|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TẬP ĐOÀN CÔNG NGHIỆP - VIỄN THÔNG QUÂN ĐỘI**  TRUNG TÂM NGHIÊN CỨU CÔNG NGHỆ MẠNG VIETTEL  LogoViettel_Text.jpg  **BÁO CÁO THU HOẠCH**   |  |  |  | | --- | --- | --- | | **Người viết** | **:** | **Trần Ngọc Hùng** | | **Người hướng dẫn** | **:** | **Nguyễn Sư Vĩnh** | | **Người quản lý trực tiếp**  **Phòng Tổ chức Lao động** | **:**  **:** | **Nguyễn Thanh Tiến**  **Lê Thu Hằng** | |  |  |  |   *TP.HCM, ngày 06 tháng 06 năm 2019* |

**TẬP ĐOÀN CÔNG NGHIỆP - VIỄN THÔNG QUÂN ĐỘI**

TRUNG TÂM NGHIÊN CỨU CÔNG NGHỆ MẠNG VIETTEL



**BÁO CÁO THU HOẠCH**

|  |  |  |
| --- | --- | --- |
| **Người viết** | **:** | **Trần Ngọc Hùng** |
| **Người hướng dẫn** | **:** | **Nguyễn Sư Vĩnh** |
| **Người quản lý trực tiếp**  **Phòng Tổ chức Lao động** | **:**  **:** | **Nguyễn Thanh Tiến**  **Lê Thu Hằng** |
|  |  |  |
|  |  |  |
|  |  |  |

*TP.HCM, ngày 06 tháng 06 năm 2019*

**MỤC LỤC**

[MỤC LỤC 3](#_Toc4592444)

[DANH MỤC HÌNH VẼ 6](#_Toc4592445)

[LỜI NÓI ĐẦU 8](#_Toc4592446)

[1. NHẬN THỨC VỀ TẬP ĐOÀN – VĂN HÓA VIETTEL 9](#_Toc4592447)

[1.1 Tập đoàn Công nghiệp - Viễn thông Quân đội 9](#_Toc4592448)

[1.1.1 Những mốc son trong lịch sử phát triển của Tập đoàn 9](#_Toc4592449)

[1.1.2 Các dấu mốc phát triển của các dịch vụ bưu chính viễn thông 11](#_Toc4592451)

[1.1.3 Mô hình của tập đoàn tại thời điểm viết bài thu hoạch 12](#_Toc4592452)

[1.2 Tổng công ty công nghiệp – công nghệ cao Viettel 13](#_Toc4592453)

[1.2.1 Giới thiệu chung 13](#_Toc4592454)

[1.2.2 Nhiệm vụ chính 14](#_Toc4592455)

[1.2.3 Quan điểm tổ chức xây dựng mô hình Tổng công ty 14](#_Toc4592456)

[1.2.4 Một số mục tiêu chiến lược của Tổng công ty Công nghiệp – Công nghệ cao 14](#_Toc4592457)

[1.2.5 Mô hình khung của Tổng công ty 15](#_Toc4592458)

[1.2.6 Trung tâm Kĩ thuật công nghệ 17](#_Toc4592459)

[1.3 Triết lý thương hiệu 18](#_Toc4592460)

[1.3.1 Tầm nhìn thương hiệu 18](#_Toc4592461)

[1.3.2 Ý nghĩa slogan: “Hãy Nói Theo Cách Của Bạn” 19](#_Toc4592462)

[1.3.3 Ý nghĩa logo 20](#_Toc4592463)

[1.3.4 Triết lý kinh doanh 21](#_Toc4592464)

[1.4 Giá trị cốt lõi của văn hóa Viettel 21](#_Toc4592465)

[1.4.1 Thực tiễn là tiêu chuẩn để kiểm nghiệm chân lý 22](#_Toc4592466)

[1.4.2 Trưởng thành qua những thách thức và thất bại 23](#_Toc4592467)

[1.4.3 Thích ứng nhanh là sức mạnh cạnh tranh 23](#_Toc4592468)

[1.4.4 Sáng tạo là sức sống 24](#_Toc4592469)

[1.4.5 Tư duy hệ thống 24](#_Toc4592470)

[1.4.6 Kết hợp Đông – Tây 25](#_Toc4592471)

[1.4.7 Truyền thống và cách làm người lính 25](#_Toc4592472)

[1.4.8 Viettel là ngôi nhà chung 26](#_Toc4592473)

[2. BÁO CÁO CHƯƠNG TRÌNH TRAINING 27](#_Toc4592474)

[2.1 Tìm hiểu về C trong môi trường Linux 27](#_Toc4592475)

[2.1.1 Hàm và cấu trúc chương trình 27](#_Toc4592476)

[2.1.2 Con trỏ và mảng 28](#_Toc4592477)

[2.1.3 Dữ liệu có cấu trúc 30](#_Toc4592478)

[2.1.4 Input và Output trong C 32](#_Toc4592479)

[*2.2 Tìm hiểu về socket* 34](#_Toc4592480)

[2.2.1 Định nghĩa Socket 34](#_Toc4592481)

[2.2.2 Cấu trúc địa chỉ của Socket 36](#_Toc4592482)

[2.2.3 Các hàm sử dụng trong socket 37](#_Toc4592483)

[2.2.4 TCP Socket 41](#_Toc4592484)

[2.2.5 UDP Socket 43](#_Toc4592485)

[2.3 Tìm hiểu về cơ chế giao tiếp giữa các process 44](#_Toc4592486)

[2.3.1 Process 44](#_Toc4592487)

[2.3.2 IPC 45](#_Toc4592488)

[2.4 Tìm hiểu về Posix Thread 54](#_Toc4592489)

[2.4.1 Định nghĩa Thread 54](#_Toc4592490)

[2.4.2 So sánh Thread và Process 54](#_Toc4592491)

[2.4.3 Đặc điểm của Thread trong Linux 55](#_Toc4592492)

[2.4.4 Semaphore 56](#_Toc4592493)

[2.4.5 Mutex 57](#_Toc4592494)

[2.4.6 So sánh Mutex và Semaphore 57](#_Toc4592495)

[2.5 Lab 1 58](#_Toc4592496)

[2.5.1 Nhiệm vụ và yêu cầu 58](#_Toc4592497)

[2.5.2 Tìm hiểu về giao thức Stop and Wait 59](#_Toc4592498)

[2.5.3 Quá trình thực hiện 61](#_Toc4592499)

[2.5.4 Kết quả thực hiện 63](#_Toc4592500)

[2.6 Lab 2 64](#_Toc4592501)

[2.6.1 Nhiệm vụ và yêu cầu 64](#_Toc4592502)

[2.6.2 Tìm hiểu về cơ chế Selective Repeat 64](#_Toc4592503)

[2.6.3 Quá trình thực hiện 67](#_Toc4592504)

[3. KẾT QUẢ ĐẠT ĐƯỢC TRONG THỜI GIAN THỬ VIỆC 69](#_Toc4592505)

[4. CÁC TÀI LIỆU ĐÃ ĐỌC, HỌC TRONG THỜI GIAN THỬ VIỆC 71](#_Toc4592506)

[5. NHẬN XÉT CỦA CÁC CÂP 72](#_Toc4592507)

[5.1 Nhận xét, đánh giá của người quản lý trực tiếp: (Mặt tích cực, hạn chế, triển vọng) 72](#_Toc4592508)

[5.2 Nhận xét của người quản lý chức năng (nếu có) 73](#_Toc4592509)

[5.3 Xét duyệt của Ban giám đốc 74](#_Toc4592510)

# DANH MỤC HÌNH VẼ

[*Hình 1‑1: Sơ đồ tổ chức tập đoàn Công nghiệp - Viễn thông Quân đội Viettel* 13](#_Toc10712298)

[*Hình 1‑2: Mô hình tổ chức Tổng công ty* 15](#_Toc10712299)

[*Hình 1‑3: Mô hình trung tâm kĩ thuật công nghệ* 17](#_Toc10712300)

[*Hình 1‑4: Logo Viettel* 20](#_Toc10712301)

[*Hình 2‑1: Mảng 10 phần tử* 28](#_Toc10712302)

[*Hình 2‑2: Ví dụ về con trỏ* 28](#_Toc10712303)

[*Hình 2‑3: Ví dụ về kiểu struct* 30](#_Toc10712304)

[*Hình 2‑4: So sánh giữa Struct và Union* 30](#_Toc10712305)

[*Hình 2‑5: Ví dụ về struct dạng Bit-Field* 31](#_Toc10712306)

[*Hình 2‑6: Mô hình socket* 33](#_Toc10712307)

[*Hình 2‑7: Vị trí của socket trong mô hình OSI* 34](#_Toc10712308)

[*Hình 2‑8: Cấu trúc địa chỉ Socket* 35](#_Toc10712309)

[*Hình 2‑9: Kernel thực hiện hàm socket* 35](#_Toc10712310)

[*Hình 2‑10: Quan hệ giữa socket và kernel linux* 38](#_Toc10712311)

[*Hình 2‑11: Quá trình thực hiện các hàm read/write dữ liệu của socket* 39](#_Toc10712312)

[*Hình 2‑12: Quá trình bắt tay 3 bước của TCP* 40](#_Toc10712313)

[*Hình 2‑13: Cấu trúc gói tin TCP* 40](#_Toc10712314)

[*Hình 2‑14: Quá trình thiết lập TCP socket* 41](#_Toc10712315)

[*Hình 2‑15: Quá trình gửi và nhận dữ liệu UDP* 42](#_Toc10712316)

[*Hình 2‑16: Cấu trúc gói tin UDP* 42](#_Toc10712317)

[*Hình 2‑17: Các loại IPC* 45](#_Toc10712318)

[*Hình 2‑18: Cấu trúc địa chỉ của Unix Domain Socket* 46](#_Toc10712319)

[*Hình 2‑19: các gói message trong message queue* 48](#_Toc10712320)

[*Hình 2‑20: Lưu trữ message queue* 49](#_Toc10712321)

[*Hình 2‑21: Cơ chế hoạt động của Message Queue* 49](#_Toc10712322)

[*Hình 2‑22: Cơ chế hoạt động của Shared Memory* 51](#_Toc10712323)

[*Hình 2‑23: Vùng nhớ process và thread* 54](#_Toc10712324)

[*Hình 2‑24: Task List trong hệ điều hành Linux* 55](#_Toc10712325)

[*Hình 2‑25: Cơ chế hoạt động của Semaphore* 58](#_Toc10712326)

[*Hình 2‑26: Cơ chế hoạt động của Mutex* 59](#_Toc10712327)

[*Hình 2‑27: Deadlock trong mutex(1)* 60](#_Toc10712328)

[*Hình 2‑28: Deadlock trong mutex(2)* 60](#_Toc10712329)

[*Hình 2‑29: Sơ đồ hoạt động Lab 1* 62](#_Toc10712330)

[*Hình 2‑30: Hoạt động của cơ chế Stop and Wait* 63](#_Toc10712331)

[*Hình 2‑31: Sơ đồ trạng thái của TCP* 63](#_Toc10712332)

[*Hình 2‑32: Kết quả Lab 1* 68](#_Toc10712333)

[*Hình 2‑33: Sơ đồ hoạt động của Lab2* 69](#_Toc10712334)

[*Hình 2‑34: Cơ chế của Selective Repeat* 71](#_Toc10712335)

[*Hình 2‑35: Kết quả Lab 2* 73](#_Toc10712336)

# LỜI NÓI ĐẦU

Qua 29 năm sáng tạo và phát triển, Tập đoàn Công nghiệp - Viễn thông Quân đội Viettel đã vươn mình trở thành tập đoàn viễn thông hàng đầu Việt Nam và là thương hiệu uy tín ở Việt Nam cũng như trên thế giới. Với các lĩnh vực ngành nghề hoạt động rộng khắp và phương châm phát triển nhanh, bền vững, Viettel đã và đang có mặt ở mọi miền đất nước, không chỉ mang đến những dịch vụ đột phá làm thay đổi bộ mặt nền viễn thông nước nhà mà còn đóng góp vào nền kinh tế quốc gia những giá trị to lớn. Bên cạnh đó, Viettel còn đạt được những thành công to lớn khi vươn ra nước ngoài, đạt được những mục tiêu thứ hạng cao trên bản đồ viễn thông thế giới.

Sau hai tháng tham gia thử việc tại Trung tâm kĩ thuật công nghệ, Tổng công ty công nghiệp – công nghệ cao Viettel, tôi thấy được sức mạnh của Viettel đến từ sự kỷ luật và sáng tạo của mỗi cá nhân góp phần vào thành công chung của một tập thể, đồng thời thấy được những thiếu sót và yếu điểm của cá nhân mình và tìm cách khắc phục để ngày một hoàn thiện hơn. Báo cáo thử việc này được thực hiện theo sự phân công và hướng dẫn của Trung tâm kĩ thuật công nghệ. Nội dung của báo cáo gồm các phần như sau:

Phần 1: Nhận thức về tập đoàn – Văn hóa Viettel

Phần 2: Báo cáo chương trình training

Phần 3: Kết quả đạt được trong thời gian thử việc

Phần 4: Các tài liệu đã đọc trong thời gian thử việc

Phần 5: Nhận xét của các cấp

Trong quá trình thử việc, tôi đã nhận được nhiều sự hướng dẫn và giúp đỡ của các anh chị, các bạn trong nhóm nghiên cứu thiết bị Site Router và nhóm nghiên cứu thiết bị ONT. Tôi xin chân thành cảm ơn!

*TP.HCM, ngày 06 tháng 06 năm 2019*

Người viết báo cáo

**Trần Ngọc Hùng**

# NHẬN THỨC VỀ TẬP ĐOÀN – VĂN HÓA VIETTEL

## Tập đoàn Công nghiệp - Viễn thông Quân đội

### Những mốc son trong lịch sử phát triển của Tập đoàn

|  |  |
| --- | --- |
| **Thời gian** | **Sự kiện** |
| 01/06/1989 | Thành lập Tổng Công ty Điện tử thiết bị thông tin trực thuộc Bộ Tư lệnh Thông tin liên lạc, Bộ Quốc phòng. Đây là dấu son lịch sử đánh dấu sự ra đời và ngày 1/6 đã trở thành ngày truyền thống hàng năm của Tập đoàn Viễn thông Quân đội. |
| 27/07/1993 | Bộ Quốc phòng ra quyết định số 336/QĐ-QP (Do thứ trưởng Thượng tướng Nguyễn Trọng Xuyên ký) Thành lập lại doanh nghiệp Nhà nước: Công ty Điện tử thiết bị thông tin, tên giao dịch là SIGENCO, trụ sở chính tại 16 Cát Linh, Hà Nội. |
| 14/07/1995 | Bộ Quốc phòng ra Quyết định số 615/QĐ- QP đổi tên Công ty Điện tử thiết bị thông tin thành Công ty Điện tử - viễn thông Quân đội, tên giao dịch quốc tế là VIETEL (Lúc này cụm chữ chỉ có 01 chữ T). |
| 29/04/2003 | Thủ tướng Chính phủ ra Quyết định số 80/2003/QĐ-TTg Phê duyệt phương án tổng thể sắp xếp, đổi mới doanh nghiệp Nhà nước trực thuộc Bộ Quốc phòng giai đoạn 2003-2005. |
| 27/04/2004 | Bộ trưởng Bộ Quốc phòng ra quyết định số 51/QĐ-QP (do thứ trưởng BQP, trung tướng Nguyễn Văn Rinh ký) quyết định từ 01 tháng 7 năm 2004 điều chuyển Công ty viễn thông Quân đội từ Bộ Tư lệnh Thông tin về trực thuộc Bộ Quốc Phòng với tên gọi Công ty Viễn thông Quân đội tên giao dịch là VIETTEL. |
| 02/03/2005 | Thủ tướng Phan Văn Khải đã ký quyết định thành lập Tổng công ty Viễn thông Quân đội. Đây là dấu ấn khẳng định bước phát triển mới của Tổng Công ty cả về quy mô, năng lực và kinh nghiệm trong lĩnh vực viễn thông (từ Công ty phát triển thành Tổng Công ty). |
| 06/04/2005 | Bộ Quốc phòng có quyết định số 45/2005/BQP về việc thành lập Tổng công ty Viễn thông Quân đội, tên giao dịch quốc tế tiếng Anh là VIETTEL CORPORATION, viết tắt là VIETTEL. |
| 14/12/2009 | Thủ tướng Chính phủ ký quyết định số 2078/QĐ-TTg về việc phê duyệt Đề án thí điểm thành lập Tập đoàn Viễn thông Quân đội và Quyết định số 2079/QĐ-TTg thành lập Tập đoàn Viễn thông Quân đội. |
| 12/01/2010 | Tại trụ sở số 1 Giang Văn Minh, Ba Đình, Hà Nội, Viettel đã long trọng tổ chức Lễ ra mắt Tập đoàn và đón nhận Huân chương Độc lập Hạng Ba. Đây là dấu ấn khẳng định bước phát triển vượt bậc, một mốc son quan trọng đánh dấu sự lớn mạnh của Viettel cả về quy mô, năng lực và kinh nghiệm (Mô hình Tập đoàn thí điểm, trực thuộc bộ chủ quản, không có hội đồng quản trị). |
| 05/01/2018 | Chính phủ ban hành Nghị định số 05/2018/NĐ-CP về Điều lệ tổ chức và hoạt động của Công ty mẹ - Tập đoàn Viettel. Theo đó, Tập đoàn Công nghiệp – Viễn thông Quân đội được đổi tên từ Tập đoàn Viễn thông Quân đội là doanh nghiệp quốc phòng, an ninh, do Nhà nước nắm giữ 100% vốn điều lệ. |



