TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

**BÀI TẬP LỚN**

MÔN: 93327

THỰC HÀNH LẬP TRÌNH MẠNG

**XÂY DỰNG DỊCH VỤ MÃ HÓA VÀ GIẢI MÃ VĂN BẢN SỬ DỤNG THUẬT TOÁN BASE64**

Sinh viên thực hiện : **LÊ THỊ KHANH**

Mã số sinh viên : **20142269**

Lớp : **93327 – K59**

Giáo viên hướng dẫn : **TS. Trần Hải Anh**

***Hà Nội, tháng 11 năm 2017***

NỘI DUNG

[**LỜI NÓI ĐẦU** 4](#_Toc499347860)

[**CHƯƠNG I: MÔ TẢ YÊU CẦU BÀI TOÁN** 5](#_Toc499347861)

[**CHƯƠNG II: TCP CLIENT/SERVER** 5](#_Toc499347862)

[**2.1.** **Mô hình Client/Server** 5](#_Toc499347863)

[**2.1.1.** **Client** 5](#_Toc499347864)

[**2.1.2.** **Server** 6](#_Toc499347865)

[**2.2.** **Tầng giao vận - UDP, TCP** 6](#_Toc499347866)

[**2.2.1.** **Tầng giao vận** 6](#_Toc499347867)

[**2.2.2.** **UDP (User Datagram Protocol – Giao thức datagram người dùng)** 7](#_Toc499347868)

[**2.2.3.** **TCP (Transmission Control Protocol – Giao thức điều khiển truyền vận)** 7](#_Toc499347869)

[**2.3.** **Lập trình socket** 8](#_Toc499347870)

[**2.3.1.** **Port** 8](#_Toc499347871)

[**2.3.2.** **Socket** 8](#_Toc499347872)

[**2.4.** **Lập trình socket với TCP trên Unix** 9](#_Toc499347873)

[**2.4.1.** **Quá trình giao tiếp giữa server và client** 9](#_Toc499347874)

[**2.4.2.** **Các hàm được sử dụng trong lập trình socket trên Unix** 10](#_Toc499347875)

[**CHƯƠNG III: THUẬT TOÁN BASE64** 11](#_Toc499347876)

[**3.1.** **Tổng quan về thuật toán Base64** 11](#_Toc499347877)

[**3.2.** **Mã hóa** 12](#_Toc499347878)

[**3.2.1.** **Bảng mã Base64** 12](#_Toc499347879)

[**3.2.2.** **Chuyển từ mã ASCII sang mã Base64** 12](#_Toc499347880)

[**3.2.3.** **Chuyển từ mã Base64 sang mã ASCII** 14](#_Toc499347881)

[**CHƯƠNG IV: HOÀN THIỆN CHƯƠNG TRÌNH** 15](#_Toc499347882)

[**4.1. Cách sử dụng chương trình** 15](#_Toc499347883)

[**4.2. Kết luận** 15](#_Toc499347884)

[Figure 1: Giao tiếp giữa Client-Server 5](#_Toc499346805)

[Figure 2: Packet exchange for TCP connection 8](#_Toc499346806)

[Figure 3: Lập trình Socket với TCP 9](#_Toc499346807)

[Table 1: Các hàm trong lập trình socket trên Unix 10](#_Toc499346808)

[Figure 4: Bảng mã Base64 12](#_Toc499346809)

[Figure 5: Mã hóa Base 64 13](#_Toc499346810)

[Figure 6: Mã hóa Base64 - Trường hợp đặc biệt 1 14](#_Toc499346811)

[Figure 7: Mã hóa Base64 - Trường hợp đặc biệt 2 14](#_Toc499346812)

[Table 2: Decode Base64 14](#_Toc499346813)

