บทที่ 5

การพัฒนาโปรแกรมในส่วนให้บริการ

ส่วนของการทำงานภายใต้ระบบปฏิบัติการดอสบนเครื่องคอมพิวเตอร์ระดับพีซี คือชุดโปรแกรมให้บริการรหัสผ่านแบบใช้ครั้งเดียว (PC-based one-time password service) ซึ่งประกอบด้วย

1. ส่วนให้บริการตรวจสอบรหัสผ่านและบริการบำรุงรักษาระบบรหัสผ่าน โดยทำการเปิดช่องทางการสื่อสารเพื่อรอรับการขอตรวจสอบรหัสผ่านหรือบริการอื่น ๆ ที่ขอบริการมาจากโปรแกรมล็อกอินที่ทำงานภายใต้ระบบยูนิกซ์ และติดต่อสื่อสารโดยใช้โพรโตคอลทีซีพี/ไอพี

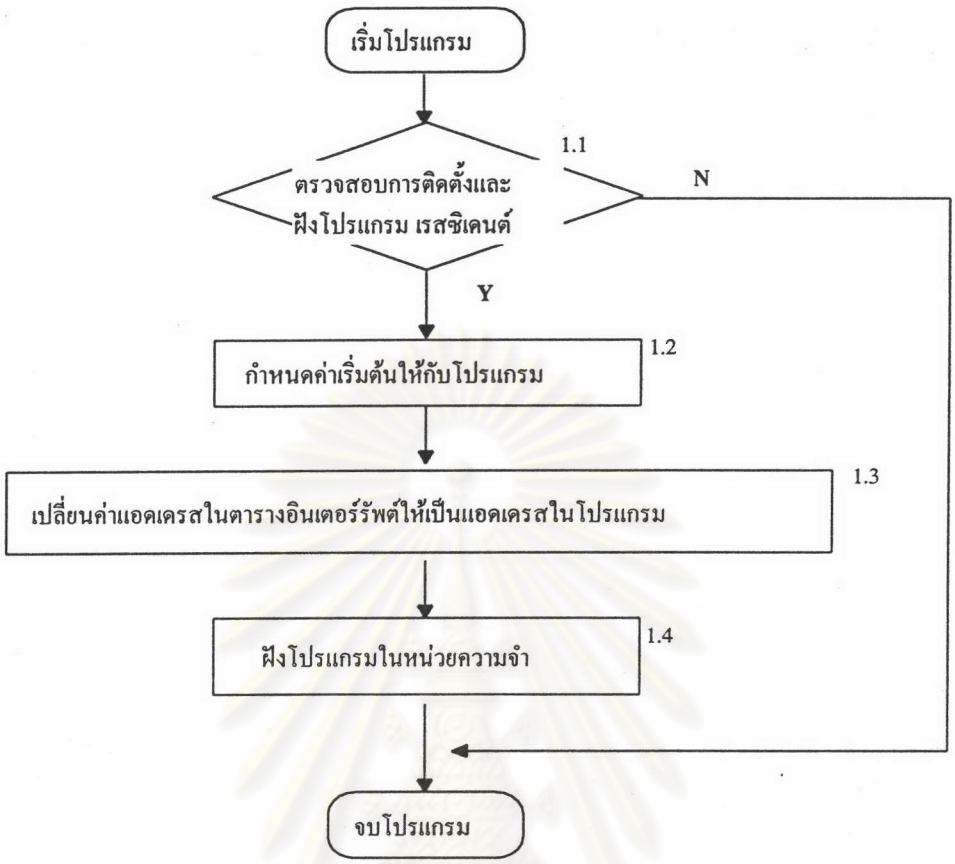
2. ส่วนบำรุงรักษาระบบรหัสผ่าน โดยโปรแกรมบำรุงรักษาระบบรหัสผ่านจะมีการรับคำสั่งผ่านทางหน้าจอของเครื่องบริการรหัสผ่าน เพื่อทำให้ระบบการให้บริการรหัสผ่านแบบใช้ครั้งเดียวสามารถดำเนินงานได้ต่อไปอย่างถูกต้องและมีประสิทธิภาพ การใช้งานโปรแกรมนี้จะถูกใช้ได้เฉพาะผู้ดูแลระบบหรือบุคคลที่ได้รับมอบอำนาจเท่านั้น

