Computer Network 2011

Homework 1

B97501046　資訊工程學系　三年級　李卿澄

檔案結構

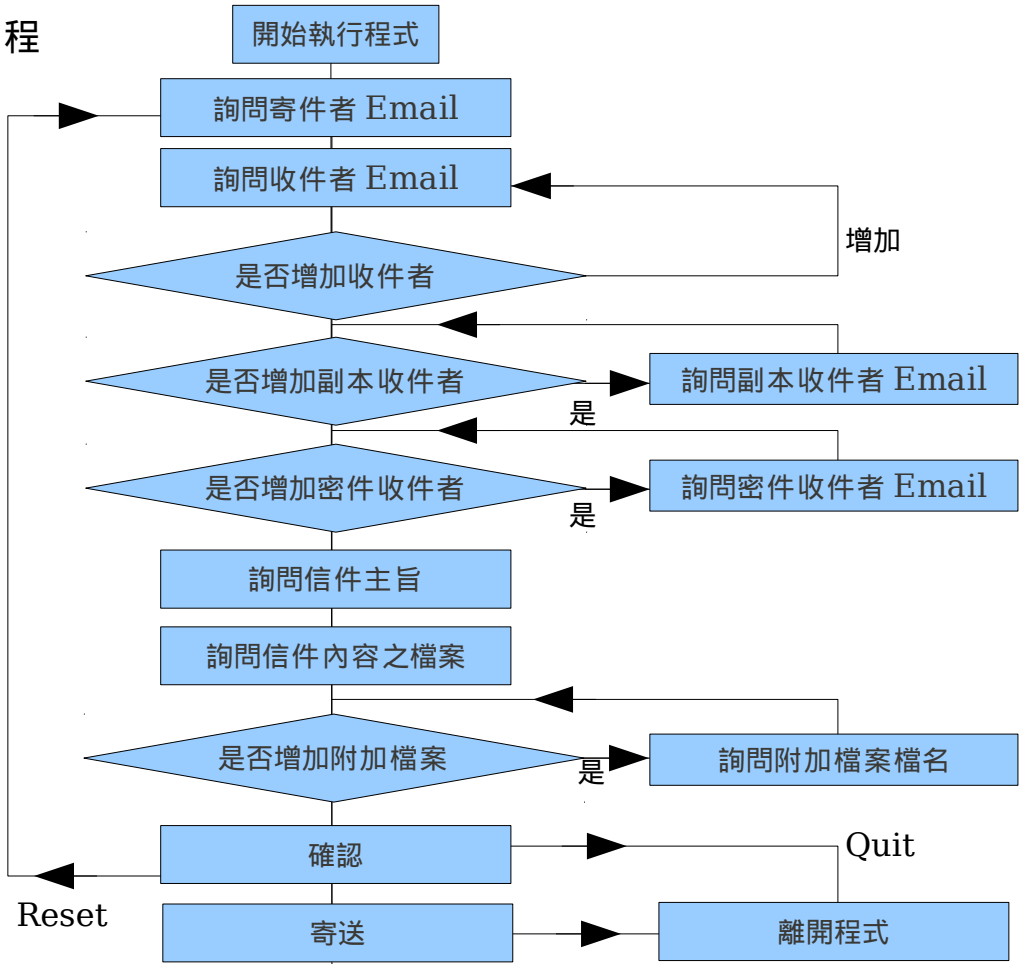
b97501046.tar.gz 內包含了：

b97501046/	資料夾
– b97501046.c	*Simple Mail* 主程式
– SMCF.h	*Simple Mail Client Function Header* 檔
– SMCF.c	*SMCF.h* 之實作
– gui.c	*Gui* 的主程式
– SMCF_GUI.h	*Gui* 所用函式之 *header* 檔
– SMCF_GUI.c	*SMCF_GUI.h* 之實作
– Makefile	*Makefile*
– Report.pdf	*Report*（本文件）

執行程式

1. 編譯執行檔
   $ make
   會編譯出名為 SimpleMail 的可執行檔。
2. 執行
   $ ./SimpleMail *hostname port*
   （$ make run 可直接執行 $ ./SimpleMail cnmail.csie.org 25）