### Các dấu mốc phát triển của các dịch vụ bưu chính viễn thông

|  |  |
| --- | --- |
| **Thời gian** | **Sự kiện** |
| 01/07/1997 | Triển khai dịch vụ Bưu chính. |
| 05/10/2000 | Thử nghiệm dịch vụ điện thoại đường dài 178, công nghệ VoIP. |
| 09/10/2002 | Khai trương dịch vụ Internet. |
| 09/2003 | Triển khai dịch vụ điện thoại cố định. |
| 15/10/2004 | Khai trương dịch vụ Điện thoại Di động. |
| 2006 | Đầu tư sang Campuchia. |
| 2007 | Đầu tư sang Lào. |
| 03/2007 | Triển khai dịch vụ Điện thoại cố định không dây. |
| 19/02/2009 | Khai trương dịch vụ Metfone tại Campuchia. |
| 16/10/2009 | Khai trương dịch vụ Unitel tại Lào. |
| 25/03/2010 | Khai trương dịch vụ 3G tại Việt Nam. |
| 07/09/2011 | Khai trương mạng viễn thông Natcom tại Haiti. |
| 15/05/2012 | Khai trương mạng viễn thông Movitel tại Mozambique. |
| 10/7/2013 | Khai trương mạng Telemor tại Đông Timor. |
| 12/9/2014 | Khai trương mạng Nexttel tại Camaroon. |
| 15/10/2014 | Khai trương mạng Bitel tại Peru. |
| 01/2015 | Thủ tướng phê duyệt dự án A1. |
| 03/2015 | Khai trương Lumitel tại Burundi. |
| 10/2015 | Khai trương Halotel tại Tanzania. |
| 12/2015 | Thử nghiệm 4G tại Vũng Tàu. |

### Mô hình của tập đoàn tại thời điểm viết bài thu hoạch

**Tập đoàn Công nghiệp Viễn Thông - Quân Đội VIETTEL:**

* Trụ sở giao dịch: Số 1 – Trần Hữu Dực – Mỹ Đình – Từ Liêm – Hà Nội.
* Website: [www.viettel.com.vn](http://www.viettel.com.vn)
* Tên cơ quan sáng lập: Bộ Quốc phòng

Tập đoàn Viễn thông Quân đội được thành lập theo quyết định 2097/2009/QĐ-TTg của Thủ tướng Chính phủ ký vào ngày 14/12/2009, là doanh nghiệp kinh tế quốc phòng 100% vốn nhà nước với số vốn điều lệ 50.000 tỷ đồng, có tư cách pháp nhân, có con dấu, biểu tượng và điều lệ tổ chức riêng.

**Hoạt động kinh doanh:**

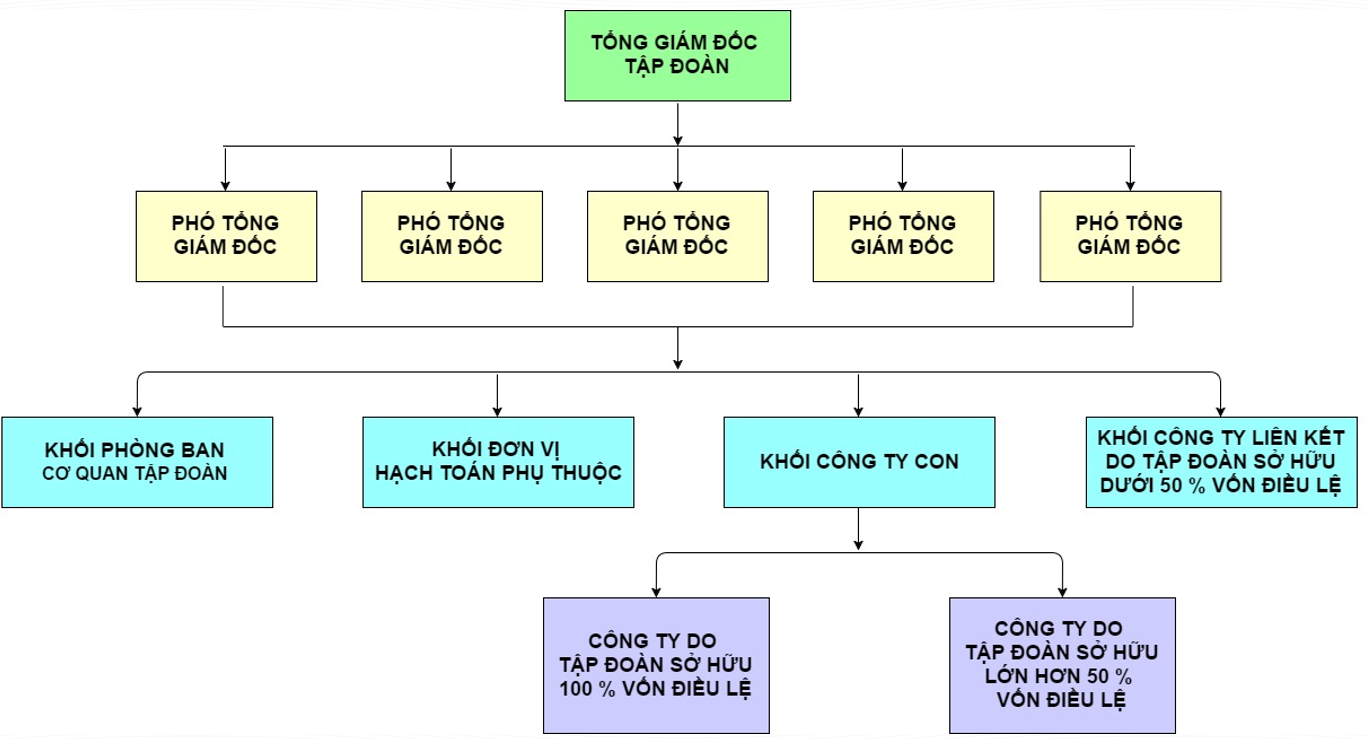
|  |  |
| --- | --- |
| * Cung cấp dịch vụ Viễn thông * Truyền dẫn * Bưu chính * Phân phối thiết bị đầu cuối * Đầu tư tài chính | * Truyền thông * Đầu tư bất động sản * Xuất nhập khẩu * Đầu tư nước ngoài |

**Ban Tổng giám đốc tập đoàn:**

* Tổng Giám đốc: Thiếu tướng Lê Đăng Dũng
* Phó Tổng Giám đốc: Đại tá Hoàng Sơn
* Phó Tổng Giám đốc: Đại tá Nguyễn Đình Chiến
* Phó Tổng Giám đốc: Đại tá Nguyễn Thanh Nam
* Phó Tổng Giám đốc: Thượng tá Đỗ Minh Phương
* Phó Tổng Giám đốc: Thượng tá Tào Đức Thắng

**Sơ đồ tổ chức tập đoàn Viettel:**

Tập đoàn Công nghiệp - Viễn thông Quân đội Viettel gồm các khối phòng ban và cơ quan chức năng như sau:



*Hình 1‑1: Sơ đồ tổ chức tập đoàn Công nghiệp - Viễn thông Quân đội Viettel*

## Tổng công ty công nghiệp – công nghệ cao Viettel

### Giới thiệu chung

Tổng công ty Công nghiệp Công nghệ cao Viettel là đơn vị hạch toán phụ thuộc, trực thuộc Tập đoàn Công nghiệp – Viễn thông Quân đội; thực hiện chức năng Nghiên cứu, phát triển, chế tạo, sản xuất, sửa chữa; kinh doanh sản phẩm, dịch vụ; cho thuể, xuất khẩu, nhập khẩu hàng lưỡng dụng (*trang thiết bị kỹ thuật, dịch vụ, vật tư, hàng hóa, có thể sử dụng cho quốc phòng và cả trong lĩnh vực kinh tế, xã hội*).

Tên tiếng Anh: VIETTEL HIGH TECHNOLOGY INDUSTRIES CORPORATION, BRANCH OF VIETTEL GROUP (tên Viết tắt: VHT)

### Nhiệm vụ chính

Tổng công ty có hai nhiệm vụ chính quan trọng là:

* Thực hiện toàn trình (Nghiên cứu thị trường – Phát triển sản phẩm, Sản xuất, Xúc tiến, Bán hàng, Triển khai, chuyển giao, Chăm sóc khách hàng) các sản phẩm/dự án công nghệ cao về quân sự và dân sự.
* Chịu trách nhiệm về hiệu quả kinh doanh, doanh thu, lợi nhuận và chi phí của Tổng công ty trước Chủ tịch, Tổng giám đốc Tập đoàn.

### Quan điểm tổ chức xây dựng mô hình Tổng công ty

Gồm có 3 quan điểm chính:

* **Tập trung hóa (Centralize)**:Tổ chức các đơn vị hỗ trợ tập trung tại cấp Tổng công ty, hoạt động theo mô hình đối tác chiến lược kinh doanh.
* **Dùng chung nguồn lực (Pooling)**: Tổ chức các pool dùng chung tại Khối nghiên cứu và Phát triển sản phẩm, hoạt động theo mô hình dự án.
* **Tinh gọn (Lean)**: Việc tổ chức các bộ phận và các vị trí quản lý phải đảm bảo theo quy định về Phạm vị kiểm soát (SOC) của tập đoàn (hiện tại là phải >= 7), giảm số lớp trung gian.

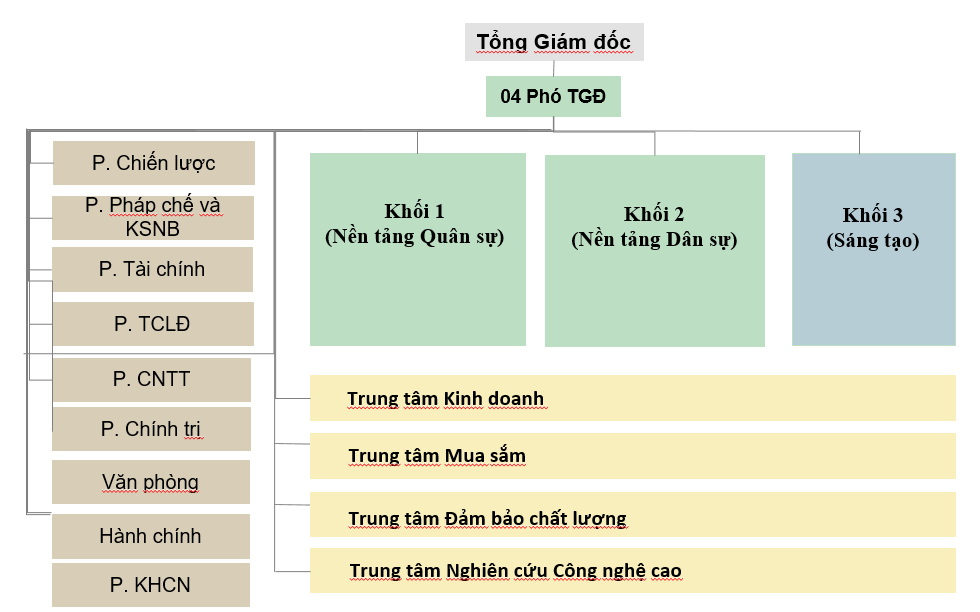
### Một số mục tiêu chiến lược của Tổng công ty Công nghiệp – Công nghệ cao

Tổng công ty được thành lập với 5 mục tiêu chiến lược:

* Đưa sản phẩm cả quân sự và dân sự của Viettel ra thị trường, đặc biệt là thị trường nước ngoài. Qua đó, chúng ta chứng minh được chất lượng sản phẩm của Viettel đạt chuẩn thế giới.
* Trở thành doanh nghiệp công nghệ cao dẫn đầu Việt Nam trong cuộc cách mạng công nghiệp lần thứ 4 với đội ngũ nhân lực toàn cầu chất lượng cao, hệ sinh thái đổi mới, sáng tạo và mô hình kinh doanh xuất sắc.
* Trở thành doanh nghiệp đứng trong Bảng xếp hạng 100 doanh nghiệp quốc phòng công nghệ cao hàng đầu thế giới; Top 5 toàn cầu về sản xuất thiết bị viễn thông.
* Doang thu từ các sản phẩm dân sự/quân sự đạt tỷ trọng 50/50.
* Làm chủ các công nghệ lõi sản xuất các sản phẩm quân sự; công nghệ lõi của các thiết bị hạ tầng viễn thông chủ chốt; sở hữu một hệ thống văn bằng bảo hộ sáng chế và giải pháp hữu ích đủ lớn để đáp ứng các yêu cầu kinh doanh của Tập đoàn.

### Mô hình khung của Tổng công ty

Tổng công ty được tổ chức như mô hình bên dưới:



*Hình 1‑2: Mô hình tổ chức Tổng công ty*

Mô hình tổ chức trên gồm có:

Ban giám đốc gồm:

* Tổng Giám đốc: Đồng chí Trung tá Nguyễn Vũ Hà
* Phó Tổng Giám đốc: Đồng chí Trung tá Đỗ Thanh Hải
* Phó Tổng Giám đốc: Đồng chí Thiếu tá Nguyễn Cương Hoàng
* Phó Tổng Giám đốc: Đồng chí Đại úy Đào Vũ Kiên
* Phó Tổng Giám đốc: Đồng chí Thiếu tá Nguyễn Minh Quang

Các phòng ban quản lý:

* Phòng Chiến lược
* Phòng Pháp chế và Kiểm soát nội bộ
* Phòng Tài chính
* Phòng Tổ chức lao động
* Phòng Công nghệ thông tin
* Phòng Chính trị
* Văn phòng
* Hành chính
* Phòng Khoa học công nghệ

Các trung tâm:

* Trung tâm Kinh doanh
* Trung tâm Mua sắm
* Trung tâm Đảm bảo chất lượng
* Trung tâm Nghiên cứu công nghệ cao

Các khối:

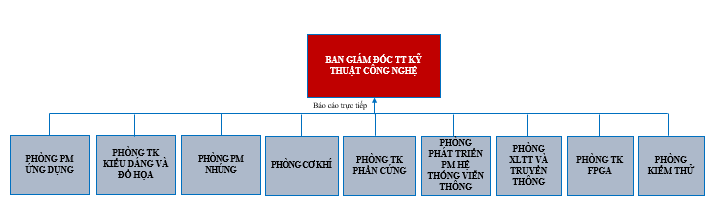
* Khối 1 (Nền tảng Quân sự)
* Khối 2 (Nền tảng Dân sự)
* Khối 3 (Sáng tạo)

### Trung tâm Kĩ thuật công nghệ

Trung tâm kĩ thuật công nghệ được chia ra làm 9 phòng ban chính:

* Phòng Phần mềm Ứng dụng
* Phòng Thiết kế kiểu dáng và Đồ họa
* Phòng Phần mềm Ứng dụng
* Phòng Phần mềm nhúng
* Phòng Cơ khí
* Phòng Thiết kế phần cứng
* Phòng Phát triển phần mềm hệ thống viễn thông
* Phòng Xử lí thông tin và truyền thông
* Phòng Thiết kế FPGA
* Phòng Kiểm thử

Sơ đồ của Trung tâm Kĩ thuật công nghệ



*Hình 1‑3: Mô hình trung tâm kĩ thuật công nghệ*

Các phòng ban sẽ báo cáo trực tiếp lên Ban giám đốc Trung tâm Kĩ thuật công nghệ.

## Triết lý thương hiệu

### Tầm nhìn thương hiệu

Tầm nhìn thương hiệu được cô đọng từ việc thấu hiểu những mong muốn của khách hàng và những nỗ lực đáp ứng của VIETTEL.

VIETTEL hiểu rằng, khách hàng luôn muốn được lắng nghe, quan tâm chăm sóc như những cá thể riêng biệt.

