|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**  hutlogo  **BÁO CÁO BÀI TẬP LỚN** MÔN HỌC THỰC HÀNH LẬP TRÌNH MẠNG  **ĐỀ TÀI:** **XÂY DỰNG DỊCH VỤ MÃ HÓA GIẢI MÃ VĂN BẢN SỬ DỤNG THUẬT TOÁN HUFFMAN**   |  |  |  |  | | --- | --- | --- | --- | | Nhóm sinh viên thực hiện: | |  |  | |  | **Nguyễn Đình Hùng** | | **20131879** | |  | **Nguyễn Đình Chính** | | **20111243** |   Giảng viên hướng dẫn**: thầy Trần Hải Anh**  **Hà Nội, 12 – 2016.** |

**MỤC LỤC**

MỤC LỤC

[MỞ ĐẦU 1](#_Toc469439497)

[1.**Xác định yêu cầu của đề tài** 1](#_Toc469439498)

[2.**Phạm vi áp dụng** 1](#_Toc469439499)

[3.**Phân công công việc** 1](#_Toc469439500)

[4.**Tóm tắt nội dung báo cáo** 1](#_Toc469439501)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 2](#_Toc469439502)

[1.1 **Thuật toán Huffman** 2](#_Toc469439503)

[1.2 **Lập trình socket trên Linux** 2](#_Toc469439504)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ CHƯƠNG TRÌNH 4](#_Toc469439505)

[2.1 **Phía Client**: 4](#_Toc469439506)

[2.2 **Phía Server** 4](#_Toc469439507)

[2.3 **Mã hóa và giải mã theo kiểu Huffman** 4](#_Toc469439508)

[2.3.1 Mã hóa: 4](#_Toc469439509)

[2.3.2 Giải mã: 5](#_Toc469439510)

[CHƯƠNG 3: CÀI ĐẶT – KIỂM THỬ 6](#_Toc469439511)

[3.1 **Công cụ , môi trường lập trình và thử nghiệm**: 6](#_Toc469439512)

[3.2 **Cài đặt phía server và phía client**: 6](#_Toc469439513)

[3.3 **Cài đặt các hàm mã hóa và giải mã**: 6](#_Toc469439514)

[3.4 **Kiểm thử và kết quả**: 6](#_Toc469439515)

[CHƯƠNG 4: KẾT LUẬN 7](#_Toc469439516)

[4.1 **Nhận xét về chức năng** 7](#_Toc469439517)

[4.2 **Ưu điểm** 7](#_Toc469439518)

[4.3 **Nhược điểm** 7](#_Toc469439519)

[4.4 **Hướng phát triển** 7](#_Toc469439520)

[TÀI LIỆU THAM KHẢO 8](#_Toc469439521)

# MỞ ĐẦU

## 1.**Xác định yêu cầu của đề tài**

Đề tài “XÂY DỰNG DỊCH VỤ MÃ HÓA GIẢI MÃ VĂN BẢN SỬ DỤNG THUẬT TOÁN HUFFMAN” yêu cầu chương trình phải cung cấp được các chức năng sau :

1. Kết nối Client - Server.
2. Thực hiện được việc gửi file từ client đến server.
3. Ở phía server, thực hiện mã hóa hoặc giải mã file gửi tới bằng thuật toán Huffman.
4. Gửi trả file kết quả từ server về client.

## 2.**Phạm vi áp dụng**

Hệ điều hành Linux 64bit.

File văn bản ASCII 256 kí tự.

## 3.**Phân công công việc**

Nguyễn Đình Hùng(leader) : phân tích thiết kế tổng quan chương trình, cài đặt một số modun gửi và nhận file trên server và client.

Nguyễn Đình Chính: cài đặt các modun liên quan đến việc mã hóa và giải mã file.  
 Các nhiệm vụ nói trên chỉ là nhiệm vụ cơ bản của mỗi thành viên, trong quá trình làm chúng em có hỗ trợ lẫn nhau và giải thích cho nhau thêm về phần việc của mình nữa.

## 4.**Tóm tắt nội dung báo cáo**

Bài báo cáo của nhóm em gồm 4 phần sau:

Chương 1: Cơ sở lý thuyết.

Chương 2: Phân tích thiết kế chương trình.

Chương 3: Cài đặt và kiểm thử.

Chương 4: Kết luận.