# **LỜI NÓI ĐẦU**

Mục tiêu của khoa học kỹ thuật nói chung và công nghệ thông tin nói riêng luôn là nâng cao tri thức của nhân loại, cải thiện đời sống, đưa nền văn minh của con người lên những bước tiến mới vĩ đại. Mà trong đó, sự ra ra đời của mạng máy tính là một bước tiến nền tảng. Một trong những đóng góp không thể phủ nhận của mạng máy tính đó là nó đã thay đổi hoàn toàn bộ mặt của thế giới, phủ sóng toàn bộ tới các lĩnh vực trong đời sống con người, kể cả lao động, làm việc, văn hóa tinh thần,…

Trong bài báo cáo này, em xin được trình bày tìm hiểu của mình về một mô hình mạng máy tính được phân loại theo chức năng: *Mô hình client/server.* Kèm theo đó là cài đặt thuật toán Base64 - một thuật toán được ứng dụng rộng rãi trong việc mã hóa dữ liệu thành dạng chuẩn trong các giao thức mạng, để không bị mất mát dữ liệu khi truyền đi - phục vụ việc encode và decode văn bản theo mô hình client/server.

Em xin chân thành cám ơn thầy Trần Hải Anh đã chỉ dạy cho chúng em những kiến thức nền tảng và kỹ thuật lập trình mạng máy tính, để em có thể hoàn thành bài tập lớn này.

*Hà Nội, 25/11/2017*

**Sinh viên**

*Lê Thị Khanh*

# **CHƯƠNG I: MÔ TẢ YÊU CẦU BÀI TOÁN**

**Đề bài:** Xây dựng dịch vụ mã hóa và giải mã văn bản sử dụng thuật toán Base64

**Yêu cầu:**

* Server:
  + Nghe trên một cổng (port)
  + Nhận file văn bản người dùng gửi lên từ client
  + Mã hóa/giải mã các xâu văn bản bằng thuật toán Base64, tạo thành file và gửi lại cho client. Việc mã hóa hay giải mã phụ thuộc vào tín hiệu gửi lên từ client.
* Client:
  + Cho phép người dùng nhập đường dẫn đến file cần mã hóa/giải mã
  + Gửi file văn bản cần mã hóa/giải mã lên server, sau đó nhận file đã mã hóa/giải mã về

# **CHƯƠNG II: TCP CLIENT/SERVER**

* 1. **Mô hình Client/Server**

Mô hình Client/Server là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy con (đóng vai trò là máy khách) gửi một yêu cầu đến máy chủ (đóng vai trò máy cung ứng dịch vụ), máy chủ sẽ xử lý và gửi trả kết quả về cho máy khách.

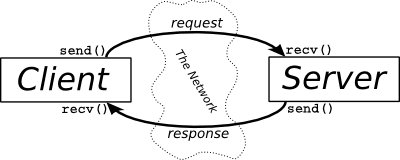


Figure 1: Giao tiếp giữa Client-Server

* + 1. **Client**

Người ta định nghĩa cụ thể cho một máy client là một máy trạm mà chỉ được sử dụng bởi 1 người dùng với để muốn thể hiện tính độc lập cho nó. Máy client có thể sử dụng các hệ điều hành bình thường như Win9x, DOS, OS/2... Bản thân mỗi một client cũng đã được tích hợp nhiều chức năng trên hệ điều hành mà nó chạy, nhưng khi được nối vào một mạng LAN, WAN theo mô hình client/server thì nó còn có thể sử dụng thêm các chức năng do hệ điều hành mạng (NOS) cung cấp với nhiều dịch vụ khác nhau (cụ thể là các dịch vụ do các server trên mạng này cung cấp), ví dụ như nó có thể yêu cầu lấy dữ liệu từ một server hay gửi dữ liệu lên server đó... Thực tế trong các ứng dụng của mô hình client/server, các chức năng hoạt động chính là sự kết hợp giữa client và server với sự chia sẻ tài nguyên, dữ liệu trên cả hai máy Vai trò của client Trong mô hình client/server, client được coi như là người sử dụng các dịch vụ trên mạng do một hoặc nhiều máy chủ cung cấp và server được coi như là người cung cấp dịch vụ để trả lời các yêu cầu của các clients. Điều quan trọng là phải hiểu được vai trò hoạt động của nó trong một mô hình cụ thể, một máy client trong mô hình này lại có thể là server trong một mô hình khác. Ví dụ cụ thể như một máy trạm làm việc như một client bình thường trong mạng LAN nhưng đồng thời nó có thể đóng vai trò như một máy in chủ (printer server) cung cấp dịch vụ in ấn từ xa cho nhiều người khác (clients) sử dụng. Client được hiểu như là bề nổi của các dịch vụ trên mạng, nếu có thông tin vào hoặc ra thì chúng sẽ được hiển thị trên máy client.

