建置環境與使用說明:

作業系統: Ubuntu 18.04.4 LTS

使用IDE: Eclipse

使用編譯器: g++

使用語言: c++

使用方式:

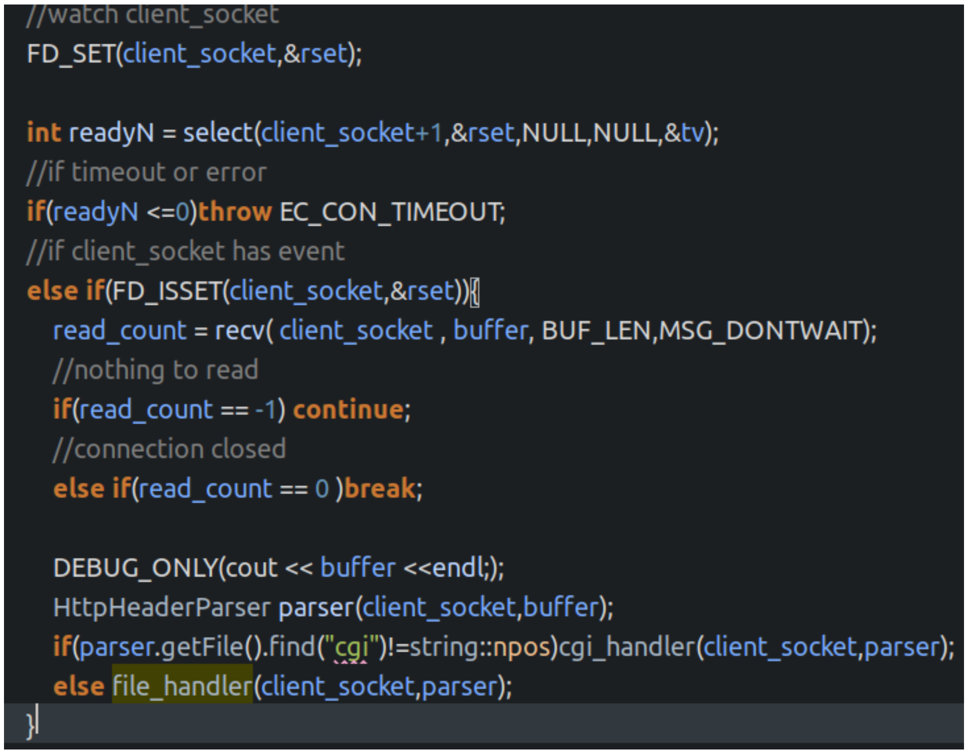
1.開啟終端機

2.執行make

3.執行NSHW1

重要程式碼說明:

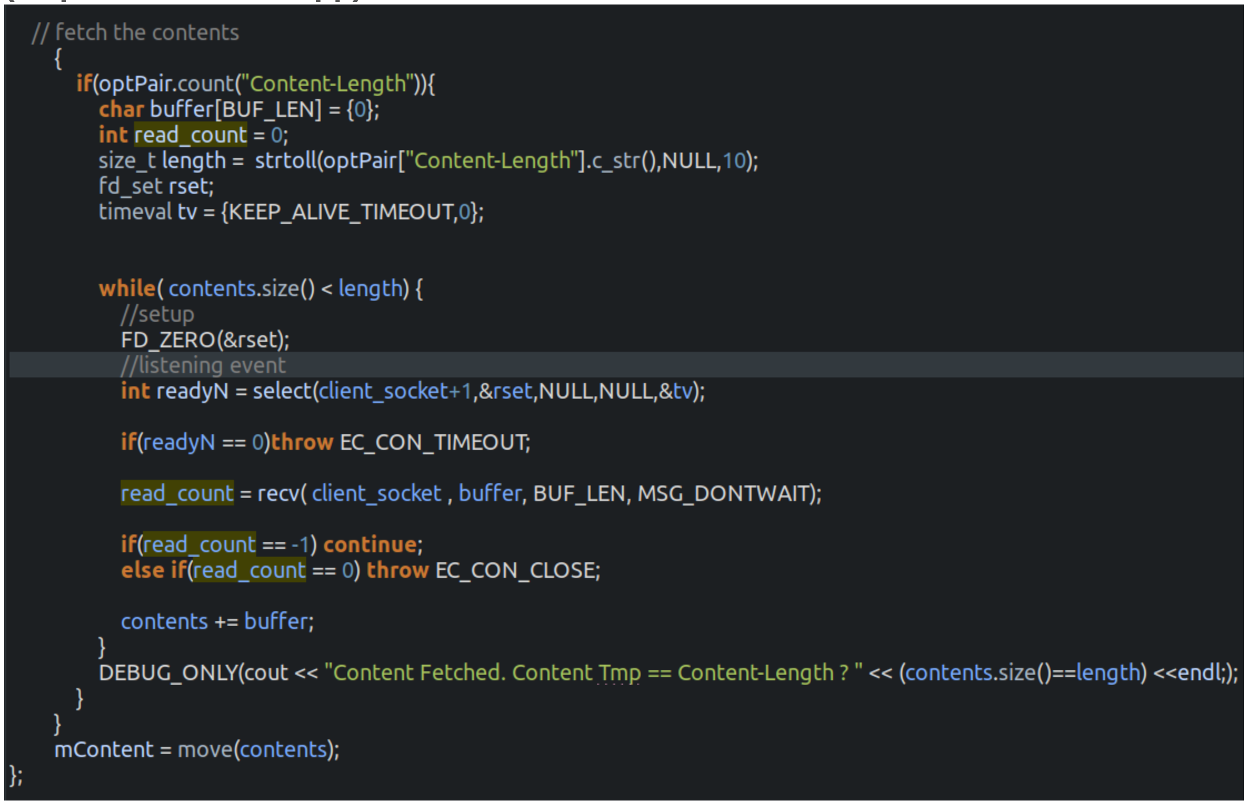
(NSHW1.cpp)



```
//watch client_socket
FD_SET(client_socket,&rset);

int readyN = select(client_socket+1,&rset,NULL,NULL,&tv);
//if timeout or error
if(readyN <=0)throw EC_CON_TIMEOUT;
//if client_socket has event
else if(FD_ISSET(client_socket,&rset)){
   read_count = recv( client_socket , buffer, BUF_LEN,MSG_DONTWAIT);
   //nothing to read
   if(read_count == -1) continue;
   //connection closed
   else if(read_count == 0 )break;

   DEBUG_ONLY(cout << buffer <<endl;);
   HttpHeaderParser parser(client_socket,buffer);
   if(parser.getFile().find("cgi")!=string::npos)cgi_handler(client_socket,parser);
   else file_handler(client_socket,parser);
}
```

