บทที่ 4

การพัฒนาระบบ

การพัฒนาระบบได้นำสิ่งที่ออกแบบไว้มาดำเนินการพัฒนาโดยใช้ภาษา C ภายใต้สภาพแวดล้อมบนวินโดวส์ โดยอาศัยเครื่องมือช่วยคือวินซ็อค ซึ่งมีรายละเอียดตามหัวข้อ 4.1-4.8

4.1 โครงร่างการเขียนโปรแกรมบนเครือข่าย

โปรแกรมประยุกต์ทุกๆ โปรแกรมที่ทำงานบนเครือข่ายไม่ว่าจะทำงานในลักษณะสถานีใช้บริการหรือสถานีให้บริการ จะมีขั้นตอนในการเขียนโปรแกรมหลักๆ ดังนี้

-เปิดใช้ซ็อคเก็ต

-ให้ชื่อแก่ซ็อคเก็ต

-สร้างความสัมพันธ์กับซ็อคเก็ตอื่น

-ทำการรับและส่งข้อมูลระหว่างซ็อคเก็ต

-เลิกใช้ซ็อคเก็ต

จะเห็นว่าขั้นตอนดังกล่าวจะคล้ายกับการกระทำกับแฟ้มข้อมูล ซึ่งเปรียบเทียบให้เห็นได้ดังตารางข้างล่างนี้

File I/O	**Network I/O**
เปิดแฟ้ม	เปิดซ็อคเก็ต
	ให้ชื่อแก่ซ็อคเก็ต
	สร้างความสัมพันธ์กับซ็อคเก็ตอื่น
อ่านและเขียน	รับและส่งข้อมูลระหว่างซ็อคเก็ต
ปิดแฟ้ม	เลิกใช้ซ็อคเก็ต

ตาราง 4-1 เปรียบเทียบระหว่างการเปิดใช้แฟ้มข้อมูลกับการเปิดใช้ซ็อคเก็ต

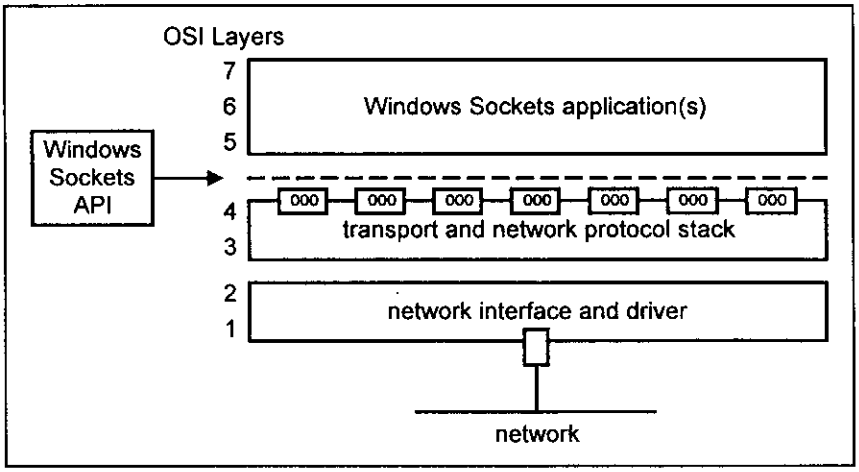
จากตาราง 4-1 จะเห็นว่าการเปิดแฟ้มข้อมูลจะเปรียบเทียบได้กับการกระทำ 3 อย่างด้วยกันบนเครือข่าย คือ การเปิดใช้ซ็อคเก็ต การให้ชื่อแก่ซ็อคเก็ต และการสร้างความสัมพันธ์กับซ็อคเก็ตอื่น หลังจากเปิดแฟ้มข้อมูลแล้วสามารถที่จะใช้งานได้ทันที แต่หลังจากเปิดซ็อคเก็ตแล้วยังไม่สามารถใช้งานได้ จะต้องทำการให้ชื่อกับซ็อคเก็ต และสร้างความสัมพันธ์กับซ็อคเก็ตอื่นจึงจะใช้งานได้ แต่สิ่งที่คล้ายกันก็คือหลังจากที่ได้เปิดใช้ซ็อคเก็ตหรือแฟ้มแล้วจะมีการส่งค่าแฮนเดิล (Handle) กลับมา โดยค่านี้จะใช้ในการเข้าถึงทรัพยากรที่จะเรียกใช้ โดยจะมีการซ่อนรายละเอียดของทรัพยากรไว้

