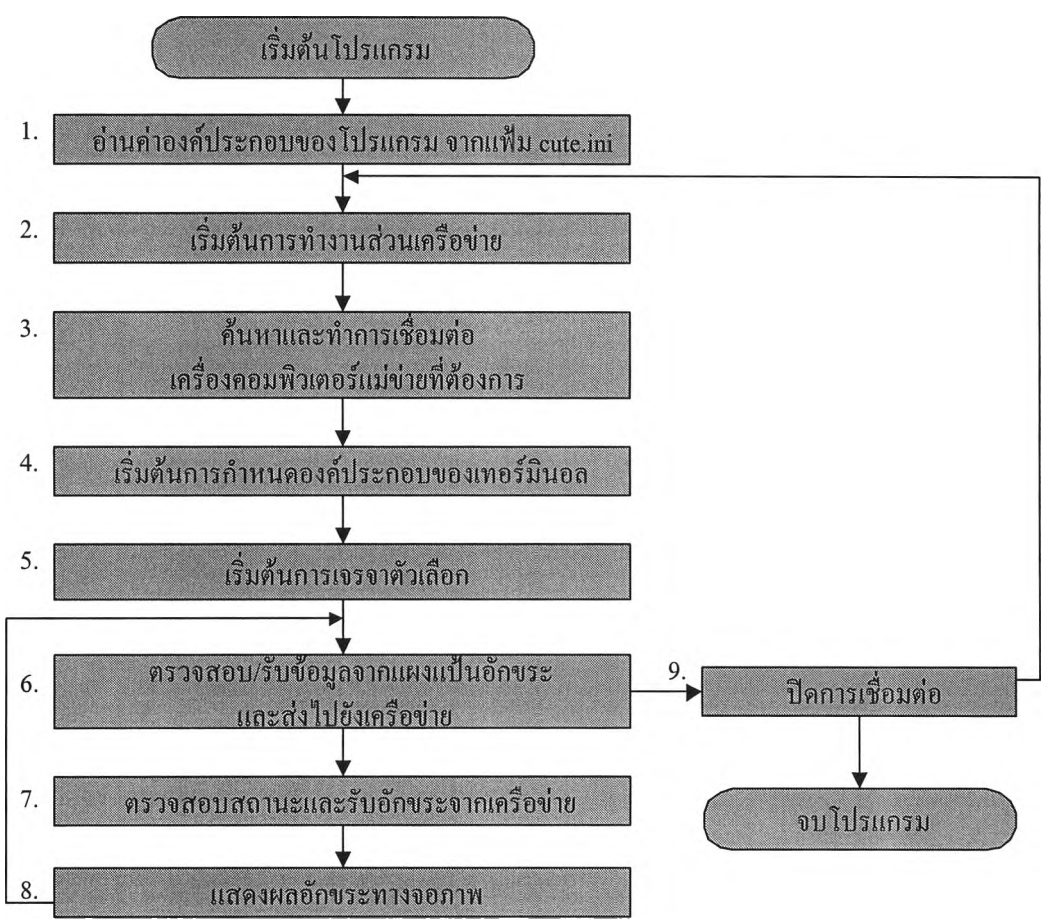
บทที่ 3
ส่วนประกอบการทำงานและการพัฒนาโปรแกรม

ในบทนี้จะกล่าวถึงส่วนประกอบการทำงานและการพัฒนาโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE เวอร์ชัน 3.00 ทั้งนี้จะอาศัยโค้ด บางส่วนของโปรแกรม CUTE เวอร์ชัน 2.10 มาใช้ประกอบการพัฒนา เพื่อให้สามารถใช้งานบนระบบปฏิบัติการวินโดวส์ที่มีแพลตฟอร์มแบบ Win32 และเป็นเวอร์ชันภาษาไทย

ขั้นตอนการทำงานของโปรแกรม CUTE 3.00

ขั้นตอนการทำงานของโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00 มีขั้นตอนการทำงานที่สามารถแสดงได้ดังรูปที่ 3.1 (อำพล, 2540)



รูปที่ 3.1 แสดงขั้นตอนการทำงานของโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00

