逢　甲　大　學

資訊工程學系專題報告

JavaMail與資料庫技術開發系統

學　　生　：　陳峻儀（四甲）
賴明德（四甲）
指導教授　：　李維斌 老師

中華民國九十二年十二月

目　　錄

圖表目錄

摘要

時代在進步，科技不斷地提昇，電腦從研究、應用，到與人類生活結合。資訊的傳遞一天比一天快速。有效的管理也使資訊更具有價值，有規劃地運用資訊，才能使我們跨步向前邁進。資訊在人的生活中，以各種不同的型式存在著，其存在的結構亦不盡相同。而「資料庫管理系統」對它肩負著儲存、管理與運用之重責大任。當我們為了某種特定目的，將相關資料逐日、逐筆或整批儲存於資料庫中，再依各種不同的需要，將這些資料經過運算綜合或分類產生對我們有用的資訊，藉以輔助分析、判斷和決策。

資料庫管理系統理論明白定義著幾個重要的特性：

1. 對於實體資料結構（file、index、....）的維護功能。
2. 提供對資料存取（增加、刪除、修改等等）之程式語言界面。
3. 保證資料的整合性及保密性。
4. 控制多使用者同步存取之環境。
5. 具備資料庫備份及復原能力。
6. 對於系統資料字典的維護功能。
7. 確保資料之獨立性。
8. 提供資料庫管理者管理工具的使用。

在學術界不斷地推展理論，工業界不斷地研究開發下，資料庫理論架構越來越成熟，而資料庫管理系統產品在一代接著一代的更替下，越來越能為使用者廣泛應用。資料模式從階層式 (Hierachical)、網路式 (Network)、到關連式(Relational)，以至於尚在混沌下的物件導向式資料庫系統，理論與技術每每向前邁進一步。雖然每個在市場上的資料庫系統產品之開發技術不同， 但都得朝向前述的八大特性目標看齊。二十世紀八O年代，資料庫管理系統在工業界的簇擁下，產品陸續為許多企業所鍾愛。IBM 的 DB2 在其硬體的帶領下，佔領了市場，到九0年代，Sybase、Oracle、Informix 在開放操作系統 (Open Operating System) 的世界裏，形成三足鼎立的局面。從專屬系統(Proprietary System) 到開放系統，資料庫管理系統一直在各式各樣的資訊應用系統下扮演著非常重要的角色。然而在即將

邁入二十一世紀之時，第四代程式語言（4th Gerneration Language；4GL） 改變了應用系統的開發環境，把應用系統從文字模式（TextMode）帶進了圖形模式（Graphic Mode)。隨後，微軟（Microsoft）提出了開放式資料庫連接界面 （Open DataBase Connectivity；ODBC)，同時把資料庫管理系統推進了開放的世界（圖一說明開放系統的整體架構)。而我們也將可預期二十一世紀是開放式資料庫管理系統（Open DataBase Management System）佔領資訊系統 （Information System）的世紀。

第一章　導論

1.1 動機

在網路的世界裏，Email 是 Internet 上的重要資訊服務方式，它為世界各地的 Internet 用戶提供了一種極為快速、簡便和經濟的通信以及交換資訊的方法。與常規信函相比，Email 非常迅速，它把資訊傳遞時間由幾天、十幾天減少到幾分鐘；而且 Email 使用非常方便和自由，既寫既發，省去了粘貼郵票和跑郵局的煩惱；與電話相比，Email 的使用非常經濟，傳輸幾乎是免費的。正是由於這些優點,Internet 上數以億計的用戶都有自己的 Email 地址，Email 也成為利用率最高的 Internet 應用。 Email 地址是以區域為基礎的地址，如 d8888888@fcu.edu.tw 就是逢甲大學學生所使用的 Email 位址。除了為用戶提供基本的電子郵件服務，還可以使用 Email 給郵寄列表（Mailing List）中的每個註冊成員分發郵件以及提供電子期刊。Email 的傳遞是一個標準化的簡單郵件傳輸協議 SMTP 來完成的。SMTP 是 TCP/IP 協定的一部分，它概述了電子郵件的資訊格式和傳輸處理方法。使用 Email 不僅可以發送和接收文字，有些專用 Email(Eudora)收發工具軟體還可以收發圖像、聲音、執行程式等各種類型的檔案，所以根據以上我們利用 javamail 開發了 webmail 與 sybase 當作 Email 的儲存媒體。

1.2 目的

Webmail 是現在一種常見到的收送信的介面，我們根據 Java 公司所推行的一套 JavaMail API 且搭配了 Jsp（Java server page）來建構我們的 Webmail，且 JavaMail 是一種可選的、能用於讀取、編寫和發送電子訊息。您可使用這種 API 來創造郵件的格式，它類似於 Eudora、 Pine 及 Microsoft Outlook 這些郵件程式。

本系統的功能有包含一般 Webmail 的基本功能，可分為信件處理的前端和後端，前端包含了收信、寄信、刪除信件等功能，比較不一樣的地方像寄信方面，你可以選擇外部寄信或內部寄信，那何謂外部寄信？也就是你可以透過別的 Mailserver 來寄信，那內部寄信就是透過本系統所架設的 Mailserver 來送信，同理收信也是一樣。至於後端我們採用本校一樣的資料庫系統 Sybase 來儲存 Email 資料。

1.3 系統簡介

1.3.1 webmail介紹

1. 在 WebMail 產生信件時，
   WebMail 介面 → 以 JavaMail 直接在 WebServer 上寄出。
2. 在 WebMail 收信時，
   信件 → MailServer(FreeBSD) → lex 把 MIME 標準格式信轉換成存 Database 的格式 → jConnector 把信存在資料庫。

1.3.2 Email Parser介紹

首先Parser會去mail server收信的資料夾，利用掃描目錄的函式，一一的把信件所在位置讀出，當讀完一個信件所在的絕對位置後，會經由Parser變成3個.eml然後再分別地存入資料庫，那Psrser如何變成3個eml了？他會根據lex的rule下去判斷，判斷到哪些欄位就存到相對應的eml，以方便在webmail顯示，且當有信件要寄出去的時候，會再把信件組合，變成一封完整的email，也就是說還原成原本的樣子來寄出。

1.3.3 Sybase儲存欄位規格

把Cparser所產生的3個eml經由Jconnect分別讀取各eml的欄位，並在Sybase建立3個tables來儲存欄位。

第二章　系統需求背景

2.1 Tomcat的簡介

2.1.1 Tomcat的緣由

Tomcat是一個開放的程式碼，它是用Java語言來發展的一套Server，運用在servlet和JSP Web，且Tomcat Server是根據servlet和JSP規範來執行操作，且它也延續著Apache-Jakarta範疇來進行開發與測試，所以比大多數商業應用軟體Server要好。

2.1.2 架設Tomcat方法

需要j2sdk+jakarta-tomcat這兩種軟體，以下介紹在win2000，Linux，Freebsd上的架設

win2000 底下：

執行 j2sdk-1_4_0_01-windows-i586.exe，安裝至 c:\jdk(可自行修改)，安裝完會要求重新啟動，請還不要重新啟動。將 jakarta-tomcat-4.0.5.zip 解壓縮至 c:\tomcat 下。

1. 點選　開始→設定→控制台→系統→進階→環境變數
2. 找到系統變數 PATH，按「編輯」鈕,在最後面加上 C:\JDK\BIN;，然後按確定。
3. 按「新增」，變數名稱輸入 JAVA_HOME，變數值輸入 C:\JDK，然後按確定。
4. 按「新增」，變數名稱輸入 CATALINA_HOME，變數值輸入 C:\TOMCAT，然後按確定。
5. 按「新增」，變數名稱輸入 CLASSPATH，變數值輸入 C:\JDK\LIB，然後按確定。
6. 重新開機，讓上述設定生效，之後執行 TOMCAT，進入 msdos 模式，執行\tomcat\bin\startup 便可以啟動 TOMCAT，如要關閉 TOMCAT，執行\tomcat\bin\shutdown

7. 測試 TOMCAT 是否啟動起來，可以開啟瀏覽器，在網址列輸入 http://localhost:8080，如果可以看得到網頁，表示安裝設定都已完成，可以開始開發程式了

LINUX 底下：

但注意一點，在不同版本的 OS，Java 所使用的版本不一樣。今以 j2sdk-1_4_0_01 和 jakarta-tomcat-4.0.5 為範例。

1. chmod 755 j2sdk-1_4_0_01-linux-i586.bin，
2. 輸入./j2sdk-1_4_0_01-linux-i586.bin，會自動解壓縮，解完，將它移至/usr/jdk(可自行修改)。再來將 jakarta-tomcat-4.0.5.tar.gz 這個檔 tar 開，將它移至 /usr/tomcat(可自行修改)
3. 設定執行環境：以 root 身份修改/etc/profile 檔，在最後面加入以下幾行： PATH=$PATH:/usr/jdk/bin export JAVA_HOME=/usr/jdk
   export CATALINA_HOME=/usr/tomcat
   export CLASSPATH=$JAVA_HOME/lib
4. 執行 source /etc/profile，讓剛才所做的修改馬上生效
5. 啟動 TOMCAT，執行/usr/tomcat/bin/startup.sh，如要關閉 TOMCAT，執行/usr/tomcat/bin/shutdown.sh
6. 測試 TOMCAT 是否啟動，可以開啟瀏覽器，在網址列輸入 http://Linux 機器 IP:8080，如果可以看得到網頁，表示安裝設定都已完成，可以開始開發程式了

FreeBSD 底下:

跟 linux 設定差不多，但環境變數變為

```
# For JAVA jdk1.3.1
export PATH=$PATH:/usr/local/jdk1.3.1/bin
export JAVA_HOME=/usr/local/jdk1.3.1
export CLASSPATH=/usr/local/jdk1.3.1/lib/tools.jar
:/usr/local/share/java/classes:/usr/local/jakarta-tomcat-4.
0/bin/startup.sh
```

2.1.3 Tomcat的目錄架構

目錄名稱	描述
Bin	包含啟動/關閉 Tomcat
Conf	包含不同的配置文件,包括 server.xml(Tomcat 的主要配置文件)和為不同的 Tomcat 配置的 web 應用設定值的文件 web.xml
Doc	包含各種 Tomcat 文檔
Lib	包含 Tomcat 使用的 jar 文件，在 unix 平台此目錄下的任何文件都被加到 Tomcat 的 classpath 中
Logs	Tomcat 擺放日誌文件的地方
Src	ServletAPI 程式碼文件
webapps	包含 web 項目範例

表一

此外你可以 ，建立如下目錄:

Work	Tomcat 自動生成,放置 Tomcat 運行時的臨時文件(如編譯後的 JSP 文件),如在 Tomcat 運行時刪除此目錄.JSP 頁面將不能運行。
classes	你可以創建此目錄來添加一些附加的類別路徑中，任何你加到此目錄中的類別都可在 Tomcat 的類別路徑中找到本身。

表二

2.1.4 Tomcat與Jserv區別

很多人會常常誤認為Tomcat是Jserv，但其實不是這樣的，因為Jserv是Servlet API2.0兼並與Apache一起使用的Server.但Tomcat是一個完全重寫的並與Servlet API2.2和JSP1.1兼容的Server。兩這相同之處是在，Tomcat使用了一些為Jserv而寫的程式代碼,特別是Jserv的Apache的port。而且 Tomcat 未來會取代 JServ, 成為 Apache 主要的 Servlet & JSP Engine. Tomcat 在設計上是以獨立的 Server 執行, 而不像 Jserv 是附在 Apache 中。

2.2 JSP 和 Javamail 與 EWA 構造說明

2.2.1 Jsp 簡介

JSP 是 Java Server Pages 的簡寫。JSP 技術能讓 Web 開發員和網頁設計員快速地開發容易維護的動態 Web 主頁。用 JSP 開發的 Web 應用是跨平台的，即能在 Linux 下運行，也能在其他操作系統上運行。JSP 技術使用 Java 編程語言編寫類 XML 的 tags 和 scriptlets，來封裝產生動態網頁的處理邏輯。網頁還能通過 tags 和 scriptlets 訪問存在服務端的資源(例如 JavaBesns)的應用邏輯。JSP 將網頁邏輯與網頁設計和顯示分離，支持可重用的基組件的設計，使基 Web 的應用程序的開發變得迅速和容易。JSP 技術是 Servlet 技術的擴展。Servlet 是平台無關的，100%純 Java 的 Java 服務端組件。

2.2.2 Javamail 簡介

JavaMail API 是一個 Java 套件，用於閱讀、編寫及傳送郵件訊息（包括附件）。您能用它建立基於標準的郵件客戶端，它配置了各種網際網路信件協定包括 SMTP、POP、IMAP 和 MIME，還包括相關的 NNTP、S/MIME 及其它協定。此 API 自然的劃分為兩個部件。第一部件著眼於獨立於使用協定的傳送、接收及管理訊息，而第二部件則著眼於協定的特定用途。本教程的目的在於展示如何利用 API 的第一部分，並不試圖涉及協定供應商。核心 JavaMail API 由七個類別組成 — Session、Message、 Address、Authenticator、Transport、Store 及 Folder — 它們都來自 javax.mail、即 JavaMail API 頂級套件。我們用這些類別完成了大量常見的郵件工作，包括傳送訊息、檢索訊息、刪除訊息、認證、回復訊息、轉信訊息、管理附件、處理基於 HTML 文件格式的訊息以及搜尋或過濾信件清單。

2.2.2.1 javamail 內部構造

一般而言，光只有javamail和提供Jsp環境的Tomcat Server這樣是不夠的，也就是說這樣Javamail不能夠啟用，因為如果你要使用Javamail來建立你的webmail伺服器，應該還需要一個JAF（JavaBeans Activation Frameword）屬於JavaBean的套件，它也是由Sun Microsystems免費的提供，這樣javamail才能順利的啟用他所有的功能，其中在JavaMail API中定義了七個主要元件

1. Session

Session 類別定義了一個基本 mail session。所有其它類別都是經由這個 session 才得以生效。Session 物件用 java.util.Properties 物件獲取訊息，如信件伺服器、使用者名稱、密碼及整個應用程式中共享的其它訊息。此類別的 constructors 為 private。您可以得到單個預設 session，它能用 getDefaultInstance() 方法被共享：

```
Properties props = new Properties();

  // fill props with any information
Session session = Session.getDefaultInstance(props, null);
```

或者，您還可以用 getInstance() 建立一個獨立的 session：

```
Properties props = new Properties();
// fill props with any information
Session session = Session.getDefaultInstance(props,
null);
```

對於這兩種情況，null 參數都是 Authenticator 物件（在這次沒有使用）。對於大多數情況，共享的 session 已經夠用了，即

使要處理多個使用者信箱的信件 session 也一樣。您可以在通信過程中稍後的步驟加入使用者名稱和密碼組合,讓一切保持獨立。

2. Message

一旦獲得 Session 物件，就可以繼續建立要傳送的訊息。這由 Message 類別來完成。因為 Message 是個抽象類別，您必需用一個子類別，多數情況下為 javax.mail.internet.MimeMessage。MimeMessage 是個能理解 MIME 類型和 header 的信件訊息，正如不同 RFC 中所定義的。雖然在某些 header 網域非 ASCII 字元也能被編譯，但 Message header 只能被限制為用 US-ASCII 字元。

要建立一個 Message，請將 Session 物件傳遞給 MimeMessage 的 constructor

```
MimeMessage message = new MimeMessage(session);
```

注意：還存在其它 constructors ，如用按 RFC822 格式的 input streams 來建立訊息。一旦獲得訊息，您就可以設定各個部分，因為 Message 實作 Part 介面（且 MimeMessage 實作 MimePart ）。設定內容的基本機制是 setContent() 方法，同時使用參數，分別代表內容和 mime 類型：

```
message.setContent("Hello", "text/plain");
```

但如果，您知道您在使用 MimeMessage，而且訊息是純文字格式，您就可以用 setText() 方法，它只需要代表實際內容的參數，（ MIME 類型預設為 text/plain）：

```
message.setText("Hello");
```

後一種格式是設定純文字訊息內容的偏好機制。至於傳送其它類型的訊息,如 HTML 文件格式的訊息,我們偏好前者。用 setSubject() 方法設定 subject（主題）：message.setSubject("First");

3. Address

一旦您建立了 Session 和 Message,並將內容填入訊息後,就可以用 Address 確定郵件位址了。和 Message 一樣,Address 也是個抽象類別。您用的是 javax.mail.internet.InternetAddress 類別。

若建立的位址只包括郵件位址，只要傳遞郵件位址到 constructors 就行了。

```
Address address = new
InternetAddress("mulder-lance@yahoo.com.tw");
```

若希望名字緊接著郵件位址顯示，也可以把它傳遞給 constructors ：

```
Address address = new
InternetAddress("mulder-lance@yahoo.com.tw", "mulder");
```

需要為訊息的 from 網域和 to 網域建立位址物件。除非信件伺服器阻止,沒什麼能阻止你傳送一段看上去是來自任何人的訊息。

一旦建立了 address（位址）,將它們與訊息連線的方法有兩種。如果要識別發件人，您可以用 setFrom() 和 setReplyTo() 方法。

```
message.setFrom(address)
```

需要訊息顯示多個 from 位址，可以使用 addFrom() 方法：

```
Address address[] = ...;
message.addFrom(address);
```

若要識別訊息 recipient(收件人),您可以使用 addRecipient() 方法。除 address（位址）外，這一方法還請求一個 Message.RecipientType。

```
message.addRecipient(type, address)
```

三種預先定義的位址類型是：

1. Message.RecipientType.TO
2. Message.RecipientType.CC
3. Message.RecipientType.BCC

JavaMail API 沒有提供郵件位址有效性核查機制。雖然透過程式，自己能夠掃瞄有效字元（如 RFC 822 中定義的）或驗證信件交換(mail exchange,MX)記錄,但這些功能不屬於 JavaMail API。

4. Authenticator

與 java.net 類別一樣，JavaMail API 也可以利用 Authenticator 透過使用者名稱和密碼存取受保護的資源。對於 JavaMail API 來說，這些資源就是信件伺服器。JavaMail Authenticator 在 javax.mail 套件中，而且它和 java.net 中同名的類別 Authenticator 不同。兩者並不共享同一個 Authenticator，因為 JavaMail API 用於 Java 1.1，它沒有 java.net 類別。

要使用 Authenticator，先建立一個抽象類別的子類別，並從 getPasswordAuthentication() 方法中返回

PasswordAuthentication 實例。建立完成後，您必需向 session 註冊 Authenticator。然後，在需要認證的時候，就會通知 Authenticator。您可以跳出視窗，也可以從組態文件中（雖然沒有加密是不安全的）讀取使用者名稱和密碼，將它們作為 PasswordAuthentication 物件返回給呼叫程式。

```
Properties props = new Properties();
// fill props with any information
Authenticator auth = new MyAuthenticator();
Session session = Session.getDefaultInstance(props,
auth);
```

5. Transport

訊息傳送的最後一部分是使用 Transport 類別。這個類別用協定指定的語言傳送訊息（通常是 SMTP）。它是抽象類別，它的工作方式與 Session 有些類似。只有呼叫靜態 send() 方法，就能使用類別的 預設 版本：

```
Transport.send(message);
```

或者，您也可以從針對您的協定的 Session 中獲得一個特定的實例，傳遞使用者名稱和密碼（如果不必要就不傳），傳送訊息，然後關閉連線。

```
message.saveChanges(); // implicit with send()
Transport transport = session.getTransport("smtp");
transport.connect(host, username, password);
transport.sendMessage(message,message.getAllRecipien
ts());
transport.close();
```

後面這種方法在您要傳送多條訊息時最好，因為它能保持信件伺服器在訊息間的活動狀態。基本 send() 機制為每個方法的呼叫設定與伺服器獨立的連線。

注意：要觀察傳到信件伺服器上的信件指令，請用 session.setDebug(true) 設定除錯旗標。

6. Store 和 folder

用 Session 獲取訊息與傳送訊息開始很相似。但是，在 session 得到後，很可能是利用使用者名稱和密碼或使用 Authenticator 連線到一個 Store。類似於 Transport ，您告知 Store 使用什麼協定：

```
Store store = session.getStore("imap");
或
Store store = session.getStore("pop3");
store.connect(host, username, password);
```

連線到 Store 之後，接下來，您就可以獲取一個 Folder，您必需先開啟它，然後才能讀裡面的訊息。

```
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
Message message[] = folder.getMessages();
```

POP3 唯一可以用的文件夾是 INBOX。如果使用 IMAP，還可以用其它文件夾。雖然 Message message[] = folder.getMessages(); 看上去是個很慢的作業，它從伺服器上讀取每一條訊息，但只有在你實際需要訊息的一部分時，訊息的內容才會被檢索。一旦有了要讀的 Message，您可以用 getContent() 來獲取其內容，或者用 writeTo() 將

