報告題名：

Voice Messenger

在 ARM 嵌入式系統平台之實作

Implementation of Voice Messenger on ARM Embedded System

作者：劉正強、陳衍詳、蔣雯玲、柯以諾

系級：資訊四乙

學號：D9120325、D9147530、D9147721、D9147778

開課老師：劉嘉政

課程名稱：專題研究

開課系所：資訊工程學系

開課學年：94 學年度 第 1 學期

摘 要

本專題研究在 ARM 嵌入式系統平台實作語音交談的網路通訊軟體。學習在 ARM 平台建置 Linux 及使用目標板的 I/O 裝置，並且用 Microwindows 來開發圖形使用者介面(GUI)，以及規劃 Voice Messenger 的 Client-Server 架構和 P2P 通訊協定，透過 socket 使用 TCP/UDP 來傳輸資料和聲音。最後將 Voice Messenger、Root File System 和 Linux Kernel 載到目標板的 Flash Memory，完成一個完整 WiFi 的 VoIP 系統。

關鍵詞：ARM、嵌入式系統、Linux、Linux Kernel、Root File System、Microwindows、語音通訊、Instant Messenger

致 謝

- 感謝本專題指導教授劉嘉政老師給予相當重要的方向與目標
- 感謝系上提供 ARM 實驗板給予實作
- 感謝劉嘉政老師提供 Wireless VPN Router 方便測試實作
- 感謝陳青文老師提供 Pocket Wireless Access Point 方便測試實作
- 感謝李丕耀學長給予實作 ARM 實驗板上的協助與經驗分享
- 感謝林秉毅學長給予專題與書面排版上的經驗分享
- 感謝岳庭誼同學後期提供筆記型電腦方便組員討論以及修改程式

目 錄

表目錄

圖目錄

●前言

在詳述專題內容之前，先了解作此專題的動機、目的與每個組員分配到的工作，對整個專題的架構有個初步的認識。

1.1 動機

其實一開始的動機主要是以嵌入式 Linux 為主，想了解一個作業系統的運作和嵌入式系統，因為嵌入式系統目前算很熱門的話題，無論是任何電子產品都要求小還要更小，在未來也是一種趨勢，但需要有一個應用來實作，所以透過老師的指引，讓我們對於利用網路傳遞聲音很感興趣，因此才定出 Voice Messenger 這個主題。

1.2 目的

實作此專題的主要目的在於深入了解 Linux 作業系統運作與嵌入式系統的應用，學習一個完整系統架構與運作方式，從實作 Voice Messenger 中得到網路傳遞聲音的觀念與應用，而利用 microwindows 實作圖形化使用者介面學習如何設計介面更貼近使用者的心。

1.3 工作分配

- 系統整合與移植、書面報告撰寫—柯以諾
- 系統建置、書面報告撰寫—陳衍詳
- 實作 Voice Messenger 主要架構、書面報告撰寫—劉正強
- 使用者介面設計、書面報告撰寫—蔣雯玲

●Instant Messenger 探討

本章將經由對 Instant Messenger (IM)的探討，實際作為專題的參考對象。首先將介紹 IM 的發展與走向，讓讀者瞭解 IM 對網路通訊與使用者的影響。然後針對支援語音交談的 IM 與 VoIP (Voice over Internet Protocol)技術的應用多加介紹，比較各個知名 IM 的語音功能，並說明網路語音交談對傳統通訊業者所帶來的衝擊。

2.1 網路通訊的發展與趨勢

一、網路通訊的發展

自從 90 年代 Internet 開始流行，短短十年來，人與人之間的通訊方式，出現了新的形態，以往必須透過傳統郵件和電話才能通訊的時代，逐漸轉變成透過 Internet 來實現。同時，網路的虛擬社會和以文字為基礎的交流，改變了許多人的溝通習慣，對於社交與人際關係造成不小的影響，特別是十幾二十歲的年輕世代。

無論是即時或非即時通訊，對單人或多人，透過 Internet 來與他人溝通，已成為現代人習以為常的方式。在非即時通訊方面，E-mail、BBS、NEWS 始終是最常見的幾種型式；然而即時通訊(Instant Messaging)的發展，則對多方面產生更大的衝擊。

二、IM 的發展

早期在 Unix 環境下的 talk，可說是最早的即時通訊軟體之一，使用者在終端機上互相連線，然後即時地使用文字交談，是當時工程師們的最愛。隨著 Internet 與 GUI 作業系統的蓬勃發展，即時通訊也出現了更新的形態，例如以 web 為基礎的聊天室(使用 CGI 等的互動技術)，以及至今仍然相當流行的

IRC(Internet Relay Chat)，與 talk 不同的是，這是以多人交談為主的聊天室形態。1996 年 11 月，ICQ 的誕生成為往後即時通訊軟體 IM 的基本雛形。目前主流的 IM 有：MSN Messenger、Yahoo! Messenger、QQ 等等，以語音通訊為主要走向的有 Skype，以及最近新推出的 Google Talk。

三、IM 的特色

IM 具備私有性與個人化的特性，使用者擁有個人帳號和個人資訊，可以將好友加入清單，並且即時顯示好友的上線狀態，經由點選好友名單選擇通訊的對象。更多附屬功能，例如表情圖案、個人圖片、交換檔案、連線遊戲等等，而對即時語音、影音交談的支援，更是擴展 IM 的應用。即時通訊、多功能性與親善便利的使用者介面，吸引了廣大的網路使用族群，甚至已經融入生活之中，使 IM 成為新世代的通訊工具。

四、IM 的走向

近年來的 IM，開始具備了語音交談的功能，例如 MSN Messenger、Yahoo! Messenger、QQ 等。另外也有本身就是以語音交談為發展目標，但也擁有類似一般 IM 的介面和功能，例如 Skype 和最近新推出 Google Talk，其實也算 IM 的一類。IM 業者同時提供許多週邊服務與社群組織來吸引更多使用者參與。

最近一兩年，市面上推出了支援 IM 的手機機種，IM 與電信業者的合作，讓人們在手機上也可以使用 IM 與使用電腦的朋友傳送訊息，更是加強了 IM 的魅力。雖然 IM 的主要功能仍是免費使用，不過可以發現 IM 的多樣性背後潛藏著無限商機，而業者也不排除針對特定服務收取費用。

五、取代傳統

拜網路發達所賜，這些新形態的通訊方式，提供人們更多新的選擇，大幅降低時間、空間和成本上的限制，同時也對傳統通訊業者造成極大的衝擊。例如 E-mail 的快速與極低成本，逐漸取代傳統郵件的地位；VoIP 技術的發展，IM 的語音交談和網路電話，甚至線上影音會議，皆可透過網路傳輸，因而省下大筆的電信費用，對於經常與外國公司聯繫的企業用戶有很大的益處。網路通訊的蓬勃發展，趨使電信業者紛紛加入與網路業者合作的行列，使得 PC-to-Phone 之間也可以互相通訊，未來網路通訊與電信通訊間的界線，將越來越模糊。

透過網路傳輸即時語音和影音，技術上有許多問題值得探討，例如音質、畫質、編碼與壓縮、效率、網路延遲與不確定性、安全性、成本等等。下節將介紹與探討主流 IM 的語音功能，並以此作為我們專題的參考對象。

2.2 各大Instant Messenger比較

支援 VoIP 的 IM，隨著 IM 支援 VoIP 的技術，使網路電話越來越流行。網路電話不使用傳統電話線，取而代之的是利用網路來傳遞聲音，在大部分的情況之下，我們只要將電話插在由網路電話公司提供具有網路能力的裝置上，再按下對方的電話號碼，便可以進行通話；但現在我們最常使用的是 softphone,，電腦就成為使用的電話，只要用一個可通話的裝置如麥克風，接上電腦 USB 的 port，就可以透過網路來通訊。

- 透過 VoIP 主要的特點：
  1. **省錢**—使用網路電話最大的好處莫過於便宜、省錢，例如：只要有網路的地方就可以進行通訊，如果要打遠洋電話，價錢遠比傳統電話價錢還來的便宜。
  2. **不受地區限制**—因為網路電話是透過封包傳輸，不用受到地區的限制，而且可以取得想要的區域號碼。
  3. **與傳統電話溝通**—VoIP 服務提供了由網路和 PSTN(Public Switched Telephone Network)之間的閘道，網路電話便可以和傳統電話做通話的動作。

IM 不僅能傳遞文字、圖片、檔案，甚至影音串流，如線上會議、線上社群、搜尋、買賣等等，功能可以說是越來越強大；因而網路電話盛行原因也是因為 IM 的支援，許多的 IM 軟體現在都已經慢慢的提供了網路電話的功能，只要安裝相同軟體，透過 PC-to-PC 就可以進行網路電話的通訊了。支援 VoIP 的 IM 比較著名的有 MSN、Yahoo Messenger with voice、ICQ、Skype、及 GoogleTalk 等，下頁表 1 將比較各 IM 的異同。

表 1 各大 IM 比較表

	MSN Messenger	Yahoo Messenger	ICQ	Skype	Google Talk
平台支援	Windows98 以上	Windows98 以上	Windows98 以上	Linux、Windows2000 以上、Pocket PCs、Mac	Windows 2000 以上
網路電話功能	有	有	有	有	有
資料加密	無	無	無	有	無
Video chat	有	有	有	無	無
Voicemail	無	有	無	有(需付費)	無
會議電話	無	有	無	有(最高五人)	無
傳統電話互通	無	有(需付費)	有(需付費)	有(需付費)	無
照片分享	有	有	無	無	無
線上遊戲	有	有	有	無	無
Web Search	有	有	無	無	無
Blogs	有	有	無	無	無
Phone Numbers	無	無	無	有(需付費)	無
E-Mail	有	有	無	無	有
介面廣告	有	有	有	無	無
客製化背景	有	有	有	無	無
客製化鈴聲	無	有	無	有	無
月曆	有	有	無	無	無
多國語言支援	有	有	有	有	無

Yahoo 即時通 7.0 增加網路電話的功能，可以發現的是自從 Skype 成功的發展語音功能之後，越來越被重視的趨勢，因為各家 IM 漸漸加強語音傳輸的功能，Skype 也加入了傳統 IM text messaging 的功能，除了語音功能外也積極增加其他額外的功能。

從網路電話的功能來看，主要可以分成兩個方向，一個是從文字即時通訊功能導向，一個是從語音通訊功能導向，除了 Skype 以語音功能導向為主之外，其他都是以文字通訊為導向。可以發現 Skype 支援的平台最多，在聲音的傳輸過程中，具有資料加密的功能，對於隱私和安全性具有重大的幫助，SkyOut 的功能，更讓傳統的電話與網路電話可以結合，吸引了不少的使用者；另外，MSN 和 Yahoo Messenger 原本是主要對打的兩大通訊軟體，性質相似，可以發現許多功能極為相似，在日前宣布日後將能互相通訊，讓兩邊的 IM 使用者相互通訊；而 ICQ 則是最早的通訊軟體，與其他 IM 亦相似，但是功能並不如 MSN 和 Yahoo Messenger 多，因此使用者並不如前兩者來的多，Google Talk 則是由 Google 所開發，目前只支援英文版本，也只允許擁有 Google 的 GMail 使用者才可使用。

值得一提的是 IM 所提供的介面，都是容易安裝、容易使用，讓使用者能夠容易的上手，即 Yahoo 最新加入的 Yahoo360，就如 MSN space 一樣，加強了網路社群的功能，再加上 Yahoo 最新的 Messenger 7.0 是針對網路電話有所支援，加強了語音通訊的品質，更加顯示網路電話在未來的趨勢。

從 IM 所衍生出的 VoIP 技術，網路電話對於未來在通訊上的一大改變，在短期還並不會對於傳統電話有太大影響，不過，值得注意的是，網路電話將從純粹軟體上的在 PC-to-PC，因為頻寬的大幅提升，發展出 PC-to-Phone、Phone-to-PC 也就是由傳統 IP-to-IP、IP-to-PSTN、PSTN-to-IP、PSTN-to-PSTN 等不同的形式，傳統 PSTN 為傳輸媒介的獨占時代也將被打破。

對於我們專題有相關的是網路電話未來的趨勢與要求，主要有幾個重要的議題可以探討：

(1) **QoS(Quality of Service)**

和傳統 PSTN 相比較之下，網路電話必須達到的是服務的品質，因為網路封包傳輸所造成語音傳遞的延遲以及封包的遺失率，究竟要取在哪個可接受的成本下，達到符合效率的品質，將是個重要的議題。

(2) **High Available (H.A.)**

比較於發展已久傳統 PSTN 的成熟度、穩定度、可用性、可管理性、甚至可擴充性等，是網路電話必須去加強的，而透過技術上如負載平衡、路由備份來達到所要求，高可用性是必須達到的目標。

(3) **開放性及相容性**

各家 IM 所開發制定的協定都不盡相同，雖然目前 RFC 的 H.323、SIP、MGCP 三種協定可供參考，但是在開發上如何在開放式的架構之下達到互通與相容，而開發上產品的相容性及網路電話與傳統電話互通的困難度，也是必須解決的事項。

(4) **可管理性與安全性問題**

隨著資訊安全的重視，在管理與安全性上，因為開放式的架構使得網路電話未來開發上的勢必會遇到的障礙。

(5) **多媒體應用**

與傳統電話相比，網路電話除了價格外，最重要的應該是在於多媒體的應用之上，結合多媒體服務如電子商務、多媒體視訊會議等等，將是 VoIP 未來大為蓬勃發展的主要原因。

2.3 導入專題目標

前幾節討論的 Instant Messenger 幾乎都是以 PC 上的輔助軟體為主，大大節省了電話費，但難道沒有一個產品可以成為電話的替代品呢？一般沒有個人電腦或 NoteBook 的民眾都用不到 IM，所以消費市場不夠大，而最近智邦科技將研發出把 Skype 的技術內建到單獨的產品內，也就是嵌入式系統的觀念，讓全球的 Skype 用戶以及尚未使用過 Skype 的家庭與企業用戶，能夠不必開電腦，即可利用 Skype 點對點與網路電話的功能，撥通任何一個電話號碼，擁有免費或便宜的長途與國際電話。有了無線網路，就可把 Instant Messenger 帶入嵌入式系統的世界，本專題的主旨就是由此發展出來的。

圖 1 智邦科技的無線網路 Skype 話機外型

軟體方面由於嵌入式系統的可攜帶性，加上文字傳遞攜帶上不好使用，而考慮安全性上影像不太適合攜帶，所以專題的 Voice Messenger 主要以傳遞聲音為主。在 Instant Messenger 中以語音通訊為主要走向的 Skype 改變第一代 P2P (Peer-to-Peer)的架構，加入內建超節點（SuperNode；SN）功能，使對外頻寬高的用戶提升為 SN，讓用戶透過 SN 有效提升 Skype 的語音傳送效能。這樣的設計對於 Skype 用戶而言相當的方便，但是對於具有較大頻寬的企業用戶來說，卻

容易因為 SN 佔用網路頻寬，影響到公司日常業務。而語音傳送部份本專題是用 8kHz 的取樣頻率藉由 P2P 方式建立 UDP 連線，以 Client-Server 架構利用 server 來檢查線上使用者名單，並未用到任何市面上的 VoIP 協定。

●開發環境與使用平台

由於網路與多媒體的進步，加上嵌入式產品的推陳出新，產品的品質成為考量的標準導致開發系統的選擇。在本章當中將介紹關於嵌入式 Linux 的介紹及業界實際應用，也談到開發時所採用的嵌入式 Linux，使用的系統以 Linux 來當作開發平台，並對開發嵌入式系統在環境及工具上作基本介紹。

為什麼選擇 Linux?

3.1.1 Linux 優點

近年來由於 e 化的來臨，生活中各種電子產品如行動電話、MP3 Player 以及醫療設備等等，都與嵌入式作業系統的領域相關，然而嵌入式系統雖然是針對「特定用途」，不過由於現在的產品功能趨於複雜化，例如行動電話兼具 PDA 的功能，因此，需要 OS 來作資源管理和協調，而使用者介面，讓我們在使用上更具人性化。讓開發者願意從傳統嵌入式作業系統轉變成使用 Linux 來開發嵌入式系統，除了廠商的支援，最主要是因為 Linux 包含許多優秀能力，如下：

(1) **程式碼的品質與可靠度(Quality & Reliability)**

品質與可靠度對於程式碼是主觀性的考量，其實對於優秀的程式碼無法做出最精確的定義，不過一般設計者有以下的看法。

- ***品質：***

(a) 模組與結構

不同功能放在不同的模組，例如檔案的配置，而每個模組複雜的功

能會被分成適當的獨立函式。

(b) 容易修改

了解程式碼內部的人應該能輕易進行修改。

(c) 可擴充

可輕易加入新的功能。

(d) 可設定

可以選擇設定程式碼的那些功能應該出現在應用中。

- ***可靠度：***

(a) 可預測

執行時程式必須按照所定義的方式進行，不應該背離定義。

(b) 從錯誤中復原能力

程式發生錯誤應該有相對應的程式或機制能採取一些補救措施，從問題中恢復，而且必須是有適當權限，如系統管理者。

(c) 長期執行能力

必須有長期執行並且有保護自己的能力。

Linux kernel 和系統大部分計畫的程式碼都符合上面對於品質和可靠度的描述，由於開放原始碼的開發模型，開發後可以有效率改善問題、修正問題，找出問題速度有多快，解決速度就有多快，加上程式運行多年也很少出問題，很少有作業系統可以達到這種品質和水準。

(2) **成本(Cost)**

傳統的嵌入式系統的建立，有三種軟體元件成本：基本開發套件、額外工具、執行時的權利金；利用 Linux 來開發嵌入式系統，如開發工具、OS 元件(kernel)根據自由軟體和開放原始碼的理念，可以自由取得甚至

免費，所以使用 Linux 來開發是一項重要的考量因素。

(3) **程式碼的可用性(Usability)**

任何人都可以隨意的取用 Linux 原始碼和所有工具，但是也必須遵守 GPL 的規範。

(4) **硬體支援(Hardware Support)**

Linux 支援各種不同類型的硬體平台與裝置，也因為是由高階語言所撰寫而成，所以對於系統有很高的移植性(portable)，如 ARM、PowerPC、Motorola 的機器都有移植的例子。

(5) **通訊協定與軟體標準**

由於網路的發達，各種網路協定和軟體標準，Linux 都有支援，例如結合既有的 Windows 網路，透過 Samba 來服務各個客戶端。還有許多在 UNIX 上執行的商業應用程式，都會移植到 Linux 上，所以這與一般傳統嵌入式作業系統受限制有所差別。

(6) **可用工具(Utility)**

許多的工具程式在網路上可以供人下載使用，例如 GNU 的 Free Software。

(7) **社群支援**

受到社群支援可能是 Linux 最大的優點，這就是自由軟體和開放原始碼的精神所在，例如網路郵件論壇，當需要某種應用程式或遇到某些問題的時候，有可能其他人在這之前也遇過類似的狀況，很容易就可以找到解決方法。

3.1.2 嵌入式 Linux 定義

Linux、Embedded Linux 和 Real-Time Linux 等術語用法常有不同，究竟何謂嵌入式 Linux？一般常常會誤解其真正意義。實際上，並沒有發行過嵌入式版本的核心，嵌入式 Linux 只是代表它是一個基於 Linux 核心的作業系統，並不表示使用

了任何特定的函式庫或工具。一個嵌入式 Linux 可能包括開發架構、原始碼瀏覽器(source browser)、交叉編譯器(cross compiler)、除錯器(debugger)、bootloader、或是專案管理軟體(project manager)，以及修改過的應用程式，通常會利用開發平台(Host System)和目標板(Target Board)的方式來開發，所發展的開發架構就稱為開發發行套件(Development Distribution)。

3.1.3 嵌入式 Linux 實際應用

嵌入式 Linux 的實際例子，許多嵌入式作業系統以漸漸採用嵌入式 Linux，類型主要可以分成大型、中型、小型，參考表 2。

表 2 嵌入式 Linux 的實際應用

名稱	類型	規模	時限	網路能力	使用者互動
加速器控制	工業控制程序	中型	嚴格	有	低
電腦輔助訓練系統	航空	大型	嚴格	無	高
Ericsson "blip"	網路	小型	寬鬆	有	非常低
SCADA 協定轉換器	工業控制程序	中型	嚴格	無	非常低
Sharp Zaurus	消費性電子產品	中型	寬鬆	有	非常高
Linux 伺服器	網路	中型	寬鬆	有	高

開發環境

3.2.1 平台選擇

一般開發平台可以分成如 Linux 工作站、Unix 工作站、Windows 工作站。大部分開發嵌入式 Linux 的開發平台(Host)都是以 Linux 最常見，但是相較於一般常用的 Windows 平台，許多指令需要花些心思去熟悉，最好的方式是成為主要的平台，由於在開發嵌入式系統所用的 Cross Compiling，也是應用在 Linux 上，所以，本專題的使用平台便是 Red-Hat Linux。

此外，Unix 的工作平台，也可以用來當成開發平台，如 Solaris，適用於電信開發方案，Linux 與 Unix 很類似，對於 Linux 可以適用的通常也適用於 Unix，例如開發工具鏈、GNU 的工具等。

Windows 是一般人的工作平台，使用 Windows 為主要平台的人來開發嵌入式系統，主要遇到的問題是 toolchain 的使用，藉由模擬的軟體，像是 VMware 或 Cygwin 等軟體，可以模擬 Linux 的作業平台，雖然工作平台是 Windows，仍然可以在虛擬環境中執行 Linux，在實驗時也成為我們主要的工具之一，也可同時測在使用 NAT 網路環境的使用者在連線時所遇到的問題，另外的問題是我們在實驗時主要是在筆記型電腦上，並不支援 RS-232 的序列埠，但是可以利用 USB serial dongle(USB 序列轉換器)來解決這個問題。

3.2.2 目標版(target board)介紹

本專題所使用的目標板是 HyBus 的 X-Hyper255B-TKUIII，處理器是使用 ARM 架構的 Intel PXA255。此目標板整合了許多週邊設備，功能相當齊全，讓我們有很完整的實驗平台。在這次專題使用到的裝置如下：

- Processor Intel PXA255 (400MHz)
- Flash Memory (32Mbyte)
- SDRAM (64Mbyte)
- TFT LCD 6.4'' (640*480)
- Touch Screen
- Audio
- Ethernet / PCMCIA slot (for wireless network)

(實驗板的詳細規格列於附錄 A)

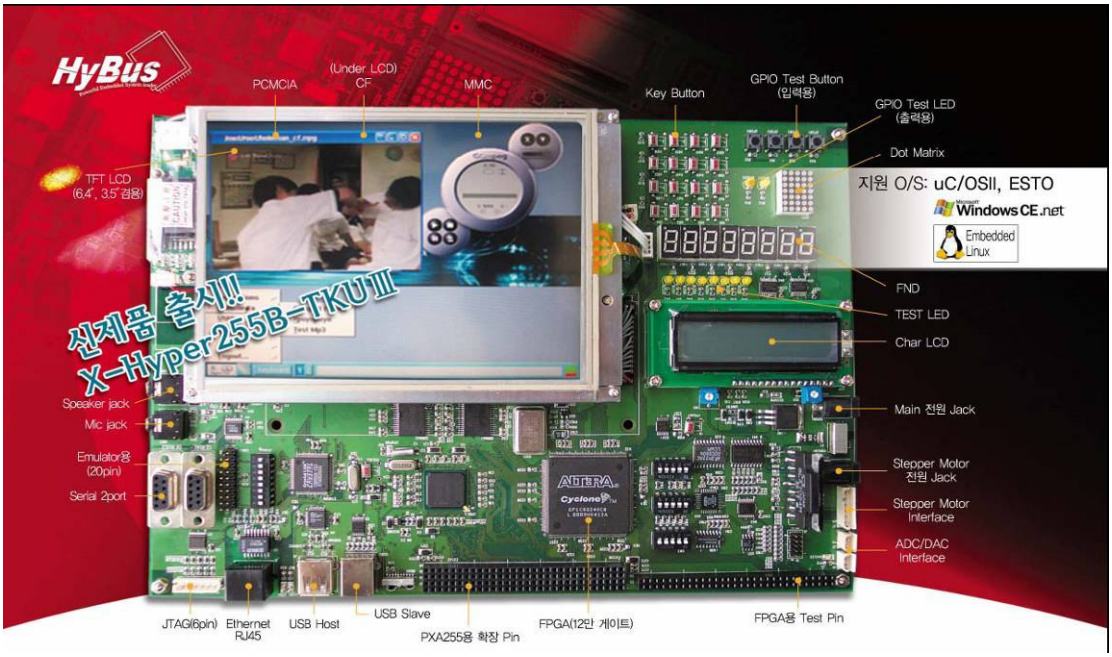

圖 2 Target Board 全圖

我們會將 Bootloader、Linux Kernel、Root File System 放置在 Flash Memory 中的特定位址，讓系統能順利啟動，有關 Memory Map 請見圖 3。GUI 將顯示在目標板的 TFT LCD 上，使用者可經由附在 LCD 上的 Touch Screen，選擇、使用系統的功能。在 Audio 方面，交由晶片的 ADC 與 DAC 轉換聲音的格式，經由 Speaker(喇叭)和 Mic(麥克風)兩個插孔，與外接設備進行聲音的輸出、輸入。網路連線則採用有線網路和無線網路兩種方式，使用者可以自行選擇。有線網路使用內建於目標板的 Ethernet 晶片和 port；無線網路必須另外準備 PCMCIA 無線網路卡，並插在目標板的 PCMCIA slot 來使用。

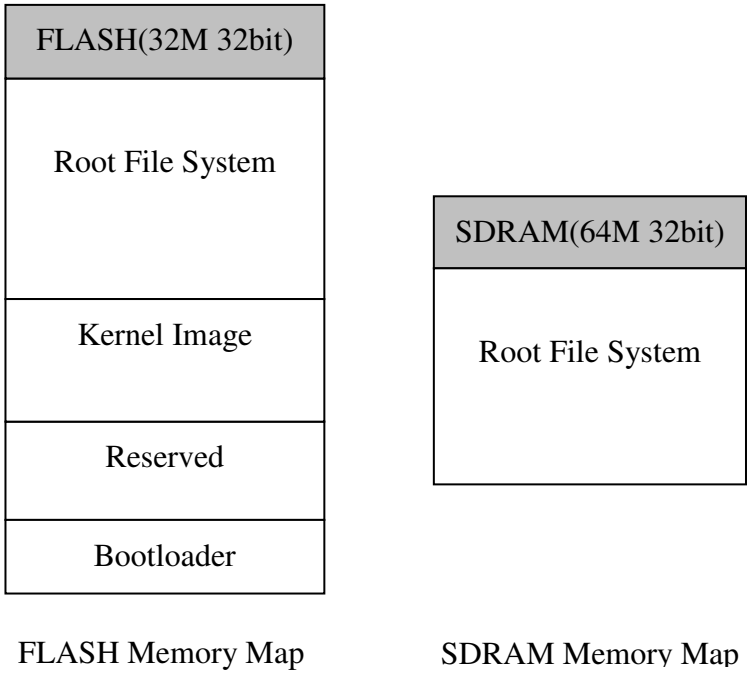

圖 3 Memory Map

3.2.3 開發方式

由於嵌入式系統的系統資源不如一般 PC 強大，所以通常選擇 PC 作為開發平台(Host System)，當程式碼開發完成，便透過 cross compiler 產生目標平台可以執行的程式，最後下載至目標板(Target Board)進行測試。Host System 和 Target Board 之間的傳輸方式，常見的有 JTAG、Serial、Ethernet(須跳線)。

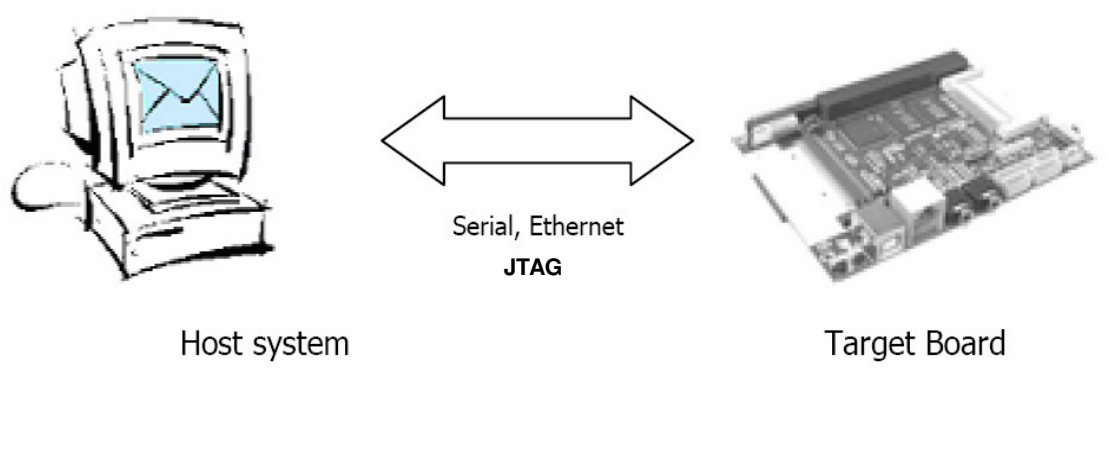

圖 4 開發環境示意圖

- 工具程式和用途將決定使用的傳輸方式：

  一、 HyBus 提供的 JFlash-XHYPER255 工具程式，透過 JTAG 將 Bootloader 燒錄在 Flash Memory 的位址 0x0。

  二、 HyBus 提供的 Bootlooader 支援經由 Serial 操縱終端機，在開機 3 秒內按任意鍵，便可進入該終端機。並且提供數個指令，其中的 bootp 指令和 tftp 指令可透過 Ethernet 從 Host System 的 tftp server 下載 Bootloader、Kernel、Root File System 等檔案，使用 Ethernet 的優點是比 Serial 快上許多，大幅減少等待時間。tftp 指令下載的檔案會暫存在 SDRAM 裡，使用 flash 指令可將檔案依其用途儲存到 Flash Memory 的特定位址。

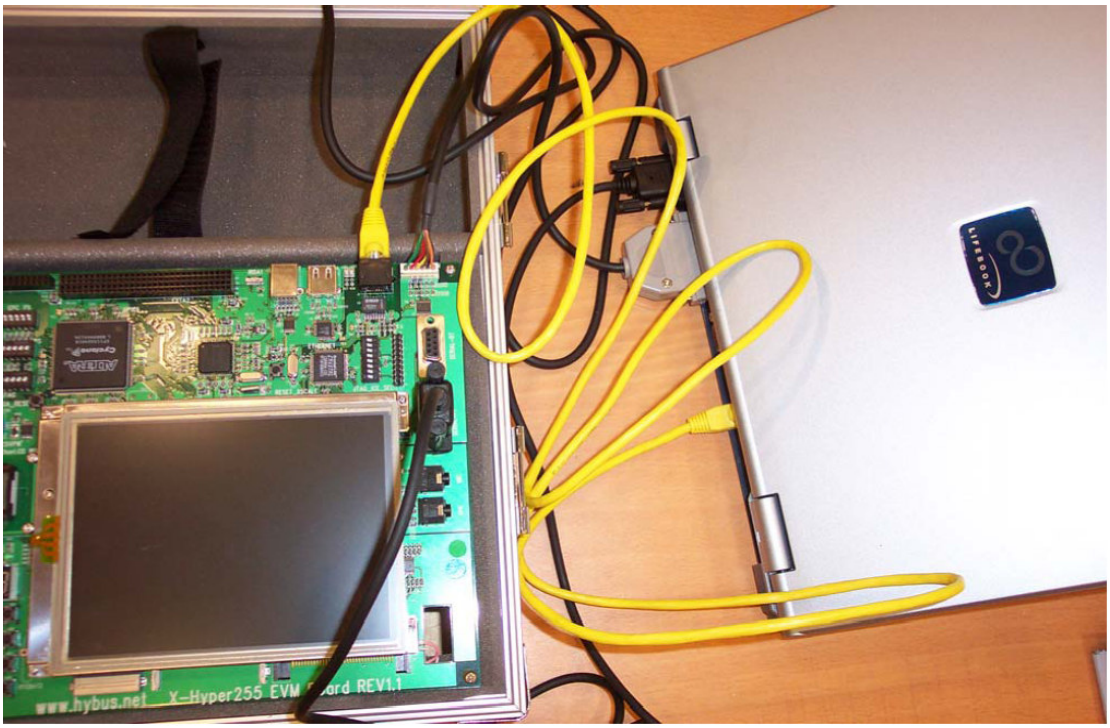

圖 5 Host System 和板子間用 JTAG、Serial、Ethernet Cable 連接

三、 Host System 透過 Serial 來操縱 Target Board 的終端機是很重要的開發方式之一。Target Board 的 bootloader 或 Linux 的命令列終端機將只透過 Serial 來輸入、輸出，開發者利用這個特點，便可以在 Host System 操縱和觀察 Target Board 的系統狀況，或測試程式的正確性；LCD 只顯示與應用相關的畫面，不應出現多餘的訊息。