จากรูปที่ 3.1 งานวิจัยนี้ได้ทำการพัฒนาโปรแกรม CUTE 3.00 โดยสามารถแบ่งการพัฒนาโปรแกรมออกเป็นส่วนๆ ที่สำคัญดังนี้

1. การพัฒนาส่วนเชื่อมต่อระบบเครือข่ายด้วยวินโดว์ซ็อกเก็ต
2. การพัฒนาปรับปรุงการเจรจาตัวเลือกและลำดับหลีก วีที100
3. การพัฒนาส่วนการแสดงผลโปรแกรม

การพัฒนาส่วนเชื่อมต่อระบบเครือข่ายด้วยวินโดว์ซ็อกเก็ต

การพัฒนาส่วนเชื่อมต่อระบบเครือข่าย จะใช้คลาสซีวินซ็อก และคลาสซีสตรีมซ็อกเก็ตในตัวควบคุมการทำงานของวินโดว์ซ็อกเก็ต ทั้งนี้เพื่อให้เหมาะสมกับการทำงานโดยใช้โพรโตคอลเทลเน็ต จึงต้องมีการปรับปรุงเพิ่มเติมที่สำคัญ 2 ประเด็น คือ

1. เนื่องจากการเชื่อมต่อกับเครื่องแม่ข่ายโดยใช้โพรโตคอลเทลเน็ตจะมีการรับส่งข้อมูลขนาด 8 บิต จึงทำการปรับปรุงชนิดของข้อมูลที่มีการรับส่งกันของคลาสซีสตรีมซ็อกเก็ตโดยข้อมูลที่รับส่งกันจะเป็นข้อมูลแบบ (unsigned char *st)

2. การเพิ่มการทำงานแบบ Out-Of-Band เนื่องจากคลาสซีสตรีมซ็อกเก็ต ไม่มีการเขียนโปรแกรมในส่วนของข้อมูลที่มาจากเครือข่ายที่เป็นแบบ Out-Of-Band (OOB) ซึ่งโปรแกรมเทลเน็ตเซิร์ฟเวอร์ที่ทำงานบนระบบปฏิบัติการยูนิกซ์บนเครื่องคอมพิวเตอร์แม่ข่ายบางชนิดมีการส่งข้อมูลแบบ Out-Of-Band มาด้วย ดังนั้นฟังก์ชัน WSAAsyncSelect( ) ที่อยู่ใน ฟังก์ชันสมาชิกของคลาสซีสตรีมซ็อกเก็ต ต้องเป็นดังต่อไปนี้

```
WSAAsyncSelect(m_s, m_hWnd, CWINSOCK_EVENT_NOTIFICATION, lEvent);
```

โดยที่ lEvent = FD_READ| FD_WRITE| FD_CONNECT |FD_CLOSE |FD_OOB;

รายละเอียดของคลาส ซีสตรีมซ็อกเก็ต ที่ทำการปรับปรุงแล้วจะมีลักษณะดังนี้

```
////////////////////////////////////////////////////////////////////////////
// CStreamSocket
//
class CStreamSocket : public CWnd
```

```
{
private:
    CWnd *m_pParentWnd;     // window to receive event notification
    UINT m_uMsg;            // message to send to m_pParentWnd on event
    SOCKET m_s;             // socket handle
    SOCKADDR_IN m_sinLocal;  // name bound to socket m_s
    SOCKADDR_IN m_sinRemote; // name on other side of m_s
    int m_nLastError;       // last WinSock error
    BOOL m_bServer;         // TRUE if socket m_s is bound to a name
    CPtrList m_listWrite;   // data waiting to be sent
    CPtrList m_listRead;    // data read
    BOOL ignoreTextUntilDM;

public:
    CStreamSocket(CWnd *pParentWnd, UINT uMsg);
    virtual ~CStreamSocket( );
    int CreateSocket(int nLocalPort);
    int CreateSocket(LPSTR pszLocalService = NULL);
    int DestroySocket( );
    int Connect(LPSTR pszRemoteName, int nRemotePort);
    int Connect(LPSTR pszRemoteName, LPSTR pszRemoteService);
    int Connect(LPSOCKADDR_IN psinRemote);
    int Accept(CStreamSocket *pStreamSocket);
    int Write(int nLen, unsigned char * pData);
    unsigned char * Read(LPINT pnLen);
    int GetPeerName(LPSOCKADDR_IN psinRemote);
    int LastError( ) { return m_nLastError; }

private:
    void InitVars(BOOL bInitLastError = TRUE);
```

```
    LONG HandleRead(WPARAM wParam, LPARAM lParam);
    LONG HandleWrite(WPARAM wParam, LPARAM lParam);
    LONG HandleOOBRead(WPARAM wParam, LPARAM lParam);
// message map functions
protected:
    //{{AFX_MSG(CStreamSocket)
    //}}AFX_MSG
        LONG OnWinSockEvent(WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP( )
};
```