內容寫入 stream。getContent() 方法只能得到訊息內容，而 writeTo() 的輸出卻包括 header。

```
System.out.println(((MimeMessage)message).getContent
());
```

一旦讀完信件，要關閉與 folder 和 store 的連線。

```
folder.close(aBoolean);
store.close();
```

傳遞給 folder 的 close() 方法的 boolean 表示是否清除已刪除的訊息從而更新 folder。

注意：當Session 物件用 java.util.Properties 物件獲取訊息，如信件伺服器、使用者名稱、密碼及整個應用程式中共享的其它訊息，但不能藉由Javamail來新增，修改，刪除使用者的帳號和密碼，javamail不提供此功能，因為各種類的mail server的存取控制標準不一致。

2.2.2.2 Javamail傳輸模式

1. SMTP 傳輸格式：

簡單郵件傳輸協定（SMTP）是用於傳送電子郵件的機制。在 JavaMail API 環境 中，您的基於 JavaMail 的程式將與您或 Internet 服務提供商（ISP）的 SMTP 服務器通信。該 SMTP 伺服器將會把訊息轉發給用作接收消息的 SMTP 伺服器，最後用戶可通過 POP 或 IMAP 協定獲取該消息。由於支援身份驗證，所以不需要 SMTP 伺服器 是一種開放的轉發器，但需要確保 SMTP 伺服器配置正確。

JavaMail API 中沒有用於處理諸如配置伺服器以轉發消息或添加/刪除電子郵件帳戶這一類任務的功能。

2. POP 傳輸格式：

POP 的含義是郵局協定，當前的版本爲 3，也稱作 POP3，該協定是在 RFC 1939 中定義的。POP 是 Internet 上的大多數人用來接收郵件的機制。它爲每個使用者的每個信箱定義支援，這是它所做的全部工作，也是大多數問題的根源。在使用 POP 協議時，人們熟悉的很多功能，像查看收到了多少新郵件消息的功能，POP 根本不支持。這些功能都內置到諸如 Eudora 或 Microsoft Outlook 之類的郵件程式中，能爲您記住接收的上一封郵件，以及計算有多少新郵件這類資訊。因此，使用 JavaMail API 時，如果想獲取這類資訊，將需要由自己進行計算之前有接收多少封信件並作上標記，等到下一次收信的時候就可以比較之前接收信件的標記與封數，相剪即可求出新信件的封數。

3. IMAP 傳輸格式：

IMAP 是用於接收消息的更加高級的協定，它是在 RFC 2060 中定義的。IMAP 的含義是“Internet 消息訪問協定”，當前版本是第 4 版，也稱作 IMAP4。使用 IMAP 時，您的郵件伺服器必須支援該協定。您不能只是簡單地把程式轉變爲支援 IMAP，而不是支援 POP，就指望能支援 IMAP 中的一切。假定您的郵件服 務器支援 IMAP，那麼基於 JavaMail 的程式就可利用在伺服器上擁有多個文件夾的用戶，並且這些文件夾可以被多個用戶共用的功能。由於 IMAP 協定具有更高級的功能，您也許會想 IMAP 應該被每一個人使用，但事實不是這樣。因爲 IMAP 會加重郵件伺

服器的負荷，它需要伺服器接收新郵件，發送郵件給請求的用戶，並在多個文件夾中爲每個用戶維護這些郵件。而且郵件需要集中備份,因而長期下去用戶的文件夾會變得越來越大，當磁碟空間用光了時，每個人都會遭受損失。但使用 POP 協定時，可以將保存郵件可以刪除，減少伺服器的負擔，不用備份。

4. MIME 格式：

MIME 的含義是“多用途的網際郵件擴充協定”。它不是一種郵件傳輸協定，相反地，它定義傳輸的內容：郵件的格式、附件等。許多傳輸的文字檔都定義了 MIME 協定，包含：RFC 822、RFC 2045、 RFC 2046 和 RFC 2047。 用 JavaMail API 發展的 webmail，一般不需要擔心這些格式。但是，這些格式確實存在， 只是 JavaMail 幫你把這些協定，如何傳輸與連結都幫你寫好。

2.2.3 EWA 規格

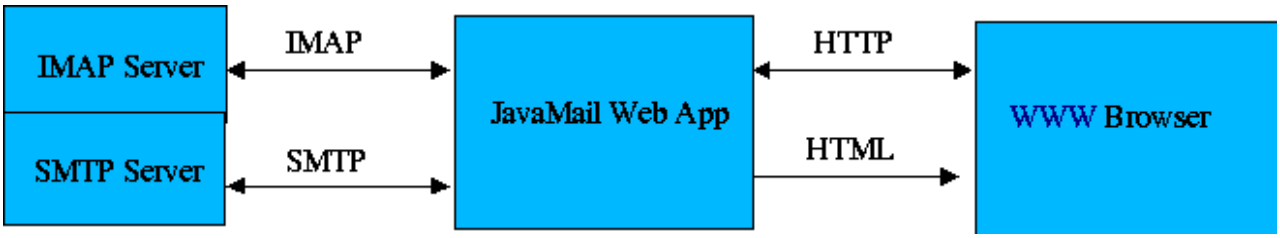
何謂 EWA（Email Web Application），它提供了基本功能的 Email 應用程式的，像是登入、檢查密碼、發送電子郵件，其中它的功能如下：

登入一個 IMAP 或是使用 POP & SMTP 的 SERVER。
列出所有你在信箱中所有的郵件（在 Mail Server 的 INBOX 中）
可以選擇所要看的郵件訊息。
可以檢視附加檔案的資料與下載。
可以回信與新增電子郵件。

表三

2.2.3.1 EWA基本架構

EWA是一個 Three-tiered Web Application，其架構圖如下



圖一

2.2.2.3.2 EWA Model

MailUserBean	利用 javabeans 來儲存郵件使用者的資訊,其中 Bean 使用在 IMAP 的連線與設定，使用者的登入與登出的方法，以及主機的密碼和帳號連結判斷方法，通常可以用 get 和 set 的方法來得到下列的參數，像是主機名稱，使用者名字、密碼、Session 和傳輸的 Protocol。	
Tag Library	Message tag	用來讀取郵件訊息，像是 MessageTag.java, MessageTEI.java
	List messages tag	用來列出完整郵件的訊息，像是 ListMessagesTag.java, ListMessagesTEI.java
	Sendmail tag	用來寄信，像是SendTag.java
Attachment Servlet	用來擷取或展示 multipart message ，像是Body裡面的訊息或者夾帶檔案。	

表四

2.3 Lex 的構造說明

2.3.1 Lex簡介

他是一個用來幫助我們撰寫資料的轉換工具，尤其是在轉換具有結構性的資料，再處理一個結構性資料的程式中，有兩件事是不可避免的，一個是將輸入的資料切割成有意義的小單位，另外是進一步處理這些小單位之間的關係，舉例來說，對一個搜尋文字程式來說，處理的資料是文字，所以分割出來的小單位是一行一行的文字，而這些小單位的關係就是:有些航包括有想找的字,有些行則沒有,同樣地，對一個C語言的編譯器而言，他所處理的是C語言程式碼，所以分割出來的小單位包括，變數名稱、常數、字串、運算元、標點符號等等。資料分割成小單位的過程，就叫做「字彙分析」簡稱lexing，那麼lex要怎麼來幫助我們做字彙分析呢？首先我們必須要對lex說明這些token可能長的怎麼樣，然後lex就會幫你做出一個適合C語言函式，然後我們就可以呼叫這個函式來分割出token。這樣子的函式叫做，「字彙分析器」，簡稱lexer或是scanner，而那些對lex的說明，就稱作lex 規格。Lex對於token長相的描述，是採用「常規表示法」，簡稱Regex，跟一般在grep或是egrep命令所使用的表示法很像，而且這個樣式可以讓字彙分析器可以很快遞掃描輸入資料，搜尋所描述的token。就算有再多的常規表示法，lex依然可以轉換出一個快速搜尋的token函式。這使的使用lex來產生字彙分析器，比一般人用C語言寫的字彙分析器，在速度上快上很多。

2.3.2 Lex的規格架構

一個lex程式包括3個部分：定義段落、規則段落、以及使用者定義函式段落。

…definition section … （定義段落）

%%

…rules section …　　（規則段落）

%%
…user subroutines…　　（使用者定義函式段落）
每個段落之間用到兩個百分比符號來分開，而且這兩個百分比要單獨寫在一行。前兩個段落是一定要的，但是可以為空，第三個段落與其之前的%%都可以省略。

2.3.3 Lex & yacc的關係

lex 與 yacc 的 關係是密切的，因為字彙分析器必須由 lex 來產生。我們利用 lex 來比對常規表示法所描述的字串，yacc 則是用來判斷句子是否符合語法，換句話說 lex 是將輸入的資料切割成小單位，yacc 再將這些小單位以邏輯的方式加以組織。

2.4 Shell 構造說明

2.4.1 shell簡介

shell 是一個程式，負責解譯及執行使用者所下達的指令，由於身負如此之重任，所以通常它必須在使用者簽入（login）系統後，便必須載入記憶體中，這個 shell 有一個專有名詞，我們叫她做 login shell。她會為使用者處理輸入、輸出及系統的錯誤訊息顯示（standard input, standard output and standard error）；並讀取特殊的起始檔案（startup files）用來設定使用者個人所制訂的環境變數與預設變數（environment variables and predefined variables）。當然只要使用者還停留在系統中，shell 便會為使用者解譯輸入的指令。直到使用者退出系統前，login shell 都會存在記憶體中為使用者默默地服務著。

2.4.2 Shell的種類與規格

UNIX 作業系統在這 20 幾年的發展過程當中，實際上產生過的 shell 實在是不計其數的多。但在各版本之間通用且具有重要的地位的，只有三個。如果依產生的前後次序來排列的話，它們分別是 Bourne shell、C shell 及 Korn shell。以下是一個簡單的對照表。

Shell	創作者	符號	指令名稱
Bourne	S. R. Bourne	$	sh
C	Bill Joy	%	csh
Korn	David G. Kron	$	ksh

表五

1.Bourne Shell

UNIX 系統最早出現的 Shell。開發於貝爾實驗室（Bell Laboratories, Murray Hill, New Jersey.），它的創造者是 S. R. Bourne，所以命名為“Bourne Shell”。一直到今天，在各版 UNIX 作業系統內所採用的 Standard Shell 一直均是以它為主。所謂的 standard shell 的意思便是指當一個使用者在沒有指定要使用何種 shell 的情況下，作業系統會自動幫你設定的 shell，我們稱它為 standard shell。當然它也已經被視為是 UNIX 作業系統必備的一部份了。因為 Bourne Shell 在執行效率上優於其他的 Shell，所以你也可在 UNIX 系統中找尋到以它的語法所寫成的執行檔。但是它並不支援 aliases 與 history 功能，同時在 Job control 的功能上也比較簡單。它的前景與背景工作是無法任意調換的。在整體的功能上，對今日的使用者而言已明顯地有些不足之處。所以也有一些 shell 以它為基礎，發展出包含新功能的新版本。譬如，你可以在 UNIX System V Release 4 版本中發現到有一個 shell 叫作“jsh”。它便是改進了關於 job control 功能的 Bourne Shell 版本。另外還有 Free Software Foundation 也有發展一個 shell 叫“bash”，縮寫的意思是“bourne-again”。它的整體功能趨近於後來所發展出來的“Korn shell”。不過它並不包括在各 UNIX 作業系統中。Bourne Shell 所使用的起始檔案名稱為“.profile”和“.login”。在變數上支援局部變數（local variables）與整體變數（global variables）兩種。整體變數必須以 export glo_var_name 的方式宣告之。所支援的控制程序有“if-then-else”，“case”，“for”，“while”及“until”等。但請注意在 Bourne Shell 中是沒有“goto”功能的（這是比較嚴謹的作法）。

2.Korn Shell

Korn Shell 出現在 Bourne Shell 與 C Shell 之後，在功能上則吸收了 C Shell 的 aliases, history 等。而語法則是與 Bourne Shell 相容，所以在 Bourne Shell 之下所開發的 shell script，也可以在其中執行。它也是以作者的名字命名的 shell。原作者是 David G. Kron。Korn Shell 也是開發於美國貝爾實驗室（AT&T Bell Laboratories, Murray Hill, New Jersey.）。因為在開發的時間上比較晚，現今有支援它的版本，除了 UNIX System V Release 4 版本已將它列為標準配備外，其他版本的 UNIX 作業系統中並不多見。在大部份的 BSD 版本的 UNIX 作業系統並不支援。Korn Shell 所使用的起始檔案名稱與 Bourne Shell 所使用的名稱相同也叫做“.profile”和“.login”。

3.C Shell

C Shell 發展於美國加州州立大學柏克萊分校（University of California, Berkeley, California.），原始版本的 C Shell 創作者是 Bill Joy，當時這位還是柏克萊的研究生。由於這個 shell 的語法與 C 語言(C Language)極相似而得此名。此 Shell 對慣用 C 語言的使用者而言無疑是一大福音。它出現的時間相當早，早到在 UNIX Time-Sharing System, Sixth Edition 中便已支援。所以如今各版本的 UNIX 作業系統中，BSD 版本的 UNIX 作業系統均有支援，而 SYSTEM V 版本的 UNIX 作業系統，要找到不支援 C Shell 也實在是“很難”（因為我目前找不到不支援 C Shell 的 UNIX OS)。所以可以說 C shell 已經是今日 UNIX 作業系統的一部份了。

在特殊功能上，它是最早提供“別名（aliases）”、“指令使用記錄（history）”與“工作控制（Job Control）”功能的 shell。在今天，這幾樣功能幾乎可以說是 UNIX 作業系統的重要特色。C Shell 所使用的啟始檔案名稱與前面的兩個 shell 不

同。它的特殊檔案為“.cshrc”、“.login”與“.logout”。在其中所使用的設定語法也不相同，所以在使用上並不能相容。C Shell 所提供的變數有預設變數（predefined variables）及環境變數（environment variables）兩種。分別以內建指令 set 及 setenv 去設定它們。

在目前 C Shell 也有他的更新版，不過並非原來的開法者所開發，而是在 1980 年左右，由一群工商與學術界的人士共同合作的成果 -- 也就是今天的 'T' C Shell，簡稱與指令相同叫 tcsh。 tcsh 目前發展的相當不錯，除了修更 csh 的一些 bugs 外，還增加了不少新的功能。

2.4.3 Shell的變數環境介紹

C shell 對其本身在整體的環境控制與部份功能的設定和使用上，都有專屬的變數，提供使用者自己設定與應用。同時在變數的型態上，也區分為環境變數（environment variables）與預設變數（predefined variables）兩種。前者相當於整體變數，而後者相當於區域變數。以下我們將為以這兩大分類來為你分別介紹 C shell 本身所制定的各種變數。

2.4.3.1 環境變數的設定影響

環境變數的設定目的在於管理 shell，這是它之所以重要的原因。它的特性相當於整體變數（global variable）。也就是說，你僅需要把環境變數設定在你的“.cshrc”檔案中，由 login shell 所產生的 subshell 或者是執行的 shell 文稿、程式或指令等，均不需再重新設定，便可以直接呼叫或使用該變數。所以環境變數是具有遺傳（inherited）性的。因為在 UNIX 作業系統中，由一個處理程序（process）會將它全部的環境變數遺傳給它所衍生出的子處理程序（child preoceese）。譬如你在 login

shell 之下執行一個 vi 指令，設定的 TERM 變數會決定使用何種終端機模式，同時 vi 程式本身也會繼承了原來的 login shell 所定義的所有環境變數，所以當你想要在 vi 程式中用指令“:sh”的方式產生一個新的 shell 時，vi 程式還會依據你所定義的 SHELL 變數，產生那個你所指定的 shell 的原因。當然因 vi 程式所產生的 new subshell，依然會繼承來自於 vi 程式的所有環境變數。C shell 的環境變數全部都是以大寫字母命名。事實上這也是一個不成文的規定。所以當你要自行定義一些環境變數時，請你也能夠這樣做。設定環境變數的使用語法如以下所示：

設定語法 setenv ENVNAME string
解除設定語法 unsetenv variable
顯示所有設定 env

C shell 的環境變數並不多，僅有基本且重要的特殊資訊才被列入。如使用者的簽入目錄（login directory），存放郵件的目錄，終端機的模式，執行指令依據的搜尋路徑等。在這些環境變數中，部份會由系統依據某些特殊檔案內的資料，為使用者自動設定初始值。如 HOME 變數以及 USER 變數（有些 UNIX 版本不叫做 USER 變數，改稱為 LOGNAME 變數）的初設值便是來自於“/etc/passwd”檔案。又如 TERM 變數初始值是來自於檔案“/etc/ttytab”。除此之外，環境變數中的 HOME, PATH, MAIL, TERM 等，還會將它們的內容拷貝到相同名稱的預設變數中，以做為預設變數的初始值。不僅如此，這兩者之間還保有一種互動的關係，也就是其中的任何一方有改變，另一方變數也會自動地將變數內容更新。這些都是環境變數的特點。

2.4.3.2 預設變數的設定影響

在 C shell 本身所制定的兩種變數當中，真正用來控制 C shell 功能的是以下我們所要為你介紹的這些預設變數。制定這些變數關係到指令的執行、hisroty 功能的使用、 指令 time 的顯示、job control 的顯示、指令行模式下的資訊顯示、wildcard 功能的使用、退出 C shell 的設定、filename completion 功能的使用、輸出/輸入重導向功能的重寫保護等 C shell 的特殊功能。所以對一個使用者或對於 C shell 的使用而言，這些變數是相當重要的。因為在上述的特殊功能中，有一小部份功能如果不自行設定，系統的初始設定值是不使用的，C shell 的預設變數在設定上為了要和環境變數有明顯的區別，一般的習慣均使用小寫字母來做為變數名稱。各項設定的語法如下所示：

設定語法 set variable

set variable = string

解除設定語法 unset variable

顯示所有設定 set

C shell 的預設變數一般均集中設定在“.cshrc”檔案中。為什麼要集中到“.cshrc”檔案中呢？因為預設變數的特性是局部性的，在實際上當你在設定這些變數之後，這些預設變數的設定狀態，並不像環境變數那樣，會將設定值“遺傳”給在它之下所產生的 subshell。如果你想要每個 C shell 及它的 subshell 均要保有你想要的預設變數的設定狀態，你便應該將它設定在 C shell 的起始檔案“.cshrc”中。因為 C shell 本身的在產生 subshell 時，會自動地去讀取“.cshrc”檔案，以該檔案內的設定做為 subshell 的初始環境設定。所以，你想要的各項預設變數的設定，為了要能做到，自 login shell 到所有的

subshell 均能完全一致，最佳的方式便是將它們設定在“.cshrc”檔案中。如果是臨時性的、短暫性的預設變數設定，當然還是以在指令行模式下手動設定方式比較適合。所以這類的變數在設定上，請務必留意使用上的需要，來選擇適合的、正確的設定方式。

2.5 Sybase 的世界

2.5.1 Sybase 的歷史

Sybase 公司在 1984 年由 Mark Hoffman 和 Bob Epstein 共同創立，總公司位於美國加州的 Emeryville。由於 Sybase 是較年輕的公司，且成立時為開放式系統起飛的年代，沒有專屬系統的歷史包袱，因此可以創新觀念，開發功能優異之大型 DBMS

Sybase 這名稱是由“System Database”這二個英文字分別取其前 2 個及後 4 個字母所組合而成，它的原意為系統資料庫，所以代表了 Sybase 的資料庫管理系統可以和各平台的作業系統緊密地結合，並達到高效率及高可靠性的目的。

圖二

Sybase 的商標也滿有意思的，商標圖中的曲線將一系列在 Fibonacci Series 的定義下由“由小到大”遞增的矩形角連結起來，所謂 Fibonacci Series 就是一連串的數值，並且下一個數值等於前兩個數值之和。Sybase 之所以採用這樣黃金比例矩形所繪出的 Fibonacci Series 創意構圖作為他們的商標，主要是因為他們對產品的要求：精準，簡單，擴充性。

Sybase 的 RDBMS（關連式資料庫管理系統 Relational Database Management Systems）於 1988 年問世，這是專問為了網路化的線上交易處理應用而設計的後端資料庫伺服器。1992 年推出 system10，率先提供全企業 Client/Server 運作的完整架構。於 1995 年發表新一代的 SQL Server：Sybase System 11，其效能、擴充性、及品質都大幅超越業界標準，更於 1997 年發表最新的全產品架構：Adaptive Component Architecture（ACA），以及最新版的 SQL Server：Adaptive Server Enterprise(ASE)，全新的平行處理能力以及彈性的控管能力，不僅能為交易處理及決策支援提供高效能，更具備獨特的特性能應付 Internet 無可預測的需求。Sybase 公司並於 1990 年至 1992 年連續三年在美國財星（Fortune）雜誌上被評選為美國成長最快速的前十名企業之一。