- Host System 常見的終端機連線軟體：
  - Linux 的「minicom」，如圖 6。
  - Windows 的「超級終端機」，如圖 7。

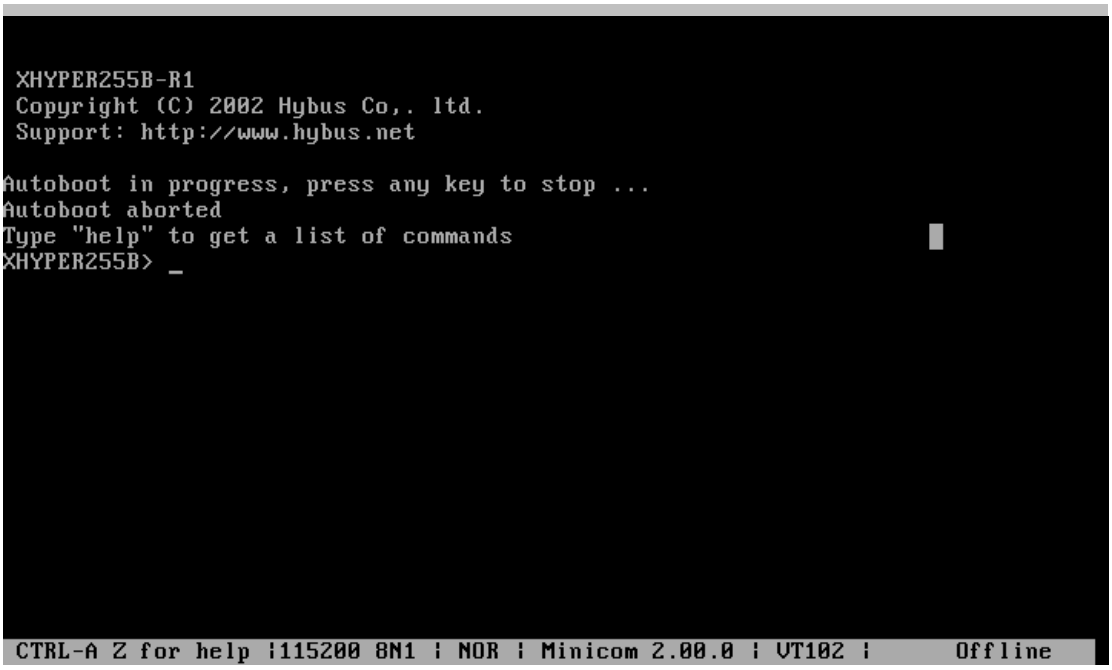

圖 6 Host System 之 minicom 畫面

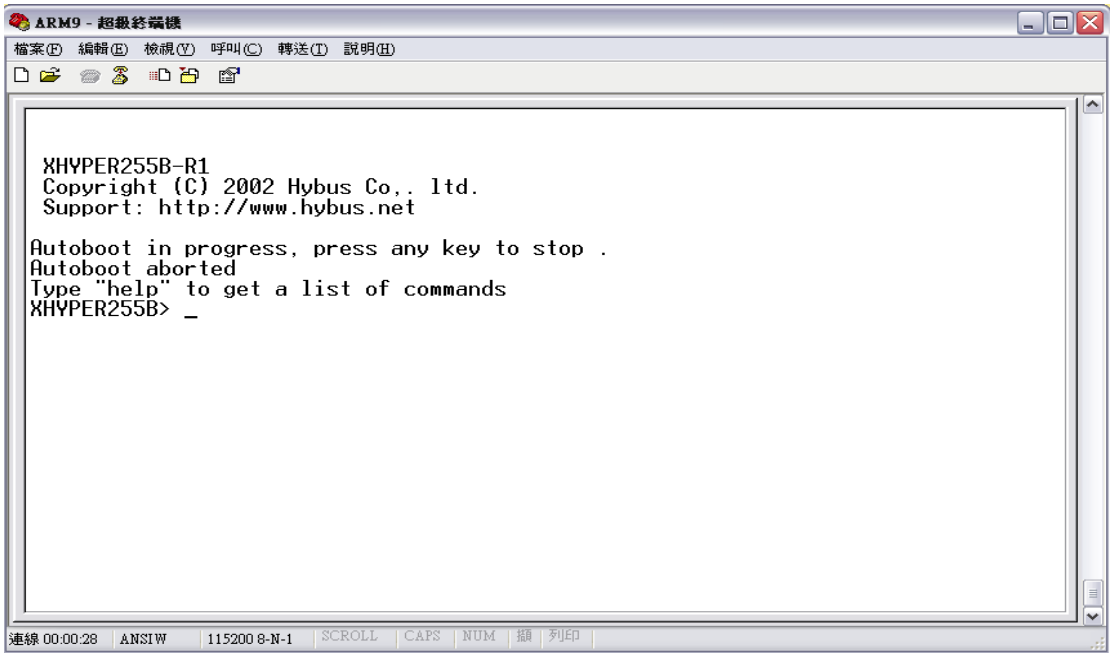

圖 7 Host System 之超級端終機畫面

3.2.4 工具鏈(toolchain)

通常 Host System 和 Target Board 的平台並不相同，因此在 Host System 開發的程式，必須經由 cross compiler 編譯出 Target Board 可執行的程式。Toolchain 指的

就是一連串的開發工具集合，包括 compiler、linker、library 等等。由於 X-Hyper255B-TKUIII 使用的處理器 PXA255 是屬於 ARM 架構，因此必須使用 arm cross compiler 來編譯目標板的程式。專題實作中我們使用 HyBus 提供的 Toolchain 來開發 bootloader、Linux kernel 及應用程式，使用了以下的 GNU Tools：

- GNU gcc
- GNU binutil
- GNU C Library
- GNU C header

使用方式和一般 PC 幾乎相同，指令字首則多了 arm-linux-，例如 arm-linux-gcc、arm-linux-strip。並且使用 toolchian 下的 library 和 header。

3.2.5 GUI環境 - Microwindows & Frame buffer介紹

一、Framebuffer

Framebuffer device 是圖形化獨立硬體的一種抽象概念，允許應用軟體透過 well-defined interface 存取圖形化硬體，以至於軟體不需要知道任何事情關於 low-level 的 interface stuff，且在不依靠 system-specific libraries 的平台上顯示圖形，例如 svgalib 或 X Window 系統沉重的 overhead。而 Framebuffer 是 VESA(Video Electronic Standards Association)的一項圖形介面標準，Linux kernel 在 2.12 之後就支援 VESA framebuffer，且目前多數顯示卡都相容於 VESA framebuffer 規格。

現今幾個 Linux 程式，例如 Mplayer 播放軟體和像 SDL 或 GTK 這樣的資料庫能立即使用 Framebuffer 避免 XFree86 系統的 overhead，這在嵌入式系統裡更是受歡迎。使用到 Framebuffer 的主要原因是 Linux 支援，加上

Framebuffer 在所有平台上都可以顯示，減少移植上的困擾；以 Microwindows 為例，Framebuffer 與一般顯示比較如圖 8(註：MW 為 Microwindows)。

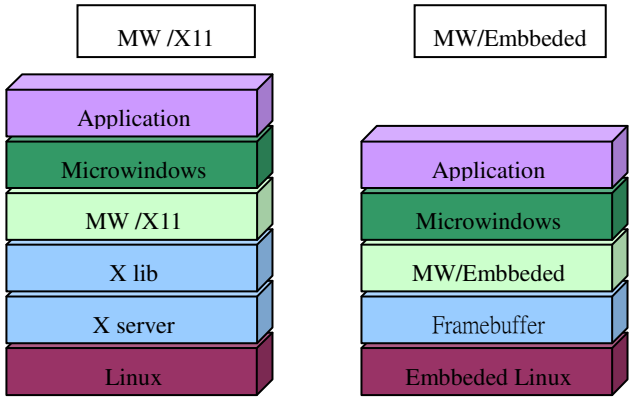

圖 8 Framebuffer 與一般顯示比較圖

二、Microwindows

Microwindows 是美國 Century Software 公司的 open source 專案，設計用於攜帶小型顯示單元的微型設備。能與微軟視窗、UNIX 作業系統上的 X server 相容的圖形介面的視窗系統，且需要的記憶體比較少。Microwindows 體系是基於典型 Client-Server 結構體系的 GUI 系統，並且具有分層設計，Microwindows 是採用三層式架構設計，如表 3，分層清楚讓每一層各司其職。

表 3 Microwindows 的分層式架構(Layered Design)

Lowest Level	螢幕、mouse/touchpad、keyboard 與其它硬體的 driver。
Mid Level	可移植(portable)與 device-independent 的繪圖引擎(graphics engine)提供對線的繪製、區域的填充、多邊形、裁剪以及顏色模型的支援，並進行視窗管理。
Upper Level	提供給 programmer 的 API，我們的程式只能呼叫 Microwindows 的 API 來繪圖，相容於 X Window 和 Windows CE（Win32 子集）的 API。

Microwindows 的優點為—在 PC 上支援 X，易於測試發展、針對 Embedded System 支援 Framebuffer、系統小、API 簡單易學、支援 TrueType fonts 字型處理和 ARM/MIPS/ELKS (註:ELKS 為 16-bit 的 Linux kernel)等等,其中「API 簡單易學」和「針對 Embedded System 支援 framebuffer」是本專題使用 Microwindows 的最主要原因。

而 Microwindows 支援兩種 API — Microwindows API 和 Nano-X API：Microwindows API 分別與 Microsoft Win32、WinCE GDI 相容，因此使用 Microwindows API 的視窗系統為 message-based 的架構，就是 message-passing system 的視窗系統;Nano-X API 可以比喻為小型的 X server 系統，Nano-X API 是依據 Xlib API 來設計，因此可以很直覺的了解到利用 Nano-X API 所設計的視窗系統為 client-server 架構。

既然是使用 Microwindows，當然會有相關的驅動程式來啟動各相關 I/O，Microwindows 的驅動程式對於 Linux Driver 的設備支援如表 4。

表 4 Microwindows 的設備支援

螢幕	支援 FrameBuffer,在 16 位元的 Embedded Linux Kernel(ELKS)亦可運作。
滑鼠	支援 GPM 和 Serial Mouse。
鍵盤	支援 tty keyboard 讀取。

有了驅動程式以後，我們需要 Microwindows 的核心圖形功能，其內嵌在與 device-independent 的繪圖引擎，並成為螢幕、滑鼠和鍵盤與硬體聯繫的橋樑,使用者應用程式不用直接呼叫核心繪圖引擎程式,而是透過 programmer 的 API。核心繪圖引擎的程式因為許多因素和應用 API 分離，使核心程序在

Client-Server 架構的伺服器環境裡。而核心程式使用 internal text font 和 bitmap 的方式，是為了速度而設計，並且可以不與標準 API 架構裡使用的一樣。另外核心程序是透過指標，能同時使用更多的複雜函數。

●ARM 上之 Linux 系統建置

學習建置一個 Linux 系統，對我們而言也是重要的學習與工作，在嵌入式系統必須盡可能的量身訂做，以符合需求並且去除掉不必要的元件，因此有必要自行編譯 Linux kernel 和製作 Root File System，挑選適當的系統程式、應用程式，編輯撰寫系統相關的組態設定和 script。在這些工作之中，也必須對 Linux 啟動程序有所瞭解。

4.1 系統需求

系統需求方面，主要有兩方面需要考慮：

(1) **核心**

我們欲建置最小需求之 Linux，基於只需要用到聲音和網路的部份，所以在編譯核心的時候盡量將不必要的選項拿掉，但是必須特別注意的是，使用 Microwindows 來實做使用者介面時，必須將 frame buffer 的選項編進核心當中，另外，如果跟檔案系統需要用到 MTD 技術，如 jffs2 的檔案系統格式，必須將 MTD 選項編進核心當中。

(2) **Root File System**

此方面有兩種考量：ramdisk， jffs2 的檔案系統格式，而函式庫的選擇是利用 GNU Glibc 的函式庫中，挑出我們需要的函式庫， 值得注意的是，我們編譯的方式是使用 shared library 的方式，比起一般靜態編譯，可以更有效的使用記憶體，另外，透過 busybox 的基本系統工具來取代原本的工具，也可以節省 sdram 的使用。

4.2 系統架構

一個可啟動的 Linux 系統，包含三類基本軟體：Boot Loader、Linux Kernel、Root File system。在目標板上，這些軟體將儲存於 Flash Momery 中固定的位址，可參考圖 9，而 Boot Loader 則是使用 Hybus(目標板廠商)提供的。

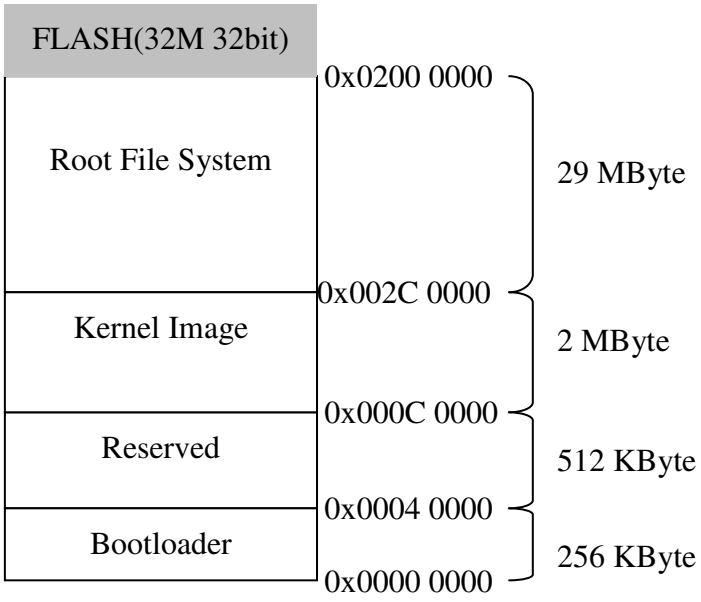

圖 9 FLASH Memory Map

- Linux 軟體架構：

  Applicaton 可以透過 Shared Library，間接使用 System call，可說是 Application 與 Kernel 間的介面。圖 10 為 Linux 的軟體架構，白色區塊存放於 Root File system，灰色區塊存放於 Kernel Image，值得注意的是，Modules 雖放置於 Root File system，但卻是工作於 Kerenl space，可於系統啟動後，再另行載入、卸載 Modules，可說是 Linux 的一大優點。

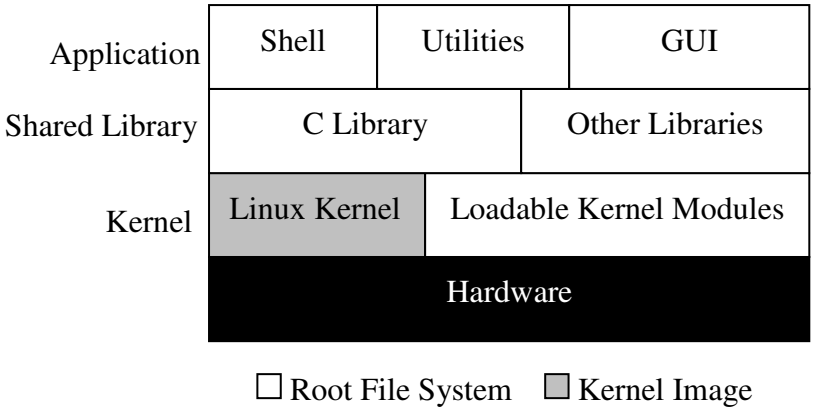

圖 10 Linux 的軟體架構

- Linux 系統與 GUI 系統架構：

  GUI 系統並不屬於 Linux 核心的部份，因此是在基本的 Linux 系統上，再加以建置。如圖 11 所示。

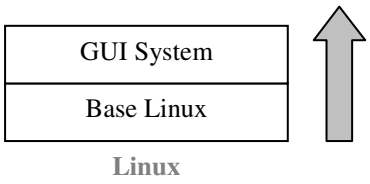

圖 11 Linux 系統與 GUI 系統架構

4.2.1 啟動程序

利用 x86 平台加上 RedHat9.0 作業系統為實驗，在開機的過程中，從使用者打開電源(Power)直到螢幕出現命令行提示字元(login)的整個 Linux 啟動過程，了解整個系統的開機流程到底是什麼。

在開機整個流程，了解 Linux 原始碼，其實是學習 Linux 的最好方法，Linux 啟動過程的介紹中，也嘗試從原始碼的視角來更深入的探討，其中也簡單涉及到部分相關的 Linux 原始碼，Linux 啟動的原始碼主要使用的是 C 語言，也涉及到了

少量的組合語言。而啟動過程中也執行了大量的所寫的 Shell Script(主要是 bash shell)，可以透過圖 12 來了解開機的整個流程。

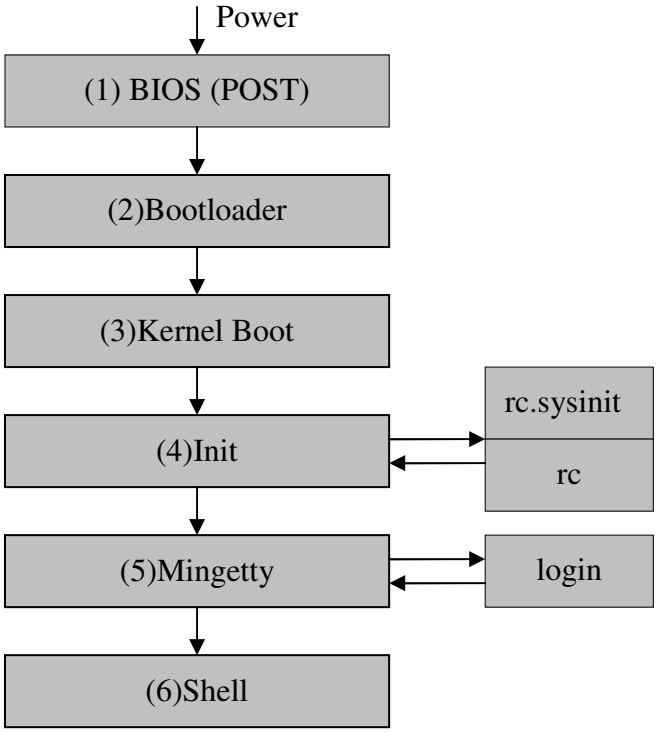

圖 12 開機流程

當開啟 PC 的電源，BIOS 開機自我檢查（POST），按照 BIOS 中設置的啟動設備(通常是硬碟)啟動，接著啟動設備上安裝的導引程序(boot loader)lilo 或 grub 開始導引 Linux，Linux 首先進行核心(Kernel)的導引運作，接下來執行 init 程序，init 程序呼叫了 rc.sysinit 和 rc 等程序，rc.sysinit 和 rc 當完成系統初始化(initialization)和執行服務的任務後，返回 init；init 啟動了 mingetty 後，打開了終端機供使用者登錄系統，使用者登錄成功後進入了 Shell，這樣就完成了從開機到登錄的整個啟動過程。

主要的開機程序可以分成圖 12 的數個流程，接下來我們簡單的敘述每個流程的動作，而從原始碼角度來看的詳細過程放於附錄 B 當作參考。

(1) **BIOS (POST)**

BIOS (Basic Input/Output Setup)的主要作用是作自我檢測(POST) ，是第一個被載入電腦的資料，包含 CPU 資料、開機順序、硬碟大小、晶片組工作狀態、PnP (Plug and Play)的開啟與否、記憶體的時脈等都記錄在 BIOS 當中，開機之後，系統會先去找 BIOS 作硬體初始的動作。

(2) **Bootloader**

個人電腦的系統在讀完 BIOS 之後，會先去讀取第一個開機硬碟的第一個磁區（master boot record， MBR），這個磁區主要就是在記錄開機的資訊，LILO 或 GRUB 等 Bootloader 紀錄就是在 MBR，例如 LILO 記錄的資訊就會被讀出來，並依內容執行不同的系統，也就是多重開機的設定。

(3) **Kernel Boot**

當在 LILO 的選單中選擇我們要的 Linux 系統，開發平台跑到 Linux 所在的硬碟之下，就開始將所需要的核心載入。在 Linux 的系統下，通常開機的核心都擺在 /boot 底下，因此這時候的 boot loader 就會到/boot 去尋找相關的核心。本專題的 kernel 名稱通常是/boot/vmlinuz-2.4.20-8(Red Hat 9.0)的格式，所以使用 uname –r 的指令就可了解目前的核心版本。當 kernel 載入之後會作一系列系統初始和設置的動作，我們可以從 trace 原始碼的角度來觀察，而這一系列的核心資料結構初始化及外部初始化的動作請參考附錄 B。

(4) **Init**

init 的 process id (pid)為 1，因此 init 行程是系統所有行程的起點，Linux 在完成 kernel 導引以後，就開始執行 init 程序，init 程序需要讀取配置文件 /etc/inittab 的初始化檔案。inittab 是一個不可執行的文字檔案，由若干指令所組成。Inittab(initialize table)的詳細內容請見附錄 C。在 init 作系統初始化

的動作時，呼叫執行/etc/rc.d/rc.sysinit，故在 init 的 inittab 中有這麼一行—

si::sysinit:/etc/rc.d/rc.sysinit

rc.sysinit 是一個 shell script，主要是完成一些系統初始化的工作，rc.sysinit 是每一個 run-level 都要首先執行的 shell script。主要工作有啟動 swap partition、檢查硬碟、mount 硬體模組區塊、以及其它一些需要優先執行任務。

在 rc.sysinit 執行後，返回 init 繼續其它的動作，接下來啟動核心的外掛式模組 (/etc/modules.conf)，接著執行到/etc/rc.d/rc 程序，rc 會根據不同的 run-level 啟動不同的 system service 的 daemon，當 daemon 啟動完成，rc 程序也就執行完畢，然後再返回 init 繼續下一步。

(5) **Mingetty**

rc 執行完畢後，返回 init。此時候基本系統環境已設置好，各種 daemon 也已經啟動。init 接下來會打開 6 個終端機，以便使用者登錄系統。通過按 Alt+Fn(n 對應 1-6)可在這 6 個終端機中切換，也可以自己設定終端機的個數。

- 1:2345:respawn:/sbin/mingetty tty1
- 2:2345:respawn:/sbin/mingetty tty2
- 3:2345:respawn:/sbin/mingetty tty3
- 4:2345:respawn:/sbin/mingetty tty4
- 5:2345:respawn:/sbin/mingetty tty5
- 6:2345:respawn:/sbin/mingetty tty6

在 2、3、4、5 的 run-level 中都將以 respawn 方式執行 mingetty 程序，mingetty 程序能打開終端機、設置模式。同時它會顯示一個文本登錄界面(run-level 3)，這個界面就是我們經常看到的登入界面，在這個登錄界面中會提示輸入使用者帳號、密碼，傳給 login 程序來驗證使用者的身份。當 init 執行完之後，在/etc/rc.local 當中我們可以設定一些平常手動載入的模組例如光碟或是 NAT 的服務，可以寫到 rc.local 中，在開機時就會自動載入，

(6) **Shell**

login 程序成功後，會向對應的終端機再輸出最近一次登入的訊息(在 /var/log/lastlog 中有記錄)，並檢查使用者是否有新郵件(在/usr/spool/mail/的對應使用者名目錄下)。接著開始設置各種環境變數，對於 bash 來說，系統首先尋找/etc/profile Script 文件，並執行它；如果使用者的主目錄中存在.bash_profile 文件便執行，在這些文件中有可能呼叫了其它配置文件，所有的配置文件執行後，各種環境變數也設好了，這時會出現熟悉的命令列提示(command prompt)，到此整個啟動過程就結束了。

4.3 Kernel

4.3.1 核心選擇

核心原始碼主要可以在 http://www.kernel.org/ 中找到，裡面提供了許多版本的 kernel。本專題主要的硬體平台是在 x86 和 ARM-9 目標板上，首先，在 x86 上是以 2.4.20 為實驗版本，要特別注意的是，並不是每一個版本都可以支援開發的架構，因此選擇的必須是比較穩定的版本，ARM 的 kernel 版本可以參考 http://www.arm.linux.org.uk/developer 來找到相關的資訊，以 ARM 為例，主要是使用 HyBUS 已經 Patch 過的 Kernel(2.4.18-rmk7-pxa1-xhyper255)，主要是使用 ARM 架構(patch-2.4.18-rmk7)、CPU 型號(diff-2.4.18-rmk7-pxa1)、及 ARM 目標版(xhyper255_TKU3-2.4.18-rmk7-pxa1) 的 Patch，Patch 過的 kernel 當中便包含了許多 ARM 和目標板所需要的功能核心組態詳細設定請見 http://bbs.iecs.fcu.edu.tw/~viggo/project/config/arm_kernconf.htm。

在 x86 核心組態的設定上可以參考

http://bbs.iecs.fcu.edu.tw/~viggo/project/config/x86_kernelconf.htm ，核心編譯的步驟簡單可分為：

(1) make mrproper

(2) make menuconfig

(3) make dep

(4) make clean

(5) make bzImage

(6) make modules

(7) make modules_install

(8) make install

4.3.2 模組與驅動程式

在探討核心模組之前，先比較一下核心模組和一般應用程式的差別。一般應用程式從啟動到結束，執行過程都是保持在單一任務(single task)的狀態，而模組必須先向系統註冊(register)，註冊完之後隨即結束。當系統收到要求(request)命令後，模組就可以為其服務。大部分的作業系統會將程式的運作空間分開，屬於作業系統的程式通常會和一般使用者的程式分開，因此有所謂的使用者空間(user space)和核心空間(kernel space)，模組是在核心空間執行，而一般應用程式是在使用者空間執行。

此外，在專題當中聲音的使用是直接讀寫 device file(裝置檔)來實作，例如開啟一個聲音裝置檔/dev/dsp，可以透過這個裝置檔使應用程式經由核心來跟裝置溝通，並直接利用 read( )/write( )等系統呼叫對其作存取，而不用擔心程式與核心或裝置的溝通問題。實際上，裝置檔本身並非真正的檔案，而只是與核心溝通的

一道橋樑，裝置檔會有兩個欄位值，分別是主編號和次編號，當我們使用 I/O 發出請求時會傳給核心，核心再針對主編號找到對應的驅動程式，該驅動程式則利用次編號來選擇的裝置實體。裝置檔的命名其實是沒有限制，核心只管主編號和次編號。有時為了方便記憶，會利用符號鏈結來作裝置的存取，而不直接使用裝置檔，例如符號鏈結/dev/mouse 經常鏈結到/dev/ttyS1。而在本專題的 Voice Messenger 中，主要是使用到網路和聲音的模組。

4.4 Root File System

在 Linux，檔案都存放在 File System 裡，File System 通常佔據一個磁碟分割區(partition)，或映像檔(如 loopback file)。File System 會對應到樹狀目錄(directory tree)上的某個部份，稱為 mount point。

Linux 需要 mount 一個必備的 File System，這裡面包括啟動、系統相關、一般用途等等相關的檔案，以及整個系統的樹狀目錄，這就是所謂的 Root File System，它會對應樹狀目錄的最上層目錄「/」(根目錄)。

4.4.1 目錄結構與所需檔案

在這節將針對我們的 Linux 系統，描述 Root File System 所包含的幾類型重要檔案。詳細的「目錄與檔案結構」列表於附錄 D。

- **系統相關：**
  - 一、 系統初始化相關的 script 和組態設定檔—如 inittab、rc、sysinit.rc、fstab、module.conf 等等，其存放在/etc。
  - 二、 device file—應用程式可經由 device file 使用各種裝置，其存放在/dev。

三、 shared library—從 glibc 套件中挑選應用程式需要的 library，其存放在 /lib。

- **工具程式：**

shell、init、mount、ls 等工具程式採用 busybox，它是一種專為小型系統所開發的系統工具套件，可以選擇所要包含的指令，編譯可以採用動態或靜態編譯。編譯完成會產生一個 busybox 的執行檔，所需的命令則是用 symbolic link 的方式連結，如：ln -s busybox ls，會產生名為 ls 的符號連結，使用時會連結到 busybox，一般存放在/bin 和/sbin。

- **其它檔案：**

使用者相關的應用程式，如 Microwindows、Window Manager、軟鍵盤、Voice Messenger，一般存放在/usr/bin。

4.4.2 選擇 JFFS2 (MTD 應用)或 ramdisk

Linux 支援多種的 file system，每個 file system 各有它的優缺點，可依系統需求選擇。以下針對在 Embedded Linux 常見的兩種做介紹。

JFFS2(Journalling Flash File System 2)使用 MTD(Momery Technology Device) driver，讓儲存於 Flash Momery 的 file system 可以直接 mount，也可以直接在 Flash Momery 上修改、刪除、與新增檔案。JFFS2 也支援壓縮，雖然不如 gzip 效果那麼好。但它最大的優點是不必將 file system 整個載到 RAM 裡，可以節省很多 RAM 空間。

RAM Disk 相當常見，一般的 Linux、開機片經常使用的 initrd(initial ram disk)都

是使用這種技術。它將 RAM 的部份空間當成 disk，把一個 loopback file system 的檔案 mount 在上面來存取，該檔案可先用 gzip 壓縮儲存。由於 file system 會載入至 RAM 裡，因此會浪費許多 RAM 的空間，但系統的效率也比較好。然而在 file system 裡所做的修改，都會隨著關機而消失。

由於我們對速度的需求不大，並讓使用者可修改偏好設定。因此選擇 JFFS2 較為方便。

●應用程式實作

本章為專題的主要重心—實作 voice messenger。首先介紹其程式架構，可了解整個 voice messenger 流程。再對網路運作與聲音處理做詳細說明。接著了解 microwindows 如何實作使用者介面。最後附上整個實作成果，助於讀者們對 voice messenger 認識。

5.1 需求規格與功能

製作一個線上語音交談軟體，和專門用來管理名單、協助連線的 server 並且能在 ARM 跟 x86 上運作。

- **Server 的功能：**
  1. 能夠處理 Client 斷線和錯誤
  2. 告知 Client 目前上線名單
  3. 作為 Client 間線上語音交談的一個橋樑
  4. 通知 Client 停止線上語音交談

- **Client 的功能：**
  1. 圖形化使用者介面
  2. 線上語音交談
  3. 使用者偏好設定

5.2 自訂協定與運作架構

TCP/IP 的協定下，最主要的協定為 TCP 和 UDP。TCP 提供的是一個連線導向(Connection Oriented)的可靠傳輸，在封包遺失的情況下，將會重新傳送。相反地，UDP 則是一個非連線型(Connectionless)的非可靠傳輸協定，它沒有確認封包是否收到的機制，對於網路頻寬的負擔較小。在傳輸大量且具時效性的資料，UDP 是個不錯的選擇。

由於 Server 端需要掌握正確即時的線上名單，所以採用保持連線的 TCP。在傳輸聲音的部份，使用 TCP 的方式並不恰當，於是採用 UDP 來減少網路頻寬的使用。

5.2.1 運作架構

一、client-server 架構

server 統合管理系統資源之配置與使用方式，以供 client 應用。

(1) *server 特性：*

- 被動
- 等待需求
- 答覆與服務 client

(2) *Client 特性：*

- 主動
- 傳送需求
- 等待答覆與服務

其優點是可以有效管理統一存取的資源，但畢竟 server 只有一個，當有更多的要求服務時，勢必將會引響整個存取的效率。

二、P2P(peer to peer)架構

每個 Peer 同時都是提供服務的 server 跟接受服務的 client，peer 間地位都是平等的。以一般我們交談的方式來看，並沒有所謂的 client-server 架構，而且為了減少 server 的負擔，所以傳輸聲音採用 P2P 的方式，不再經過 server。也就是說運作架構統一存取的部份採用 client-server 架構，而兩兩間私底下的採用 P2P 架構。

- **以下為運作流程：**

1. 各 Client 端連上 Server 端

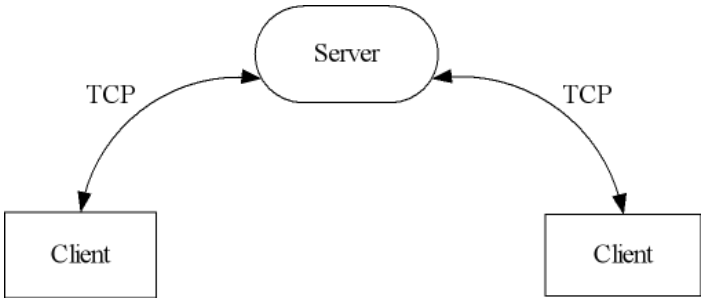

圖 13 運作流程步驟 1

2. 當 Client 間需要開啟線上語音交談時，Server 端開啟一個 UDP 的服務
3. 各 Client 端連上剛剛 Server 端開啟的 UDP 服務
4. Server 利用此 UDP 服務，收集連上 UDP 服務的 Client 端連線資訊
5. Server 利用 TCP 將開啟線上語音交談的必要連線資訊傳給 Client 端

