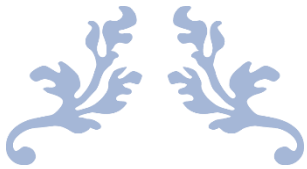
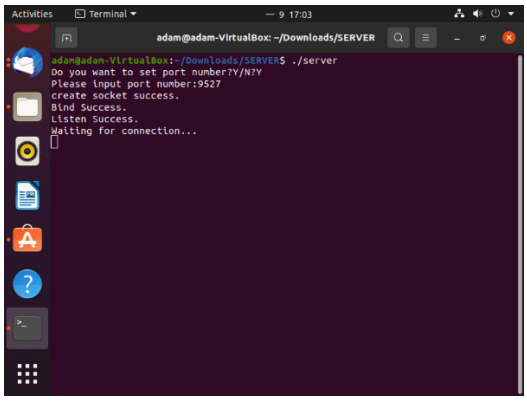


# 計算機網路作業報告

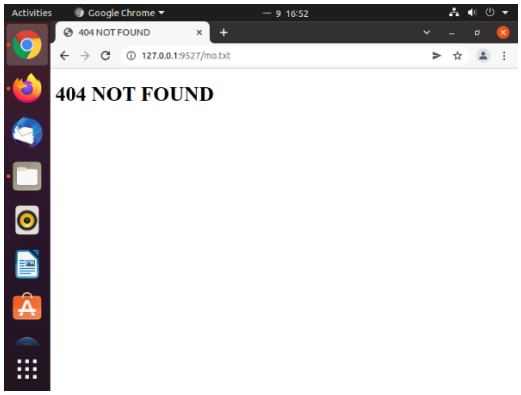
B0829039 王語堂

2022 年 1 月 9 日

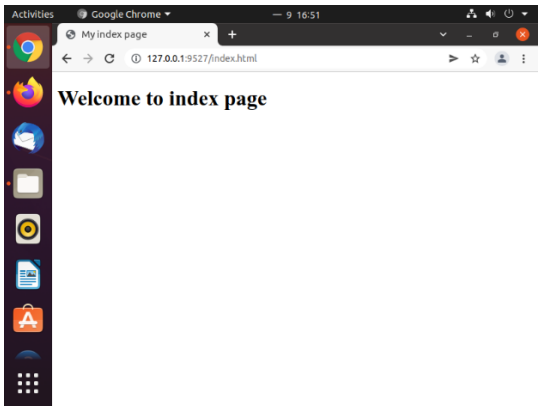
操作方式:由於本次的系統需使用 fork()函數，因此請在 linux 環境下運行，開發時使用的是 ubuntu，為了確保 ubuntu 能夠正常撥放影片，請先確保 ubuntu 系統有下載 MPEG-4 AAC decoder 以及 H.264 decoder.下載方式可參考該連結，並確保使用的是最新版的 Chrome 瀏覽器，以及系統中有 gcc 及 g++之編譯器。
首先須用 terminal cd 到放置 server.cpp 的資料夾，並以 g++ server.cpp -o server 或 make 對 server.cpp 進行編譯，並得到 server 的執行檔，之後便可以用./server 進行執行。
可選擇 port，這次使用的是 9527，IP 則為 127.0.0.1。