ในบทนี้จึงกล่าวถึง การพัฒนาโปรแกรมเรสซิเดนต์และการพัฒนาส่วนของโปรแกรมที่ติดต่อสื่อสารกับเครือข่ายที่มีโพรโตคอลทีซีพี/ไอพี

การพัฒนาโปรแกรมเรสซิเดนต์

เนื่องจากการทำงานทั้ง 2 ส่วนถูกออกแบบให้ทำงานพร้อมกัน ทฤษฎีสำคัญที่เกี่ยวข้อง คือ การเขียนโปรแกรมเรสซิเดนต์ หรือโปรแกรมประเภทฝังตัวในหน่วยความจำ ซึ่งแบ่งการพัฒนาโปรแกรมออกเป็น 2 ส่วน คือ การติดตั้งและฝังโปรแกรมเรสซิเดนต์ไว้ในหน่วยความจำและการเรียกโปรแกรมเรสซิเดนต์ขึ้นมาใช้งาน

1. การติดตั้งและฝังโปรแกรมเรสซิเดนต์ไว้ในหน่วยความจำ โดยใช้รูป 5.1 แสดงขั้นตอนการติดตั้งและฝังโปรแกรมในหน่วยความจำพร้อมคำอธิบาย



รูปที่ 5.1 แสดงลำดับการทำงานของการติดตั้งและฝังโปรแกรมเรสซิเดนต์ในหน่วยความจำ

1.1 ตรวจสอบการติดตั้งและฝังโปรแกรมเรสซิเดนต์ โดยใช้อินเตอร์รัพต์ 2fH (47) ฟังก์ชั่น 00H โดยมีการใช้งานรีจีสเตอร์ (register) ดังนี้

รีจีสเตอร์ AH = ระบุถึงโปรแกรมที่ต้องการตรวจสอบ (C0H - ffH)

รีจีสเตอร์ AL = 00H (เก็บหมายเลขฟังก์ชั่น) และผลของการตรวจสอบที่หาได้ (ถ้าเป็น 0 แสดงว่ามีโปรแกรมนั้นฝังอยู่ในหน่วยความจำอยู่แล้ว)

1.2 กำหนดค่าเริ่มต้นให้กับโปรแกรม

- InDOS flag เป็นแฟล็กขนาด 1 ไบต์ใช้แสดงสถานะการขอเข้าใช้ฟังก์ชั่นของดอส เนื่องจากระบบปฏิบัติการดอสสามารถทำงานได้เพียงงานเดียวในเวลาใดเวลาหนึ่ง จึงต้อง

มีการทดสอบแฟล็กนี้ก่อนที่จะขอเข้าใช้ฟังก์ชั่นของดอส ถ้าตรวจสอบแล้วพบว่ามีโปรเซสใช้ฟังก์ชั่นของดอสอยู่ โปรเซสอื่นจะไม่สามารถเข้าใช้ฟังก์ชั่นได้ ถึงแม้ว่าคำกล่าวนี้จะเป็นพื้นฐานของการขอเข้าใช้ฟังก์ชั่นของระบบปฏิบัติการดอส แต่มีข้อยกเว้นที่ทำให้สามารถใช้ฟังก์ชั่นของดอสในเวลาเดียวกันได้และข้อยกเว้นดังกล่าวจะอธิบายในภายหลัง วิธีอ้างถึง InDOS flag ทำโดยใช้อินเตอร์รัพต์ 21H (33) ฟังก์ชั่น 34H โดยมีการใช้งานรีจีสเตอร์ดังนี้