Còn VIETTEL sẽ nỗ lực để sáng tạo phục vụ những nhu cầu riêng biệt ấy với một sự chia sẻ, thấu hiểu nhất.

|  |  |  |
| --- | --- | --- |
| **Mong muốn của Khách hàng** | | **Sự đáp ứng của Viettel** |
| Được đối xử | * Mong muốn được lắng nghe. * Muốn được người khác hiểu nhu cầu và ước muốn. * Muốn được nhìn nhận như một cá thể riêng biệt. | * Viettel là người tiên phong. |
| Tiêu chuẩn dịch vụ | * Muốn được đáp ứng. * Muốn phục vụ bởi những nhà cung cấp tin cậy về chất lượng. * Muốn được quan tâm, chăm sóc. | * Viettel luôn luôn đổi mới. * Viettel luôn sáng tạo. |
| Quyền của Khách hàng | * Muốn được đối xử công bằng, thẳng thắn. * Muốn có được nhiều sự lựa chọn. * Khách hàng muốn được lắng nghe, quan tâm chăm sóc như những cá thể riêng biệt. | * Viettel lắng nghe. * Viettel thẳng thắn. * Viettel nhân từ. * Viettel thành thật. * Viettel có tinh thần giúp đỡ, hỗ trợ. * Viettel thông cảm với mọi người. * Viettel là nhà sáng tạo với một trái tim nhân từ. |

**Ý nghĩa tầm nhìn thương hiệu Viettel**

Tầm nhìn thương hiệu định ra một hướng đi chung cho các hoạt động của Viettel, được cô đọng từ việc tổng hợp cơ sở mong muốn của khách hàng và sự đáp ứng của Viettel, kết hợp giữa văn hoá phương Đông và phương Tây.

* INNOVATOR (phương Tây):
* Tiên phong, sáng tạo.
* Liên tục đổi mới, cải cách.
* Làm việc và tư duy logic có hệ thống.
* Cá thể hoá.
* CARING (phương Đông):
* Luôn lắng nghe, quan tâm, chăm sóc.
* Tư duy trực quan sinh động.
* Cơ chế cân bằng, ổn định.
* Tình cảm, có trách nhiệm xã hội, tham gia các hoạt động nhân đạo.

### Ý nghĩa slogan: “Hãy Nói Theo Cách Của Bạn”

VIETTEL luôn mong muốn phục vụ khách hàng như những cá thể riêng biệt. VIETTEL hiểu rằng, muốn làm được điều đó phải thấu hiểu khách hàng, phải lắng nghe khách hàng. Và vì vậy, khách hàng được khuyến khích nói theo cách mà họ mong muốn và bằng tiếng nói của chính mình – “Hãy nói theo cách của bạn”.

### Ý nghĩa logo



*Hình 1‑4: Logo Viettel*

Logo VIETTEL được thiết kế dựa trên ý tưởng lấy từ hình tượng dấu ngoặc kép. Khi bạn trân trọng câu nói của ai đó, bạn sẽ trích dẫn trong dấu ngoặc kép. VIETTEL quan tâm và trân trọng từng nhu cầu cá nhân của mỗi khách hàng. VIETTEL luôn luôn biết lắng nghe và cảm nhận, trân trọng những ý kiến của mọi người như những cá thể riêng biệt – các thành viên của Tập đoàn, khách hàng và đối tác. Đây cũng chính là nội dung của câu khẩu hiệu của VIETTEL: Hãy nói theo cách của bạn (Say it your way).

Logo VIETTEL mang hình elip được thiết kế đi từ nét nhỏ đến nét lớn, nét lớn lại đến nét nhỏ, biểu tượng cho sự chuyển động liên tục, sáng tạo không ngừng (văn hóa phương Tây) đồng thời cũng biểu tượng cho âm dương hòa quyện vào nhau (văn hóa phương Đông). Khối chữ VIETTEL được thiết kế có sự liên kết với nhau, thể hiện sự gắn kết, đồng lòng, kề vai sát cánh của các thành viên trong Tập đoàn.

Ba màu trên logo biểu tượng cho những ý nghĩa đặc biệt: màu xanh (thiên), màu vàng (địa), màu trắng (nhân). Màu xanh thiên thanh biểu hiện cho màu của trời, màu của khát vọng vươn lên, màu của không gian sáng tạo. Màu vàng đất biểu thị cho đất, màu của sự đầm ấm, gần gũi, đôn hậu, đón nhận. Màu trắng là nền của chữ Viettel, thể hiện sự chân thành, thẳng thắn, nhân từ. Sự kết hợp giao hòa giữa trời, đất và con người “Thiên thời – Địa lợi – Nhân hoà” theo những quan điểm của triết học và cũng gắn liền với lịch sử, định hướng của Tập đoàn thể hiện cho sự phát triển vững bền của thương hiệu VIETTEL.

### Triết lý kinh doanh

Mỗi khách hàng là một con người – một cá thể riêng biệt, cần được tôn trọng, quan tâm và lắng nghe, thấu hiểu và phục vụ một cách riêng biệt. Liên tục đổi mới, cùng với khách hàng sáng tạo ra các sản phẩm, dịch vụ ngày càng hoàn hảo.

Nền tảng cho một doanh nghiệp phát triển là xã hội. VIETTEL cam kết tái đầu tư lại cho xã hội thông qua việc gắn kết các hoạt động sản xuất kinh doanh với các hoạt động xã hội, hoạt động nhân đạo.

## Giá trị cốt lõi của văn hóa Viettel

Là một Tập đoàn thuộc Quân Đội nên Viettel đã lấy bản chất Bộ đội Cụ Hồ để xây dựng nền văn hoá của riêng mình. Nó không cứng nhắc, không mang tính cục bộ, mà luôn mềm dẻo, hướng ngoại, học hỏi. Qua 29 năm phát triển, nét văn hoá riêng của Viettel đã mang đậm tính chất doanh nhân, được thể hiện một cách linh hoạt qua cách ứng xử với thị trường, sẵn sàng giúp đỡ, thấu hiểu và chịu trách nhiệm cao thông qua việc đưa ra các giải pháp, các sản phẩm, dịch vụ hoàn hảo nhằm đáp ứng tốt nhất nhu cầu của khách hàng, đem lại sự lựa chọn tối ưu nhất của mọi khách hàng.

Nét văn hoá của Viettel từng bước được xây dựng mang đậm tính chuyên nghiệp, thể hiện của một Tập đoàn có tầm cỡ quốc tế. Viettel luôn coi con người là chủ thể để phát triển, do vậy mà nét văn hoá tổ chức được xây dựng có tính hướng ngoại với quan điểm cá thể con người với con người (Viettel với khách hàng) nên khách hàng là đối tượng được trân trọng, sẵn sàng được phục vụ, đáp ứng các dịch vụ tốt nhất. Mặt khác, luôn coi trọng nhân viên Tập đoàn với tình cảm chân thành, tạo điều kiện, khuyến khích, hỗ trợ phát huy khả năng, năng lực, quan tâm đến chính sách đào tạo và trọng dụng nhân tài, đề cao vai trò của từng cá nhân, con người trong sự phát triển của Tập đoàn và chăm lo đến đời sống tinh thần, vật chất cho cán bộ công nhân viên của mình, điều này xác định tính hướng nội của nét văn hoá và cũng là chính sách nhân sự của Viettel.

Nét văn hoá của Viettel là sự kết hợp hài hoà hai phong cách ***hướng ngoại*** và ***hướng nội***, với đặc trưng là sự phối hợp của hai tính chất kiểu văn hoá tổ chức ***doanh nhân*** và ***chuyên nghiệp***. Đây là niềm tự hào mà mỗi nhân viên Viettel đều cảm nhận và tự giác tuân thủ thực hiện để xây dựng và phát triển truyền thống tốt đẹp, khẳng định một phong cách riêng, một văn hoá tổ chức của chính mình.

Nét văn hoá của Viettel đã được khẳng định và được đúc kết thành 8 giá trị cốt lõi của Viettel:

1. Thực tiễn là tiêu chuẩn để kiểm nghiệm chân lý.
2. Trưởng thành qua những thách thức và thất bại.
3. Thích ứng nhanh là sức mạnh cạnh tranh.
4. Sáng tạo là sức sống.
5. Tư duy hệ thống.
6. Kết hợp Đông – Tây.
7. Truyền thống và cách làm người lính.
8. Viettel là ngôi nhà chung.

### Thực tiễn là tiêu chuẩn để kiểm nghiệm chân lý

**Chúng ta nhận thức:**

* Lý thuyết màu xám, chỉ có cây đời là mãi xanh tươi. Lý luận để tổng kết thực tiễn rút ra kinh nghiệm, tiệm cận chân lý và dự đoán tương lai. Chúng ta cần có lý luận và dự đoán để dẫn dắt. Nhưng chỉ có thực tiễn mới khẳng định được những lý luận và dự đoán đó đúng hay sai.
* Chúng ta nhận thức và tiếp cận chân lý thông qua thực tiễn hoạt động.

**Chúng ta hành động:**

* Phương châm hành động của chúng ta “Dò đá qua sông” và liên tục điều chỉnh cho phù hợp với thực tiễn.
* Chúng ta đánh giá con người thông qua quá trình thực tiễn.

### Trưởng thành qua những thách thức và thất bại

**Chúng ta nhận thức:**

* Thách thức là chất kích thích. Khó khăn là lò luyện. “Vứt nó vào chỗ chết thì nó sẽ sống”.
* Chúng ta không sợ mắc sai lầm. Chúng ta chỉ sợ không dám nhìn thẳng vào sai lầm để tìm cách sửa. Sai lầm là không thể tránh khỏi trong quá trình tiến tới mỗi thành công. Sai lầm tạo ra cơ hội cho sự phát triển tiếp theo.

**Chúng ta hành động:**

* Chúng ta là những người dám thất bại. Chúng ta động viên những ai thất bại. Chúng ta tìm trong thất bại những lỗi sai của hệ thống để điều chỉnh. Chúng ta không cho phép tận dụng sai lầm của người khác để đánh đổ người đó. Chúng ta sẽ không lặp lại những lỗi lầm cũ.
* Chúng ta phê bình thẳng thắn và xây dựng ngay từ khi sự việc còn nhỏ. Chúng ta thực sự cầu thị, cầu sự tiến bộ.

### Thích ứng nhanh là sức mạnh cạnh tranh

**Chúng ta nhận thức:**

* Cái duy nhất không thay đổi đó là sự thay đổi. Trong môi trường cạnh tranh sự thay đổi diễn ra từng ngày, từng giờ. Nếu nhận thức được sự tất yếu của thay đổi thì chúng ta sẽ chấp nhận thay đổi một cách dễ dàng hơn.
* Mỗi giai đoạn, mỗi qui mô cần một chiến lược, một cơ cấu mới phù hợp. Sức mạnh ngày hôm nay không phải là tiền, là qui mô mà là khả năng thay đổi nhanh, thích ứng nhanh.
* Cải cách là động lực cho sự phát triển.

**Chúng ta hành động:**

* Tự nhận thức để thay đổi. Thường xuyên thay đổi để thích ứng với môi trường thay đổi. Chúng ta sẽ biến thay đổi trở thành bình thường như không khí thở vậy.
* Liên tục tư duy để điều chỉnh chiến lược và cơ cấu lại tổ chức cho phù hợp.

### Sáng tạo là sức sống

**Chúng ta nhận thức:**

* Sáng tạo tạo ra sự khác biệt. Không có sự khác biệt tức là chết. Chúng ta hiện thực hoá những ý tưởng sáng tạo không chỉ của riêng chúng ta mà của cả khách hàng.

**Chúng ta hành động:**

* Suy nghĩ không cũ về những gì không mới. Chúng ta trân trọng và tôn vinh từ những ý tưởng nhỏ nhất.
* Chúng ta xây dựng một môi trường khuyến khích sáng tạo để mỗi người Viettel hàng ngày có thể sáng tạo.
* Chúng ta duy trì Ngày hội ý tưởng Viettel.

### Tư duy hệ thống

**Chúng ta nhận thức:**

* Môi trường kinh doanh ngày càng phức tạp. Tư duy hệ thống là nghệ thuật để đơn giản hoá cái phức tạp.
* Một tổ chức phải có tư tưởng, tầm nhìn chiến lược, lý luận dẫn dắt và hệ thống làm nền tảng. Một hệ thống muốn phát triển nhanh về qui mô thì phải chuyên nghiệp hoá.
* Một hệ thống tốt thì con người bình thường có thể tốt lên. Hệ thống tự nó vận hành phải giải quyết được trên 70% công việc. Nhưng chúng ta cũng không để tính hệ thống làm triệt tiêu vai trò các cá nhân.

**Chúng ta hành động:**

* Chúng ta xây dựng hệ thống lý luận cho các chiến lược, giải pháp, bước đi và phương châm hành động của mình.
* Chúng ta vận dụng qui trình 5 bước để giải quyết vấn đề: Chỉ ra vấn đề ⇨ Tìm nguyên nhân ⇨ Tìm giải pháp ⇨ Tổ chức thực hiện ⇨ Kiểm tra và đánh giá thực hiện.
* Người Viettel phải hiểu vấn đề đến gốc: Làm được là 40% ⇨ Nói được cho người khác hiểu là 30% ⇨ Viết thành tài liệu cho người đến sau sử dụng là 30% còn lại.
* Chúng ta sáng tạo theo quy trình: Ăn ⇨ Tiêu hoá ⇨ Sáng tạo.

### Kết hợp Đông – Tây

**Chúng ta nhận thức:**

* Có hai nền văn hoá, hai cách tư duy, hai cách hành động lớn nhất của văn minh nhân loại. Mỗi cái có cái hay riêng có thể phát huy hiệu quả cao trong từng tình huống cụ thể. Vậy tại sao chúng ta không vận dụng cả hai cách đó?
* Kết hợp Đông – Tây cũng có nghĩa là luôn nhìn thấy hai mặt của một vấn đề. Kết hợp không có nghĩa là pha trộn.

**Chúng ta hành động**

* Chúng ta kết hợp tư duy trực quan với tư duy phân tích và hệ thống.
* Chúng ta kết hợp sự ổn định và cải cách.
* Chúng ta kết hợp cân bằng và động lực cá nhân.

### Truyền thống và cách làm người lính

**Chúng ta nhận thức:**

* Viettel có cội nguồn từ Quân đội. Chúng ta tự hào với cội nguồn đó.
* Một trong những sự khác biệt tạo nên sức mạnh Viettel là truyền thống và cách làm quân đội.

**Chúng ta hành động:**

* Truyền thống: Kỷ luật, Đoàn kết, Chấp nhận gian khổ, Quyết tâm vượt khó khăn, Gắn bó máu thịt.
* Cách làm: Quyết đoán, Nhanh, Triệt để.

### Viettel là ngôi nhà chung

**Chúng ta nhận thức:**

* Viettel là ngôi nhà thứ hai mà mỗi chúng ta sống và làm việc ở đó. Mỗi người Viettel phải trung thành với sự nghiệp của Công ty. Chúng ta phải hạnh phúc trong ngôi nhà này thì chúng ta mới làm cho khách hàng của mình hạnh phúc được.
* Mỗi chúng ta là một cá thể riêng biệt, nhưng chúng ta cùng chung sống trong một ngôi nhà chung Viettel - ngôi nhà mà chúng ta cùng chung tay xây dựng. Đoàn kết và nhân hoà trong ngôi nhà ấy là tiền đề cho sự phát triển.

**Chúng ta hành động:**

* Chúng ta tôn trọng nhau như những cá thể riêng biệt, nhạy cảm với các nhu cầu của nhân viên. Chúng ta lấy làm việc nhóm để phát triển các cá nhân. Các cá nhân, các đơn vị phối hợp với nhau như các bộ phận trong một cơ thể.
* Mỗi người chúng ta qua các thế hệ sẽ góp những viên gạch để xây lên ngôi nhà ấy.
* Chúng ta lao động để xây dựng đất nước, Viettel phát triển, nhưng chúng ta phải được hưởng xứng đáng từ những thành quả lao động đó. Nhưng chúng ta luôn đặt lợi ích của đất nước của doanh nghiệp lên trên lợi ích cá nhân.

# BÁO CÁO CHƯƠNG TRÌNH TRAINING

## Tìm hiểu về C trong môi trường Linux

### Hàm và cấu trúc chương trình

#### External

* Khai báo biến toàn cục, được chia sẻ giữa các source file.
* Ưu điểm: Các source file chia sẻ dữ liệu với nhau dễ dàng.
* Nhược điểm: Sử dụng biến external nhiều có thể làm rối cấu trúc code.

#### Static

* Biến static: Chỉ được sử dụng nội bộ trong file mà biến được khai báo.
* Hàm static: Hàm static chỉ được sử dụng trong file mà hàm được khai báo.