ฟังก์ชันสมาชิก CStreamSocket::HandleOOBRead( )

เป็นฟังก์ชันสมาชิกของคลาสซีสตรีมซ็อกเก็ตที่ถูกประกาศแบบไพร์เวท มีหน้าที่อ่านข้อมูลจากเครือข่ายที่เป็นลักษณะ Out-Of-Band สำหรับโพรโตคอลเทลเน็ตเมื่อไคลเอ้นท์ได้รับข้อมูลที่เป็น Out-Of-Band จะตรวจสอบว่าไบต์แรกของข้อมูลมีค่าเท่ากับ 255 (0xFF) หรือไม่ ถ้าใช่ จะข้ามข้อมูลไบต์ต่อๆ ไป โดยไม่สนใจ จนกว่าจะพบไบต์ที่มีความหมายว่า Data Mark (0xF2) ข้อมูลถัดจากนี้จะถูกนำไปประมวลผลต่อไป โดยการทำงานจะมีลักษณะคล้ายกับการทำงานของฟังก์ชันสมาชิก CStreamSocket::HandleRead( ) จะต่างกันตรงที่ลักษณะการอ่านข้อมูลจากระบบเครือข่าย คือ

```
    nBytesRead = recv(m_s, (LPSTR)pData, READ_BUF_LEN,MSG_OOB);
```

นอกจากนั้นเป็นการเพิ่มส่วนของการตรวจสอบข้อมูลแบบ Out-Of-Band โดยรายละเอียดของฟังก์ชัน HandleOOBRead() มีดังต่อไปนี้

```
LONG CStreamSocket::HandleOOBRead(WPARAM wParam, LPARAM lParam)
{
    m_pParentWnd->PostMessage(m_uMsg, CWINSOCK_OUT_OF_BAND);
    while (1)       {
        unsigned char *pData = (unsigned char *)malloc(READ_BUF_LEN);
```

```
if ((pData==NULL))          {
    if (pData!=NULL)        // free anything that was allocated
        free(pData);
    pData = NULL;
    // tell the parent that a possible data read failed
    m_pParentWnd->PostMessage(m_uMsg, CWINSOCK_ERROR_READING);
    // fake the event to try again
    PostMessage(CWINSOCK_EVENT_NOTIFICATION, m_s,
                WSAMAKESELECTREPLY(FD_OOB, 0));
    break;
}
// receive data
int nBytesRead = recv(m_s, (LPSTR)pData, READ_BUF_LEN,MSG_OOB);

if (nBytesRead == SOCKET_ERROR)         {
    // free memory for incoming data
    free(pData);
    pData = NULL;

    // if the error is just that the read would block,
    // don't do anything; we'll get another FD_OOB soon
    m_nLastError = WSAGetLastError();

    if (m_nLastError == WSAEWOULDBLOCK)
        m_nLastError = 0;
    else
        // tell the parent that a data read failed
        m_pParentWnd->PostMessage(m_uMsg,CWINSOCK_ERROR_READING);

    break;
```

```
        }
        // make sure some data was read
        if (nBytesRead == 0)          {
            // free memory for incoming data
            free(pData);
            pData = NULL;
            break;
        }

        //Assume that the OOB data was in response to ^C, eventually we
        //will get the prompt back - we must allow it to be shown be
        //restoring character flow.

        if(*pData==0xff) {
            ignoreTextUntilDM = TRUE;
            break;
        }
    }
    return 0L;
}
```