การกระทำที่คล้ายกันอีกอย่างก็คือซ็อคเก็ตจะมีการส่งและรับข้อมูล ซึ่งจะคล้ายกับแนวคิดของแฟ้มที่จะมีการอ่านและเขียน โดยการกระทำดังกล่าวใช้ตัวแฮนเดิลในการเข้าถึงทรัพยากร ซึ่งทรัพยากรในที่นี้ คือ แฟ้มหรือซ็อคเก็ต และเมื่อทำการปิดแฟ้มหรือซ็อคเก็ตแล้วก็จะทำการคืนทรัพยากรให้แก่ระบบเพื่อนำไปใช้ต่อไป แต่การปิดแฟ้มนั้นยังคงมีแฟ้มนั้นอยู่ แต่การปิดซ็อคเก็ตจะหมายถึงการทำลายซ็อคเก็ตนั้น

สรุปสั้นๆ ก็คือการกระทำของซ็อคเก็ตจะคล้ายกับการกระทำกับแฟ้มข้อมูล

4.2 การเปิดใช้ซ็อคเก็ต

ซ็อคเก็ตเป็นจุดปลายของการติดต่อสื่อสารที่ถูกสร้างขึ้นจากโปรแกรม ซึ่งสามารถเทียบได้กับการเชื่อมต่อระหว่างคอมพิวเตอร์บนเครือข่ายโดยใช้แผงวงจรฮาร์ดแวร์ ซ็อคเก็ตจะทำให้โปรแกรมประยุกต์สามารถเชื่อมต่อกับเครือข่ายได้ นอกจากนี้สามารถใช้งานหลายๆ ซ็อคเก็ตได้ในเวลาเดียวกันโดยอาศัยแผงวงจรฮาร์ดแวร์ของเครือข่ายเพียงอันเดียว แสดงได้ดังภาพประกอบ 4-1



ภาพประกอบ 4-1 แผงวงจรฮาร์ดแวร์เพียงอันเดียว แต่สามารถใช้งานได้หลายๆ ซ็อคเก็ต

ที่มา: Quinn, Bob. and Shute, Dave., 1995 : 51

ฟังก์ชันของวินซ็อค API ที่เกี่ยวข้องกับการเปิดใช้ซ็อคเก็ต คือ socket() ซึ่งมีรายละเอียดดังนี้

socket()

ทั้งคีย์บอร์ดเทรดและเมนเทรดจะต้องใช้ซ็อคเก็ตในการเข้าถึงเครือข่าย ในการเปิดใช้งานซ็อคเก็ตจะใช้ฟังก์ชัน socket() แสดงได้ดังภาพประกอบ 4-2 โดยฟังก์ชันนี้มีรูปแบบดังนี้

```
SOCKET PASCAL FAR socket (int af,          /* protocol suit */
                          int type,        /* protocol type */
                          int protocol);   /* protocol name */
```

af : ซ็อคเก็ตโดเมน

type : ชนิดของซ็อคเก็ต

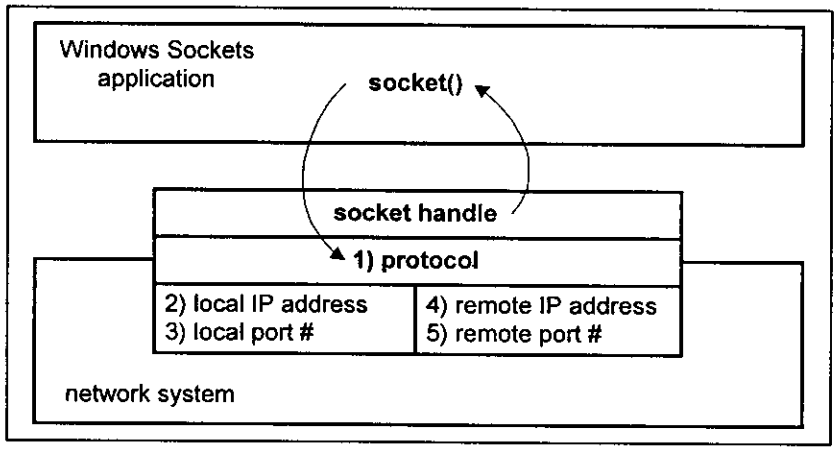
protocol : โปรโตคอลที่ใช้งาน

ในการเปิดใช้งานซ็อคเก็ตจะมีการส่งค่าซ็อคเก็ตเดสคริพเทอร์ (Socket Descriptor) กลับมาโดยใช้ค่านี้ในการเข้าถึงซ็อคเก็ต

พารามิเตอร์ af จะแทนตระกูลของโปรโตคอลที่จะใช้งาน ในที่นี้จะใช้ PF_INET สำหรับการใช้งานโปรโตคอล TCP/IP

พารามิเตอร์ type จะแทนชนิดของซ็อคเก็ตที่จะใช้งาน ในที่นี้จะแทนด้วย SOCK_STREAM สำหรับการใช้ TCP

พารามิเตอร์ protocol จะแทนโปรโตคอลที่จะใช้งาน ในที่นี้จะแทนด้วย IPPROTO_TCP



ภาพประกอบ 4-2 ฟังก์ชัน socket() จะต้องการชนิดของโปรโตคอลที่จะใช้งาน และจะส่งซ็อคเก็ตแฮนเดิลกลับมา

