

ไมโครซอฟต์ ได้จัดแบ่งชนิดของ Print Providers กับ Windows 2000 และกับเวอร์ชั่นก่อนหน้านั้น ดังนี้

- *localspl.dll*

  เป็น Local Print Provider , งานพิมพ์ทั้งหมดจะถูกส่งโดยตรงไปยังเครื่องพิมพ์ ซึ่งจะถูกจัดการโดย Local Server

- *win32spl.dll*

  เป็น Network Print Provider , งานพิมพ์ทั้งหมดจะถูกส่งไปยัง Win 32 Srever แบบ Remote (ใน NT - base - operating - system หรือ Windows for Workgroups) และเมื่องานพิมพ์มาถึงยัง Remote Server มันจะถูกส่งต่อไปยัง Server ของ Print provider ในแบบ Local

- *nwprovau.dll*

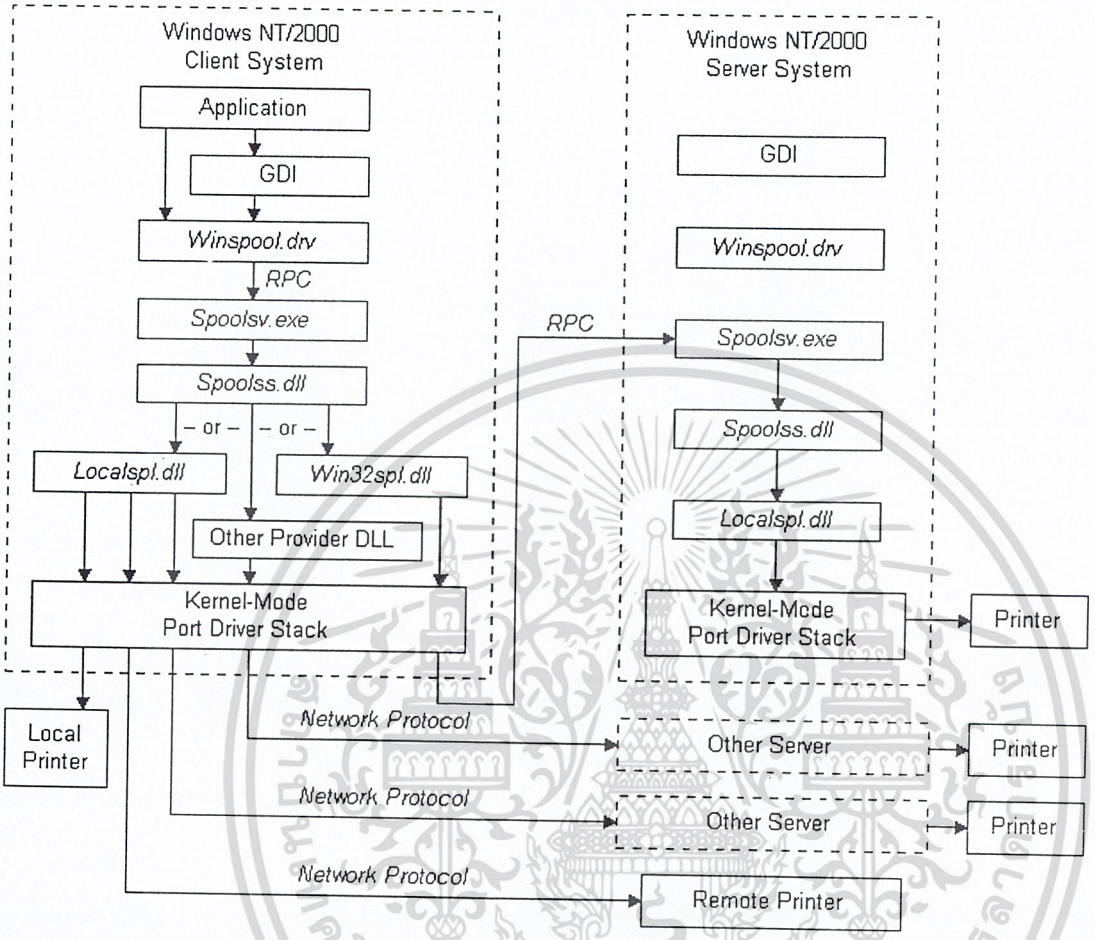
  เป็น Novell Netware Print Provider , งานพิมพ์ทั้งหมดจะถูกส่งไปยัง Novell Netware Print Server.

- *Inetpp.dll*

  เป็น HTTP Print Provider , งานพิมพ์ทั้งหมดจะถูกส่งไปยัง URL (ดู Internet Printing)

นอกจากนี้ผู้แทนจำหน่ายยังสามารถสร้าง Network Print Providers เพิ่มเติมเข้าไปได้อีก สามารถดูข้อมูลเพิ่มเติมได้ใน Writing a Network Print Provider

Diagram แสดง Flow Path ของ Print Provider



เมื่อเราดูจาก diagram แล้ว มีจุดที่ควรพิจารณา ดังนี้

- ถ้า Print provider ถูกจัดการโดยระบบ client , งานพิมพ์จะถูกควบคุมโดย Local Print provider (localspl.dll) , เครื่องพิมพ์ที่ถูกจัดการโดย localspl.dll นั้นไม่จำเป็น Local อย่าง physical ไปยังเครื่อง client พวกมันสามารถเชื่อมต่อโดยตรงกับ network การ์ดก็ได้
- ถ้าเครื่องพิมพ์ถูกติดตั้งอยู่บนเครื่อง Server ที่เป็นระบบปฏิบัติการ Window NT , Network Provider (win32spl.dll) จะใช้ RPC เรียก (แบบ redirect) จากเราท์เตอร์ของ client ไปยังกระบวนทำงาน spoolsv.exe ของเครื่อง Server เพราะว่าเครื่องพิมพ์นั้นจะถูกต่ออยู่แบบ Local กับ Server ซึ่ง Local Print Providers ของเครื่อง Server นั้นจะเป็นตัวที่จัดการงานพิมพ์เหล่านั้น
- ถ้าเครื่องพิมพ์ถูกติดตั้งบน Server ชนิดอื่นๆ , มันก็จะสามารถเข้าถึงได้โดย Local Print Provider หรือโดย Network Print Provider สนับสนุนกับ Server ชนิดนั้นๆ โดยใช้ในรูปแบบของข้อมูลต่างๆหรือ Network Protocol ที่สนับสนุนโดย server นั้น