#### Inline

Gợi ý compiler thực hiện chèn code của hàm đó tại địa chỉ mà nó được gọi.

* Ưu điểm: Giảm thời gian thực thi hàm.
* Nhược điểm: Làm tăng kích thước của chương trình đối với các hàm inline có kích thước lớn. Các hàm inline đệ quy, kích thước lớn hoặc bên trong hàm có lệnh goto, switch hoặc hàm chứa biến static sẽ không được compiler xem đó là hàm inline mà biên dịch như hàm bình thường.

#### MACRO

* Là phần định nghĩa viết trong preprocessor.
* Ưu điểm: Do được định nghĩa ở preprocessor nên MACRO có thể dùng đẻ tránh compiler include trùng các thư viện hoặc có thể dùng để debug dễ dàng hơn mà không ảnh hưởng tới khi build release.
* Nhược điểm: MACRO không kiểm tra đối số vào.

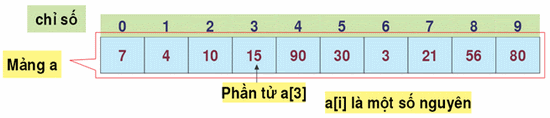
#### Phân biệt hàm inline và MACRO

|  |  |
| --- | --- |
| **Hàm inline** | **MACRO** |
| Kiểm tra đối số vào | Không kiểm tra đối số vào |
| Gợi ý compiler chèn đoạn code inline vào nơi gọi hàm, compiler sẽ quyết định có nên compile hàm với inline không | Chắc chắn chèn đoạn code vào nơi xuất hiện MACRO |

### Con trỏ và mảng

#### Mảng

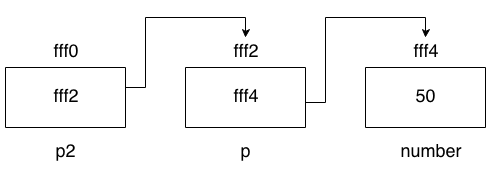
* Mảng là tập hợp các phần tử có cùng kiểu dữ liệu.
* Truy cập các phần tử của mảng thông qua chỉ số của mảng.



*Hình 2‑1: Mảng 10 phần tử*

#### Con trỏ

* Con trỏ là một biến dùng để chứa địa chỉ trên bộ nhớ nơi dùng để lưu trữ giá trị của biến.
* Dùng con trỏ: trỏ đến bộ nhớ động, truyền giá trị từ hàm ra ngoài thông qua đối số, để truy cập biến, mảng, dùng trong linked list, binary tree...



*Hình 2‑2: Ví dụ về con trỏ*

* Ưu điểm: Hữu dụng trong việc cấp phát bộ nhớ, thao tác với các kiểu dữ liệu như chuỗi, linked list, binary tree…
* Nhược điểm: Dùng con trỏ không cẩn thận có thể sinh ra nhiều lỗi phức tạp bởi vì truy cập đến vùng nhớ không cho phép, sai dữ liệu, gây tổn hao bộ nhớ vì có thể xử lý code không tốt.

#### So sánh con trỏ và mảng

|  |  |
| --- | --- |
| **Con trỏ** | **Mảng** |
| Có thể trỏ đến các biến, mảng, hàm. | Có thể được dùng như con trỏ nếu kích thước mảng = 0. |
| Chưa có giá trị khi khai báo | Được khởi tạo giá trị trong mảng khi khai báo |
| Trả về kích thước bản thân con trỏ (chứa địa chỉ) đối với toán tử sizeof() | Trả về kích thước mảng (số phần tử) đối với toán tử sizeof() |

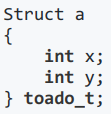
#### Con trỏ hàm

* Dùng để lưu trữ địa chỉ của một hàm.
* Dùng để gọi callback hàm.

### Dữ liệu có cấu trúc

#### Struct

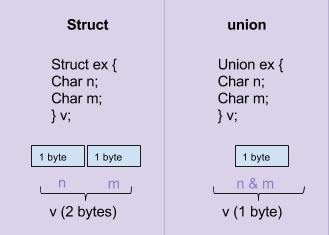
* Khai báo 1 kiểu dữ liệu.
* Kiểu dữ liệu được tạo ra từ các kiểu dữ liệu khác.



*Hình 2‑3: Ví dụ về kiểu struct*

#### Union

* Là kiểu dữ liệu phức hợp được tạo ra từ các kiểu dữ liệu khác.
* Các thành phần dữ liểu của union luôn có cùng địa chỉ.
* Kích thước của union bằng kích thước của thành phần dữ liệu lớn nhất của union.



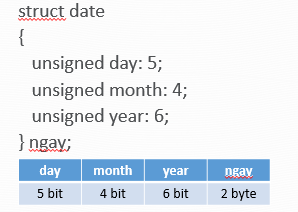
*Hình 2‑4: So sánh giữa Struct và Union*

#### Typedef

* Định nghĩa tên một kiểu dữ liệu mới.
* Thường dùng để định nghĩa các kiểu dữ liệu như struct, union, enum,...

#### Bit-Field

* Là một kiểu khai báo struct theo bit hoặc nhóm bit, giới hạn số bit 1 thành phần trong struct.
* Dùng để tiết kiệm bộ nhớ.



*Hình 2‑5: Ví dụ về struct dạng Bit-Field*

### Input và Output trong C

#### Standard In/Out

* Standard Input: Đầu vào chuẩn của hệ điêu hành.
* Standard Output: Đầu ra chuẩn của hệ điều hành.
* Để đọc một kí tự từ stdin, ta sử dụng hàm getchar(). Hàm sẽ trả về 1 kí tự hoặc EOF nếu nó đọc đến cuối tập tin.
* Để đưa một kí tự ra stdout, ta sử dụng hàm putchar(). Hàm sẽ trả về kí tự đã ghi hoặc EOF nếu có lỗi xảy ra.
* Cả hai hàm trên đều là các macro trong file stdio.h
* Có thể read/write stdin hoặc stdout thông qua file descriptor trong linux.

#### Printf Format

* Hàm printf dùng để ghi dữ liệu ra stdout theo một chuỗi định dạng.
* Hàm printf trả về số kí tự đã in ra được
* Chuỗi định dạng xuất nhập có dạng như sau:

%[flag][width][.prec][F:N:h:I:L]type

Trong đó:

* Type là kiểu của tham số, có thể là số nguyên hay số thực, chuỗi...
* Flag dùng để canh chỉnh lề khi xuất liệu.
* Width dùng để định kích thước in ra.
* Pre dùng để định số lẻ sau dấu chấm thập phân cần in ra.
* [F:N:h:I:L] là các bổ túc cho phần Type.

#### Scanf Format

* Hàm scanf dùng để nhập dữ liệu từ stdin theo một chuỗi định dạng
* Hàm scanf chỉ nhận số kí tự khi gặp kí tự trắng đầu tiên hoặc khi đủ width trong chuỗi định dạng, sau đó biến đổi và lưu vào biến
* Chuỗi định dạng điều khiển có dạng như sau:

*%[width]type*

* Width xác định số kí tự tối đa sẽ được nhập vào
* Type xác định kiểu giá trị nhập vào
* Kí tự trắng có thể là kí tự tab \t, enter \n, khoảng trắng ‘ ‘
* Nếu có kí tự trắng trong chuỗi định dạng thì phải nhập kí tự trắng đó trước khi nhập kí tự trắng khác mới có thể thoát khỏi hàm scanf

#### File Access

* File là một kiểu đối tượng mà thông qua nó có thể thao tác với dữ liệu trên máy tính.
* Các hàm thao tác với file thông qua kiểu dữ liệu FILE được định nghĩa trong thư viện stdio.h
* Hàm fopen: mở/tạo 1 file trong đường dẫn, mode xác định kiểu truy cập file.

*FILE \*fopen(const char \*file, const char \*mode)*

* Hàm fclose. Nếu đóng file thành công thì trả về giá trị 0, ngược lại trả về EOF.

*int fclose(FILE \*file);*

* Ghi dữ liệu vào file thông qua các lệnh fputc, fputs, fprintf.

*int fprintf(FILE \*f, const char \*format,…)*

* Đọc dữ liệu từ file thông qua các lệnh fgetc, fgets, fscanf

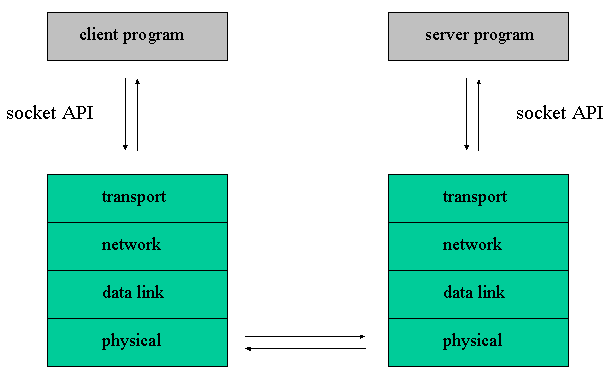
*int fscanf(FILE \*f, const char \*format,…)*

## *Tìm hiểu về socket*

### Định nghĩa Socket

#### Khái niệm

* Socket là một API liên lạc giữa các process trên nhiều/1 máy tính dưới dạng các hàm hệ thống theo ngôn ngữ C.
* Có hai loại socket là: STREAM SOCKET và DATAGRAM SOCKET



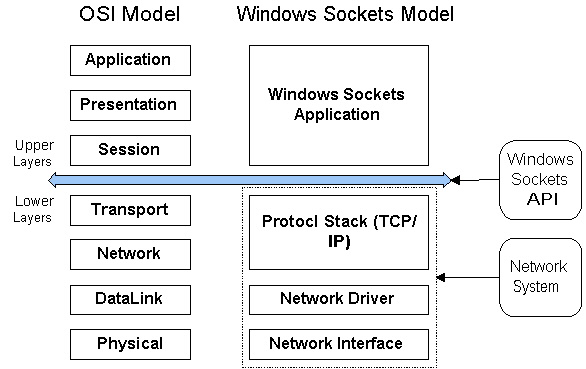
*Hình 2‑6: Mô hình socket*

#### Các thành phần

* Họ giao thức (family)
* AF\_INET address family: Cho phép các process trong cùng hệ thống hay khác hệ thống giao tiếp với nhau. Bao gồm địa chỉ ip và cổng port
* AF\_INET6 address family: Tương tự như AF\_INET nhưng phục vụ cho Internet Protocol ver6.
* AF\_UNIX address family: Chỉ cho phép các process trong cùng hệ thống giao tiếp với nhau.
* Loại (Type)
* STREAM SOCKET
* DATAGRAM SOCKET
* Giao thức (Protocol): Có thể chỉ định protocol cho socket, nếu mặc định 0 chương trình sẽ tìm protocol thích hợp.
* Local Socket Address: Địa chỉ của socket của process cục bộ.
* Remote Socket Address: Địa chỉ của socket của process từ xa.

#### Vị trí trong mô hình OSI

Socket nằm ở giữa tầng Transport và tầng Session trong mô hình OSI.

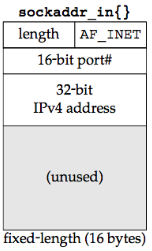
**

*Hình 2‑7: Vị trí của socket trong mô hình OSI*

### Cấu trúc địa chỉ của Socket

Địa chỉ của một Socket Ipv4 bao gồm:

* Độ dài cấu trúc (*length*)
* Họ giao thức (*sin\_family = AF\_INET*)
* Port
* Địa chỉ IPv4 (*IPv4 addr*)
* Phần không dùng đến (*unused*)



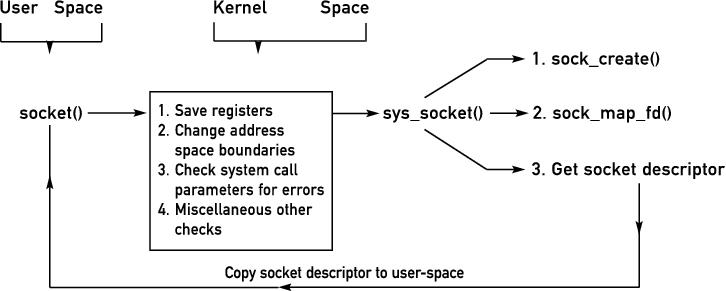
*Hình 2‑8: Cấu trúc địa chỉ Socket*

### Các hàm sử dụng trong socket

#### Định nghĩa các hàm

Socket sử dụng các hàm để khởi tạo, hủy, chuyển đổi địa chỉ, ghi và đọc dữ liệu:

* Hàm tạo socket: *int socket(int domain, int type, int protocol);*
  + Tạo socket, tham số domain(ipv4-v6-UNIX), type(stream hoặc dgram), protocol(0 để socket chọn protocol thích hợp sử dụng), hàm trả về số tham chiếu file descriptor.

**No table of figures entries found.**

*Hình 2‑9: Kernel thực hiện hàm socket*

* Hàm hủy socket: *int close(int fd);*
  + Đóng file descriptor của socket đa tạo trước đó.
* Hàm khóa socket: *int bind(int sockfd, struct sockaddr \*my\_addr, int addrlen);*
  + Gán địa chỉ, cổng, giao thức vào socket.
  + Sao chép my\_addr vào kernel space.
* Hàm lắng nghe: *int listen(int sockfd,int backlog);*
  + (only TCP) Lắng nghe client, chuyển sang socket thụ động, chờ chấp nhận kết nối từ client, backlog là số lượng tối đa server có thể accept (tùy vào hệ điều hành backlog khác nhau thì số lượng tối đa client khác)
* Hàm kết nối: *int connect(int sockfd, struct sockaddr \*serv\_addr, int addrlen);*
  + (only TCP) Thiết lập kết nối tới port và ip định sẵn.
* Hàm chấp nhận kết nối: *int accept (int sockfd, struct sockaddr \*cliaddr, socklen\_t \*addrlen);*
  + (only TCP) chấp nhận kết nối từ client.
  + Cấp 1 vùng nhớ socket khác(fd khác) dưới kernel space, các giao tiếp connect sẽ thông qua socket mới này.
* Một số hàm chuyển đổi địa chỉ: *inet\_aton, inet\_addr, inet\_ntoa, inet\_pton, inet\_ntop, sock\_ntop:*
* **Hàm inet\_aton:** *int inet\_aton(const char \*cp, struct in\_addr \*inp);*

Chuyển đổi địa chỉ của Host Internet cp từ địa chỉ IPv4 dạng dotted-decimal (string) sang dạng địa chỉ số(32bit) và lưu vào trong struct mà inp trỏ tới.

Trả về giá trị khác 0 nếu địa chỉ là hợp lệ, ngược lại trả về 0.

* **Hàm inet\_addr:** *in\_addr\_t inet\_addr(const char \*cp);*

Tương tự inet\_aton, trả về địa chỉ số(32bit) từ đối số cp, trả về -1 nếu địa chỉ không hợp lệ. Tuy nhiên, -1 lại là một địa chỉ hợp lệ (255.255.255.255) trong network -> tránh dùng, nên sử dụng inet\_aton.

* **Hàm inet\_ntoa:** *char \*inet\_ntoa(struct in\_addr in);*

Ngược lại với hàm inet\_aton, hàm này chuyển địa chỉ IP từ dạng số sang dạng dotted-decimal (string). Hàm trả về một chuỗi chứa địa chỉ.

* **Hàm inet\_pton:** *int inet\_pton(int af, const char \*src, void \*dst);*

Chuyển số ip(v6 hoặc v4 dựa vào đối số af) dotted-decimal (string) sang dạng số binary.

Hàm trả về 1 nếu thành công, 0 nếu src chứa địa chỉ sai, -1 nếu af sai.

* **Hàm inet\_ntop:** *const char \*inet\_ntop(int af, const void \*src, char \*dst, socklen\_t size);*

Chuyển số binary sang ip(v6 hoặc v4 dựa vào đối số af) dotted-decimal (string).

Hàm trả về một pointer đến dst nếu thành công nếu sai trả về NULL.

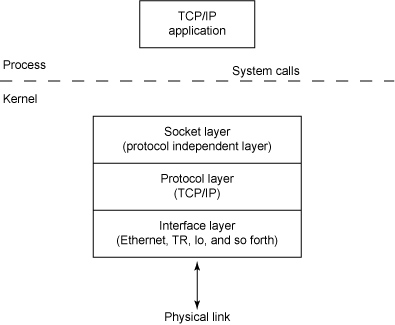
* **Hàm sock\_ntop:** *char \*sock\_ntop(const struct sockaddr \*sockaddr, socklen\_t addrlen);*

Nhận vào một địa chỉ dạng struct sockaddr và trả về pointer trỏ đến địa chỉ dạng dotted-decimal (string). Nếu hàm không thành công, trả về NULL.