4.3 การให้ชื่อแก่ซ็อคเก็ตของเมนเทรด

เมนเทรดจะต้องทำการให้ชื่อแก่ซ็อคเก็ต ไม่เช่นนั้นแล้วคีย์บอร์ดเทรดจะไม่สามารถติดต่อมายังเมนเทรดได้ การให้ชื่อแก่ซ็อคเก็ตเป็นการกำหนดคุณลักษณะต่างๆ ของซ็อคเก็ตซึ่งมีอยู่ด้วยกัน 3 อย่าง คือ โปรโตคอล หมายเลขพอร์ตและเลขที่อยู่

โครงสร้าง sockaddr_in ของวินซ็อค API

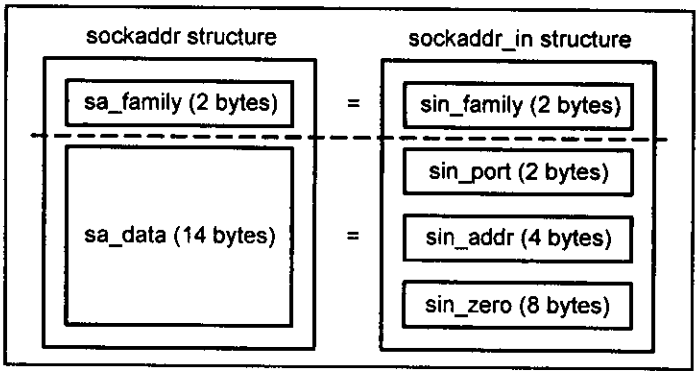
ในการให้ชื่อแก่ซ็อคเก็ต เมนเทรดจะต้องกำหนดค่าต่างๆ ให้กับโครงสร้างของซ็อคเก็ตแอดเดรสก่อน แล้วจึงเรียกใช้ฟังก์ชัน bind() โดยโครงสร้างดังกล่าวเป็นดังนี้

```
struct sockaddr_in {
    short          sin_family;          /* address family (AF_INET) */
    u_short        sin_port;            /* port (service) number */
    struct         in_addr  sin_addr;   /* IP address (32-bit) */
    char sin_zero[8];                   /* <unused filter> */
};
```

sin_family : ตระกูลของเลขที่อยู่

sin_port : หมายเลขพอร์ตขนาด 16 บิต ในแบบเน็ตเวิร์คออร์เดอร์

sin_addr : หมายเลขที่อยู่อินเตอร์เน็ตขนาด 32 บิต ในแบบเน็ตเวิร์คออร์เดอร์



ภาพประกอบ 4-3 โครงสร้างของ sockaddr และ sockaddr_in

ที่มา: Quinn, Bob. and Shute, Dave., 1995 : 55

โครงสร้าง sockaddr_in ใช้ในการกำหนดคุณลักษณะต่างๆ ของซ็อคเก็ต และให้ตัวพอยท์เตอร์ของคุณลักษณะดังกล่าวชี้ไปยังโครงสร้าง sockaddr

สำหรับโครงสร้าง in_addr ที่อยู่ใน sockaddr_in เป็นดังนี้

```
struct in_addr {
    union {
        struct { u_char s_b1, s_b2, s_b3, s_b4; } S_un_b;
        struct { u_short s_w1, s_w2; } S_un_w;
        u_long S_addr;
    } S_un;
};
```

ดังได้กล่าวแล้วว่าการให้ชื่อแก่ซ็อคเก็ตเป็นการกำหนดคุณลักษณะของซ็อคเก็ต 3 อย่างด้วยกัน คือ โปรโตคอล หมายเลขพอร์ต และเลขที่อยู่ IP ในการกำหนดโปรโตคอลจะกำหนดในขั้นตอนของการเรียกใช้ฟังก์ชัน socket() ส่วนการกำหนดหมายเลขพอร์ตและเลขที่อยู่ IP จะกำหนดไว้ในโครงสร้าง sockaddr_in แล้วทำการเรียกใช้ฟังก์ชัน bind()

bind()

ฟังก์ชัน bind() ของวินซ็อค API จะใช้ในการให้ชื่อแก่โลคอลซ็อคเก็ตด้วยค่าพารามิเตอร์ในโครงสร้าง sockaddr_in ฟังก์ชัน bind() มีโครงสร้างดังนี้