對於大量資料庫的處理，Sybase 也以領先的技術於 1994 年提出 MPP（Massively Parallel Processing）產品，它提供強大、高效能的平行資料庫處理功能，並滿足企業儲存資料所需的彈性、擴充性及管理能力。美國大通銀行（CHASE Manhattan）即為使用 Sybase MPP 的成功案例，它的實際資料量超過 500GB，並且連續 5 年每年省下 2 仟 9 佰萬美元的軟體投資。

除了中大型資料庫之外，對於機動式、桌上型及工作群組環境的小型資料庫應用，Sybase 於 1995 年發表了 Sybase SQL Anywhere 5.0，並且於 1998 年年中發表最新一代第一個內含 Java SQL 的 Sybase 資料庫，Sybase Adaptive Server Anywhere 6.0。

為了幫助企業達成適型化（Rightsizing Solution）及現有系統整合的目的，Sybase 於 1990 年推出了第一個將 IBM MVS 大型主機整合到區域網路 Client/Server 環境的產品。於 1993 年推出了 OmniSQL Gateway，使系統在不同元件的資料庫間，可以達成透通及跨平台地運作。更於 1994 年發表了 Enterprise CONNECT 的架構，提供了關聯或或非關聯式、大型主機及 Client/Server 架構環境前所未有的功能。此產品獲得了 Fortune 100 中之 89 個企業的青睞，並運用於內部的系統架構及程式開發環境。

在分散式架構的推展上，Sybase 於 1993 年率先發表 Replication Server，它對企業內分散式處理的需求而設計，因此可作為企業級、高可用性資料庫的解決方案。

開發工具為應用系統不可或缺的一環，PowerBuilder 系列產品為 PowerSoft 公司全球佔有率超過 54%的 Client/Server 應用程式開發工具。Sybase 於 1995 年合併了 PowerSoft 公司，並成為全世界第 7 大的軟體公司。同年 PowerSoft 事業處合併了法國 SDP Technologies 公司，該公司開發的 S-Designor，為一功能完整並手於分析、設計、及建構 Client/Server 資料庫與應用程式的工具組。

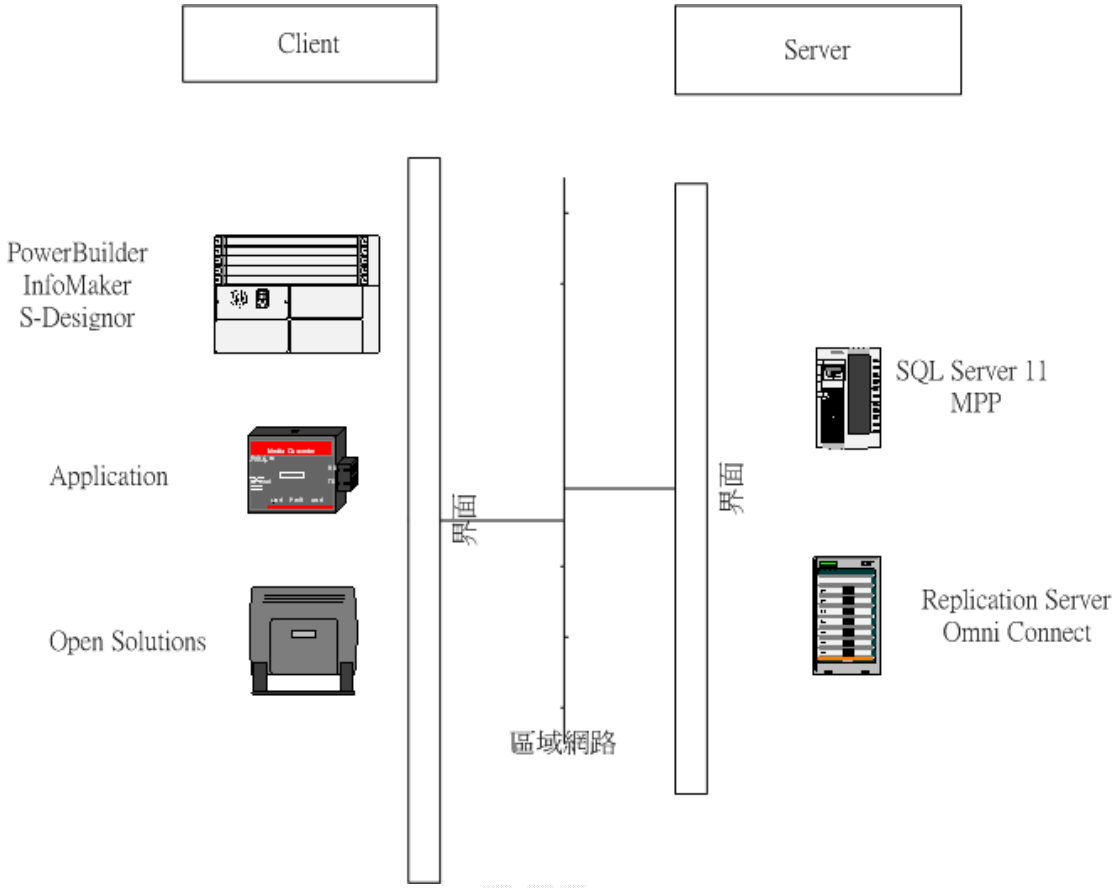
軟體品質為提供客戶一個有效率且穩定工作環境的重要因素，Sybase 為美國唯一率先通過 ISO 9001 認證的軟體工司。這代表了 Sybase 在品管上的嚴格要求，並給予客戶使用上的最佳保障。

2.5.2 Open Client / Open Server 簡介

長久以來，順暢的連結及互通能力一直是 Client/Server 運算架構的承諾，因此無論你使用任何開發工具或應用程式都能以隨插即用的方式，輕鬆與任何資料來源運作。Sybase 一直以他強大且附有延展性的連結能力著稱，其中 Open Client 和 Open Server 實踐了順暢的連結及互通能力的諾言。

Open Client 和 Open Server 為提供連結功能的產品，他們建構了開放式 Client/Server 環境互通功能的基礎，這種互通的功能正是現今異質運算環境所不可或缺的重要特性。

Open Client 和 Open Server 提供了一組應用程式開發界面(Application Programming Interfaces’ APIs)及程式庫。開發工具或應用程式可以透過 Open Adaptive Server、其他 Sybase Server(如 Replication Server、MPP、OmniCONNECT…等等)、以及非 Sybase 的異質資料來源。Open Server 則為 Server 的應用程式開發界面，因此可協助開發人員將任何資料來源或服務，整合到企業的線上作業裡。Open Client 及 Open Server 的系統架構如圖所示：



圖三

2.5.3 Open Client

Open Client 代表了一組支授 Sybase 可程式化界面(Programmable interfaces)的產品。它包含了 Client-Library、Emabedded SQL、XA-Library、DB-Library 及和 Microsoft 的 Open Database Connectivity(ODBC)連結的界面。Open Client 的最大作用在於有效地管理前端開發工具或應用程式與後端 Adaptive Server 或 Open Server 應用程式間的通訊。

2.5.4 Open Server

Sybase 是目前的資料庫廠商中唯一提供了 Server 端的開發界面 – Open Server / Server-Library 的軟體廠商，因此開啟了讓使用者撰寫 Server 應用程式的大門。

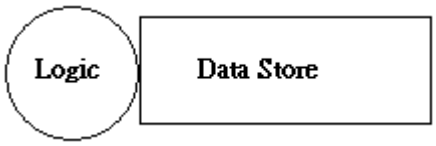
Open Server 為程式設計工具組，容許你或第三協力廠商所開發的非 Sybase 應用程式，對使用 Open Client 的應用程式能以 Sybase 的資料形式顯示。簡而言之，你可以使用 Open Server 程式設計工具組開發出一個像 Sybase Adaptive Server (ASE)這樣具有 power 的伺服器應用程式，讓使用 Open Client 界面的應用程式連接。因此 Open Server 使你能夠將重要的資料和服務傳送給提出請求的用戶端。

2.5.5 Sybase 的優勢 ：

Adaptive Component Architecture

為了滿足客戶對資訊的需求及提升其企業的競爭力，Sybase 於 1997 年提出一個新的架構，稱為適應型元件架構(Adaptive Component architecture)簡稱 ACA，這個架構是根據今日企業的資訊環境需求而訂定的，包含以下幾個特性：

- 使用標準元件
- 能採用快速應用發展工具
- 交運資料
- 降低複雜度



圖四

在 Sybase 的新 ACA 架構中，包含豐富的開發工具來建立開放的元件邏輯(Logic)，以及最佳化的元件資料儲存(Data Store)，並且在每個層次中都能支授言兩種元件型態。其中，元件邏輯代表一些事先建立、事先測試好的元件，可重複使用於多層架構中的各個層次；而資料儲存表示為各種應用程式的使用提供最佳的資料儲存方式，而且不同的資料儲存都使用相同的應用程式界面(ＡＰＩ)。

這個特性可以給與企業能夠根據他們的需求設計、發展、以及佈署重要的程式於各個層次中，並且，ACA 架構也提供一種平台，能夠讓他們很容易地發展以及部署應用系統於傳統兩階層的 client/server，多層架構，以及網際網路環境中。

第三章　系統建構

3.1 軟體環境

主機 \ 軟體與環境	軟體版本	作業系統
WebServer主機	Tomcat4.1.29 + MailServer	FreeBSD 4.9Stable
Database主機	Sybase ASE SQL-Server	Debian Linux 30r1

表六

3.2 硬體環境

採用分散式架構，因為 FreeBSD 的 Thread 是 Process 的 Thread，我們並不考慮把 Database 裝在 FreeBSD，而是裝在 Linux。Linux 的 User Thread 能幫我們減輕很多 Database Server 的 throughput

主機	主機配件
WebServer 主機	中央處理器：Pentium II 200MHz 記憶體：256 Mb SDRAM 乙太網路卡：D-Link TX530 程式：JDK1.4.1 + lex
Database 主機	中央處理器：Pentium I 133MHz 記憶體：128 Mb SDRAM 乙太網路卡：D-Link TX530

表七

3.3 FreeBSD系統最佳化

3.3.1 安裝cvsup

```
# cd /usr/ports/net/cvsup
# make install clean
```

3.3.2 使用cvsup 得到最新的source tree和ports tree

a) source tree

把default host改成cvsup2.tw.FreeBSD.org（此為離我們較近的Server）

```
# vi /usr/share/examples/cvsup/stable-supfile
```

取得最新的原始碼(STABLE version)

```
# cvsup /usr/share/examples/cvsup/stable-supfile
```

(or Edit /etc/make.conf, then, cd /usr/src; make update)

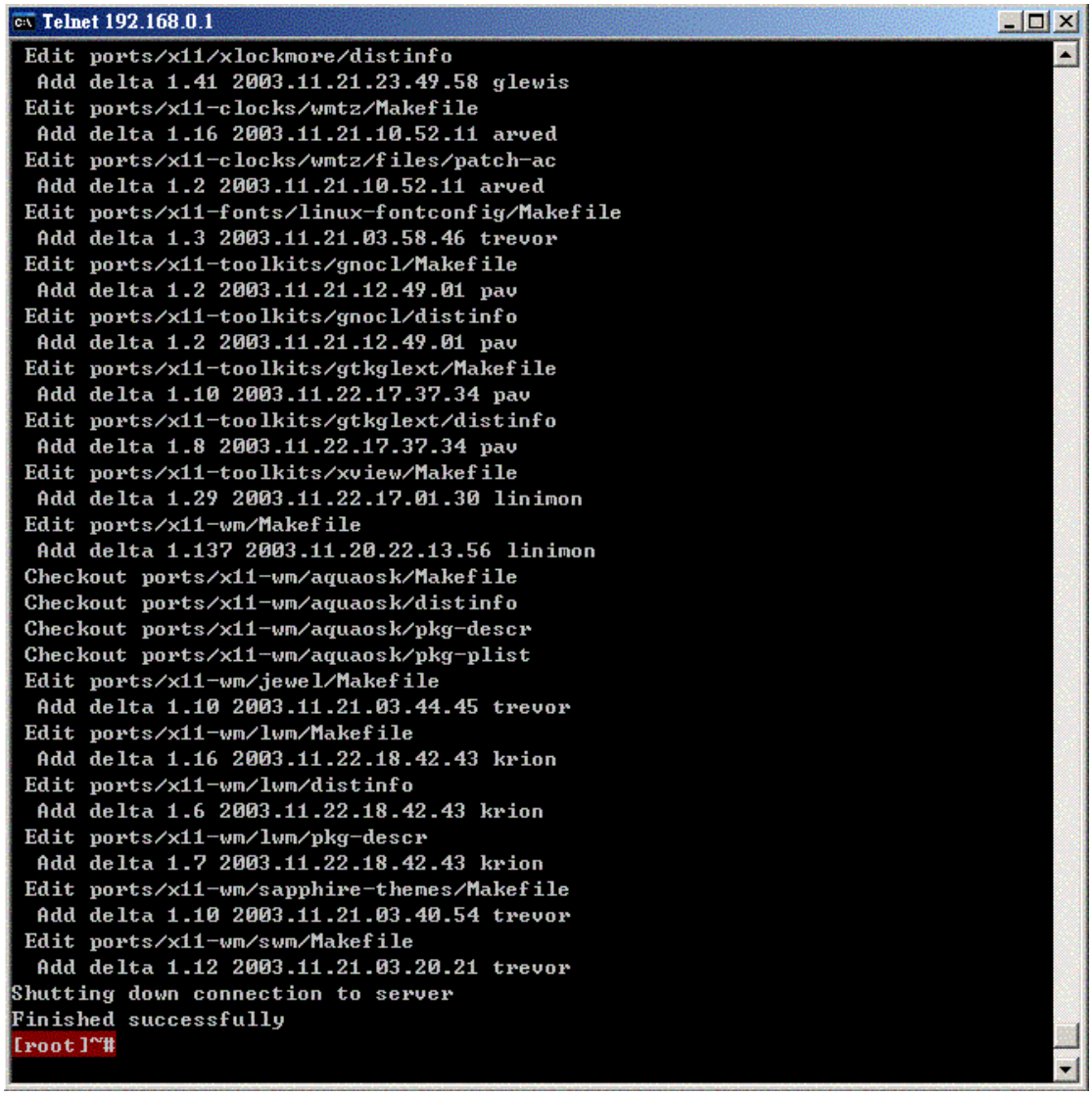
b) ports tree

```
vi /usr/share/examples/cvsup/ports-supfile
```

把default host改成cvsup2.tw.FreeBSD.org（此為離我們較近的Server）

```
cvsup -g -L 2 /usr/share/examples/cvsup/ports-supfile
```

更新完ports-tree的畫面如下：



圖五

3.3.3 開始更新系統

編譯所有的系統程式

```
# make buildworld
```

編譯並安裝新的核心

```
# make buildkernel KERNCONF=MySetup;make installkernel
KERNCONF=MySetup
```

安裝新的系統程式

```
# make installworld
```

更新系統設定檔

```
# mergemaster
```

flush disk cache AND reboot

```
# sync ; sync ; sync ; reboot
```

重開後看系統版本

```
# uname -a
```

圖六

嗯，已經是現行FreeBSD的最新stable分支版本了！

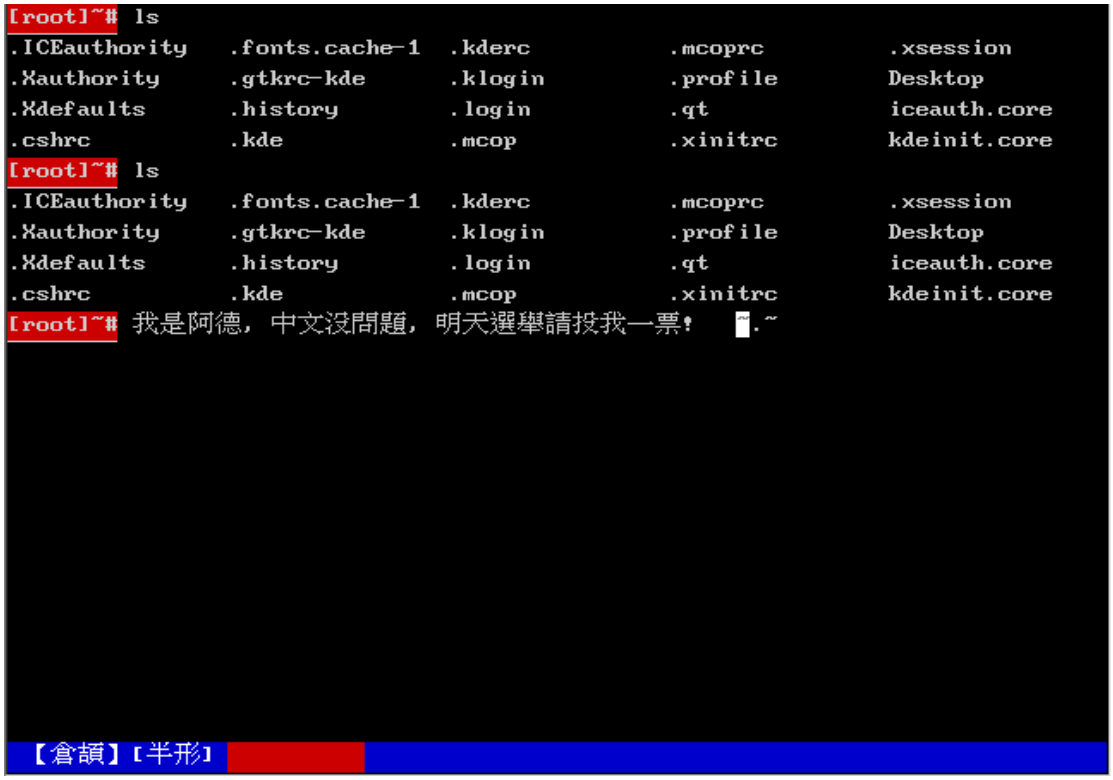
3.3.4 系統中文化

如果要在Console下（機器面前）使用中文的環境，必須安裝big5con軟體才可。我們使用ports安裝

```
# cd /usr/ports/Chinese/big5con
```

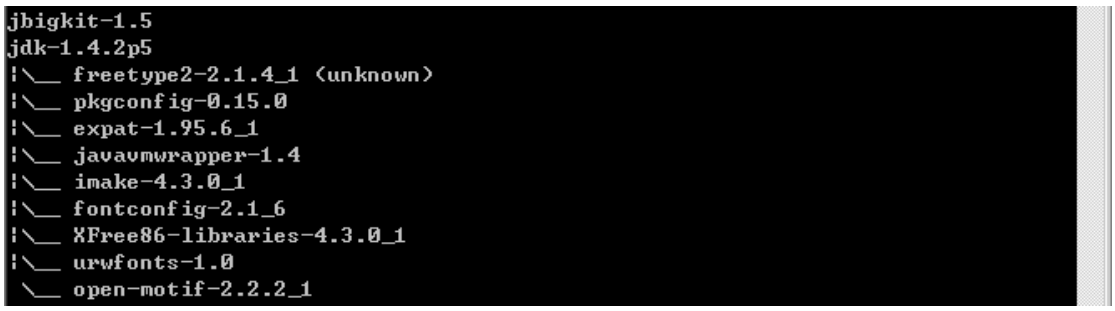
```
# make install clean
```

安裝完再rehash完後，就可以使用b5c來使用像DOS那樣的中文境境



圖七

因為作業方便，有時我們會用到xwindow，jdk的環境也須要用到xwindow。我們可以用pkg_tree來看jdk所須要的package。



圖八

所以我們開始進行x-window的中文化

Step1. 先用/stand/sysinstall裝好x-windows

Step2. 安裝中文化字型

如果要在x window中看到中文的選單，必須先裝中文字型kcfont（國喬字型）及arphicttf（文鼎字型），及中文訊息檔(i18n)

```
# cd /usr/ports/chinese/kcfonts
# make install clean
# cd /usr/ports/Chinese/arphicttf
# make install clean
# cd /usr/ports/Chinese/kde3-i18n-zh_TW
# make install clean
```

Step3. 產生設定檔

```
# XFree86 -configure
# mv ~/XF86Config.new /etc/X11/XF86config
```