รีจีสเตอร์ AH = 34H (เก็บหมายเลขฟังก์ชั่น)

รีจีสเตอร์ ES:BX = เก็บค่าเซกเมนต์ (segment) และออฟเซต (offset) ของ InDOS flag ที่หาได้

- SDA (Swappable Data Area) ใช้แสดงสถานะการทำงานของโปรเซสรวมทั้งข้อมูลการเรียกใช้ฟังก์ชั่นของดอส ดังนั้นเมื่อต้องการสลับโปรเซสอื่นขึ้นมาทำงานจึงต้องทำการเก็บพื้นที่ส่วนนี้เพื่อนำกลับมาใช้งานภายหลังเมื่อต้องการสลับโปรเซสเดิมขึ้นมาทำงาน ข้อดีของการใช้ SDA คือ สามารถเข้าใช้ฟังก์ชั่นของดอสได้ถึงแม้ว่า InDOS flag จะแสดงว่ามีโปรเซสอื่นใช้งานฟังก์ชั่นของดอสอยู่ วิธีอ้างถึงหน่วยความจำส่วนนี้โดยใช้อินเตอร์รัพต์ 21H (33) ฟังก์ชั่น 5d06H โดยมีการใช้งานรีจีสเตอร์ดังนี้

รีจีสเตอร์ AH = 5d06H (เก็บหมายเลขฟังก์ชั่น)

รีจีสเตอร์ DS:SI = เก็บค่าเซกเมนต์ และออฟเซต

รีจีสเตอร์ CX = ขนาดหน่วยความจำเมื่อมีโปรเซสอื่นขอใช้ฟังก์ชั่นดอสอยู่มีขนาด 73cH ไบต์

รีจีสเตอร์ DX = ขนาดหน่วยความจำเมื่อไม่มีโปรเซสขอใช้ฟังก์ชั่นดอสอยู่มีขนาด 18H ไบต์

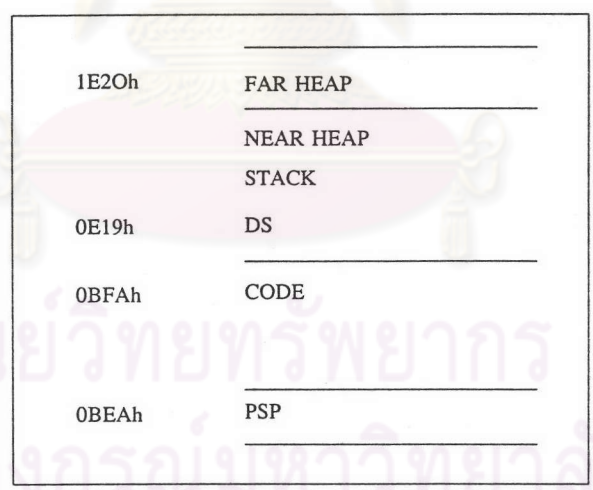
1.3 การเปลี่ยนแอดเดรสในตารางอินเตอร์รัพต์ให้เป็นแอดเดรสจากโปรแกรม ในระบบปฏิบัติการดอสซีพียูมีหน้าที่คอยให้บริการแก่อุปกรณ์ที่ร้องขอเข้ามาโดยใช้อินเตอร์รัพต์เป็นสัญญานในการส่ง เมื่ออุปกรณ์ต้องการทำงานจะส่งอินเตอร์รัพต์ไปยังซีพียูและที่ซีพียูจะมีตัวชี้บ่งโปรแกรมที่รองรับการทำงานนั้นไว้ในตารางอินเตอร์รัพต์ ซีพียูจะทำงานตามหน้าที่ของอินเตอร์รัพต์ที่ส่งเข้ามา