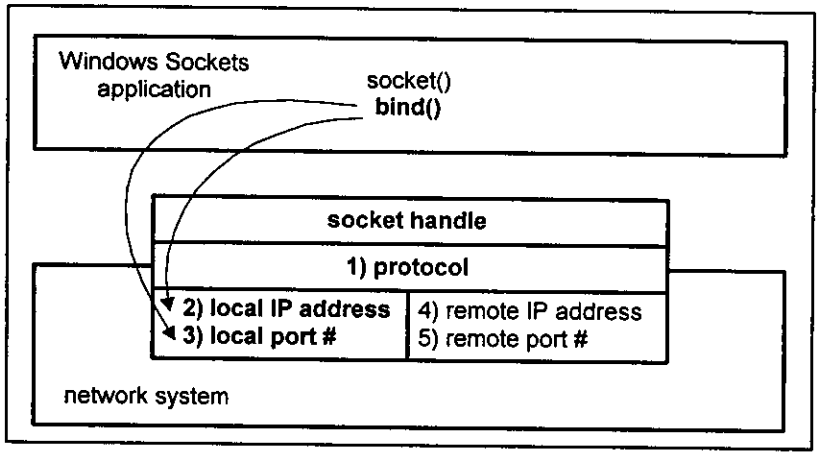
```
int PASCAL FAR bind (SOCKET s,          /* an unbound socket */
    struct sockaddr FAR *addr,          /* local port and IP addr */
    int namelen);                       /* addr structure length */
```

s : ซ็อคเก็ตแฮนเดิล

addr : พอยท์เตอร์ที่ชี้ไปยังโครงสร้างข้อมูล sockaddr_in

namelen : ขนาดของโครงสร้างข้อมูลที่ถูกชี้โดย addr

เมนเทรดจะต้องทำการให้ชื่อแก่ซ็อคเก็ต ดังภาพประกอบ 4-4 เพื่อให้คีย์บอร์ดเทรดสามารถติดต่อไปยังเมนเทรดได้



ภาพประกอบ 4-4 ฟังก์ชัน bind() จะกำหนดเลขที่อยู่ IP และ พอร์ตที่เป็นโลคอลให้กับโลคอลซ็อคเก็ต

4.4 การสร้างความสัมพันธ์ระหว่างซ็อคเก็ตของคีย์บอร์ดเทรดและเมนเทรด

ดังได้กล่าวแล้วว่าการเปิดใช้ซ็อคเก็ตจะกระทำทั้งในส่วนของคีย์บอร์ดเทรดและเมนเทรด ส่วนการให้ชื่อแก่ซ็อคเก็ตนั้นกระทำในส่วนของเมนเทรด ขั้นตอนถัดไปเมนเทรดจะต้องเตรียมให้ซ็อคเก็ตพร้อมที่จะรับข้อมูล และทางคีย์บอร์ดเทรดจะทำการส่งข้อมูลมา เมื่อคีย์บอร์ดเทรดพร้อมที่จะทำการส่งข้อมูลก็จะทำการสร้างความสัมพันธ์ระหว่างซ็อคเก็ตของคีย์บอร์ดเทรดและเมนเทรดเพื่อให้แต่ละเทรดรู้จักชื่อซ็อคเก็ตของกันและกัน

จากที่กล่าวมาแสดงได้ดังภาพประกอบ 4-5 ซึ่งจะมีองค์ประกอบอยู่ด้วยกัน 5 ส่วน ในการสร้างความสัมพันธ์ระหว่างซ็อคเก็ต คือ

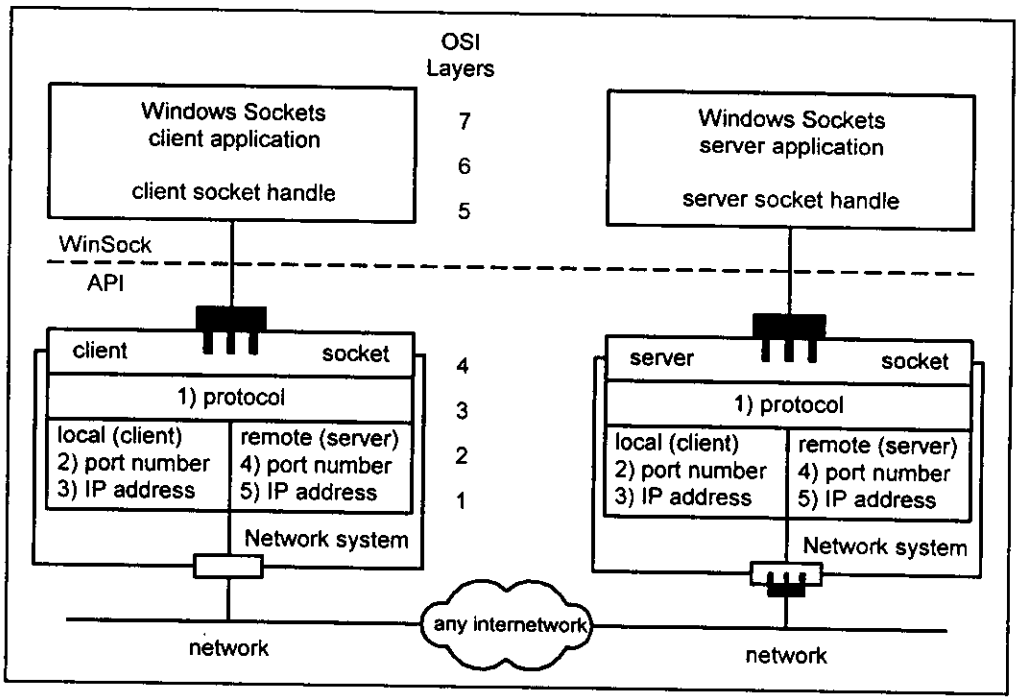
-โปรโตคอล (ต้องเป็นชนิดเดียวกันทั้งคีย์บอร์ดเทรดและเมนเทรด)