* + 1. **Server**

Server được định nghĩa như là một máy tính nhiều người sử dụng (multiuser computer). Vì một server phải quản lý nhiều yêu cầu từ các client trên mạng cho nên nó hoạt động sẽ tốt hơn nếu hệ điều hành của nó là đa nhiệm với các tính năng hoạt động độc lập song song với nhau như hệ điều hành UNIX, WINDOWS... Server cung cấp và điều khiển các tiến trình truy cập vào tài nguyên của hệ thống. Các ứng dụng chạy trên server phải được tách rời nhau để một lỗi của ứng dụng này khô ng làm hỏng ứng dụng khác. Tính đa nhiệm đảm bảo một tiến trình không sử dụng toàn bộ tài nguyên hệ thống. Vai trò của server. Như chúng ta đã bàn ở trên, server như là một nhà cung cấp dịch vụ cho các clients yêu cầu tới khi cần, các dịch vụ như cơ sở dữ liệu, in ấn, truyền file, hệ thống... Các ứng dụng server cung cấp các dịch vụ mang tính chức năng để hỗ trợ cho các hoạt động trên các máy clients có hiệu quả hơn. Sự hỗ trợ của các dịch vụ này có thể là toàn bộ hoặc chỉ một phần thông qua IPC. Để đảm bảo tính an toàn trên mạng cho nên server này còn có vai trò như là một nhà quản lý toàn bộ quyền truy cập dữ liệu của các máy clients, nói cách khác đó là vai trò quản trị mạng. Có rất nhiều cách thức hiện nay nhằm quản trị có hiệu quả, một trong những cách đang được sử dụng đó là dùng tên Login và mật khẩu

* 1. **Tầng giao vận - UDP, TCP**
     1. **Tầng giao vận**

Trong các ngành [tin học](https://vi.wikipedia.org/wiki/Tin_học) và [viễn thông](https://vi.wikipedia.org/wiki/Viễn_thông), ***tầng giao vận*** là tầng thứ tư trong bảy tầng của [mô hình OSI](https://vi.wikipedia.org/wiki/Mô_hình_OSI). Tầng này chịu trách nhiệm đáp ứng các đòi hỏi về dịch vụ của [tầng phiên](https://vi.wikipedia.org/wiki/Tầng_phiên) và đưa ra các yêu cầu dịch vụ đối với [tầng mạng](https://vi.wikipedia.org/wiki/Tầng_mạng).

Tầng giao vận cung cấp dịch vụ xuyên dụng chuyển dữ liệu giữa các [máy chủ](https://vi.wikipedia.org/wiki/Máy_chủ) (*hosts*). Tầng này chịu trách nhiệm sửa lỗi (*error recovery*), [điều khiển lưu lượng dữ liệu](https://vi.wikipedia.org/w/index.php?title=Điều_khiển_lưu_lượng_dữ_liệu&action=edit&redlink=1), đảm bảo dữ liệu được chuyển tải một cách trọn vẹn. Trong [Bộ giao thức liên mạng](https://vi.wikipedia.org/wiki/TCP/IP) - [TCP/IP](https://vi.wikipedia.org/wiki/TCP/IP), chức năng này thường được thực hiện bởi giao thức định hướng kết nối [TCP](https://vi.wikipedia.org/wiki/TCP).