- สำหรับ Local Print Provider จะเข้าถึงเครื่องพิมพ์ในแบบ Remote นั้นจำเป็นต้องรวม port monitor เอาไว้ด้วย ซึ่งสามารถใช้ Network Protocol ที่จดจำโดยเครื่องพิมพ์ในแบบ Remote หรือ โดย Server

3.3.1 ความสามารถของ Print Provider

การกำหนดในตอนต้นนั้นสนับสนุนในกลุ่มของ API ฟังก์ชั่น , Windows 2000 และเวอร์ชั่นก่อนหน้านั้น Print Provider สามารถรองรับความสามารถดังนี้

- ***Print Queue Management***

  Adding , deleting , opening , closing , enumerating , และกำหนดค่าพารามิเตอร์ต่างๆสำหรับการจัดลำดับการพิมพ์ รวมทั้งจัดการแจ้งสถานะการเปลี่ยนแปลงของลำดับงานพิมพ์ด้วย

- ***Printer Driver Management***

  Adding , deleting , enumerating , และระบุไดเร็คทอรี่สำหรับไดร์เวอร์ของเครื่องพิมพ์

- ***Print Job Scheduling***

  Scheduling , enumerating , และกำหนดค่าพารามิเตอร์สำหรับงานพิมพ์

- ***Forms Management***

  Adding , deleting , enumerating , และกำหนดค่าพารามิเตอร์สำหรับรูปแบบของงานพิมพ์

- ***Print Processor Management***

  Adding , deleting , enumerating , ระบุไดเร็คทอรี่สำหรับชนิดของข้อมูลที่สนับสนุนโดย Print Processors

- ***Print Monitor Management***

  Adding , deleting , และแสดงค่าต่างๆของเครื่องพิมพ์

- ***Port management***

  Adding , deleting , configuring , enumerating , และกำหนดค่าพารามิเตอร์สำหรับพอร์ตของเครื่องพิมพ์

- ***Registry Management***

  Creating , deleting , และแสดงค่าที่ระบุของ registry keys และค่าต่างๆที่รวมอยู่กับ Print Provider

- ***Other Capabilities***

  แสดง message box , ชัทดาวน์ Print Provider , อ่านหน่วยความจำเพื่อไป mapped กับ spool file , จัดสรรการติดต่อสื่อสารระหว่าง port monitor UI DLL กับ port monitor server DLL

ความสามารถต่างๆเหล่านี้จะทำในเซตของฟังก์ชั่นที่ถูกระบุโดย Print Provider

บทที่ 4

การจัดระบบการใช้เครื่องพิมพ์บนระบบเครือข่าย

การจัดระบบการใช้เครื่องพิมพ์ร่วมกันบนระบบเครือข่าย Windows NT สามารถทำได้อย่างง่ายๆ โดยติดตั้งเครื่องพิมพ์ให้กับ Windows NT และกำหนดสิทธิ์ในการเข้าใช้งานเครื่องพิมพ์ให้แก่ User ซึ่งจะมีส่วนประกอบทั้งหมด 4 ส่วนด้วยกันคือ

Client computer คอมพิวเตอร์ที่สั่งพิมพ์งาน ซึ่งอาจติดตั้งระบบปฏิบัติการต่างๆกัน เช่น Windows NT , Windows 95/98 , Windows for Workgroup หรือ DOS รวมไปถึง Unix , NetWare หรือ System 7 , System 8 ของเครื่องแมคอินทอชก็ได้

Printer ชื่อของเครื่องพิมพ์ที่ติดตั้งอยู่กับ print server ซึ่งมองเห็นใน Windows NT (เป็น logical printer หรือจะมองเป็นคิวที่เอกสารจะต้องไปรอการส่งพิมพ์ต่อไปก้อได้ แต่ยังไม่ใช่ตัวเครื่องพิมพ์จริงที่เรียกว่า Print Device)

Print Server เครื่องเซิร์ฟเวอร์ซึ่งให้บริการการพิมพ์กับผู้ใช้ในเครือข่าย โดยผู้ใช้ส่งข้อมูลที่จะพิมพ์เข้ามาให้เซิร์ฟเวอร์ประมวลผลก่อน แล้วส่งเข้าไปพิมพ์ยังเครื่องพิมพ์

Print Device คือตัวอุปกรณ์ฮาร์ดแวร์ของเครื่องพิมพ์จริงๆ ซึ่งทำหน้าที่พิมพ์งานต่างๆที่ส่งมาจาก (logical) Printer (ซึ่งอาจมีหลายตัวแต่ผูกกับ logical Printer หรือ "คิวในการพิมพ์" เดียวกันได้) โดยแบ่งเป็น 2 ประเภทคือ

- **Local Print Device** คือเครื่องพิมพ์ที่ต่อตรงกับพอร์ตของ Print Server
- **Network Interface Print Device** คือเครื่องพิมพ์แบบที่มี network card เสียบอยู่ในตัวเอง ซึ่งอยู่ในตัวเอง ซึ่งสามารถเชื่อมต่อกับเครือข่ายได้โดยตรง

User สามารถสั่งพิมพ์งานจากคอมพิวเตอร์เครื่องใดก็ได้ภายในเครือข่ายของ Windows NT และอาจกำหนดสิทธิ์ในการเข้าใช้งาน Printer ที่กำหนดให้กับ User นี้ อาจแบ่งได้เป็น 4 ระดับ คือ

No Access ไม่สามารถเข้าใช้งานเครื่องพิมพ์ได้เลย

Print สั่งพิมพ์ได้

Manage documents จัดการเอกสารที่ส่งไปพิมพ์ได้

Full Control ควบคุมได้ทั้งหมด

โดยทั่วไป User ทุกคนจะถูกกำหนดสิทธิ์อยู่ที่ระดับ Print อัตโนมัติ ถ้าหากเราเป็นผู้บริหารระบบ (Administrator) ก็สามารถจำกัดสิทธิ์ของ User คนอื่นๆในการเข้าใช้เครื่องพิมพ์บางตัวได้ ในองค์กรใหญ่ๆมักจะมอบงานให้กับคนหรือสองคนทำหน้าที่รับผิดชอบดูแลเกี่ยวกับเรื่องการพิมพ์หรือการกำหนดสิทธิ์ให้กับ User รายอื่นๆได้ นอกจากนี้เราสามารถกำหนดสิทธิ์มากกว่า 1 อย่างให้กับ User คนเดียวได้ด้วย เช่น User รายหนึ่งอาจมีสิทธิ์ทั้ง Print , Manage documents และ Full Control ก็ได้