接著編輯/etc/X11/XF86config，很多人在這個檔上都會遇到挫折，我們的設定檔請見<附錄E>

我們的中文化KDE

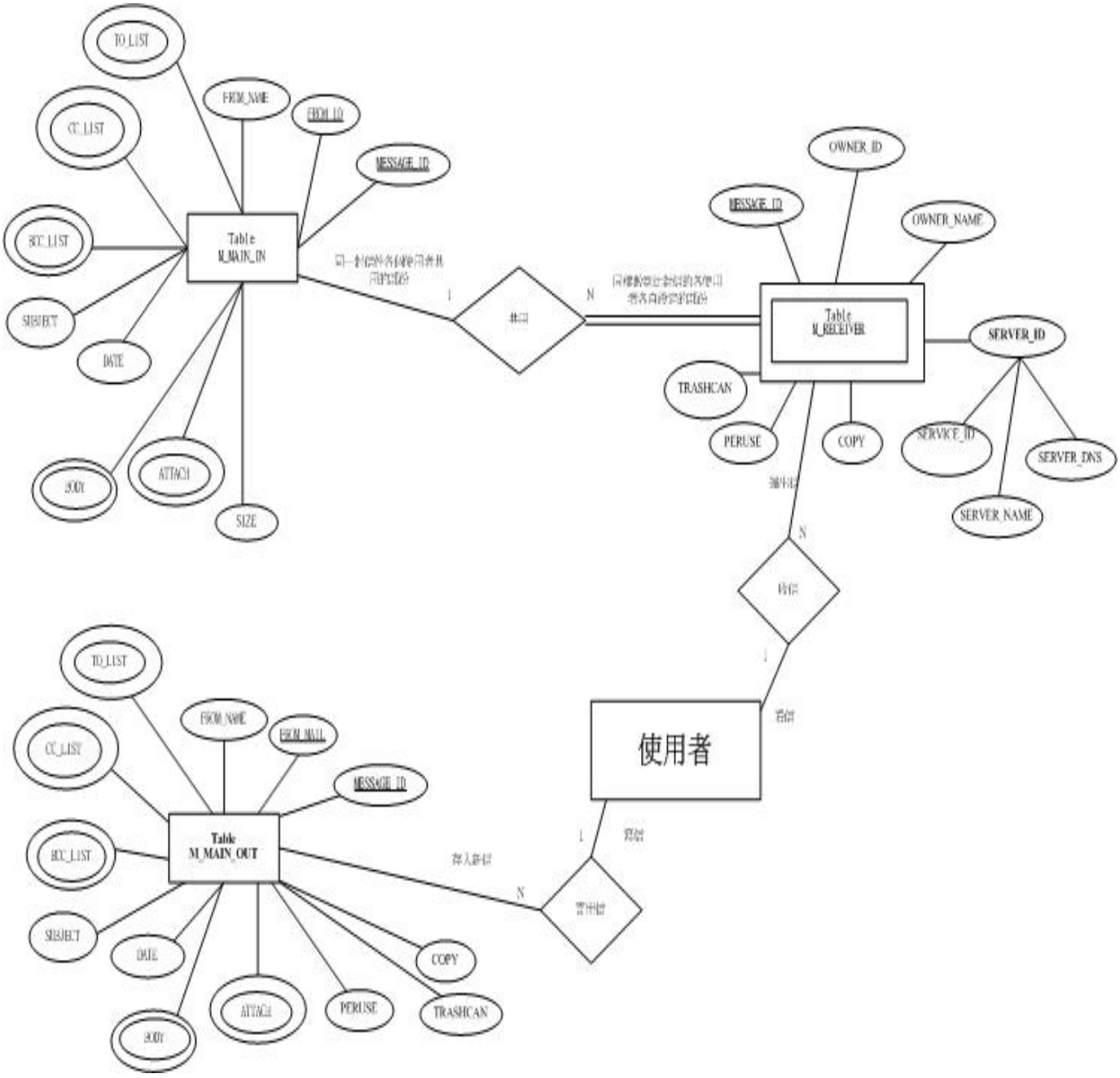
圖九

3.4 Sybase資料庫最佳化

3.4.1 Mail Table

因為Table的Size太大，請見<附錄F>

3.4.2 ER Model



圖十

3.4.3 建Database的SQL Script

```
use master
go
/* ISEC Database master device */
create database ISEC on default = 100
go
sp_addlogin 'isecuser','??????','ISEC'
go
use ISEC
go
sp_changedbowner isecuser
go

print    '1.M_MAIN_IN'
/* SETUSER 'dbo' */
go
create table dbo.M_MAIN_IN (
FROM_ID     varchar(16) not null,
FROM_NAME   varchar(20) not null,
TO_LIST     TEXT        not null,
CC_LIST     TEXT        not null,
BCC_LIST    TEXT        not null,
SUBJECT     varchar(255) not null,
DATE        varchar(50) not null,
MESSAGE_ID  varchar(20) not null,
BODY        TEXT        not null,
ATTACH      TEXT        null,
SIZE        int         not null,
MISSION_ID  varchar(3)  not null
)
go

print    '2.M_RECEIVER'
/* SETUSER 'dbo' */
go
create table dbo.M_RECEIVER (
MESSAGE_ID  varchar(20) not null,
```

```
OWNER_ID    varchar(10) not null,
OWNER_NAME  varchar(10) not null,
SEVER_ID    varchar(20) not null,
COPY        bit         not null,
TRASHCAN    bit         not null,
PERUSE      bit         not null
)
go

print    '3.M_MAIN_OUT'
/* SETUSER 'dbo' */
go
create table dbo.M_MAIN_OUT (
FROM_MAIL   varchar(40) not null,
TO_LIST     TEXT        not null,
CC_LIST     TEXT        not null,
BCC_LIST    TEXT        not null,
SUBJECT     varchar(255) not null,
DATE        varchar(50) not null,
MESSAGE_ID  varchar(50) not null,
BODY        TEXT        not null,
ATTACH      TEXT        null,
COPY        bit         not null,
TRASHCAN    bit         not null,
PERUSE      bit         not null
)
go
print    '4.M_SERVER'
/* SETUSER 'dbo' */
go
create table dbo.M_SERVER(
SEVER_ID    varchar(3)  not null,
SEVER_DNS   varchar(20) not null,
SEVER_NAME  varchar(20) not null,
SERVICE_ID  varchar(2)  not null
)
go
```

第四章 系統實做

4.1 工作分配表

工作 \ 分派者	陳峻儀	賴明德
資料收集	◎	◎
系統評估	◎	◎
系統架構	◎	◎
軟體的架設	◎	◎
FreeBSD系統最佳化		◎
Linux系統最佳化		◎
Sybase資料庫最佳化		◎
Web美工	◎	
Webmail的撰寫 (use JSP)	◎	◎
Dispatcher 派送管理程式(Java)		◎
cParser MIME格式轉換程式(Lex)	◎	
jConnector 資料庫連結程式 (Java + jconnector)		◎
各系統測試	◎	◎
系統整合		◎
報告撰寫	◎	◎
報告整合	◎	◎

表八

工作時間表

月份	工作
1 月 ~ 2 月	資料收集
2 月 ~ 3 月	資料收集、系統分析
3 月 ~ 4 月	分析發展工具
4 月 ~ 5 月	軟體的架設
5 月 ~ 6 月	Email parser、Sybase資料庫處理
6 月 ~ 7 月	Email parser、Sybase資料庫處理
7 月 ~ 8 月	Email parser、Sybase資料庫處理
8 月 ~ 9 月	Email parser、Sybase資料庫處理
9 月 ~ 10 月	C shell、Webmail的撰寫
10 月 ~ 11 月	系統整合、測試
11 月 ~1 2 月	書面報告

表九

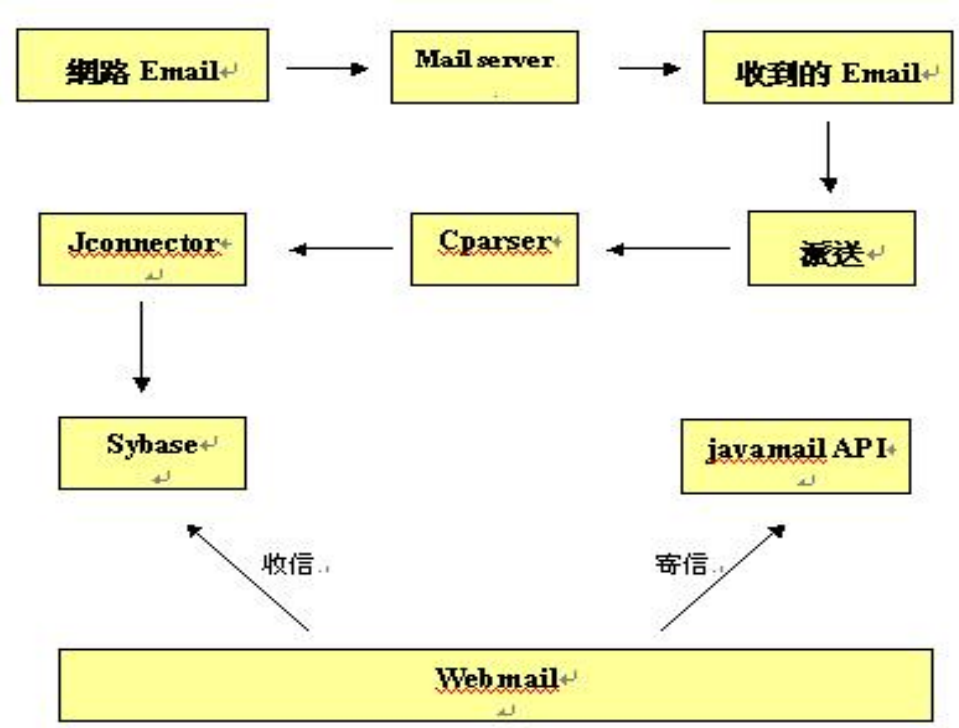
4.2 系統測試與流程圖

4.2.1 系統設定檔

```
############################################################
#     Security Gateway 機器的參數                          #
############################################################
SERVER_ID= 001
SERVER_NAME= sars
SERVER_DNS= sars.mclab.iecs.fcu.edu.tw
//   不合法或有問題的 mail 放置的目錄
Filter_Match_Dir=/usr/local/sgdg/dp/Err_Mail/Filter_Match_Dir
No_Receiver_Dir=/usr/local/sgdg/dp/Err_Mail/No_Receiver_Dir
//   VS 在送不到 InterScan 時, 要放轉放 mail 的資料夾
InterScan_Err_Dir=/usr/local/sgdg/dp/Err_Mail/InterScan_Err_Dir
############################################################
#     Part Of DP(Dispatcher)                               #
############################################################
//   DP 掃描目錄內是否有新增檔案, 再做動作 的頻率（以 秒 爲單位）
dp_scan_ratio= 6
//   DP 送往各程式的資料夾, 也就是各程式要來收信的資料夾
dp2db_mail_dir=/usr/local/sgdg/dp/dp2db
dp2xml_mail_dir=/usr/local/sgdg/dp/dp2xml
dp2ms_mail_dir=/usr/local/sgdg/dp/dp2ms
############################################################
#     Part Of DB(DataBase)                                 #
############################################################
//   cParser 和 jConnector 掃描目錄內是否有新增檔案, 再做動作 的頻率（以 秒 爲單位）
db_scan_ratio= 6
//   jConnector 和 cParser 做同步用的目錄路徑
db2sg_mail_dir= /usr/local/sgdg/db/db_to_file
sg2db_mail_dir= /usr/local/sgdg/db/file_to_db
db2sg_tmp_dir= /usr/local/sgdg/db/db_to_file/temp
sg2db_tmp_dir= /usr/local/sgdg/db/file_to_db/temp
//   Java 的 DB Driver 所要用的各參數
db.addr= 140.134.25.98
db.port= 5000
db.dbName= ISEC
db.username= isecuser
db.password= ??????
//   cParser 和 jConnector 中間的中間檔的所有 Header, 格式 db.TableName
db.M_MAIN_IN= FROM_ID, FROM_NAME, TO_LIST, CC_LIST, BCC_LIST, SUBJECT, DATE,
MESSAGE_ID, SIZE, MISSION_ID, OWNER_ID, OWNER_NAME, SEVER_ID, COPY, TRASHCAN,
PERUSE
db.M_MAIN_OUT= FROM_MAIL, TO_LIST, CC_LIST, BCC_LIST, SUBJECT, DATE,
MESSAGE_ID, BODY, ATTACH, COPY, TRASHCAN, PERUSE
//   db->jConnector->cParser->出了 cParser 後的 Mail 的 Header
db.MIME_Header= From, To, Cc, Bcc, Subject, Date, Message-ID, Copy, Trashcan, Peruse, Mission-ID
```