Giao vận kiểu [datagram](https://vi.wikipedia.org/w/index.php?title=Datagram&action=edit&redlink=1), [UDP](https://vi.wikipedia.org/wiki/UDP) - *Giao thức Datagram Người dùng*, không cung cấp dịch vụ sửa lỗi hay điều khiển lưu lượng dữ liệu mà dành nhiệm vụ này cho [phần mềm ứng dụng](https://vi.wikipedia.org/wiki/Phần_mềm_ứng_dụng). Mục đích của tầng giao vận là cung cấp dịch vụ xuyên dụng chuyển dữ liệu giữa các người dùng đầu cuối, nhờ đó các tầng trên không phải quan tâm đến việc cung cấp dịch vụ truyền dữ liệu đáng tin cậy và hiệu quả.

Tậng giao vận thường biến dịch vụ đơn giản, có độ tin cậy thấp của tầng mạng thành một dịch vụ mạnh hơn. Có một danh sách dài liệt kê những dịch vụ có thể được cung cấp bởi tầng này. Không có một dịch vụ nào trong đó là bắt buộc cả, bởi vì không phải chương trình ứng dụng nào cũng yêu cầu tất cả những dịch vụ hiện có. Một số dịch vụ làm lãng phí chi phí phụ, hoặc trong vài trường hợp còn gây phản tác dụng.

* + 1. **UDP (User Datagram Protocol – Giao thức datagram người dùng)**

Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là ***datagram*** tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo.

Tuy nhiên UDP nhanh và hiệu quả đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn yêu cầu.

Những ứng dụng phổ biến sử dụng UDP như ***DNS*** (Domain Name System), ứng dụng ***Streaming data***, ***Voice over IP***, Trivial File Transfer Protoco***l*** (***TFTP***), và ***Game trực tuyến***.

* + 1. **TCP (Transmission Control Protocol – Giao thức điều khiển truyền vận)**

Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các kết nối với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và Email) đồng thời chạy trên cùng một máy chủ. TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet, trong đó có WWW, Email, và Secure Shell.

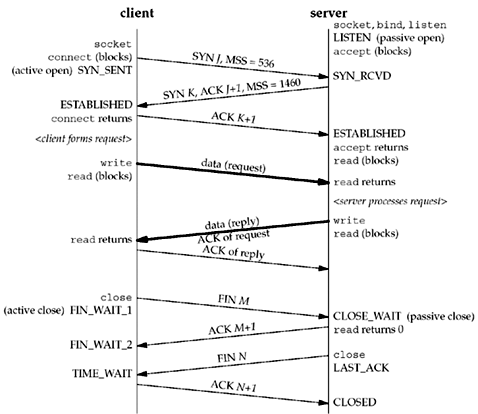


Figure 2: Packet exchange for TCP connection

* 1. **Lập trình socket** 
     1. **Port**

Trong giao thức TCP và UDP, port là số nguyên 16 bit được chèn vào phần đầu (header) của mỗi gói tin. Chẳng hạn, phía người dùng (client) có thể yêu cầu một máy chủ nào đó trên Internet cung cấp dịch vụ chia sẻ tập tin (file) qua máy chủ FTP. Để đáp ứng yêu cầu này, lớp phần mềm TCP trên máy của bạn phải nhận diện được port số 21 (đăng ký sẵn cho dịch vụ FTP) trong số các port 16 bit số nguyên được ghép theo gói tin yêu cầu của bạn. Tại máy chủ, lớp TCP sẽ đọc port 21 và chuyển tiếp yêu cầu đến máy chủ FTP.   
  
 Hay nói cách khác, với một địa chỉ IP, chúng ta chỉ có thể xác định được một máy tính duy nhất trên mạng, tuy nhiên khi một máy tính chạy nhiều dịch vụ khác nhau thì chúng phải được phân biệt bởi khái niệm port.