　這一整段負責處理連入的瀏覽器客戶的請求，使用Non-Blocking的recv/send搭配select來處理資料。當recv有資料可以進行接收時，表示有可能是Http Header或者Socket被關閉，如果是Http Header，則將資料交給HttpHeaderParser來處理/分析Header內容，並且把Content資料接收完全。

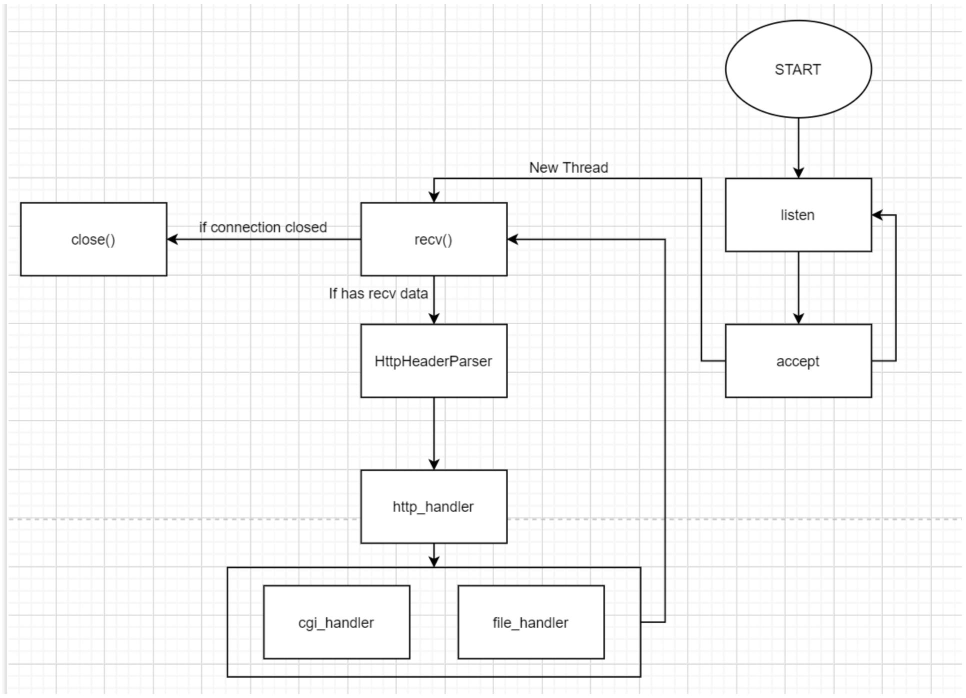
(HttpHeaderParser.cpp)



```
  // fetch the contents
    {
      if(optPair.count("Content-Length")){
        char buffer[BUF_LEN] = {0};
        int read_count = 0;
        size_t length =  strtoll(optPair["Content-Length"].c_str(),NULL,10);
        fd_set rset;
        timeval tv = {KEEP_ALIVE_TIMEOUT,0};


        while( contents.size() < length) {
          //setup
          FD_ZERO(&rset);
          //listening event
          int readyN = select(client_socket+1,&rset,NULL,NULL,&tv);

          if(readyN == 0)throw EC_CON_TIMEOUT;

          read_count = recv( client_socket , buffer, BUF_LEN, MSG_DONTWAIT);

          if(read_count == -1) continue;
          else if(read_count == 0) throw EC_CON_CLOSE;

          contents += buffer;
        }
        DEBUG_ONLY(cout << "Content Fetched. Content Tmp == Content-Length ? " << (contents.size()==length) <<endl;);
      }
    }
    mContent = move(contents);
};
```