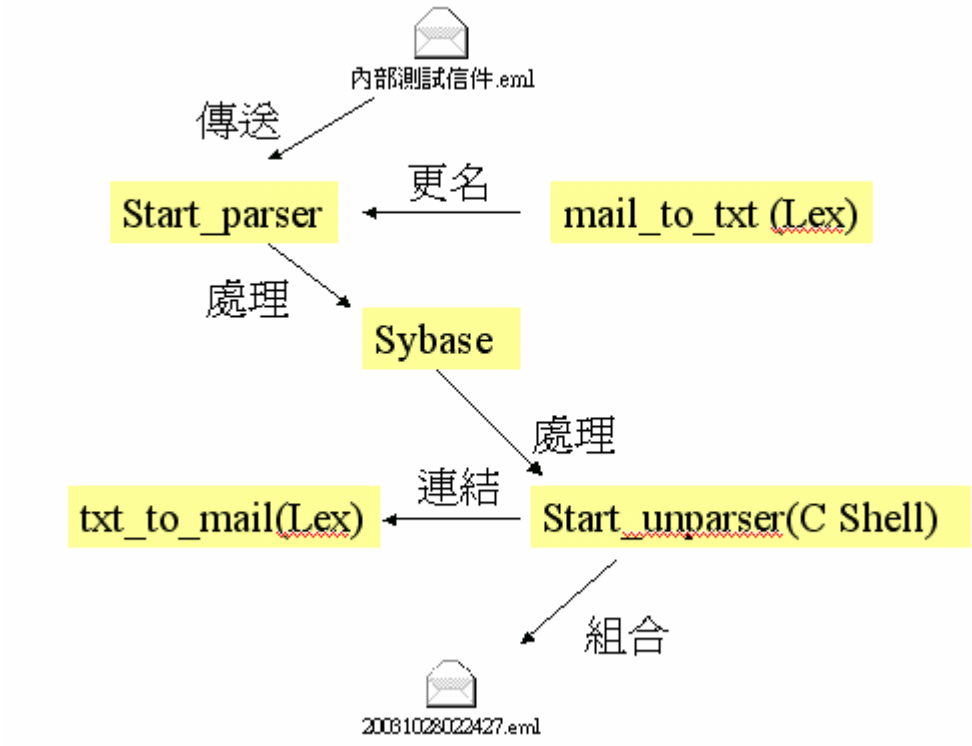
表十

4.2.2 總體流程圖



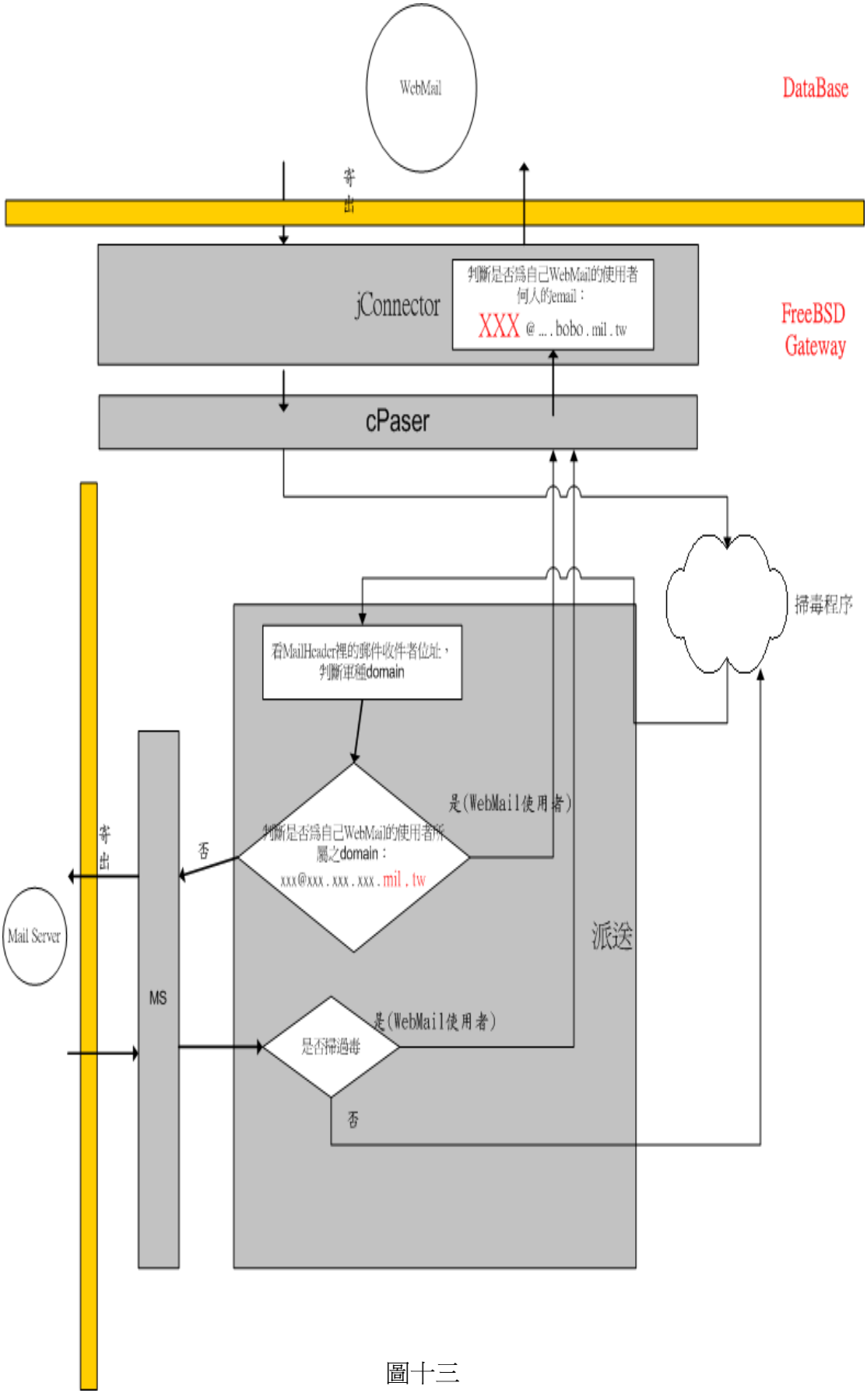
圖十一

4.2.3 Cparser 流程圖



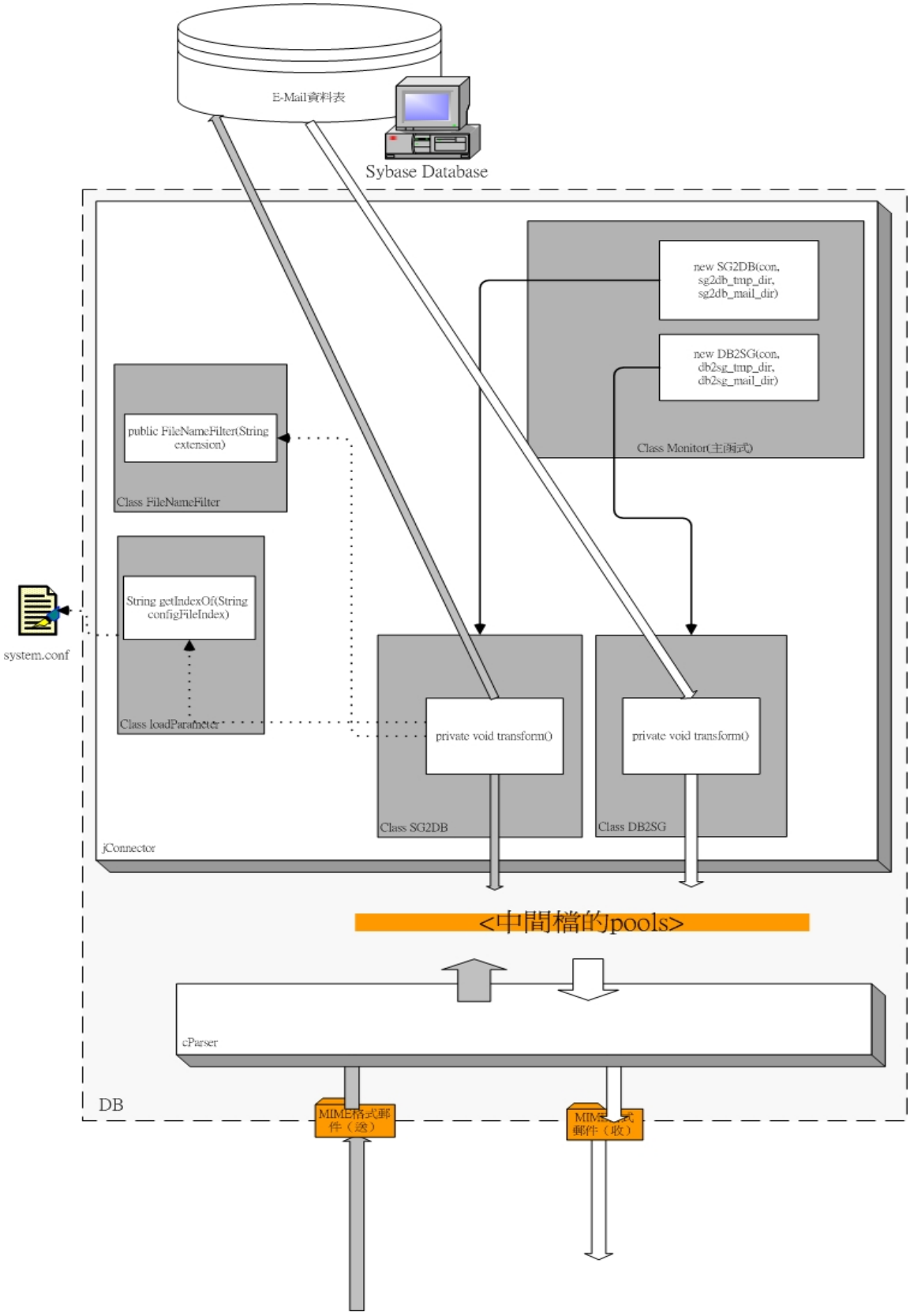
圖十二

4.2.4 DP(派送) 流程圖



圖十三

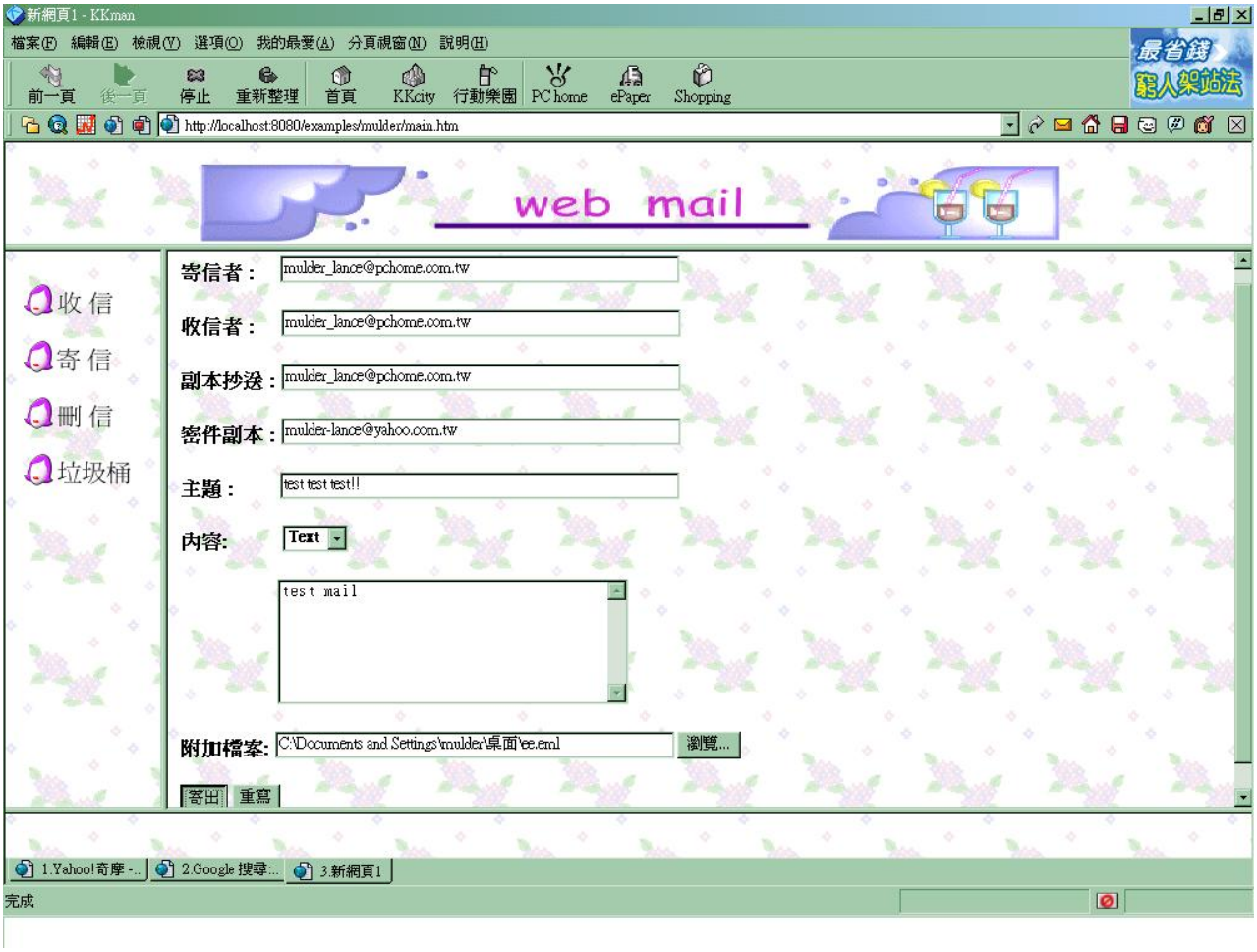
4.2.5 jConnect 流程圖



圖十四

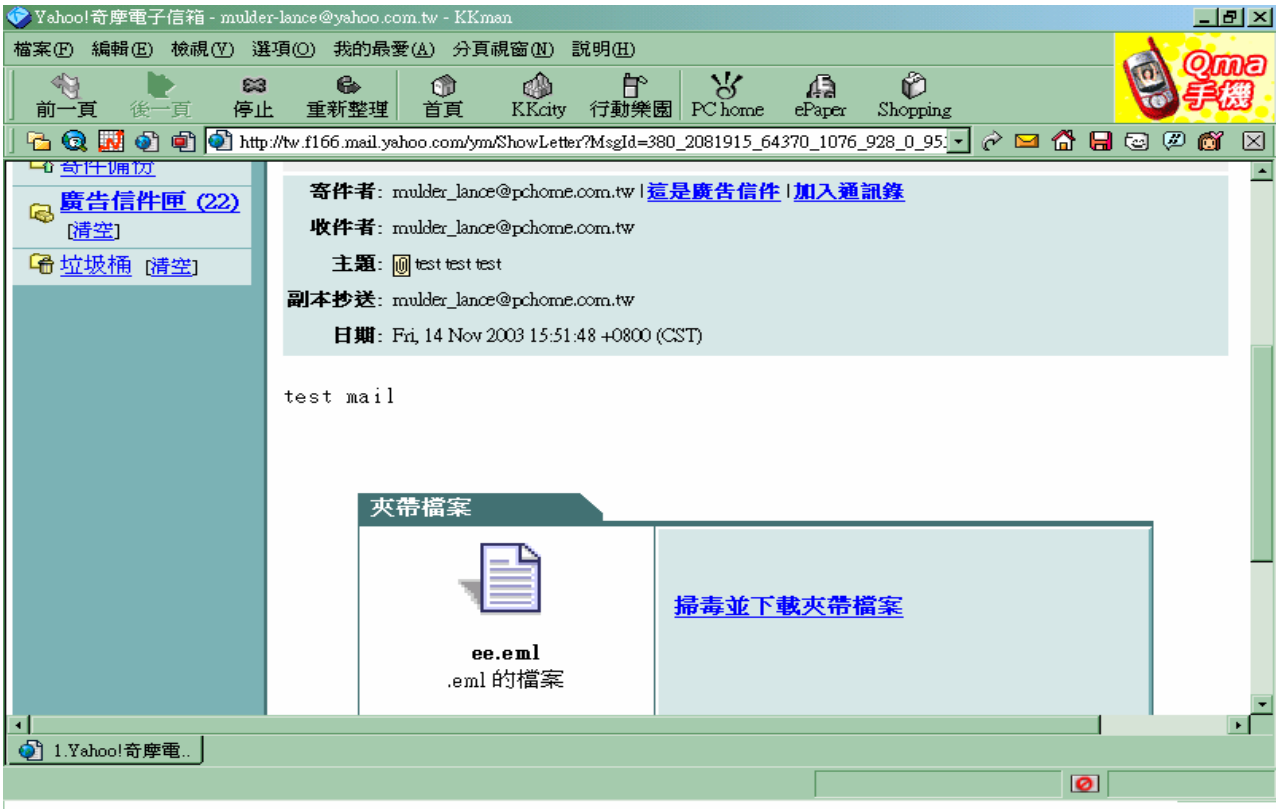
4.3 WebMail 之 JavaMail SMTP 協定寄信

用 text 格式寄信畫面



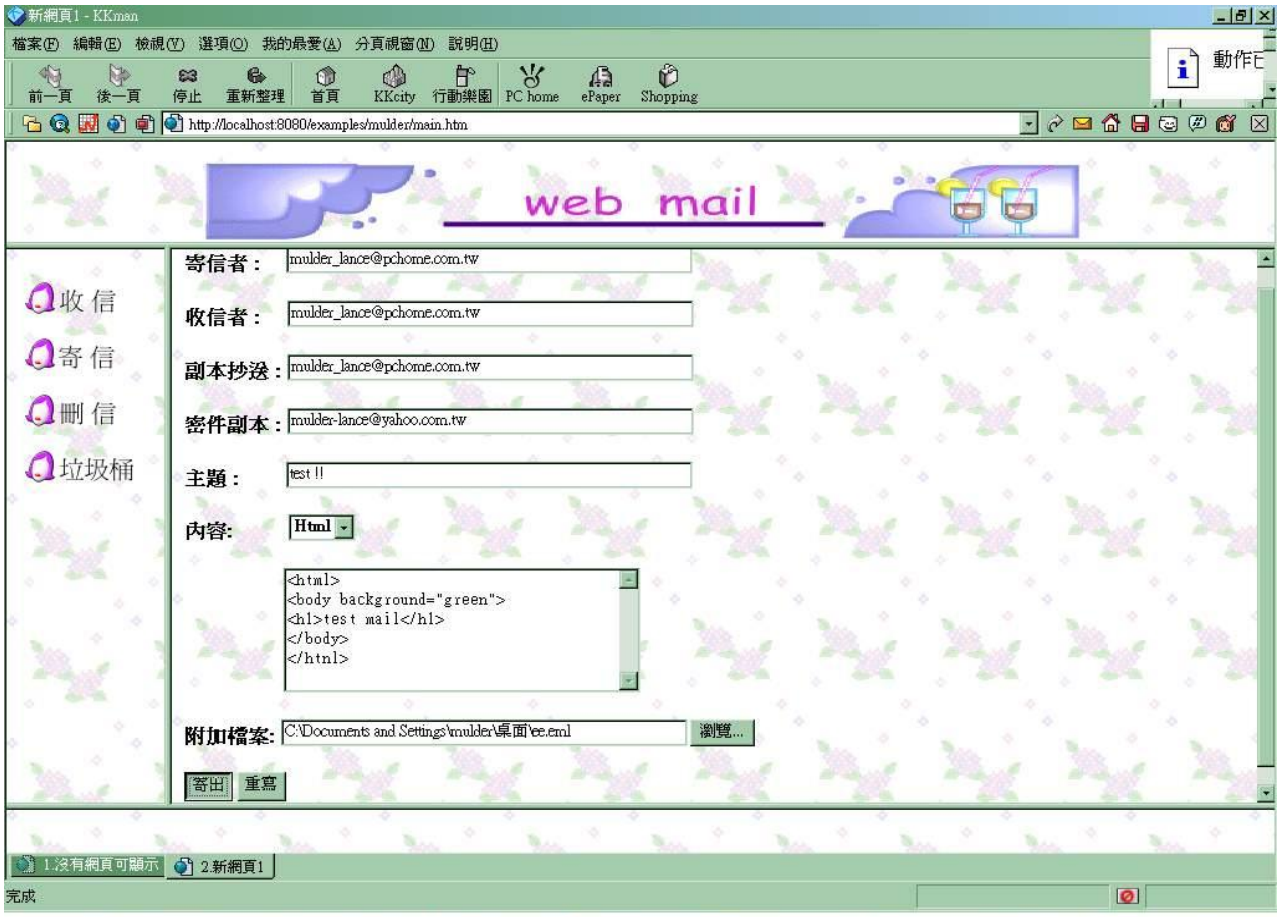
圖十五

收到信件的畫面


Yahoo!奇摩電子信箱 - mulder-lance@yahoo.com.tw - KKman
檔案(F) 編輯(E) 檢視(V) 選項(O) 我的最愛(A) 分頁視窗(N) 說明(H)
前一頁 後一頁 停止 重新整理 首頁 KKcity 行動樂園 PC home ePaper Shopping
廣告信件匣 (22)
[清空]
垃圾桶 [清空]
寄件者: mulder_lance@pchome.com.tw | 這是廣告信件 | 加入通訊錄
收件者: mulder_lance@pchome.com.tw
主題: test test test
副本抄送: mulder_lance@pchome.com.tw
日期: Fri, 14 Nov 2003 15:51:48 +0800 (CST)
test mail
夾帶檔案
ee.eml
.eml 的檔案
掃毒並下載夾帶檔案
1.Yahoo!奇摩電..

圖十六

用 html 格式寄信畫面



圖十七

收到信件的畫面

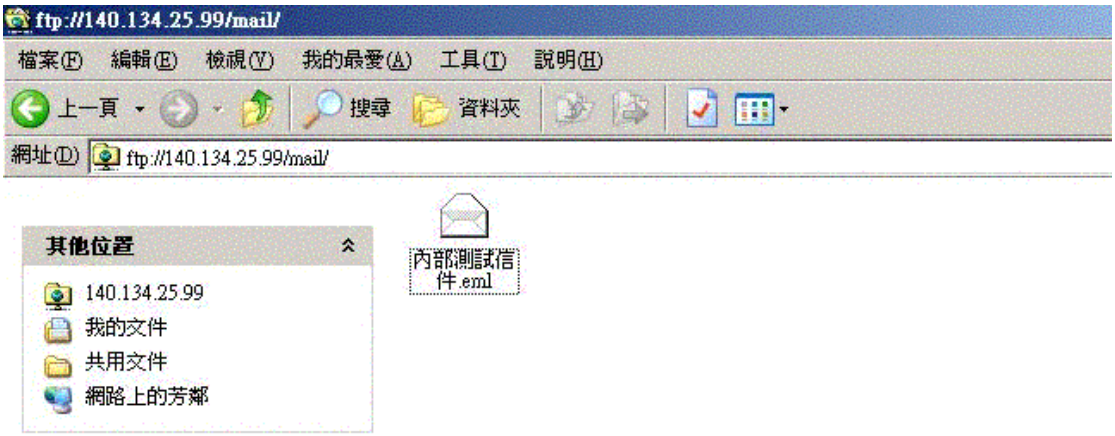

Yahoo!奇摩電子信箱 - mulder-lance@yahoo.com.tw - KKman
檔案(F) 編輯(E) 檢視(V) 選項(O) 我的最愛(A) 分頁視窗(N) 說明(H)
前一頁 後一頁 停止 重新整理 首頁 KKcity 行動樂園 PC home ePaper Shopping
Switch Hub
http://tw.f166.mail.yahoo.com/ym/ShowLetter?MsgId=275_2079879_64015_1068_966_0_95
廣告信件匣 (22)
[清空]
垃圾桶 [清空]
寄件者: mulder_lance@pchome.com.tw | 這是廣告信件 | 加入通訊錄
收件者: mulder_lance@pchome.com.tw
主題: test !!
副本抄送: mulder_lance@pchome.com.tw
日期: Fri, 14 Nov 2003 15:51:02 +0800 (CST)
test mail
夾帶檔案
ee.eml
.eml 的檔案
掃毒並下載夾帶檔案
1.Yahoo!奇摩電..

圖十八

4.4 cParser 與 jConnect 實作

用所架的 Mail Server 收信至資料庫

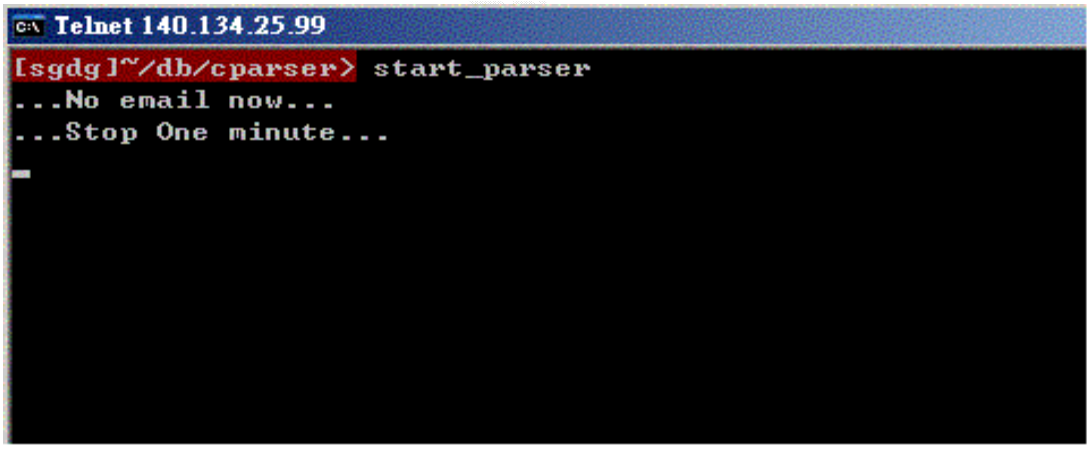
Step1. 在把 Mail Server 的 Unix 格式信件 parser 成一封一封的 MIME 格式信件後，的情形