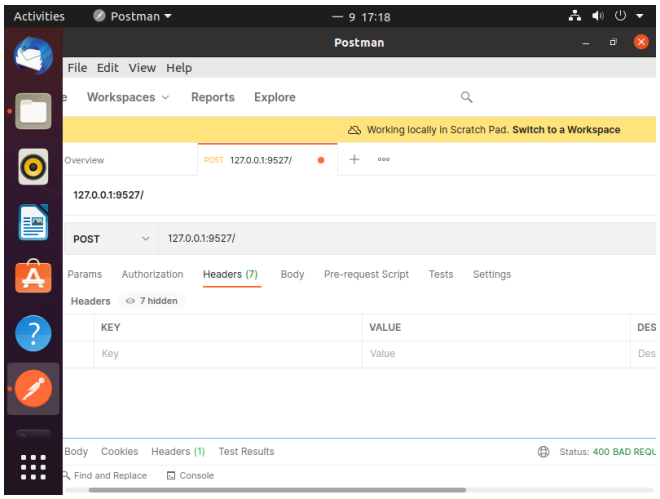

若選擇了不存在的檔案會顯示 404。



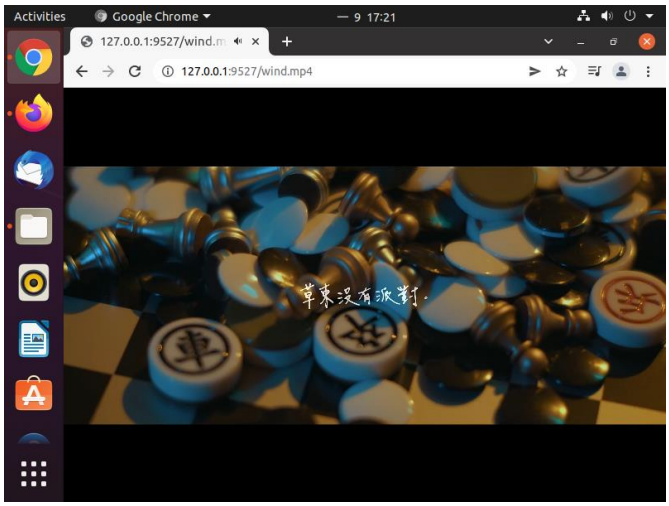

而要是存在的 html 則會正常顯示。



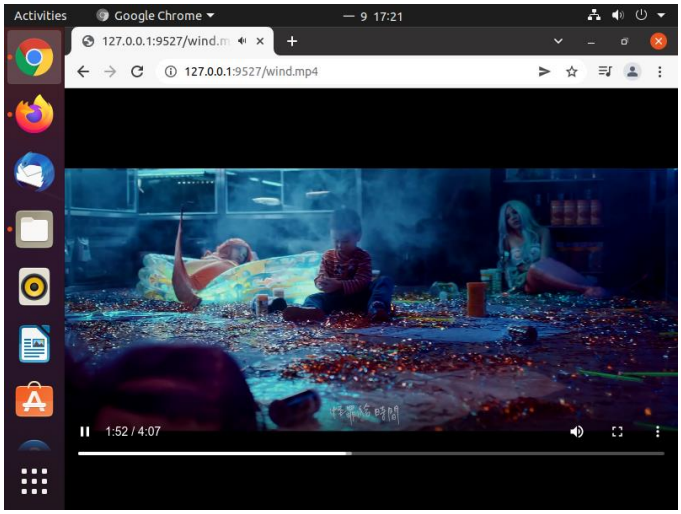

若使用非正確格式則會回傳 400。

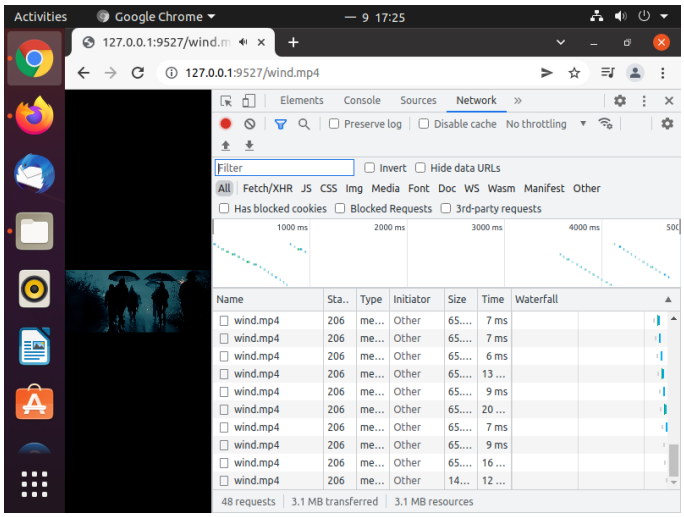



可撥放影片。

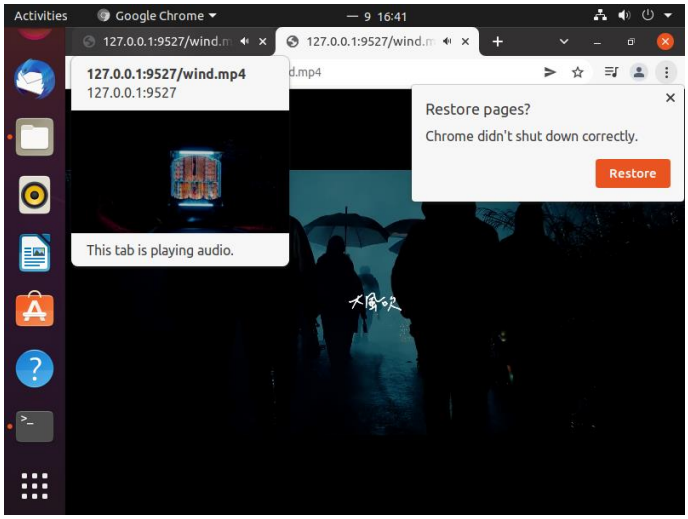



影片可快轉(HTTP range requests)。






利用 Fork()可同時開啟兩個 client 並各自播放影片。




程式說明:

在本程式中 NOTFOND(int socketfd)及 BADREQUEST(int socketfd)負責傳輸對應 404 及 400 這兩種情況的 Header 及 html 檔給瀏覽器。並由 request_parser(int fd,char *sizeRequest,Objective& Obj)利用 strtok 來拆解 buffer 中所儲存的 request，以此取得檔案路徑，method，和 http 版本等情報，並以 strstr()來取得 range，藉此判斷是否為 range request，request_handler(int fd)則會去利用 request_parser 分離出的資訊，來判斷檔案是否存在，request 的格式是否正確，以及應該給予怎樣的 response，並利用 fseek 來獲取檔案長度以處理 request 需要的 content length，計算要傳送的 response 次數與起始位置，透過斷點續傳方式傳送影片，而 main()則負責創建 socket 並進行監聽，並利用 fork 出的子程序來執行 request_handler，藉此達成支援多個 client 同時連線的效果。

```
#include <iostream>
#include <stdlib.h>
#include <cstring>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

using namespace std;
class Objective{
public:
    bool  Range;//儲存是否為 range request
    int   startPoint;//檔案讀取的開始點
    char  requestMethod[32];//儲存該 request 的 method
    char  requestFilename[1024];//儲存該 request 要求的 filename
    char  requestVersion[32];//儲存該 request 的 version
};
int BADREQUEST(int socketfd)
{
    const char *sendbuf = "HTTP/1.0 400 BAD REQUEST\nContent-Type:
text/html\n\n<!DOCTYPE html><html><head><title>400 BAD
REQUEST</title></head><body><h1>400 BAD REQUEST</h1></body></html>";
    //cout << "400 BAD REQUEST" << endl;
    //----------------------
    // Send an initial buffer
    send(socketfd,sendbuf,(int)strlen(sendbuf),0);
    close(socketfd);
    exit(0);

}
int NOTFOND(int socketfd)
{
    const char *sendbuf = "HTTP/1.0 404 NOT FOUND\nContent-Type:
text/html\n\n<!DOCTYPE html><html><head><title>404 NOT
FOUND</title></head><body><h1>404 NOT FOUND</h1></body></html>";
```

```
    //cout << "404 NOT FOUND" << endl;
    //----------------------
    // Send an initial buffer
    write(socketfd,sendbuf,(int)strlen(sendbuf));
    close(socketfd);
    exit(0);
}
void request_parser(int fd,char *sizeRequest,Objective& Obj)
{
    string reqMethod = "", target = "",version = "";
    char* range;
    char *request;
    if((range = strstr(sizeRequest, "Range: bytes=")) == NULL){
        cout << "no range in this request." << endl;//儲存該 request 沒有
range 的事實
        Obj.Range = false;
        Obj.startPoint = 0;
    }
    else{
        cout << "find out range in this request." << endl;
        range += 13; //將 range 指向第一個數字
        range = strtok(range, "-");
        Obj.Range = true;
        string start = range;//儲存該 request 有 range 的事實
        Obj.startPoint = stoi(range);
        cout << "start at: " << Obj.startPoint << endl;
    }
    request = strtok(sizeRequest, " ");//擷取使用的 method
    reqMethod += request;
    //cout << reqMethod << endl;
    request = strtok(NULL, " ");    //擷取檔名
    if(request == NULL){    //沒有要求，格式錯誤
        BADREQUEST(fd);
    }
    request = request + 1; // 跳過第一個斜槓，取得路徑與檔名
    target += request;
    //cout << target << endl;
    request = strtok(NULL, " \r\n");
```

```
    if(request == NULL){//沒有版本，格式錯誤
        BADREQUEST(fd);
    }
    version += request;
    //cout << version << endl;
    strcpy(Obj.requestMethod, reqMethod.c_str());
    strcpy(Obj.requestFilename, target.c_str());
    strcpy(Obj.requestVersion, version.c_str());
}
void request_handler(int fd)
{
    Objective obj;
    char buffer[4096]={0};
    while(1)
    {
        cout << "a connection was found.\n";//收到 request
        memset(buffer, 0, sizeof(buffer));//清空 buffer
        read(fd, buffer, sizeof(buffer));//將 request 儲存於 buffer
        cout << buffer << endl;
        request_parser(fd, buffer, obj);
        if(strcmp(obj.requestMethod, "GET")){//非 GET，格式錯誤，回傳 400
            cout << "wrong method" << endl;
            BADREQUEST(fd);
        }