執行流程



```
檔案(F)  編輯(E)  檢視(V)  終端機(T)  求助(H)
Simple Mail Client (CN2011HW#1)
Please input Sender's email:b97501046@cnmail.csie.org
Please input Reciver's email:qc.linux@gmail.com
Add reciver? (y/n)n
Add carbon copy? (y/n)y
Please input Cc's email:b97501046@gmail.com
Add carbon copy? (y/n)n
Add bline carbon copy? (y/n)n
Please input Subject:This is a test
Please input Mailbody file:mail
Add attachment? (y/n)y
Please input filename:homework1.pdf
Add attachment? (y/n)n

Your mail:
From:
        b97501046@cnmail.csie.org
To:
        qc.linux@gmail.com
Cc:
        b97501046@gmail.com
Bcc:
Subject:
        This is a test
Mailbody:
        mail
Attachment:
        homework1.pdf

[S]end it, [R]eset or [Q]uit this program? (s/r/q):s
```

在程式執行一開始，會先詢問寄件者要顯示在信件上的寄件人 Email。

接著詢問收件人的 Email。

然後尋問要不要增加副本收件人、密件副本。

接著請使用者輸入信件的標題

使用者要寄送的信件本文需要寫在一個檔案裡面，讓程式讀入後寄送。因此這裡要輸入信件本文的檔案名稱

輸入完信件本文檔案名稱後，會詢問是否要夾帶附加檔案，並請輸入檔名。

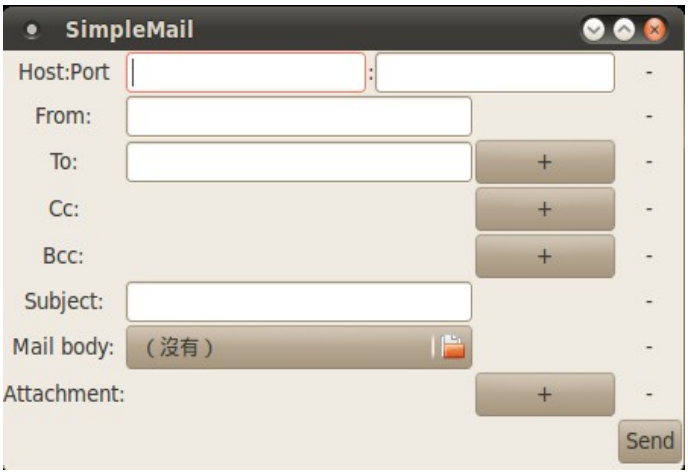
所有的輸入都完成後，會出現確認訊息，可以輸入 s 以寄送信件，或輸入 q 離開程式，抑或是輸入 r 重新設定。

輸入 s 後就會開始寄送郵件的過程。

圖形使用者介面（GUI）

為了使操作起來感覺更直覺，所以就嘗試著製作了使用者介面的程式。

整個 GUI 是以 Gtk+撰寫，欲產生 GUI 執行檔只要下指令：
$ make gui
就會產生名為 SimpleMailGui 之可執行檔（於 Linux Gnome 桌面環境下可執行）。



按「＋」鈕可以新增欄位，按「Send」會進入寄信動作，若信件正確寄出，右方所有的「-」會變成「O」，若有「X」，則可能是該步驟出了錯誤，可以用指令列開啟此 GUI 執行檔並觀察終端機上的輸出來檢測錯誤所在。

程式撰寫

Socket

連線部份使用了 *socket* 建立一個 socket，並得到一個 socket 的 fd；使用 *gethostbyname* 取得 host 的位置資訊，並利用 *inet_ntop* 及 *inet_pton* 處理之，最後使用 *connect* 連結 host，連接 host 後便進行 *read* 與 *write* 的操作。連線部份的程式碼置於 SMCF.c 的 *connectToHost* 函式中，連線成功便將 fd 回傳以利其他函式操作。

函式切割

最初看到這份作業的時候我是想要使用 python 來寫（socket 相關的函數都包好好的），後來因為規定的關係，最終是使用了 C。一開始寫著寫著，一下要跟使用者 *scanf* 東西，一下又要

read、*write*，不知不覺就變成一團炒麵了，整個 main 就長得非常難看，所以就依照流程，把流程切割成數個函式，只在主程式中留下流程以及部份處理錯誤的程式，在另外的檔案中才實做流程的內容。只要整個寄送信件的流程不改變，我們可以不必修改主程式，只要修改實作方式就可以，也可以有多種不同的實作方式。

各個連上 host 的函數則是以 socket 的 fd 作為共同的參數，分別對 fd 進行操作，因此能夠拆開來實作。

因為把流程切開的關係，在後來試作 GUI 介面時只要照著流程呼叫函式就可以了。在 GUI 中，我把所會用到 GtkWidget 都包在一個 struct 裡面，其實已經有點像是 C++的物件了，這樣也讓 *g_signal_connect* 以事件呼叫自訂函數時只要將 struct 指標作為參數傳遞就可以操作整個 struct，不必製造奇怪的全域變數，讓程式碼好看些，也方便些，只是離真正的物件還有一段距離就是。