圖十九

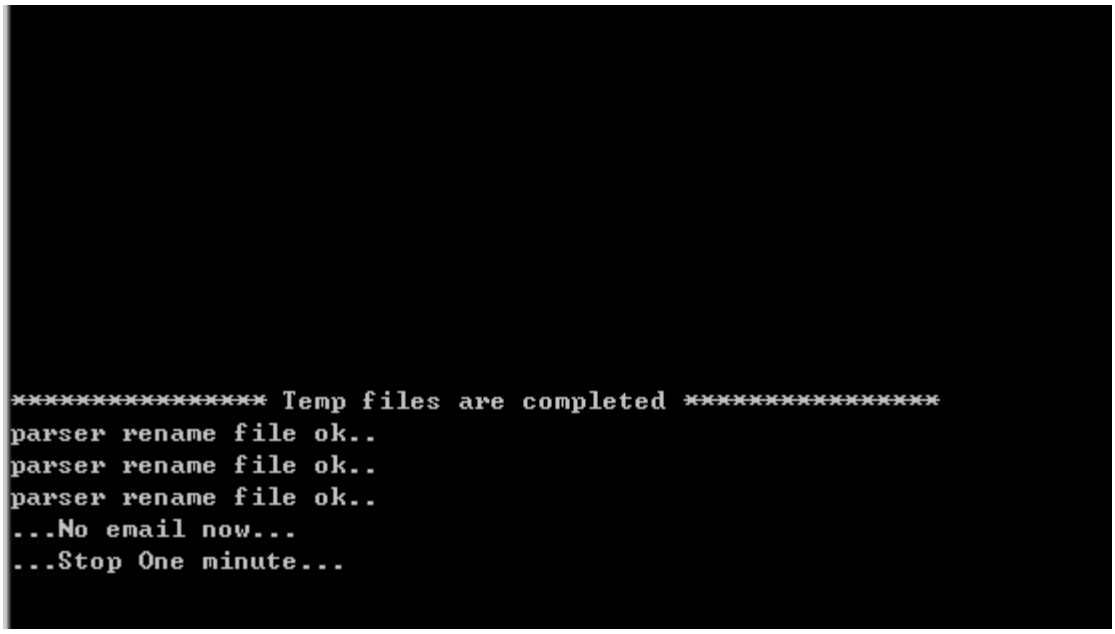
Step2. 看看 cparser 的 parser 情形

偵測到有新進的 mail



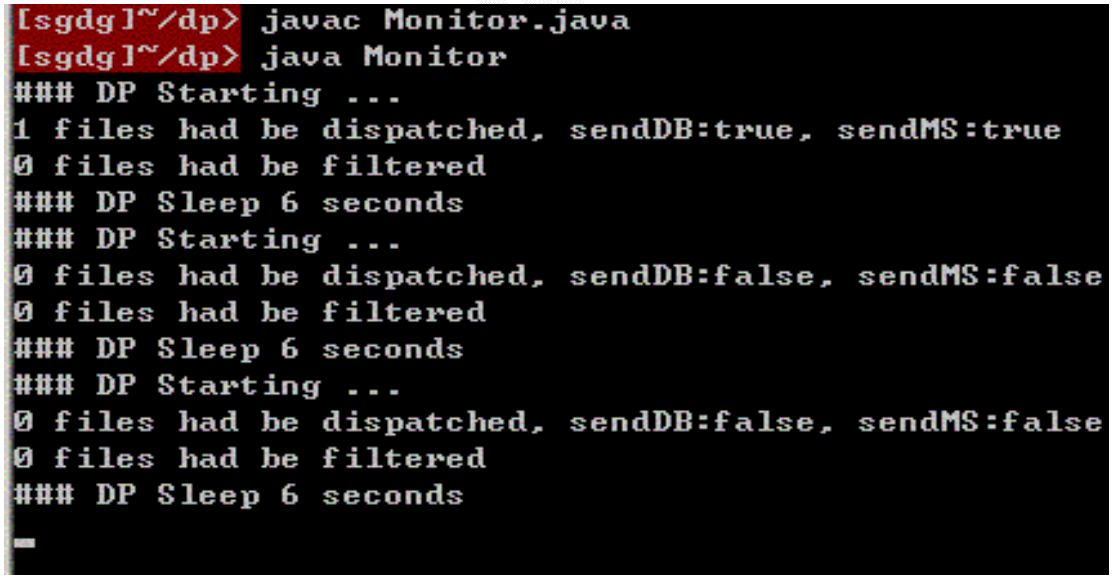
圖二十

開始進行 parser



圖二十一

Dispatcher 派送，因該信含有要寄給此 Web Server 的使用者和非此 Server 的使用者，所以派送也確實有往二邊送



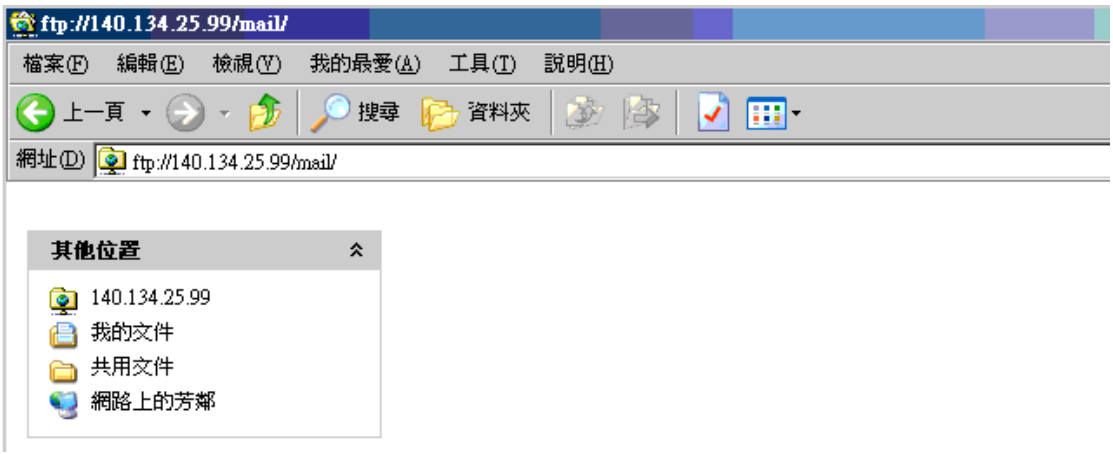
圖二十二

看看 DB 和 MS 的資料夾收到派送 relay 的 mail

圖二十三

圖二十四

回到原來新進郵件的地方看，郵件 parser 完已被 DP 轉送到到準備讓 jConnector 進行存入資料的資料夾

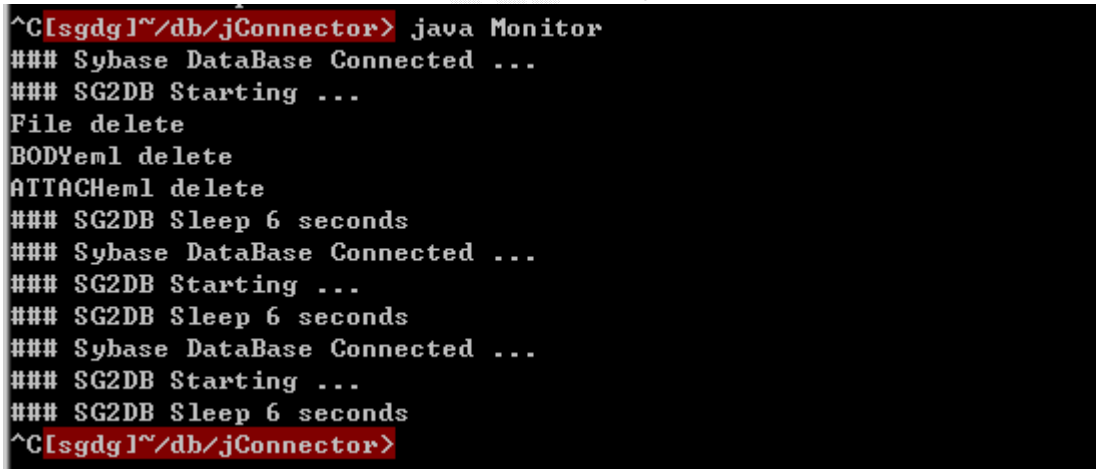


圖二十五

看一下，果然資料來到了 jConnector 的地方了，而且還把 BODY 和 ATTACH 的部份獨立出來了呢！

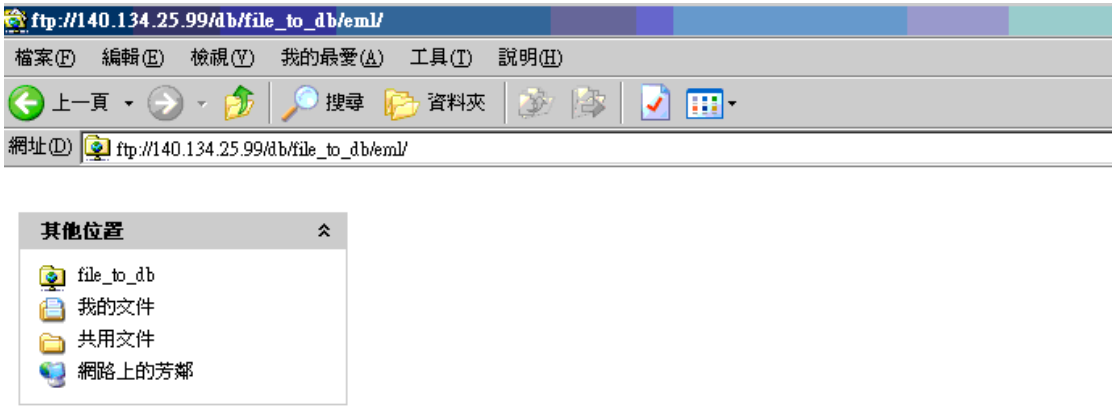
圖二十六

Step3. 再回去看另外的 jConnector 的工作情形 jConnector 也偵測到了新郵件喔！也成功地把郵件存入了 Sybase Database 囉



圖二十七

我們再看看原來的郵件是否還在，哇～果然不見了呢！



圖二十八

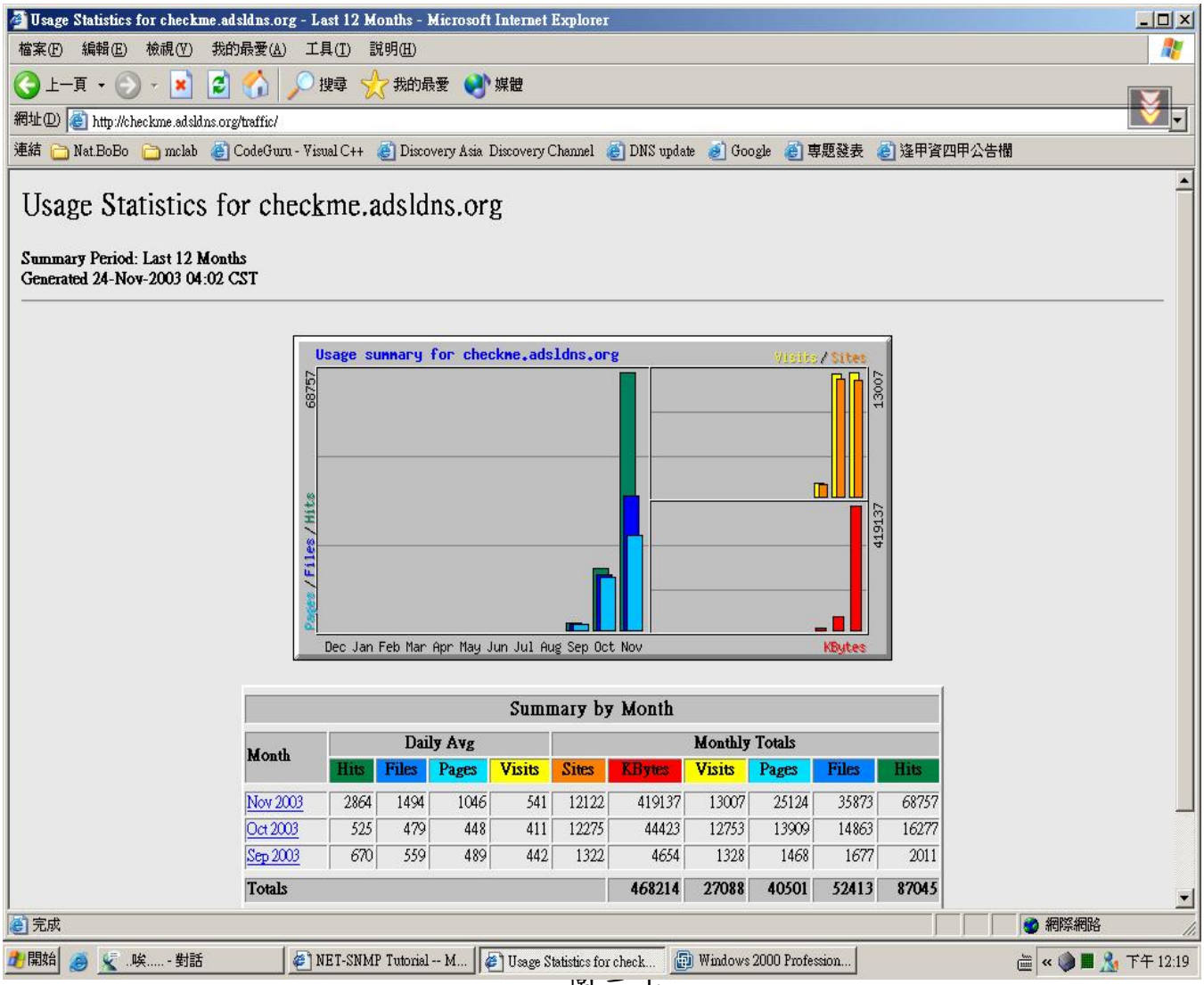
最後，到WebMail收信看看，該使用者也已收到了

圖二十九

4.5 WebServer 的 Througthput 及效能分析

4.5.1 Webalizer網頁記錄分析介紹

在我們將所製作好的網頁放在網頁伺服器之後，我們會想知道每一個網頁的瀏覽次數，使用者的停留時間等等。在apache的連線記錄中，可以看到很多使用者瀏覽網頁的記錄，我們可以使用一些工具來分析記錄檔，讓這些記錄檔能更易於閱讀。我們在這裡要介紹webalizer這套軟體。Webalizer會讀取apache的log檔，並將分析結果存成網頁，讓我們可以經由網頁的圖形更輕鬆的了解每一個頁面的使用情形。由於webalizer所進行的分析是經由讀取log檔，因此log檔的資料越多，分析出來的結果也就越詳盡。



圖三十

4.5.2 SNMP & MRTG 介紹

MRTG 流量分析，是一個可以從支援 SNMP 網路設備中取得流量資訊，然後分析這些資訊，繪成網頁格式圖表的工具. 它可以讓網管人員，很快地藉由流量負載， 來判斷網路或設備發生問題的可能原因.且要瞭解 MRTG 的運作，就必須瞭解一下 SNMP (Simple Network Management Protocol) 這個協定，因為 MRTG 是透過 SNMP 協定來監控流量的。

所以，所有的 MRTG 所偵測的裝置都必須符合 SNMP 的協定。那什麼是 SNMP 呢？簡單的說，就是一種可以提供裝置（主機設備）的各類資訊的一種協定，諸如：網路流量、主機名稱、CPU用量等等的資訊都可以藉由此一協定來提供。不過，由於不同廠牌的裝置可能會有無法相容的情況，因而後來又有所謂 MIB (Management Information Base) 的協定產生。不論如何，MRTG 就是藉由 SNMP 這個協定來監測與取得相關的資訊以製作圖表的！

SNMP也有抓取CPU和MEM的資訊，所以我們也可以直接設定取用。另外也可以使用到系統程式，諸如 systat、netstat ...。

4.5.3 webalizer的裝設

1.先到/usr/ports/www/webalizer把軟體make install好

2.再設定把/usr/local/etc/webalizer.conf-dist複製到
/usr/local/etc/webalizer.conf

3.設定我們自己的webalizer.conf

```
########################################################
# 只列我們有修改到的參數，不列出表示我們使用預設值，
# 預設值請見/usr/local/etc/webalizer.conf-dist
########################################################
# 設定Apache連線記錄檔的位置
LogFile         /var/log/httpd-access。log

# 在網頁根目錄建好traffic的資料夾， 放置分析資料的位置
OutputDir       /usr/local/www/data-dist/traffic

# 由於我們可能會設定某一段時間自動將Apache的log壓縮或刪除。而Incremental這
# 個變數可以讓我們在產生分析資料時，只更新增加的部份，
# 而分析過的資料就不再分析，以免覆蓋了舊有的資料
Incremental     yes

# PageType可以讓你設定何種副檔名結尾的頁面要加入分析資料。 因爲在log檔中有一些圖片，
# 而這些圖片我們並不希望加入分析資料中，或者我們也可以增加php頁面的分析資料。
# 所以在這裡，我們加上二行用來分析PHP頁面和jsp頁面
PageType  htm*
PageType  cgi
PageType  php
PageType  jsp
#PageType phtml
#PageType php3
#PageType pl

# 定時執行
Quiet           yes

# 設定顯示每日分析資料
DailyGraph      yes
DailyStats      yes

# 設定顯示每小時分析資料
HourlyGraph     yes
HourlyStats     yes

# 設定顯示圖表的格線數量，我們設到最多
GraphLines      20
```

表十一

接著在/etc/crontab裡加入：

```
10      *       *       *       *       root    /usr/local/bin/webalizer
```

讓系統每小時的第１０分執行一次webalizer

4.5.4 SNMP的裝設

用ports裝snmp

```
# cd /usr/ports/net/net-snmp
# make install clean
```

安裝完成後，我們必須先新增一個可以讀取SNMP資訊的community name。所謂community name是一個明碼的字串，我們可以將它視為management station和agent之間的密碼，是MRTG和net-SNMP溝通時必須要先傳送的字串。

我們可以依不同的網域或主機給予不同的權限，依community name的設定來決定不同權限。一個網路元件可以有多個community name，一般SNMP Agent所預設公開的community name 是public。我們不一定要將community name設定為public，因為public是一般SNMP的預設值，為了安全問題，我們不將它設為public。

這裡我們設定community name為mrtg，而且只有read only的權限。

要設定community name就要先新增一個文字檔/usr/local/share/snmp/snmpd.conf並加入下列設定：

```
rocommunity     mrtg

syslocation     Your_Host_Name

syscontact      mrtg@localhost
```

上面的設定中，mrtg為唯讀的community name，Your_Host_Name是你機器所在的位置，而syscontact所接的字串是你的E-mail。

另外我們還須在/etc/rc.conf中加入

```
net_snmp_enable="YES"
```

啟動snmp

```
/usr/local/etc/rc.d/snmpd.sh start
```

4.5.5 v的裝設

用ports裝mrtg

```
# cd /usr/ports/net/mrtg
# make install clean
```

產生mrtg設定檔在/home/mrtg/public_html資料夾

```
# cfgmaker --global "WorkDir:
/home/mrtg/public_html" \
        --global "Options[_]: growright" \
        --ifref=nr mrtg@localhost >
/usr/local/etc/mrtg/mrtg.cfg
```

接著在設定檔/usr/local/etc/mrtg/mrtg.cfg裡加入我們要的設定，如中文化、用byte或bit統計 … 等等。
我們的設定檔請見<附錄F>

接著把MRTG的一些圖片複製到mrtg的目錄裡

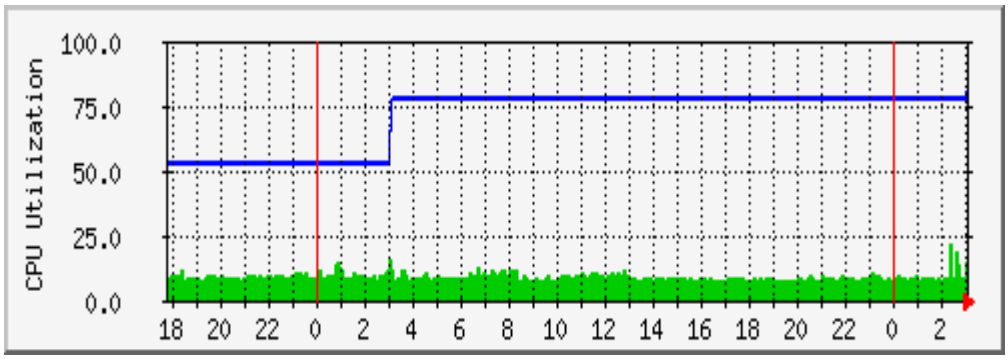
```
# cd /usr/ports/net/mrtg/work/mrtg*
# cd images
# cp * /home/www/mrtg/
```

