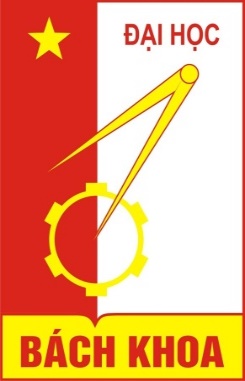
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN ĐIỆN TỬ - VIỄN THÔNG**

**----------**



**ĐA PHƯƠNG TIỆN NÂNG CAO**

**Truyền thông đa phương tiện thời gian thực sử dụng VLC**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **: TS. Phạm Văn Tiến** |
| **Sinh viên thực hiện** | **: Lê Trung Kiên** |
| **MSSV** | **: 20142401** |
| **Lớp** | **: KSTN - ĐTVT K59** |

**Hà Nội, ngày 22 tháng 11 năm 2018**

# Các bước build VLC

## Build VLC từ source code

Repo chính thức của VLC : https://github.com/videolan/vlc

Git clone source code VLC về máy:

git clone <https://github.com/videolan/vlc.git>

Các bước cài đặt VLC :

* Cài đặt các dependency :

sudo apt-get install -y libdbus-1-dev lua5.1 liblua5.1-dev libmad0-dev libavcodec-dev libavformat-dev libswscale-dev libpostproc-dev liba52-dev libxcb1-dev libxcb-shm0-dev libxcb-xv0-dev libx11-xcb-dev libgl1-mesa-dev libqt4-dev libgcrypt11-dev

* Build sử dụng make :

./bootstrap

./configure

make

sudo make install

Lý thuyết về Socket và ứng dụng làm một app chat đơn giản

Khái niệm:

“Socket là một cổng logic mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên một máy tính khác trên Internet. Chương trình mạng có thể sử dụng nhiều Socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.”

Ở đây ta hiểu Socket trong Winsock như là một “phương tiện” để ứng dụng mạng có thể trao đổi dữ liệu. Nghĩa là 1 Server thì sẽ cần một Socket để lắng nghe, chờ đợi các kết nối từ client và Client thì phải cần có một Socket để kết nối tới Sever.

Hàm khởi tạo socket

|  |
| --- |
| 1. SOCKET socket ( 2. **int** af, 3. **int** type, 4. **int** protocol 5. **);** |

Ví dụ:

|  |
| --- |
| SOCKET s = socket(AF\_INET,SOCK\_STREAM,IPPROTO\_IP)**;** |

Trong đó:

af: Là một con số ID để quyết định Socket của chúng ta sử dụng giao thức (protocol) để kết nối.

* AF\_INET : TCP/IP (Phổ biến nhất hiện nay -> dùng địa chỉ IP để truyền dữ liệu)
* AF\_NETBIOS: NetBIOS (Giao thức dùng tên máy để truyền dữ liệu)
* AF\_APPLETALK: AppleTalk
* AF\_ATM: ATM

…

Blind

|  |
| --- |
| 1. **int** bind( 2. SOCKET                     s, 3. **const** **struct** sockaddr FAR\* name, 4. **int**                        namelen 5. **);** |

Tác dụng dụng của BIND là sẽ giúp cho SOCKET của SERVER biết rằng nó sẽ chờ đợi kết nối và nhận dữ liệu trên IP nào và PORT napf

Một số thông tin về các port

* 0 -1023: Là những PORT đã được sử dụng bởi các dịch vụ như WEB, FTP, …
* 1024-49151: Là PORT dành cho SERVER lắng nghe. SERVER nên chọn trong khoảng này.
* 49152-65535: Là PORT khởi tạo ngẫu nhiên dành cho CLIENT kết nối tới Server.

Hàm Bind gồm có 3 thông số:

* SOCKET s: Socket được thiết lập
* sockaddr name: Cấu trúc ADDR bao gồm địa chỉ IP và PORT
* int namelen: Kích thước của cấu trúc sockaddr

Listen:

|  |
| --- |
| 1. **int** listen( 2. SOCKET s, 3. **int**    backlog 4. )**;** |

Kể từ khi gọi hàm này thì SERVER sẽ bắt đầu lắng nghe kết nối của mình.