การหาแอดเดรสในตารางอินเตอร์รัพต์โดยใช้ฟังก์ชั่น _dos_getvect( ) และใช้ฟังก์ชั่น _dos_setvect() เพื่อนำเอาค่าแอดเดรสของโปรแกรมเรสซิเดนต์ไปแทนที่ภายในตารางอินเตอร์รัพต์ ซึ่งทั้งสองเป็นฟังก์ชั่นในไลบราลีฟังก์ชั่น (library function) ของภาษาซี

```
#include <dos.h>
void (interrupt far *) _dos_getvect(unsigned intnum);
void (interrupt far *) _dos_setvect(unsigned intnum);
```

1.4 การฝังโปรแกรมในหน่วยความจำ

1.4.1 จองหน่วยความจำที่ใช้เป็นสแต็ก (stack) ของโปรแกรมเรสซิเดนต์โดยใช้ฟังก์ชั่น malloc() จากนั้นนำขนาดสแต็กที่กำหนดไว้มาบวกเพิ่มกับพอย์นเตอร์ที่ชี้ไปยังสแต็กทำให้พอย์นเตอร์ชี้ไปยังส่วนล่างของสแต็กและถือว่าจุดนี้เป็นจุดเริ่มต้นของโปรแกรมเรสซิเดนต์ในหน่วยความจำ และรูปที่ 5.2 แสดงรูปแบบการติดตั้งโปรแกรมในหน่วยความจำ



รูปที่ 5.2 แสดงรูปแบบของโปรแกรมภาษาซีที่ถูกติดตั้งในหน่วยความจำ

1.4.2 หาตำแหน่งท้ายของโปรแกรมเรสซิเดนต์จากรีจีสเตอร์ DS ซึ่งเป็นค่าของดาต้าเซกเมนต์ (data Segment) ของโปรแกรมและตำแหน่งเริ่มต้นของโปรแกรมเรสซิเดนต์จากค่าของ _psp ซึ่งเป็นตำแหน่งเซกเมนต์ของ PSP (program segment prefix : มีขนาด 256 ไบต์ถูก

สร้างขึ้นเมื่อโปรแกรมถูกบรรจุลงในหน่วยความจำเพื่อใช้งาน ทำหน้าที่เก็บข้อมูลต่างที่เกี่ยวข้องกับโปรแกรม)

1.4.3 คำณวนขนาดโปรแกรมเรสซิเดนต์ที่ต้องเก็บในหน่วยความจำจากตำแหน่งท้ายโปรแกรมลบด้วยตำแหน่งต้นโปรแกรมและบวกด้วยขนาดของสแต็ก

1.4.4 การจบการทำงานแต่ไม่ทำการคืนหน่วยความจำแก่ระบบปฏิบัติการโดยใช้ฟังก์ชั่น _dos_keep()

```
void  _dos_keep( unsigned status,   // ค่าที่ส่งกลับให้แก่ระบบปฏิบัติการ
                 unsigned npara );  // ขนาดของหน่วยความจำที่เก็บ
```

2. การเรียกโปรแกรมเรสซิเดนต์ขึ้นมาใช้งาน

ส่วนของการทำงานภายใต้ระบบปฏิบัติการดอสประกอบด้วยการทำงานสองส่วนที่มีการทำงานพร้อมกันจึงมีโปรเซสที่ทำงาน 2 โปรเซสที่ทำงานสลับกัน โดยโปรเซสที่ทำงานด้านหน้าหรือโฟว์กราวน์โปรเซสเป็นโปรเซสที่ทำงานอยู่ตลอดเวลา และโปรเซสทำงานเบื้องหลังหรือแบ็คกราวน์โปรเซส เป็นโปรเซสที่จะเริ่มทำงานต่อเมื่อมีการส่งอินเตอร์รัพต์เข้ามายังซีพียู โดยเป็นอินเตอร์รัพต์ที่เรากำหนดการทำงานขึ้นใหม่แทนการทำงานเดิม ต่อไปแสดงขั้นตอนการสลับเอาแบ็คกราวน์โปรเซสขึ้นมาทำงานขณะที่โฟร์กราวน์โปรเซสกำลังทำงานอยู่