-เลขที่อยู่ IP ของคีย์บอร์ดเทรด

-หมายเลขพอร์ตของคีย์บอร์ดเทรด

-เลขที่อยู่ IP ของเมนเทรด

-หมายเลขพอร์ตของเมนเทรด



ภาพประกอบ 4-5 การสร้างความสัมพันธ์กับระหว่างซ็อคเก็ตของคีย์บอร์ดเทรดและเมนเทรด

ในการสร้างความสัมพันธ์ระหว่างซ็อคเก็ตจะมีอยู่ด้วยกัน 3 ขั้นตอน ดังนี้คือ

-การเตรียมพร้อมของเมนเทรดในการสร้างความสัมพันธ์

-การเริ่มต้นการสร้างความสัมพันธ์ของคีย์บอร์ดเทรด

-เมนเทรดสร้างความสัมพันธ์ได้อย่างสมบูรณ์

4.4.1 การเตรียมพร้อมของเมนเทรดในการสร้างความสัมพันธ์

การใช้โปรโตคอล TCP จะต้องมีการเตรียมการก่อน โดยการเรียกใช้ฟังก์ชัน listen() ของวินซ็อค API

listen()

เมนเทรดใช้โปรโตคอล TCP ต้องเตรียมการที่จะรับการเชื่อมต่อจากคีย์บอร์ดเทรดที่ใช้โปรโตคอล TCP ด้วยเช่นกัน โดยการเรียกใช้ฟังก์ชัน listen() ซึ่งมีรูปแบบดังนี้

```
int PASCAL FAR listen (SOCKET s,      /* a named, unconnected socket */
                       int backlog);  /* pending connect queue length */
```

s : ซ็อคเก็ตแฮนเดิลที่ได้มีการให้ชื่อไว้แล้วโดยใช้ฟังก์ชัน bind() แต่ยังไม่ได้ทำการเชื่อมต่อ

backlog : ความยาวของคิวในระหว่างการเชื่อมต่อ

พารามิเตอร์ backlog เป็นจำนวนของการร้องขอการเชื่อมต่อที่จะเก็บไว้ในคิวในขณะที่เมนเทรดกำลังเชื่อมต่ออยู่แล้ว โดยปกติค่านี้จะมีค่าอยู่ระหว่าง 1-5

หลังจากที่ได้เรียกใช้ฟังก์ชัน listen() บนซ็อคเก็ตชนิด TCP แล้ว จะต้องทำการเตรียมพร้อมที่จะรับการเชื่อมต่อที่จะมีการร้องขอมาจากคีย์บอร์ดเทรดโดยการเรียกใช้ฟังก์ชัน accept()

4.4.2 การเริ่มต้นสร้างความสัมพันธ์ของคีย์บอร์ดเทรด

จะเริ่มต้นสร้างความสัมพันธ์โดยการเรียกใช้ฟังก์ชัน connect() ของวินซ็อค API ในการสร้างการเชื่อมต่อไปยังเมนเทรด หลังจากนั้นจะเรียกใช้ฟังก์ชัน send() ของวินซ็อค API ในการส่งอักขระที่ได้รับจากแป้นพิมพ์ไปยังเมนเทรด

connect()

คีย์บอร์ดเทรดที่ใช้โปรโตคอล TCP จะมีการสร้างความสัมพันธ์โดยสร้างการเชื่อมต่อด้วยการเรียกใช้ฟังก์ชัน connect() โดยมีรูปแบบดังนี้