        if(strcmp(obj.requestVersion,"HTTP/1.1"))//版本錯誤，回傳 400
        {
            cout << "wrong version" << endl;
            BADREQUEST(fd);
        }
        char contentType[30];
        if(strstr(obj.requestFilename, ".mp4") != NULL){//要求檔案類型為
mp4，設定 content type 為 video/mp4
            cout << "A Video request" << endl;
            strcpy(contentType, "video/mp4");
        }
        else if(strstr(obj.requestFilename, ".html") != NULL){//要求檔案
類型為 html，設定 content type 為 text/html
            cout << "A Text request" << endl;
```

```
            strcpy(contentType, "text/html");
        }
        char Resource[4096]={0};
        if(strcmp(contentType, "video/mp4") == 0)//以是否為影片進行不同的
處理
        {
            FILE *reader = fopen(obj.requestFilename, "rb+");//讀取影片檔
案
            if(reader == NULL){
                cout << obj.requestFilename << ":does not exist" <<
endl;
                NOTFOND(fd);//檔案不存在，回傳 404
                exit(1);
            }
            else
            {
                //檔案存在，回傳 200
                cout << obj.requestFilename << ":does  exist" << endl;
                // 以 fseek 獲取文件大小好方便回傳 Content Length
                fseek(reader, 0L, SEEK_END);
                int fileLength = ftell(reader);
                fseek(reader, 0, SEEK_SET); //將讀寫位置設為檔案的開頭
                if(!obj.Range){  // 第一次先送 header 跟 range
                    cout << "200 OK" << endl;
                    snprintf(Resource, 4096, "HTTP/1.1 200 OK\nContent-
Type: %s\nContent-Length: %d\nAccept-Ranges: bytes\n\n", contentType,
fileLength);
                    cout << Resource << endl;
                    send(fd, Resource, strlen(Resource), MSG_NOSIGNAL);
                    cout << "send header success" << endl;
                }
                else//以斷點續傳方式傳送後續影片，每次傳送部分片段
                {
                    long contentLeft = fileLength - obj.startPoint;//檔案
剩餘大小
                    int fragSize = 65536;//response 的檔案大小(64KB)，
64*1024=65536
```

```
                    int fragNum = contentLeft / fragSize;//需要傳送的
Response 次數
                    if(contentLeft % fragSize != 0)
                    {
                        fragNum++;//多增加一次 response 以處理餘數
                    }
                    for(int i = 0; i < fragNum; i++)
                    {
                        if(i+1 == fragNum)
                        {
                            fragSize = fileLength - obj.startPoint; //將
response 大小重設以處理無法整除的剩餘部分
                        }
                        snprintf(Resource, 4096, "HTTP/1.1 206 Partial
Content\nContent-Type: %s\nContent-Length: %d\nContent-Range: bytes %d-
%d/%d\nAccept-Ranges: bytes\n\n",contentType, fragSize, obj.startPoint,
obj.startPoint+fragSize-1, fileLength);
                        send(fd, Resource, strlen(Resource),
MSG_NOSIGNAL);
                        char sizeBuffer[fragSize]={0};
                        fseek(reader, obj.startPoint, SEEK_SET);//將讀寫
位置設為瀏覽器要求的位置
                        if(fread(sizeBuffer, 1, fragSize, reader) == 0){
                            cout << "read video file error." << endl;
                            exit(1);
                        }
                        else
                        {
                            send(fd, sizeBuffer, fragSize, MSG_NOSIGNAL);
                            obj.startPoint += fragSize;//在 response 後移動
下次要傳送的讀寫位置
                            memset(sizeBuffer, 0, fragSize);//在每次寫入後
清空 buffer
                        }
                    }
                    cout << "Transfer video done" << endl;