最後，啟動mrtg

```
#/usr/local/bin/mrtg  /usr/local/etc/mrtg/mrtg.cfg
```

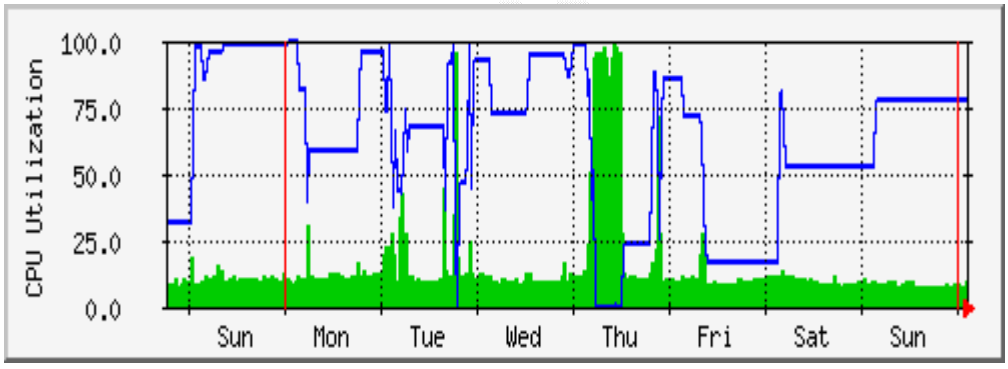
MRTG所分析的cpu使用量可以分為

每日 圖表（5 分鐘 平均）



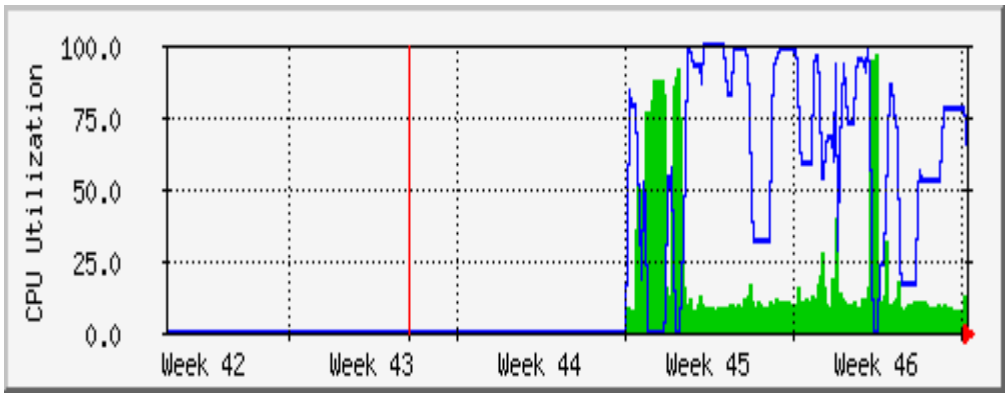
圖三十一

每週 圖表（30 分鐘 平均）



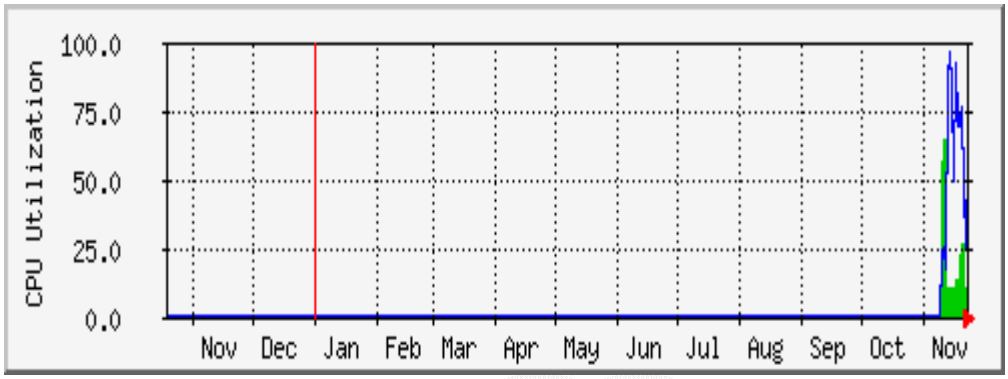
圖三十二

每月 圖表（2 小時 平均）



圖三十三

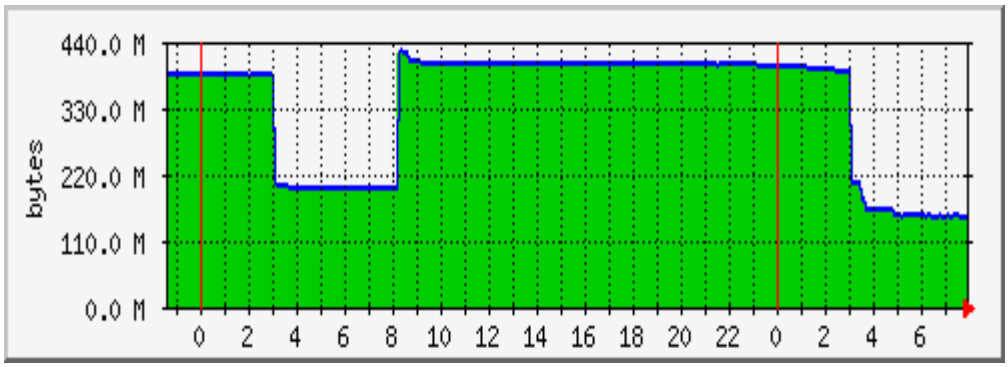
每年 圖表（1 天 平均）



圖三十四

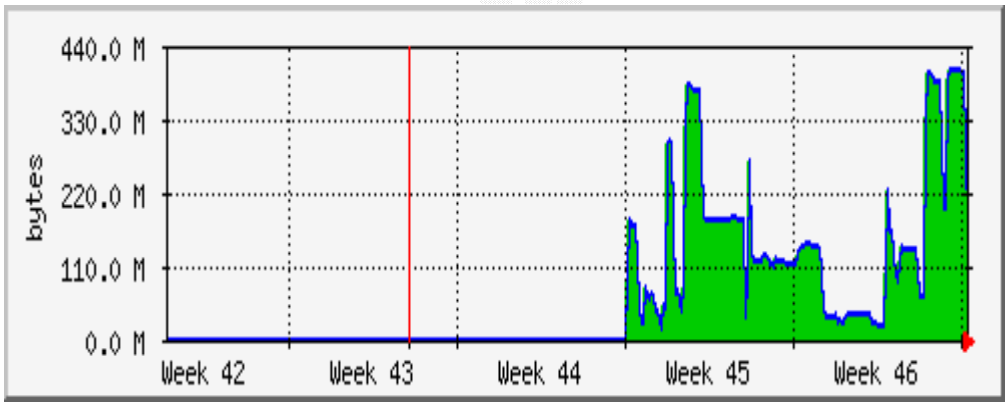
MRTG所分析的Memory使用量可以分為

每日 圖表（5 分鐘 平均）



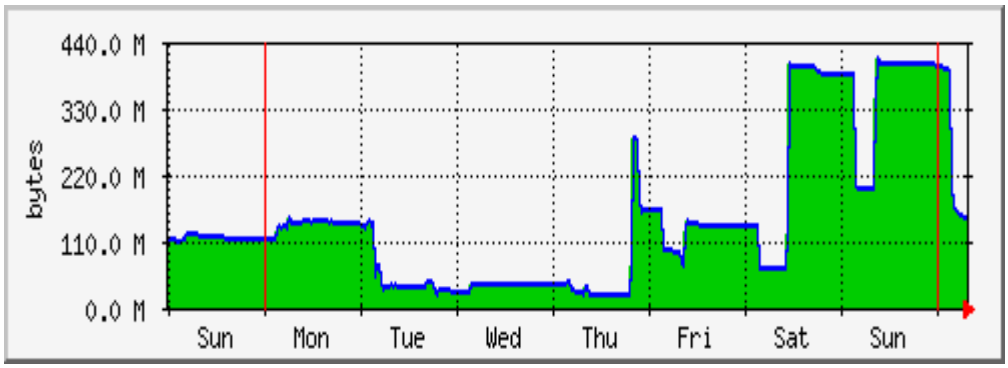
圖三十五

每週 圖表（30 分鐘 平均）



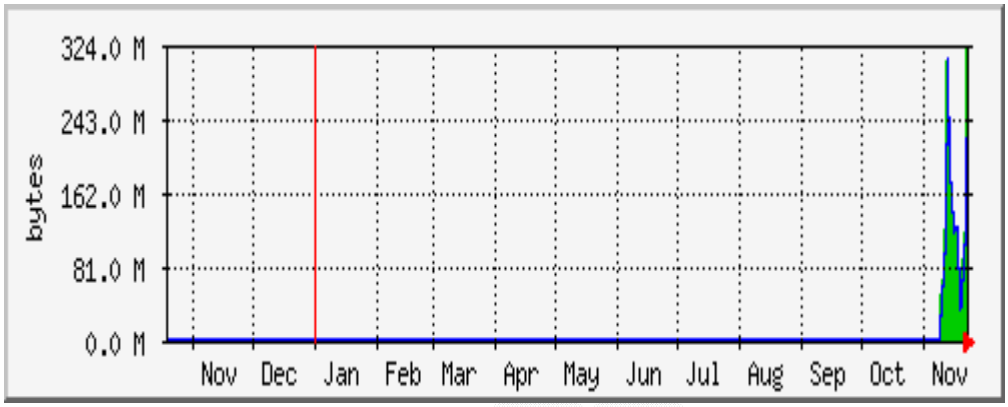
圖三十六

每月 圖表 (2 小時 平均)



圖三十七

每年 圖表 (1 天 平均)



圖三十八

那我們以每日 圖表 (5 分鐘 平均)來看一下分析，在圖中可以有最大.平均.目前的流量分析並以藍.綠色來分辨user和 Idle CPU狀態圖，當使用者在使用cpu時，cpu不一定會忙碌中，且和在free memory中的圖比較一下，一定可以發現使用cpu時free memory必大大降低，所以為了讓cpu在閒置的後能多做些些事情，在每週 圖表 (30分鐘 平均)，可以看出每日user使用cpu負荷直飆上漲，
那時候正在整合程式與測試信件的效能所以cpu的資源吃的非常兇，因為本系統中因為隨時保有3個程式在執行(cparser 與 jconnect與Dispatcher)，通常要等到一項行程一直執行等到他必須等待什麼的時候才停止，

一般來講都是等待一些I/O的要求，在一個簡單的系統之中，此刻cpu只能為閒置狀態，所以這類的等待都是浪費時間的，一點有用的工作也沒有，所以在cparser做I/O儲存到資料夾的同時，會把cpu的控制存轉交給jconnect，等到cpu處理jconnect程式中判斷有沒有信件進來eml的資料夾，後cparser所使用的I/O會發出中斷訊息給cpu，並此時jconnect正處於停止一分鐘狀態，且把cpu經查詢中斷向量表後，在繼續處理cparser的I/O處理，在組合信件的時候也一樣，所以在cpu使用率可以比沒有使用一分鐘的機制好，不會cpu常常閒置在那邊。

第五章　結 論

5.1 遭遇困難與解決方法

在這一次的專題裡，遭遇最多的困難應該是在規格的定義，因為信件的規格形式有太多種，且欄位的順序與規格都不一。至於我們內部程式所遇到比較困難的問題是在判斷一封信裡面的欄位，因為一封Email裡面的欄位值可能有重複，像是boundary可以出現在內文與附加檔案的邊界，那如何來判別哪一個是內文和哪一個是附加檔，使用Content-Type:來分別，但Content-Type:又可以出現在內文與附加檔裡，所以又用了name：這個欄位來分出內文與附加檔案，因為name：這個欄位只有出現在附加檔案有；且在Cparser與Jconnect程式內部中所遇到同步的問題我們是以時間來控制以及temp檔案來解決，因為當parser一封信變成3個eml的途中，如果Jconnect來讀取pasrer的eml，一定會發生同步的問題且檔案儲存到資料庫會不完全，所以利用了一分鐘的時間來控制；當parser在讀取信件時，Jconnect就會去判斷parser後暫存資料夾是否為空，如果為空，則表示信件已經完全parser與儲存到真正要被資料庫讀取的資料夾，反之從資料庫讀取出來的信件也事先存成temp檔在轉成eml並加入一分鐘機制來判斷同步的問題，因此檔案就可以被正常處理，系統正常運作。

5.2 心得

此專題完成到現在，花了最久的時間是去研究那個一分鐘的同步機制和暫存檔的運用也是花了很久的時間來制定的，且MIME格式都是原文說明，更需要花一番功夫來研究。還有此專題的原本環境是在Windows發展的，但後來因為發生疾風病毒的影響，原本用Asp來發展的東西，改成了Jsp，並且在Unix-like下的環境發展，一剛開始對於這種環境陌生的我們，去圖書館借了好幾本書回來研讀，才有能力來發展程式，一路走來所看的書應該不少於十幾二十本，感覺獲益良多。除此之外，在分工與溝通之間，我們彼此有學到了不少東西，像是程式不懂的地方都會互相研討並加以溝通，所以到最後不會因為各自完成的程式到最後無法整合，這些都是我們在此專題中學到的。

附錄 A 使用手冊

1.1 The java Connect Database Engine（jConnector）

jConnector 利用到了 Sybase 提供的 jconnect55 的 API, 來做 Sybase Database 存取
1. 它包含以下幾支類別檔:
==========

Monitor.java　　jConnector 的各類別檔均用 Monitor.java 來控制，
包括參數值和程式運作流程。
簡單的說，Monitor.java 是為好掌控而設計的主類別檔。
故在啟動 jConnector 時，只須在 compiler 後，執行 "#java Monitor" 即可。

SG2DB.java　　工作:
轉換 Database 的 M_MAIN_OUT Table 的待寄出郵件 --> cParser 和 jConnector 的中間檔。
Step:
1. 過濾副檔名，只對副檔名為*.M_MAIN_IN.eml 的做事
2. 取得暫存檔和中間檔(cParser 和 jConnector 同步用)的目錄內的
所有檔案或資料夾的物件
3. cParser 和 jConnector 階段性工作的判斷:
a)Temp 資料夾是否為空(為空表示 cParser 做完階段性工作)
b)再判斷 Eml 資料夾是否為空(不為空表示有 mail 要存到 DB 了)
c)Temp 資料夾為空 && Eml 資料夾不為空時, jConnector 才工作
4. 分析一中間檔名:
XXXXXXXXXXXXXXXXX.table_name.eml
^^^^^^^^^^^^^^^^^ ^^^^^^^^^^ ^^^
同步字串　　Table 名稱　副檔名
同步字串共 17 個 char 長度
a)利用 Class mail_hash 來取出中間檔(M_MAIN_IN)內對應 columns 的值
b)且 ATTACH.eml 和 BODY.eml 的內容分別直接 dump 進 ATTACH 和 BODY 欄位
5. 對單封 mail, 得到所有需收到 mail 的人的 E-Mail（包括 To, Cc, Bcc）
6. a)根據該封所有 E-Mail, 儲存該 Mail 到 Database 的 Table,
且所有收件者(OWNER_ID)都需收到一份
b)重覆 a 直到所有要派送的 mail 都送完
7. 移除已存入資料庫的檔案

DB2SG.java　　工作:
轉換 cParser 和 jConnector 的中間檔 --> Database 的 M_MAIN_IN 和 M_RECEIVER 二 Table。

Step:
1.取 Database 的 table 裡每一 row 值, 存成存成 files,
2.然後清空已被取出的 row

1.2 The java Dispatcher (DP)

DP 包含以下幾支類別檔:

==========

Monitor.java　DP 的各類別檔均用 Monitor.java 來控制，包括參數值和程式運作流程。
簡單的說，Monitor.java 是爲好掌控而設計的主類別檔。
故在啓動 DP 時，只須在 compiler 後，執行 "#java Monitor" 即可。

Dispatcher.java　Step:

1. 過濾副檔名，只對副檔名爲*.eml 的做事
2. 取得暫存檔和中間檔(VS 和 DP 同步用)的目錄內的所有檔案或資料夾的物件
3. VS 和 DP 階段性工作的判斷:
   a)Temp 資料夾是否爲空(爲空表示 VS 做完階段性工作)
   b)再判斷 Eml 資料夾是否爲空(不爲空表示有 mail 要派送了)
   c)Temp 資料夾爲空 && Eml 資料夾不爲空時, DP 才工作
4. 分析一中間檔名:
   XXXXXXXXXXXXXXXX.eml
   ^^^^^^^^^^^^^^^^ ^^^
   同步字串　副檔名
   同步字串共 17 個 char 長度
5. 對單封 mail, 得到所有需收到 mail 的人的 E-Mail
6. 派送:
   a) 內容過濾
   b) 根據該封所有 E-Mail,
      判斷須把該郵件送至 MS 或 DB 或 XML 或 ILLEGAL_MAIL_DIR
   c) 複製該郵件至該到達的地方(MS 或 DB 或 XML 或 ILLEGAL_MAIL_DIR)
   d) 重覆 b 直到所有要派送的 mail 都送完
7. 移除已派送的 mail

ContentFilter.java　利用 Javal.4 的 Reglar Expression,
可以從~/conf/filter.conf 裡讀出所有要過濾的關鍵字
信件裡只要符合關鍵字, 就可以做過濾的動作

1.3 jConnector 和 DP 共用的類別檔

jConnector 和 DP 用到的共用類別檔:

==========

FileNameFilter.java　設計用來做附檔名過濾用，因爲 jConnector 只處理.eml，並不對其它檔案做動作。

<呼叫方法>

FileNameFilter fileFilter = new FileNameFilter(我要的副檔名字串);

File tempDir = new File(目錄路徑);

File[] DirContents = tempDir.listFiles(fileFilter);

--> 可以"只"得到我要的副檔名的檔案

例如你過濾副檔名字串取 java

則做抓取所有檔案(File[])時則只會抓到我要的*.java

loadParameter.java　設計來對 system.conf 與 cparser 和 jConnector 的中間檔做處理的

要得到 system.conf 裡 index 值，

用 String getIndexOf(String fileIndex)要得到中間檔 index 的值，

用 Map collect(FileReader file, String table_name)。

在 loadParameter.java 的程式頭有詳細的規格說明。

emlAddrParser.java　a) Email 的 Address 的格式有二種

1. <name> and <address>

2. <address> only.這裡 address(aaa@bbb.cc)再分成

2.1 id : aaa

2.2 Domain : bbb.cc

b) SGDG 預設的 E-Mail 格式:

" Name " < ID @ Domain >

WebMail 在產生時須限制的

c) 限制: 使用者的 Name 裡不允許用在 Parser 時的 delimiter:"和 ,

d) 使用方式:

1. new 新的類別:

emlAddrParser eap = new emlAddrParser();

2. 分別 parser 你需要的值

eap.getNickName(Mail_Str),

eap.getAddr(Mail_Str),

eap.getID(Mail_Str),

eap.getDomain(Mail_Str),

```
eap.getMailDist(Mail_Str)
```

範例:

```
...
emlAddrParser eap = new emlAddrParser();
String Mail_Str = "\"aaa\" <xxx@yyy.zzz>";
System.out.println(eap.getNickName(Mail_Str) +
           ", " + eap.getAddr(Mail_Str)  +
           ", " + eap.getID(Mail_Str)  +
           ", " + eap.getDomain(Mail_Str)  +
           ", " + eap.getMailDist(Mail_Str));
```

1.4 The Lex Parser (cParser)

mail_to_txt.l　　使用方式：
lex mail_to_txt.l (之後會產生一個 lex.yy.c)
cc lex.yy.c -o start_parser -ll (之後會產生一個 mail_to_txt 執行檔)，
之後在把 lex.yy.c 改成 mail_to_txt.c

txt_to_mail.l　　使用方式：
lex txt_to_mail.l (之後會產生一個 lex.yy.c)
cc lex.yy.c -o txt_to_mail -ll (之後會產生一個 txt_to_mail 執行檔)，
之後在把 lex.yy.c 改成 txt_to_mail.c

parser 和 unparser　start_parser(表示開始執行 parser email)
start_unparser(表示開始執行 unparser email)

附錄 B 過濾設定檔 - filter.conf

```
#################################################################
#                                                               #
#   File: Content Filter Keywords                               #
#   Group: DP (Dispatcher)                                      #
#   Introduce:                                                  #
#     This is a set of keywords                                 #
#     that is not allowed in any e-mails.                       #
#   使用上請注意:                                               #
#     1. 英文字母大小寫是當作不同來區分                         #
#     2. 請以 "," 隔開每個 KeyWord                               #
#     3. 如在 KeyWord 這行裡要換行, 但下一行內容也是 KeyWord 的  #
#        則請在 KeyWord 的最後加 "," 或下一行的行頭加上 ","     #
#                                                               #
#################################################################

KeyWord= 幹, fuck, damage
         ,陳水扁是笨蛋,
```

附錄 C　用到的相關 RFC 及整理

==========

Internet Official Protocol Standards
中的 STD11(http://www.sri.ucl.ac.be/normes/rfc/rfc822.txt)裡面的定義了 E-Mail 的標準架構

==========

RFC 822

STD 11, defines a message representation protocol specifying considerable detail about US-ASCII message headers, and leaves the message content, or message body, as flat US-ASCII text. This set of documents, collectively called the Multipurpose Internet Mail Extensions, or MIME, redefines the format of messages to allow for

(1) textual message bodies in character sets other than US-ASCII,
(2) an extensible set of different formats for non-textual message bodies,
(3) multi-part message bodies, and
(4) textual header information in character sets other than US-ASCII.

RFC 2046

defines the general structure of the MIME media typing system and defines an initial set of media types

RFC 2047

describes extensions to RFC 822 to allow non-US-ASCII text data in Internet mail header fields

==========

RFC822(ARPA INTERNET TEXT MESSAGES)裡的

[Page16]MESSAGE SPECIFICATION 所定義的

1. when present, some fields, must be in a particular order
2. Header fields are NOT required to occur in any particular order, except that the message body must occur AFTER the headers
3. It is recommended that, if present, headers be sent in the order "Return-Path", "Received", "Date", "From", "Subject",

"Sender", "To", "cc", etc.

4.This specification permits multiple occurrences of most fields.

Except as noted, their interpretation is not specified here, and their use is discouraged.

5.Header 有 prefix 的, 也有不是 prefix 的

6.occurrence of legal "Resent-" fields are treated identically with

the occurrence of fields whose names do not contain this prefix.

各 Field 的 Header 功能:

01.MIME Header Fields

a)TRACE FIELDS - it indicates a route back to the sender of the message.

"Return-Path" - is used to identify a path back to the originator

"Reply-To" - is added by the originator and serves to direct replies

"Received" - A copy of this field is added by each transport

service that relays the message

b)ORIGINATOR FIELDS

"From"

"Sender"

"Reply-To"

c)RECEIVER FIELDS

"To" - the primary recipients of the message.

"Cc" - the secondary(informa-tional) recipients of the message.

"Bcc" - additional recipients of the message.

d)REFERENCE FIELDS

"Message-ID" - This field contains a unique identifier

(the local-part address unit)

which refers to THIS version of THIS message.