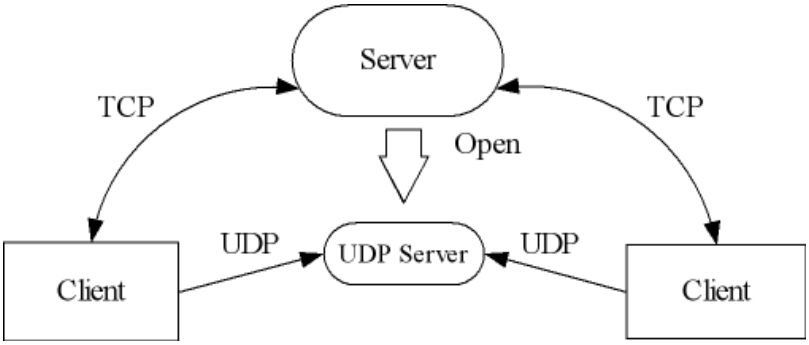

圖 14 運作流程步驟 2~5

6. Server 關閉 UDP 的服務
7. Client 端利用收到的連線資訊，以 P2P 架構開啟線上語音交談

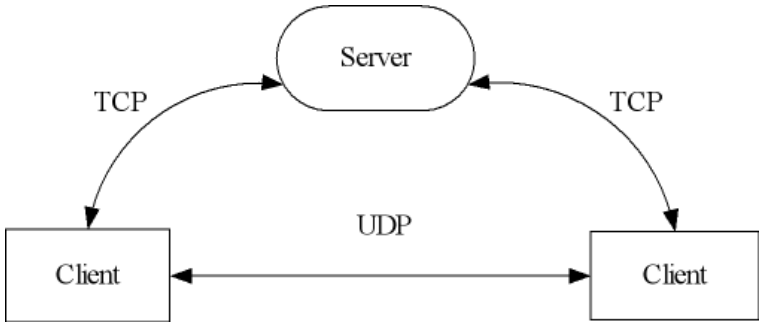

圖 15 運作流程步驟 6~7

在進行語音交談時，Server 開啟了一個額外的 UDP 服務，而不直接 Client 端互連，主要用意是在解決 Private IP 的問題，這將在下一節詳細描述。

5.2.2 NAT & Private IP 問題

由於近年來 Internet 的蓬勃發展，IPv4(Internet Protocol Version 4)位址空間即將耗盡，再加上未來無線上網及 3G 的潛在用戶，都需要龐大的 IP 位址來支援。IPv4 位址已顯得越來越缺乏，而在下一代的網路通訊協定 IPv6 完全運作前，採取 NAT(Network Address Translation)來暫時解決 IP 位址不足的問題。

NAT 的機制主要用於單一個 IP Address 下，讓多台電腦能透過該 IP 來存取網路資源。舉例說明實際的運作：

＜User 透過 NAT 主機來與外界做溝通＞

1. User 將訊息傳送給 NAT 主機
2. NAT 主機開啟一個 PORT 去對應該訊息來源(User)的 IP 跟 PORT，並紀錄在 NAT Table
3. 更改訊息來源為 NAT 主機的 IP 和剛開啟的 PORT
4. 將訊息傳給外界
5. 外界回傳訊息給 NAT 主機
6. NAT 主機根據訊息目標的 PORT，查詢 NAT Table，將訊息傳給 User

NAT Table

NAT PORT	電腦1 IP	電腦1 PORT
5000	192.168.10.5	7899

圖 16 NAT 機制運作

在這裡看到 NAT 的機制是內部電腦對外連線時，NAT 主機建立一個對應該電腦的 Port，除非有事先設定 NAT 主機，在沒有對外的連線動作之下，NAT 內部的 Private IP，外界是無法直接存取他們。雖說有安全性上的好處，但在一般的 Client-Server 架構下，將無法與擁有 Private IP 的 Server 端作溝通。

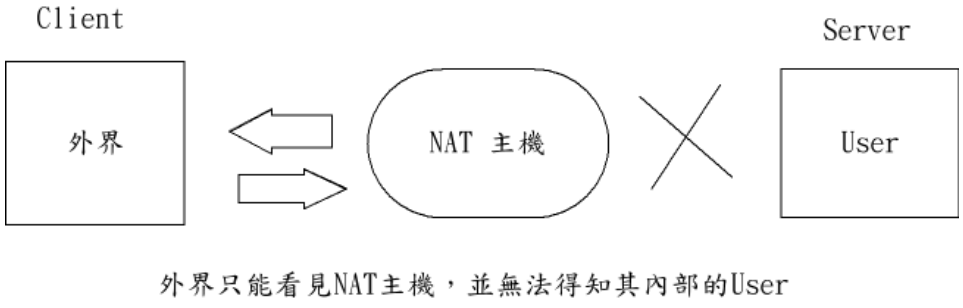

圖 17 Private IP 的問題

- **使用 UDP 協定來解決 NAT & Private IP 的問題：**

  由於 UDP 不像 TCP 一樣需要持續的連線，也就沒有很強烈的 Client-Server 的關係，網路傳輸的負擔也比較輕。雖然它是一種不可靠的協定，有可能會遺失封包，不過在傳送大量即時的聲音，少許的遺失是可以被容忍的。

  例如有兩台電腦都是在 NAT 的機制下運作，而想要直接溝通。由於 UDP 沒有真正 Client-Server 的關係，利用一點小技巧，改變傳送訊息的目的地，讓兩台電腦只是單純的傳送跟接收訊息，另外 NAT 主機所開啟的 Port 對應機制也可以被利用到。

- **實際運作：**

  1. 使用者透過 NAT 主機先連上一個 UDP 的 Server 端，Server 端抓取各 NAT 主機連線過來的 IP 與 Port

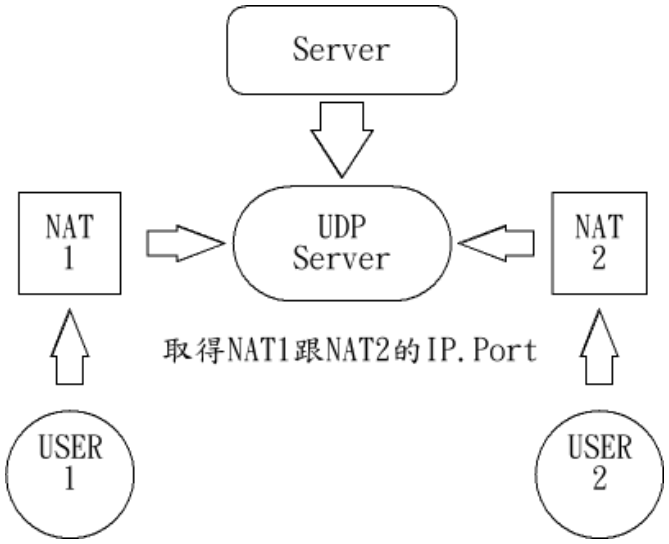

圖 18 小技巧步驟 1

2. 接下來 Server 用 TCP 把對方 NAT 主機 IP 跟 Port 傳送給 User

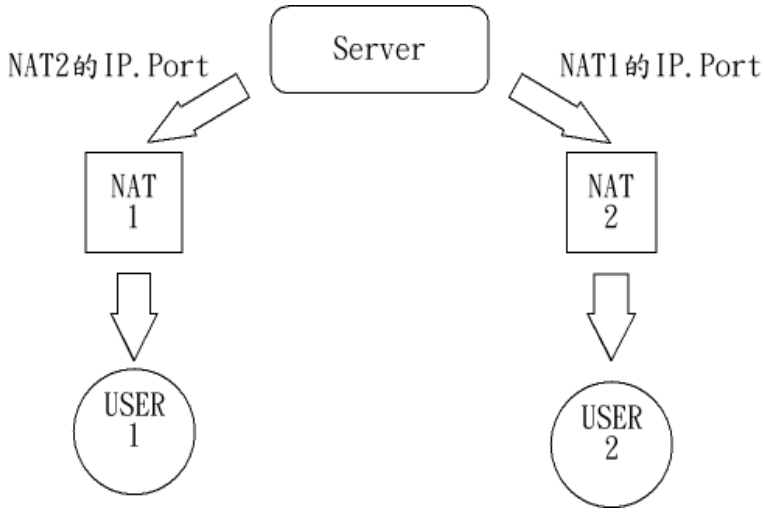

圖 19 小技巧步驟 2

3. User 利用所得到的 IP 跟 Port 改變 UDP 傳輸的目的地，由於已經有過對外的連線動作，如此一來就可利用 NAT 主機先前開啟的 Port 去與對方電腦溝通，而解決了 Private IP 的問題

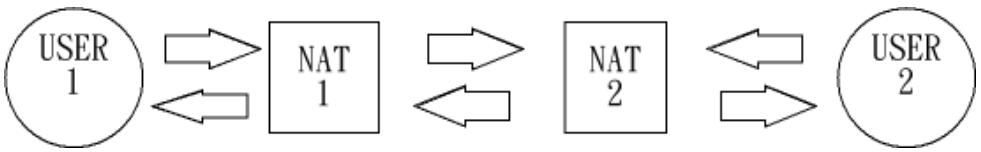

圖 20 小技巧步驟 3

5.2.3 聲音處理

一、聲音數位化與所需規格

數位化聲音的品質主要取決於取樣頻率(sampling rate)、取樣大小(sampling size)，以及頻道數(channel)。詳細說明如下，用途與規格的比較參考表 5。

(1) 取樣頻率(sampling rate)是指每秒取樣的次數，根據 Nyquist 的理論，只要取樣頻率是原始音源的 2 倍，就能完整記錄訊息，而人耳的接受頻率為 20~20kHz，因此高品質的數位聲音之取樣頻率也約為 2 倍，如 CD 品質為 44.1kHz。然而，語音的頻率範圍較窄，因此對於電話、語音用途，並不需要太高的取率頻率，一般使用 8kHz 左右。

(2) 取樣大小(sampling size)是指每個 sample 所佔的位元，也就是響度(loudness)可以表現的精確度。

(3) 頻道數(channel)，一般分為單音(mono)，和雙聲道以上的立體音(stereo)。由於語音通訊並不需要多個聲道來源，而且錄音裝置通常僅支援 mono，所以在此僅使用單聲道。

表 5 用途與規格比較

用途	取樣頻率 Sampling rate	取樣大小 Sampling size	頻道 channel
CD	44.1kHz	16 bits	Stereo
語音、電話	8kHz	8/16 bits	Mono

- 資料量 bitrate 的計算：

  取樣頻率 × 取樣大小 × 頻道數 = bitrate bps

- 以語音用途為例：8,000 x 16 x 1 = 128,000 bps (16,000 bytes/sec)

二、audio 程式設計

Linux 的音效系統有許多不同的團體在維護，其中的 open sound system(oss)，Linux 2.4 本身就有支援，因此這裡使用 oss 來開發聲音處理的程式。oss 的好處是相當簡單，使用我們熟悉的 Linux 的 System Call 就可以達成。

- 使用音效裝置的基本流程：
  1. 以 open( )開啟 device file，將回傳 file descriptor(以下簡稱 fd)。接下來將透過這個 fd 來使用裝置。
  2. 以 read( )/write( )來讀寫裝置，或以 ioctl( )設定裝置行為。
  3. 以 close( )關閉 device file。

Voice Messenger 所使用的 device file 和其用途如表 6 所示。不過 device file 的名稱(file name)只對應用程式有意義，kernel 則是以 major number 來辨識，這個部份已在「4.3.2 模組與驅動程式」說明過。

表 6 使用的 device file 與用途

major number	minor number	device file	description
14	0	/dev/mixer	設定錄音來源、設定音量
14	3	/dev/dsp	設定全雙工(full duplex)、設定音效參數(取樣頻率、取樣大小、頻道數)、錄音、放音

錄音與放音，其實就是直接對 device file 寫入資料或讀取資料，也就是使用 write( )、read( )來達成。

設定裝置行為，都是使用 ioctl( )來完成，ioctl( )可對裝置做 I/O 操作，所能控制的功能和裝置本身及驅動程式息息相關。ioctl( )所須傳入的第一個參數 fd 用來決定目標裝置，第二個參數 command 會決定控制功能，第三個參數(或以上)則是額外所需的資訊或將回傳的資訊。oss 的程式設計可以參考「OSS Programmer's guide v1.1」http://www.opensound.com/pguide/oss.pdf

三、程式流程

- 以下為 client 主程式開始語音交談的流程：

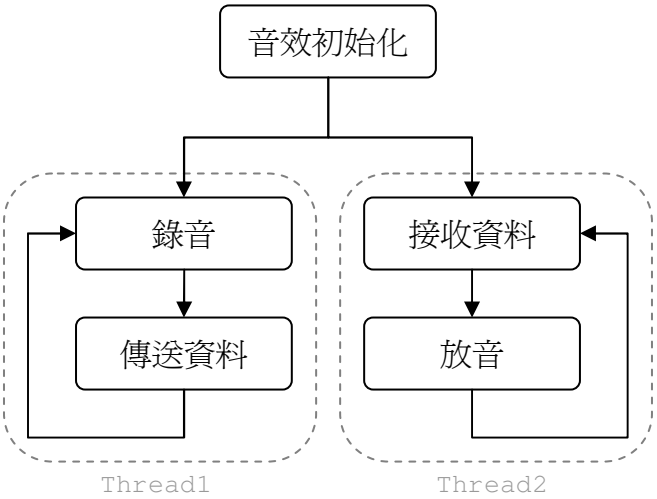

圖 21 語音交談流程圖

- 音效初始化的工作包括：
  1. 使用 mixer 裝置設定錄音來源與音量。
  2. 使用 dsp 裝置設定全雙工與設定音效參數(取樣頻率、取樣大小、頻道數)。

「錄音、傳送資料」和「放音、接收資料」各由一個 thread 負責，並且重覆運作。直到主程式下達終止動作或對方斷線。

5.3 利用 Microwindows 實作使用者介面

Microwindows 支援 1、2、4 和 8 bpp（每像素的位元數）的調色顯示，以及 8、16、24 和 32 bpp 的真實色彩顯示。Nano-X Server 佔用的記憶體大小約只有 100 K 到 150 K 位元組，除此之外 Nano-X 與 Xlib 實現不同，仍在每個 Client 端上同步運行。意指一旦發送了 Client 端要求封包，Server 在另一個 Client 端提供服務之前一直等待，直到整個封包都送達為止。這使 Server 程式碼非常簡單，而運行的速度仍非常快，故綜合以上優點選用 Nano-X API 來實作。libnano-X 為 Microwindows 的 Nano-X API 程式庫，共分為十大類，如表 7。

表 7 Nano-X API 程式庫種類

General 一般類	Window 視窗類	Graphic 繪圖類	event 事件處理類	Fonts 字型處理類
pointer 滑鼠指標類	colours 顏色處理類	Regions 視窗區域類	selections 內文選取類	misc 其他類

而專題實作的介面中主要用了 general〈一般類〉、window〈視窗類〉、graphic〈繪圖類〉的函式，其中比較特殊的是 GrNewGC( )函式，當我們對一個視窗做繪圖上的變化時需要引用 GC，設定 GC 函式有 GrSetGCForeground( )、GrSetGCBackground( )、GrSetGCUseBackground( )、GrSetGCMode( )、GrSetGCFont( )、GrSetGCClipOrigin( )、GrSetGCGraphicsExposure( )。

實作的按鍵就是使用兩種不同的 GC 去做變化，用 GrSetGCForeground( )、GrSetGCBackground( )函式定義前景與背景顏色，再利用畫邊框方式使視窗產生視覺上的立體感，按鍵的外觀就成形了。當滑鼠按下時，按鍵呈現如圖 22；當滑鼠彈起時，按鍵呈現如圖 23。

圖 22 介面按鍵按下

圖 23 介面按鍵彈起

視窗用 GrNewWindowEx( )或 GrNewWindow( )函式設定初始值，兩個函式所設定的資訊大同小異，GrNewWindow( )特殊的是可設定 border 的寬度和顏色，而 GrNewWindowEx( )特殊的是可設定視窗的屬性和標題文字，視窗屬性種類請參照表 8。

表 8 一般視窗屬性

視窗屬性值	描述
GR_WM_PROPS_NOBACKGROUND	不繪製視窗背景色
GR_WM_PROPS_NOFOCUS	不將焦點設定至此視窗
GR_WM_PROPS_NOMOVE	不允許使用者移動此視窗
GR_WM_PROPS_NORAISE	不允許使用者將此視窗移至上層
GR_WM_PROPS_NODECORATE	不重新繪製邊框
GR_WM_PROPS_NOAUTOMOVE	第一次 MAP 視窗時，不移動視窗
GR_WM_PROPS_NOAUTORESIZE	第一次 MAP 視窗時，不調整視窗大小

表中提到的 MAP 是指用 GrMapWindow( )函式設定視窗與其子視窗為可見的，GrUnmapWindow( )函式則相反。實作中的視窗切換就是使用這兩個函式，當我們按下 LOGIN 按鍵時，如果成功登入 Server 就會切換成 List 視窗。

在 Microwindows 的視窗中，要讓視窗與使用者互動來自於 GrSelectEvents( )函式，其用來設定視窗的 Event Mask，使視窗經由 Event Mask 接收 Event Type 的事件產生反應。由 GrGetNextEvent( )函式截取事件，然後由 Event Type 來分辨事件種類。藉由表 9 介紹實作方面有使用到的 Event Type。

表 9 使用到的 Event Type

Event Type 值	描述
GR_EVENT_ TYPE _EXPOSURE	當視窗任何一部份需要重繪時送出此事件型別給 nano-X client 端，此事件會伴隨出一個 GR_EVENT _EXPOSURE 結構送出。
GR_EVENT_ TYPE _BUTTON_DOWN	當按下滑鼠任何按鍵時送出此事件型別給 nano-X client 端，此事件會伴隨出一個 GR_EVENT _ BUTTON 結構送出。
GR_EVENT_ TYPE _BUTTON_UP	當放開滑鼠任何按鍵時送出此事件型別給 nano-X client 端，此事件會伴隨出一個 GR_EVENT _ BUTTON 結構送出。
GR_EVENT_ TYPE _KEY_DOWN	當按下鍵盤按鍵時送出此事件型別給 nano-X client 端，此事件會伴隨出一個 GR_EVENT _KEYSTROKE 結構送出。

5.4 實作成果

一、程式架構：

以下為程式概略之敘述，詳細程式碼可參考附錄 E。

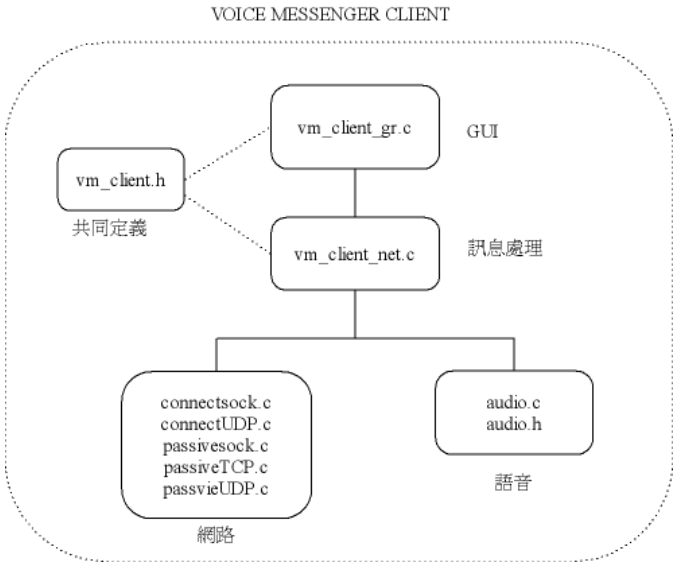

圖 24 Client 端程式架構

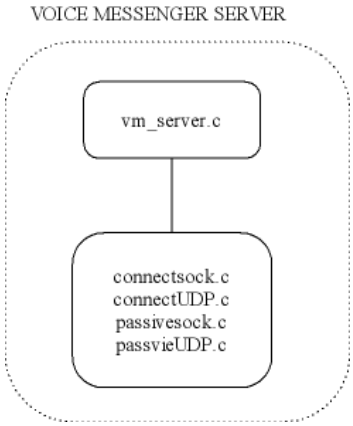

圖 25 Server 端程式架構

- vm_client_gr.c：使用者圖形介面
- vm_client_net.c：cient 端網路與聲音的實作
- vm_server.c：server 端網路服務實作
- audio.c audio.h：聲音介面

- connectsock.c：網路連線 socket 實作
- connectTCP.c：網路 TCP 連線介面
- connectUDP.c：網路 UDP 連線介面
- passivesock.c：網路服務 socket 實作
- passiveTCP.c：網路 TCP 服務介面
- passiveUDP.c：網路 UDP 服務介面

二、使用者介面介紹：

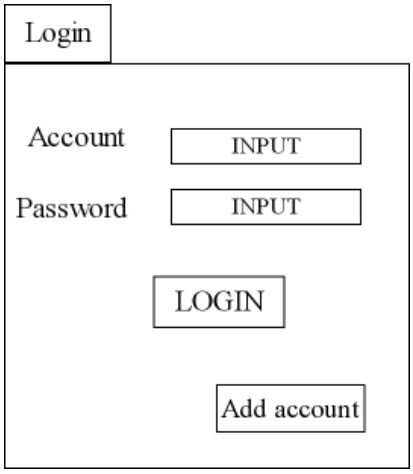

圖 26 登入介面

1. INPUT—感應 GR_EVENT_ TYPE _KEY_DOWN，用來接收鍵盤或軟鍵盤輸入
2. LOGIN—讀取 Account 跟 Password，之後連上 server 進行登入動作
3. Add account—讀取 Account 跟 Password，之後連上 server 進行創造帳號動作

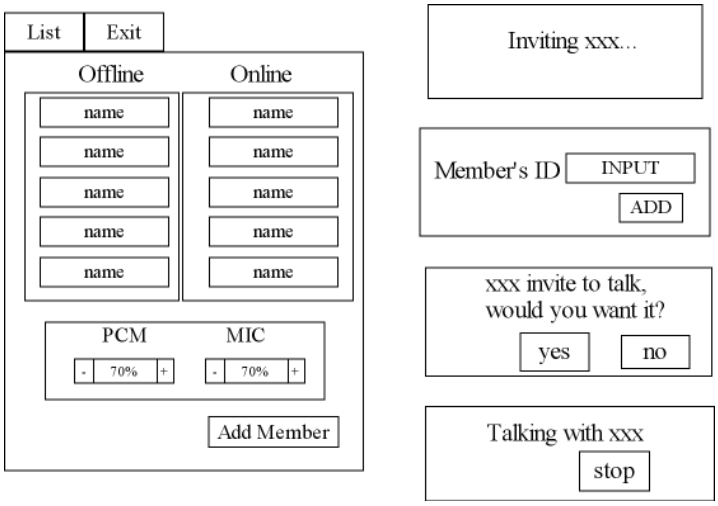

圖 27 左為名單介面 右為各項告知訊息框

1. Exit—登出，回登入畫面
2. Online—顯示線上好友名單，點選名單可進行邀請線上語音交談
3. Offline—離線好友名單
4. PCM—調整聲音輸出音量
5. MIC—調整麥克風音量
6. Add Member—開啟增加好友名單訊息框(告知訊息框之二)

三、Target Board 上的運作：

將建構的系統跟應用程式，放在 Target Board 的上，所需的 I/O，如聲音的收入\輸出、網路，以及 Touch screen 都能正常運作。

圖 29 為在 Target Board 上 GUI 的 Client 端實際運作畫面。我們可以透過 Touch Screen 點取軟鍵盤輸入使用者帳號來登入或建立帳號。

圖 28 板子上運作情形-開機畫面

圖 29 板子上運作情形-登入畫面

圖 30 為登入後，自動顯示線上名單、能夠新增好友以及調整音效裝置之音量。

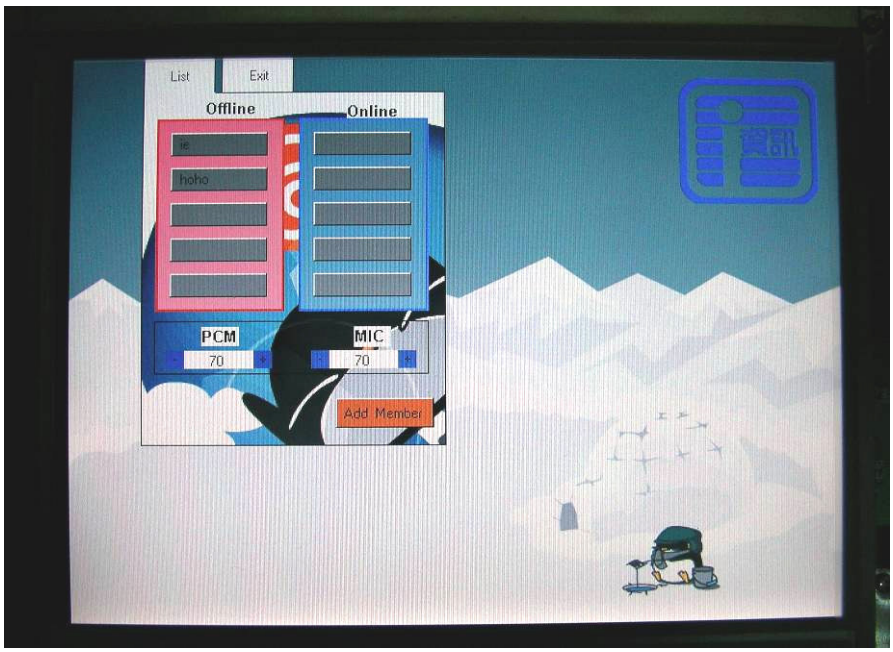

圖 30 板子上運作情形-線上名單

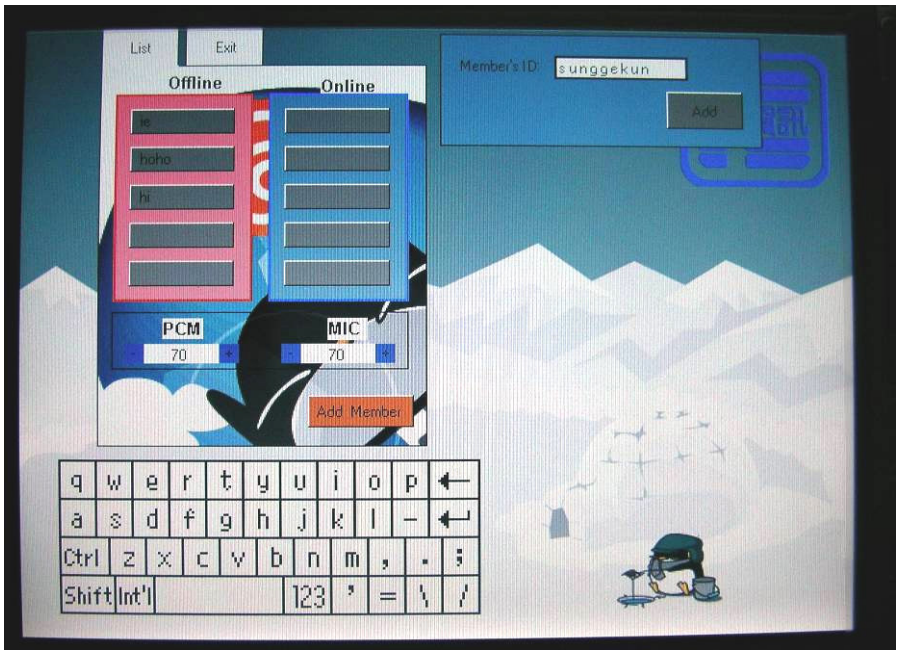

圖 31 板子上運作情形-新增好友名單

透過觀看線上名單，點選線上好友的帳號來開啟線上語音交談。圖 32 為被另一方使用者邀請之畫面，顯示邀請之訊息以及能夠答覆邀請。

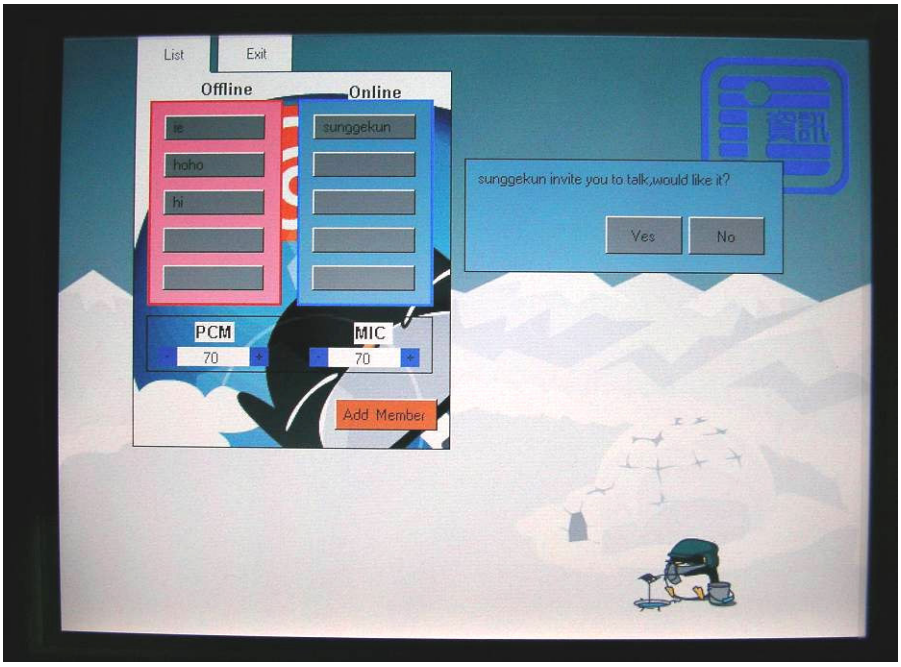

圖 32 板子上運作情形-被邀請

如圖 33 所示正式開啟線上交談後，顯示訊息並提供停止線上交談之功能。

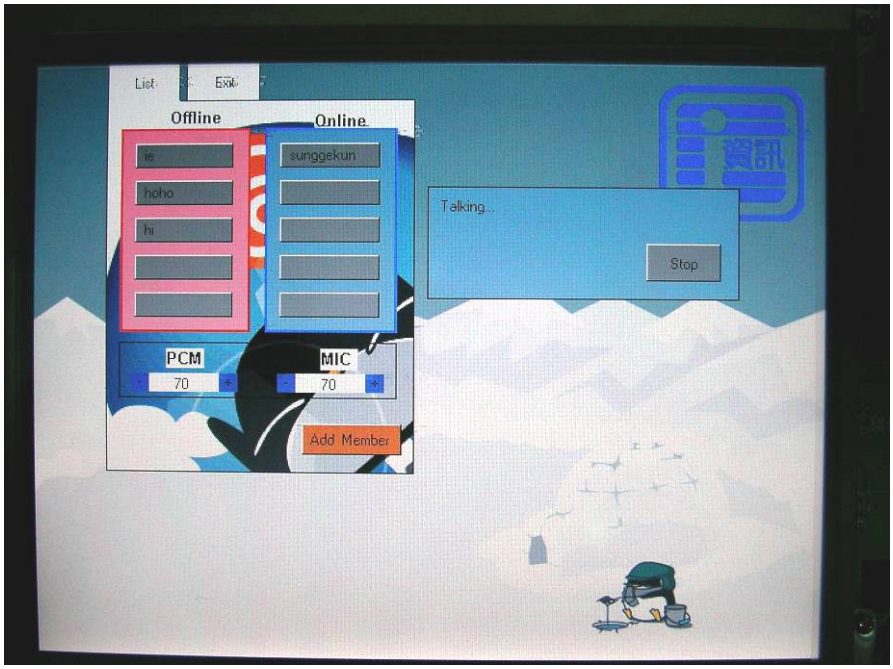

圖 33 板子上運作情形-線上交談中

如果想要聊天的好友不在線上，可經由點選離線名單留言給對方，如圖 34。

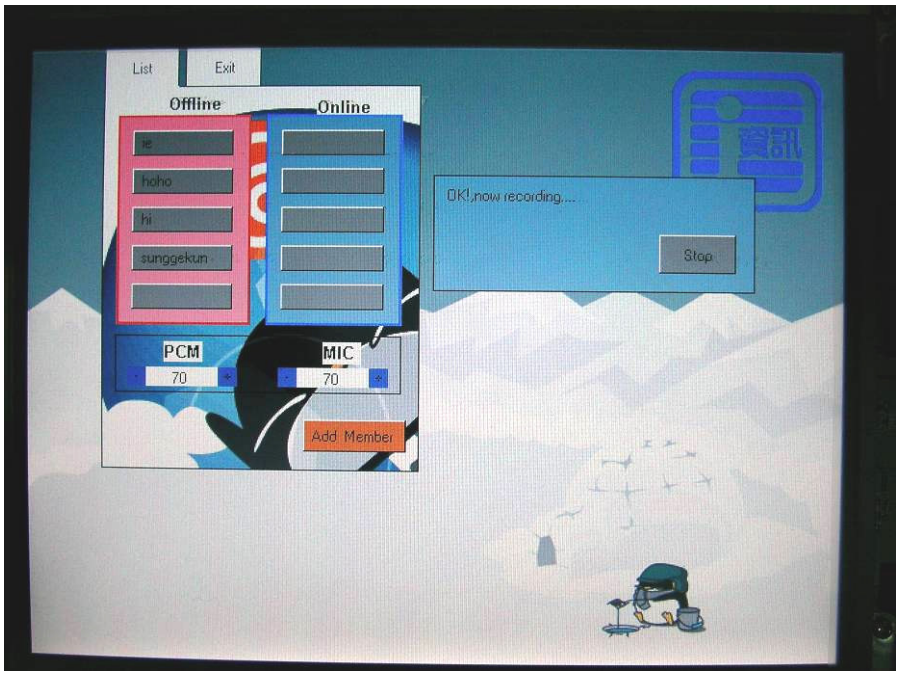

圖 34 板子上運作情形-留言

四、NAT & Private IP 問題方面：

為了證明先前所說的解決 NAT & Private IP 問題的小技巧能夠運作，使用我們實作的應用軟體進行測試，結果如下表所示。

表 10 在 NAT 機制下的測試

	User A	User B	聲音傳輸
狀況 1	Private IP	Public IP	Work
狀況 2	Private IP	Private IP	Work