* Các hàm chuyển đổi địa chỉ nằm trong thư viện ***arpa/inet.h***
* Hàm đọc dữ liệu socket: *ssize\_t read(int filedes, void \*buf, size\_t nbytes);*
  + Đọc nbyte từ file descriptor vào buf.
* Hàm ghi dữ liệu socket: *ssize\_t write(int filedes, void \*buf, size\_t nbytes);*
  + Ghi nbyte từ buf vào file descriptor.
* Ngoài ra còn một số hàm đọc/ghi khác như:
  + int sendto(int sockfd, const void \*msg, int len, unsigned int flags, const struct sockaddr \*to, int tolen);
    - Gửi data tới 1 socket(socket có thể chưa connect, thường dùng với kiểu UDP)
  + int recvfrom(int sockfd, void \*buf, int len, unsigned int flags struct sockaddr \*from, int \*fromlen);
    - Gửi nhận data từ 1 socket(socket có thể chưa connect, thường dùng với kiểu UDP)

#### Quan hệ giữa socket với kernel linux

Trong hệ điều hành linux, Userspace và Kernelspace được phân tách rõ ràng. User giao tiếp và sử dụng các tính năng, bộ nhớ của kernel thông qua system call.



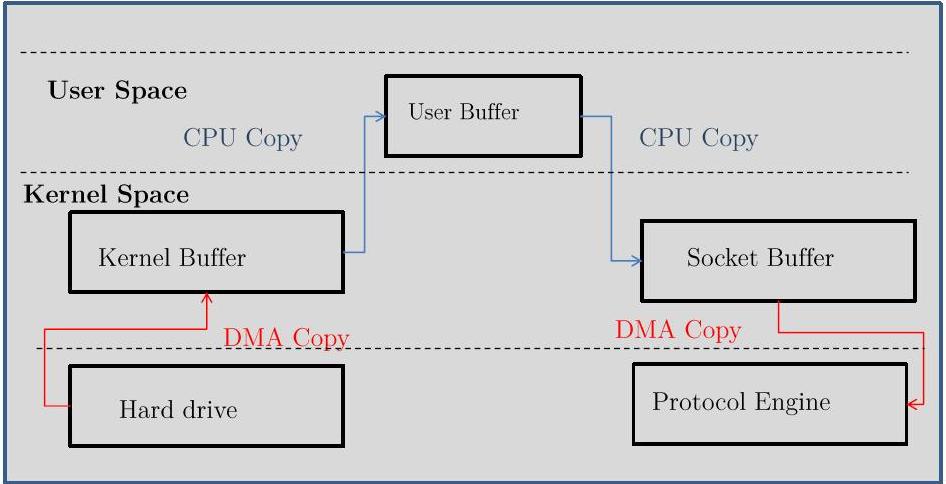
*Hình 2‑10: Quan hệ giữa socket và kernel linux*

Khi một tiến trình gọi một system call, CPU sẽ có thể truy cập các vùng nhớ và thực hiện các chỉ lệnh của kernel.

Mỗi system call được kernel định danh bằng một số duy nhất. Tiến trình trên tầng user không cần biết đến các số này, thay vào đó, nó gọi một system call bằng tên hàm.

#### Thực hiện các hàm đọc/ghi dữ liệu

* System call read đọc từ kernel buffer của file descriptor(socket hoặc input…) và copy data lên userspace, write copy data từ userspace đến kernel buffer của file descriptor socket và xử lý lớp protocol để truyền đi.

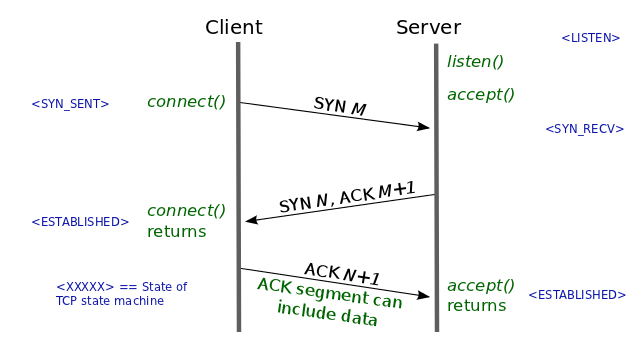


*Hình 2‑11: Quá trình thực hiện các hàm read/write dữ liệu của socket*

### TCP Socket

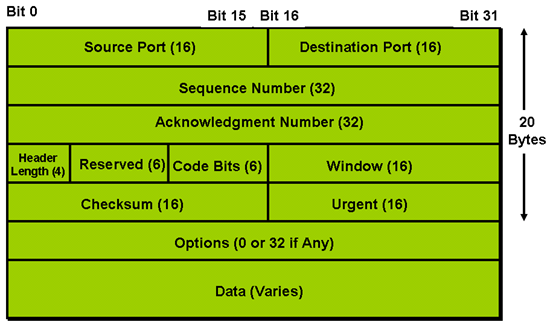
#### Quá trình bắt tay

* Client gửi gói SYN(IP TCP header, TCP option) để khởi tạo quá trình bắt tay.
* Server nhận request từ Client, sau đó gửi lại gói SYN-ACK để xác nhận.
* Client gửi lại gói ACK xác nhận với server -> Thiết lập kết nối.



*Hình 2‑12: Quá trình bắt tay 3 bước của TCP*

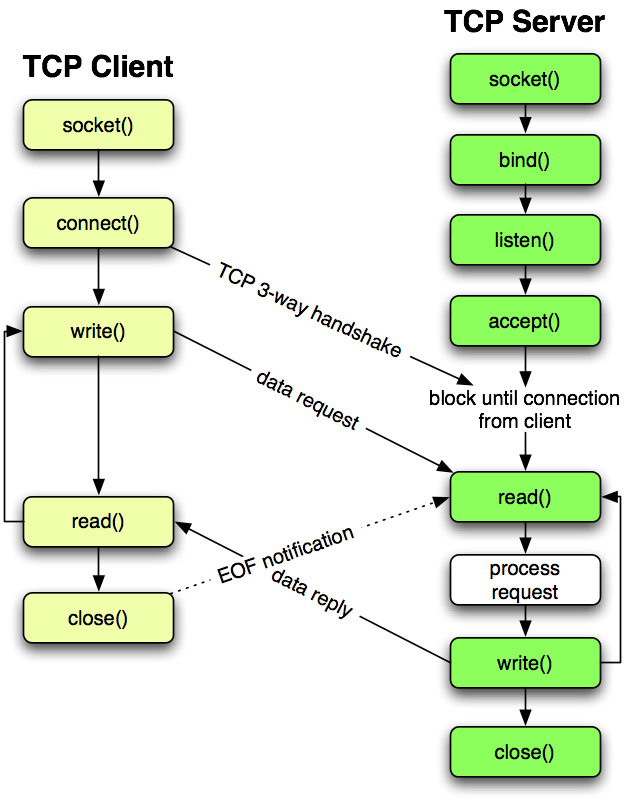
#### Gói tin TCP



*Hình 2‑13: Cấu trúc gói tin TCP*

* Sequence Number: Dùng để đánh dấu thứ tự gói tin. Giúp nhận gói tin theo đúng thứ tự
* Acknowledge Number: Thông báo cho bên gửi biết đã nhận thành công
* Window: Cơ chế cửa sổ gửi và nhận để điều khiển luồng.
* Header Length(variable length) chỉ định bao nhiêu word(32bit) trong TCP header. Vì Options field không cố định nên header length cũng vậy.
* Reserved field (6 bits) mặc định 0.
* Flags field (6 bits) gồm các flag sau::  
  URG—urgent data.  
  ACK—acknowledgement number valid.  
  PSH—Báo data nên được chuyển đi càng sớm càng tốt.  
  RST—Resets the connection.  
  SYN—Synchronizes sequence numbers.  
  FIN—Báo bên gửi đã gửi xong data.
* Checksum field (16 bits) để xác nhận gói tin tcp không bị corrupt.
* Urgent pointer field (16 bits) chỉ đến byte data khẩn cấp đầu tiên trong packet
* Options field (variable length) chỉ định 1 số option của tcp.
* Data field (variable length) chứa data.

#### Qúa trình thiết lập TCP socket



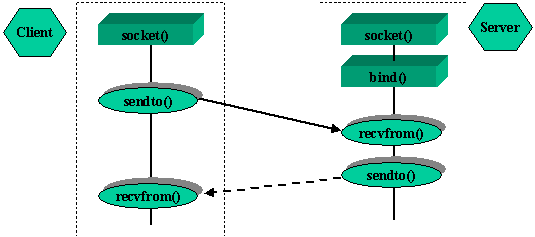
*Hình 2‑14: Quá trình thiết lập TCP socket*

* Server và client gọi hàm tạo SOCK\_STREAM socket -> bind để gánIP address, PORT, AF\_FAMILY vào socket đã tạo -> listen chờ có request connect từ client.

### UDP Socket

#### Quá trình gửi và nhận dữ liệu

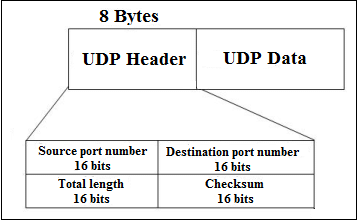
* Quá trình gửi và nhận dữ liệu đơn giản hơn so với TCP
* Không cần phải thiết lập kết nối giữa server và client



*Hình 2‑15: Quá trình gửi và nhận dữ liệu UDP*

* Server và Client gọi hàm tạo SOCK\_DGRAM socket.
* Server gọi hàm bind để gán IP address, PORT, AF\_FAMILY vào socket đã tạo.
* Lúc này, Client và server có thể trao đổi dữ liệu với nhau.

#### Cấu trúc gói tin UDP



*Hình 2‑16: Cấu trúc gói tin UDP*

* Cấu trúc gói tin đơn giản hơn TCP, chỉ gồm 8 bytes header.
* Header UDP chỉ bao gồm: *Source Port, Destination Port, Length, Checksum*

#### So sánh TCP Socket và UDP Socket

|  |  |
| --- | --- |
| **TCP** | **UDP** |
| Có thiết lập kết nối | Không thiết lập kết nối |
| Truyền dữ liệu chính xác, không lỗi, trùng lặp hay mất gói tin | Chấp nhận mất mát, lỗi hoặc trùng dữ liệu |
| Có cơ chế kiểm soát luồng, tắc nghẽn | Không kiểm soát luồng, tắc nghẽn |
| Thích hợp cho HTTP, FTP,... | Thích hợp cho DNS, DHCP, game, phát thanh trực tuyến,... |

## Tìm hiểu về cơ chế giao tiếp giữa các process

### Process

#### Khái niệm

Process là một tiến trình điều khiển đoạn mã lệnh có riêng một không gian địa chỉ, có ngăn xếp stack riêng rẽ, có bảng chứa các thông số mô tả file được mở cùng tiến trình và đặc biệt có một định danh PID (Process Identify) duy nhất trong toàn bộ hệ thống vào thời điểm tiến trình đang chạy.

#### Các đặc điểm của process

* Mỗi process được quản lí bởi một PCB (process control block) với các thông tin đặc trưng như id, process con, process cha, trạng thái, độ ưu tiên, signal...
* Mỗi process khi khởi chạy được cấp phát một vùng nhớ riêng.
* Các process được lập lịch chạy và được đồng bộ tiến trình bởi hệ điều hành.

Bởi vì các process có các vùng nhớ riêng rẽ và không chia sẻ cho nhau. Do đó, cần phải có cơ chế để cho mỗi process có thể kết nối và trao đổi dữ liệu với process khác.

### IPC

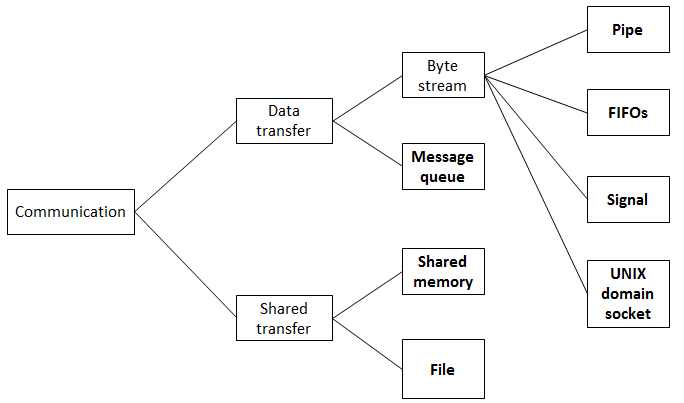
#### Khái niệm

Inter Process Communication (IPC) là một phương thức giúp các process trao đổi thông tin với nhau.

#### Các loại IPC

IPC có các loại như sau:

* Pipe: kênh dữ liệu một chiều. Data gửi đi theo kênh ghi sẽ được lưu vào bộ đệm của hệ điều hành đến khi nó được đọc tại kênh đọc. Do đó một process muốn có dữ liệu hai chiều gửi, nhận thì phải tạo hai pipe.
* FIFOs: giống như pipe. Khác ở chỗ là nó tồn tại như một device file trong file system, các process khác có thể truy cập và share data thông qua tên; khi tất cả I/O đã xong sau khi trao đổi dữ liệu giữa các process, FIFO vẫn được giữ lại trong file system để sử dụng về sau. Nó được xóa đi sau một khoảng thời gian không sử dụng.
* Unix domain socket: cung cấp một cơ chế triển khai các ứng dụng giao tiếp bằng kiến trúc client-server. Nó hỗ trợ cả hai loại kết nối stream và datagram, hai chiều. Sử dụng file system như không gian địa chỉ của nó. Process coi domain như một inode và nhiều process có thể kết nối chỉ với một socket.
* Message queue: có cơ chế là một process sẽ ghi gói tin lên một message queue và thoát. Một process khác có thể truy cập vào gói tin từ message queue đó tại một thời điểm khác. Nó có một ưu điểm so với pipe và FIFO là process gửi không cần phải đợi cho đến khi process nhận kết nối.
* Shared memory: cho phép các process có thể truy cập vào một vùng nhớ vật lí được chia sẻ.
* File: một bản ghi được lưu trên ổ đĩa hoặc một file server mà có thể được truy cập bởi nhiều process.



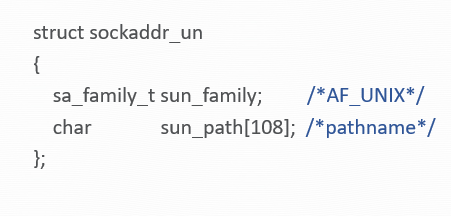
*Hình 2‑17: Các loại IPC*

#### IPC sử dụng Unix Domain Socket

* *Khái niệm:*

Unix Domain Socket hay còn gọi là IPC Socket là một socket được sử dụng để trao đổi dữ liệu giữa các process được thực hiện ở trên một host OS

* *Cấu trúc địa chỉ:*



*Hình 2‑18: Cấu trúc địa chỉ của Unix Domain Socket*

Trong đó, sun\_family phải là AF\_UNIX

Sun\_path chứa pathname trỏ tới một file đặc biệt dạng socket. File này được tạo ra trong lúc bind socket. Trong suốt quá trình giao tiếp, file này không có nội dung. Thành phần sun\_path phải ở dạng null-terminated, tức là kết thúc bằng kí tự null.

Một khi đã được tạo ra, file socket này sẽ tiếp tục tồn tại, thậm chí ngay cả khi server kết thúc. Nếu server sau đó khởi động lại, file này sẽ ngăn cản server thực hiện lệnh bind. Do đó, server nên unlink socket pathname trước khi thực hiện bind đến nó.

Unix Domain Socket biết rằng nó đang thực hiện trên cùng hệ thống nên nó có thể tránh được các bước của Internet Domain Socket như routing chẳng hạn. Do đó, Unix Domain Socket nhanh hơn Internet Socket trong cùng một host.

* *Phân loại*
* Unix Domain Stream client/server
* Unix Domain Datagram client/server
* *Các hàm sử dụng trong Unix Domain Socket*
* Hàm bind(): Gọi ra hàm unix\_bind() trong /net/unix/af\_unix.c bên trong kernel linux.