*Chúng em xin cảm ơn thầy Trần Hải Anh đã hướng dẫn rất tận tình về mặt kiến thức và kĩ năng để nhóm em có thể hoàn thành được đề tài này!*

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1 **Thuật toán Huffman**

Trong [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh) và [lý thuyết thông tin](https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_th%C3%B4ng_tin), mã hóa Huffman là một thuật toán mã hóa dùng để nén dữ liệu. Nó dựa trên bảng tần suất xuất hiện các kí tự cần mã hóa để xây dựng một bộ mã nhị phân cho các kí tự đó sao cho dung lượng (số bít) sau khi mã hóa là nhỏ nhất.

Một giải thuật phổ biến được sử dụng để tạo ra một cây nhị phân biểu diễn bảng mã Huffman có tên là giải thuật tham lam:

Trong giải thuật tham lam giải bài toán xây dựng cây mã tiền tố tối ưu của Huffman, ở mỗi bước ta chọn hai chữ cái có tần số thấp nhất để mã hóa bằng từ mã dài nhất. Giả sử có tập A gồm n n {\displaystyle n} nnký hiệu và hàm trọng số tương ứng W ( i ) , i = 1.. n {\displaystyle W(i),i=1..n}W(i), i=1..n .

* Khởi tạo: Tạo một rừng gồm n cây, mỗi cây chỉ có một nút gốc, mỗi nút gốc tương ứng với một kí tự và có trọng số là tần số/tần suát của kí tự đó W ( i ) {\displaystyle W(i)}W(i).
* Lăp:
  + Mỗi bước sau thực hiện cho đến khi rừng chỉ còn một cây:
  + Chọn hai cây có trọng số ở gốc nhỏ nhất hợp thành một cây bằng cách thêm một gốc mới nối với hai gốc đã chọn. Trọng số của gốc mới bằng tổng trọng số của hai gốc tạo thành nó.

Như vậy ở mỗi bước số cây bớt đi một. Khi rừng chỉ còn một cây thì cây đó biểu diễn mã tiền tố tối ưu với các ký tự đặt ở các lá tương ứng.

## 1.2 **Lập trình socket trên Linux**

Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ ( client) và một chương trình cung cấp dịch vụ (server) trên mạng LAN, WAN hay Internet và đôi lúc là giữa những quá trình ngay bên trong máy tính. Mỗi socket có thể được xem như một điểm cuối trong một kết nối. Một socket trên máy yêu cầu dịch vụ có địa chỉ mạng được cấp sẵn để “gọi” một socket trên máy cung cấp dịch vụ. Một khi socket đã được thiết lập phù hợp, hai máy tính có thể trao đổi dịch vụ và dữ liệu.

Một Socket Address bao gồm: host name và port. Ví dụ: socket(127.0.0.1; 80)//cổng dịch vụ của HTTP

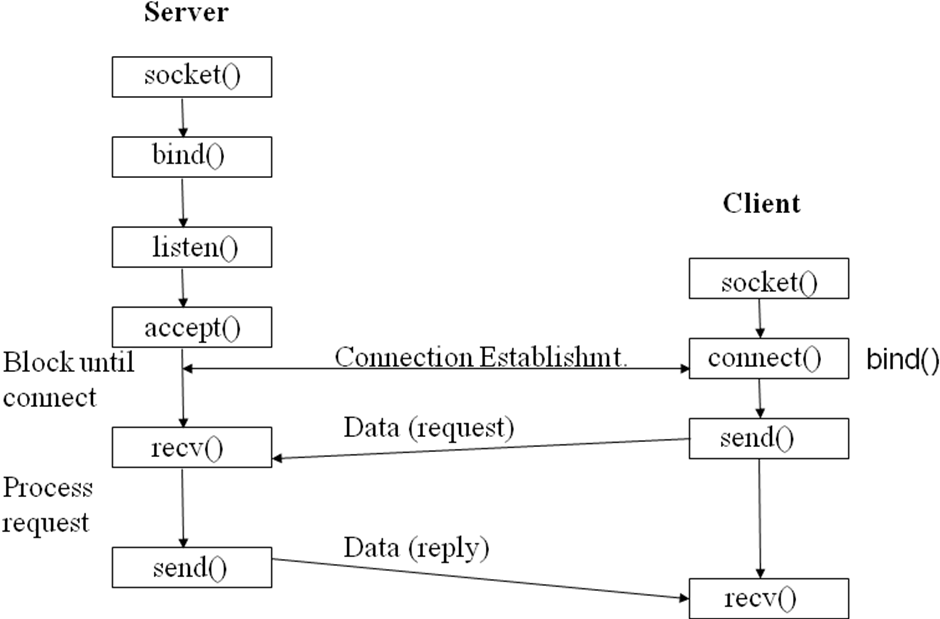
Trên Windows, thư viện Winsock được nhiều người sử dụng để lập trình mạng trên Windows. Tuy nhiên trong yêu cầu của chương trình là làm việc trong môi trường mã nguồn mở. Em xin được trình bày những nội dung về lập trình C Socket trên UNIX, với thư viện:

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

Có hai loại socket: stream sockets(SOCK\_STREAM) và datagram sockets(SOCK\_DGRAM). Tuy nhiên với dịch vụ mà chúng em cài đặt , cần yêu cầu cao về độ tin cậy của dữ liệu nên nhóm em sử dụng stream sockets. Stream sockets sử dụng giao thức TCP(Transmission Control Protocol) nên rất đáng tin cậy. Stream thường được dùng để chuyển các gói dữ liệu nối tiếp nhau(thứ tự của các gói dữ liệu(packets) là điều quan trọng nhất). Ví dụ, bạn có 3 gói dữ liệu "A,B,C", stream sockets sẽ chuyển theo đúng thứ tự "A,B,C". 

*Stream sockets*(TCP)‏

# CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ CHƯƠNG TRÌNH

Trong chương này chúng em sẽ mô tả cụ thể hơn phương pháp mà chúng em đã sử dụng để hoàn thành chương trình.

## 2.1 **Phía Client**:

Nhiệm vụ của client :

+chọn dịch vụ mã hóa hay là giải mã, nhận tên file mà người dùng nhập vào,

+mở file đó và xác định kích thước file, gửi tín hiệu xác định giải mã hay mã hóa và kích thước đó cho server,

+đọc file và gửi nội dung file cho server,

+nhận file kết quả từ server gửi về.

## 2.2 **Phía Server**

Nhiệm vụ của server:

+lắng nghe trên một cổng xác định, nếu có client nào yêu cầu kết nối đến thì fork ra một tiến trình con để làm việc với client đó,

+tiến trình server con mới tạo ra sẽ nhận tín hiệu xác định rằng client yêu cầu mã hóa hay giải mã, và nhận kích thước file,

+server con tiến hành giải mã hoặc mã hóa tùy theo yêu cầu, sau đó gửi file kết quả lại cho client.

## 2.3 **Mã hóa và giải mã theo kiểu Huffman**

### 2.3.1 Mã hóa:

Điều quyết định của việc mã hóa là tạo ra bảng mã tương ứng với file đầu vào. Hơn nữa, chúng em nhận thấy nếu dùng mảng tĩnh để lưu cây Huffman thì index của mỗi lá sẽ xác định duy nhất 1 lá cho nên thay vì việc dựng cây Huffman em đi thẳng vào việc xác định index của mỗi lá( mỗi lá chỉ chứa 1 chữ cái) :

+Đọc file đầu vào, đếm số lần xuất hiện của mỗi chữ cái trong file,

+Đưa các chữ cái đó vào các nút của một danh sách liên kết theo thứ tự tang dần tần số.

+Ghép 2 nút đầu danh sách thành một nút mới đồng thời cập nhật giá trị index của mỗi chữ cái đó. Rồi chèn nút đó vào danh sách sao cho danh sách vẫn có thứ tự.

+Lặp lại việc ghép đó cho đến khi danh sách chỉ còn 1 nút thì dừng. Lúc này ta đã có được bảng mã Huffman, mỗi chữ cái bây giờ sẽ có index tương ứng của nó .

+Ghi bảng mã ra file kết quả theo dạng: index,chữ cái, index, chữ cái,…

+Đọc lại file đầu vào một lần nữa, mỗi kí tự đọc được ta sẽ chuyển index của nó thành 1 dãy bit và ghi ra file kết quả.

### 2.3.2 Giải mã:

+Đọc file đã được mã hóa để lấy bảng mã

+Tiếp theo đọc từng bit của file ấy , sau khi đọc mỗi bit ta check xem chuỗi bit đã đọc được có phải là một chữ cái hay không. Nếu phải thì ghi chữ cái đó ra file giải mã, không phải thì đọc tiếp cho đến khi hết file.