2.1 ทำการเก็บสแต็กของโฟว์กราวน์โปรเซสและนำสแต็กของแบ็คกราวน์โปรเซสขึ้นมาใช้

2.2 ทำการเก็บ SDA ของโฟว์กราวน์โปรเซสโดยพิจารณาขนาดจาก InDOS flag ถ้ามีโปรเซสอื่นขอใช้ฟังก์ชั่นของดอสอยู่ขนาดของ SDA ที่ต้องเก็บคือ 73cH ไบต์ และเมื่อไม่มีโปรเซสขอใช้ฟังก์ชั่นของดอสอยู่จะมีขนาด 18H ไบต์

2.3 ทำการเก็บตำแหน่ง PSP ของโฟว์กราวน์โปรเซสโดยใช้อินเตอร์รัพต์ 21H (33) ฟังก์ชั่น 62H โดยมีการใช้งานรีจีสเตอร์ดังนี้

รีจิสเตอร์ AH = 62H (เก็บหมายเลขฟังก์ชั่น)
รีจีสเตอร์ BX = ตำแหน่งของเซกเมนต์ของโปรเซสที่ได้กลับมา

2.4 ติดตั้งตำแหน่งของ PSP ของแบ็คกราวน์โปรเซสโดยใช้อินเตอร์รัพต์ 21H ฟังก์ชั่น 50H โดยมีการใช้งานรีจีสเตอร์ดังนี้

รีจีสเตอร์ AH = 50H (เก็บหมายเลขฟังก์ชั่น)
รีจีสเตอร์ BX = ตำแหน่งเซกเมนต์ของโปรเซสที่จะติดตั้ง

2.5 ติดตั้งตำแหน่งของ DTA ของแบ็คกราวน์โปรเซสโดยใช้อินเตอร์รัพต์ 21H ฟังก์ชั่น 1aH โดยมีการใช้งานรีจีสเตอร์ดังนี้

รีจีสเตอร์ AH = 1aH (เก็บหมายเลขฟังก์ชั่น)
รีจีสเตอร์ DX = ตำแหน่งออฟเซตของ DTA (80Hจากตำแหน่งของ PSP)
รีจีสเตอร์ DS = ตำแหน่งเซกเมนต์ของ DTA (ตำแหน่งของ PSP)

โดยที่ DTA (data transfer area) เป็นหน่วยความจำขนาด 128 ไบต์อยู่ที่ตำแหน่ง 0x80h ของ PSP ทำหน้าที่เก็บข้อมูลที่ใช้รับส่งระหว่างหน่วยความจำกับจานบันทึกข้อมูล (disk)