User A： 為學校宿網

User B： 使用一般 ISP 的網路，Private IP 方面則是使用 VMware 來模擬

這兩種狀況幾乎可以代表一般 NAT 機制運作下的情況，測試結果都能夠傳輸，所以驗證先前所提的方法是可行的。

●從 x86 到 ARM 問題之探討

在此章節中將提到專題實作中，實際上遇到的觀念與技術上的問題，以及針對問題的解決方案。接著討論對未來想要更深入探討的議題，包含對未來發展方向展望、及在嵌入式領域一些重要的主題，如 RISC 和 CISC 的探討，程式設計及電源管理等重要的方向作基本的討論。

6.1 遭遇問題&解決方案

6.1.1 Touch Screen

Target Board 上的 touch screen 為 ADS7843， Microwindows 本身並未支援此 drivers，所以我們採取土法煉鋼的方式，用 Microwindows 支援且最相近的 drivers 來嘗試看看。

- ADS driver：

  如圖 35 所示首先採用 ADS driver，在點取 touch screen 後，顯示一堆錯誤訊息而 touch screen 無法使用，也就是說此 ADS driver 並不相似於 ADS7843 的 driver 。

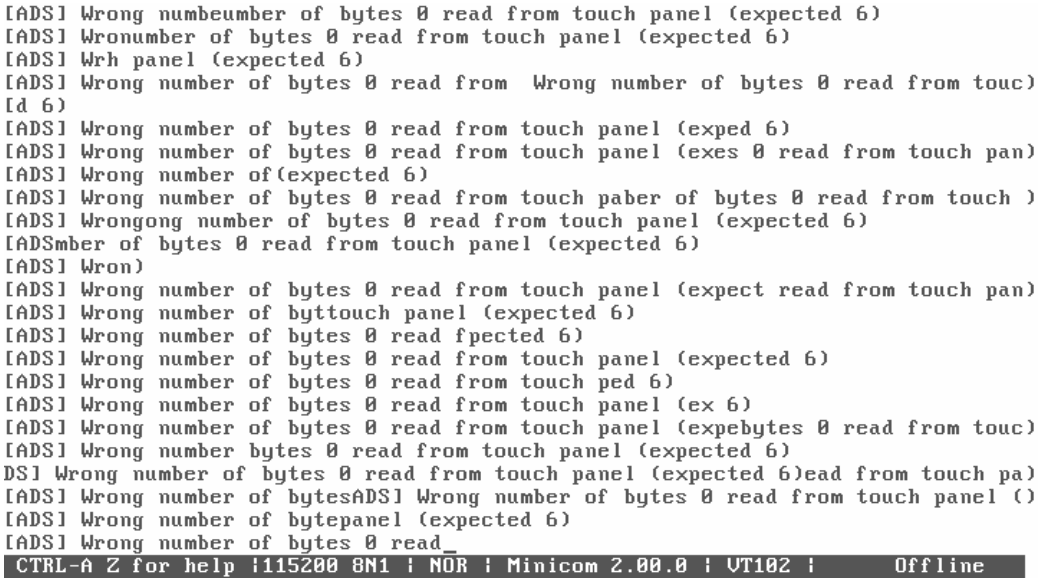

圖 35 使用 ADS driver 測試結果

- ADS7846 driver：

  接下來，嘗試使用 ADS7846 driver，如圖 36，結果只有一種錯誤，訊息告訴讀取到的只有 8 bytes，而非預期的 16 bytes。也就是說，修改 ADS7846 driver 相當有機會可以正常運作。於是開始研究其 driver 的 source code。

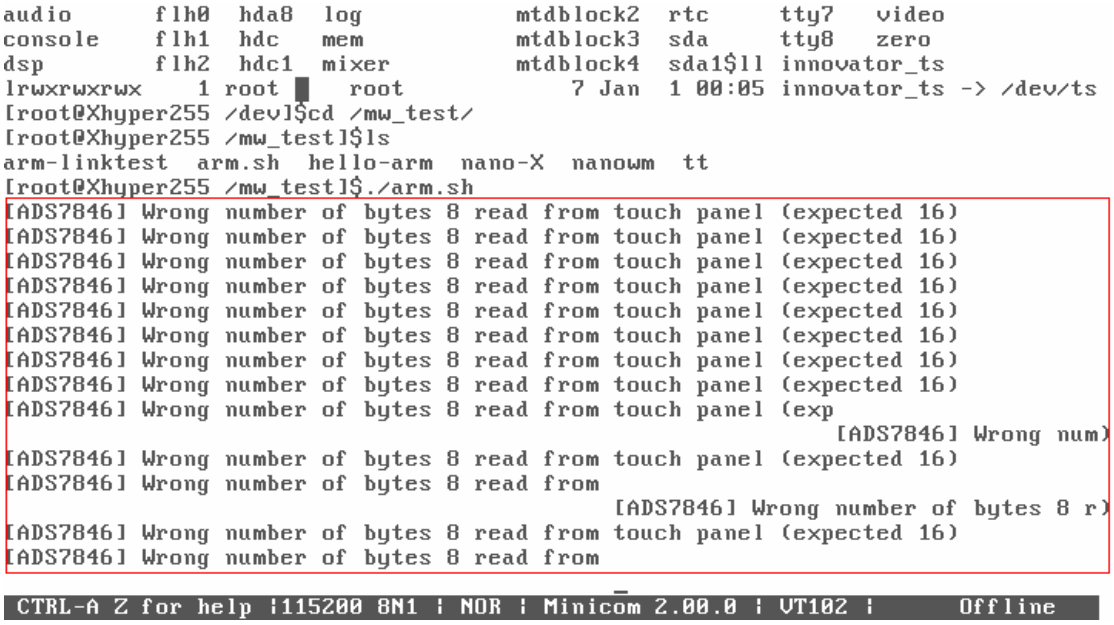

圖 36 使用 ADS7846 driver 測試結果

觀察結果，各 driver 間的差別只有在 struct ts_event(點取資訊)，下圖為 ADS7846 driver 的 ts_event。

(microwindows-0.90\src\drivers\touchscreen_ads7846.h)

```
struct ts_event {
    short pressure;
    short x;
    short y;
    short pad;
    struct timeval stamp;
};
```

圖 37 ADS7846 的 ts_event

前面 4 個變數的大小就已經符合點取 touch screen 讀取到的 8 bytes，而後面的 struct timeval stamp 在 Microwindows 所支援的 touch screen driver 裡面並非必須，所以我們嘗試將 struct timeval stamp 去除掉。很幸運地，去掉該變數後，touch screen 就能夠正常運作。

6.1.2 軟鍵盤

軟鍵盤為 Microwindows 裡的 demo 程式之一，我們把它放在 Target Board 用來當作輸入。但由於原本軟鍵盤大小為 160* 61，放在 Target Board 上畫面顯得有些過小，使用起來十分不方便，所以將其放大一倍(320*122)。

- 更動項目如下，變動畫面參考圖 38：
  1. 把軟鍵盤圖放大一倍。
  2. 修改程式碼(microwindows-0.90\src\demos\nxkbd\nxkbd.c)，並將對應軟鍵盤按鍵的數值做調整。

圖 38 左：原本大小 右：軟鍵盤放大一倍

6.1.3 FCU 無線網路之網頁認證

使用校內的無線網路，必須在認證網頁輸入使用者帳號和密碼，這是個不小的麻煩，因為我們的系統裡並不包含瀏覽器，首先觀察認證網頁的行為，從網頁原始碼可以知道，它使用 HTML 的 form 標籤來取得和設定參數值，再將這些參數傳送給 logon 這支程式處理，這裡使用了 CGI(Common Gateway Interface)的技術，client 從瀏覽器上傳遞參數給 web server 上的程式，這支程式便會依參數來回應 client，以達到 client 和 web server 之間的互動，CGI 所制訂的就是兩者之間傳遞的標準，CGI 程式並沒有規定要使用何種程式語言，所以我們也不知道 logon 是以何種程式語言撰寫而成，對 client 而言顯然也不重要。第二個麻煩是認證網頁使用 https(http secure)協定，簡單說是 http 的安全版，使用 SSL(Secure Sockets Layer)技術。

- 使用 wget 解決：

  和瀏覽器相比，wget 顯然較為輕巧，而且支援 https 協定。而利用 wget 是為了達成兩項重要的目標：

  1. 抓取網頁，從中找出所須資訊。
  2. 傳遞參數給 CGI 程式。

- 詳細過程如下：

首先用 wget 抓取認證網頁，搭配 bash script 從網頁原始檔中抓出參數名稱和參數值，表 11 所列之參數值是固定不變的，可以事先寫死在 script，真正重要的是表 12 中 secret 和 verify_vernier 之參數值，這是由 server 產生，效用只有一分鐘。其次是 username 和 password，分別是學校 knight 主機的 e-mail 和密碼，由使用者自行輸入。當取得所有參數名稱和參數值後，將它們和 logon 程式串在一起傳送，例如『wget　https://mobile.fcu.edu.tw/logon?參數=值&參數=值&參數=值』

最後我們僅是利用 wget 來傳遞參數，這時取得的網頁內容已經沒有特別的用途，web server 會將這串參數傳遞給 logon 這個 CGI 程式做處理，這時所有的網路功能都能使用了。至於登出認證，只須傳遞一個參數，請參見表 13。

表 11 登入所須參數一(固定)

name	value	name	value
javaworks	0	guest_allowed	1
vernier_id	VernierNetworks	realm_required	0
product_id	VNSS	logon_action	Logon User
releast_id	1.0	query_string	<原指定網址>
logon_status	0		

表 12 登入所須參數二(變動)

name	value	name	value
secret	<server 產生>	username	<client 輸入>
verify_vernier	<server 產生>	password	<client 輸入>

表 13 登出所須參數(固定)

name	value
logon_action	Logoff

6.1.4 音效全雙工

為了達成即時語音，聲音處理必須具備全雙工(full duplex)的能力，也就是可以同時進行錄音(input)與放音(output)，硬體和 driver 都必須支援，缺一不可。目前一般 PC 的音效卡幾乎都具備了這項能力，我們使用的目標板也有支援；不幸的是我們 PC 音效卡使用的 driver 卻不支援，Linux 2.4 預設使用的音效系統是 OSS，但這張音效卡的 OSS driver(i810_audio)並不支援全雙工，也沒有找到其它可替代的 OSS driver，於是我們採用了另一個音效系統 ALSA(Advanced Linux Sound Architecture)所提供的 driver(intel8x0)，ALSA driver 的功能較為齊全，不像 OSS 只有部份免費，最重要的是還有在維護，它目前已被加入至 Linux 2.6 裡。ALSA driver 也包含 OSS 模擬器，讓使用 OSS API 的程式也可以使用 ALSA driver，我們的 Voice Messenger 在音效處理方面就是使用 OSS API，因此這些問題都獲得解決。

- ALSA(Advanced Linux Sound Architecture)：

  http://www.alsa-project.org

- intel8x0：

  http://www.alsa-project.org/alsa-doc/doc-php/template.php?company=Intel&card=.&chip=440MX%2C+i810%2C+i810%2C+i810E%2C+i820%2C+i820&module=intel8x0 或縮短網址為 http://0rz.net/150Su

6.2 未來研究之方向

6.2.1 專題未來展望

- 以未來展望而言可分為以下幾點：

1. 硬體方面—雖然我們實作平台是 HyBus 的 X-Hyper255B-TKUIII，並不是所有的硬體設備都有用到，如果要進一步提升硬體，就要談到 SOC(system-on-chip)部分，也可以從 HDL 去下手，簡化及縮小硬體，接著去除不必要的硬體設備後，其實就跟一般的手機硬體用途很類似，但跟現在市面上的網路電話不一樣，而是可以開發成類似手機的外型，跟無線網路一起配合，讓未來的手機不是只有一種傳輸選擇，透過無線網路省下的通話費相當可觀。

   除了透過網路傳遞聲音外，其實現在流行的 VoIP 是與市話結合，跟傳統的通訊線路作整合，讓通訊無所不在，Skype 就是很好的例子，圖 39 的 skype USB 話機就是可以跟 Skype 用戶之間網路通訊外，還可撥打市話與手機，但要收通訊費，所以本專題也可以與市話結合，了解傳統通訊與網路通訊的差別，進而改善通話品質，在音質方面可以好好加強。

圖 39 skype USB 話機

2. 軟體方面—利用網路傳輸固然方便，而與市話結合使得更普及化，加上

本來就與網路結合，一些附加功能就可以加入，想必是未來手機發展的趨勢，但這一切都要看無線網路的推廣速度。

VoIP 現今所使用的通訊技術有兩大主流，分別是 H.323 及 MGCP。本專題要跟的上主流就必須要跟隨 H.323 或 MGCP 通訊協定，但 H.323 由於當初主要是設計給視訊會議來使用，在設計上較為繁瑣；而 MGCP 大部份的軟體控制皆在 Call Agent 伺服器端，對用戶端來說就簡化了許多，因此無法支援像 H.323 的點對點彈性架構。最有可能就是兩種協定都使用，因主程式有「P2P 架構傳遞聲音」可使用 H.323，也同時有「Client-Server 架構檢查線上使用者」可用 MGCP。

6.2.2 ARM 與 嵌入式系統探討

一、CISC&RISC

在嵌入式 Linux 系統開發過程，我們是使用 ARM 的實驗版，ARM 是 Advanced RISC Machine 的縮寫，RISC(Reduced Instruction Set Computer)是精簡指令集的電腦，現下開發嵌入式系統已經越來越多使用 RISC 架構來開發嵌入式系統，以前在開發嵌入式系統是以 CISC(Complex Instruction Set Computer；複雜指令集電腦)架構為設計主流，但是漸漸地改以 RISC 架構來開發，這當中值得探討的是從 CISC 到 RISC 轉變的趨勢。

和單純的硬體開發或軟體開發比較不同，嵌入式 Linux 系統在研發過程中通常都要涉及到硬體和軟體兩個層面。嵌入式系統的硬體體系架構必須有所了解，因為嵌入式系統開發又對硬體的要求非常高。所以我們可以探討 RISC 和 CISC 的差別如下：

(1) ***CISC：***

利用微程式化電腦的便利，將許多會使用到的複雜功能，作成電腦的基

礎指令，所以電腦內部的運作愈作愈複雜，提供的指令功能也越來越多，使程式設計的人在設計時減少許多負擔，是八零年代的主流。

(2) ***RISC：***

根據統計，電腦 80%的執行期間都只用到 20%的簡單指令，一旦設計太多指令功能可能會造成硬體不必要的負擔而造成速度的變慢，所以主張 CPU 只要提供基本的指令即可，雖然會造成程式設計人員設計的負擔，但是卻可以節省硬體設計成本並且加快執行速度，現今已經被廣泛的接受。

表 14 CISC 與 RISC 比較表

CISC v.s. RISC： 項目	**RISC**	**CISC**
指令數	100 以內	數百個
定址法	固定個數	變化多而複雜
資料暫存器數目	至少 32 個	十個以下
指令長度與格式	固定	長度不定，格式多變
指令週期	固定	長度不定，格式多變
控制邏輯(C.U.)	Hardwired	Microprogramming
程式所需指令數	較多	較少
架構優點	適於 Pipeline、Cache 等設計	程式設計較簡單、Code Size 小
運算形式	R-R Type Operation (Load-Store Machine)	Mixed Type Operation (R-R、R-M、M-M)
代表機種	Alpha、PowerPC	Pentium

可以發現兩者的是截然不同的設計，RISC 的設計指令雖然簡單，但是所帶

來的好處，其實是現下被廣泛所接受，從硬體設計的角度來看，硬體的設計有四大原則：

(1) 簡單且具有規律性

(2) 元件小速度快

(3) 好的設計需有好的折衷方法

(4) 使常見的情形變得更快

從硬體的設計的角度來看，RISC 的確佔有比較大的優勢，在相同的條件下，RISC 平均速度比 CISC 快 2 到 5 倍，目前的典型架構為 ARM 系列、和 MIPS 等，都是 RISC 架構

現在幾乎所有的半導體製造商都有能力生產嵌入式中央處理器，也幾乎都是 RISC 的架構，根據所用技術的不同，可以將嵌入式中央處理器分成如下四種類型：

(1) 嵌入式 Micro-Processor（Embedded Micro Processor Unit，EMPU）

(2) 嵌入式 Micro-Controller（Embedded Micro-controller Unit，EMU）

(3) 嵌入式 DSP 處理器（Embedded Digital Signal Processor，EDSP）

(4) 嵌入式單晶片系統（Embedded System on Chip，ESoC）

而我們針對我們研究的實驗版來說，是嵌入式 CPU 即 PXA-255，現在的電腦都是以 CPU 為中心。從意義上說，指的是晶片內部只包括 CPU 本身，而不包括 Storage 和 I/O 界面等其它功能模組區塊，採用的是 Von Neumann 體系架構。在一些高階的嵌入式應用中，透過將 Micro-Processor 裝配在專門設計的電路板上，並配以必要的外圍界面電路和擴展電路，可以快速構造出一個實用的嵌入式系統。

為了滿足嵌入式應用的特殊要求，嵌入式 Micro-Processor 雖然在功能上與

一般的 Micro-Processor 基本相同，但在工作電壓、工作溫度、抗干擾性和可靠性等方面都做了增強，一般說來，採用 Micro-processor 來構建嵌入式系統具有良好的即時性能、支持 Multi-tasking、便於儲存區保護、提供處理器擴展、豐富的測試功能和完善的電源管理等優勢。

二、電源管理

可攜式設備的多媒體功能近年得到快速發展，性能也不斷提升，但隨之而來的功耗問題對開發者帶來極大的挑戰。性能與功耗是成正比的，因此有效地進行電源的管理，盡可能來延長電池工作時間，為可攜式設備最主要的研究方向之一。以我們專題來看，要成為一個可攜式的商品，在電源方面必須考量到以下：

1. 待機時間起碼能連續工作 100 多小時。
2. 通話時間應多於 5 小時，這樣通話量較多的人可以使用一整個工作日。
3. 電池容量與重量、尺寸成正比，電池容量與價格也是成正比。

在可攜式的因素下，電池的重量和與尺寸會被侷限在一個範圍，這意味著電池容量會有個上限，長時間的運作將不容易達成。正因如此，必須進行電源的管理。以下舉三點管理方式：

1. 電源轉換必須盡可能高效地將電池功率轉換為系統元件可用的功率。
2. 電池管理必須能處理電池充電及電量測量。
3. 電池的使用進行最佳化，分析各裝置實際電源消耗並加以控制。

前兩點可透過選擇合適的電源管理元件來專門來達成，這是電路與硬體方面的問題。第三點主要與處理軟體的開發有關，透過軟體對於主要消耗能源的設備加以管理，例如：

1. 顯示方面可以在非使用中關閉。

2. CPU 在非使用中降低頻率。

3. 將要傳輸的資料壓縮，減少無線網路傳輸的時間。

電源管理是一門相當深奧的學問，牽扯的範圍相當廣，在這裡僅能概略的探討。

●專題的酸甜苦辣

完成一個專題必定經過一些不為人知的酸甜苦辣，從過程回顧與組員的心得來一探本專題的秘辛。

7.1 過程回顧

- *93 學年度第一學期：*

  主要為 Linux programming，雖然此時題目未定。，但以「Linux programming 程式設計教學手冊」書中，研讀「處理程序與信號」、「POSIX 執行緒」、「管線」、及「號誌、共享記憶體、訊息佇列」等章節去撰寫程式，學期末用 GTK+ 初次撰寫 GUI 介面。

- *93 學年度寒假：*

  主要為 Linux programming 和 socket programming。由於老師在 93 學年度第一學期末給了一個傳訊息的方向，實作出成果後，大家對於這方面還滿有興趣，也形成今日的專題主題開端。

- *93 學年度第二學期：*

  主要為 small Linux 與 microwindows GUI。開始朝嵌入式 IM 軟體前進，除了縮小與重新編譯 Linux 外，還研究 microwindows 實作 GUI，學期末合併了 small Linux 與 microwindows GUI，有了專題雛形。

- *93 學年度與 94 學年度間的暑假：*

  主要為研究 arm 目標板與 voice messenger。研究使用目標板就花了滿多時

間，但有李丕耀學長的幫助下，進度還算可以，而 voice messenger 也在此期間完成架構，主程式與 GUI 分開撰寫。

- *94 學年度第一學期：*

  主要為延續暑假內容、整合與書面報告整理。合併主程式與 GUI，voice messenger 初版成形，書面報告整理與撰寫，準備發表的相關事宜。

7.2 心得感想

7.2.1 柯以諾

回顧「專題研究」的過程，從 Linux programming、瞭解啟動程序、製作小系統，到在目標板上開發應用系統，這一連串的過程，讓我感到做一個系統實際上是很不容易的，即是說一個完整的系統，它所牽扯的領域相關廣泛，可以簡單分成硬體和軟體，軟體又可分系統軟體和應用軟體。網路方面，要考慮傳輸協定、流量、資料完整性與安全性。I/O 設備和聲音方面，要考慮格式是否符合、driver 和硬體是否支援。對於一個嵌入式系統，還要考慮到系統資源的貧乏，而剔除不必要的軟硬體元件。需要瞭解的東西實在太多。我們也不可避免的將焦點放在某些部份，剩餘的部份使用現成元件，過於進階的問題則暫時剔除。這次專題的經驗，趨使我想針對某個特定主題研究。

除了研究主題，「專題研究」另外帶來的兩大挑戰分別是“teamwork”和“報告”。雖然大學三年來經歷了多次不太正經的 teamwork，也寫了多篇不像樣的報告、投影片、甚至上台獻醜被教授砲轟，都比不上長達三學期悠關畢業的「專題研究」來得嚴謹，不可否認的，這的確也是「專題研究」這門課所要學習的重

點。

對這三大重點的成果，雖然我仍然覺得不滿意，但在長途跋涉中，多少也累積和吸收了各方面的知識與能力。最後，我相當感謝教援不厭其煩的指導，以及組員們對我的容忍，沒有因為不知所謂的嘴砲與龜毛，甚至是愚蠢至極的餿主意，而和我翻臉。

7.2.2 陳衍詳

在專題當中，所學習到的不只是在課本上僅有的知識，課本上所學往往是當成背景知識，實際上應用上要學的東西真的還有很多，能夠有機會能學學真正 Linux 作業系統，讓所學的不只是口頭上說說，瞭解在系統設計實作上很多人聰明微妙的想法與作法，從一開始 Linux Programming 慢慢到了瞭解系統內部，從 make kernel 一個晚上失敗了數十次，到漸漸熟悉，往往都花上大半天，真的很欽佩當初設計 Linux 的作者 Linus，不僅在硬體和軟體背景知識都非常熟悉，而在 Linux 的早期版本已經有大陸的學者實際 trace 早期版本而寫出詳解，並提供論壇可以供大家討論分享心得，雖然對於整個核心並不是很熟悉，希望在專題結束後能夠繼續的研究。

另外，要感謝同組的同學能夠在每次的討論中提供寶貴的意見，讓自己在學習上能夠學習的更快，加上因為要補習的緣故，使得大家討論的時間都要因為自己補習的緣故而有所調動，還有有時因為私事翹掉自己 meeting，對大家深感抱歉；同時，也要感謝老師及師長們在疑惑上的解答，省去了許多問題困擾的時間。最後希望在專題結束之後，能夠好好的 trace Linux kernel，核心裡面包含了許多有趣的主題，不僅僅是軟體、硬體，當中還有許多有趣的議題值得去探討，進而深入研究。

7.2.3 劉正強

此次專題個人是負責比較多在程式的撰寫上,在以前寫的程式都是以簡單的文字顯示為主,這次專題的應用程式則需要圖形化的使用者介面。由於 microwindows 並未像 Java 有那麼多包好的物件可以使用，所以許多的功能必須要自己撰寫出來，步驟相當的繁瑣，以往我們操作使用者介面感覺是一件很自然的事情而忽略它的難處，現在才體會到撰寫設計它真是一件不容易的事情。

經過了一年多的漫長專題研究,感謝組員的合作、學長的幫助、以及教授的指導。在這期間，學習 Linux programming、課堂以外的知識、如何自行尋找資料來解決問題的能力以及最重要的 teamwork。資訊發達的時代裡，想樣單打獨鬥很難在有限的時間完成工作，以後畢業必定都是團隊的合作，這次的專題不管是分配工作、撰寫報告、討論以及整合，都是一個很好的 teamwork 經驗。

另外比較可惜的是由於一開始的目標訂的太大，而整個專題不知該如何下手，導致有一段時間組員們漸漸地懶散起來，直到有了一個明確的目標後，才又開始專注於專題上，所以最後的成果可能不像一開始所想像的。但這是一個很好的經驗，以後不管是研究還是工作，先訂立幾個明確的目標後才開始動手，而不是定一個涉及層面很廣的方向。

7.2.4 蔣雯玲

在大學最重要的一門課之一莫過於專題，在此專題中我學到的很多，不僅是專業方面的知識，團隊合作與溝通都是專題學習的一部份。對於我主要負責的 microwindows 部份，在實作過程中慢慢地產生興趣，從無到有進而美化，但與主程式的配合就要一再的討論，這就顯示出溝通的重要性，組員之間算有向心力，也都不會感情用事去討論專題。只是對於團體討論方面，除了意見不合要解

決，還有大家的時間要配合，我們四個人算多的，每個人時間課表又不太相同，光是訂一個討論時間就不太容易，對於時間的管理就很重要，雖然時間不好訂，但是可以把事情排個優先順序，把專題擺優先就沒問題了。

作專題的感想就是專題是所有課程的整合，而且讓我們在畢業前瞭解如何去運作一個專題，未來可以用在出社會上，例如 OS、系統程式、資料結構、網路程式設計等等課程，都與此專題有關，有些還是修過以後給專題靈感。感謝一路走來都陪著我的指導老師與組員，過程中的酸甜苦辣只有我們知道，有成果出來使得辛苦有了代價。

附錄 A 目標板詳細規格

取於手冊 64 頁

Item	Description
Processor	Intel PXA255 400MHz
On-board FPGA	120M Gate(EP1C6240PQFP)
SDRAM	Samsung 64Mbyte (expand to 128M possible)
Flash	Intel strata flash 32Mbyte
Ethernet	CS8900A 10BaseT
Audio	AC'97 CS4299 Stereo audio
LCD Display	TFT LCD 6.4" (640 * 480)
Touch Screen	ADS7843 (Touch screen)
USB Host/Slave	SL811HS / USB Slave
RTC	RTC4513
PCMCIA	1 Slot
IrDA	HDSL3600
CF	1 Slot
MMC	1 Slot
GPIO Test Button	4EA (GPIO input)
GPIO Test LED	2EA (GPIO output)
Char LCD	2 x 16 Character LCD
TEST LED	8 EA
7 Segment	7 Segment – 8EA (FND)
Stepping Motor	Stepping Motor Interfac (L297)
Key Matrix	4 x 4 Key Matrix
ADC/DAC	ADC/DAC(DAC0800, ADC0804)
Dot Matrix	7 x 5 Dot Matrix
Interface	Ethernet 1 port, Serial 2 port, Emulator port JTAG port 120 pin Connector for GPIO, Address & Data Bus 60 pin FPGA Test, Mic, Speaker Jack

附錄 B 啟動程序詳細過程

一、Bootloader

- 執行步驟：

  1. 讀取 MBR 的 Kernel Loader (亦即是 lilo、grub、spfdisk 等等)開機資訊
  2. 載入 Kernel 的作業系統核心資訊

- 詳細內容：

由於個人電腦的系統在讀完 BIOS 之後，會先去讀取第一個開機硬碟的第一個磁區，就是 master boot record(MBR)，MBR 主要就是在記錄開機的資訊，也就是儲存 lilo 及 grub 的地方。讓硬體認識核心是 kernel loader 的主要功能。

「arch/i386/boot」下就是製作 Linux Bootloader 的文件。「head.S」文件提供了對 OSF PAL/1 的調用入口，它將被編譯後置於引導扇區(Sector)(註：ARC 的分區首扇區或 SRM 的磁盤 0 扇區)，得到控制後初始化一些數據結構，再將控制轉給「main.c」中的 start_kernel( )，start_kernel( )向控制台輸出一些提示，調用 pal_init( )初始化 PAL 代碼，調用 openboot( ) 打開引導設備(通過讀取 Firmware 環境)，調用 load( )將核心程式碼加載到 START_ADDR(「include/asm-i386/system.h」)，再將 Firmware 中的核心引導參數加載到 ZERO_PAGE(0)中，最後調用 runkernel( )將控制轉給 0x100000 的 kernel，bootloader 部分結束。

i386 系統中一般都有 BIOS 做最初的引導工作，那就是將四個主分區表中的第一個可引導分區的第一個扇區加載到實模式地址 0x7c00 上，然後將控制轉交給它。在「arch/i386/boot」目錄下，bootsect.S 是生成引導扇區的彙編源碼，它首先將自己拷貝到 0x90000 上，然後將緊接其後的 setup 部分(第二

扇區)拷貝到 0x90200，將真正的內核代碼拷貝到 0x100000。

以上這些拷貝動作都是以 bootsect.S、setup.S 以及 vmlinux 在磁盤上連續存放為前提的，也就是說，我們的 bzImage 文件或者 zImage 文件是按照 bootsect、setup、vmlinux 這樣的順序組織，並存放於始於引導分區的首扇區的連續磁盤扇區之中。

bootsect.S 完成加載動作後，就直接跳轉到 0x90200，這裡正是 setup.S 的程序入口，setup.S 的主要功能就是將系統參數(包括記憶體、磁盤等，由 BIOS 返回)拷貝到 0x90000-0x901FF 記憶體中，這個地方正是 bootsect.S 存放的地方，這時它將被系統參數覆蓋。以後這些參數將由保護模式下的代碼來讀取。除此之外，setup.S 還將 video.S 中的代碼包含進來，檢測和設置顯示器和顯示模式。最後，setup.S 將系統轉換到保護模式，並跳轉到 0x100000(對於 bzImage 格式的大內核是 0x100000，對於 zImage 格式的是 0x1000)的內核引導代碼，Bootloader 過程結束。

二、Kernel：

- 執行步驟：kernel 執行 init 程式並取得 run-level 資訊。

- 詳細說明：在 Linux 的系統下，通常開機的核心都擺在/boot 底下，因此，這個時候的 boot loader 就會到/boot 去尋找相關的核心。核心載入之後，由核心執行的第一個程式/sbin/init，而第一個目標當然就是確定主機是要以怎樣的情況登入，這時就必須要以/sbin/init 來載入/etc/inittab 的資訊。