# CHƯƠNG 3: CÀI ĐẶT – KIỂM THỬ

## 3.1 **Công cụ , môi trường lập trình và thử nghiệm**:

+Text Editor : SublimeText

+Ngôn ngữ lập trình C

+Hệ điều hành Ubuntu 16.04

## 3.2 **Cài đặt phía server và phía client**:

## Phía server:

## <https://github.com/huntojkojqw01/netpro/blob/master/sv.c>

## 

## Phía client:

## <https://github.com/huntojkojqw01/netpro/blob/master/cl.c>

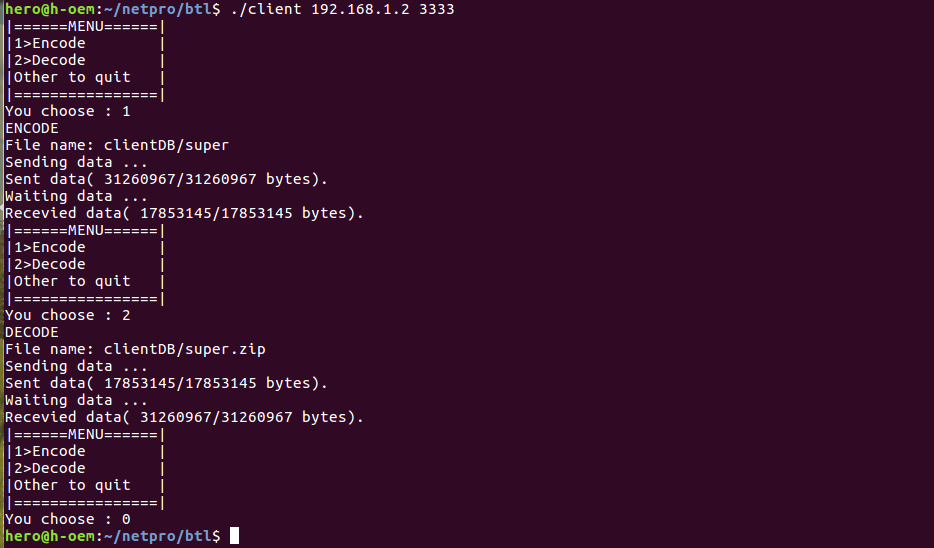
## 

## 3.3 **Cài đặt các hàm mã hóa và giải mã**:

## <https://github.com/huntojkojqw01/netpro/blob/master/lib/huffman.c>

## 3.4 **Kiểm thử và kết quả**:





# CHƯƠNG 4: KẾT LUẬN

## 4.1 **Nhận xét về chức năng**

Chương trình đã thực hiện thành công các yêu cầu cơ bản mà bài toán đặt ra. Cụ thể:

* Server nhận file thành công
* Mã hóa, giải mã file chính xác
* Client nhận được file kết quả đúng và đủ.
* Server có thể phục vụ được nhiều client

## 4.2 **Ưu điểm**

Xử lí trên từng bit và lưu ra file mã hóa nên đảm bảo được đặc điểm nén của thuật toán.

Server có thể phục vụ được nhiều client truy cập tới, lưu lại được nhật kí cung cấp dịch vụ.

Đã test thành công với file văn bản cỡ hàng chục MB.

File mã hóa có kích thước lớn vẫn không ảnh hưởng nhiều đến thời gian trả file sau khi server đã mã hóa hay giải mã.

## 4.3 **Nhược điểm**

Giao diện đơn giản, không có đồ họa.

Chưa có điều kiện thử nghiệm với số lượng lớn client thực tế.

Kích thước file văn bản đầu vào cho việc mã hóa bị giới hạn. Nguyên nhân là do chúng em chỉ dùng 4 byte(unsigned int) để lưu index của mỗi chữa cái tức là cái cây Huffman chỉ có thể có độ cao tối đa là 31. Nếu một file văn bản có kích cỡ từ 2 mũ 31 bytes trở lên thì có khả năng sẽ sinh ra một cây Huffman có độ cao 32 trở lên.

## 4.4 **Hướng phát triển**

Xây dựng đồ họa cho phía client . Có thể sử dụng GTK – phần mềm mã nguồn mở được sử dụng nhiều trong việc thiết kế đồ họa cho các chương trình chạy trên Linux.

Tạo các file CONFIG cho phía server và phía client.

Thêm chức năng lưu file (và có thể truyền lại )khi một phía tạm dừng hoặc ngắt kết nối.

# TÀI LIỆU THAM KHẢO

[1] Slide bài giảng thực hành lập trình mạng của thầy Trần Hải Anh.

[2] Các trang web về lập trình C, lập trình socket, thuật toán Huffman:

+https://en.wikipedia.org/wiki/Huffman\_coding +<http://www.cprogramming.com/tutorial/computersciencetheory/huffman.html>

+https://vi.wikipedia.org/wiki/M%C3%A3\_h%C3%B3a\_Huffman