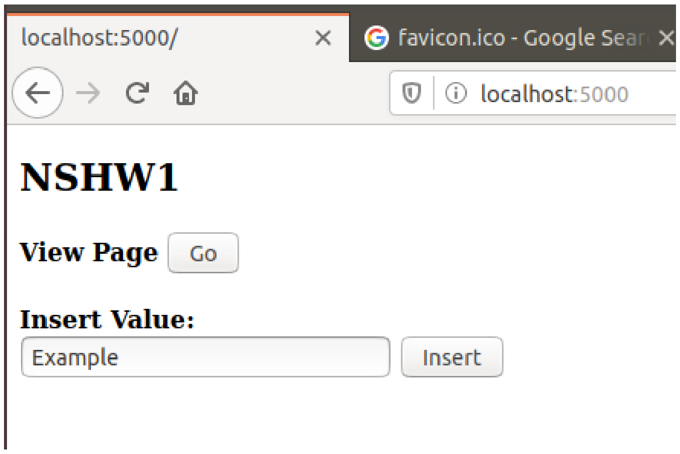
　這一段是解析完Http Header的Content-Length後發現還有更多Content需要接收時，做的處理:把剩下的資料接收完成。

設計架構與功能說明:


START
New Thread
listen
close()
if connection closed
recv()
If has recv data
HttpHeaderParser
accept
http_handler
cgi_handler
file_handler

成果截圖:

(index.html)


localhost:5000/

favicon.ico - Google Sear

localhost:5000

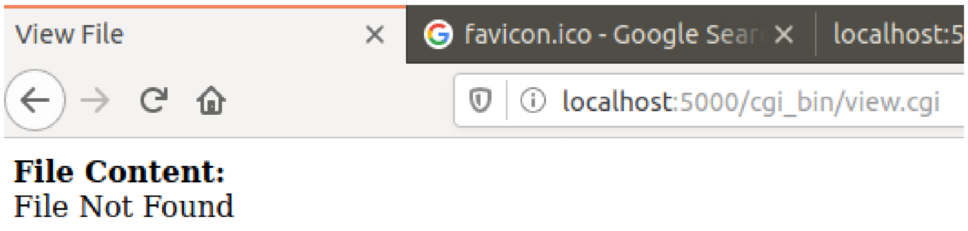
NSHW1

View Page Go

Insert Value:

Example Insert

(go to view page)


View File

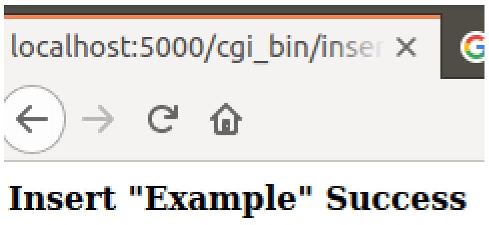
favicon.ico - Google Sear

localhost:5

localhost:5000/cgi_bin/view.cgi

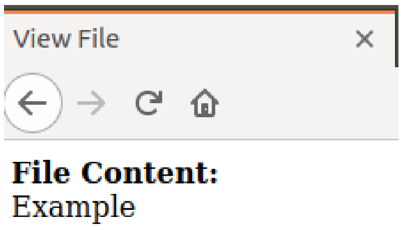
File Content:
File Not Found

(insert example)


localhost:5000/cgi_bin/inser

Insert "Example" Success

(go to view page)


View File

File Content:
Example

困難與心得:

原本想要直接做keep-alive連線，可是被send的buffering(或是blocking)搞了很久，一直搞不定。最後只好先寫closed連線，把整個架構先弄出來。弄完之後再專心用keep-alive才發現可以用non-blocking來達成，最後把keep-alive修一修才弄好，搞了整整兩天。其實還蠻好玩的，看到瀏覽器裡面有內容的剎那，爽。