สำหรับการควบคุม User ที่จะพิมพ์งานก็จำเป็นต้องมีการกำหนดกลุ่มและสิทธิ์ให้เหมาะสม ซึ่งข้อแนะนำในการติดตั้งและกำหนดสิทธิ์การใช้งานร่วมกันของเครื่องพิมพ์ในระบบเครือข่ายมีดังนี้

1. ให้พิจารณาว่า User หรือ Group เป็น Local (อยู่ใน Domain เดียวกัน) หรือ Global
2. ตั้งชื่อ Printer ที่ใช้ร่วมกัน (Share Name) ให้ง่ายต่อการจดจำ
3. พิจารณาระดับชั้นของความปลอดภัยในการใช้ Printer ให้กับ Group หรือ User แต่ละคน

4.1 Windows NT จัดการกับงานพิมพ์อย่างไร?

ความเข้าใจในเรื่องขบวนการพิมพ์จะช่วยให้เข้าใจการจัดการเครื่องพิมพ์ของ windows NT มากขึ้น ภายใน windows NT จะมี*ตัวจัดคิวการพิมพ์ หรือ spooler* ทำหน้าที่ประมวลผลงานพิมพ์และจัดช่วงเวลา (schedule) การพิมพ์หรือที่เรียกว่า spooling ซึ่งเป็นกระบวนการจัดเก็บข้อมูลหรือเอกสารที่จะส่งพิมพ์ไว้ชั่วคราวบนฮาร์ดดิสก์เป็น spool file ก่อนที่จะทยอยส่งไปพิมพ์จริงบนเครื่องพิมพ์ต่อไป

ถ้าเอกสารติดขัดอยู่ใน spooler เราสามารถที่จะหยุดและทำการ Restart Spooler ได้โดยผ่าน Control Panel

4.1.1 Windows NT Server กับ Windows 95/98 Client

การสั่งพิมพ์เอกสารจากเครื่อง client ซึ่งใช้ในระบบปฏิบัติการ Windows 95/98 มีลำดับขั้นตอนการพิมพ์เกิดขึ้นดังต่อไปนี้

1. จะมีการจัดรูปแบบและแบ่งเอกสารออกเป็นส่วนๆ เพื่อให้เข้ากับรูปแบบของเครื่องพิมพ์รุ่นนั้นๆ เช่น HP LaserJet 6
2. เอกสารจะถูกส่งไปเข้า Spooler ของเครื่อง client นั้นๆก่อน จากนั้นก็จะมีการส่งต่อไปยัง Windows NT โดยเข้าไปเก็บใน Spooler ของ Print Server อีกทอดหนึ่ง
3. Spooler ของ Print Server จะเก็บเอกสารที่ถูกส่งมาจาก Spooler ของ client โดยเอกสารจะรออยู่จนกระทั่งเครื่องพิมพ์พร้อมที่จะทำงานและรับงานพิมพ์ จากนั้นก็จะทำการพิมพ์

ดังนั้นการจัดการกับเอกสารที่ค้างอยู่ใน Spooler ก็จะต้องทำให้ถูกที่และจังหวะ ทั้งนี้ใน Window95/98 ผู้ใช้จะเห็นรายชื่อเอกสารในวินโดว์ Printers (ที่ต่ออยู่กับ Windows NT Server) ก็ต่อเมื่อไปถึงขั้นที่ 3 แล้วเท่านั้น

ในกรณีที่เครื่อง client ใช้ระบบปฏิบัติการอื่นเช่น UNIX ก็จะมีเพียง Spooler ของ Print Server เท่านั้น นั่นคือเมื่อ User สั่งพิมพ์ ก็จะมีการแบ่งเอกสารและส่งไปยัง Spooler ของ Print Server ทันที เพื่อส่งไปยังเครื่องพิมพ์ต่อไป

บทที่ 5

คุณสมบัติของโปรแกรม

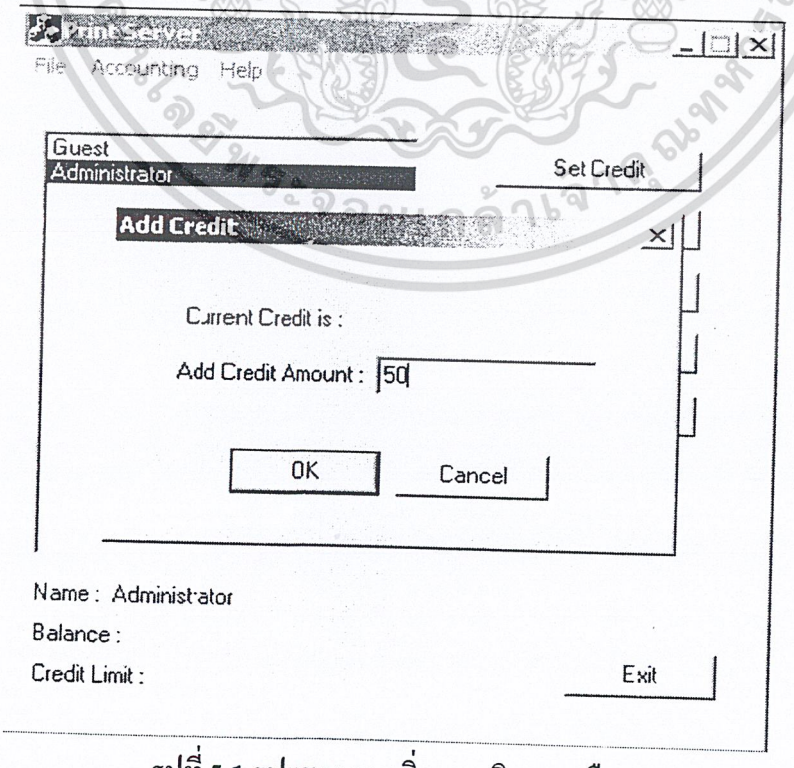
ในบทนี้ขอกล่าวถึงคุณสมบัติต่างๆของโปรแกรม ซึ่งจะอธิบายถึงคุณสมบัติหลักๆที่จะใช้ในการจัดการระบบบัญชีการให้บริการการพิมพ์ ดังนี้