Hàm unix\_bind() sẽ kiểm tra địa chỉ có phù hợp với Unix Domain Socket hay không, tạo ra một file dạng socket dựa theo pathname trong địa chỉ,…

* Hàm listen(): Gọi ra hàm unix\_listen() bên trong /net/unix/af\_unix.c trong kernel linux

Hàm unix\_listen sẽ cập nhật các giá trị bên trong *struct sock* như: sock state, sk\_max\_ack\_backlog, struct pid,…

* Hàm accept(): Hàm accept sẽ tạo ra một socket mới, gọi ra hàm unix\_accept() trong /net/unix/af\_unix.c và lưu địa chỉ của socket connect tới vào tham số hàm.
* Hàm connect(): Hàm sẽ gọi ra hàm unix\_stream\_connect() trong /net/unix/af\_unix.c nếu socket là SOCK\_STREAM, gọi ra hàm unix\_dgram\_connect() trong /net/unix/af\_unix.c nếu socket là SOCK\_DGRAM.
* Các hàm send(), sendto(), sendmsg(): Các hàm này sẽ gọi ra hàm unix\_stream\_sendmsg() trong /net/unix/af\_unix.c nếu socket là SOCK\_STREAM. Gọi ra hàm unix\_dgram\_sendmsg() nếu socket là SOCK\_DGRAM. Cuối cùng, đều gọi ra hàm skb\_queue\_tail() để gửi dữ liệu vào queue.
* Các hàm recv(), recvfrom(), recvmsg(): Các hàm này sẽ gọi ra hàm unix\_stream\_recvmsg() trong /net/unix/af\_unix.c nếu socket là SOCK\_STREAM. Gọi ra hàm unix\_dgram\_recvmsg() trong /net/unix/af\_unix.c nếu socket là SOCK\_DGRAM. Cuối cùng đều gọi ra hàm skb\_dequeue() để lấy dữ liệu ra từ queue.
* *Ứng dụng*

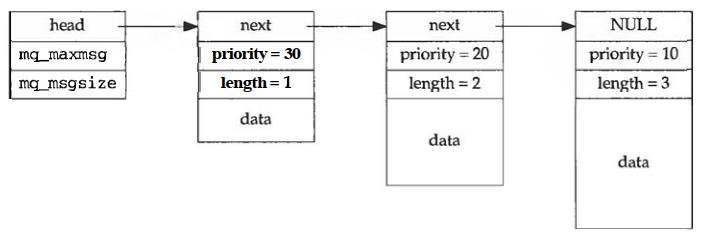
Unix Domain Socket được sử dụng rộng rãi trong quá trình giao tiếp và kết nối giữa các process.

Được ứng dụng trong xử lí đa luồng trong mô hình client/server. Với mỗi client kết nối tới, server sẽ tạo ra một tiến trình con xử lí riêng biệt cho client đó. UNIX domain socket là sự lựa chọn tốt cho những chương trình có các luồng sử dụng khác nhau, phù hợp với từng trường hợp cụ thể cho mỗi client.

UNIX domain socket hỗ trợ chuyển file descriptor theo dạng message giữa các process, điều mà các loại IPC khác không có. Việc có thể trao đổi file descriptor giữa các process giúp quản lí, đóng gói thông tin tốt hơn.

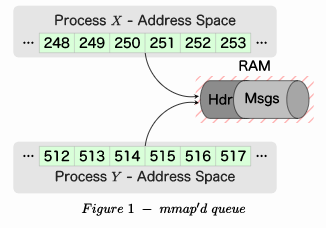
#### IPC sử dụng Message Queue

* *Khái niệm*
* Là một phương thức IPC sử dụng message để trao đổi thông tin giữa các process. Các message này được tổ chức dưới dạng Queue.
* Các tiến trình có thể đẩy và lấy data từ queue.



*Hình 2‑19: các gói message trong message queue*

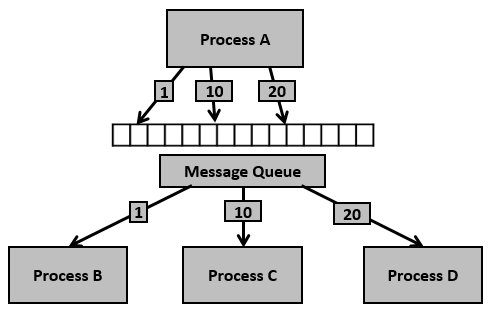
* *Đặc điểm:*
* Các process giao tiếp với nhau bằng cách gửi data vào message queue và process khác sẽ lấy data đó từ queue.
  + Không có cơ chế lưu thông tin process gửi trong message queue.
  + Giới hạn kích thước cho mỗi message queue tùy thuộc hệ điều hành (trong linux là 8192 bytes). Và số lượng message tùy theo tài nguyên hệ thống (trong linux là 16384).
  + Message queue data được cấp phát bởi hàm mmap và lưu trữ trên RAM, dữ liệu có thể được truy cập bằng cách tham chiếu đến vùng nhớ ảo.
  + Các message được gửi đi và lấy ra khỏi hàng đợi theo thứ tự và mức ưu tiên message.



*Hình 2‑20: Lưu trữ message queue*

* *Quá trình hoạt động:*

Cơ chế: một process sẽ đẩy gói tin lên một message queue. Một process khác có thể lấy gói tin từ message queue đó tại một thời điểm khác.

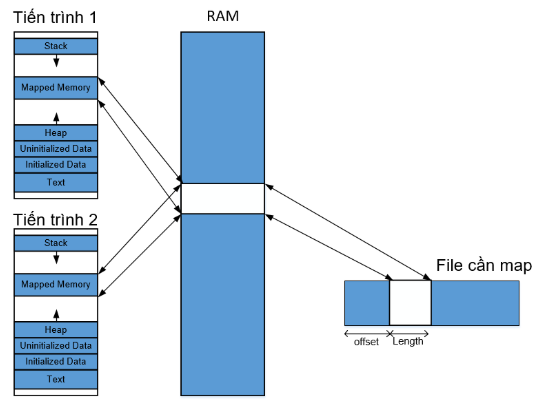


*Hình 2‑21: Cơ chế hoạt động của Message Queue*

* + Đầu tiên, phải tạo/mở một message queue bằng mq\_open
  + Bên gửi sẽ đóng gói message cần gửi và đẩy message đó vào message queue bằng hàm mq\_send().
  + Bên nhận mở message queue đó và lấy message đầu tiên trong queue với priority <= priority yêu cầu bằng hàm mq\_receiver()
* *Ưu điểm và ứng dụng:*
  + Bên gửi có thể gửi message vào một thời điểm và bên nhận message có thể truy cập vào message từ message queue đó tại một thời điểm khác.
  + Nếu process gửi hoặc nhận bị kết thúc không chủ ý thì dữ liệu trong message queue vẫn tồn tại và giữ nguyên. Có thể khôi phục lại process và tiếp tục nhận thông tin trong queue mà không bị mất.
  + Dữ liệu được gửi vào và lấy ra tuần tự, tránh bị lộn xộn.
  + Cung cấp cơ chế đơn giản, dễ sử dụng, có notify.
* *Nhược điểm*:
  + Bị giới hạn kích thước message gửi đi.
  + Chỉ phù hợp cho việc giao tiếp và trao đổi các khối dữ liệu nhỏ.
  + Với system V, tốc độ trao đổi dữ liệu không nhanh do mỗi lần đọc/ghi kernel phải thực hiện copy RAM vào kernelspace, sau đó copy từ kernelspace sang userspace. POSIX đã khắc phục điều này.
  + Khá phức tạp khi quản lí và xử lí đa luồng hỗ trợ nhiều bên cùng gửi nhận. Do đó, có thể sẽ phải sử dụng nhiều message queue.

#### IPC sử dụng Shared Memory

* *Khái niệm*
* Là một phương thức IPC cho phép các process truy cập sử dụng vùng nhớ chung.
* *Đặc điểm:*
* Quá trình truyền nhận dữ liệu không cần qua kernel.
* Vùng nhớ chia sẻ có thể tùy biến tùy vào tài nguyên của hệ thống.
* *Quá trình hoạt động:*



*Hình 2‑22: Cơ chế hoạt động của Shared Memory*

Cơ chế hoạt động của shared memory:

* 1 process tạo 1 vùng shared memory, sau đó các process khác có thể truy cập chung vào vùng nhớ chung này.
* *Ưu điểm và ứng dụng:*
* Có tốc độ nhanh. Thường được sử dụng để tối ưu hệ thống hay trong các hệ thống thời gian thực.
* Cho phép các process khác truy cập trực tiếp và ngay lập tức đến khối dữ liệu trong vùng nhớ đó.
* Do chia sẻ chung vùng nhớ nên sẽ dễ dàng hơn trong việc các process chạy đồng bộ và cách chia sẻ dữ liệu cho nhau.
* *Nhược điểm:*
* Vùng nhớ chia sẻ có thể dễ dẫn tới race condition, deadlock là treo chương trình. Do đó cần phải có các phương pháp đồng bộ giữa các process. Các phương pháp này khá phức tạp, đặc biệt trong quản lí đa luồng. Đây chính là lí do lớn nhất cho việc shared memory không được sử dụng rộng rãi.
* Việc yêu cầu hệ điều hành cấp phát vùng nhớ mà quên giải phóng có thể dẫn tới thiếu bộ nhớ, mất dữ liệu.

## Tìm hiểu về Posix Thread

### Định nghĩa Thread

#### Khái niệm

Thread là một đơn vị nhỏ nhất của 1 chương trình. Thread có thể được quản lí bởi scheduler, là một phần của hệ điều hành.

Rất nhiều thread có thể tồn tại trong một process, thực thi đồng thời và có thể chia sẻ chung tài nguyên được cấp phát.

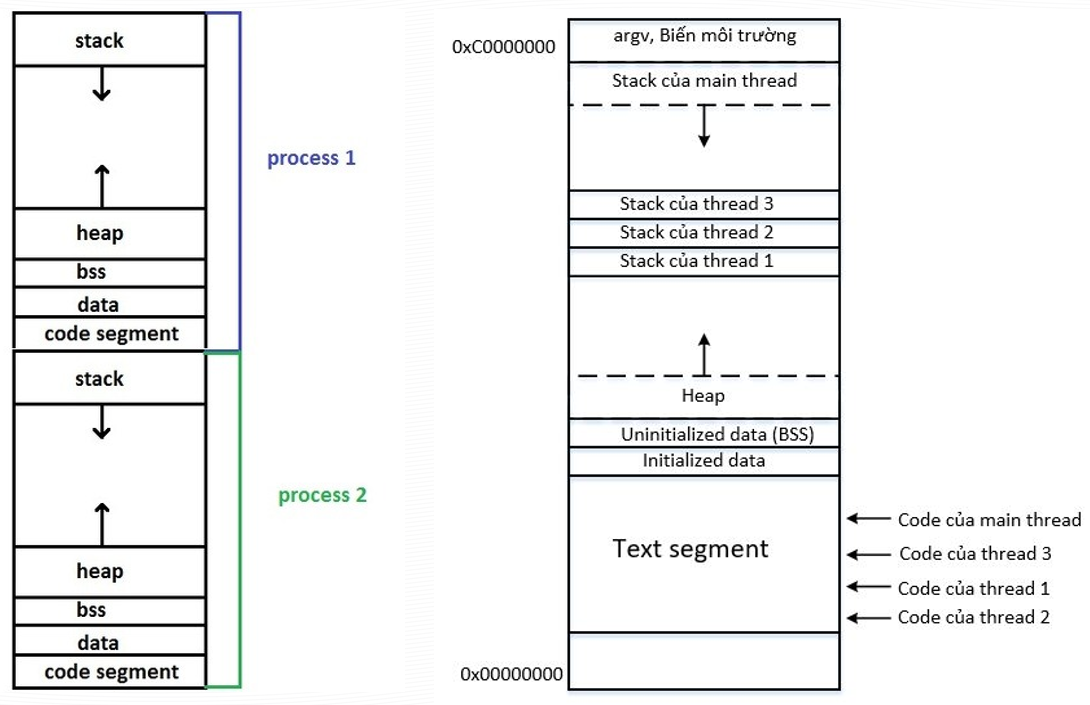
#### Posix Thread

Posix là một chuẩn của IEEE định nghĩa các dịch vụ trong hệ điều hành.

Các chương trình tuân theo Posix có thể dễ dàng chuyển từ hệ điều hành này sang hệ điều hành khác. Linux là một hệ điều hành theo chuẩn Posix.

Posix Thread là một API được định nghĩa bởi POSIX. 1c, Thread Extension (IEEE Std 1003. 1c – 1995) để thao tác với thread.

### So sánh Thread và Process



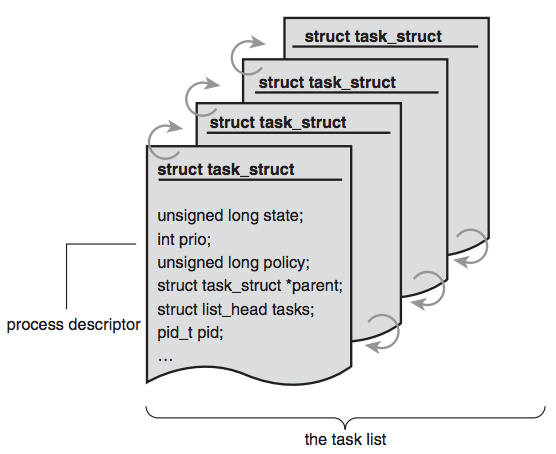
*Hình 2‑23: Vùng nhớ process và thread*

|  |  |
| --- | --- |
| **Process** | **Thread** |
| Process cần nhiều thời gian trong việc context switch giữa các process | Thread yêu cầu thời gian ít hơn trong việc context switch giữa các process |
| Process hoàn toàn độc lập và không chia sẻ bộ nhớ | Một thread sẽ chia sẻ một vài vùng nhớ với các thread trong cùng một process |
| Thời gian tạo và hủy lâu hơn thread | Thời gian tạo và hủy nhanh hơn process |
| Process chiếm nhiều tài nguyên | Thread chiếm ít tài nguyên hơn |
| Nếu 1 process bị block, các process còn lại vẫn có thể chạy bình thường | Nếu 1 thread bị block, các thread khác cũng bị block |

### Đặc điểm của Thread trong Linux

Trong hệ điều hành Linux, các thread có các đặc điểm như sau:

* Trong kernel Linux, không có sự phân biệt giữa process và thread.
* Kernel Linux xem thread như một process đặc biệt chia sẻ các tài nguyên với các process khác.
* Có thể dùng task để gọi các process (thread)
* Các task trong linux sẽ được kernel quản lý bằng cách lưu task\_struct của mỗi task vào một cấu trúc linked list gọi là task list
* Các Posix Thread trong Linux được thực thi thông qua thư viện glibc, trong đó, libpthread.so là một phần của thư viện này.
* Các hàm tạo thread và process con như pthread\_create(), fork(),.. cuối cùng đều gọi đến system call clone() nhưng với các tham số(flag) khác nhau.
* Một process (thread) sau khi kết thúc thì Process Descriptor của process vẫn còn tồn tại và chỉ được free bằng lệnh release\_task()



*Hình 2‑24: Task List trong hệ điều hành Linux*

* Task\_struct là một cấu trúc dữ liệu tương đối lớn, khoảng 1,7 kilobytes, nằm trong mục (linux/sched.h)
* Một task\_struct của process được gọi là một Process Discriptor

### Cơ chế quản lý tài nguyên thread

 Mỗi thread có không gian bộ nhớ thread stack riêng. Khi một thread kết thúc với hàm pthread\_exit(), kernel sẽ không tự động thu hồi tài nguyên của thread đó, lúc này thread đã kết thúc sẽ có trạng thái là zombie thread. Việc này gây ra lãng phí tài nguyên bộ nhớ vì tài nguyên này không thể tái sử dụng cho các thread khác. Bên cạnh đó nếu số lượng zombie thread quá lớn, hệ thống không thể tạo thêm thread được nữa.

Như vậy, một thread được tạo ra cũng giống như một tiến trình con, cần phải được theo dõi trạng thái và thu hồi tài nguyên khi kết thúc, tránh việc tạo ra zombie thread gây lãng phí tài nguyên của tiến trình. Có 2 kỹ thuật quản lý việc thu hồi tài nguyên thread là joinable thread và detached thread.

**Joinable thread**

Thread được kernel theo dõi trạng thái kết thúc và thu hồi tài nguyên.

Mặc định khi khai báo thread là joinable thread.

Trong Posix thread, hàm pthread\_join() có prototype như sau:

int pthread\_join(pthread\_t thread, void \*\*retval);

Hàm pthread\_join() sẽ block chương trình và chờ cho thread với ID là “**thread**” kết thúc, và giá trị return của thread đó được lưu vào biến con trỏ “**retval**”.  Nếu thread đã kết thúc trước khi gọi pthread\_join(), thì hàm sẽ return ngay lập tức. Việc gọi hàm pthread\_join() 2 lần với cùng một thread ID có thể dẫn đến lỗi “unpredictable behavior”.