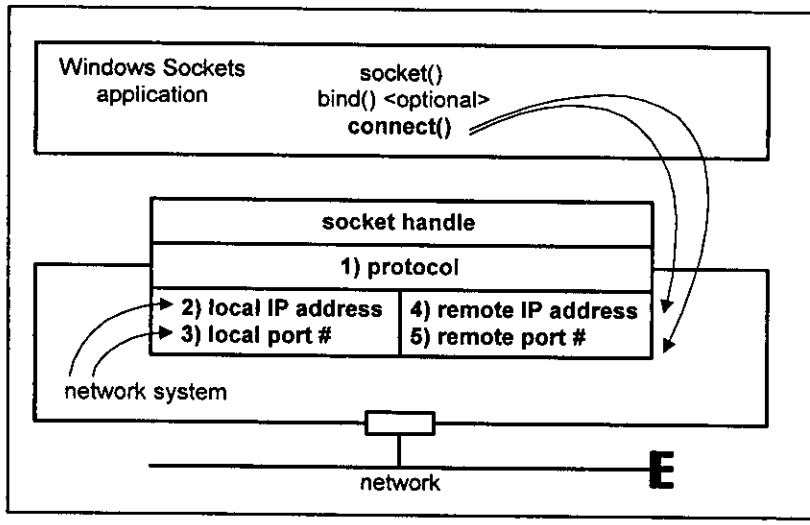
```
int PASCAL FAR connect (SOCKET s,  /* an unconnected socket */
   struct sockaddr FAR addr,       /* remote port and IP addr */
   int namelen);                   /* addr structure length */
```

s : ซ็อคเก็ตแฮนเดิล

addr : พอยท์เตอร์ที่ชี้ไปยังโครงสร้างของซ็อคเก็ตแอดเดรส สำหรับ TCP/IP จะหมายถึงโครงสร้าง sockaddr_in

namelen : ขนาดของโครงสร้างที่ถูกชี้โดย addr

ภาพประกอบ 4-6 จะแสดงรายละเอียดการทำงานของฟังก์ชันนี้



ภาพประกอบ 4-6 ฟังก์ชัน connect()

4.4.3 การสร้างความสัมพันธ์ที่สมบูรณ์ของเมนเทรด

เมนเทรดที่ใช้โปรโตคอล TCP จะสามารถทราบได้ว่าขณะนั้นมีการส่งอักขระมาจากคีย์บอร์ด เทรดโดยใช้ฟังก์ชัน accept() ของวินซ็อค API การสร้างความสัมพันธ์ให้เสร็จสมบูรณ์กรณีใช้ฟังก์ชัน accept() จะเกิดขึ้นเมื่อการใช้ฟังก์ชัน accept() เป็นผลสำเร็จ

accept()

ฟังก์ชัน accept() จะมีการส่งซ็อคเก็ตอันใหม่กลับมาเพื่อใช้ในการสร้างการเชื่อมต่อภายหลัง จากที่ใช้ฟังก์ชัน listen() เพื่อตรวจสอบการร้องขอ การเรียกใช้ฟังก์ชัน accept() มีรูปแบบดังนี้

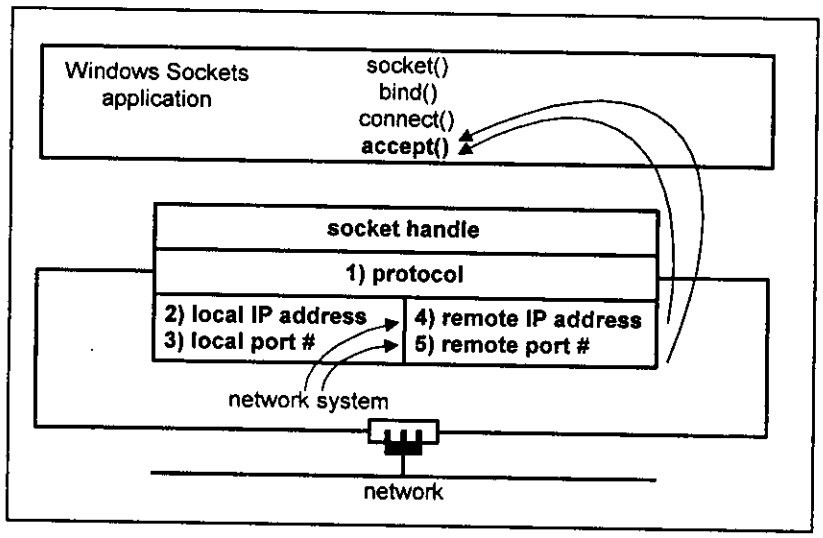
```
SOCKET PASCAL FAR accept (SOCKET s,        /* a listening socket */
   struct sockaddr FAR addr,               /* name of incoming socket */
   int FAR *addrlen);                      /* length of sockaddr */
```

s : ซ็อคเก็ตแฮนเดิล

addr : พอยท์เตอร์ที่ชี้ไปยังโครงสร้างของซ็อคเก็ตแอดเดรส สำหรับ TCP/IP จะหมายถึงโครงสร้าง sockaddr_in

addrlen : ขนาดของโครงสร้างที่ถูกชี้โดย addr

ฟังก์ชัน accept() จะมีพารามิเตอร์เหมือนกับฟังก์ชัน connect() และ bind() แต่จะต่างกันตรงที่ไม่ต้องทำการกำหนดค่าเริ่มต้นให้กับโครงสร้างของซ็อคเก็ตแอดเดรส โดยตรงส่วนนี้ฟังก์ชัน accept() จะทำให้เองโดยอัตโนมัติ แสดงได้ดังภาพประกอบ 4-7