5.1 คุณสมบัติหลักของโปรแกรม

- การเพิ่มยอดเงินคงเหลือ
- การลดยอดเงินคงเหลือ
- การกำหนดอัตราค่าบริการงานพิมพ์
- การกำหนดจำนวนเงินต่ำสุดที่จะอนุญาตให้ทำการพิมพ์ได้
- การกำหนดจำนวนเงินสูงสุด
- การแสดงรายละเอียดการพิมพ์ของผู้ใช้แต่ละราย
- การแสดงยอดเงินคงเหลือ

5.1.1 การเพิ่มยอดเงินคงเหลือ

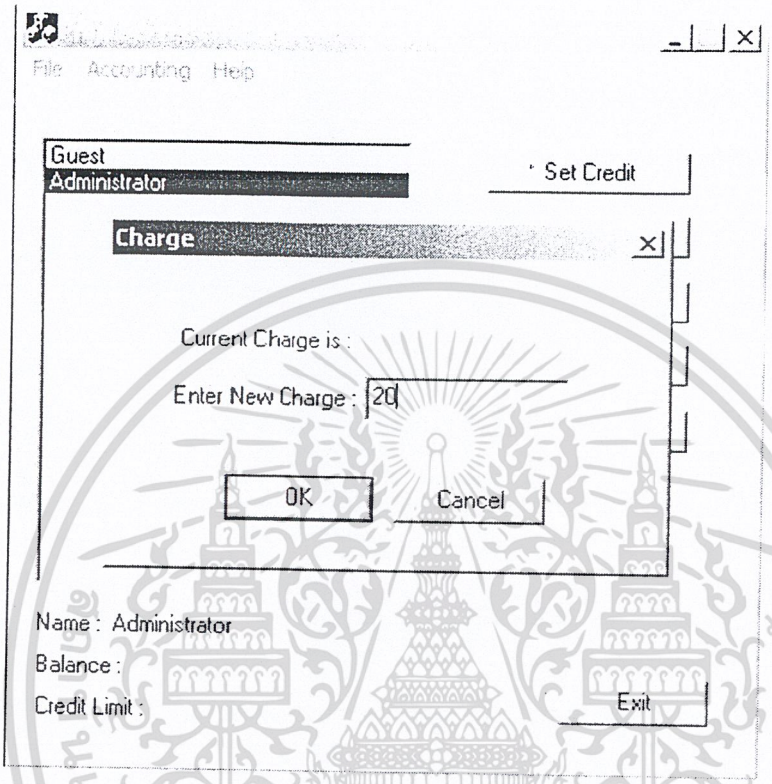
เป็นการเพิ่มจำนวนเงินจากยอดเงินคงเหลือเดิมที่เหลืออยู่ของผู้ใช้นั้นๆ ซึ่งสามารถเพิ่มยอดเงินคงเหลือของแต่ละบุคคล หรือเพิ่มทีละกลุ่มของผู้ใช้ก็ได้



รูปที่ 5.1 รูปแสดงการเพิ่มยอดเงินคงเหลือ

5.1.2 การลดยอดเงินคงเหลือ

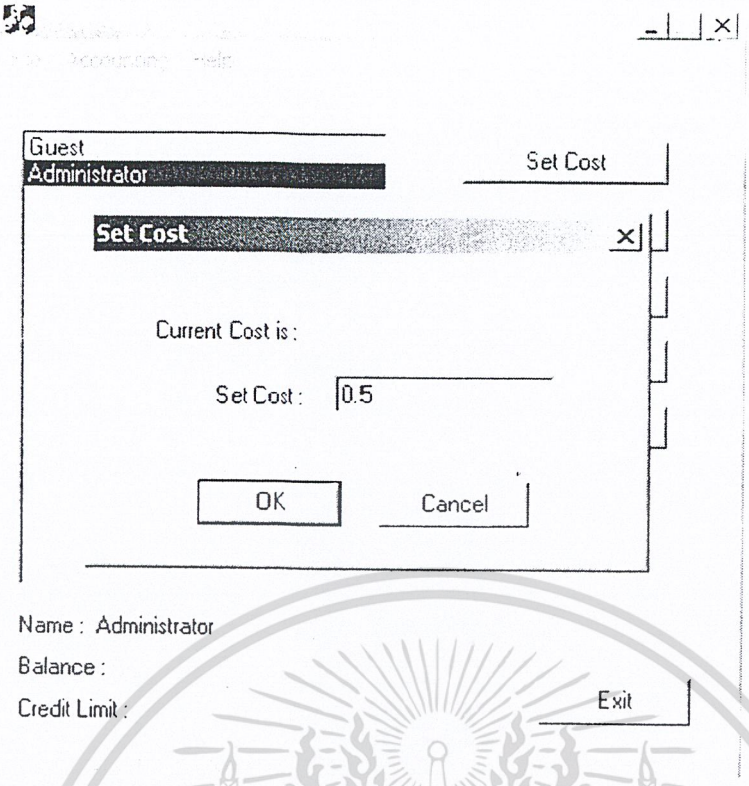
เป็นการลดจำนวนเงินตามที่ระบุ จากยอดเงินคงเหลือ เช่น ค่าธรรมเนียมพิเศษ หรือกรณีอื่นๆ ตามแต่วัตถุประสงค์ของผู้จัดการบริหารการพิมพ์ โดยทำการเลือกที่เมนู **charge** แล้วทำการใส่จำนวนที่ต้องการลงไป