Hàm LISTEN gồm có 2 thông số:

* SOCKET s: Socket đã được thiết lập IP và PORT.
* int backlog: Số kết nối cho phép chờ trong hàng đợi khi Server chưa chấp nhận kết nối. (vì đôi lúc có thể có tới 2 hay 3 client kết nối tới cùng 1 lúc). Giá trị tốt nhất là khoảng từ 5 – 10.

Connect:

|  |
| --- |
| 1. **int** connect( 2. SOCKET s, 3. **struct** sockaddr \*serv\_addr, 4. **int** addrlen )**;** |

Hàm được gọi từ CLIENT nếu nó muốn kết nối tới SERVER

* SOCKET s: Socket đã được khởi tạo.
* sockaddr \*serv\_addr: IP và PORT của Server.
* int addrlen: Sizeof của cấu trúc sockaddr.

Accept

|  |
| --- |
| 1. SOCKET accept( 2. SOCKET  s, 3. **struct** sockaddr FAR\*  addr, 4. **int** FAR\*  addrlen 5. **);** |

Khi Client kết nối tới Server. Nó phải chờ Server chấp nhận kết nối (nếu ở giao thức TCP) bằng hàm accept trên.

Hàm ACCEPT gồm có 2 thông số:

* SOCKET s: Socket lắng nghe của SERVER.
* sockaddr addr: Là cấu trúc sockaddr lưu địa chỉ IP và PORT của CLIENT kết nối tới SERVER.
* int addrlen: Kích thước cấu trúc địa chỉ IP này.

Hàm ACCPET trả về 1 SOCKET mới Socket mới được tạo này đại diện cho 1 Connection (kết nối) mới giữa Server và Client. Sau khi đã truyền dữ liệu thì ta phải đóng SOCKET này lại bằng hàm close như closesocket (connect);

Send/Receive

|  |
| --- |
| 1. **int** recv( 2. SOCKET s, 3. **char** FAR\* buf, 4. **int** len, 5. **int flags** 6. )**;** |

|  |
| --- |
| 1. **int** send( 2. SOCKET s, 3. **const** **char** FAR \* buf, 4. **int** len, 5. **int flags** 6. )**;** |

SOCKET s: Là SOCKET được tạo ra khi Server chấp nhận kết nối từ CLIENT

* char FAR\* buf: Là dữ liệu (dạng BYTE – char) nhận hay gửi.
* int len: Kích thước của dữ liệu.
* int flags: Một số cờ hiệu đi kèm (thông thường là 0).

Socket Server :

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <time.h>

int main(int argc, char \*argv[])

{

int listenfd = 0, connfd = 0;

struct sockaddr\_in serv\_addr;

char sendBuff[1025];

time\_t ticks;

listenfd = socket(AF\_INET, SOCK\_STREAM, 0);

memset(&serv\_addr, '0', sizeof(serv\_addr));

memset(sendBuff, '0', sizeof(sendBuff));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

serv\_addr.sin\_port = htons(5000);

bind(listenfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr));

listen(listenfd, 10);

while(1)

{

connfd = accept(listenfd, (struct sockaddr\*)NULL, NULL);

ticks = time(NULL);

snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks));

write(connfd, sendBuff, strlen(sendBuff));

close(connfd);

sleep(1);

}

}

Socket Client :

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <arpa/inet.h>

int main(int argc, char \*argv[])

{

int sockfd = 0, n = 0;

char recvBuff[1024];

struct sockaddr\_in serv\_addr;

if(argc != 2)

{

printf("\n Usage: %s <ip of server> \n",argv[0]);

return 1;

}

memset(recvBuff, '0',sizeof(recvBuff));

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

printf("\n Error : Could not create socket \n");

return 1;

}

memset(&serv\_addr, '0', sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(5000);

if(inet\_pton(AF\_INET, argv[1], &serv\_addr.sin\_addr)<=0)

{

printf("\n inet\_pton error occured\n");

return 1;

}

if( connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

{

printf("\n Error : Connect Failed \n");

return 1;

}

while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)

{

recvBuff[n] = 0;

if(fputs(recvBuff, stdout) == EOF)

{

printf("\n Error : Fputs error\n");

}

}

if(n < 0)

{

printf("\n Read error \n");

}

return 0;

}