ในการใช้งานวินซ็อกของโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00 จะทำใน ออบเจกต์ซีดิววิว (CCuteView) ซึ่งถ่ายทอดมาจากคลาสซีอีดิทวิว (CEditView) นั่นคือออบเจกต์ซีดิววิว จะทำหน้าที่เป็นวินโดว์แม่ของวินซ็อกวินโดว์ โดยมีตัวแปรและฟังก์ชันสมาชิกที่สำคัญสำหรับการเชื่อมต่อเครือข่าย ดังนี้ คือ

```
#include "cwinsock.h"   // Windows Sockets classes
#include "Variable.h"
class CCuteView : public CEditView
{
```

```
    // Attributes
    public:
    CWinSock * m_pWinSock;        // WinSock sub-system startup/.shutdown
    CStreamSocket * m_pStream;    // Stream socket to receive from
    char m_pszServer[100];        // host name or IP address of stream server
    char m_pszPort[10];
    // Operations
    public:
    void OnConnect( );
    void Negotiate(unsigned char *st,int cnt);
    void Start_Negotiate( );
    void HandleRead( );
    void StartSockObj( );
    void KillSockObj( );
    // Generated message map functions
    protected:
    //{{AFX_MSG(CCuteView)
        afx_msg LONG OnCuteStream(WPARAM wParam, LPARAM lParam);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP( )
};
```

ในส่วนของ เมสเสจแมพ (Message Map) ที่เกี่ยวกับการเชื่อมต่อเครือข่ายของออบเจกต์ CCuteView จะเป็นดังนี้

```
#include "CuteView.h"
////////////////////////////////////////////////////////////////////////////
// CCuteView

IMPLEMENT_DYNCREATE(CCuteView, CEditView)
```

```
BEGIN_MESSAGE_MAP(CCuteView, CEditView)
//{{AFX_MSG_MAP(CCuteView)
   ON_MESSAGE(WM_CUTE_STREAM, OnCuteStream)
//}}AFX_MSG_MAP
END_MESSAGE_MAP( )
```

โดยที่

WM_CUTE_STREAM เป็นเมสเสจที่จะใช้ติดต่อกันระหว่างออบเจกต์ CCuteView และ CStreamSocket ออบเจกต์

*m_pWinSock หมายถึง ตัวชี้ไปยังออบเจกต์แบบซีวินซ็อก

*m_pStream หมายถึง ตัวชี้ไปยังออบเจกต์แบบซีสตรีมซ็อกเก็ต

m_pszServer[100] หมายถึง สตริงที่ใช้เก็บชื่อของเครื่องคอมพิวเตอร์แม่ข่าย

m_pszPort[10] หมายถึง สตริงที่ใช้เก็บชื่อของบริการ เช่น เทลเน็ต

ฟังก์ชันสมาชิกที่สำคัญของคลาสซีคิววิว คือ

1. ฟังก์ชันสมาชิก CCuteView::OnConnect( )

เป็นฟังก์ชันที่ใช้ทำการเชื่อมต่อกับเครื่องคอมพิวเตอร์แม่ข่าย

2. ฟังก์ชันสมาชิก CCuteView::StartSockObj( )

เป็นฟังก์ชันที่ใช้สำหรับเริ่มต้นการทำงานของวินซ็อกโดยจะเริ่มต้นการกำหนดค่าของออบเจกต์แบบซีวินซ็อกและซีสตรีมซ็อกเก็ต

3.ฟังก์ชันสมาชิก CCuteView::KillSockObj( )

เป็นฟังก์ชันที่ทำหน้าที่สิ้นสุดการใช้งานวินซ็อก โดยจะทำลายวินโดว์ซ็อกเก็ตและลบหน่วยความจำที่เป็นของออบเจกต์ซีวินซ็อกและซีสตรีมซ็อกเก็ต