รูปที่ 5.2 รูปแสดงการลดยอดเงินคงเหลือ

5.1.3 การกำหนดอัตราค่าบริการงานพิมพ์

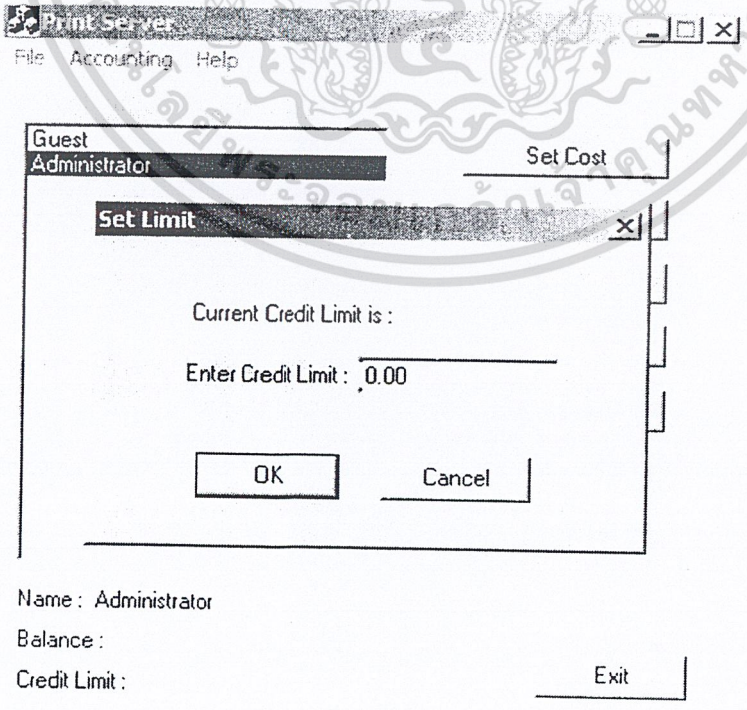
เป็นการกำหนดอัตราค่าบริการของการพิมพ์ โดยเป็นการให้ระบุราคาค่าบริการต่อการพิมพ์หนึ่งหน้าว่ามีราคาค่าบริการเท่าไร โดยทำการเลือกที่รายการ **set cost** แล้วทำการใส่ค่าที่ต้องการลงไป



รูปที่ 5.3 รูปแสดงการกำหนดอัตราค่าบริการงานพิมพ์

5.1.4 การกำหนดจำนวนเงินต่ำสุดที่จะอนุญาตให้ทำการพิมพ์ได้

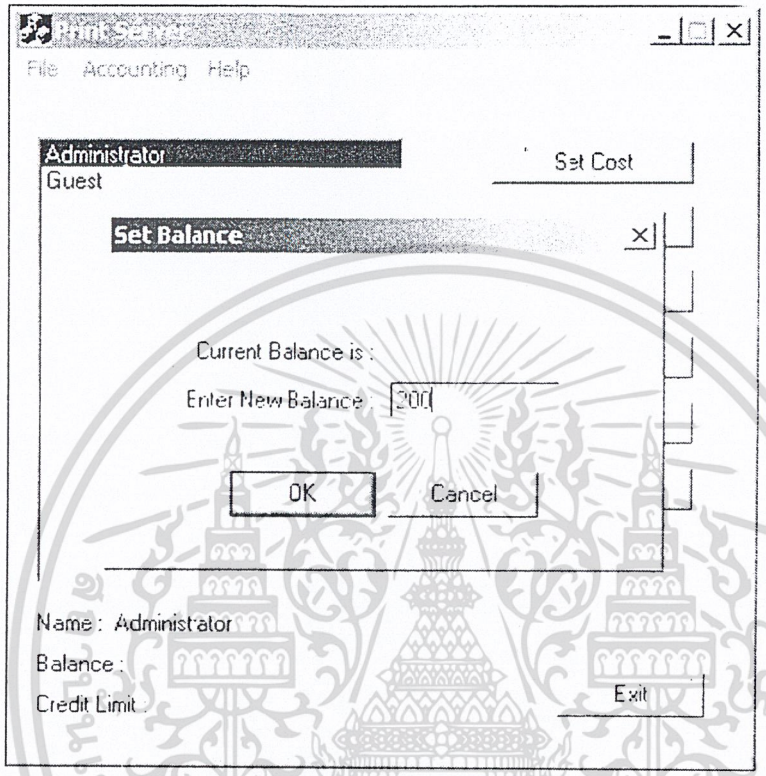
เป็นการกำหนดว่านักศึกษาจะสามารถทำการพิมพ์ได้จนกว่าจำนวนเงินจะเหลือเท่าไร โดยปกติแล้วจะทำการกำหนดไว้ที่ศูนย์ หลังจากยอดคงเหลือเป็นศูนย์ก็จะไม่สามารถทำการพิมพ์ต่อได้ หรือหากว่าไม่ได้ทำการกำหนดไว้ ก็จะสามารถทำการพิมพ์ได้ไม่จำกัด โดยเลือกที่รายการ **set credit limit**



รูปที่ 5.4 แสดงการกำหนดจำนวนเงินต่ำสุดที่จะอนุญาตให้ทำการพิมพ์ได้

5.1.5 การกำหนดจำนวนเงินสูงสุด

เป็นการกำหนดจำนวนยอดเงินคงเหลือสูงสุดที่ผู้ใช้จะสามารถเพิ่มจำนวนเงินเข้าไปได้ โดยจะกำหนดค่ามาตรฐานไว้ที่ 200 บาท การทำงานนี้ทำได้โดยเลือกที่รายการ **set balance**



รูปที่ 5.5 แสดงการกำหนดจำนวนเงินสูงสุด

5.1.6 การแสดงรายละเอียดการพิมพ์ของผู้ใช้แต่ละราย

เป็นการแสดงรายละเอียดต่างๆของการพิมพ์ในแต่ละผู้ใช้บริการว่ามีอะไรบ้าง โดยจะแสดงรายการต่างๆ เช่น

- จำนวนหน้าทั้งหมดที่ได้ทำการพิมพ์
- ชนิดของเอกสารและชื่อไฟล์ของงานพิมพ์
- พริ้นเตอร์และชื่อเซิร์ฟเวอร์ที่ใช้ทำการพิมพ์
- วันและเวลาที่ทำการการพิมพ์

โดยแสดงออกมาเป็นรายงานในรูปของเท็กซ์ไฟล์(.txt) โดยจะแสดงออกมาทั้งหมดทุกครั้งที่ทำการพิมพ์

5.1.7 การแสดงยอดเงินคงเหลือ

เป็นการแสดงจำนวนเงินคงเหลือในปัจจุบันว่ามีจำนวนเท่าไร โดยจะสามารถแสดงได้ทั้งแบบรายบุคคลและแสดงออกมาแบบเป็นรายกลุ่มของผู้ใช้ได้ โดยจะแสดงออกมาเหมือนกับการแสดงรายละเอียดการพิมพ์ของผู้