The uniqueness of the message identifier is guaranteed by the

host which generates it.

```
                "In-Reply-To" - The contents of this field identify
previous
                                correspon-dence which this message
answers.
                "References"  - The contents of this field identify
other correspondence
                                which this message references.
                "Keywords"    - This field contains keywords or phrases,
separated by commas.
            e)OTHER FIELDS
                "Subject"
                "Comments"
                "Encrypted"
            f)Extension-FIeld
                "X-"
            f)USER-DEFINED-FIELD - Non-Multiple with other header
                "***:"
            g)DATE AND TIME SPECIFICATION
                "Date"
        02.MIME-Version Header Field
            "MIME-Version"
        03.Content-Type Header Field
            "Content-Type"
        04.Content-Transfer-Encoding Header Field
            "Content-Transfer-Encoding"
        05.Content-ID Header Field
            "Content-ID"
        06.Content-Description Header Field
            "Content-Description"
        07.Additional MIME Header Fields -  Any RFC 822 header field
                                            which begins with the
string "Content-">
            "Content-"


        內文的部份:
         Multipart:
        -- Recognize the mixed subtype.  Display all relevant
```

```
        information on the message level and the body part
        header level and then display or offer to display
        each of the body parts individually.
        -- Recognize the "alternative" subtype, and avoid
        showing the user redundant parts of
        multipart/alternative mail.
        -- Recognize the "multipart/digest" subtype,
        specifically using "message/rfc822" rather than
        "text/plain" as the default media type for body parts
        inside "multipart/digest" entities.
        -- Treat any unrecognized subtypes as if they were
        "mixed".
```

==========

有關編碼:

RFC2047, RFC2049

==========

RFC 2047 Message Header Extensions

Generally, an "encoded-word" is a sequence of printable ASCII characters that begins with "=?", ends with "?=", and has two "?"s in between. It specifies a character set and an encoding method, and also includes the original text encoded as graphic ASCII characters, according to the rules for that encoding method.

附錄D　X-Window的XF86Config設定檔

```
Section "ServerLayout"
    Identifier     "XFree86 Configured"
    Screen      0  "Screen0" 0 0
    InputDevice    "Mouse0" "CorePointer"
    InputDevice    "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
    RgbPath      "/usr/X11R6/lib/X11/rgb"
    ModulePath   "/usr/X11R6/lib/modules"
    FontPath     "/usr/X11R6/lib/X11/fonts/TrueType"
    FontPath     "/usr/X11R6/lib/X11/fonts/local"
    FontPath     "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath     "/usr/X11R6/lib/X11/fonts/Speedo/"
    FontPath     "/usr/X11R6/lib/X11/fonts/Type1/"
    FontPath     "/usr/X11R6/lib/X11/fonts/75dpi/"
    FontPath     "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "Module"
    Load  "xtt"
    Load  "extmod"
    Load  "xie"
    Load  "dbe"
    Load  "dri"
    Load  "glx"
    Load  "record"
    Load  "xtrap"
    Load  "speedo"
    Load  "type1"
EndSection

Section "InputDevice"
    Identifier  "Keyboard0"
    Driver      "keyboard"
EndSection
```

```
Section "InputDevice"
    Identifier  "Mouse0"
    Driver      "mouse"
    Option      "Protocol" "auto"
    Option      "Device" "/dev/sysmouse"
EndSection

Section "Monitor"
    Identifier   "Monitor0"
    VendorName   "Monitor Vendor"
    ModelName    "Monitor Model"
    Horizsync    31.5-57.0
    VertRefresh  50-100
EndSection

Section "Device"
        ### Available Driver options are:-
        ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
        ### <string>: "String", <freq>: "<f> Hz/kHz/MHz"
        ### [arg]: arg optional
        #Option     "SWcursor"              # [<bool>]
        #Option     "HWcursor"              # [<bool>]
        #Option     "NoAccel"               # [<bool>]
        #Option     "ShowCache"             # [<bool>]
        #Option     "ShadowFB"              # [<bool>]
        #Option     "UseFBDev"              # [<bool>]
        #Option     "Rotate"                # [<str>]
        #Option     "VideoKey"              # <i>
        #Option     "FlatPanel"             # [<bool>]
        #Option     "FPDither"              # [<bool>]
        #Option     "CrtcNumber"            # <i>
    Identifier  "Card0"
    Driver      "nv"
    VendorName  "nVidia Corporation"
    BoardName   "NV5M64 [RIVA TNT2 Model 64/Model 64 Pro]"
    BusID       "PCI:1:0:0"
EndSection
```

```
Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    Monitor    "Monitor0"
    DefaultColorDepth 16
    SubSection "Display"
        Depth     1
    EndSubSection
    SubSection "Display"
        Depth     4
    EndSubSection
    SubSection "Display"
        Depth     8
    EndSubSection
    SubSection "Display"
        Depth     15
    EndSubSection
    SubSection "Display"
        Depth     16
        Modes "800x600" "1024x768"
        Virtual 800 600
        ViewPort 0 0
    EndSubSection
    SubSection "Display"
        Depth     24
    EndSubSection
EndSection
```

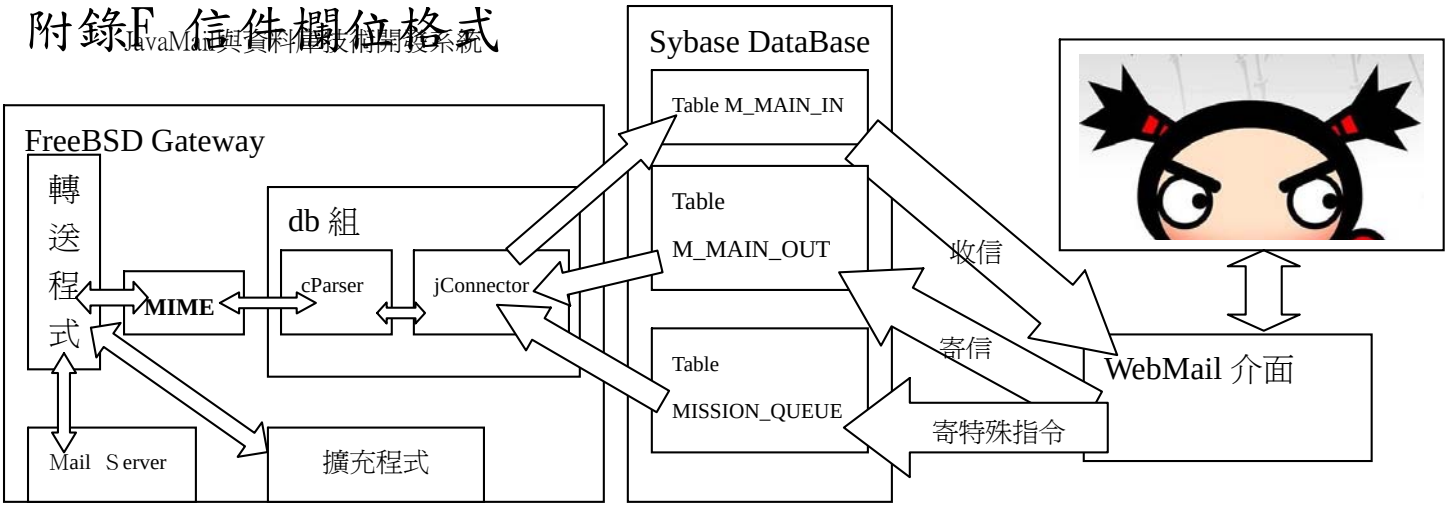
附錄 E　MRTG 設定檔

```
# Created by
# /usr/local/bin/cfgmaker --global 'WorkDir: /home/mrtg/public_html'
--global 'Options[_]: growright' --ifref=nr mrtg@localhost


### Global Config Options

#  for UNIX
# WorkDir: /home/http/mrtg

#  or for NT
# WorkDir: c:\mrtgdata

### Global Defaults

#  to get bits instead of bytes and graphs growing to the right
# Options[_]: growright, bits

WorkDir: /home/mrtg/public_html
Options[_]: growright
Language: big5

######################################################################
#
# System: checkme.adsldns.org
# Description: FreeBSD checkme.adsldns.org 4.9-STABLE FreeBSD 4.9-STABLE
#0: Fri Oct  i386
# Contact: mrtg@localhost
# Location: checkme.adsldns.org
######################################################################
#


### Interface 1 >> Descr: 'vr0' | Name: '' | Ip: '10.0.0.1' | Eth: '' ###

Target[localhost_1]: 1:mrtg@localhost:
```

```
SetEnv[localhost_1]: MRTG_INT_IP="10.0.0.1" MRTG_INT_DESCR="vr0"
MaxBytes[localhost_1]: 130000
Title[localhost_1]: Traffic Analysis for 1 -- checkme.adsldns.org
PageTop[localhost_1]: <H1>Traffic Analysis for 1 --
checkme.adsldns.org</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>checkme.adsldns.org in
checkme.adsldns.org</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>mrtg@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>vr0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ethernetCsmacd (6)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>130.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>10.0.0.1 ()</TD></TR>
 </TABLE>


### Interface 2 >> Descr: 'rl0' | Name: '' | Ip: '192.168.0.1' | Eth: ''
###

Target[localhost_2]: 2:mrtg@localhost:
SetEnv[localhost_2]: MRTG_INT_IP="192.168.0.1" MRTG_INT_DESCR="rl0"
MaxBytes[localhost_2]: 100000
Title[localhost_2]: Traffic Analysis for 2 -- checkme.adsldns.org
PageTop[localhost_2]: <H1>Traffic Analysis for 2 --
checkme.adsldns.org</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>checkme.adsldns.org in
checkme.adsldns.org</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>mrtg@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>rl0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ethernetCsmacd (6)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>100.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>192.168.0.1
(checkme.adsldns.org)</TD></TR>
 </TABLE>
```

```
### Interface 8 >> Descr: 'tun0' | Name: '' | Ip: '210.244.81.117' | Eth:
'' ###

Target[localhost_8]: 8:mrtg@localhost:
SetEnv[localhost_8]: MRTG_INT_IP="210.244.81.117"
MRTG_INT_DESCR="tun0"
MaxBytes[localhost_8]: 130000
Title[localhost_8]: Traffic Analysis for 8 -- checkme.adsldns.org
PageTop[localhost_8]: <H1>Traffic Analysis for 8 --
checkme.adsldns.org</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>checkme.adsldns.org in
checkme.adsldns.org</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>mrtg@localhost</TD></TR>
   <TR><TD>Description:</TD><TD>tun0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ppp (23)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>130.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>float</TD></TR>
 </TABLE>

### CPU
LoadMIBs: /usr/local/share/snmp/mibs/UCD-SNMP-MIB.txt

Target[BoBo.cpu]:ssCpuRawUser.0&ssCpuRawIdle.0:mrtg@localhost
RouterUptime[BoBo.cpu]: mrtg@localhost
MaxBytes[BoBo.cpu]: 100
Title[BoBo.cpu]: CPU LOAD
PageTop[BoBo.cpu]: <H1>User vs Idle CPU usage</H1>
Unscaled[BoBo.cpu]: ymwd
ShortLegend[BoBo.cpu]: %
YLegend[BoBo.cpu]: CPU Utilization
Legend1[BoBo.cpu]: User CPU in % (Load)
Legend2[BoBo.cpu]: Idle CPU in % (Load)
Legend3[BoBo.cpu]:
Legend4[BoBo.cpu]:
LegendI[BoBo.cpu]:  User
```

```
Legend0[BoBo.cpu]:  Idle
Options[BoBo.cpu]: growright,nopercent

### Memory
Target[freemem]: .1.3.6.1.4.1.2021.4.11.0&.1.3.6.1.4.1.2021.4.11.0:mr
tg@localhost
Options[freemem]: nopercent,growright,gauge,noinfo
Title[freemem]: Free Memory
PageTop[freemem]: <H1>Free Memory</H1>
MaxBytes[freemem]: 134152192
kMG[freemem]: k,M,G,T,P,X
YLegend[freemem]: bytes
ShortLegend[freemem]: bytes
Legend1[freemem]:  Free Memory:
Legend0[freemem]:
Legend1[freemem]: Free memory, not including swap, in bytes
```



M_MAIN_IN **同一封信件在 Databse 裡會共用的部份(註 1)**

欄位名稱	資料型態	備　　註
FROM_ID	Varcahr(16)	寄件者帳號 格式：A~Z,0~9 等字元(身份證字號)
FROM_NAME	Varchar(20)	寄件者姓名 格式：中英文字 限制： １使用者的 Name 裡的字元不能有 ”(上引號)或 ,(逗號)，因爲上引號和逗號是用來 Parser 出 Name 的參考符號 ２Name 的前後必須用" " ３E-Mail address 前後必須用< > 例如： “我是酷司拉” <d3939889@some.where>
TO_LIST	TEXT	主要的收件者 list 名單 格式：”name” <address>
CC_LIST	TEXT	次要的收件者 list 名單 格式：同 TO_LIST
BCC_LIST	TEXT	被隱藏的收件者 list 名單 格式：同 TO_LIST 說明：這欄位供系統留記錄用，**使用者看信時是不顯示出來的**
SUBJECT	Varchar (255)	信件主題
DATE	Varchar(50)	信件日期 說明：由 WebMail 產生 格式：沒有統一格式， 格式則由存入欄位的程式決定， 取出欄位資料的程式須用 varchar 而非 datetime 的方式取資料
MESSAGE_ID	Varchar (20)	信件編號 格式：< SysTime:”系統時間” @ GroupNum:”群組號碼”> 系統時間取至毫秒(ms) 例如：<SysTime:20030904014978000@GroupNum:3939889> 說明：WebMail 產生， 到了 MIME 的階段就是 MIME 裡的 Message-Id 這個 Header
BODY	TEXT	信件內文（MIME 內文，除 attach 之外的） **說明：WebMail 須藉由 BODY 裡的， 第一行 Content-Type Header 裡的 boundary 來判斷 Multi-Part 中的各個 Part 的區域**
ATTACH	TEXT	副加檔案（MIME 內文的 attach 部份） 說明：附檔的Parser rule：除了內文外　其它都當附檔 因爲如果從外部信件進來，content-type 會不只 attachment 一種， 都當附檔看待
SIZE	Int	信件大小 格式：容量(Bytes)，附件大小+內文大小 說明：**由 cParser 計算後產生給 jConnector 存入 Database(M_MAIN_IN)**
MISSION_ID	Varchar (3)	任務編號 說明：放在一封 MIME 格式信件最前面的 Header 用來區別存入的這封是一個普通用途”信件”或是其它用途的”檔案”
INDEX : FROM_ID, MESSAGE_ID		

表十一

（註１）爲了系統的記錄功能，
而把一封郵件分成“該保留的部份(M_MAIN_IN)”和“與使用者設定相關(M_RECEIVER)”的二個部份

（註 2）**TO_LIST，CC_LIST，BCC_LIST** 如果是從 **WebMail** 產生，在各別的 **mail address** 間不會有換行符號，
如果是從外部的郵件進來時，**cParser** 負責把多行轉成同一行，再存入 **WebMail**

M_RECEIVER

同一封信件在 Databse 裡會獨立的部份，也就是在 to 或 cc 或 bcc 裡的人都會存一份的，而非像 M_MAIN_IN 共用

欄位名稱	資料型態	備　註
MESSAGE_ID	Varchar (20)	
OWNER_ID	Varchar(10)	收信者 ID 範例：”阿貓” <aaa@bbb.cc> 收信者 ID 爲 **aaa**
OWNER_NAME	Varchar(10)	收信者暱稱 範例：”阿貓” <aaa@bbb.cc> 收信者暱稱爲**阿貓**
SEVER_ID **<001 會放不是 001 的??** **不會，這欄就可拿掉>**	Varchar (20)	**如果 E-Mail 的 Domain 爲*.mil.tw 的話，就是指內部郵件，.mil 前的就是 ServerID** **如果 Domain 不是*.mil.tw，表示此爲外部郵件** **在派送程式須判斷是外部或內部郵件** **另外，.mil 的前面的欄位是指 Server 的編號** 範例：”阿貓” <aaa@0002.mil> Domain 最後一位爲 mail，故爲內部郵件，且該 Server 的 ID 是 0002
COPY	Bit	備份(Flag) 格式：0:不選 1:備份
TRASHCAN	Bit	刪除(Flag) 格式：0:寄件匣 1:垃圾筒
PERUSE	Bit	閱讀信件(flag) 格式：0:未讀 1:已讀
INDEX:MESSAGE_ID + KIND + OWNER_ID + SEVER_ID		

表十二

M_MAIN_OUT 專門負責放要寄出信件的 Table		
欄位名稱	資料型態	備　　註
FROM_MAIL	Varcahr(40)	**寄件者 E-Mail** 格式：”name” <address> 例如： “我是酷司拉” <d3939889@some.where>
TO_LIST	TEXT	主要的收件者 list 名單 格式：”name” <address>
CC_LIST	TEXT	次要的收件者 list 名單 格式：同 TO_LIST
BCC_LIST	TEXT	被隱藏的收件者 list 名單 格式：同 TO_LIST 說明：M_MAIN_OUT 除了有 M_MAIN_IN 的 TO_LIST 和 CC_LIST 外，還多 **BCC 這欄位**
SUBJECT	Varchar (255)	信件主題
DATE	Varchar(50)	信件日期 說明：由 WebMail 產生 格式：沒有統一格式， 格式則由存入欄位的程式決定， 取出欄位資料的程式須用 varchar 而非 datetime 的方式取資料
MESSAGE_ID	Varchar (20)	信件編號 格式：< SysTime:”系統時間” @ GroupNum:”群組號碼”> 系統時間取至毫秒 例如：<SysTime:20030904014978000@GroupNum:3939889>
BODY	TEXT	信件內文（MIME 內文，除 attach 之外的） **說明：WebMail 須產生 Content-Type 和 boundary 在 BODY 裡** **例如：** **Content-Type:multipart/mixed;　boundary="----=_NextPart_000_000B_01C3666E.ED61C830"**
ATTACH	TEXT	副加檔案（MIME 內文的 attach 部份） 說明：附檔的Parser rule：除了內文外　其它都當附檔 因爲如果從外部信件進來，content-type 會不只 attachment 一種，都當附檔看待
COPY	Bit	備份(Flag) 格式：因爲是剛產生，未送出的郵件，故一定是 0
TRASHCAN	Bit	刪除(Flag) 格式：因爲是剛產生，未送出的郵件，故一定是 0
PERUSE	Bit	閱讀信件(flag) 格式：因爲是剛產生，未送出的郵件，故一定是 0
INDEX : FROM_ID, MESSAGE_ID		

表十三

M_SERVER

1. 此 Table 只有 WebMail 會去參考

2. WebMail 在產生郵件時，須根據所有的 E-Mail 的 Domain，把所有用到的 SERVER_ID 附到 Mission-Part 裡

3. 派送程式須根據 TO, BCC, CC 三個 Header，
去看要送到同 FreeBSD Gateway 的 DB，或ＭＳ，或ＸＭＬ

欄位名稱	資料型態	備　　註
SERVER_ID	Varchar (3)	伺服器編號
SERVER_DNS	Varchar (20)	該機器的 Domain Name
SERVER_NAME	Varchar (20)	該機器的電腦名稱 說明：FreeBSD 的電腦名稱命名規則：機器名.網域名 例如：Machine1.Net
SERVICE_ID	Varchar (2)	提供服務編號 說明：不同的電腦提供的 Service 會不同， 且一台電腦也可能只有一個 Service 或是多個 Service 格式：如果有三個 Service → “00, 01, 06” (假設值，無義)
INDEX: SEVER_ID		

表十四

參考資料

[1]. 專業 jsp 程式設計 professional JSP 蘇東士譯。
[2]. Lex & yacc 中譯本
[3]. Java Server Page 程式設計實務 / 楊洸, 沈建男作
[4]. Linux 9.X 指令應用 / 李蔚澤, 胡銘珍作
[5]. LINUX & UNIX 指令與辭彙篇 / 和碩 Linux 研究小組著
[6]. Linux 平台開發工具應用手冊 / Rafeeq Ur Rehman, Christopher Paul 著; 江俊龍譯
[7]. http://www.jsptw.com/Old/index_old.html
[8]. http://www.jsptw.com/
[9]. http://search.java.sun.com/search/java/?qt=javamail
[10]. http://forum.vclxx.org/default.asp
[11]. http://bima.astro.umd.edu/checker/node23.html
[12]. http://www.injunea.demon.co.uk/pages/page203.htm#6-4
[13]. http://www.bo.infn.it/alice/alice-doc/mll-doc/impgde/node94.html
[14]. http://vertigo.hsrl.rutgers.edu/ug/shell_help.html
[15]. http://www.starlink.rl.ac.uk/star/docs/sc4.htx/sc4.html#xref_
[16]. http://programmer.eforum2000.net/pc2020v5/TitleList.asp
[17]. http://java.fromtw.com/doc/java1.html#InstanceSample
[18]. http://cn.sun.com/developers/onlineTraining/JavaMail/