ภาพประกอบ 4-7 ฟังก์ชัน accept()

4.5 การส่งและรับข้อมูลระหว่างซ็อคเก็ตของคีย์บอร์ดเทรดและเมนเทรด

เมื่อถึงขั้นตอนนี้การสร้างความสัมพันธ์ระหว่างเมนเทรดและคีย์บอร์ดเทรดได้เกิดขึ้นเป็นผลสำเร็จแล้ว ทั้งสองเทรดต่างมองเห็นซึ่งกันและกัน พร้อมที่จะทำการส่งข้อมูลและรับข้อมูลได้แล้ว

4.5.1 การส่งข้อมูลจากคีย์บอร์ดเทรดไปยังเมนเทรด

สามารถที่จะส่งข้อมูลได้โดยใช้ฟังก์ชัน send() ของวินซ็อค API ซึ่งมีรูปแบบดังนี้

send()

```
int PASCAL FAR send (SOCKET s,          /* associated socket */
    const char FAR buf,                 /* buffer with outgoing data */
    int len;                            /* bytes to send */
    int flags);                         /* option flags */
```

s : ซ็อคเก็ตแฮนเดิล

buf : พอยท์เตอร์ที่ชี้ไปยังบัฟเฟอร์ที่เก็บข้อมูลที่จะส่ง

len : ขนาดของข้อมูลที่จะส่ง

flags : พฤติกรรมการส่งข้อมูล

คีย์บอร์ดเทรดซึ่งใช้โปรโตคอล TCP จะต้องทำการเรียกใช้ฟังก์ชัน connect() เพื่อให้ทำการสร้างวงจรเสมือนก่อนที่จะเรียกใช้ฟังก์ชัน send() มิฉะนั้นแล้วฟังก์ชัน send() จะทำงานผิดพลาด

4.5.2 การรับข้อมูลของเมนเทรดจากคีย์บอร์ดเทรด

เมนเทรดรับข้อมูลโดยใช้ฟังก์ชัน recv() ของวินซ็อค API ซึ่งมีรูปแบบดังนี้

recv()

```
int PASCAL FAR recv (SOCKET s,          /* associated socket */
   char FAR buf,                        /* buffer with outgoing data */
   int len,                             /* bytes to send */
   int flags);                          /* option flags */
```

s : ซ็อคเก็ตแฮนเดิล

buf : พอยท์เตอร์ที่ชี้ไปยังบัฟเฟอร์ที่เก็บข้อมูลที่ได้รับ

len : ขนาดของข้อมูลที่ได้รับ

flags : พฤติกรรมการรับข้อมูล

ฟังก์ชัน send() และ recv() ต่างก็มีพารามิเตอร์ที่เหมือนกัน อีกทั้งต่างก็ใช้งานในขณะที่มีการเชื่อมต่อระหว่างซ็อคเก็ต แต่จะต่างกันตรงที่ฟังก์ชัน send() จะมีหน้าที่ในการส่งข้อมูล ส่วนฟังก์ชัน recv() จะมีหน้าที่ในการรับข้อมูล

4.6 การเลิกใช้ซ็อคเก็ต

การเลิกใช้ซ็อคเก็ตสามารถทำได้โดยเรียกใช้ฟังก์ชัน closesocket() ของวินซ็อค API

closesocket()

```
int PASCAL FAR closesocket (SOCKET s);        /* a valid socket */
```

s : ซ็อคเก็ตแฮนเดิล

4.7 กระบวนการทำงานของเทลเน็ตไคลเอนเมื่อได้รับอักขระจากเครื่องคอมพิวเตอร์ที่อยู่ระยะไกล

จากภาพประกอบ 3-4 ผู้วิจัยได้ทำการพัฒนากระบวนการทำงานของเทลเน็ตไคลเอนเมื่อได้รับอักขระจากเครื่องคอมพิวเตอร์ที่อยู่ระยะไกล โดยสามารถอธิบายแต่ละกระบวนการได้ดังนี้

เมื่อ s คือ ซ็อคเก็ต

และ c คือ อักขระที่ได้รับ

กระบวนการ A
ชื่อกระบวนการ do_echo (SOCKET s, int c);
คำอธิบาย จัดการในส่วนของทางเลือกการแสดงอักขระ

กระบวนการ B
ชื่อกระบวนการ do_noga (SOCKET s, int c);
คำอธิบาย จัดการในส่วนของทางเลือกการระงับคำสั่ง GA

กระบวนการ C
ชื่อกระบวนการ do_notsup (SOCKET s, int c);
คำอธิบาย เมื่อเทลเน็ตไคลเอนได้รับ WILL หรือ WONT ตามด้วยทางเลือกที่ไม่รู้จัก เทลเน็ตไคลเอนจะตอบกลับไปด้วย DONT ตามด้วยทางเลือกที่ไม่รู้จักนั้น