* + 1. **Socket**

Socket là một giao tiếp cục bộ trên host, được tạo bởi ứng dụng và được điều khiển bởi hệ điều hành, qua đó quá trình ứng dụng có thể truyền (hay nhận) đến (hay từ) quá trình khác.

Có 2 loại Socket:

* Stream Socket: Dựa trên giao thức TCP – việc truyền dữ liệu chỉ thực hiện giữa 2 tiến trình đã thiết lập kết nối.
* Datagram Socket: Dựa trên giao thức UDP – việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 tiến trình.
  1. **Lập trình socket với TCP trên Unix**

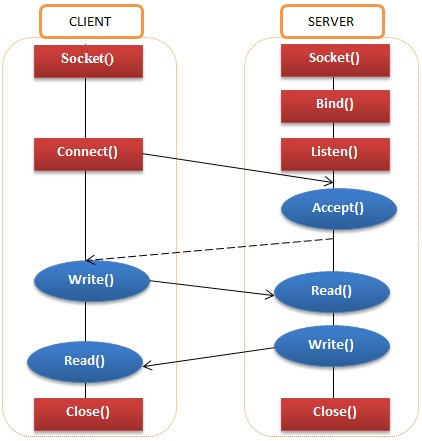


Figure 3: Lập trình Socket với TCP

* + 1. **Quá trình giao tiếp giữa server và client**
* Client phải liên lạc với server:
  + Trước hết quá trình trên server phải chạy
  + Server phải tạo socket để đón tiếp client
* Client liên lạc với server bằng cách:
  + Tạo ra TCP socket
  + Chỉ ra IP address, port number của quá trình trên server
  + Khi client tạo socket: client TCP thiết lập kết nối đến server TCP
* Khi được liên hệ bởi client, server TCP tạo socket mới để quá trình server giao tiếp với client:
  + Cho phép server giao tiếp với nhiều quá trình client
  + Các chỉ số port được dùng để phân biệt các client
    1. **Các hàm được sử dụng trong lập trình socket trên Unix**

Cấu trúc địa chỉ socket:

* Địa chỉ này lưu trữ địa chỉ IP, port number, và dạng giao thức (family protocol)
* Tên cấu trúc là ***sockaddr\_in***



|  |  |
| --- | --- |
| **Hàm** | **Ý nghĩa** |
| **SOCKET** | **Tạo cổng giao tiếp** |
| **BIND** | **Thêm địa chỉ local vào socket** |
| **LISTEN** | **Thông báo sẵn sang nhận kết nối** |
| **ACCEPT** | **Block cho đến khi nhận được kết nối** |
| **SEND** | **Gửi data đến tiến trình khác** |
| **RECEIVE** | **Nhận data từ tiến trình khác** |
| **CLOSE** | **Đóng socket, ngắt kết nối** |

Table 1: Các hàm trong lập trình socket trên Unix

Cụ thể các hàm như sau:











# **CHƯƠNG III: THUẬT TOÁN BASE64**

* 1. **Tổng quan về thuật toán Base64**

Thuật toán Base64 là một cơ chế biểu diễn dữ liệu nhị phân dưới dạng văn bản, sử dụng một hệ gồm 64 ký tự A  Z, a  z, 0  9 và ‘+’, ‘/’. Vì thế thuật toán này có tên gọi là Base64.

Tuy tiếng Việt đều gọi chung là ***“mã hóa”***, nhưng chúng ta cần phân biệt hai thuật ngữ ***“encoding”*** và ***“encryption”***. “Encoding” đơn thuần là chuyển dữ liệu từ dạng này sang dạng khác theo quy định của một hệ mã nào đó, phục vụ cho việc truyền dữ liệu, không nhằm mục đích bảo mật thông tin. Ví dụ như chuyển từ dạng văn bản ASCII sang hệ nhị phân. Còn “encryption” cũng biến đổi dữ liệu, nhưng có thể không cần thay đổi dạng thức của dữ liệu, việc này nhằm mục đích che giấu, bảo mật thông tin, để giải mã cần phải biết khóa bí mật.