1. Kernel 引導入口

在 i386 系統中，當內核以 bzImage 的形式壓縮，即大內核方式(__BIG_KERNEL__)壓縮時就需要預先處理 bootsect.S 和 setup.S，按照大核模式使用$(CPP)處理生成 bootsect.S 和 setup.S，然後再編

譯生成相應的.o 文件，並使用 "arch/i386/boot/compressed/build.c" 生成的 build 工具，將實際的 Kernel(未壓縮的，含 kernel 中的 head.S 代碼)與"arch/i386/boot/compressed"下的 head.S 和 misc.c 合成到一起，其中的 head.S 代替了"arch/i386/kernel/head.S"的位置，由 Bootloader 引導執行(startup_32 入口)，然後它呼叫 misc.c 中定義的 decompress_kernel( )函數，使用 "lib/inflate.c"中定義的 gunzip( )將內核解壓到 0x100000，再轉到其上執行"arch/i386/kernel/head.S"中的 startup_32 程式碼。

2. 核心數據結構初始化--Kernel 引導第一部分

start_kernel( )中呼叫了一系列初始化函數，以完成 kernel 本身的設置。這些動作有的是公共的，有的則是需要配置的才會執行的。在 start_kernel( )函數中：

- 輸出 Linux 版本信息[printk(linux_banner)]
- 設置與體系結構相關的環境[setup_arch( )]
- 頁表結構初始化[paging_init( )]
- 使用"arch/i386/kernel/entry.S"中的入口點設置系統自陷入口 [trap_init( )]
- 使用 alpha_mv 結構和 entry.S 入口初始化系統 IRQ[init_IRQ( )]
- 核心行程調度器初始化[包括初始化幾個預設的 Bottom-half，sched_init( )]
- 時間、定時器初始化[包括讀取 CMOS 時鐘、估測主頻、初始化定時器中斷等，time_init( )]
- 提取並分析核心啟動參數[從環境變量中讀取參數，設置相應標誌位等待處理，parse_options( )]
- 控制台初始化[為輸出信息而先於 PCI 初始化，console_init( )]

- 剖析器數據結構初始化[prof_buffer 和 prof_len 變量]
- 核心 Cache 初始化[描述 Cache 信息的 Cache，kmem_cache_init( )]
- 延遲校準[獲得時鐘 jiffies 與 CPU 主頻 ticks 的延遲，calibrate_delay( )]
- 記憶體初始化[設置記憶體上下界和頁表項初始值，mem_init( )]
- 建立外設初始化--內核引導第二部分 init( )函數作為核心 thread，首先鎖定內核(僅對 SMP 機器有效)，然後調用 do_basic_setup( )完成外設及其驅動程序的加載和初始化。過程如下：

  (1) 總線初始化[比如 pci_init( )]

  (2) 網絡初始化[初始化網絡數據結構，包括 sk_init( )、skb_init( )和 proto_init( )三部分，在 proto_init( )中，將調用 protocols 結構中包含的所有協議的初始化過程，sock_init( )]

  (3) 建立 bdflush 核心 thread[bdflush( )過程常駐核心空間，由核心喚醒來清理被寫過的記憶體緩衝區，當 bdflush( )由 kernel_thread( )啟動後，它將自己命名為 kflushd]

  (4) 建立 kupdate 核心 thread[kupdate( )過程常駐核心空間，由核心按時調度執行，將記憶體緩衝區中的信息更新到磁盤中，更新的內容包括超級塊和 inode 表]

  (5) 設置並啟動核心調頁 threadkswapd[為了防止 kswapd 啟動時將版本信息輸出到其他信息中間，核心線調用 kswapd_setup( )設置 kswapd 運行所要求的環境，然後再建立 kswapd 核心 thread]

  (6) 建立事件管理核心 thread[start_context_thread( )函數啟動

context_thread( )過程，並重命名為 keventd]

(7) 設備初始化[包括並口 parport_init( )、字元設備 chr_dev_init( )、塊設備 blk_dev_init( )、SCSI 設備 scsi_dev_init( )、網絡設備 net_dev_init( )、磁盤初始化及分區檢查等等，device_setup( )]

(8) 執行文件格式設置[binfmt_setup( )]

(9) 啟動任何使用__initcall 標識的函數[方便核心開發者添加啟動函數，do_initcalls( )]

(10) 文件系統初始化[filesystem_setup( )]

(11) 安裝 root 文件系統[mount_root( )]

至此 do_basic_setup( )函數返回 init( )，在釋放啟動記憶體段 [free_initmem( )]並給內核解鎖以後，init( )打開/dev/console 設備，重定向 stdin、stdout 和 stderr 到控制台，最後，搜索文件系統中的 init 程序[或者由 init=命令行參數指定的程序]，並使用 execve( ) 系統調用加載執行 init 程序。

init( )函數到此結束，內核的引導部分也到此結束了，這個由 start_kernel( )建立的第一個 thread 已經成為一個用戶模式下的進程了。此時系統中存在著六個運行實體：

(1) start_kernel( )本身所在的執行體，這其實是一個"手工"建立的 thread，它在建立了 init( ) thread 以後就進入 cpu_idle( )循環了，它不會在進程(thread)列表中出現

(2) initthread，由 start_kernel( )建立，當前處於用戶態，加載了 init 程序

(3) kflushd 核心 thread，由 initthread 建立，在核心態運行 bdflush( )函數

(4) kupdate 核心 thread，由 initthread 建立，在核心態運行 kupdate( )函數

(5) kswapd 核心 thread，由 initthread 建立，在核心態運行 kswapd( )函數

(6) keventd 核心 thread，由 initthread 建立，在核心態運行 context_thread( )函數和設置內部及通用 cache["slab_cache"，kmem_cache_sizes_init( )]

(12) 建立 uid taskcount SLAB cache["uid_cache"，uidcache_init( )]

(13) 建立文件 cache["files_cache"，filescache_init( )]

(14) 建立目錄 cache["dentry_cache"，dcache_init( )]

(15) 建立與虛存相關的 cache["vm_area_struct"，"mm_struct"，vma_init( )]

(16) 塊設備讀寫緩衝區初始化[同時建立"buffer_head"cache 用戶加速訪問，buffer_init( )]

(17) 建立頁 cache[記憶體頁 hash 表初始化，page_cache_init( )]

(18) 建立信號隊列 cache["signal_queue"，signals_init( )]

(19) 初始化記憶體 inode 表[node_init( )]

(20) 建立記憶體文件描述符表["filp_cache"，file_table_init( )]

(21) 檢查體系結構漏洞[對於 alpha，此函數為空，check_bugs( )]

(22) SMP 機器其餘 CPU(除當前引導 CPU)初始化[對於沒有配置 SMP 的內核，此函數為空，smp_init( )]

(23) 啟動 init 過程[建立第一個核心 thread，調用 init( )函數，原執行序列調用 cpu_idle( )等待調度，init( )]

至此 start_kernel( )結束，基本的核心環境已經建立起來了。

3. 外設初始化--Kernel 引導第二部分

init( )函數作為核心 thread，首先鎖定內核(僅對 SMP 機器有效)，然後調用 do_basic_setup( )完成外設及其驅動程序的加載和初始化。過程如下：

(1) 總線初始化[比如 pci_init( )]

(2) 網絡初始化[初始化網絡數據結構，包括 sk_init( )、skb_init( )和 proto_init( )三部分，在 proto_init( )中，將調用 protocols 結構中包含的所有協議的初始化過程，sock_init( )]

(3) 建立 bdflush 核心 thread[bdflush( )過程常駐核心空間，由核心喚醒來清理被寫過的記憶體緩衝區，當 bdflush( )由 kernel_thread( )啟動後，它將自己命名為 kflushd]

(4) 建立 kupdate 核心 thread[kupdate( )過程常駐核心空間，由核心按時調度執行，將記憶體緩衝區中的信息更新到磁盤中，更新的內容包括超級塊和 inode 表]

(5) 設置並啟動核心調頁 threadkswapd[為了防止 kswapd 啟動時將版本信息輸出到其他信息中間，核心線調用 kswapd_setup( )設置 kswapd 運行所要求的環境，然後再建立 kswapd 核心 thread]

(6) 建立事件管理核心 thread[start_context_thread( )函數啟動 context_thread( )過程，並重命名為 keventd]

(7) 設備初始化[包括並口 parport_init( )、字元設備 chr_dev_init( )、塊設備 blk_dev_init( )、SCSI 設備 scsi_dev_init( )、網絡設備 net_dev_init( )、磁盤初始化及分區檢查等等，device_setup( )]

(8) 執行文件格式設置[binfmt_setup( )]

(9) 啟動任何使用__initcall 標識的函數[方便核心開發者添

加啟動函數，do_initcalls( )]

(10) 文件系統初始化[filesystem_setup( )]

(11) 安裝 root 文件系統[mount_root( )]

至此 do_basic_setup( )函數返回 init( )，在釋放啟動記憶體段 [free_initmem( )]並給內核解鎖以後，init( )打開/dev/console 設備，重定向 stdin、stdout 和 stderr 到控制台，最後搜索文件系統中的 init 程序(或者由 init=命令行參數指定的程序)，並使用 execve( )系統調用加載執行 init 程序。

init( )函數到此結束，內核的引導部分也到此結束了，這個由 start_kernel( )建立的第一個 thread 已經成為一個用戶模式下的進程了。此時系統中存在著六個運行實體：

1. start_kernel( )本身所在的執行體，這其實是一個"手工"建立的 thread，它在建立了 init( )thread 以後就進入 cpu_idle( )循環了，它不會在進程[thread]列表中出現
2. initthread，由 start_kernel( )建立，當前處於用戶態，加載了 init 程序
3. kflushd 核心 thread，由 initthread 建立，在核心態運行 bdflush( )函數
4. kupdate 核心 thread，由 initthread 建立，在核心態運行 kupdate( )函數
5. kswapd 核心 thread，由 initthread 建立，在核心態運行 kswapd( )函數
6. keventd 核心 thread，由 initthread 建立，在核心態運行 context_thread( )函數

三、Init：

- 執行步驟：
  1. init 執行 /etc/rc.d/rc.sysinit 檔案
  2. 啟動核心的外掛式模組 (/etc/modules.conf)
  3. init 執行 run-level 的各個批次檔(Scripts)
  4. init 執行 /etc/rc.d/rc.local 檔案
  5. 執行 /bin/login 程式
  6. 登入之後開始以 Shell 控管主機
- 詳細說明：
  1. init 的第一個執行內容 /etc/rc.d/rc.sysinit，內容包括：設定預設路徑（ PATH ）、設定主機名稱、執行 /etc/sysconfig/network 所記錄的網路資訊、掛載 /proc 這個保存在記憶體當中的主機基本訊息、以及其他幾個 Linux 作業系統最基本的幾個資訊。
  2. 啟動核心的外掛式模組 (/etc/modules.conf)可以選擇使用模組的型態來進行驅動程式的載入，如果系統原本找不到的模組就可以在 /etc/modules.conf 寫入。
  3. init 執行 run-level 的各個 scripts，由於不同的 run-level 所需要載入的模組並不相同，所以系統為不同的 run-level 設定了一些批次檔(scripts)，而 run-level 早就在前面的時候以 /etc/inittab 當中取得。
  4. init 執行 /etc/rc.d/rc.local，cdrom 必須要載入兩個模組之後才能使用，分別是 modprobe cdrom， modprobe ide-cd 加入到 /etc/rc.d/rc.local 中。

附錄 C Inittab 的詳細內容

```
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author        Miquel van Smoorenburg,

#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not havenetworking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
##表示當前預設執行層級為 5(initdefault)；
id:5:initdefault:

##啟動時自動執行/etc/rc.d/rc.sysinit 腳本(sysinit)
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
```

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
##當執行層級為 5 時，以 5 為參數執行/etc/rc.d/rc 腳本，init 將等待其返回(wait)
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

##在啟動過程中允許按 CTRL-ALT-DELETE 重啟系統
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.    Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

##在 2、3、4、5 層級上以 ttyX 為參數執行/sbin/mingetty 程序，打開 ttyX 終端用
於用戶登錄，
##如果行程退出則再次執行 mingetty 程序(respawn)
```

```
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

##在 5 層級上執行 xdm 程序，提供 xdm 圖形方式登錄界面，並在退出時重新執行(respawn)
# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```

附錄 D 目錄與檔案結構

X-Hyper255B-TKUIII：

/	*linuxrc -> busybox*				
	bin/	busybox *(utility -> busybox)*			
	dev/	console dsp fb0 null mixer ram0 ram1 ram2 ts tty tty0 ttyS0			
	etc/	fstab inittab resolv.conf protocols			
		rc.d/	rc.sysinit		
	lib/	ld-2.3.2.so libc-2.3.2.so libpthread libnss_dns libnss_files			
		modules/	2.4.18-rmk7-pxa1-xhyper255/	pcmcia/	hermes.o orinoco.o orinoco_cs.o
				modules.dep	
	sbin/	*init -> ../bin/busybox*			
	usr/	vm_client			

		vmc.sh autowl			
		bin/	nano-X nanowm nxkbd		
	proc/	**(ram)**			
	tmp/	**(ram)**			
	var/	**(ram)**			

未來若有更新將可從以下網址取得。

- 內容列表：

  http://bbs.iecs.fcu.edu.tw/~ianstar/linux/94a/rootfs_list.doc

- Root File System(xhyper255B):

  http://bbs.iecs.fcu.edu.tw/~ianstar/linux/94a/rootfs_xhyper255b.tar.bz2

- Root File System(x86):

  http://bbs.iecs.fcu.edu.tw/~ianstar/linux/94a/rootfs_x86.tar.bz2

附錄 E Voice Messenger v1.0 程式碼

```
/* vm_client_net.c -*/

#include <pthread.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include "vm_client.h"
#include <arpa/inet.h>
#include <netdb.h>


#define    LINELEN        256
extern int Answer;
extern int Invite;
extern int Invited;
extern int Leave;   /*1=leave action 2=leaving a message 3=geting
                      message from server*/
extern int Cancel_voip_able;     /*1=set can cancel voip */
extern int Next_message; /*use to play next message*/
extern int errno;
extern char *Host,*Service;

char Myname[20];

pthread_t Voip_thread_send,Voip_thread_receive,Receive_thread;
int Server_socket; /*server socket*/
int Address;      /*to save another client's address*/
int Voip_port;
int Voip_socket;
short int IP_send; /*tell server ,ip had send*/
short int Get_ip;  /*to client that server had got ip*/
short int prepare_voip;
struct  sockaddr_in SIN;/* an Internet endpoint address  */

int audio_init();
int audio_exit();
int rec(unsigned char *data, int size);
int play(unsigned char *data,int size);
int connectTCP(const char *host,const char *service);
int connectUDP(const char *host,const char *service);
int passiveUDP(const char *service);
int Send_Command(char *mgs );
void *start_voip(void *arg);
void *voip_send(void *arg);
void *voip_receive(void *arg);
void *receive_message(void *arg);/*get message(commamd) from server*/
void *leave_message(void *arg);
void *get_message(void *arg); /*get voice message*/
void *send_still_alive(void *);
void cancel_thread(pthread_t);
void get_address(void);     /*get udp address and port*/
void cancel_voip(short int);
void cancel_leave_message(short int);
void play_message_file(void);/*play voice message file*/

int  check_list(int mode,char *id);
void set_list(char *);
void add_list(int mode,char *id);   /*mode 1= Online_list
2=Offline_list*/
void remove_list(int mode,char *id);/*mode 1= Online_list
2=Offline_list*/
void reflash_list(void);
void back_to_login(void); /*back to login windows*/
int  check_friend(char *);/*check is your friend?*/
int  add_friend(char *string);
void logout(short int);
void show_invite_window(char *);
void show_warning_message(char *string,int mode);
void close_warning_message(void);
int errexit(const char *format, ...);

/*------------------------------------------------------------------
 * Connect_server - connect to server,send name and password
 *------------------------------------------------------------------
 */
int Connect_server(const char *host, const char *service,char
                     *account,char *password,int mode)
{
    char temp[50],buf[LINELEN+1];
    int      res,count;    /* socket descriptor, read count*/

     Server_socket = connectTCP(host, service);/*connect server*/

    if(!mode)
      Send_Command("#login");
    else
      Send_Command("#create");

    usleep(100000);

    sprintf(temp,"%s %s",account,password);

    strcpy(Myname,account);

    /*send name and password to server*/
    Send_Command(temp);

    count=read(Server_socket, buf, LINELEN);
```

```
    buf[count]='\0';

    if(strcmp(buf,"#error")==0)
    {
     close(Server_socket);
     return 1;
     }

  res=pthread_create(&Receive_thread, NULL, receive_message,(void
                     *)Server_socket);
  if(res != 0)
  {
   printf("Thread create failed!!!\n");
   exit(1);
  }

    return 0;

}

/*---------------------------------------------------------------------
 * receive_message - get message or get ip address from server
 *---------------------------------------------------------------------
 */
void *receive_message(void *arg)
{
 char   buf[LINELEN+1],temp[50],*tok;
 int    count,res,sock;
 struct  in_addr add;
 pthread_t start_thread;
 pthread_t still_alive;

 res=pthread_setcancelstate(PTHREAD_CANCEL_ENABLE,NULL);
 if(res != 0)
  {printf("Thread setcancelstste failed!!!\n");
   exit(1);
    }

 res=pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS,NULL);
 if(res != 0)
  {printf("Thread setcanceltype failed!!!\n");
   exit(1);
    }

 sock=(int)arg;


 /*get list from server*/
    Send_Command("#who");
 /*set list*/
    count=read(sock, buf, LINELEN);
 buf[count]='\0';
 puts(buf);/*show string "#list"*/

 count=read(sock, buf, LINELEN);
 buf[count]='\0';
 puts(buf);
   set_list(buf);

   usleep(100000);

   Send_Command("#message");

   pthread_create(&still_alive, NULL,send_still_alive ,NULL);

/*get service from server*/
   while(1)
   { count=read(sock, buf, LINELEN);
     buf[count]='\0';
     printf("Get: ");
     puts(buf);

     if (count <= 0)
      {
       close_warning_message();
       show_warning_message("Server down!!!",1);
       logout(0);/*server down mode */
       }

     /*another client login*/
     else if(strncmp(buf,"#login",6)==0)
     {
      tok = strtok(buf," ");
      tok = strtok(NULL," ");

      if(check_friend(tok))
        {
        add_list(1,tok); /*add online list*/
        remove_list(2,tok);/*remove offline list*/
        reflash_list();
        }
      }
     /*another client logout*/
     else if(strncmp(buf,"#logout",7)==0)
     {
      tok = strtok(buf," ");
      tok = strtok(NULL," ");

      if(check_friend(tok))
       {
          remove_list(1,tok);/*remove online list*/
          add_list(2,tok); /*add offline list*/
```

```
            reflash_list();
          }
        }
      /*someone invite you to open voip*/
      else if(strncmp(buf,"#invite",7)==0)
       {/*who*/
      tok = strtok(buf," ");
      tok = strtok(NULL," ");
      sprintf(temp,"%s invite you to talk,would like it?",tok);

       /*start invition,get input(yes or no)*/
       Invited=1;

       if(Leave)/*while leaving message or geting message ,can't be
                 invited*/
        Answer=2;
       else
        show_invite_window(temp);

       while(!Answer); /*wait for input answer*/

       if(Answer==1)
        pthread_create(&start_thread, NULL,start_voip ,NULL);
        /*get ip*/
         else
          Invited=0;

           /*reset answer*/
           Answer=0;
        }
         /*anthoer client accept your invition*/
       else if(strcmp(buf,"#accept")==0 && Invite==1)
       {
          close_warning_message();
          prepare_voip=1;
          pthread_create(&start_thread, NULL,start_voip ,NULL);
       }
       /*anthoer client not  accept your invition*/
       else if(strcmp(buf,"#accept")!=0 && Invite==1 && (!prepare_voip) )

       {
          close_warning_message();
          show_warning_message(buf,1);
          Invite=0;
        }
          /*server had got ip message */
       else if(strcmp(buf,"#get_v_message")==0)
        {
          puts("get ip_send_message");
          IP_send=1;
        }
          /*server send anthoer client's ip to you */
      else if(strncmp(buf,"#ip",3)==0)
        {
            tok = strtok(buf," ");
            tok = strtok(NULL," ");

            Address=atoi(tok);         /*string covert to address*/
            add.s_addr=htonl(Address);  /*covert to local address
                                          format*/
            printf("Address %s\n",inet_ntoa(add));

            tok = strtok(NULL," ");
            Voip_port=atoi(tok);   /*string covert to port*/
            printf("Port %d\n",Voip_port);

            Get_ip=1;

        }
         /*server tell you to stop voip */
      else if(strcmp(buf,"#stop_voip")==0 && Cancel_voip_able)
          cancel_voip(1);
         /*server send online list to you */
      else if(strcmp(buf,"#list")==0)
        {
         printf("Update list\n");
         count=read(sock, buf, LINELEN);
         buf[count]='\0';
         puts(buf);
         set_list(buf);
         }
       /*friend's account had been found in server */
      else if(strcmp(buf,"#find")==0 && Leave==1)
       {
           close_warning_message();
           show_warning_message("OK!,now recording....",2);
           Leave=2;
            pthread_create(&Voip_thread_send, NULL,
                           leave_message ,NULL);
        }
         /*friend's account had been not found in server */
      else if(strcmp(buf,"#find")!=0 && Leave==1)
       {
          close_warning_message();
           show_warning_message(buf,1);
           Leave=0;
       }
         /*server tell you that get voice message */
      else if(strcmp(buf,"#get_message")==0)
        {
            show_invite_window("get message,would you like to get
                                it?");
```

```
291             Leave=3;
292             while(Leave!=4);
293
294             pthread_create(&Voip_thread_receive, NULL,
                               get_message ,NULL);
295         }
296
297
298     }
299
300     pthread_exit(NULL);
301
302 }
303
304 /*---------------------------------------------------------------
305  *  send_still_alive - let server know you still online(alive)
306  *---------------------------------------------------------------
307  */
308 void *send_still_alive(void *arg)
309 {
310    int state=1;
311
312  while(state)
313   {
314       state=Send_Command("#still_alive");
315       sleep(60);
316    }
317
318    logout(0);/*server down mode */
319 }
320
321 /*---------------------------------------------------------------
322  *  set_list - set online and offline list
323  *---------------------------------------------------------------
324  */
325 void set_list(char *input)
326 {
327    FILE *fp;
328    char buf[256],temp[50];
329    char *tok;
330
331   if(access("friend",R_OK)==0)
332    {
333     fp=fopen("friend","r");
334      while(fgets(buf,256,fp))
335       {
336      sscanf(buf,"%s\n",temp);
337      if(((check_list(2,temp))==-1) && ((check_list(1,temp))==-1))
338      {
339          printf("add offline list: %s\n",temp);
340          add_list(2,temp); /*add offline list*/
341          }
342       }
343     fclose(fp);
344
345     tok = strtok(input," ");
346     printf("1st token : %s \n",tok);
347
348   while(tok!=NULL)
349   {
350      if(check_friend(tok))
351      {
352       if(((check_list(1,tok))==-1))
353       {printf("add online list: %s\n",tok);
354         add_list(1,tok); /*add online list*/
355         remove_list(2,tok);/*remove offline list*/
356       }
357      }
358     tok = strtok(NULL," ");
359      }
360    }
361
362    reflash_list();
363 }
364
365 /*---------------------------------------------------------------
366  *  check_friend - is your friend? 1=yes ,0=no
367  *---------------------------------------------------------------
368  */
369 int check_friend(char *list)
370 {
371    FILE *fp;
372    char buf[256],temp[50];
373    int count=0;
374
375    if(access("friend",R_OK)==0)
376    {
377    fp=fopen("friend","r");
378     while(fgets(buf,256,fp))
379      {
380      sscanf(buf,"%s\n",temp);
381      if(strcmp(temp,list)==0)
382        {
383          fclose(fp);
384          return 1;
385        }
386       count++;
387      }
388     fclose(fp);
389    }
390
391    if(Maxlist==count)
```

```
392     return 1;
393    else
394     return 0;
395 }
396
397
398 /*---------------------------------------------------------------
399  *  add_friend - add friend to list
400  *---------------------------------------------------------------
401  */
402 int add_friend(char *string)
403 {
404   FILE *fp;
405   char buf[LINELEN+1];
406   strcpy(buf,string);
407
408    if(!(check_friend(buf)))
409    {
410     fp=fopen("friend","a");
411     fprintf(fp,"%s\n",buf);
412     fclose(fp);
413     return 0;
414    }
415
416      return 1;
417
418 }
419
420 /*---------------------------------------------------------------
421  *  Send_Command - send command(message) to server
422  *---------------------------------------------------------------
423  */
424 int Send_Command(char *mgs )
425 {
426    char buf[LINELEN+1];
427    int result;
428
429    strcpy(buf,mgs);
430     printf("Send: ");
431     puts(buf);
432     result=write(Server_socket, buf, strlen(buf));
433
434     if(result)
435      return 1;
436     else
437      return 0;
438 }
439
440 /*---------------------------------------------------------------
441  *  start_voip - start voip service
442  *---------------------------------------------------------------
443  */
444 void *start_voip(void *arg)
445 {
446    /*get another client's ip and port */
447    get_address();
448
449    audio_init();
450
451    pthread_create(&Voip_thread_send, NULL,voip_send ,NULL);
452    pthread_create(&Voip_thread_receive, NULL,voip_receive ,NULL);
453
454    Cancel_voip_able=1;
455    show_warning_message("Talking...",2);
456
457 }
458
459 /*---------------------------------------------------------------
460  *  get_address - get another client's ip and port
461  *---------------------------------------------------------------
462  */
463 void get_address(void)
464 {
465  char *service = "7777";  /* default service name       */
466  struct  hostent *phe;/* pointer to host information entry    */
467  struct  sockaddr_in sin;  * an Internet endpoint address  */
468  char buf[50];
469
470  /*connect server*/
471  Voip_socket = socket(AF_INET ,SOCK_DGRAM ,0);
472  sin.sin_family = AF_INET ;
473  sin.sin_port = htons((u_short)atoi(service));
474
475  if ( (phe = gethostbyname(Host)) )
476       memcpy(&sin.sin_addr, phe->h_addr, phe->h_length);
477  else if ( (sin.sin_addr.s_addr = inet_addr(Host)) == INADDR_NONE )
478       errexit("can't get \"%s\" host entry\n", Host);
479
480    /*send message to server,so server can get NAT's ip and port*/
481    sprintf(buf,"%s",Myname);
482    printf("sending...\n");
483
484    while(1)
485   {
486      sendto(Voip_socket,buf, sizeof(buf), 0, (struct sockaddr *)&sin,
             sizeof(sin));
487      usleep(50000);
488      if(IP_send)
489       break;
490      printf("IP_send=%d\n",IP_send);
491     }
492
```

```
 puts("waiting for ip");

 while(!Get_ip);

   Get_ip=0;
   IP_send=0;

 printf("Start VOIP\n");

}

/*---------------------------------------------------------------------
 *  voip_send - rec and send
 *---------------------------------------------------------------------
 */
void *voip_send(void *arg)
{
  unsigned char  buf[BUFSIZE];
  int res;

 res=pthread_setcancelstate(PTHREAD_CANCEL_ENABLE,NULL);
 if(res != 0)
  {printf("Thread setcancelstste failed!!!\n");
   exit(1);
    }

 res=pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS,NULL);
 if(res != 0)
  {printf("Thread setcanceltype failed!!!\n");
   exit(1);
    }

   SIN.sin_addr.s_addr=htonl(Address);/*covert to network address
                                        format*/
   SIN.sin_port=htons(Voip_port);

   while(1)
     {rec(buf, BUFSIZE);
    sendto(Voip_socket, buf, sizeof(buf), 0, (struct sockaddr *)&SIN,
           sizeof(SIN));
     }

    pthread_exit(NULL);

}

/*---------------------------------------------------------------------
 *  voip_receive - receive,play
 *---------------------------------------------------------------------
 */
void *voip_receive(void *arg)
{
   unsigned char  buf[BUFSIZE];
   int res,n,alen;                 /* from-address length
*/


 res=pthread_setcancelstate(PTHREAD_CANCEL_ENABLE,NULL);
 if(res != 0)
  {printf("Thread setcancelstste failed!!!\n");
   exit(1);
    }

 res=pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS,NULL);
 if(res != 0)
  {printf("Thread setcanceltype failed!!!\n");
   exit(1);
    }

   alen=sizeof(SIN);

   while(1)
    {
     n=recvfrom(Voip_socket, buf, sizeof(buf), 0, (struct sockaddr
                 *)&SIN, &alen);
     play(buf, n);
    }

    pthread_exit(NULL);

}

/*---------------------------------------------------------------------
 *  leave_message -
 *---------------------------------------------------------------------
 */
void *leave_message(void *arg)
{
 unsigned char  buf[BUFSIZE];
 char   *service = "9855";
 int res;

 res=pthread_setcancelstate(PTHREAD_CANCEL_ENABLE,NULL);
 if(res != 0)
  {printf("Thread setcancelstste failed!!!\n");
   exit(1);
    }

 res=pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS,NULL);
 if(res != 0)
  {printf("Thread setcanceltype failed!!!\n");
   exit(1);
```

```
    }

    audio_init();

    Voip_socket=connectUDP(Host,service);

    while(1)
      {
     rec(buf, BUFSIZE);
     write(Voip_socket, buf, BUFSIZE);
      }

    pthread_exit(NULL);
}

/*------------------------------------------------------------------
 *  get_message -
 *------------------------------------------------------------------
 */
void *get_message(void *arg)
{
  char *service = "8885";  /* default service name       */
  struct  hostent *phe;/* pointer to host information entry    */
  struct  sockaddr_in sin; /* an Internet endpoint address    */
  unsigned char buf[BUFSIZE];
  char temp[100],me[50],size[50];
   int alen,sock,n,sum;
   FILE *fp;

  /*connect server*/
  sock = socket(AF_INET ,SOCK_DGRAM ,0);
  sin.sin_family = AF_INET ;
  sin.sin_port = htons((u_short)atoi(service));

  if ( (phe = gethostbyname(Host)) )
       memcpy(&sin.sin_addr, phe->h_addr, phe->h_length);
  else if ( (sin.sin_addr.s_addr = inet_addr(Host)) == INADDR_NONE )
       errexit("can't get \"%s\" host entry\n", Host);

    /*send message to server,so server can get NAT's ip and port*/
    sprintf(temp,"%s",Myname);
    printf("sending...\n");

    while(1)
    {
     sendto(sock,temp, sizeof(temp), 0, (struct sockaddr *)&sin,
            sizeof(sin));
     usleep(5000);
     if(IP_send)
      break;
    }

  IP_send=0;

  alen=sizeof(sin);
  printf("get message\n");
  n=recvfrom(sock,temp, sizeof(temp), 0, (struct sockaddr
*)&sin,&alen);
  temp[n]='\0';

    show_warning_message("Downloading...",3);

  while(strcmp(temp,"#end")!=0)
  {
      sscanf(temp,"%s %s",me,size);

   printf("file size : %s\n",size);

   /*write message log*/
   fp=fopen("messagelog","a");
   fprintf(fp,"%s %s\n",me,size);
   fclose(fp);

   sum=0;

   fp=fopen(me,"a");/*get message*/
   while(strcmp(buf,"#message end")!=0 || (sum-BUFSIZE)== atoi(size))
   {
       n= recvfrom(sock,buf, sizeof(buf), 0, (struct sockaddr
                    *)&sin,&alen);
       fwrite (buf ,1, n, fp);
       sum=sum+n;

     }
     strcpy(buf,"");

     printf("actuality get : %d\n",(sum-800));
     fclose(fp);

      n=recvfrom(sock,temp, sizeof(temp), 0, (struct sockaddr
                    *)&sin,&alen);
     temp[n]='\0';
   }

   close_warning_message();
   puts("download ok");
    show_warning_message("download complete!!",1);

    close(sock);

   play_message_file();
```

```
    pthread_exit(NULL);
}

/*---------------------------------------------------------------------
 *  play_message_file -  message file from disk
 *---------------------------------------------------------------------
 */
void play_message_file(void)
{
 unsigned char buf[BUFSIZE];
 char temp[50],filename[50],size[50];
 int sleeptime,bps;
 FILE *fp,*fplog;

 audio_init();
 Leave=4;

 fplog=fopen("messagelog","r");

 while(fgets(temp,256,fplog))
 {
      sscanf(temp,"%s %s\n",filename,size);

      close_warning_message();
      show_warning_message(filename,3);

      puts(temp);

      fp=fopen(filename,"r");

      while(fread (buf,1,BUFSIZE, fp))
       play(buf,BUFSIZE);

      bps=(SPEED* SAMPLESIZE* (STEREO+1))/8;
      sleeptime= atoi(size)/bps;/*wait for sound I/O complete*/
      printf("Sleep time : %d\n",sleeptime);
      usleep(1000000*sleeptime);

      close_warning_message();
      show_warning_message("Next message",1);

    while(!Next_message);
    Next_message=0;

    puts("next file");

      fclose(fp);
      remove(filename);

   }

   puts("play over");

 fclose(fplog);
 remove("messagelog");

 Leave=0;
 close_warning_message();
 show_warning_message("No next message",1);

 audio_exit();
}

/*---------------------------------------------------------------------
 *  cancel_thread
 *---------------------------------------------------------------------
 */
void cancel_thread(pthread_t thread)
{
   int res;
 res=pthread_cancel(thread);
     if(res != 0)
    {printf("Thread cancel failed!!!\n");
     exit(1);
      }

}

/*---------------------------------------------------------------------
 *  cancel_voip
 *---------------------------------------------------------------------
 */
void cancel_voip(short int mode)
{
   /*stop voip*/
   if(mode)
    Send_Command("#stop_voip");

   close_warning_message();
   show_warning_message("Stop talking",1);

   cancel_thread(Voip_thread_send);
   cancel_thread(Voip_thread_receive);
   audio_exit();
   Cancel_voip_able=0;
    Invited=0;
   Invite=0;
   prepare_voip=0;
   close(Voip_socket);
   printf("voip stop\n");

}
```

```
/*---------------------------------------------------------------------
 *  cancel_leave_message
 *---------------------------------------------------------------------
 */
void cancel_leave_message(short int mode)
{
   /*stop leave a message*/
   if(mode)
    Send_Command("#stop_leave_message");

   close_warning_message();
   show_warning_message("Stop leave a message",1);

   cancel_thread(Voip_thread_send);
   audio_exit();
   Leave=0;
   close(Voip_socket);
   printf("message socket stop\n");

}

/*---------------------------------------------------------------------
 *  logout - mode = 0 server down,do not send message to server
 *---------------------------------------------------------------------
 */
void logout(short int mode )
{
  if(Cancel_voip_able)
   cancel_voip(mode);

  if(Leave==2)
   cancel_leave_message(mode);

  if(mode)
   Send_Command("#bye");

  cancel_thread(Receive_thread);

  close(Server_socket);

  back_to_login();

}
```