กระบวนการ D
ชื่อกระบวนการ do_txbinary (SOCKET s, int c);
คำอธิบาย จัดการในส่วนของทางเลือกการส่งข้อมูลแบบไบนารี

กระบวนการ E
ชื่อกระบวนการ no_op (SOCKET s, int c);
คำอธิบาย ไม่มีการจัดการใดๆ กับอักขระที่ได้รับ

กระบวนการ F
ชื่อกระบวนการ rec_opt (SOCKET s, int c);
คำอธิบาย เก็บอักขระที่ได้รับเพื่อไว้ดำเนินการในสถานะต่อไป

กระบวนการ G
ชื่อกระบวนการ sub_end (SOCKET s, int c);
คำอธิบาย สิ้นสุดการต่อรองทางเลือกย่อย

กระบวนการ H
ชื่อกระบวนการ sub_opt (SOCKET s, int c);
คำอธิบาย จัดการเกี่ยวกับการต่อรองทางเลือกย่อย

กระบวนการ J
ชื่อกระบวนการ tcdm (SOCKET s, int c);
คำอธิบาย จัดการสัญญาณการซิงค์ของเทลเน็ต

กระบวนการ K
ชื่อกระบวนการ ttputc (SOCKET s, int c);
คำอธิบาย แสดงอักขระออกทางจอภาพของผู้ใช้งาน

กระบวนการ L

ชื่อกระบวนการ will_notsup (SOCKET s, int c);

คำอธิบาย เมื่อเทลเน็ตไคลเอนได้รับ DO หรือ DONT ตามด้วยทางเลือกที่ไม่รู้จัก เทลเน็ตไคลเอนจะตอบกลับไปด้วย WONT ตามด้วยทางเลือกที่ไม่รู้จักนั้น

กระบวนการ M

ชื่อกระบวนการ will_termtype (SOCKET s, int c);

คำอธิบาย จัดการในส่วนของทางเลือกการกำหนดชนิดของเทอร์มินอล

กระบวนการ N

ชื่อกระบวนการ will_txtbinary (SOCKET s, int c);

คำอธิบาย จัดการในส่วนของทางเลือกการส่งข้อมูลแบบไบนารี โดยเมื่อได้รับ DO หรือ DON"T จะแจ้งกลับไปด้วย WILL หรือ WONT

4.8 กระบวนการทำงานของเทลเน็ตไคลเอนเมื่อได้รับอักขระจากแป้นพิมพ์

จากภาพประกอบ 3-6 ผู้วิจัยได้ทำการพัฒนากระบวนการทำงานของเทลเน็ตไคลเอนเมื่อได้รับอักขระจากแป้นพิมพ์ โดยสามารถอธิบายแต่ละกระบวนการได้ดังนี้

เมื่อ s คือ ซ็อคเก็ต

และ c คือ อักขระที่ได้รับ

กระบวนการ A

ชื่อกระบวนการ dcon (SOCKET s, int c);

คำอธิบาย ยกเลิกการเชื่อมต่อ

กระบวนการ B

ชื่อกระบวนการ no_op (SOCKET s, int c);

คำอธิบาย ไม่มีการจัดการใดๆ กับอักขระที่ได้รับ

กระบวนการ C
ชื่อกระบวนการ soputc (SOCKET s, int c);
คำอธิบาย ส่งอักขระไปยังเครื่องคอมพิวเตอร์ที่อยู่ระยะไกล

4.9 กระบวนการทำงานของเทลเน็ตไคลเอนเมื่อมีการต่อรองทางเลือกย่อย

จากภาพประกอบ 3-7 ผู้วิจัยได้ทำการพัฒนากระบวนการทำงานของเทลเน็ตไคลเอนเมื่อมีการต่อรองทางเลือกย่อย โดยสามารถอธิบายแต่ละกระบวนการได้ดังนี้

เมื่อ s คือ ซ็อคเก็ต
และ c คือ อักขระที่ได้รับ

กระบวนการ A
ชื่อกระบวนการ no_op (SOCKET s, int c);
คำอธิบาย ไม่มีการจัดการใดๆ กับอักขระที่ได้รับ

กระบวนการ B
ชื่อกระบวนการ subtermtype (SOCKET s, int c);
คำอธิบาย ส่งชนิดของเทอร์มินอลไปยังเครื่องคอมพิวเตอร์ที่อยู่ระยะไกล

สำหรับบทนี้ได้กล่าวถึงการพัฒนาระบบตามที่ได้ออกแบบไว้ในบทก่อนหน้านี้ ในการพัฒนาใช้ภาษา C ภายใต้สภาพแวดล้อมบนวินโดวส์ โดยอาศัยเครื่องมือช่วยคือวินซ็อค บทต่อไปจะเป็นบทสรุป ปัญหา และข้อเสนอแนะ