4. ฟังก์ชันสมาชิก CCuteView::OnCuteStream (WPARAM wParam, LPARAM lParm)

เป็นฟังก์ชันที่จะถูกเรียกใช้เมื่อมีเมสเสจจากวินโดว์ซีสตรีมซ็อกเก็ต โดยที่เมสเสจต่างๆ จะถูกบรรจุอยู่ในตัวแปร wParam ส่วนตัวแปร lParam จะแทนค่าองค์ประกอบอื่นๆ ที่จำเป็นสำหรับเมสเสจแต่ละเมสเสจ

เมสเสจที่ส่งมาจากวินโดว์ซีสตรีมซ็อกเก็ต ได้แก่

ก. CWINSOCK_DONE_WRITING ถูกส่งมาเมื่อสามารถส่งข้อมูลไปในระบบเครือข่ายได้แล้ว โดยที่ lParam จะเป็นตัวชี้ที่ชี้ไปยังข้อมูลที่สามารถส่งไปได้

ข. CWINSOCK_ERROR_WRITING ถูกส่งมาเมื่อเกิดความผิดพลาดในขณะที่กำลังส่งข้อมูลไปในระบบเครือข่าย โดยที่ lParam จะเป็นตัวชี้ที่ชี้ไปยังข้อมูลส่วนที่ไม่สามารถส่งไปได้

ค. CWINSOCK_DONE_READING ถูกส่งมาเพื่อบอกว่ามีข้อมูลถูกส่งมาจากระบบเครือข่าย

ง. CWINSOCK_OUT_OF_BAND ถูกส่งมาเพื่อบอกว่ามีข้อมูลแบบ Out-Of-Band มาถึง

จ. CWINSOCK_ERROR_READING ถูกส่งมาเพื่อบอกว่า ไม่สามารถอ่านข้อมูลจากระบบเครือข่ายได้

ฉ. CWINSOCK_YOU_ARE_CONNECTED ถูกส่งมาเพื่อบอกว่าสามารถทำการเชื่อมต่อกับเครื่องแม่ข่ายได้แล้ว

ช. CWINSOCK_LOST_CONNECTION ถูกส่งมาเพื่อบอกว่า ถูกตัดการเชื่อมต่อจากเครื่องแม่ข่าย

5. ฟังก์ชันสมาชิก CCuteView::Start_Negotiate( )

เป็นฟังก์ชันที่ถูกเรียกใช้เมื่อสามารถเชื่อมต่อกับเครื่องคอมพิวเตอร์แม่ข่ายได้แล้ว โดยจะทำการเจรจาตัวเลือกต่างๆ

6. ฟังก์ชันสมาชิก CCuteView::HandleRead( )

เป็นฟังก์ชันที่ถูกเรียกใช้เมื่อมีเมสเสจ CWINSOCK_DONE_READING มาจากวินโดว์ซีสตรีมซ็อกเก็ต โดยจะไปอ่านข้อมูลที่ได้รับมาเป็น (unsigned char *st) เพื่อส่งต่อให้ฟังก์ชันสมาชิก Negotiate ต่อไป