2.6 เข้าสู่การทำงานของแบ็คกราวน์โปรเซส

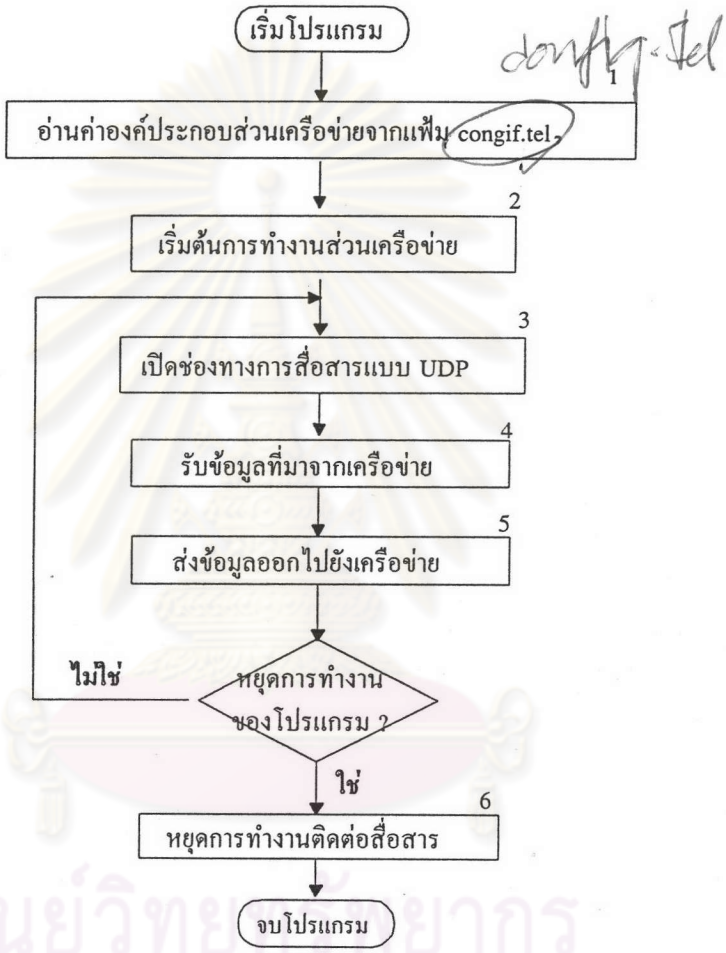
2.7 ทำการติดตั้ง SDA และ PSP ของโฟว์กราวน์โปรเซส

2.8 ติดตั้งสแต็กของโฟว์กราวน์โปรเซสและเก็บสแต็กของแบ็คกราวน์โปรเซส

2.9 กลับสู่การทำงานของโฟว์กราวน์โปรเซส

การพัฒนาส่วนของโปรแกรมที่ติดต่อสื่อสารกับเครือข่ายที่มีโพรโตคอลทีซีพี/ไอพี

ทำการพัฒนาโปรแกรมโดยใช้ไลบรารี่ฟังก์ชั่นของ NCSA Telnet 2.307 ซึ่งประกอบด้วยฟังก์ชั่นต่าง ๆ ที่สำคัญโดยจะกล่าวเป็นลำดับตามขั้นตอนดังแสดงในรูป 5.3



รูป 5.3 แสดงขั้นตอนการทำงานในส่วนติดต่อสื่อสาร

1. อ่านค่าองค์ประกอบส่วนเครือข่ายของโปรแกรมจากแฟ้ม config.tel เพื่อเก็บค่าเหล่านี้ไว้ในตัวแปรกำหนดค่า โดยใช้ฟังก์ชั่น Shostfile()

2. เริ่มต้นการทำงานส่วนเครือข่าย เป็นการกำหนดค่าเริ่มต้นต่าง ๆ ให้กับตัวแปร เช่น การกำหนดชนิดของฮาร์ดแวร์ที่ใช้ ไอพีแอดเดรส ไอพีสับเน็ตมาค (IP subnetmask) การเริ่มต้นลำดับเหตุการณ์ (eventqueue) โดยใช้ฟังก์ชั่น Snetinit() เป็นฟังก์ชั่นเริ่มต้นการทำงานกับส่วนเครือข่าย และเรียกใช้ฟังก์ชั่นอื่น ๆ อีกดังต่อไปนี้

2.1 ฟังก์ชั่น netinit() เพื่อกำหนดฮาร์ดแวร์และกำหนดค่าเริ่มต้นให้กับตัวแปร เช่น ไอพีแอดเดรส ไอพีสับเน็ตมาค

2.2 ฟังก์ชั่น netparms() เพื่อกำหนดฮาร์ดแวร์ให้ตรงกับค่าในแฟ้ม config.tel และมีรูปแบบดังนี้

```
netparms(irq, address, ioaddress);
int irq;          /* อินเตอร์รัพต์ที่ใช้ในการติดต่อสื่อสาร  */
int address;      /* ตำแหน่งเริ่มต้นที่แพ็คเก็จไดเวอร์อยู่ในหน่วยความจำ */
int ioaddress;    /*  พอร์ตหรือตำแหน่งที่ใช้ติดต่อสื่อสารของฮาร์ดแวร์  */
```