Base64 là thuật toán được dùng trong “encoding”. Ứng dụng phổ biến và rộng rãi nhất của Base64 đó là chuyển đổi user name và password của người dùng khi đăng ký các trang web để đưa vào header của HTTP. Ngày nay, Base64 được sử dụng rộng rãi hơn: để mã hóa các tập tin đa phương tiện (hình ảnh, âm thanh, video, …) ứng dụng vào việc truyền tải hình ảnh trên website.

* 1. **Mã hóa**
     1. **Bảng mã Base64**

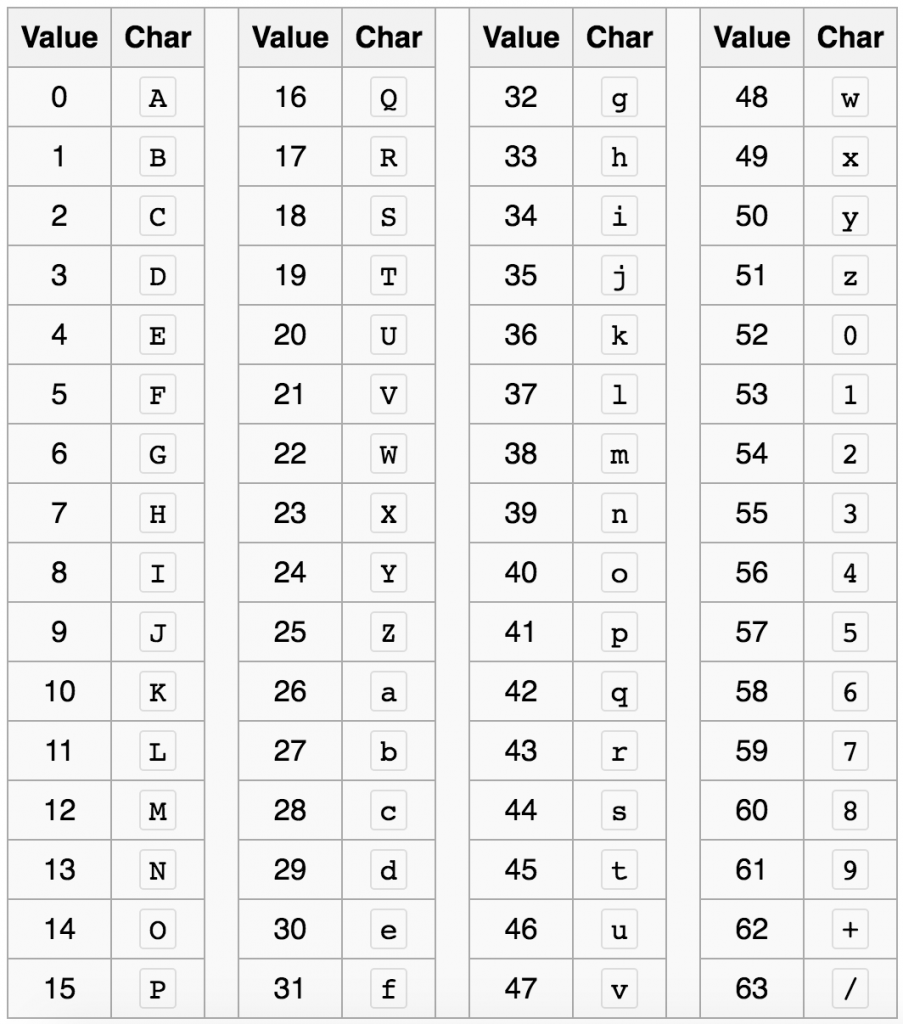


Figure 4: Bảng mã Base64

* + 1. **Chuyển từ mã ASCII sang mã Base64**

Ý tưởng của Base64 đó là thay vì dùng 8 bit để mã hóa một ký tự như trong bảng mã ASCII, ta sẽ dùng 6 bit (dùng 6 bit cần 2^6 = 64 ký tự để biểu diễn, thay vì dùng 8 bit cần 2^8 = 256 ký tự). Vì bội chung nhỏ nhất của 6 và 8 là 24, nên khi mã hóa ta sẽ mã hóa từng nhóm 24 bit của dữ liệu, tức là từ 3 ký tự ASCII sang 4 ký tự Base64. Sau khi chuyển sang mã Base64, kích thước file tăng lên gấp 4/3 1.33 lần.

Một ví dụ về mã hóa Base64. Giả sử ta có đoạn văn bản như sau:

“Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight

in the continued and indefatigable generation of knowledge, exceeds the short

vehemence of any carnal pleasure.”