/* vm_client_gr.c*/

```
#include "nxcolors.h"
#include "vm_client.h"
#include <signal.h>

GR_WINDOW_ID    Tablew;
GR_WINDOW_ID    Loginw, EnterAw, EnterPw, Addw, EnterCw, Addbw;
GR_WINDOW_ID    Loginbw, Tool1w, Tool2w, Tool3w, Listw, Onlinew,
          Offlinew;
GR_WINDOW_ID   AddbAw, AddbMw, ViewIMw, ViewIMbw, Invitew, InvitebYw,
          InvitebNw;
GR_WINDOW_ID    PCMw, PCMupbw, PCMdownbw, MICw, MICupbw, MICdownbw;
GR_GC_ID   Gc, Gck, Gcw, Gcg;
GR_EVENT   Event;
GR_COORD   Begxpos;
GR_COORD   Xpos, Xposp;
GR_COORD   Ypos;
GR_IMAGE_INFO   Table_skin_info, Login_skin_info, List_skin_info;
GR_IMAGE_ID Table_skin_id, Login_skin_id, List_skin_id;

Member_list Online_list[Maxlist], Offline_list[Maxlist];

char  Account[20];        /*account*/
char  Password[20];       /*password*/
int   Count;              /*the number of inpute*/
int   Countp;             /*the number of password*/
short int  Answer;       /*been invited,do you want to
                        talt?,1=yes , 2=no*/
short int  Invite;        /*in the action of inviting? ,0=no 1=yes */
short int  Invited;       /*been invited situation?, 0=no ,1=yes*/
short int  Leave;        /*1= leaving a message*/
short int  Cancel_voip_able;   /*1=set can cancel voip */
int   Child;
int Next_message;
char  *Host,    *Service;

int Connect_server(const char *host, const char *service,char
*account,char *password,int mode);
void init_windows(void);
void do_keystroke(GR_EVENT_KEYSTROKE *kp);
void do_exposure(GR_EVENT_EXPOSURE  *ep);
void do_buttondown(GR_EVENT_BUTTON  *bp);
void do_buttonup(GR_EVENT_BUTTON    *bp);
void show_warning_message(char *string,int mode);
int  add_friend(char *string);
void back_to_login(void);

void Clean_windows(GR_WINDOW_ID wid);
void show_invite_window(char *);
void close_warning_message(void);
int  check_list(int mode,char *id);
void add_list(int mode,char *id);   /*mode 1= Online_list
                                2=Offline_list*/
void remove_list(int mode,char *id);/*mode 1= Online_list
```

```
                                   2=Offline_list*/
void clean_list(void);
void reflash_list(void);
void Send_Command(char *mgs );
void reflash_sound_state(int p,int m);
void cancel_voip(short int);
void cancel_leave_message(short int);
void logout(short int);


/*---------------------------------------------------------------------
 * main - main program
 *---------------------------------------------------------------------
 */
int main(int argc, char *argv[])
{
 Service="7750";

 switch (argc) {
 case 1:
      Host = "bbs.iecs.fuc.edu.tw";
      break;
 case 3:
      Service = argv[2];
      /* FALL THROUGH */
 case 2:
      Host = argv[1];
      break;
 default:
      fprintf(stderr, "usage: voipclient [host [port]]\n");
      exit(1);
 }

 init_windows();

 while(1)
 {
      GrGetNextEvent(&Event);
      switch(Event.type)
      {
           case GR_EVENT_TYPE_EXPOSURE :
                do_exposure(&Event.exposure);
                break;
           case GR_EVENT_TYPE_BUTTON_DOWN :
                do_buttondown(&Event.button);
                break;
           case GR_EVENT_TYPE_BUTTON_UP :
                do_buttonup(&Event.button);
                break;
           case GR_EVENT_TYPE_MOUSE_POSITION :
                break;
           case GR_EVENT_TYPE_KEY_DOWN :
                do_keystroke(&Event.keystroke);
                break;
           default:
                break;
      }
 }

 GrClose();
 return 0;
}

/*---------------------------------------------------------------------
 * init_windows - init microwindows
 *---------------------------------------------------------------------
 */
void init_windows(void)
{
 if(GrOpen() < 0)
 {
      fprintf (stderr, "GrOpen failed");
      exit (1);
 }

 Gc = GrNewGC();
 Gck = GrNewGC();
 Gcw = GrNewGC();
 Gcg = GrNewGC();
 GrSetGCUseBackground(Gc, GR_FALSE);
 GrSetGCForeground(Gc, GR_COLOR_BLACK);
 GrSetGCForeground(Gck, GR_COLOR_BLACK);
 GrSetGCBackground(Gck, GR_COLOR_WHITE);
 GrSetGCForeground(Gcw, GR_COLOR_WHITE);
 GrSetGCBackground(Gcw, GR_COLOR_WHITE);
 GrSetGCForeground(Gcg, GR_COLOR_SLATEBLUE);

 /*load Image*/
 Table_skin_id=GrLoadImageFromFile("./skin/table.gif", 0);
 GrGetImageInfo( Table_skin_id, &Table_skin_info);

 Login_skin_id=GrLoadImageFromFile("./skin/login.gif", 0);
 GrGetImageInfo( Login_skin_id, &Login_skin_info);

 List_skin_id=GrLoadImageFromFile("./skin/list.gif", 0);
 GrGetImageInfo( List_skin_id, &List_skin_info);

 /*create table window*/
 Tablew =  GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE
| GR_WM_PROPS_NODECORATE, NULL,GR_ROOT_WINDOW_ID, 0, 0, 640, 480,
GR_COLOR_LIGHTSKYBLUE);

```

```
151   /*create tool button*/
152   Tool1w =GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE |
153   GR_WM_PROPS_NODECORATE , NULL,GR_ROOT_WINDOW_ID, 60, 0, 60, 30,
154   GR_COLOR_WHITE);
155
156   Tool2w =GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE |
157   GR_WM_PROPS_NODECORATE, NULL,GR_ROOT_WINDOW_ID, 60, 0, 60, 30,
158   GR_COLOR_WHITE);
159
160   Tool3w =GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE |
161   GR_WM_PROPS_NODECORATE, NULL,GR_ROOT_WINDOW_ID, 125, 0, 60, 30,
162   GR_COLOR_WHITE);
163
164   /*create list window*/
165   Listw = GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE
166   | GR_WM_PROPS_NODECORATE, "List", GR_ROOT_WINDOW_ID, 60, 30, 250,
167   300,GR_COLOR_WHITE);
168
169   Offlinew = GrNewWindow(Listw, 14, 25, 102, 160, 2, GR_COLOR_PINK,
170 GR_COLOR_RED);
171   Offline_list[0].Wid = GrNewWindow(Offlinew, 10, 10, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
172   Offline_list[1].Wid = GrNewWindow(Offlinew, 10, 40, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
173   Offline_list[2].Wid = GrNewWindow(Offlinew, 10, 70, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
174   Offline_list[3].Wid = GrNewWindow(Offlinew, 10, 100, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
175   Offline_list[4].Wid = GrNewWindow(Offlinew, 10, 130, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
176
177   Onlinew = GrNewWindow(Listw, 132, 25, 102, 160, 2,
                      GR_COLOR_LIGHTSTEELBLUE, GR_COLOR_BLUE);
178   Online_list[0].Wid = GrNewWindow(Onlinew, 10, 10, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
179   Online_list[1].Wid = GrNewWindow(Onlinew, 10, 40, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
180   Online_list[2].Wid = GrNewWindow(Onlinew, 10, 70, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
181   Online_list[3].Wid = GrNewWindow(Onlinew, 10, 100, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
182   Online_list[4].Wid = GrNewWindow(Onlinew, 10, 130, 80, 20, 0,
                                  GR_COLOR_GREY, 0);
183
184   AddbMw = GrNewWindow(Listw, 160, 260, 80, 25, 0, GR_COLOR_ORANGE2,
                    0);
185
186   PCMw = GrNewWindow(Listw, 36, 220, 70, 15, 0, GR_COLOR_WHITE, 0);
187   PCMdownbw = GrNewWindow(Listw, 21, 220, 15, 15, 0, GR_COLOR_SLATEBLUE,
                         0);
188   PCMupbw = GrNewWindow(Listw, 93, 220, 15, 15, 0, GR_COLOR_SLATEBLUE,
                                  0);
189
190   MICw = GrNewWindow(Listw, 155, 220, 70, 15, 0, GR_COLOR_WHITE, 0);
191    MICdownbw = GrNewWindow(Listw, 140, 220, 15, 15, 0,
                                  GR_COLOR_SLATEBLUE, 0);
192   MICupbw = GrNewWindow(Listw, 211, 220, 15, 15, 0, GR_COLOR_SLATEBLUE,
                       0);
193
194   /*create login window*/
195   Loginw = GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE
196   | GR_WM_PROPS_NODECORATE, "Login", GR_ROOT_WINDOW_ID, 60, 30, 250,
    300,GR_COLOR_WHITE);
197
198   Loginbw = GrNewWindow(Loginw,100, 170, 60, 30, 0,
                                  GR_COLOR_LIGHTSALMON, 0);
199   AddbAw = GrNewWindow(Loginw, 159, 260, 80, 25, 0,
                GR_COLOR_MEDIUMSEAGREEN, 0);
200
201   EnterAw = GrNewWindow(Loginw, 120, 70, 100, 15, 2, GR_COLOR_WHITE,
                                  GR_COLOR_BLACK);
202   EnterPw = GrNewWindow(Loginw, 120, 110, 100, 15, 2, GR_COLOR_WHITE,
                                  GR_COLOR_BLACK);
203
204   /*create IM window*/
205   ViewIMw = GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE
206   | GR_WM_PROPS_NODECORATE, "View", GR_ROOT_WINDOW_ID, 320,100 , 250,
207   90, GR_COLOR_LIGHTSTEELBLUE);
208
209   ViewIMbw = GrNewWindow(ViewIMw, 175, 45, 60, 30, 0, GR_COLOR_GRAY,
                                  0);
210
211   /*create Add_Member window*/
212   Addw = GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE |
213   GR_WM_PROPS_NODECORATE, "Add", GR_ROOT_WINDOW_ID, 320, 5, 250, 90,
214   GR_COLOR_LIGHTSTEELBLUE);
215
216   EnterCw = GrNewWindow(Addw, 90, 19, 100, 15, 2, GR_COLOR_WHITE,
                                  GR_COLOR_BLACK);
217
218   Addbw = GrNewWindow(Addw, 175, 45, 60, 30, 0, GR_COLOR_GRAY, 0);
219
220   /*create Invite window*/
221   Invitew = GrNewWindowEx(GR_WM_PROPS_NOAUTOMOVE | GR_WM_PROPS_NOMOVE
222   | GR_WM_PROPS_NODECORATE , "Invitew", GR_ROOT_WINDOW_ID, 320, 100,
223   250,90, GR_COLOR_LIGHTSTEELBLUE);
224
225   InvitebYw = GrNewWindow(Invitew, 110, 45, 60, 30, 0, GR_COLOR_GRAY,
0);
226   InvitebNw = GrNewWindow(Invitew, 175, 45, 60, 30, 0, GR_COLOR_GRAY,
0);
227
```

```
228
229   /*select events for every windows*/
230   GrSelectEvents(Tablew, GR_EVENT_MASK_EXPOSURE );
231   GrSelectEvents(Loginw, GR_EVENT_MASK_EXPOSURE );
232   GrSelectEvents(Listw, GR_EVENT_MASK_EXPOSURE );
233   GrSelectEvents(Offlinew, GR_EVENT_MASK_EXPOSURE );
234   GrSelectEvents(Onlinew, GR_EVENT_MASK_EXPOSURE );
235   GrSelectEvents(ViewIMw, GR_EVENT_MASK_EXPOSURE );
236   GrSelectEvents(Addw, GR_EVENT_MASK_EXPOSURE );
237   GrSelectEvents(Invitew, GR_EVENT_MASK_EXPOSURE );
238   GrSelectEvents(PCMw, GR_EVENT_MASK_EXPOSURE );
239   GrSelectEvents(MICw, GR_EVENT_MASK_EXPOSURE );
240
241   GrSelectEvents(ViewIMbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
242   GrSelectEvents(Addbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
243   GrSelectEvents(Loginbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
244   GrSelectEvents(InvitebYw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
245   GrSelectEvents(InvitebNw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
246
247   GrSelectEvents(AddbAw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
248   GrSelectEvents(AddbMw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
249   GrSelectEvents(PCMupbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
250   GrSelectEvents(PCMdownbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
251   GrSelectEvents(MICupbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
252   GrSelectEvents(MICdownbw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
253
254   GrSelectEvents(Offline_list[0].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN | GR_EVENT_MASK_BUTTON_UP);
255   GrSelectEvents(Offline_list[1].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
256   GrSelectEvents(Offline_list[2].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
257   GrSelectEvents(Offline_list[3].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
258   GrSelectEvents(Offline_list[4].Wid, GR_EVENT_MASK_EXPOSURE |
GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
259
260   GrSelectEvents(Online_list[0].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
261   GrSelectEvents(Online_list[1].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
262   GrSelectEvents(Online_list[2].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
263   GrSelectEvents(Online_list[3].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
264   GrSelectEvents(Online_list[4].Wid, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
265
266   GrSelectEvents(EnterAw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN | GR_EVENT_MASK_BUTTON_UP |
    GR_EVENT_MASK_MOUSE_POSITION | GR_EVENT_MASK_KEY_DOWN
267         );
268   GrSelectEvents(EnterPw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP |
    GR_EVENT_MASK_MOUSE_POSITION | GR_EVENT_MASK_KEY_DOWN
269         );
270   GrSelectEvents(EnterCw, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP |
    GR_EVENT_MASK_MOUSE_POSITION | GR_EVENT_MASK_KEY_DOWN
271   );
272
273   GrSelectEvents(Tool1w, GR_EVENT_MASK_EXPOSURE);
274   GrSelectEvents(Tool2w, GR_EVENT_MASK_EXPOSURE);
275   GrSelectEvents(Tool3w, GR_EVENT_MASK_EXPOSURE |
    GR_EVENT_MASK_BUTTON_DOWN| GR_EVENT_MASK_BUTTON_UP);
276
277   GrMapWindow(Tablew);
278   GrMapWindow(Loginw);
279   GrMapWindow(Loginbw);
280   GrMapWindow(AddbAw);
281   GrMapWindow(EnterAw);
282   GrMapWindow(EnterPw);
283   GrMapWindow(Tool1w);
284
285
286         Begxpos= 1;
287         Xpos=Xposp= 1;
288         Ypos= 1;
289         Count= 0;
290
291       /*open soft keyborad*/
292         Child=fork();
293          if(Child==0)
294           execv("./bin/nxkbd",NULL);
295
296 }
297
298 /*-----------------------------------------------------------------
299  * do_exposure - init windows
300  *-----------------------------------------------------------------
301  */
```

```
void do_exposure(GR_EVENT_EXPOSURE  *ep)
{
 if(ep->wid == Tablew)
 {
      GrDrawImageToFit( Tablew, Gc, 0, 0, Table_skin_info.width,
                        Table_skin_info.height, Table_skin_id);
 }
 if(ep->wid == Loginw)
 {
      GrText(Loginw, Gc, 60, 80, "Account:", -1 , GR_TFASCII);
      GrText(Loginw, Gc, 60, 120, "Password:", -1, GR_TFASCII);

 GrDrawImageToFit(Loginw,Gc,0,0,Login_skin_info.width,Login_skin_i
                  nfo.height,Login_skin_id);

      GrLine(Loginw, Gc, 60, 0, 249, 0);
      GrLine(Loginw, Gc, 0, 0, 0, 299);
      GrLine(Loginw, Gc, 249, 0, 249, 299);
      GrLine(Loginw, Gc, 0, 299, 249, 299);
 }

 if(ep->wid == Listw)
 {
      GrText(Listw, Gc, 50, 20, "offline", -1 , GR_TFASCII);
      GrText(Listw, Gc, 170, 20, "online", -1 , GR_TFASCII);
      GrText(Listw, Gc, 52, 210, "PCM", -1 , GR_TFASCII);
      GrText(Listw, Gc, 172, 210, "MIC", -1 , GR_TFASCII);


 GrDrawImageToFit(Listw,Gc,0,0,List_skin_info.width,List_skin_info
            .height,List_skin_id);

      GrRect(Listw, Gc, 11, 195, 225, 45);
      GrLine(Listw, Gc, 60, 0, 249, 0);
      GrLine(Listw, Gc, 0, 0, 0, 299);
      GrLine(Listw, Gc, 249, 0, 249, 299);
      GrLine(Listw, Gc, 0, 299, 249, 299);
 }
 if(ep->wid == ViewIMw)
 {
      GrLine(ViewIMw, Gc, 0, 0, 249, 0);
      GrLine(ViewIMw, Gc, 249, 0, 249, 89);
      GrLine(ViewIMw, Gc, 0, 0, 0, 89);
      GrLine(ViewIMw, Gc, 0, 89, 249, 89);
 }
 if(ep->wid == ViewIMbw)
 {
      //GrText(ViewIMbw, Gc, 19, 20, "Stop",-1, GR_TFASCII);
      GrLine(ViewIMbw, Gcw, 0, 0, 59, 0);
      GrLine(ViewIMbw, Gcw, 0, 0, 0, 29);
      GrLine(ViewIMbw, Gc, 59, 0, 59, 29);
      GrLine(ViewIMbw, Gc, 0, 29, 59, 29);
 }
 if(ep->wid == Loginbw)
 {
      GrText(Loginbw, Gc, 14, 20, "LOGIN",-1, GR_TFASCII);
      GrLine(Loginbw, Gcw, 0, 0, 59, 0);
      GrLine(Loginbw, Gcw, 0, 0, 0, 29);
      GrLine(Loginbw, Gc, 59, 0, 59, 29);
      GrLine(Loginbw, Gc, 0, 29, 59, 29);
 }
 if(ep->wid == AddbAw)
 {
      GrText(AddbAw, Gc, 7, 17, "Add  Account",-1, GR_TFASCII);
      GrLine(AddbAw, Gcw, 0, 0, 79, 0);
      GrLine(AddbAw, Gcw, 0, 0, 0, 24);
      GrLine(AddbAw, Gc, 79, 0, 79, 24);
      GrLine(AddbAw, Gc, 0, 24, 79, 24);
 }
 if(ep->wid == Invitew)
 {
      GrLine(Invitew, Gc, 0, 0, 249, 0);
      GrLine(Invitew, Gc, 249, 0, 249, 89);
      GrLine(Invitew, Gc, 0, 0, 0, 89);
      GrLine(Invitew, Gc, 0, 89, 249, 89);
 }
 if(ep->wid == InvitebYw)
 {
      GrText(InvitebYw, Gc, 20, 20, "Yes",-1, GR_TFASCII);
      GrLine(InvitebYw, Gcw, 0, 0, 59, 0);
      GrLine(InvitebYw, Gcw, 0, 0, 0, 29);
      GrLine(InvitebYw, Gc, 59, 0, 59, 29);
      GrLine(InvitebYw, Gc, 0, 29, 59, 29);
 }
 if(ep->wid == InvitebNw)
 {
      GrText(InvitebNw, Gc, 22, 20, "No",-1, GR_TFASCII);
      GrLine(InvitebNw, Gcw, 0, 0, 59, 0);
      GrLine(InvitebNw, Gcw, 0, 0, 0, 29);
      GrLine(InvitebNw, Gc, 59, 0, 59, 29);
      GrLine(InvitebNw, Gc, 0, 29, 59, 29);
 }
   if(ep->wid == Offline_list[0].Wid)
       {
              GrLine(Offline_list[0].Wid, Gcw, 0, 0, 79, 0);
              GrLine(Offline_list[0].Wid, Gcw, 0, 0, 0, 19);
              GrLine(Offline_list[0].Wid, Gc, 79, 0, 79, 19);
              GrLine(Offline_list[0].Wid, Gc, 0, 19, 79, 19);
       }
 if(ep->wid == Offline_list[1].Wid)
       {
              GrLine(Offline_list[1].Wid, Gcw, 0, 0, 79, 0);
```

```
402               GrLine(Offline_list[1].Wid, Gcw, 0, 0, 0, 19);
403               GrLine(Offline_list[1].Wid, Gc, 79, 0, 79, 19);
404               GrLine(Offline_list[1].Wid, Gc, 0, 19, 79, 19);
405         }
406   if(ep->wid == Offline_list[2].Wid)
407         {
408               GrLine(Offline_list[2].Wid, Gcw, 0, 0, 79, 0);
409               GrLine(Offline_list[2].Wid, Gcw, 0, 0, 0, 19);
410               GrLine(Offline_list[2].Wid, Gc, 79, 0, 79, 19);
411               GrLine(Offline_list[2].Wid, Gc, 0, 19, 79, 19);
412         }
413   if(ep->wid == Offline_list[3].Wid)
414         {
415               GrLine(Offline_list[3].Wid, Gcw, 0, 0, 79, 0);
416               GrLine(Offline_list[3].Wid, Gcw, 0, 0, 0, 19);
417               GrLine(Offline_list[3].Wid, Gc, 79, 0, 79, 19);
418               GrLine(Offline_list[3].Wid, Gc, 0, 19, 79, 19);
419         }
420   if(ep->wid == Offline_list[4].Wid)
421         {
422               GrLine(Offline_list[4].Wid, Gcw, 0, 0, 79, 0);
423               GrLine(Offline_list[4].Wid, Gcw, 0, 0, 0, 19);
424               GrLine(Offline_list[4].Wid, Gc, 79, 0, 79, 19);
425               GrLine(Offline_list[4].Wid, Gc, 0, 19, 79, 19);
426         }
427   if(ep->wid == Online_list[0].Wid)
428         {
429               GrLine(Online_list[0].Wid, Gcw, 0, 0, 79, 0);
430               GrLine(Online_list[0].Wid, Gcw, 0, 0, 0, 19);
431               GrLine(Online_list[0].Wid, Gc, 79, 0, 79, 19);
432               GrLine(Online_list[0].Wid, Gc, 0, 19, 79, 19);
433         }
434   if(ep->wid == Online_list[1].Wid)
435         {
436               GrLine(Online_list[1].Wid, Gcw, 0, 0, 79, 0);
437               GrLine(Online_list[1].Wid, Gcw, 0, 0, 0, 19);
438               GrLine(Online_list[1].Wid, Gc, 79, 0, 79, 19);
439               GrLine(Online_list[1].Wid, Gc, 0, 19, 79, 19);
440         }
441   if(ep->wid == Online_list[2].Wid)
442         {
443               GrLine(Online_list[2].Wid, Gcw, 0, 0, 79, 0);
444               GrLine(Online_list[2].Wid, Gcw, 0, 0, 0, 19);
445               GrLine(Online_list[2].Wid, Gc, 79, 0, 79, 19);
446               GrLine(Online_list[2].Wid, Gc, 0, 19, 79, 19);
447         }
448   if(ep->wid == Online_list[3].Wid)
449         {
450               GrLine(Online_list[3].Wid, Gcw, 0, 0, 79, 0);
451               GrLine(Online_list[3].Wid, Gcw, 0, 0, 0, 19);
452               GrLine(Online_list[3].Wid, Gc, 79, 0, 79, 19);
453               GrLine(Online_list[3].Wid, Gc, 0, 19, 79, 19);
454         }
455   if(ep->wid == Online_list[4].Wid)
456         {
457               GrLine(Online_list[4].Wid, Gcw, 0, 0, 79, 0);
458               GrLine(Online_list[4].Wid, Gcw, 0, 0, 0, 19);
459               GrLine(Online_list[4].Wid, Gc, 79, 0, 79, 19);
460               GrLine(Online_list[4].Wid, Gc, 0, 19, 79, 19);
461         }
462   if(ep->wid == AddbMw)
463   {
464        GrText(AddbMw, Gc, 7, 17, "Add  Member",-1, GR_TFASCII);
465        GrLine(AddbMw, Gcw, 0, 0, 79, 0);
466        GrLine(AddbMw, Gcw, 0, 0, 0, 24);
467        GrLine(AddbMw, Gc, 79, 0, 79, 24);
468        GrLine(AddbMw, Gc, 0, 24, 79, 24);
469   }
470   if(ep->wid == PCMdownbw)
471        GrText(PCMdownbw, Gc, 5, 12, "-",-1, GR_TFASCII);
472   if(ep->wid == PCMupbw)
473        GrText(PCMupbw, Gc, 4, 12, "+",-1, GR_TFASCII);
474   if(ep->wid == MICdownbw)
475        GrText(MICdownbw, Gc, 5, 12, "-",-1, GR_TFASCII);
476   if(ep->wid == MICupbw)
477        GrText(MICupbw, Gc, 4, 12, "+",-1, GR_TFASCII);
478   if(ep->wid == Addw)
479   {
480        GrText(Addw, Gc, 15, 30, "Member's ID:",-1, GR_TFASCII);
481        GrLine(Addw, Gc, 0, 0, 249, 0);
482        GrLine(Addw, Gc, 249, 0, 249, 89);
483        GrLine(Addw, Gc, 0, 0, 0, 89);
484        GrLine(Addw, Gc, 0, 89, 249, 89);
485   }
486   if(ep->wid == Addbw)
487   {
488        GrText(Addbw, Gc, 20, 20, "Add",-1, GR_TFASCII);
489        GrLine(Addbw, Gcw, 0, 0, 59, 0);
490        GrLine(Addbw, Gcw, 0, 0, 0, 29);
491        GrLine(Addbw, Gc, 59, 0, 59, 29);
492        GrLine(Addbw, Gc, 0, 29, 59, 29);
493   }
494   if(ep->wid == Tool1w)
495   {
496        GrText(Tool1w, Gc, 16, 20, "Login",-1, GR_TFASCII);
497        GrLine(Tool1w, Gc, 0, 0, 59, 0);
498        GrLine(Tool1w, Gc, 0, 0, 0, 29);
499        GrLine(Tool1w, Gc, 59, 0, 59, 29);
500   }
501   if(ep->wid == Tool2w)
502   {
503        GrText(Tool2w, Gc, 22, 20, "List",-1, GR_TFASCII);
```

```
504         GrLine(Tool2w, Gc, 0, 0, 59, 0);
505         GrLine(Tool2w, Gc, 0, 0, 0, 29);
506         GrLine(Tool2w, Gc, 59, 0, 59, 29);
507    }
508    if(ep->wid == Tool3w)
509    {
510         GrText(Tool3w, Gc, 22, 20, "Exit",-1, GR_TFASCII);
511         GrLine(Tool3w, Gc, 0, 0, 59, 0);
512         GrLine(Tool3w, Gc, 0, 0, 0, 29);
513         GrLine(Tool3w, Gc, 59, 0, 59, 29);
514    }
515    if(ep->wid == EnterAw)
516         GrSetFocus(EnterAw);
517
518  }
519
520  /*---------------------------------------------------------------------
521   * do_buttondown - deal with buttondown Event
522   *---------------------------------------------------------------------
523   */
524  void do_buttondown(GR_EVENT_BUTTON  *bp)
525  {
526   if(bp->wid == Tool3w)
527   {
528         GrLine(Tool3w, Gcg, 0, 0, 59, 0);
529         GrLine(Tool3w, Gcg, 0, 0, 0, 29);
530         GrLine(Tool3w, Gcg, 59, 0, 59, 29);
531   }
532   if(bp->wid == ViewIMbw)
533   {
534         GrLine(ViewIMbw, Gc, 0, 0, 59, 0);
535         GrLine(ViewIMbw, Gc, 0, 0, 0, 29);
536         GrLine(ViewIMbw, Gcw, 59, 0, 59, 29);
537         GrLine(ViewIMbw, Gcw, 0, 29, 59, 29);
538   }
539   if(bp->wid == Loginbw)
540   {
541         GrLine(Loginbw, Gc, 0, 0, 59, 0);
542         GrLine(Loginbw, Gc, 0, 0, 0, 29);
543         GrLine(Loginbw, Gcw, 59, 0, 59, 29);