2.3 ฟังก์ชั่น Ssetgates() เพื่อกำหนดเกตเวย์ (gateway) ที่ได้จากแฟ้ม config.tel

2.4 ฟังก์ชั่น neteveninit() ทำการเริ่มต้นลำดับเหตุการณ์เพื่อใช้ในการตรวจสอบสถานะระหว่างการเชื่อมต่อ

3. เปิดช่องทางการติดต่อสื่อสารแบบยูดีพี คือ ยูดีพีพอร์ตที่ 6669 เพื่อรอรับการเชื่อมต่อที่มาจากเครื่องคอมพิวเตอร์แม่ข่าย โดยใช้ฟังก์ชั่น netulisten() ที่มีรูปแบบดังนี้

```
netulisten(port);
int  port;              /*  port = 6669  */
```

4. ตรวจสอบสถานะและรับอักขระที่มาจากเครือข่าย ไลบรารีฟังก์ชั่นของ NCSA Telnet ได้กำหนดข้อผิดพลาดที่เกิดขึ้นหรือสถานะการเชื่อมต่อของเครือข่ายนั้นเป็นลักษณะของชั้น (classes) และเหตุการณ์ (events) เพื่อตอบสนองเหตุการณ์ดังกล่าวได้ถูกต้อง

4.1 การตรวจสอบเหตุการณ์ที่เกิดขึ้นในเน็ตเวิร์คโดยใช้ฟังก์ชั่น Sgetevent() ที่มีรูปแบบดังนี้

```
theevent  =  Sgetevent(class, theclass, dat);
int theevent;     /*  เหตุการณ์ที่เกิดขึ้น  */
```

```
int class;      /*  ลักษณะของชั้นที่ต้องการค้นหา        */
int theclass;   /*  ลักษณะของชั้นที่เกิดขึ้น            */
char *dat;      /*  ข้อมูลที่เกี่ยวข้องกับเหตุการณ์ที่เกิดขึ้น  */
```

เมื่อเรียกใช้ฟังก์ชั่นนี้ค่าที่กลับมาคือ เหตุการณ์ที่เกิดขึ้นและถ้าเป็น 0 แสดงว่าไม่มีเหตุการณ์ใด ๆ เกิดขึ้น

4.2 ทำการอ่านข้อมูลที่มาจากเครือข่ายโดยใช้ฟังก์ชั่น neturead() ที่มีรูปแบบดังนี้

```
cnt = neturead(buf);
char  *buf;  /* ข้อมูลที่อ่านเข้ามาจากเครือข่ายมีขนาดน้อยกว่า 512 ไบต์ */
int  cnt;      /* ขนาดของข้อมูลที่อ่านเข้ามาจริงๆ  */
```

5. ส่งข้อมูลออกไปยังเครือข่ายโดยใช้ฟังก์ชั่น netusend() ที่มีรูปแบบดังนี้

```
netusend(ipnum, port, retport, buf, len);
char  *ipnum[4];          /*  IP number ของเครื่องที่ต้องการส่งไป   */
int port;                 /*  พอร์ตที่ต้องการส่งข้อมูลไป            */
int retport;              /*  พอร์ตที่ใช้รับข้อมูลกลับเข้ามา        */
char  *buf;               /*  ข้อมูลที่ต้องการส่งออกไปยังเครือข่าย  */
int len;                  /*  ขนาดของข้อมูลที่ต้องการส่งออก         */
```

6. หยุดการติดต่อสื่อสารโดยใช้ฟังก์ชั่น netshut() เป็นฟังก์ชั่นสุดท้ายก่อนทำการเลิกโปรแกรม

ในบทนี้ได้แสดงถึงการพัฒนาโปรแกรมในส่วนให้บริการ บทต่อไปจะกล่าวถึงการทดสอบระบบรหัสผ่านแบบใช้ครั้งเดียว