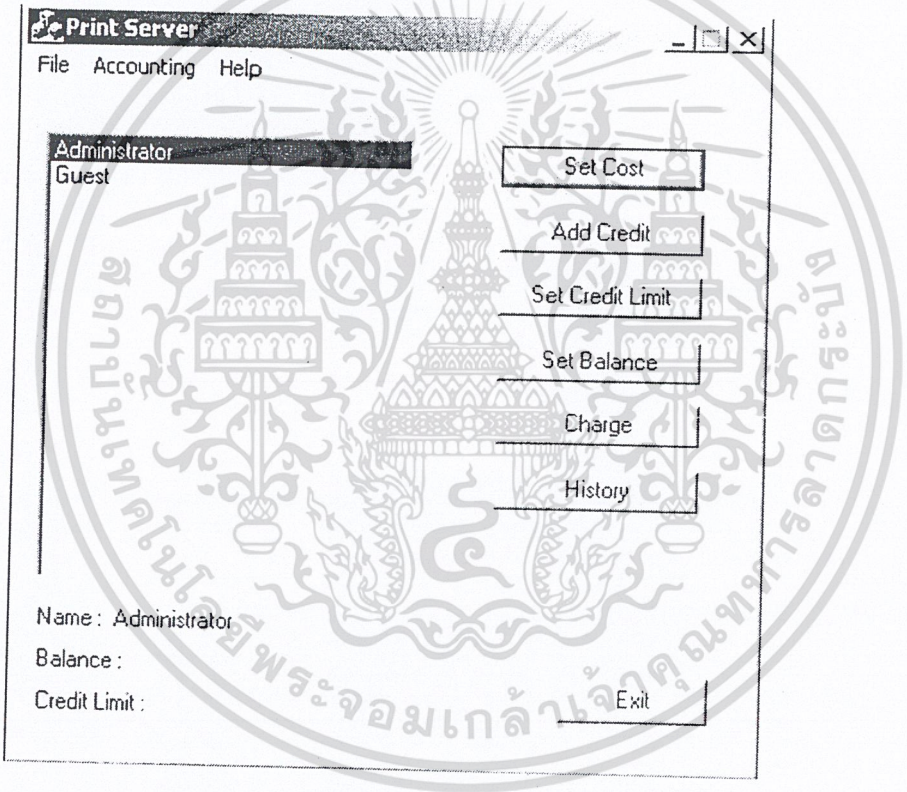
ใช้ คือเป็นรายงานแบบเท็กซ์ไฟล์(.txt) เช่นกัน

5.2 คุณสมบัติอื่นๆของโปรแกรม

จากที่ได้กล่าวถึงคุณสมบัติหลักๆของโปรแกรมจากหัวข้อที่ผ่านมานั้น เป็นการแสดงถึงคุณสมบัติที่สำคัญของโปรแกรม ในหัวข้อนี้จะขอกล่าวถึงคุณสมบัติอื่นๆ ที่เป็นองค์ประกอบร่วมในโปรแกรมนี้

5.2.1 หน้าจอหลักของโปรแกรม

แสดงรายละเอียดต่างๆ ของรายการทำงานที่ตัวโปรแกรมมี โดยรายละเอียดต่างๆของแต่ละการทำงานนั้นได้ทำการอธิบายไปในหัวข้อที่ผ่านมา



รูปที่ 5.6 แสดงรูปหน้าจอหลักของโปรแกรม

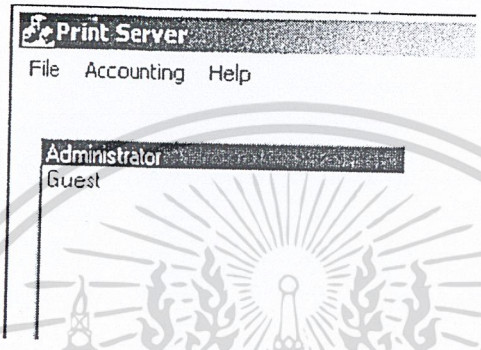
จากรูปหน้าจอหลักที่แสดงด้านบน จะขออธิบายถึงองค์ประกอบร่วมของโปรแกรม 3 ส่วน ดังนี้

- เมนูบาร์
- ลิสต์บ็อกซ์ แสดงรายนามผู้ใช้ในระบบ
- สแตติกเท็กซ์ แสดงรายนามผู้ใช้ที่ทำการเลือกและแสดงยอดคงเหลือปัจจุบัน

5.2.2 เมนูบาร์

ประกอบด้วยการทำงาน 3 ส่วน ดังนี้

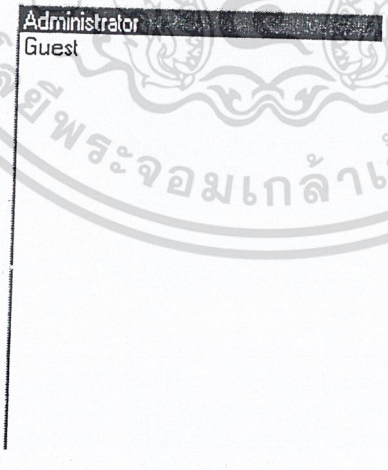
- File : ใช้สำหรับเปิดไฟล์ต่างๆ เช่น เท็กซ์ไฟล์ที่เป็นรายงานรายละเอียดของการพิมพ์ และใช้สำหรับออกจากโปรแกรม
- Account : ใช้สำหรับจัดการ และเอียดต่างๆเกี่ยวกับรายชื่อของผู้ใช้
- Help : ใช้สำหรับเปิดระบบช่วยเหลือของโปรแกรม



รูปที่ 5.7 แสดงเมนูบาร์

5.2.3 ลิสต์บ็อกซ์

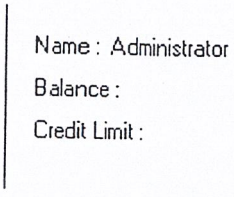
ใช้สำหรับแสดงรายนามของผู้ใช้ในระบบทั้งหมด โดยแสดงได้ดังรูป



รูปที่ 5.8 ลิสต์บ็อกซ์แสดงรายนามผู้ใช้ในระบบ

5.2.4 สแตติกเท็กซ์

แสดงรายละเอียดสามส่วน คือ (1)แสดงรายนามของผู้ใช้ที่ถูกเลือก (2)แสดงจำนวนยอดเงินคงเหลือปัจจุบันของผู้ใช้ที่ถูกเลือก (3)แสดงยอดเงินต่ำสุดที่ผู้ใช้สามารถทำการพิมพ์ได้ แสดงดังรูป



รูปที่ 5.9 สแตติกเท็กซ์แสดงรายละเอียดของผู้ใช้ที่ถูกเลือก

จากที่ได้กล่าวมาข้างต้นทั้งหมดนั้น เป็นเพียงการกล่าวถึงคุณสมบัติโดยทั่วไปของโปรแกรม แต่ไม่ได้กล่าวลึกลงไปถึงรายละเอียดของการทำงานในแต่ละส่วน ซึ่งในแต่ละส่วนก็มีการทำงานที่แตกต่างกันออกไป โดยในบทต่อไปจะขอกล่าวถึงรายละเอียดของการทำงานในแต่ละส่วน