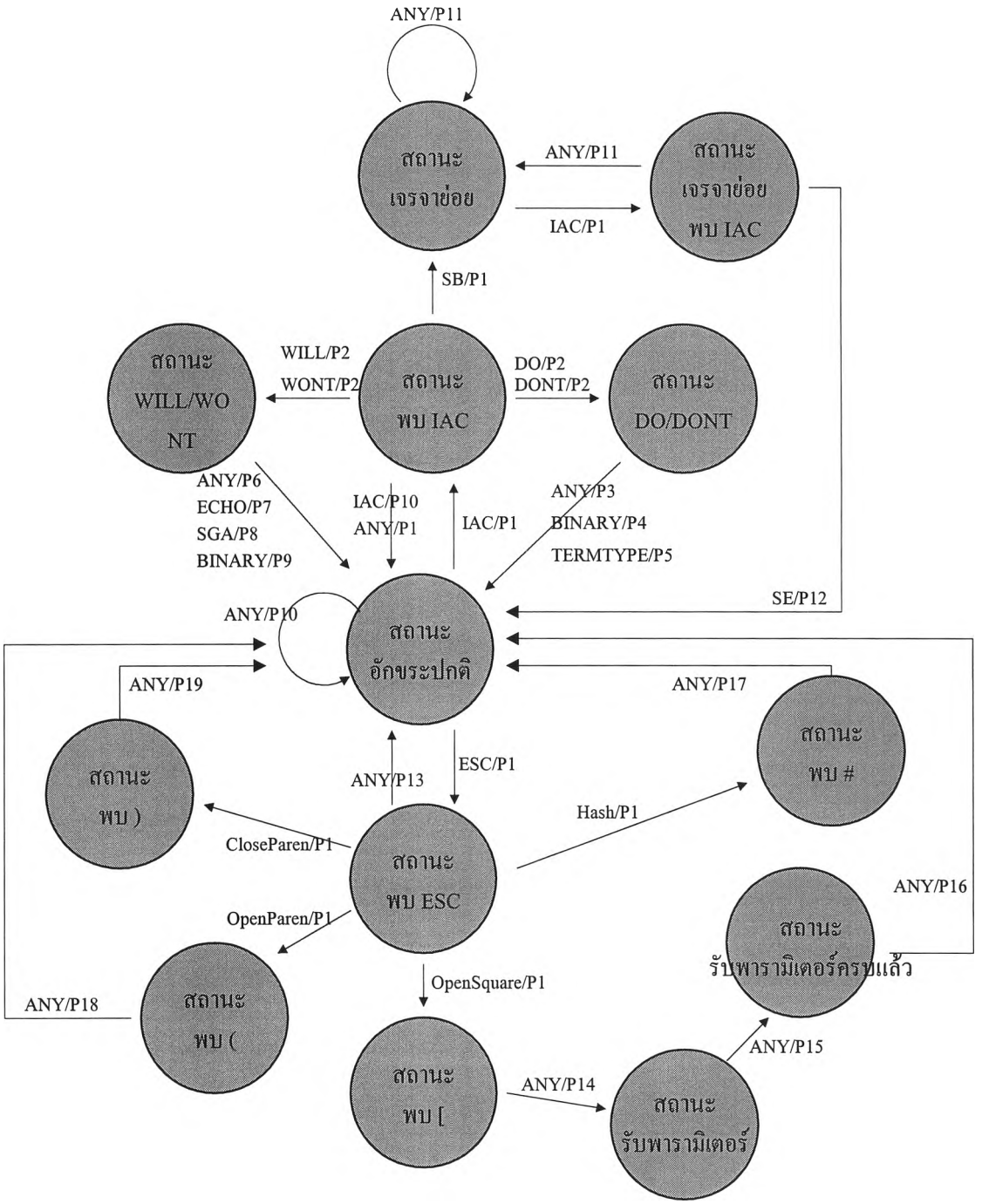
ในการคอมไพล์โปรแกรม CUTE 3.00 จำเป็นต้องทำการลิงค์กับแฟ้ม wsock32.lib ซึ่งทำได้โดยเซตที่ Build\Setting\link\ Object/ library modules: แล้วกรอก wsock32.lib ลงไป

การพัฒนาปรับปรุงการเจรจาตัวเลือกและลำดับหลีก วีที100

การพัฒนาปรับปรุงการเจรจาตัวเลือก (Option negotiation) และลำดับหลีก วีที100 (VT100 Escape sequences) ในการพัฒนาและปรับปรุงส่วนนี้ได้อาศัยโปรแกรมต้นฉบับ CUTE 2.10 ส่วนหนึ่ง โดยการทำงานหลักจะอยู่ในฟังก์ชันสมาชิก CCuteView::Negotiate (unsigned char *st, int len)

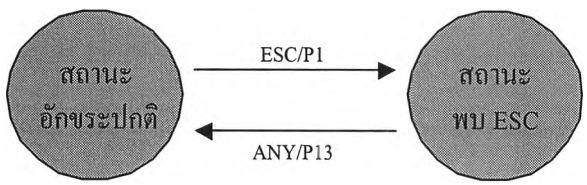
โดย *st คือ ตัวชี้ไปยังข้อมูลที่รับมาจากเครือข่ายโดยถูกส่งมาจากฟังก์ชันสมาชิก CCuteView :: HandleRead( )