544         GrLine(Loginbw, Gcw, 0, 29, 59, 29);
545   }
546   if(bp->wid == AddbAw)
547   {
548         GrLine(AddbAw, Gc, 0, 0, 79, 0);
549         GrLine(AddbAw, Gc, 0, 0, 0, 24);
550         GrLine(AddbAw, Gcw, 79, 0, 79, 24);
551         GrLine(AddbAw, Gcw, 0, 24, 79, 24);
552   }
553   if(bp->wid == InvitebYw)
554   {
555         GrLine(InvitebYw, Gc, 0, 0, 59, 0);
556         GrLine(InvitebYw, Gc, 0, 0, 0, 29);
557         GrLine(InvitebYw, Gcw, 59, 0, 59, 29);
558         GrLine(InvitebYw, Gcw, 0, 29, 59, 29);
559
560   }
561   if(bp->wid == InvitebNw)
562   {
563         GrLine(InvitebNw, Gc, 0, 0, 59, 0);
564         GrLine(InvitebNw, Gc, 0, 0, 0, 29);
565         GrLine(InvitebNw, Gcw, 59, 0, 59, 29);
566         GrLine(InvitebNw, Gcw, 0, 29, 59, 29);
567
568   }
569    if(bp->wid == Offline_list[0].Wid)
570         {
571               GrLine(Offline_list[0].Wid, Gc, 0, 0, 79, 0);
572               GrLine(Offline_list[0].Wid, Gc, 0, 0, 0, 19);
573               GrLine(Offline_list[0].Wid, Gcw, 79, 0, 79, 19);
574               GrLine(Offline_list[0].Wid, Gcw, 0, 19, 79, 19);
575         }
576   if(bp->wid == Offline_list[1].Wid)
577         {
578               GrLine(Offline_list[1].Wid, Gc, 0, 0, 79, 0);
579               GrLine(Offline_list[1].Wid, Gc, 0, 0, 0, 19);
580               GrLine(Offline_list[1].Wid, Gcw, 79, 0, 79, 19);
581               GrLine(Offline_list[1].Wid, Gcw, 0, 19, 79, 19);
582         }
583   if(bp->wid == Offline_list[2].Wid)
584         {
585               GrLine(Offline_list[2].Wid, Gc, 0, 0, 79, 0);
586               GrLine(Offline_list[2].Wid, Gc, 0, 0, 0, 19);
587               GrLine(Offline_list[2].Wid, Gcw, 79, 0, 79, 19);
588               GrLine(Offline_list[2].Wid, Gcw, 0, 19, 79, 19);
589         }
590   if(bp->wid == Offline_list[3].Wid)
591         {
592               GrLine(Offline_list[3].Wid, Gc, 0, 0, 79, 0);
593               GrLine(Offline_list[3].Wid, Gc, 0, 0, 0, 19);
594               GrLine(Offline_list[3].Wid, Gcw, 79, 0, 79, 19);
595               GrLine(Offline_list[3].Wid, Gcw, 0, 19, 79, 19);
596         }
597   if(bp->wid == Offline_list[4].Wid)
598         {
599               GrLine(Offline_list[4].Wid, Gc, 0, 0, 79, 0);
600               GrLine(Offline_list[4].Wid, Gc, 0, 0, 0, 19);
601               GrLine(Offline_list[4].Wid, Gcw, 79, 0, 79, 19);
602               GrLine(Offline_list[4].Wid, Gcw, 0, 19, 79, 19);
603         }
604   if(bp->wid == Online_list[0].Wid)
605         {
```

```
606               GrLine(Online_list[0].Wid, Gc, 0, 0, 79, 0);
607               GrLine(Online_list[0].Wid, Gc, 0, 0, 0, 19);
608               GrLine(Online_list[0].Wid, Gcw, 79, 0, 79, 19);
609               GrLine(Online_list[0].Wid, Gcw, 0, 19, 79, 19);
610         }
611   if(bp->wid == Online_list[1].Wid)
612         {
613               GrLine(Online_list[1].Wid, Gc, 0, 0, 79, 0);
614               GrLine(Online_list[1].Wid, Gc, 0, 0, 0, 19);
615               GrLine(Online_list[1].Wid, Gcw, 79, 0, 79, 19);
616               GrLine(Online_list[1].Wid, Gcw, 0, 19, 79, 19);
617         }
618   if(bp->wid == Online_list[2].Wid)
619         {
620               GrLine(Online_list[2].Wid, Gc, 0, 0, 79, 0);
621               GrLine(Online_list[2].Wid, Gc, 0, 0, 0, 19);
622               GrLine(Online_list[2].Wid, Gcw, 79, 0, 79, 19);
623               GrLine(Online_list[2].Wid, Gcw, 0, 19, 79, 19);
624         }
625   if(bp->wid == Online_list[3].Wid)
626         {
627               GrLine(Online_list[3].Wid, Gc, 0, 0, 79, 0);
628               GrLine(Online_list[3].Wid, Gc, 0, 0, 0, 19);
629               GrLine(Online_list[3].Wid, Gcw, 79, 0, 79, 19);
630               GrLine(Online_list[3].Wid, Gcw, 0, 19, 79, 19);
631         }
632   if(bp->wid == Online_list[4].Wid)
633         {
634               GrLine(Online_list[4].Wid, Gc, 0, 0, 79, 0);
635               GrLine(Online_list[4].Wid, Gc, 0, 0, 0, 19);
636               GrLine(Online_list[4].Wid, Gcw, 79, 0, 79, 19);
637               GrLine(Online_list[4].Wid, Gcw, 0, 19, 79, 19);
638         }
639   if(bp->wid == AddbMw)
640   {
641         GrLine(AddbMw, Gc, 0, 0, 79, 0);
642         GrLine(AddbMw, Gc, 0, 0, 0, 24);
643         GrLine(AddbMw, Gcw, 79, 0, 79, 24);
644         GrLine(AddbMw, Gcw, 0, 24, 79, 24);
645   }
646   if(bp->wid == Addbw)
647   {
648         GrLine(Addbw, Gc, 0, 0, 59, 0);
649         GrLine(Addbw, Gc, 0, 0, 0, 29);
650         GrLine(Addbw, Gcw, 59, 0, 59, 29);
651         GrLine(Addbw, Gcw, 0, 29, 59, 29);
652   }
653   if(bp->wid == EnterAw)
654         GrSetFocus(EnterAw);
655   if(bp->wid == EnterPw)
656         GrSetFocus(EnterPw);
```

```
657   if(bp->wid == EnterCw)
658         GrSetFocus(EnterCw);
659   if(bp->wid == PCMupbw)
660   {
661         //if(vol_p!=100)
662         if(vol_v!=100)
663               reflash_sound_state(10,0);
664     }
665   if(bp->wid == PCMdownbw)
666   {
667         //if(vol_p!=0)
668         if(vol_v!=0)
669               reflash_sound_state(-10,0);
670   }
671   if(bp->wid == MICupbw)
672   {
673         if(vol_m!=100)
674               reflash_sound_state(0,10);
675   }
676   if(bp->wid == MICdownbw)
677   {
678         if(vol_m!=0)
679               reflash_sound_state(0,-10);
680   }
681 }
682
683 /*--------------------------------------------------------------------
684  * do_buttonup - deal with buttonup Event
685  *--------------------------------------------------------------------
686  */
687 void do_buttonup(GR_EVENT_BUTTON    *bp)
688 {int i;
689    char buf[50];
690
691   if(bp->wid == Tool3w)
692   {
693         GrLine(Tool3w, Gc, 0, 0, 59, 0);
694         GrLine(Tool3w, Gc, 0, 0, 0, 29);
695         GrLine(Tool3w, Gc, 59, 0, 59, 29);
696
697         logout(1);
698
699   }
700   if(bp->wid == ViewIMbw)
701   {
702         GrLine(ViewIMbw, Gcw, 0, 0, 59, 0);
703         GrLine(ViewIMbw, Gcw, 0, 0, 0, 29);
704         GrLine(ViewIMbw, Gc, 59, 0, 59, 29);
705         GrLine(ViewIMbw, Gc, 0, 29, 59, 29);
706         if(Cancel_voip_able)
707               cancel_voip(1);
```

```
708         if(Leave==2)
709              cancel_leave_message(1);
710         if(Leave)
711              Next_message=1;
712
713         GrUnmapWindow(ViewIMbw);
714         GrUnmapWindow(ViewIMw);
715    }
716    if(bp->wid == Loginbw)
717    {
718         GrLine(Loginbw, Gcw, 0, 0, 59, 0);
719         GrLine(Loginbw, Gcw, 0, 0, 0, 29);
720         GrLine(Loginbw, Gc, 59, 0, 59, 29);
721         GrLine(Loginbw, Gc, 0, 29, 59, 29);
722
723         printf("%d %d\n", Count,Countp);
724         printf("%s %s\n",Account,Password);
725         if(!(Count==0 || Countp==0))
726         {
727              if(Connect_server(Host, Service,Account,Password,0)==0)
728              {
729                   GrUnmapWindow(Loginw);
730                   GrUnmapWindow(Tool1w);
731                   GrMapWindow(Tool2w);
732                   GrMapWindow(Tool3w);
733                   GrMapWindow(Listw);
734                   GrMapWindow(Offlinew);
735                   GrMapWindow(Offline_list[0].Wid);
736                   GrMapWindow(Offline_list[1].Wid);
737                   GrMapWindow(Offline_list[2].Wid);
738                   GrMapWindow(Offline_list[3].Wid);
739                   GrMapWindow(Offline_list[4].Wid);
740                   GrMapWindow(Online_list[0].Wid);
741                   GrMapWindow(Online_list[1].Wid);
742                   GrMapWindow(Online_list[2].Wid);
743                   GrMapWindow(Online_list[3].Wid);
744                   GrMapWindow(Online_list[4].Wid);
745                   GrMapWindow(Onlinew);
746                   GrMapWindow(AddbMw);
747                   GrMapWindow(PCMw);
748                   GrMapWindow(PCMupbw);
749                   GrMapWindow(PCMdownbw);
750                   GrMapWindow(MICw);
751                   GrMapWindow(MICupbw);
752                   GrMapWindow(MICdownbw);
753                   kill(Child,SIGINT);
754                   reflash_sound_state(0,0);
755              }
756              else
757              {
758                show_warning_message("Please check account and password
759                                      again!!",1);
760              Clean_windows(EnterAw);
761              Clean_windows(EnterPw);
762           }
763
764              Count=0;       /*reset account*/
765              Account[0]='\0';    /*clean account*/
766              Xpos=1;        /*reset input position*/
767              Countp=0;
768              Password[0]='\0';
769              Xposp=1;
770         }
771
772    }
773    if(bp->wid == AddbAw)
774    {
775         GrLine(AddbAw, Gcw, 0, 0, 79, 0);
776         GrLine(AddbAw, Gcw, 0, 0, 0, 24);
777         GrLine(AddbAw, Gc, 79, 0, 79, 24);
778         GrLine(AddbAw, Gc, 0, 24, 79, 24);
779
780         if(!(Count==0 || Countp==0))
781         {
782              if(Connect_server(Host, Service,Account,Password,1)==0)
783              {
784                   GrUnmapWindow(Loginw);
785                   GrUnmapWindow(Tool1w);
786                   GrMapWindow(Tool2w);
787                   GrMapWindow(Tool3w);
788                   GrMapWindow(Listw);
789                   GrMapWindow(Offlinew);
790                   GrMapWindow(Offline_list[0].Wid);
791                   GrMapWindow(Offline_list[1].Wid);
792                   GrMapWindow(Offline_list[2].Wid);
793                   GrMapWindow(Offline_list[3].Wid);
794                   GrMapWindow(Offline_list[4].Wid);
795                   GrMapWindow(Online_list[0].Wid);
796                   GrMapWindow(Online_list[1].Wid);
797                   GrMapWindow(Online_list[2].Wid);
798                   GrMapWindow(Online_list[3].Wid);
799                   GrMapWindow(Online_list[4].Wid);
800                   GrMapWindow(Onlinew);
801                   GrMapWindow(AddbMw);
802                   GrMapWindow(PCMw);
803                   GrMapWindow(PCMupbw);
804                   GrMapWindow(PCMdownbw);
805                   GrMapWindow(MICw);
806                   GrMapWindow(MICupbw);
807                   GrMapWindow(MICdownbw);
808                   kill(Child,SIGINT);
                      reflash_sound_state(0,0);
```

```
809             }
810             else
811             {
812               show_warning_message("Please check account and password
                                  again!!",1);
813             Clean_windows(EnterAw);
814             Clean_windows(EnterPw);
815         }
816
817             Count=0;/*reset account*/
818             Account[0]='\0';/*clean account*/
819             Xpos=1; /*reset input position*/
820             Countp=0;
821             Password[0]='\0';
822             Xposp=1;
823       }
824
825   }
826   if(bp->wid == InvitebYw)
827   {
828        GrLine(InvitebYw, Gcw, 0, 0, 59, 0);
829        GrLine(InvitebYw, Gcw, 0, 0, 0, 29);
830        GrLine(InvitebYw, Gc, 59, 0, 59, 29);
831        GrLine(InvitebYw, Gc, 0, 29, 59, 29);
832      if(Invited)
833        {
834             Send_Command("#yes");
835             Answer=1;
836        }
837        if(Leave==3)/*have a message */
838             {
839              Send_Command("#listen_message");
840              Leave=4;
841            }
842
843        GrUnmapWindow(Invitew);
844
845   }
846         if(bp->wid == InvitebNw)
847   {
848        GrLine(InvitebNw, Gcw, 0, 0, 59, 0);
849        GrLine(InvitebNw, Gcw, 0, 0, 0, 29);
850        GrLine(InvitebNw, Gc, 59, 0, 59, 29);
851        GrLine(InvitebNw, Gc, 0, 29, 59, 29);
852        if(Invited)
853        {
854             Send_Command("#no");
855             Answer=2;
856        }
857        if(Leave==3)/*have a message */
858          Leave=0;
```

```
859
860        GrUnmapWindow(Invitew);
861
862   }
863     for(i=0;i<Maxlist;i++)
864        {
865         if(bp->wid == Offline_list[i].Wid)
866            {
867            GrLine(Offline_list[i].Wid, Gcw, 0, 0, 79, 0);
868                 GrLine(Offline_list[i].Wid, Gcw, 0, 0, 0, 19);
869                 GrLine(Offline_list[i].Wid, Gc, 79, 0, 79, 19);
870              GrLine(Offline_list[i].Wid, Gc, 0, 19, 79, 19);
871
872              if(strcmp(Offline_list[i].MemberID,"")!=0 && (!Invited)
                 && (!Invite)&& (!Leave) )
873                {
874                 sprintf(buf,"#leave_message
                 %s",Offline_list[i].MemberID);
875                      Send_Command(buf);
876
877                      Leave=1;
878
879                      sprintf(buf,"leave a message to
                              %s ...",Offline_list[i].MemberID);
880                      show_warning_message(buf,3);
881
882                 }
883            }
884
885        }
886
887     for(i=0;i<Maxlist;i++)
888        {
889         if(bp->wid == Online_list[i].Wid)
890            {
891            GrLine(Online_list[i].Wid, Gcw, 0, 0, 79, 0);
892                 GrLine(Online_list[i].Wid, Gcw, 0, 0, 0, 19);
893                 GrLine(Online_list[i].Wid, Gc, 79, 0, 79, 19);
894              GrLine(Online_list[i].Wid, Gc, 0, 19, 79, 19);
895
896              if(strcmp(Online_list[i].MemberID,"")!=0 && (!Invited)
                 && (!Invite)&& (!Leave) )
897                {
898                 sprintf(buf,"#invite %s",Online_list[i].MemberID);
899                      Send_Command(buf);
900
901                      Invite=1;
902
903                      sprintf(buf,"invite
                    %s ...",Online_list[i].MemberID);
904                      show_warning_message(buf,3);
```

```
                    }

                }

        }

  if(bp->wid == AddbMw)
  {
       GrLine(AddbMw, Gcw, 0, 0, 79, 0);
       GrLine(AddbMw, Gcw, 0, 0, 0, 24);
       GrLine(AddbMw, Gc, 79, 0, 79, 24);
       GrLine(AddbMw, Gc, 0, 24, 79, 24);
       GrMapWindow(Addw);
       GrMapWindow(Addbw);
       GrMapWindow(EnterCw);
       GrSetFocus(EnterCw);
       /*open soft keyborad*/
        Child=fork();
        if(Child==0)
         execv("./bin/nxkbd",NULL);
  }
  if(bp->wid == Addbw)
  {
       GrLine(Addbw, Gcw, 0, 0, 59, 0);
       GrLine(Addbw, Gcw, 0, 0, 0, 29);
       GrLine(Addbw, Gc, 59, 0, 59, 29);
       GrLine(Addbw, Gc, 0, 29, 59, 29);

       if(Count!=0)
       {
       if(add_friend(Account))
                 show_warning_message("Already had this friend",1);
            else
            {
               show_warning_message("Add friend ok!!",1);
               Send_Command("#who");
           }

            Count=0;  /*reset account*/
            Account[0]='\0';/*clean account*/
            Xpos=1;        /*reset input position*/

            kill(Child,SIGINT);
            GrUnmapWindow(Addw);
       }
  }

}

/*--------------------------------------------------------------------
 * do_keystroke - deal with keyin Event
 *--------------------------------------------------------------------
 */
void do_keystroke(GR_EVENT_KEYSTROKE *kp)
{
  GR_SIZE        width;         /* width of character */
  GR_SIZE        height;        /* height of character */
  GR_SIZE        base;          /* height of baseline */
  GR_GC_ID  fgc;
  GR_COORD  *xpos;
  GR_WINDOW_ID   fwin;
  int  *count;
  char *input;


  fwin=GrGetFocus();

  if(fwin!=EnterPw)
  {
       count=&Count;
       xpos=&Xpos;
       input=Account;
  }
  else
  {
       count=&Countp;
       xpos=&Xposp;
       input=Password;
  }

  GrGetGCTextSize(Gc, &kp->ch, 1, GR_TFASCII, &width, &height, &base);

  if((kp->ch == '\r') || (kp->ch == '\n'))
  {
       *xpos = Begxpos;
       Ypos += height;
       return;
  }
  if(kp->ch == '\b')
  {
       if(*xpos <= Begxpos)
            return;
       *xpos -= 8;
       (*count)--;

       fgc=GrNewGC();
       GrFillRect( fwin, fgc, *xpos, Ypos-height+base+1, 8,
                 height+2);
       *(input+(*count))=' ';
       return;
  }
```

```
1006
1007  if(fwin!=EnterPw)
1008   GrText(fwin, Gck, *xpos, Ypos+base, &kp->ch, 1, 0);
1009  else
1010   GrText(fwin, Gck, *xpos, Ypos+base,"*", 1, 0);
1011
1012  *(input+(*count))= kp->ch;
1013  (*count)++;
1014  *(input+(*count))='\0';
1015  *xpos += 8;
1016
1017 }
1018
1019 /*-----------------------------------------------------------------
1020  * show_invite_window
1021 *-----------------------------------------------------------------
1022 */
1023 void show_invite_window(char *string)
1024 {
1025  GrMapWindow(Invitew);
1026  GrMapWindow(InvitebYw);
1027  GrMapWindow(InvitebNw);
1028
1029  GrText(Invitew, Gc, 10, 20,string,-1, GR_TFASCII);
1030  GrFlush();
1031 }
1032
1033 /*-----------------------------------------------------------------
1034  * show_warning_message
1035 *-----------------------------------------------------------------
1036 */
1037 void show_warning_message(char *string,int mode)
1038 {
1039  GrMapWindow(ViewIMw);
1040  if(mode!=3)
1041  GrMapWindow(ViewIMbw);
1042
1043  if(mode==1)
1044   GrText(ViewIMbw, Gc, 19, 20, "OK",-1, GR_TFASCII);
1045  else if(mode==2)
1046   GrText(ViewIMbw, Gc, 19, 20, "Stop",-1, GR_TFASCII);
1047
1048    GrText(ViewIMw, Gc, 10, 20, string, -1, GR_TFASCII);
1049    GrFlush();
1050 }
1051
1052 /*-----------------------------------------------------------------
1053  * close_warning_message
1054 *-----------------------------------------------------------------
1055 */
1056 void close_warning_message(void)
1057 {   Clean_windows(ViewIMw);
1058  GrUnmapWindow(ViewIMw);
1059  GrUnmapWindow(ViewIMbw);
1060  GrFlush();
1061 }
1062
1063 /*-----------------------------------------------------------------
1064  * Clean_windows
1065 *-----------------------------------------------------------------
1066 */
1067 void Clean_windows(GR_WINDOW_ID wid)
1068 {
1069   GrUnmapWindow(wid);
1070     GrMapWindow(wid);
1071
1072 }
1073 /*-----------------------------------------------------------------
1074  * back_to_login - back to login windows
1075 *-----------------------------------------------------------------
1076 */
1077 void back_to_login(void)
1078 {
1079       GrUnmapWindow(Tool2w);
1080       GrUnmapWindow(Tool3w);
1081       GrUnmapWindow(Listw);
1082       GrUnmapWindow(Addw);
1083       GrUnmapWindow(ViewIMw);
1084       GrUnmapWindow(Invitew);
1085       GrMapWindow(Tool1w);
1086       GrMapWindow(Loginw);
1087
1088       GrFlush();
1089
1090       clean_list();
1091
1092       Child=fork();
1093        if(Child==0)
1094         execv("./bin/nxkbd",NULL);
1095 }
1096
1097 /*-----------------------------------------------------------------
1098  * add_list - add id from online or offline list
1099 *-----------------------------------------------------------------
1100 */
1101
1102 void add_list(int mode,char *id)   /*mode 1= Online_list
                                         2=Offline_list*/
1103 {
1104  int i;
1105
1106  if(mode==1)
```

```
  {
       for(i=0;i<Maxlist;i++)
       {
            if(strcmp(Online_list[i].MemberID,"")==0)
            {
              strcpy(Online_list[i].MemberID,id);
              break;
          }
       }
  }
 else
  {
       for(i=0;i<Maxlist;i++)
       {
            if(strcmp(Offline_list[i].MemberID,"")==0)
            {
               strcpy(Offline_list[i].MemberID,id);
               break;
          }
       }
  }

}

/*---------------------------------------------------------------
 * remove_list - remove id from online or offline list
*---------------------------------------------------------------
 */
void remove_list(int mode,char *id)
{int i;

 if(mode==1)
  {
       i=check_list(mode,id);

       if(i==(Maxlist-1))
       {
            strcpy(Online_list[i].MemberID,"");
            return;
       }

       for(i=i+1;i<Maxlist;i++)
 strcpy(Online_list[i-1].MemberID,Online_list[i].MemberID);
  }
 else
  {
       i=check_list(mode,id);

       if(i==(Maxlist-1))
       {
            strcpy(Offline_list[i].MemberID,"");
            return;
       }

       for(i=i+1;i<Maxlist;i++)
 strcpy(Offline_list[i-1].MemberID,Offline_list[i].MemberID);

  }

}
/*---------------------------------------------------------------
 * clean_list - reflash online and offline list
*---------------------------------------------------------------
 */
void clean_list(void)
{
 int i;
 for(i=0;i<Maxlist;i++)
 {
       strcpy(Online_list[i].MemberID,"");
       strcpy(Offline_list[i].MemberID,"");
 }

}

/*---------------------------------------------------------------
 * check_list - check id , is it had heen showed in online or offline
              list
*---------------------------------------------------------------
*/
int  check_list(int mode,char *id)
{
 int i;

 if(mode==1)
  {
       for(i=0;i<Maxlist;i++)
         if(strcmp(Online_list[i].MemberID,id)==0)
              return i;
    }
 if(mode==2)
  {
       for(i=0;i<Maxlist;i++)
            if(strcmp(Offline_list[i].MemberID,id)==0)
             return i;
  }

 return -1;
}
/*---------------------------------------------------------------
 * reflash_list - reflash online and offline list
*---------------------------------------------------------------
```

```
 */
void reflash_list(void)
{int i;

 for(i=0;i<Maxlist;i++)
      Clean_windows(Online_list[i].Wid);

 for(i=0;i<Maxlist;i++)
      if(strcmp(Online_list[i].MemberID,"")!=0)
           GrText(Online_list[i].Wid, Gc, 7, 14,
           Online_list[i].MemberID,-1, GR_TFASCII);

 for(i=0;i<Maxlist;i++)
      Clean_windows(Offline_list[i].Wid);

 for(i=0;i<Maxlist;i++)
      if(strcmp(Offline_list[i].MemberID,"")!=0)
           GrText(Offline_list[i].Wid, Gc, 7, 14,
           Offline_list[i].MemberID,-1, GR_TFASCII);

 GrFlush();
}

/*-----------------------------------------------------------------
 * reflash_sound_state -
*-----------------------------------------------------------------
*/
void reflash_sound_state(int p,int m)
{   char temp[20];
    int pcm,mic;

 Clean_windows(PCMw);
 Clean_windows(MICw);

 pcm=volume_vol(vol_v+p);

 //pcm=volume_pcm(vol_p+p);
   mic=volume_mic(vol_m+m);

    sprintf(temp,"%d",pcm);
 GrText(PCMw, Gck, 20, 13,temp,-1, GR_TFASCII);
 sprintf(temp,"%d",mic);
 GrText(MICw, Gck, 20, 13,temp,-1, GR_TFASCII);

 GrFlush();
}
```