Đoạn văn bản trên sau khi mã hóa sang Base64 sẽ thành:

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu

dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=

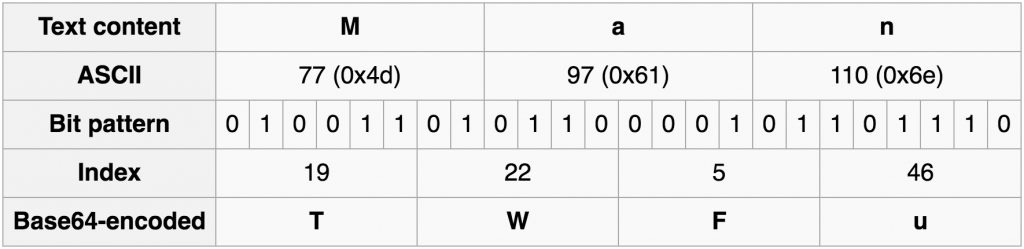


Figure 5: Mã hóa Base 64

Trong bảng trên, chữ “Man” được chuyển thành “TWFu” theo bảng mã Base64. Chữ *M, a, n* lần lượt được mã hóa thành các bytes 77, 97, 110 theo mã ASCII, tương ứng với các giá trị 8-bit: *01001101, 01100001, và 01101110.* Ba giá trị này ghép với nhau tạo thành một chuỗi 24-bit là *010011010110000101101110.* Chuỗi này được chia thành 4 nhóm 6-bit, và mỗi nhóm này được chuyển sang số thập phân, rồi chuyển thành ký tự tương ứng trong bảng mã Base64.

Tuy nhiên, khi mã hóa một dãy ký tự, hay một văn bản, tổng tất cả các ký tự có thể không chia hết cho 3. Vì vậy nhóm bit cuối cùng của file có thể không phải là 3 bytes mà là 2 bytes hoặc 1 bytes. Giải pháp đó là, thay các nhóm 6 bit chứa toàn số 0 bởi ký tự ‘=’. Ví dụ:



Figure 6: Mã hóa Base64 - Trường hợp đặc biệt 1

Trong trường hợp trên, xâu “Ma” được mã hóa thành “TWE=”.

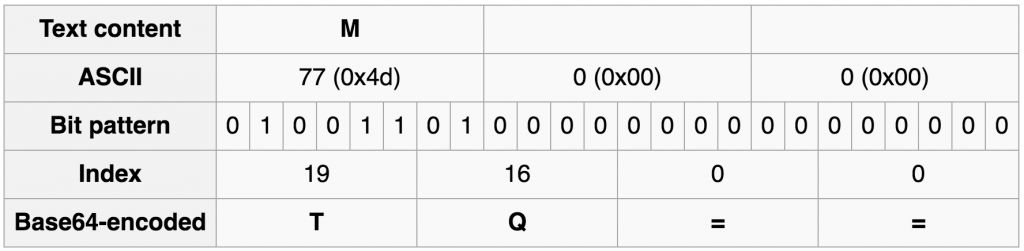


Figure 7: Mã hóa Base64 - Trường hợp đặc biệt 2

Trong trường hợp trên, xâu “M” được mã hóa thành “TQ==”.