ในฟังก์ชันสมาชิก CCuteView::Negotiate(unsigned char *st, int len) จะทำงานในส่วนของการเจรจาตัวเลือก และการจัดการกับลำดับหลีก วีที100 โดยการทำงานจะอาศัยหลักการของเครื่องสถานะจำกัด (Finite State Machine) ดังรูปที่ 3.2 (อำพล, 2540)



รูปที่ 3.2 แสดงการทำงานของเครื่องสถานะจำกัด สำหรับโปรแกรม CUTE 3.00

จากรูปที่ 3.2 เป็นแผนภาพที่แสดงเครื่องสถานะจำกัด สำหรับโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00 เพิ่มเติมส่วนของการจัดการกับลำดับหลีก วีที100 เข้าไป ตัวอย่างการทำงานแสดงดังรูปที่ 3.3



รูปที่ 3.3 แสดงตัวอย่างการทำงานของเครื่องสถานะจำกัด

จากรูปที่ 3.3 เมื่อโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00 ได้รับข้อมูลจากเครือข่าย โดยปัจจุบันอยู่ที่สถานะอักขระปกติ ถ้าได้รับอักขระ ESC (0x1B) จะเปลี่ยนสถานะไปที่สถานะพบ ESC โดยทำงานตามกระบวนการ P1 แล้วทำการวิเคราะห์ข้อมูลไบต์ถัดไป ถ้าอักขระใดๆ ที่ไม่ใช่ #, (,), [แสดงว่ามีพารามิเตอร์ที่เป็นอักขระต่อจาก ESC เพียงอักขระเดียว เช่น ESC D โดยจะทำงานดังนี้คือ จะเลื่อนเคอร์เซอร์ลงล่าง 1 บรรทัด และถ้าอยู่บรรทัดสุดท้ายต้องทำการเลื่อนหน้าจอด้วย

กระบวนการทำงานต่าง ๆ ของเครื่องสถานะจำกัด แสดงดัง ตารางที่ 3.1

กระบวนการทำงาน	คำอธิบาย
P1	เปลี่ยนสถานะเท่านั้น
P2	เปลี่ยนสถานะ และเก็บรหัสขอการรับบริการ ได้แก่ DO DONT WILL หรือ WONT ขึ้นอยู่กับข้อมูลที่ได้รับ
P3	ไม่ต้องตอบสนองเครื่องแม่ข่าย กรณีที่ร้องขอ หรือปฏิเสธการทำใดๆ
P4	ตอบสนองเครื่องแม่ข่ายโดยกรณีที่ร้องขอ หรือปฏิเสธ การทำ BINARY
P5	ตอบสนองเครื่องแม่ข่ายโดยกรณีที่ร้องขอ หรือ ปฏิเสธการทำ TERMTYPE
P6	ไม่ต้องตอบสนองเครื่องแม่ข่ายกรณีที่ ยินยอม หรือไม่ยินยอม การทำใด ๆ
P7	ตอบสนองเครื่องแม่ข่ายโดยกรณีที่ยินยอม หรือไม่ยินยอม การทำ ECHO
P8	ตอบสนองเครื่องแม่ข่ายโดยกรณีที่ยินยอม หรือ ไม่ยินยอม การทำ SGA