Việc gọi hàm pthread\_join() sau khi tạo ra thread mới bằng hàm pthread\_create() cũng tương tự như việc gọi [system call wait()](https://vimentor.com/vi/lesson/quan-ly-tien-trinh-con) sau khi tạo ra tiến trình con bằng fork(). Chỉ khác là pthread\_join() có thể được gọi từ bất kỳ thread nào trong tiến trình, còn wait() phải được gọi bởi tiến trình cha đã tạo ra nó.

**Detached thread**

Một thread được thiết lập là detached thread khi lập trình viên không cần quan tâm đến trạng thái kết thúc của nó. Khác với joinable thread, kernel sẽ tự động thu hồi tài nguyên của detached thread khi nó kết thúc.

Một thread mới được tạo ra sẽ mặc định là joinable. Để chuyển thread thành detached, có thể dùng hàm pthread\_detach() với prototype như sau:

int pthread\_detach(pthread\_t thread);

Khi là detached thread, bạn không cần gọi hàm pthread\_join() để chờ thread kết thúc. Tuy nhiên, chúng ta sẽ không thể truy vấn trạng thái kết thúc của thread đó, cũng không thể chuyển thread đó thành joinable thread nữa.

### Semaphore

Là cơ chế đồng bộ giúp bảo vệ vùng bộ nhớ dễ xảy ra xung đột dữ liệu bởi nhiều thread.

Semaphore được lưu trong kernel, nó ngăn thread sửa đổi gián tiếp giá trị khi chưa có semaphore. Nó giống như key, ai có nó thì được sửa giá trị.

**Counting semaphore**

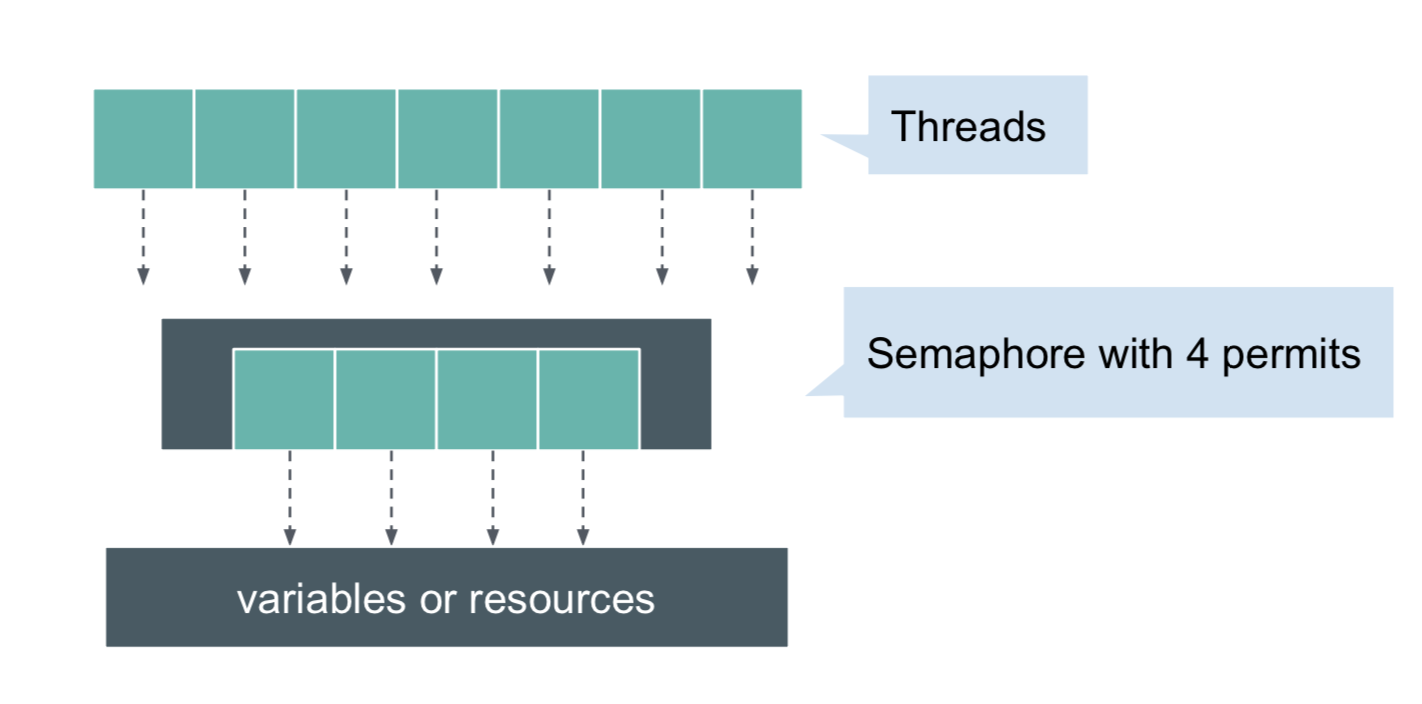
Cơ chế của counting semaphore gồm có hai bước:

* Khi gọi semaphore thì sẽ khởi tạo cho semaphore một giá trị N. Giá trị này sẽ là số lượng các thread có thể truy cập cùng một lúc vào tài nguyên. Mỗi khi thread lấy semaphore thì sẽ gọi hàm sem\_wait(S) để giảm giá trị của N. Nếu N nhỏ hơn hoặc bằng 0 thì thread sẽ bị block cho đến khi N thỏa mãn.
* Sau khi sử dụng xong thì thread gọi hàm sem\_post(S) tăng giá trị N để giải phóng semaphore cho các thread khác sử dụng.

**Binary semaphore**

Tương tự với counting semaphore, nhưng giá trị N chỉ là 0 hoặc 1.

Ứng dụng: Semaphore thường dùng để điều khiển, đồng bộ luồng truy cập.



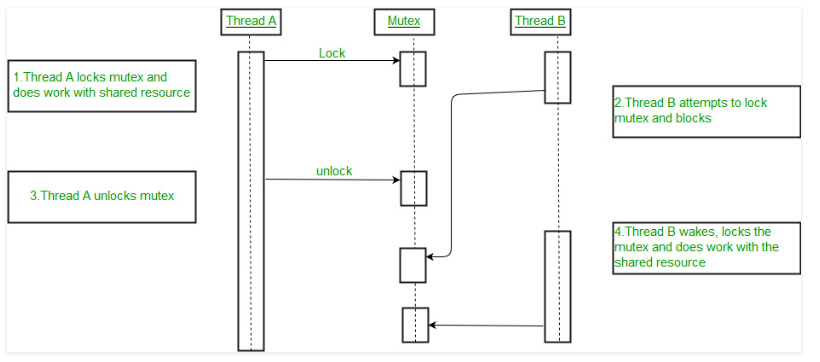
*Hình 2‑25: Cơ chế hoạt động của Semaphore*

### Mutex

Cơ chế đồng bộ theo dạng lock. Khi thread truy cập vào vùng nhớ, sẽ khóa vùng nhớ đó lại, chặn toàn bộ các luồng truy cập khác vào vùng nhớ cho đến khi chính thread đã khóa mutex giải phóng mutex.

Cơ chế của mutex tương tự với binary semaphore khi giá trị khởi tạo N là 1. Điểm khác biệt là mutex chỉ được mở khóa bởi chính thread đã khóa, còn binary semaphore sẽ chờ đến khi giá trị S = 1, tuy nhiên các thread khác có thể gọi hàm signal để tang biến S lên 1, và semaphore sẽ tín hiệu là tài nguyên có thể sử dụng.

Ứng dụng: Mutex được dùng để bảo vệ vùng nhớ trọng yếu, dễ xảy ra xung đột dữ liệu.



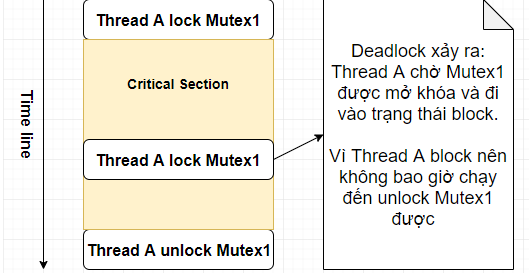
*Hình 2‑26: Cơ chế hoạt động của Mutex*

### Mutex deadlock

Khi một thread gọi hàm yêu cầu khóa để vào critical section mà có một thread khác đang khóa, nó sẽ chờ cho đến khi thread kia mở khóa rồi chiếm quyền. Nhưng vì một tình huống nào đó mà thread kia không bao giờ mở khóa được. Trạng thái một thread khóa một mutex mà rơi vào trạng thái chờ không thể giải thoát được gọi là deadlock.

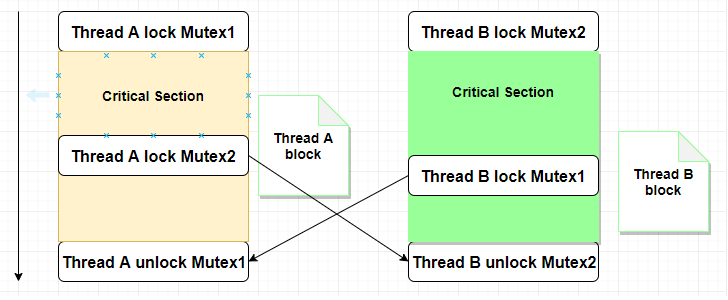
Trong lập trình multi-thread, deadlock xảy ra trong những trường hợp sau đây (giả sử có 2 thread A và B gây ra deadlock):

* Trường hợp 1: Thread A khóa một mutex mà chính nó trước đó đã bị khóa bởi chính nó.



*Hình 2‑27: Deadlock trong mutex(1)*

* Trường hợp 2: Thread A khóa Mutex1 và vào critical section, sau đó cố gắng khóa một Mutex2 mà thread B đang giữ. Trong khi đó, thread B đang khóa Mutex2 và lại cố gắng khóa Mutex1 ở trong critical section.



*Hình 2‑28: Deadlock trong mutex(2)*

Trong hai trường hợp deadlock ở trên, trường hợp 1 có thể ít xảy ra và dễ tìm thấy. Để tránh deadlock trong trường hợp 2, chúng ta có hai cách sau:

* Đảm bảo sao cho tất cả các thread khóa các mutex theo một thứ tự xác định trước. Ví dụ trong trường hợp trên, Thread B cần phải khóa Mutex1 trước, sau đó mới khóa Mutex2, khi đó nếu thread A đang giữ Mutex1 thì thread B chỉ bị block cho đến khi thread A nhả khóa chứ không bị deadlock.
* Sử dụng phương pháp "khóa thử, nếu không được thì quay lại sau" bằng cách dùng hàm pthread\_mutex\_trylock() để khóa mutex. Hàm pthread\_mutex\_trylock thay vì block thread nếu mutex đang bị khóa sẽ return ngay, qua đó tránh được việc block cũng như deadlock. Nếu pthread\_mutex\_trylock() return fail, thread release tất cả các mutex đang khóa và thử lại từ đầu. Cách này thường ít được sử dụng hơn cách trên vì yêu cầu vòng lặp khá tốn kém thời gian và phức tạp.

### So sánh Mutex và Semaphore

|  |  |
| --- | --- |
| **Semaphore** | **Mutex** |
| Cơ chế signalling | Cơ chế locking |
| Có thể cho nhiều thread truy cập vào tài nguyên được chia sẻ | Chỉ cho một thread truy cập vào tài nguyên được chia sẻ |
| Giá trị semaphore có thể được thay đổi bởi bất kì process hay thread nào | Mutex chỉ có thể được release bởi process hay thread đang giữ nó |

## Lab 1

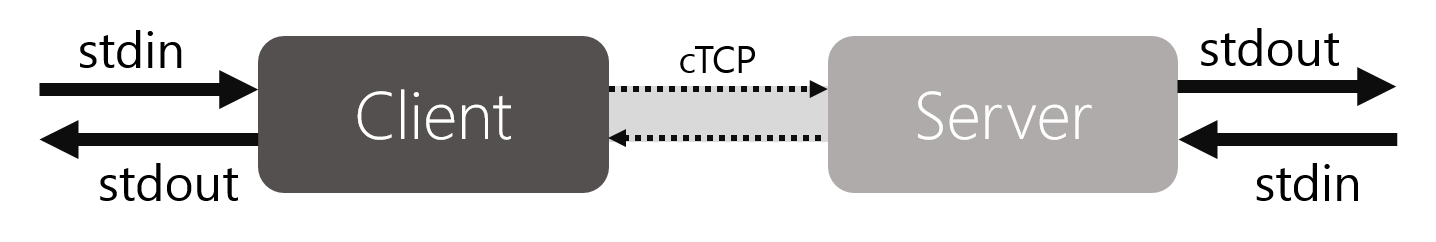
### Nhiệm vụ và yêu cầu

#### Nhiệm vụ

Thực hiện một project truyền nhận dữ liệu giữa client và server sử dựng giao thức cTCP (Cardinal TCP) sử dụng cơ chế Stop and Wait.

Client sẽ đọc một luồng dữ liệu từ standard input (STDIN), phân tách dữ liệu thành các cTCP segments, sau đó gửi các segment đó đến server. Server nhận được các segment sẽ đọc các segment đó, bỏ qua các segment bị hư hỏng hay các segment bị nhân đôi, sau đó đưa các dữ liệu trong các segment này theo thứ tự đến STDOUT. Kết nối được thực hiện ở hai hướng: Server cũng có thể đọc dữ liệu từ STDIN và gửi các segment, client đồng thời cũng có thể nhận các segment và đưa dữ liệu đến STDOUT. Cả hai client và segment có thể gửi dữ liệu cùng lúc.

Mỗi bên kết nối có thể kết thúc kết nối một cách độc lập. Khi cả hai bên kết nối được đóng, client và server phải xóa tất cả các tài nguyên mà kết nối sử dụng (bao gồm các linked lists, structs, …).



*Hình 2‑29: Sơ đồ hoạt động Lab 1*

#### Yêu cầu

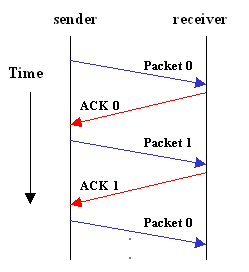
* Tốc độ gửi và nhận phải nhanh.
* Phải có khả năng xử lý input lớn hơn nhiều hơn so với window size.
* Hủy kết nối nếu truyền lại 1 segment quá 5 lần.
* Xử lý kết thúc kết nối khi nhận được EOF (Ctrl + D) từ STDIN.
* Có khả năng xử lý các segment Piggybacked có cả dữ liệu và cờ ACK.

### Tìm hiểu về giao thức Stop and Wait

#### Quá trình gửi và nhận dữ liệu

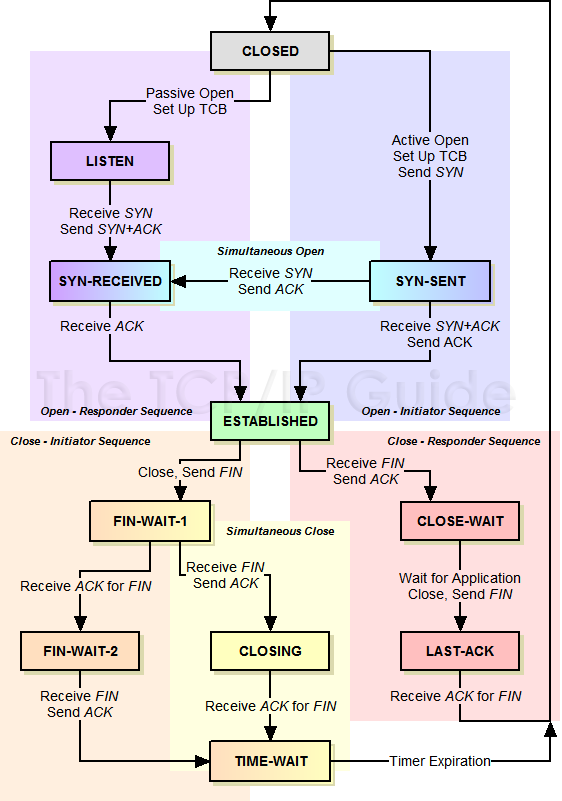
Sau khi gửi gói tin, bên gửi sẽ dừng lại và đợi cho đến khi nhận được bản tin ACK báo hiệu của bên nhận xác nhận gói tin đã đến đích thì mới tiếp tục gửi các gói tin mới.

Nếu sau thời gian timeout mà bên gửi không nhận được ACK từ bên nhận thì bên gửi sẽ gửi lại gói tin đó, nếu sau 5 lần gửi lại mà không nhận được gì từ bên nhận thì bên gửi sẽ cho rằng bên nhận đã đóng và bên gửi sẽ tiến hành đóng kết nối.