                }
            }
```

```
        }
        else
        {
            FILE *reader = fopen(obj.requestFilename, "r");
            if(reader == NULL){
                cout << obj.requestFilename << ":does not exist" <<
endl;
                NOTFOND(fd);//檔案不存在，回傳 404
                exit(1);
            }
            else
            {
                cout << obj.requestFilename << ":does  exist" << endl;
                fseek(reader, 0L, SEEK_END);
                int fileLength = ftell(reader);//利用 fseek 取得檔案長度
                fseek(reader, 0, SEEK_SET);    //將讀寫位置設為檔案的開頭
                //檔案存在，回傳 200
                snprintf(Resource, 4096, "HTTP/1.1 200 OK\nContent-Type:
%s\nContent-Length: %d\nAccept-Ranges: bytes\n\n", contentType,
fileLength);
                send(fd, Resource, strlen(Resource), MSG_NOSIGNAL);
                char siezBuffer[fileLength]={0};
                if(fread(siezBuffer, 1, fileLength, reader) == 0){
                    cout << "error.\n";
                    exit(1);
                }
                send(fd, siezBuffer, fileLength, 0);
            }
        }
    }
}
int main()
{
    char YN;
    int PORT_NUM;
    //set default port number=80
    while(1)
    {
```

```
        cout << "Do you want to set port number?Y/N?";
        cin >> YN;
        if(YN=='Y')
        {
            cout << "Please input port number:";
            cin >> PORT_NUM;
            break;
        }
        else if(YN=='N')
        {
            PORT_NUM = 80;
            break;
        }
        else
        {
            continue;
        }
    }
    int i, pid, listenfd, socketfd, sRecv;
    socklen_t length;
    static struct sockaddr_in cli_addr;
    static struct sockaddr_in serv_addr;
    if ((listenfd=socket(AF_INET, SOCK_STREAM,0))<0)
    {
        cout<<"Fail to create a socket.\n";
        exit(1);
    }
    else
    {
        cout << "create socket success.\n";
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;//設定 IP
    serv_addr.sin_port = htons(PORT_NUM);//設定 port
    if (bind(listenfd, (struct sockaddr
*)&serv_addr,sizeof(serv_addr))<0)
    {
```

```
        cout << "Bind Fail.\n";
        exit(1);
    }
    else
    {
        cout << "Bind Success.\n";
    }
    if (listen(listenfd,64)<0)
    {
        cout << "Listen Fail.\n";
        exit(1);
    }
    else
    {
        cout << "Listen Success.\n";
    }

    while(1) {
        cout << "Waiting for connection... "<<endl;
        length = sizeof(cli_addr);
        /* 等待客戶端連線 */
        if ((socketfd = accept(listenfd, (struct sockaddr *)&cli_addr,
&length))<0)
        {
            cout << "Accept Fail.\n";
            exit(1);
        }
        /*連線成功*/
        pid_t id = fork();
        if(id == -1){
            cout << "Fork Error.\n";//fork 失敗，回傳-1，結束該程序
            return -1;
        }
        if(id == 0){   // 子程序，負責處理 request
            close(listenfd);
            request_handler(socketfd);
            exit(0);//處理完畢，結束子程序
        }
```

```
        else if(id > 0){
            close(socketfd);
        }
    }
    return 0;
}
```

參考資料:

https://www.twblogs.net/a/5c56d056bd9eee06ef3686e4

https://snsd0805.github.io/jekyll/update/2019/05/27/%E7%AD%86%E8%A8%98-Linux%E7%92%B0%E5%A2%83%E7%94%A8c++%E5%BB%BA%E7%AB%8BSocket%E9%80%A3%E7%B7%9A.html

https://fred-zone.blogspot.com/2007/09/http-web-server.html

https://dangerlover9403.pixnet.net/blog/post/212391408-%5B%E6%95%99%E5%AD%B8%5Dc++-socket%E8%B3%87%E6%96%99%E6%95%B4%E7%90%86

https://blog.csdn.net/weixin_44720401/article/details/118671332