ตารางที่ 3.1 แสดงกระบวนการทำงานในเครื่องสถานะจำกัด สำหรับโปรแกรม CUTE 3.00

กระบวนการทำงาน	คำอธิบาย
P9	ตอบสนองเครื่องแม่ข่ายโดยกรณีที่ยินยอมหรือไม่ยินยอมการทำ BINARY
P10	แสดงอักขระที่ได้รับมาออกทางจอภาพ
P11	เปลี่ยนสถานะและเก็บข้อมูลที่ได้รับเพื่อทำการเจรจาตัวเลือกย่อย (subnegotiate)
P12	ทำการวิเคราะห์ข้อมูลที่เข้ามาในขั้นตอนการเจอรจาตัวเลือกย่อย เช่น ส่งชนิดของเทอร์มินอล ตัวอย่างเช่น วีที100 หรือ วีที200 ไปยังเครื่องแม่ข่าย
P13	ทำงานตามลำดับหลีก ESC x โดยที่ x คืออักขระใดๆ เช่น ESC D
P14	ทำการตรวจสอบว่าพบ ? หรือไม่ ถ้าพบจะเก็บรหัส ? โดยที่จะพบหรือไม่ก็ตามจะเข้าสู่สภาวะรับพารามิเตอร์ต่อไป
P15	ทำการรวบรวมพารามิเตอร์ทั้งหมดที่ต่อจาก ESC [หรือ ESC [?
P16	ทำงานตามลำดับหลีก ESC [ใด ๆ หรือ ESC [? ใด ๆ
P17	ทำงานตามลำดับหลีก ESC # ใด ๆ
P18	ทำงานตามลำดับหลีก ESC (ใด ๆ
P19	ทำงานตามลำดับหลีก ESC) ใด ๆ

ตารางที่ 3.1 แสดงกระบวนการทำงานในเครื่องสถานะจำกัด สำหรับโปรแกรม CUTE 3.00 (ต่อ)

จากการทำการวิจัยพบว่า โปรแกรม vi ที่ทำงานบนระบบปฏิบัติการ Digital UNIX จะส่งลำดับหลีกที่ไม่สามารถตีความหมายได้ เมื่อข้อมูลเป็นภาษาไทย โดยลำดับหลีกดังกล่าวไม่อยู่ในรูปแบบที่สามารถแก้ไขได้ และทำให้โปรแกรมทำงานผิดพลาด เพื่อไม่ให้ผู้ใช้โปรแกรมเกิดความสับสน จึงไม่ทำงานตามลำดับหลีกดังกล่าว และจะส่งข้อมูลจำนวน 3 ไบต์ ESC,1,G (0x1B, 0x31, 0x47) กลับไปให้เครื่องแม่ข่าย เพื่อเลื่อนเคอร์เซอร์ไปยังตำแหน่งแรกของข้อมูล

การพัฒนาส่วนการแสดงผลหน้าจอ

โปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00 ใช้คลาส ซีอีดิทวิว ในการแสดงผลข้อมูล ดังนั้นโปรแกรม CUTE 3.00 จึงเรียกใช้ฟังก์ชันสมาชิกต่างๆ ที่มีอยู่ในคลาสซีอีทวิว และ คลาสซีอีดิท ที่ปรากฏอยู่ในบทที่ 2

การทำงานของโปรแกรมเลียนแบบเทอร์มินอลภาษาไทย CUTE 3.00 จะคอยจัดการกับตัวแปรสมาชิก int CCuteView::cursorpos ให้มีค่าที่ถูกต้องในการแสดงผลข้อมูล

ในส่วนการแสดงผลข้อมูลโปรแกรม CUTE 3.00 จะไม่แสดงผลข้อมูลทีละตัวอักษรแต่จะแสดงเมื่อจำเป็นในกรณีที่มีข้อมูลมาครั้งละมาก ๆ จะมีการเก็บข้อมูลให้ครบบรรทัดหรือพบรหัสขึ้นบรรทัดใหม่จึงค่อยแสดงผลทำให้โปรแกรมแสดงผลได้เร็ว

เนื่องจากคลาสซีอีดิทวิว มีข้อจำกัดคือจะแสดงผลด้วยอักขระรูปแบบเดียวกันหมด ไม่สามารถมีรูปแบบพิเศษที่แตกต่างออกไปในบางตำแหน่งได้ ทำให้โปรแกรม CUTE มีข้อจำกัดดังต่อไปนี้

1. ไม่สามารถทำอักขระให้หนา เอียง หรือ ปรกติ ในบางตำแหน่งได้
2. ไม่สามารถทำให้อักขระมีสีที่แตกต่างออกไป และ ทำ inverse video ในบางตำแหน่งได้