บทที่ 6

รายละเอียดต่างๆของโปรแกรม

ในบทนี้จะกล่าวถึงรายละเอียดของการทำงานของโปรแกรมในแต่ละส่วน ว่ามีการทำงานอย่างไรบ้าง แต่ละส่วนมีหน้าที่การทำงานอย่างไร โดยมีรายละเอียดดังนี้

6.1 รูปแบบของโปรแกรม

ในโปรแกรมนี้ เป็นการเขียนโปรแกรมในประเภท Dialog-Based ซึ่งก็คือโปรแกรมที่มีไดอะล็อกเป็นวินโดวส์หลักนั่นเอง โดยโปรแกรมนี้จะใช้เครื่องมือตัวหนึ่งของ Visual C ++ ซึ่งก็คือ AppWizard และ ClassWizard โดยเราจะใช้ AppWizard สร้างซอร์สโค้ดต้นแบบขึ้นมา เพื่อให้การพัฒนาโปรแกรมนั้นสามารถทำได้ง่ายและรวดเร็วยิ่งขึ้น ซอร์สโค้ดที่ AppWizard สร้างขึ้นจะใช้คลาสจาก MFC เป็นหลัก

AppWizard เป็นอุปกรณ์ตัวหนึ่งที่ใช้สำหรับสร้างซอร์สโค้ดโปรแกรมต้นแบบ ซึ่งจะช่วยอำนวยความสะดวกให้เป็นอย่างมาก เพราะไม่จำเป็นต้องเขียนโค้ดโปรแกรมด้วยตนเอง การใช้งาน AppWizard ก็สามารถทำได้ง่ายๆ คือ AppWizard จะถามเกี่ยวกับรูปแบบของโปรแกรม ตัวเลือกที่ต้องการ จากนั้น AppWizard ก็จะสร้างซอร์สโค้ดโปรแกรมขึ้นมาให้โดยอัตโนมัติ ซึ่งสามารถนำซอร์สโค้ดนี้ไปใช้ได้ทันที

ซอร์สโค้ดที่ AppWizard สร้างขึ้นจะเป็นโค้ดโปรแกรมที่ใช้ MFC เป็นหลัก และ AppWizard จะจัดโครงสร้างของไฟล์ในโปรเจ็กต์ต่างๆให้ง่ายต่อการทำความเข้าใจ เพราะส่วนที่เป็นการประกาศคลาสและตัวแปรต่างๆจะถูกเก็บอยู่ในไฟล์ส่วนหัว (.H) และวนที่เป็นเนื้อหาของฟังก์ชั่นจะเก็บไว้ในไฟล์โปรแกรม (.CPP) โดยจะแยกกันอยู่คนละไฟล์ จึงทำให้ง่ายต่อการแก้ไขและดัดแปลงซอร์สโค้ด

6.2 รายละเอียดของโปรแกรม

ในส่วนนี้ขอกล่าวถึงรายละเอียดของโปรแกรมในส่วนต่างๆ โดยจะแบ่งออกตามไดอะล็อกต่างๆที่มีในโปรแกรม โดยจะอธิบายถึงคลาส รีซอร์สและไฟล์ที่เกี่ยวข้องกับไดอะล็อกนั้น โดยมีรายละเอียดดังนี้