* + 1. **Chuyển từ mã Base64 sang mã ASCII**

Ngược với cách làm trên, ta sẽ chuyển từng nhóm 4 ký tự của đoạn mã Base64 về 3 ký tự của mã ASCII. Ở công đoạn này, chúng ta không cần xét đến việc tổng số ký tự có chia hết cho 4 hay không, vì sau khi mã hóa sang Base64, hiển nhiên tổng số ký tự là bội số của 4. Ta chỉ cần quan tâm đến nhóm 4 ký tự kết thúc bởi ký tự ‘=’.

Nếu nhóm 4 ký tự Base64 kết thúc bởi 1 ký tự ‘=’, thì nhóm 3 ký tự ASCII tương ứng sẽ chứa 2 ký tự.

Nếu nhóm 4 ký tự Base64 kết thúc bởi 2 ký tự ‘=’, thì nhóm 3 ký tự ASCII tương ứng chỉ có 1 ký tự.

Nhóm 4 ký tự chứa ký tự ‘=’ nếu có, chỉ xuất hiện ở cuối file.

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoded** | **Số dấu ‘=’** | **Độ dài xâu** | **Decoded** |
| YW55IGNhcm5hbCBwbGVh**cw==** | 2 | 1 | any carnal plea**s** |
| YW55IGNhcm5hbCBwbGVh**c3U=** | 1 | 2 | any carnal plea**su** |
| YW55IGNhcm5hbCBwbGVh**c3Vy** | 0 | 3 | any carnal plea**sur** |

Table 2: Decode Base64

# **CHƯƠNG IV: HOÀN THIỆN CHƯƠNG TRÌNH**

Chương trình mã hóa và giải mã văn bản sử dụng thuật toán Base64 sẽ bao gồm: một chương trình client, một chương trình server, và bộ thuật toán mã hóa / giải mã base64 (source code ở trong file đính kèm). Chương trình chỉ sử dụng thư viện của Unix, các phần còn lại như mã hóa, giải mã, gửi, nhận file là sinh viên tự viết.

## **4.1. Cách sử dụng chương trình**

**Bước 1:** Người dùng bật server bằng cách gõ dòng lệnh sau trên terminal

**./server PORT\_NUMBER**

(PORT\_NUMBER có thể nhập hoặc không)

**Bước 2:** Người dùng chạy tiến trình client, đồng thời kết nối tới server đã bật ở trên bằng cách chạy dòng lệnh sau trong terminal:

**./client 127.0.0.1 PORT\_NUMBER**

(nếu dùng PORT\_NUMBER, phải nhập đúng với PORT\_NUMBER của server đã nhập ở trên)

**Bước 3:** Ở cửa sổ terminal của client, người dùng tiếp tục được yêu cầu nhập đường dẫn đến file văn bản cần mã hóa/giải mã.

**Bước 4:** Vẫn ở cửa sổ terminal của chương trình client, người dùng sẽ được yêu cầu nhập mục đích của mình. Mục đích là **encode** hoăc **decode**. Người dùng cần gõ đúng 1 trong 2 từ “**en**” hoặc “**de**” vào trong terminal.

**Bước 5:** Kết quả trả về ở file **cli\_result.txt.**

## **4.2. Kết luận**

Chương trình đã xây dựng được dịch vụ mã hóa và giải mã văn bản sử dụng thuật toán Base64: server đa tiến trình, nghe trên một cổng, client có thể gửi liên tiếp các yêu cầu, kết thúc một yêu cầu client sẽ tự ngắt kết nối tới server. Ở trong chương trình này, yêu cầu của client đối với server là encode/decode văn bản được gửi lên từ server, server làm nhiệm vụ encode/decode xong sẽ gửi trả văn bản kết quả về cho client.