/*vm_client.h*/

```

 #include <unistd.h>
 #include <stdlib.h>
 #include <string.h>
 #include <stdio.h>
 #include <sys/types.h>
 #include <errno.h>
 #include <microwin/nano-X.h>
 #include <sys/time.h>
 #include "audio.h"

 #define Maxlist 5

 struct member_list{
  GR_WINDOW_ID   Wid;
  char MemberID[20];

 };

 typedef struct member_list Member_list;
```

/* vm_server.c chet and get host voip ip address server*/

```

 #include <sys/types.h>
 #include <sys/socket.h>
 #include <sys/time.h>
```

```
#include <netinet/in.h>
#include <sys/stat.h>

#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>

#define BUFSIZE 800
#define    QLEN        4  /* maximum connection queue length */
#define    LINELEN   128
fd_set AFDS;               /* active file descriptor set */

extern int errno;
int  FD_size;
int    MSOCK;               /* master server socket        */

struct online{
   int sock;
   int talksock;   /*0=free  >0 =tlaking to another who is online*/
   int messagesock; /*0=free  >0 =leaving a message*/
   short int answer;  /*while inviting, wait for answer*/
   short int state; /*0=offline 1=online 2=not sure ,get 1 chance*/
   pthread_t thread;
    char  name[20];
};

struct online Online[QLEN];

struct send_to_thread{
   int  sock;
    char input[20];
};


int    errexit(const char *format, ...);
int   passiveTCP(const char *service, int qlen);
int    passiveUDP(const char *service);
void   start_service(int ,fd_set *);
void   Send_Message(int ,char * );
void   send_list(int);
void   broadcast_message(int ,char *,fd_set *);
void   first_connect(int , fd_set *);
void   *invite_player(void *arg);
void  *server_control(void *arg);
void   *leave_message(void *arg);
void  *listen_message(void *arg);
void   *check_online_state(void *arg);

void   start_leave_message (char *name ,int sock);
int    find_name(char *string,int fd);
int    find_id(int sock);
void   client_logout(int ,fd_set *);
void   client_cancel_voip(int );
void   client_cancel_leave_message(int s);
void  cancel_thread(pthread_t thread);
void   get_client_address(void);

/*------------------------------------------------------------------
 * main - Chat server
 *------------------------------------------------------------------
 */
int
main(int argc, char *argv[])
{
 char *service = "7750";  /* service name or port number     */
 struct sockaddr_in fsin; /* the from address of a client    */
 fd_set    rfds;                   /* read file descriptor set
 */
 int  alen;                      /* from-address length      */
 int  fd;
 pthread_t a_thread;
 pthread_t b_thread;

 switch (argc) {
 case 1:
      break;
 case 2:
      service = argv[1];
      break;
 default:
      errexit("usage: voipserver [port]\n");
 }

 MSOCK = passiveTCP(service, QLEN);

 FD_size = FD_SETSIZE;
 FD_ZERO(&AFDS);
 FD_SET(MSOCK, &AFDS);

 pthread_create(&a_thread, NULL, server_control,(void *)MSOCK);
 pthread_create(&b_thread, NULL, check_online_state,NULL);

 printf("--- Voice Messenger Server ---\n\nMain socket: %d\n",MSOCK);

 while (1) {
      memcpy(&rfds, &AFDS, sizeof(rfds));

      if (select(FD_size, &rfds, (fd_set *)0, (fd_set *)0,
        (struct timeval *)0) < 0)
```

```
             continue;
       if (FD_ISSET(MSOCK, &rfds)) {
            int  ssock;

            alen = sizeof(fsin);

            ssock = accept(MSOCK, (struct sockaddr *)&fsin,&alen);
            if (ssock < 0)
                 errexit("accept: %s\n",strerror(errno));

            printf("accect %d socket\n",ssock);
            /*get playr name*/
            first_connect(ssock,&AFDS);

       }

       for (fd=0; fd<FD_size; ++fd)
            if (fd != MSOCK && FD_ISSET(fd, &rfds))
                 start_service(fd,&AFDS);

  }

 return 0;
}

/*------------------------------------------------------------------
 * first_connect - create account or client login and set client can
       be servered or not
*------------------------------------------------------------------
 */
void first_connect(int sock, fd_set *afds)
{   char
buf[LINELEN+1],*tok,temp[256],account[20],password[20],a[20],p[20]
;
 int    count,state=0,find=0,i,login=0;/* =0 no such account or
passwors error,=1 login ok*/
 FILE *fp;

 count=read(sock, buf, LINELEN);
 buf[count]='\0';
 puts(buf);
   if(strcmp(buf,"#create")==0)/*create account*/
   {
  printf("Create account\n");
  count=read(sock, buf, LINELEN);
  buf[count]='\0';

  tok = strtok(buf," ");
  strcpy(account,tok);
  tok = strtok(NULL," ");
  strcpy(password,tok);

fp=fopen("account","a+");
/*test account exist or not?*/
while(fgets(temp,256,fp))
  {tok = strtok(temp," ");
    if(strcmp(tok,account)==0)
      {state=0;
       find=1;
       printf("Error account existed!!");
       break;
      }
  }
if(!find)
   {
 fprintf(fp,"%s %s\n",account,password);
   state=1;
  }
 fclose(fp);
 }
 else if(strcmp(buf,"#login")==0)/*client login*/
 {
     printf("Login\n");
    count=read(sock, buf, LINELEN);
     buf[count]='\0';
    puts(buf);
     if(access("account",R_OK)==0)/*is file exist?*/
      {
      fp=fopen("account","r");

       tok = strtok(buf," ");
      strcpy(a,tok);
       tok = strtok(NULL," ");
     strcpy(p,tok);

    while(fgets(temp,256,fp))
      {
          sscanf(temp,"%s %s\n",account,password);
          if(strcmp(a,account)==0)
           {
         printf("account has be finded\n");

         for(i=0;i<QLEN;i++)
          if((strcmp(Online[i].name,a)==0))
          {
          printf("account had login!!!");
          login=1;
          break;
        }
       if((strcmp(p,password)==0)&&login==0)
        {
          state=1;
```

```
            break;
          }
         else
           printf("Password incorrect\n");
       }
     }
     fclose(fp);
     }
     else
      printf("Account file not existed\n");
   }

   if(state==1)
  { printf("Login ok\n");
     Send_Message(sock,"#ok");/*login ok*/

     /*search for nonuse Online's name*/
     for(i=0;strcmp(Online[i].name,"")!=0;i++);
     strcpy(Online[i].name,account);
     Online[i].sock=sock;

     sprintf(buf,"#login %s",Online[i].name);
   broadcast_message(sock,buf,afds);

   FD_SET(sock, afds);
    }
   else /*tell client to check input again*/
     { printf("Error!!\n");
     Send_Message(sock,"#error");
     close(sock);
     }


}

/*---------------------------------------------------------------------
 * server_control -  control server
 *---------------------------------------------------------------------
 */
void  *server_control(void *arg)
{
  char command[50];
  int id,i,msock;

  msock=(int )arg;

  do
    {
   scanf("%s",command);

  if(strcmp(command,"#online")==0)
  {
    for(i=0;i<QLEN;i++)
    if(strcmp(Online[i].name,"")!=0)
     printf("ID: %d Socket: %d Name:
            %s\n",i,Online[i].sock,Online[i].name);

  }
  if(strcmp(command,"#kick")==0)
  {
      puts("which online_ID do you want to kick?");
      scanf("%d",&id);
      if(strcmp(Online[id].name,"")!=0)
        client_logout(Online[id].sock,&AFDS);
  }
   }while(strcmp(command,"#shutdown")!=0);

   for(i=0;i<QLEN;i++)
      if(strcmp(Online[i].name,"")!=0)
        client_logout(Online[i].sock,&AFDS);

 close(msock);/*close msock*/
 exit(1);
}

/*---------------------------------------------------------------------
 * check_online_state -  1.5 min to check client online state
 *---------------------------------------------------------------------
 */
void *check_online_state(void *arg)
{ int i;

   while(1)
   {
  puts("--check online state--") ;
  for(i=0;i<QLEN;i++)
    {
       if(strcmp(Online[i].name,"")!=0)
        {
           if(Online[i].state==1)
             Online[i].state=0;
           else if(Online[i].state==0)
             Online[i].state=2;
           else if(Online[i].state==2)
              {
                client_logout(Online[i].sock,&AFDS);
              }
        }
    }
  sleep(90);
```

```
   }
}

/*---------------------------------------------------------------------
 * start_service -  broadcast message or send ip to client
 *---------------------------------------------------------------------
 */
void start_service(int fd,fd_set *afds)
{
 char buf[LINELEN+1];
 int   count,i;
 struct send_to_thread *arg;
 pthread_t a_thread;

 count = read(fd, buf,LINELEN);
 buf[count]='\0';
 printf("socket:%d get: %s\n",fd,buf);/*show what server get*/
 i=find_id(fd); /*find client's id*/

 if (count <= 0)
      client_logout(fd,afds);/*socket read fail , close socket*/

 else
    {
       /*left game , send message to all,close socket*/
       if(strcmp(buf,"#bye")==0)
      {
       client_logout(fd,afds);
       return;
      }
      else if(strcmp(buf,"#still_alive")==0)
      {
         Online[i].state=1;
      }
      /*invite people ,send ip to him*/
      else if(strncmp(buf,"#invite",7)==0)
       {
          arg=(struct send_to_thread *)malloc(sizeof(struct
                send_to_thread));
          arg->sock=fd;
          strcpy(arg->input,buf);
         pthread_create(&a_thread, NULL, invite_player,(void *)arg);

       }
       /*client accept to open voip*/
      else if(strcmp(buf,"#yes")==0 && Online[i].answer==2)
       {
         Online[i].answer=1;
       }
       /*client not accept to open voip*/
      else if(strcmp(buf,"#no")==0 && Online[i].answer==2)
       {
         Online[i].answer=0;
       }
      /*get who is online*/
      else if(strcmp(buf,"#who")==0)
       {
           send_list(fd);
       } /*client want to cancle voip */
       else if(strcmp(buf,"#stop_voip")==0)
        {
          client_cancel_voip(i);
        }
        /*leave a message*/
       else if(strncmp(buf,"#leave_message",14)==0)
       {
          arg=(struct send_to_thread *)malloc(sizeof(struct
                send_to_thread));
          arg->sock=fd;
          strcpy(arg->input,buf);
        pthread_create(&Online[i].thread, NULL, leave_message,(void
                *)arg);

         }
        /*check you get message or not*/
       else if(strcmp(buf,"#message")==0)
         {
               if(access(Online[i].name,R_OK)==0)
                Send_Message(fd,"#get_message");
         }
        /*stop leave a message*/
       else if(strcmp(buf,"#stop_leave_message")==0)
          {
               if(Online[i].messagesock)
                client_cancel_leave_message(i);
          }
         /*download message*/
        else if(strcmp(buf,"#listen_message")==0)
         {
           pthread_create(&Online[i].thread, NULL,
           listen_message,(void *)fd);

         }

      }
}

/*---------------------------------------------------------------------
 * invite_player - invite player to open voip
 *---------------------------------------------------------------------
 */
void *invite_player(void *arg)
```

```
{   char   buf[LINELEN+1],*tok;
 int    i,find;
   struct send_to_thread *input;

    input=(struct send_to_thread *) arg;

    tok = strtok(input->input," ");
  tok = strtok(NULL," ");

    puts(tok);
       /*find playr's name*/
       find=find_name(tok,input->sock);

       printf("find : %d\n",find);

    if((find >= 0))
      {

     i=find_id(input->sock);
     sprintf(buf,"#invite %s",Online[i].name);
       Send_Message(Online[find].sock,buf);/*tell player there is a
                                             voip invition*/
       /*send message to tell him,make a decision*/


       Online[find].answer=2; /*set to get answer from tcp*/

       puts("waiting for answer");
       while(Online[find].answer==2);/*wait for answer ,1=yes 0=no*/
       puts("get answer");

     if(Online[find].answer==1)/*accept*/
      {
         Send_Message(input->sock,"#accept");

       /*get clients's ip and port and send to them*/
         get_client_address();
         /*set player talksock,client tlak to who(socket)*/
         Online[i].talksock=Online[find].sock;
          Online[find].talksock=input->sock;
         }

      else/*not accept*/
        {
       sprintf(buf,"%s not accept yout invition",Online[find].name);
       Send_Message(input->sock,buf);
        }
       }
     else/*no find*/
       {if(find==-2)
       sprintf(buf,"%s no such ID is online",buf);
      else
      sprintf(buf,"Can not invite yourself");
     Send_Message(input->sock,buf);
      }

 free(input);

}
/*--------------------------------------------------------------------
 * leave_message - To prepareing for leaving a message
 *--------------------------------------------------------------------
 */
void *leave_message(void *arg)
{   char   buf[20],temp[LINELEN+1],*tok;
 int    find=0,res,sock;
 FILE   *fp;
   struct send_to_thread *input;

   input=(struct send_to_thread *) arg;


 res=pthread_setcancelstate(PTHREAD_CANCEL_ENABLE,NULL);
 if(res != 0)
  {printf("Thread setcancelstste failed!!!\n");
   exit(1);
    }

 res=pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS,NULL);
 if(res != 0)
  {printf("Thread setcanceltype failed!!!\n");
   exit(1);
    }

   tok = strtok(input->input," ");
 tok = strtok(NULL," ");

  strcpy(buf,tok);

  if(access("account",R_OK)==0)/*is file exist?*/
   {
    fp=fopen("account","r");
    while(fgets(temp,256,fp))
     {
         tok = strtok(temp," ");
         if((strcmp(buf,tok)==0))/*is account exist?*/
          {
             find=1;
             break;
           }
          }
         fclose(fp);
```

```
503          }
504
505        sock=input->sock;
506        free(input);  /*will not be used later */
507
508   if(find)
509   {
510      Send_Message(sock,"#find");
511        start_leave_message(buf,sock);
512
513      }
514      else if(find==0)
515       Send_Message(sock,"no such account !!!");
516
517
518   pthread_exit(NULL);
519  }
520
521  /*---------------------------------------------------------------------
522   * start_leave_message - start leave a message
523   *---------------------------------------------------------------------
524   */
525  void  start_leave_message (char *name ,int sock)
526  {
527   char   filename[30],temptime[256];
528   char   *service = "9855";
529   struct  sockaddr_in fsin;
530   unsigned char  buf[BUFSIZE];
531     int    i,alen;
532   FILE   *fp;
533     time_t curtime;
534     struct tm *loctime;
535
536   fp=fopen(name,"a");
537
538   for(i=0;;i++)/*search nonuse message file*/
539   {
540       sprintf(filename,"message_%s_%d",name,i);
541    if(access(filename,R_OK)!=0)
542       {
543      printf("open message file: %s\n",filename);
544         break;
545     }
546     }
547
548    /* Get the current time. */
549      curtime = time (NULL);
550
551      /* Convert it to local time representation. */
552      loctime = localtime (&curtime);
553
554    strftime(temptime,256, "%Y-%m-%d_%H:%M:%S", loctime);
555    puts(temptime);
556    /*record message log*/
557      i=find_id(sock);
558
559    fprintf(fp,"%s %s %s\n",Online[i].name,temptime,filename);
560
561    fclose(fp);
562
563    Online[i].messagesock = passiveUDP(service);
564
565    puts("start leave a message");
566
567    while(1)
568     {
569      recvfrom(Online[i].messagesock, buf, sizeof(buf), 0,(struct
    sockaddr *)&fsin, &alen);
570
571      fp=fopen(filename,"a");
572      fwrite (buf , 1 , BUFSIZE, fp);
573      fclose(fp);
574
575       }
576
577
578  }
579
580  /*---------------------------------------------------------------------
581   * listen_message - send message file to user
582   *---------------------------------------------------------------------
583   */
584  void *listen_message(void *arg)
585  {
586   struct  sockaddr_in fsin; /* the from address of a client    */
587   char *service = "8885";  /* service name or port number     */
588   unsigned char buf[BUFSIZE];
589   char    temp[100],account[50],time[50],filename[50];
590   int   sock,csock,n,i;               /* server socket          */
591   int   alen;                 /* from-address length     */
592   struct in_addr add;
593   struct stat filestate;
594   FILE *fp,*mfp;
595
596   csock=(int)arg;
597
598   sock = passiveUDP(service);
599     alen = sizeof(fsin);
600
601     i=find_id(csock);
602
603     Online[i].messagesock=sock;
```

```
604
605     printf("waiting....get address\n");
606     n=recvfrom(sock, temp, sizeof(temp), 0,(struct sockaddr *)&fsin,
    &alen);
607   temp[n]='\0';
608
609     puts(temp);
610
611     add.s_addr=fsin.sin_addr.s_addr;
612     printf("ip %s port %d\n",inet_ntoa(add),ntohs(fsin.sin_port));
613
614     /*tell client that server get ip*/
615     Send_Message(csock,"#get_v_message");
616
617     puts("start send message!!!\n");
618
619
620     fp=fopen( Online[i].name,"r");
621
622     while(fgets(temp,256,fp))
623     {
624      /*read player's message log*/
625      sscanf(temp,"%s %s %s\n",account,time,filename);
626      stat(filename,&filestate);
627
628      /*send from and time*/
629      sprintf(temp,"From:%s__Time:%s
          %ld",account,time,filestate.st_size);
630      printf("sending %s size %ld\n",filename,filestate.st_size);
631        sendto(sock,temp, sizeof(temp), 0, (struct sockaddr *)&fsin,
               sizeof(fsin));
632
633        mfp=fopen(filename,"r");
634
635       while(fread (buf,1,BUFSIZE, mfp))
636          {sendto(sock,buf, sizeof(buf), 0, (struct sockaddr *)&fsin,
                    sizeof(fsin));
637           usleep(1);
638        }
639
640        usleep(100000);
641      sprintf(buf,"#message end");
642      sendto(sock,buf, sizeof(buf), 0, (struct sockaddr *)&fsin,
               sizeof(fsin));
643
644        puts("send over");
645
646      fclose(mfp);
647      remove(filename);       /*remove message file*/
648     }
649
```

```
650     sprintf(temp,"#end");
651     printf("send end\n");
652
653   sendto(sock,temp, sizeof(temp), 0, (struct sockaddr *)&fsin,
           sizeof(fsin));
654
655   fclose(fp);
656   remove(Online[i].name);/*remove message log*/
657   close(sock);
658
659     close(Online[i].messagesock);
660   Online[i].messagesock=0;
661
662   pthread_exit(NULL);
663 }
664
665 /*--------------------------------------------------------------------
666  * send_list - send list to client
667  *--------------------------------------------------------------------
668  */
669 void send_list(int fd)
670 {
671   int i;
672   char temp[LINELEN+1];
673
674   Send_Message(fd,"#list");
675   strcpy(temp," ");
676   usleep(1);
677
678   for(i=0;i<QLEN;i++)
679   if((strcmp(Online[i].name,"")!=0) && (Online[i].sock!=fd))
680        sprintf(temp,"%s%s ",temp,Online[i].name);
681
682   Send_Message(fd,temp);
683 }
684
685 /*--------------------------------------------------------------------
686  * find_name - check name which had online or not
687  *--------------------------------------------------------------------
688  */
689 int find_name(char *string,int sock)
690 {
691   int i;
692
693    for(i=0;i<QLEN;i++)
694     {printf("%d\n",i);
695      if((strcmp(Online[i].name,string)==0))
696       {if((Online[i].sock)!=sock)
697         return i;/*find*/
698        else
699           return -1;/*the name is yourself*/
```

```
700
701         }
702       }
703
704     return -2;
705 }
706
707 /*---------------------------------------------------------------------
708  * find_id - find id by socket
709  *---------------------------------------------------------------------
710  */
711 int find_id(int sock)
712 {
713  int i;
714
715  for(i=0;i<QLEN;i++)
716       if(sock==Online[i].sock)
717            break;
718
719  // printf("ID: %d\n",i);
720  return i;
721 }
722
723 /*---------------------------------------------------------------------
724  * get_client_address - get clients's ip and port and send to them
725  *---------------------------------------------------------------------
726  */
727 void get_client_address(void )
728 {      struct  sockaddr_in fsin; /* the from address of a client
729  */
730  struct  sockaddr_in temp1;     /* client1 address*/
731  struct  sockaddr_in temp2;     /* client2 address*/
732  char *service = "7777";  /* service name or port number     */
733  char buf[50],temp[50];
734  int  sock,i,csock1,csock2;              /* server socket
735  */
736  int  alen;               /* from-address length        */
737  struct in_addr add;
738  sock = passiveUDP(service);
739       alen = sizeof(fsin);
740
741
742       printf("waiting....get address\n");
743       recvfrom(sock, buf, sizeof(buf), 0,(struct sockaddr *)&fsin,
                 &alen);
744
745       puts(buf);
746       i=find_name(buf,-1);
747       strcpy(temp,buf);/*save client1's name*/
748
749       add.s_addr=fsin.sin_addr.s_addr;
750       printf("ip %s port
                  %d\n",inet_ntoa(add),ntohs(fsin.sin_port));
751       temp1.sin_addr.s_addr=fsin.sin_addr.s_addr;
752       temp1.sin_port=fsin.sin_port;
753
754        /*tell client1 that server get ip*/
755        printf("send %s ok ,socket:%d
              \n",Online[i].name,Online[i].sock);
756        Send_Message(Online[i].sock,"#get_v_message");
757        csock1=Online[i].sock;
758
759
760       printf("waiting....get address2\n");
761       /*do not take the same name*/
762       do{
763       recvfrom(sock, buf, sizeof(buf), 0,(struct sockaddr *)&fsin,
                  &alen);
764       puts(buf);
765       }while(strcmp(temp,buf)==0);
766
767
768       i=find_name(buf,-1);
769
770     add.s_addr=fsin.sin_addr.s_addr;
771       printf("ip %s port
                  %d\n",inet_ntoa(add),ntohs(fsin.sin_port));
772     temp2.sin_addr.s_addr=fsin.sin_addr.s_addr;
773       temp2.sin_port=fsin.sin_port;
774
775        /*tell client2 that server get ip*/
776        printf("send %s ok ,socket:%d
                  \n",Online[i].name,Online[i].sock);
777        Send_Message(Online[i].sock,"#get_v_message");
778        csock2=Online[i].sock;
779
780
781       printf("send address.port to client1\n");
782       sprintf(buf,"#ip %d
          %d",htonl(temp1.sin_addr.s_addr),htons(temp1.sin_port));
783
784       usleep(100000);
785       Send_Message(csock2,buf);
786       printf("send address.port to client2\n");
787
788       sprintf(buf,"#ip %d
     %d",htonl(temp2.sin_addr.s_addr),htons(temp2.sin_port));
789       usleep(100000);
790       Send_Message(csock1,buf);
791
```

```
792         close(sock);
793
794 }
795
796 /*---------------------------------------------------------------------
797  *  Send_Message - send command(message) to client
798  *---------------------------------------------------------------------
799  */
800 void Send_Message(int fd,char *mgs )
801 {
802
803    char buf[LINELEN+1];
804    int result;
805
806    strcpy(buf,mgs);
807      printf("Send to %d: ",fd);
808      puts(buf);
809      result=write(fd, buf, strlen(buf));
810
811         if(result==-1)
812          client_logout(fd,&AFDS);
813 }
814
815 /*---------------------------------------------------------------------
816  * broadcast_message -  broadcast message
817  *---------------------------------------------------------------------
818  */
819 void  broadcast_message(int fd,char *buf,fd_set *afds)
820 {   int s;
821
822  for (s=(MSOCK+1); s<FD_size; s++)
823       if (s != fd && FD_ISSET(s, afds))
824       Send_Message(s,buf);
825 }
826
827 /*---------------------------------------------------------------------
828  * client_logout -  client logoff
829 *---------------------------------------------------------------------
830  */
831 void client_logout(int fd,fd_set *afds)
832 {
833   int i;
834   char temp[50];
835
836   printf("socket: %d logoff\n",fd);
837
838   i=find_id(fd);
839
840   client_cancel_voip(i);
841   client_cancel_leave_message(i);
842
843   if(Online[i].answer==2)
844    Online[i].answer=0;
845
846   sprintf(temp,"#logout %s",Online[i].name);
847   broadcast_message(fd,temp,afds);
848
849   strcpy(Online[i].name,"");/*erase his name*/
850   Online[i].sock=0;
851   Online[i].state=0;
852
853   /*close socket*/
854   close(fd);
855   FD_CLR(fd, afds);
856
857 }
858
859 /*---------------------------------------------------------------------
860  * client_cancel_voip -  cancel voip
861  *---------------------------------------------------------------------
862  */
863 void client_cancel_voip(int id)
864 {
865   int cs;
866
867   cs=Online[id].talksock;
868
869   if(cs)
870        {printf("tell %d to stop voip\n",cs);
871         Send_Message(cs,"#stop_voip" );
872
873      /*clean player talksock*/
874      Online[cs].talksock=0;
875      Online[id].talksock=0;
876     }
877 }
878
879 /*---------------------------------------------------------------------
880  * client_cancel_leave_message -  cancel messagesock
881  *---------------------------------------------------------------------
882  */
883 void client_cancel_leave_message(int id)
884 {
885    if(Online[id].messagesock)
886    {
887   puts("clean messagesock");
888   cancel_thread(Online[id].thread);
889     close(Online[id].messagesock);
890   Online[id].messagesock=0;    /*clean player messagesock*/
891    }
892
```

```
}

/*--------------------------------------------------------------------
 *  cancel_thread
 *--------------------------------------------------------------------
 */
void cancel_thread(pthread_t thread)
{
   int res;
 res=pthread_cancel(thread);
     if(res != 0)
    {printf("Thread cancel failed!!!\n");
     exit(1);
      }

}
```

```
// version: 0.12 for audio.h 0.12

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/soundcard.h>

#include "audio.h"

#define MIXER "/dev/mixer"
#define AUDIO "/dev/dsp"

int audio;

int vol_v = DEFAULT_VALUE ;
int vol_p = DEFAULT_VALUE ;
int vol_m = DEFAULT_VALUE ;

int count_rec;
int count_play;

char *names[SOUND_MIXER_NRDEVICES] = SOUND_DEVICE_NAMES;

// /dev/mixer: query/check capbilities, select record source, set
                default volume
// /dev/dsp: set duplex, sampling size, num of channel, sampling rate
int audio_init()
{
   int mixer;
   int caps, recmask, recsrc;

   int dsp_samplesize = SAMPLESIZE; // AFMT_S16_LE = 16, AFMT_U8 = 8
   int dsp_stereo = STEREO;  /* 0=mono, 1=stereo */
   int dsp_speed = SPEED;

   // open mixer ************************************
   if( (mixer = open(MIXER, O_RDWR, 0)) == -1 ) {
      perror(MIXER);
      exit(1);
   }

   // query/check capbilities
      // ...

   // record source
   if(ioctl(mixer, SOUND_MIXER_READ_RECMASK, &recmask) == -1) {
      perror("SOUND_MIXER_READ_RECMASK");
      exit(1);
   }
   if( !((1 << SOUND_MIXER_MIC) & recmask) ) {
      perror("SOUND_MIXER_MIC");
      puts("not support mic");
      exit(1);
```

```
    }

    if(ioctl(mixer, SOUND_MIXER_READ_RECSRC, &recsrc) == -1) {
       perror("SOUND_MIXER_READ_RECSRC");
       exit(1);
    }
    if( !((1 << SOUND_MIXER_MIC) & recsrc) ) {
       puts("record source is not MIC. try to change.........");

       recsrc = (1 << SOUND_MIXER_MIC);
       if (ioctl(mixer, SOUND_MIXER_WRITE_RECSRC, &recsrc) == -1) {
          perror("SOUND_MIXER_WRITE_RECSRC");
          exit(1);
       }
       if( ((1 << SOUND_MIXER_MIC) & recsrc) )
          puts("ok! now record source is MIC!");
    }
    else
       puts("support mic");

    // set default volume
    volume_vol(vol_v) ;  // -> read default volume from conf file
    volume_pcm(vol_p) ;
    volume_mic(vol_m) ;

    close(mixer);

    // open dsp ****************************************
    if( (audio = open(AUDIO, O_RDWR, 0)) == -1 ) {
       perror(AUDIO);
       exit(1);
    }

    // set full duplex
    if( ioctl(audio, SNDCTL_DSP_SETDUPLEX, 0) == -1 ) {
       perror("SNDCTL_DSP_SETDUPLEX");
       //puts("WARNING: not support full duplex");
       //exit(1);
    }

    ioctl(audio, SNDCTL_DSP_GETCAPS, &caps);
    if( caps & DSP_CAP_DUPLEX)
       puts("support full duplex!");
    else
       puts("WARNING: not support full duplex");

    // set parameter
    if( ioctl(audio, SNDCTL_DSP_SAMPLESIZE, &dsp_samplesize) == -1 )
{
       perror("SNDCTL_DSP_SAMPLESIZE");
       exit(1);
    }
    if( ioctl(audio, SNDCTL_DSP_STEREO, &dsp_stereo) == -1 ) {
       perror("SNDCTL_DSP_STEREO");
       exit(1);
    }
    if( ioctl(audio, SNDCTL_DSP_SPEED, &dsp_speed) == -1 ) {
       perror("SNDCTL_DSP_SPEED");
       exit(1);
    }
    printf("samplesize = %d | stereo = %d | speed = %d\n",
          dsp_samplesize, dsp_stereo, dsp_speed);

    puts("audio_init finished");
    return 0;
}

// close dsp, save volume as default
int audio_exit()
{
    // save volume as default
      //...

    close(audio);

    puts("audio_exit finished");

    return 0;
}

int play(unsigned char *data, int size)
{
    int n;

 //  printf("playing %d\n", ++count_play);

    if( (n = write (audio, data, size)) == -1) {
       perror("PLAY");
       exit(1);
    }

    //printf("play over\n");

    return n;
}

int rec(unsigned char *data, int size)
{
    int n;

//   printf("recoding %d\n", ++count_rec);
```

```
    if( (n = read (audio, data, size)) == -1) {
       perror("RECORD");
       exit(1);
    }

    //printf("recode over\n");

    return n;
}

// set volume
int volume(int vol, int mix_dev)
{
    int mixer;

    if( (mixer = open(MIXER, O_RDWR, 0)) == -1 ) {
       perror(MIXER);
       exit(1);
    }

    if(vol < 0)
       vol = 0;
    if(vol > 100)
       vol = 100;

    vol = vol | (vol << 8); // left + right channel

    if( ioctl(mixer, MIXER_WRITE(mix_dev), &vol) == -1) {
       if (errno == EINVAL)
          fprintf(stderr, "Invalid mixer control %d\n", mix_dev);
       else
          perror("MIXER_WRITE");
       exit(1);
    }

    printf("%s vol: %d %d\n", names[mix_dev], (vol & 0xff), (vol >> 8) );
    close(mixer);

    return (vol & 0xff);
}
```

```
// version: 0.121 for audio.c 0.12

1
2   // header file
3   #include <sys/soundcard.h>
4
5   // define
6   #define BUFSIZE 800
7   #define DEFAULT_VALUE 70
8   #define SPEED 8000
9   #define SAMPLESIZE 16
10  #define STEREO   1  /*0=mono 1=stereo*/
11
12  // var
13  extern int vol_v;
14  extern int vol_p;
15  extern int vol_m;
16
17  // /dev/mixer: query/check capbilities, select record source, set
    default volume
18  // /dev/dsp: set num of channel, sampling size, sampling rate
19  extern int audio_init();
20
21  // close dsp, save volume as default
22  extern int audio_exit();
23
24  // read/write dsp
25  extern int play(unsigned char *data, int size);
26  extern int rec(unsigned char *data, int size);
27
28  // don't use volume() directly!
29  extern int volume(int vol, int mix_dev);
30
31  // 0.121 separate volume 0 from mute function
32  // set volume (0.121)
33  #define volume_vol(v) vol_v=volume(v, SOUND_MIXER_VOLUME)
34  #define volume_pcm(v) vol_p=volume(v, SOUND_MIXER_PCM)
35  #define volume_mic(v) vol_m=volume(v, SOUND_MIXER_MIC)
36
37  // set mute. notice: no variable stores the state
38  #define mute_vol() volume(0, SOUND_MIXER_VOLUME)
39  #define mute_pcm() volume(0, SOUND_MIXER_PCM)
40  #define mute_mic() volume(0, SOUND_MIXER_MIC)
41
```

```
/* connectTCP.c - connectTCP */

1
2   int   connectsock(const char *host, const char *service,
3         const char *transport);
4
```

```
/*------------------------------------------------------------------------
 * connectTCP - connect to a specified TCP service on a specified host
 *------------------------------------------------------------------------
 */
int
connectTCP(const char *host, const char *service )
/*
 * Arguments:
 *      host    - name of host to which connection is desired
 *      service - service associated with the desired port
 */
{
 return connectsock( host, service, "tcp");
}
```

```
/* connectUDP.c - connectUDP */

int   connectsock(const char *host, const char *service,
      const char *transport);

/*------------------------------------------------------------------------
 * connectUDP - connect to a specified UDP service on a specified host
 *------------------------------------------------------------------------
 */
int
connectUDP(const char *host, const char *service )
/*
 * Arguments:
 *      host    - name of host to which connection is desired
 *      service - service associated with the desired port
 */
{
 return connectsock(host, service, "udp");
}
```

```
/* connectsock.c - connectsock */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>


#ifndef    INADDR_NONE
#define    INADDR_NONE    0xffffffff
#endif/* INADDR_NONE */

extern int errno;

int   errexit(const char *format, ...);

/*------------------------------------------------------------------------
 * connectsock - allocate & connect a socket using TCP or UDP
 *------------------------------------------------------------------------
 */
int
connectsock(const char *host, const char *service, const char
*transport )
/*
 * Arguments:
 *      host     - name of host to which connection is desired
 *      service   - service associated with the desired port
 *      transport - name of transport protocol to use ("tcp" or "udp")
 */
{
 struct hostent *phe;     /* pointer to host information entry
 */
 struct servent *pse;     /* pointer to service information entry
 */
 struct protoent *ppe;    /* pointer to protocol information entry*/
 struct sockaddr_in sin;  /* an Internet endpoint address
 */
 int  s, type;  /* socket descriptor and socket type    */


 memset(&sin, 0, sizeof(sin));
 sin.sin_family = AF_INET;

   /* Map service name to port number */
 if ( (pse = getservbyname(service, transport)) )
      sin.sin_port = pse->s_port;
 else if ( (sin.sin_port = htons((u_short)atoi(service))) == 0 )
      errexit("can't get \"%s\" service entry\n", service);

   /* Map host name to IP address, allowing for dotted decimal */
```

```
  if ( (phe = gethostbyname(host)) )
       memcpy(&sin.sin_addr, phe->h_addr, phe->h_length);
  else if ( (sin.sin_addr.s_addr = inet_addr(host)) == INADDR_NONE )
       errexit("can't get \"%s\" host entry\n", host);

    /* Map transport protocol name to protocol number */
  if ( (ppe = getprotobyname(transport)) == 0)
       errexit("can't get \"%s\" protocol entry\n", transport);

    /* Use protocol to choose a socket type */
  if (strcmp(transport, "udp") == 0)
       type = SOCK_DGRAM;
  else
       type = SOCK_STREAM;

    /* Allocate a socket */
  s = socket(PF_INET, type, ppe->p_proto);
  if (s < 0)
       errexit("can't create socket: %s\n", strerror(errno));

    /* Connect the socket */
  if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0)
        {errexit("can't connect to %s.%s: %s\n", host,
service,strerror(errno));
        return -1;
       }
  return s;
 }
```

```
/*------------------------------------------------------------------------
 * passiveTCP - create a passive socket for use in a TCP server
 *------------------------------------------------------------------------
 */
int
passiveTCP(const char *service, int qlen)
/*
 * Arguments:
 *      service - service associated with the desired port
 *      qlen    - maximum server request queue length
 */
{
 return passivesock(service, "tcp", qlen);
}
```

```
/* passiveTCP.c - passiveTCP */

 int   passivesock(const char *service, const char *transport,
       int qlen);

```

```
/* passiveUDP.c - passiveUDP */

 int   passivesock(const char *service, const char *transport,
       int qlen);

/*------------------------------------------------------------------------
 * passiveUDP - create a passive socket for use in a UDP server
 *------------------------------------------------------------------------
 */
int
passiveUDP(const char *service)
/*
 * Arguments:
 *      service - service associated with the desired port
 */
{
 return passivesock(service, "udp", 0);
}

```

```
/* passivesock.c - passivesock */

 #include <sys/types.h>
 #include <sys/socket.h>

```

```
#include <netinet/in.h>

#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <errno.h>

extern int errno;

int   errexit(const char *format, ...);

u_short    portbase = 0;       /* port base, for non-root servers
 */

/*------------------------------------------------------------------
 * passivesock - allocate & bind a server socket using TCP or UDP
 *------------------------------------------------------------------
 */
int
passivesock(const char *service, const char *transport, int qlen)
/*
 * Arguments:
 *     service   - service associated with the desired port
 *     transport - transport protocol to use ("tcp" or "udp")
 *     qlen      - maximum server request queue length
 */
{
 struct servent *pse;     /* pointer to service information entry
 */
 struct protoent *ppe;    /* pointer to protocol information entry*/
 struct sockaddr_in sin;  /* an Internet endpoint address
 */
 int  s, type;  /* socket descriptor and socket type    */

 memset(&sin, 0, sizeof(sin));
 sin.sin_family = AF_INET;
 sin.sin_addr.s_addr = INADDR_ANY;

   /* Map service name to port number */
 if ( (pse = getservbyname(service, transport)) )
      sin.sin_port = htons(ntohs((u_short)pse->s_port)
           + portbase);
 else if ( (sin.sin_port = htons((u_short)atoi(service))) == 0 )
      errexit("can't get \"%s\" service entry\n", service);

   /* Map protocol name to protocol number */
 if ( (ppe = getprotobyname(transport)) == 0)
      errexit("can't get \"%s\" protocol entry\n", transport);

   /* Use protocol to choose a socket type */
 if (strcmp(transport, "udp") == 0)
      type = SOCK_DGRAM;
 else
      type = SOCK_STREAM;

   /* Allocate a socket */
 s = socket(PF_INET, type, ppe->p_proto);
 if (s < 0)
      errexit("can't create socket: %s\n", strerror(errno));

   /* Bind the socket */
 if (bind(s, (struct sockaddr *)&sin, sizeof(sin)) < 0)
      errexit("can't bind to %s port: %s\n", service,
           strerror(errno));
 if (type == SOCK_STREAM && listen(s, qlen) < 0)
      errexit("can't listen on %s port: %s\n", service,
           strerror(errno));
 return s;
}
```

註：connectsock.c connectTCP.c connectUDP.c passivesock.c passiveTCP. passiveUDP.c 出自於 Internetworking with TCP/IP Vol 3: Client-Server Programming and Applications, BSD Version, 2nd Ed, by D. Comer and D. Stevens, Prentice-Hall, 1996.

程式碼載點
http://140.134.27.100/~sunggekun/PO/voicemessenger-1.0.tar.bz2

參考文獻

- 書籍：

[3.1] Karim Yaghmour 著，蔣大偉譯，建構嵌入式 Linux 系統(Building Embedded Linux Systems)。O'REILLY，2004 年 04 月。

[3.2.2、3.2.3、3.2.4]HyBus Co. Ltd., *Embedded System Application using X-Hyper255B-TKUIII based on Intel PXA255*(v1.1). Korea: HyBus Co. Ltd., 2004.

[4.2] Matt Welsh、Matthias Kalle Dalheimer、Terry Dawson、Lar Kaufman 著，周敏祥譯，Linux 技術手冊 (Running Linux, 4/e)。O'REILLY，2003 年 12 月。

[5]Richard Stones、Neil Matthew 著，江俊龍譯，Linux 程式設計教學手冊(第三版)。臺北市：碁峰資訊股份有限公司，2004 年 8 月。

[5.3]陳俊宏，Embedded Linux 嵌入式系統實作演練(初版)。臺北市：學貫行銷股份有限公司，2004 年 3 月。

- 網頁：

[2.1]Instant messaging:
http://en.wikipedia.org/wiki/Instant_messenger

[2.3-1]智邦與 Skype 攜手合作引領免費網路電話行動自如：

http://www.accton.com.tw/homepage/main1/press/2005/2005_press_16.htm

[2.3-2]完全掌握 Skype 的勝者之道：

http://office.digitimes.com.tw/ShowNews.aspx?zCatId=53J

http://www.accton.com.tw/homepage/main3/Datasheet/DS_VM1188T.pdf

[3.2.5-1、5.3-1] microwindows：

http://microwindows.org/

[4.3-1]Linux Kernel source：

http://www.kernel.org/pub/linux/kernel/

[4.3-2]鳥哥的 Linux 私房菜：

http://linux.vbird.org/

[4.3-3]Linux 2.4 kernel faq：

http://www.faqs.org/docs/kernel_2_4/lki.html#toc1

[4.3-4]old Linux：

http://www.oldlinux.org/index_cn.html

4.3-5]Linux cross reference：

http://lxr.linux.no/

[4.3-6]Debian 論壇：

http://moto.debian.org.tw/index.php

[4.3.2-1]Wireless Howto：

http://www.tldp.org/HOWTO/Wireless-HOWTO.html

[4.3.2-2]airsnort：

http://airsnort.shmoo.com

[4.3.2-3]kismet：

http://www.kismetwireless.net/

[4.3.2-4]goonda：

http://www.goonda.org/

[4.3.2-5]orinoco：

http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Orinoco.html

[4.4.1-1]Filesystem Hierarchy Standard(FHS):

http://www.pathname.com/fhs/

[4.4.1-2]LANANA Linux Device List:

http://www.lanana.org/docs/device-list/

[5.2] Wikipedia,Tthe free encyclopedia

http://en.wikipedia.org/wiki/Main_Page

[5.2.2-1] Network Address Translation

http://www.vicomsoft.com/knowledge/reference/nat.html

http://turtle.ee.ncku.edu.tw/~tung/nat/nat.htm

[5.2.2-2]為什麼需要 IPv6？

http://www.ipv6.org.tw/NDHU/article/20020826.htm

[5.2.3]OSS Programmer's guide v1.1:

http://www.opensound.com/pguide/oss.pdf

[6.2.1]網路電話產品趨勢與應用：

http://www.cqinc.com.tw/grandsoft/cm/091/afo8915.htm

[6.2.2]電源管理

http://www.eettaiwan.com/CAT_675763.HTM

http://www.gentoo.org/doc/zh_tw/power-management-guide.xml