6.2.1 ไดอะล็อกหลัก

- **คลาส**

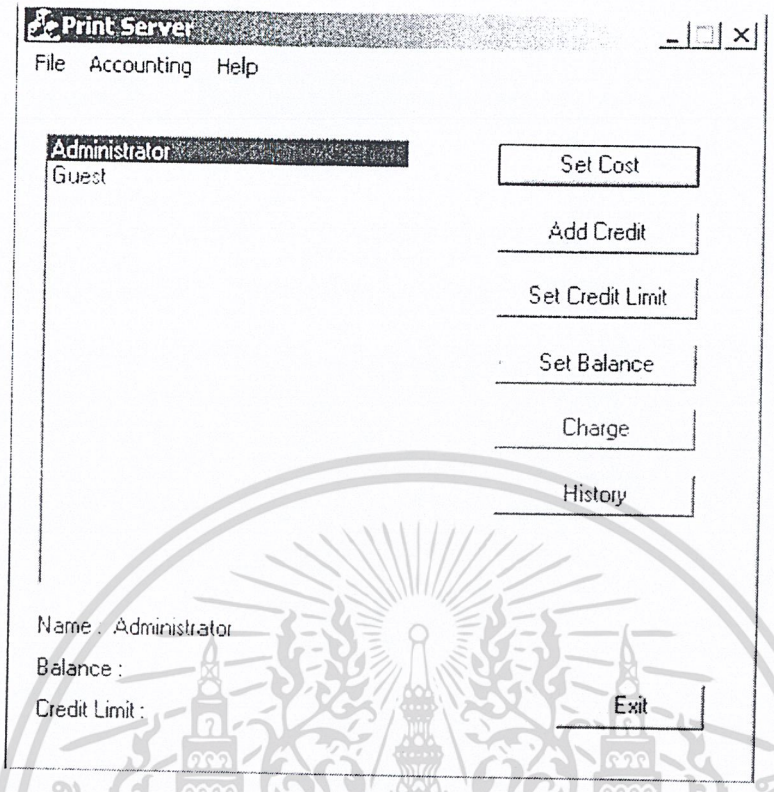
  CBasicApp , CBasicDlg

- **ไฟล์**

  Basic.h , Basic.cpp , BasicDlg.h , BasicDlg.cpp

- **รีซอร์ส**

  IDD_BASIC_DIALOG



รูปที่ 6.1 ไดอะล็อกหลักของโปรแกรม

6.2.2 ไดอะล็อก Set Cost

- คลาส

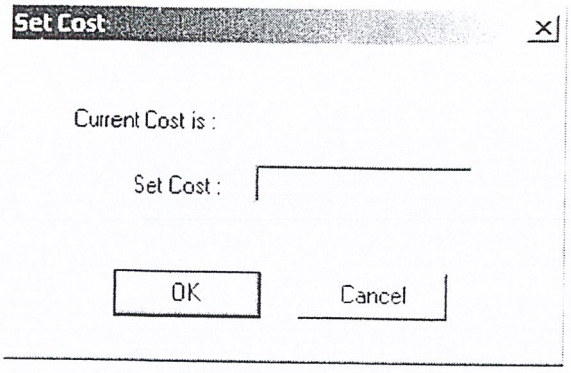
  CSetCreditDlg

- ไฟล์

  SetCreditDlg.h , SetCreditDlg.cpp

- รีซอร์ส

  IDD_SETCREDIT_DIALOG



รูปที่ 6.2 ไดอะล็อก Set Cost

6.2.3 ไดอะล็อก Add Credit

- คลาส

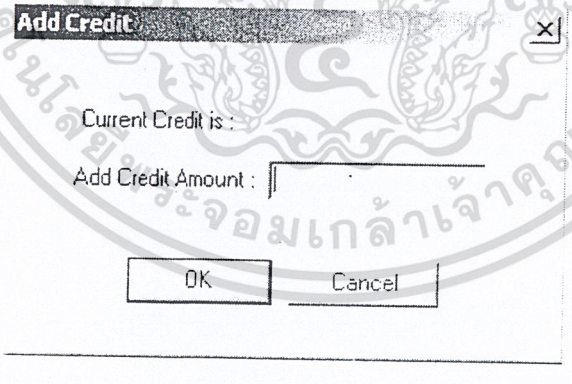
  CAddcreditDlg

- ไฟล์

  AddcreditDlg.h , AddcreditDlg.cpp

- รีซอร์ส

  IDD_ADDCREDIT_DIALOG



รูปที่ 6.3 ไดอะล็อก Add Credit

6.2.4 ไดอะล็อก Set Credit Limit

- คลาส

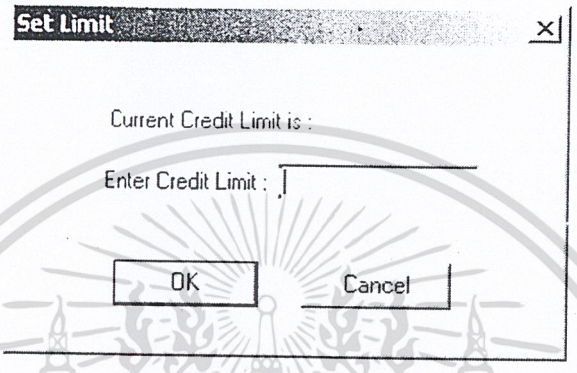
  CLimitDlg

- ไฟล์

  LimitDlg.h , LimitDlg.cpp

- รีซอร์ส

  IDD_SETLIMIT_DIALOG



รูปที่ 6.4 ไดอะล็อก Set Limit

6.2.5 ไดอะล็อก Set Balance

- คลาส

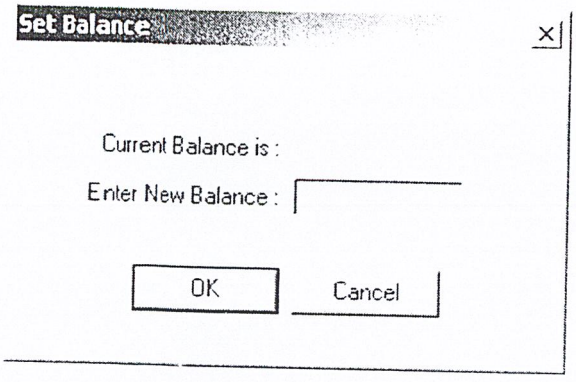
  CSetBalanceDlg

- ไฟล์

  SetBalanceDlg.h , SetBalanceDlg.cpp

- รีซอร์ส

  IDD_SETBALANCE_DIALOG



รูปที่ 6.5 ไดอะล็อก Set Balance

6.2.6 ไดอะล็อก Charge

- คลาส

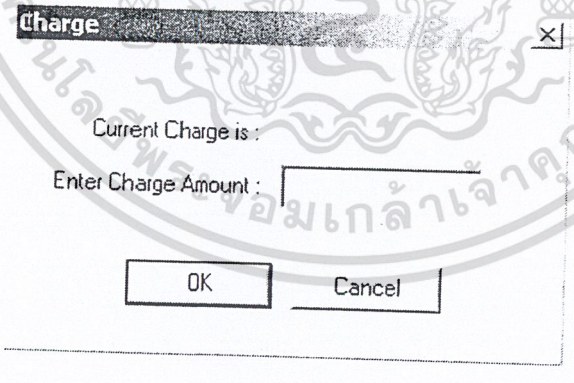
  CChargeDlg

- ไฟล์

  ChargeDlg.h , ChargeDlg.cpp

- รีซอร์ส

  IDD_CHARGE_DIALOG



รูปที่ 6.6 ไดอะล็อก Charge

6.2.7 ลิสต์บ็อกซ์ แสดงรายนามผู้ใช้ระบบ

- คลาส

  CNetInfo

- ไฟล์

  NetInfo.h , NetInfo.cpp

- รีซอร์ส

  IDC_LISTBOX

จากที่แสดงรายละเอียดต่างๆของโปรแกรมที่ผ่านมาข้างต้น ก็ครอบคลุมทุกคลาสที่มีในโปรแกรมแล้ว ซึ่งรายละเอียดของคลาสต่างๆ ว่ามีเมมเบอร์ฟังก์ชั่น และดาต้าเมมเบอร์อะไรบ้างนั้น และไฟล์ต่างๆและรีซอร์สทั้งหมดที่มีในโปรแกรมนั้น สามารถที่จะดูรายละเอียดเหล่านั้นได้จากซอร์สโค้ดโปรแกรม

บรรณานุกรม

Mickey Williams and David Hamilton, Programming Windows NT4 UNLEASHED, First Edition, Sams Publishing, 1999

นิรุธ อำนวยศิลป์, คู่มือการเขียนโปรแกรม Visual C++ Version 6.0 ฉบับเพื่อใช้งานจริง, กรุงเทพฯ : ซัคเซส มีเดีย จำกัด, 2542.

ยุทธนา ลีลาศวัฒนกุล, คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ 6.0 ฉบับโปรแกรมเมอร์, กรุงเทพฯ : อินโฟเพรส, 2544.

ศราวุฒิ ทรงเจริญ, รอบรู้ Windows NT Server 4, กรุงเทพฯ : โปรวิชั่น 2542.

เกษมสันต์ พาณิชยการ, C + + และหลักการของ OOP ฉบับเริ่มต้น, กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2537

David White and Kenn Scribner and Eugene Olafsen, MFC Programming with Visual C ++ 6 Unleashed, Sams Publishing, 1999