*Hình 2‑30: Hoạt động của cơ chế Stop and Wait*

#### Quá trình tạo một kết nối và kết thúc một kết nối



*Hình 2‑31: Sơ đồ trạng thái của TCP*

* Quá trình tạo một kết nối: Một kết nối sẽ được thực hiện khi có một bên gửi yêu cầu kết nối đến bên còn lại. Các trạng thái trong quá trình tạo một kết nối:
* LISTEN: Trạng thái lắng nghe yêu cầu kết nối từ bất kì kết nối TCP nào ở xa.
* SYN-SENT: Chờ nhận kết nối tương ứng sau khi gửi yêu cầu kết nối.
* SYN-RECEIVER: Chờ bản tin ACK xác nhận kết nối.
* ESTABLISHED: Kết nối đã được thành lập và sẵn sàng trao đổi dữ liệu.
* Quá trình kết thúc một kết nối: Một kết nối sẽ tiến hành quá trình kết thúc sau khi một bên nhận được gói tin FIN từ bên còn lại. Các trạng thái trong quá trình kết thúc kết nối:
* FIN-WAIT-1: Trạng thái chờ yêu cầu hủy kết nối từ kết nối TCP ở xa.
* FIN-WAIT-2: Trạng thái chờ yêu cầu hủy kết nối từ kết nối TCP ở xa.
* CLOSE-WAIT: Trạng thái sau khi nhận được hủy kết nối từ kết nối TCP local.
* CLOSING: Trạng thái chờ yêu cầu hủy kết nối từ kết nối TCP ở xa.
* TIME-WAIT: Chờ một khoảng thời gian vừa đủ để chắc chắn kết nối TCP ở xa nhận được bản tin ACK cho yêu cầu hủy kết nối.
* LAST-ACK: Trạng thái chờ nhận bản tin xác nhận hủy kết nối từ kết nối TCP local.

### Quá trình thực hiện

#### Cấu trúc struct ctcp\_state

typedef enum

{

ESTABLISH = 1,//can receive ACK and DATA in this state

FIN\_WAIT\_1,

FIN\_WAIT\_2,

TIME\_WAIT,

CLOSE\_WAIT,

LAST\_ACK,

CLOSING,

}conn\_status\_t;

typedef struct

{

bool is\_time\_wait\_destroy;

bool is\_time\_wait\_ack;

uint16\_t header\_len;

uint32\_t countdown;

uint32\_t time\_to\_destroy;

uint32\_t time\_wait\_ack;

uint32\_t ackno;

uint32\_t seqno;

conn\_status\_t state;

ctcp\_config\_t config;

}current\_state\_t;

struct ctcp\_state {

struct ctcp\_state \*next; /\* Next in linked list \*/

struct ctcp\_state \*\*prev; /\* Prev in linked list \*/

conn\_t \*conn; /\* Connection object -- needed in order to figure

out destination when sending \*/

linked\_list\_t \*send\_segments; /\* Linked list of segments sent to this connection.

It may be useful to have multiple linked lists

for unacknowledged segments, segments that

haven't been sent, etc. Lab 1 uses the

stop-and-wait protocol and therefore does not

necessarily need a linked list. You may remove

this if this is the case for you \*/

linked\_list\_t \*recv\_segments;

current\_state\_t current\_tcp; /\*contain current ack, seq, config tcp\*/

/\* FIXME: Add other needed fields. \*/

};

* *config*: Chứa các giá trị của window send, window receive, thời gian để truyền lại một gói tin và thời hằng của timer
* *ack, seq, state*: lưu giá trị ack, seq, trạng thái hiện tại của kết nối.
* *header\_len*: chiều dài của header ctcp
* *time\_to\_destroy*: Thời gian từ lúc TIME\_WAIT state đến khi đóng kết nối.
* *time\_to\_wait\_ack*: Thời gian cho phép chờ ack kể từ lúc gửi.
* *countdown*: Số lần gửi lại gói tin cho phép (yêu cầu là 5)
* *is\_time\_wait\_ack*: biến bool báo hiệu biết đang chờ ack
* *is\_time\_wait\_destroy*: biến bool báo hiệu biết đang chờ đến khi đóng kết nối.
* Thành phần *conn\_state\_t state\_t*: Chứa trạng thái của kết nối
* *Recv\_segments:* link list lưu các gói tin nhận được (lab1 thì link list này chỉ có độ dài lớn nhất là 1 vì stop&wait)

#### Các hàm sử dụng thêm

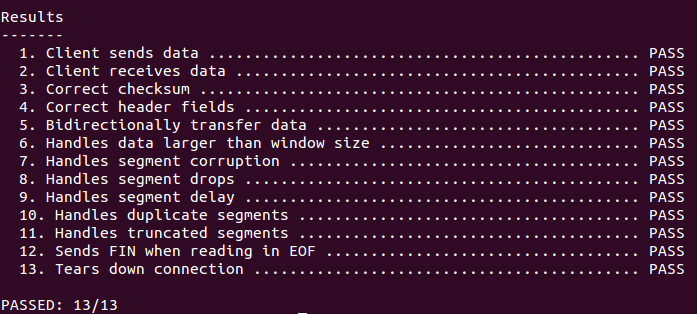
* Hàm *void create\_ack\_segment(ctcp\_state\_t \*state, ctcp\_segment\_t \*seg):* Tạo gói tin ack, output vào segment.
* Hàm *void create\_fin\_segment(ctcp\_state\_t \*state, ctcp\_segment\_t \*seg):* Tạo gói tin FIN, output vào segment.
* Hàm *void create\_data\_segment(ctcp\_state\_t \*state, ctcp\_segment\_t \*seg):* Tạo gói tin data, data lấy từ global buffer, output vào segment.
* Hàm *void start\_wait\_ack(ctcp\_state\_t \*state, bool b\_temp)*: b\_temp true thì báo hiệu chờ ack và đặt time\_wait\_ack bằng mặc định.
* Hàm *void send\_segment(ctcp\_state\_t \*state, ctcp\_segment\_t \*segment)*: Truyền gói tin segment và chờ ack.
* Hàm *bool* *checksum(ctcp\_state\_t \*state, ctcp\_segment\_t \*segment)*: Để kiểm tra checksum gói tin nhận có bị damaged hay không.
* Hàm void delete\_segment(linked\_list\_t \*segment): Xóa segment đầu trong linked list.
* Hàm *void handle\_receive\_ack(ctcp\_state\_t \*state, ctcp\_segment\_t \*segment):* Hàm xử lý khi nhận được gói tin ack.
* Hàm *void handle\_receive\_data(ctcp\_state\_t \*state, ctcp\_segment\_t \*segment):* Hàm xử lý khi nhận được gói tin data.

#### Các hàm được cung cấp

* Hàm *ctcp\_init(conn\_t \*conn, ctcp\_config\_t \*cfg):* Hàm khởi tạo và lưu các giá trị config và conn khởi tạo từ source ctcp\_sys\_internal.
* Hàm *ctcp\_read(ctcp\_state\_t \*state)*: Hàm này được gọi khi có input từ STDIN, tiếp theo hàm này sẽ đọc dữ liệu đó, tạo ra các segment từ dữ liệu nhận được và đưa vào linked list *send\_segments*. Khi đọc hết dữ liệu input, hàm sẽ gửi segment đầu tiên từ linked list *send\_segment* đi.
* Hàm  *ctcp\_receive(ctcp\_state\_t \*state, ctcp\_segment\_t \*segment, size\_t len)*: Hàm này được gọi ra khi có một segment đến kết nối. Đầu tiên, hàm sẽ xem segment đó là FIN, ACK hay DATA . Nếu là FIN thì sẽ tiến hành kết thúc kết nối, nếu là ACK thì sẽ gửi segment tiếp theo trong linked list *send\_segment.* Nếu là DATA thì sẽ đưa vào linked\_list recv\_segments và khi gửi gói tin ACK trở lại sẽ xóa segment này đi (việc đưa segment nhận được vào link list không cần thiết trong lab1 nhưng sẽ hữu dụng trong lab2), đồng thời output data lên STDOUT. Các giá trị ack, seq và state sẽ được cập nhật mỗi lần nhận đúng gói tin.

### Kết quả thực hiện

Passed 13/13 test cases



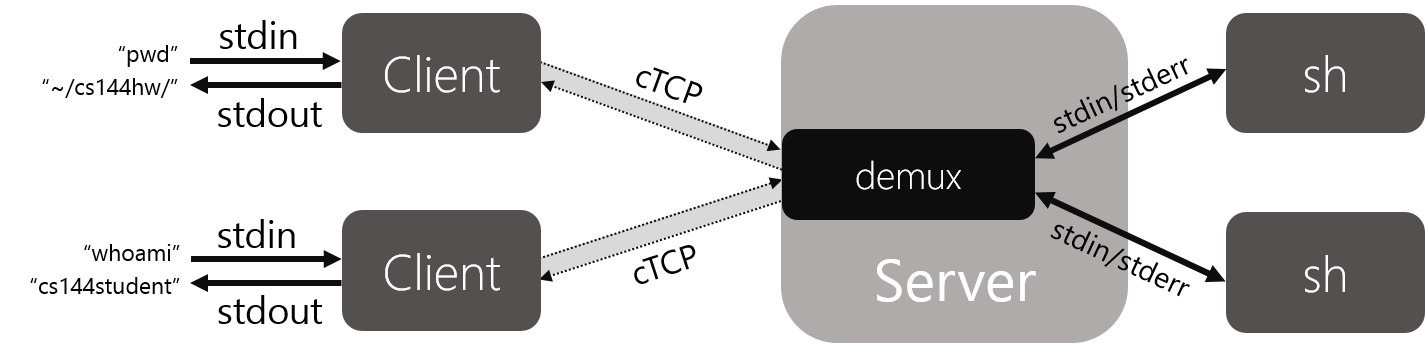
*Hình 2‑32: Kết quả Lab 1*

## Lab 2

### Nhiệm vụ và yêu cầu

#### Nhiệm vụ

Tiếp tục phát triển từ Lab 1, thay vì sử dụng cơ chế Stop and Wait, ta sẽ sử dụng cửa sổ trượt (Sliding Window) với kích thước *n\*MAX\_SEG\_DATA\_SIZE* với n>=1. Ở đây, đối với cửa sổ trượt, ta sẽ sử dụng cơ chế Selective repeat.



*Hình 2‑33: Sơ đồ hoạt động của Lab2*

#### Yêu cầu

* Lab 2 phải đáp ứng tất cả các yêu cầu của Lab 1.
* Đảm bảo dữ liệu được đưa ra theo đúng thứ tự nếu các segment không được nhận theo đúng thứ tự.
* Có khả năng xử lý khi nhiều client kết nối đến.

### Tìm hiểu về cơ chế Selective Repeat

Cơ chế hoạt động của Selective Repeat ở bên gửi và bên nhận:

#### Bên gửi

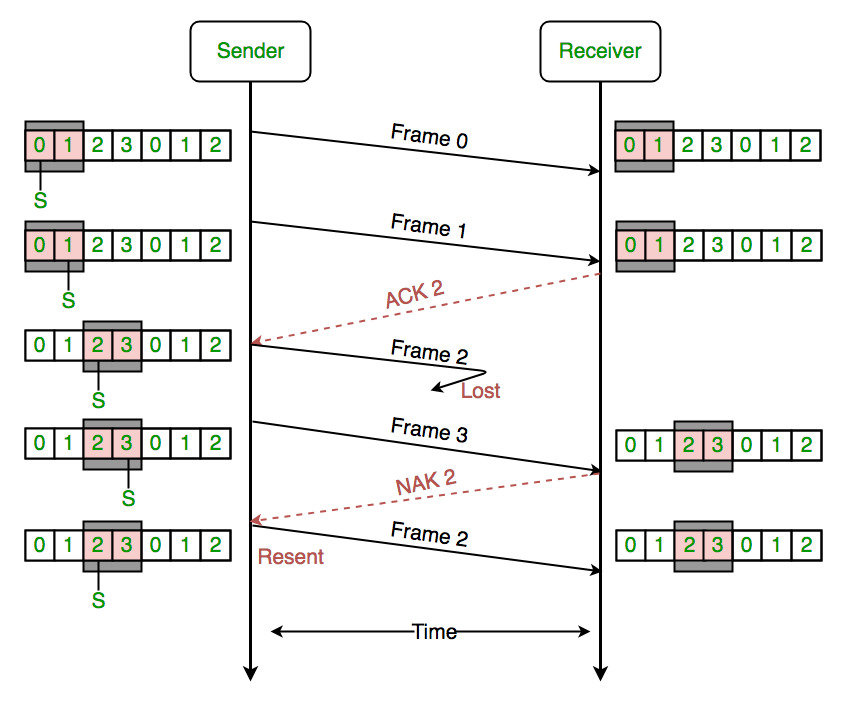
Các sự kiện xảy ra ở bên gửi:

* *Nhận dữ liệu từ tầng trên*: Khi nhận được dữ liệu từ lớp trên, bên gửi sẽ kiểm tra sequence number kế tiếp cho segment. Nếu sequence number nằm trong vùng cửa sổ gửi, dữ liệu sẽ được đóng gói và gửi; ngược lại, dữ liệu sẽ được lưu tạm trong bộ đệm hoặc trả lại cho tầng trên để chuyển sau.
* *Timeout*: Timer được sử dụng để đánh dấu các segment bị mất. Mỗi segment sẽ có 1 timer riêng cho nó bởi vì chỉ gửi lại duy nhất 1 segment khi timeout.
* *Nhận ACK*: Nếu 1 ACK được nhận, bên gửi sẽ đánh dấu segment đó đã được nhận, miễn là nó nằm trong window. Nếu segment ACK này dùng để xác nhận cho segment đã gửi có sequence number bằng với send\_base của window thì window sẽ trượt lên phía trước đến gói tin chưa được báo nhận mà có sequence number nhỏ nhất. Nếu window di chuyển và có các segment chưa gửi mà có sequence number nằm trong window thì các segment đó sẽ được gửi.

#### Bên nhận

Các sự kiện xảy ra ở bên nhận:

* *Segment với sequence number nằm trong [rcv\_base, rcv\_base + N -1] được nhận chính xác*: Trong trường hợp này, các segment nhận được sẽ rơi vào trong cửa sổ nhận và 1 Selective ACK sẽ được gửi trả tới sender. Nếu các segment trước nó chưa được nhận thì nó sẽ được đưa vào bộ đệm. Nếu segment nhận được có sequence number bằng với rcv\_window thì segment và các segment đã được đưa vào bộ đệm sẽ cùng được gửi lên lớp trên. Window sau đó sẽ di chuyển bằng số segment đã di chuyển lên lớp trên.
* *Segment với sequence number nằm trong [rcv\_base – N, rcv\_base -1] đã được nhận thành công*: Trong trường hợp này, 1 ACK phải được tạo và gửi mặc dù segment này đã được acknowledged trước đó.
* *Các trường hợp còn lại*: Bỏ qua các gói tin đến.



*Hình 2‑34: Cơ chế của Selective Repeat*

### Quá trình thực hiện

#### Các thành phần thêm vào struct ctcp\_state

Tương tự như các thành phần đã nêu ở Lab 1 nhưng có thêm các thành phần:

typedef struct

{

uint8\_t ack\_state;//0: nothing, 1: waiting ack, 2: received ack

uint32\_t time\_wait\_ack;

ctcp\_segment\_t \*segment;

}nsegment\_t;

typedef struct

{

bool is\_time\_wait\_destroy;

uint16\_t header\_len;

uint32\_t countdown;

uint32\_t time\_to\_destroy;

uint32\_t ackno;

uint32\_t seqno\_send;

uint32\_t send\_base;

uint32\_t recv\_base;

conn\_status\_t state;

ctcp\_config\_t config;

}current\_state\_t;

* *ack\_state*: trạng thái ack của gói tin hiện tại.
* *send\_base:* sequence number đầu của send window
* *recv\_base:* sequence number đầu của receive window

#### Các hàm sử dụng thêm

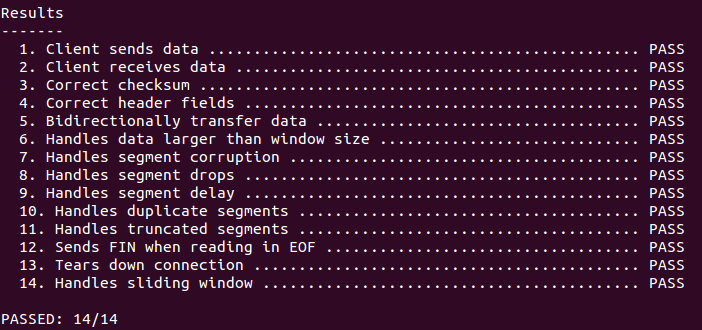
Tương tự như Lab 1, nhưng chỉnh sửa 1 số hàm để phù hợp với lab2.

#### Các hàm được cung cấp

* Hàm *ctcp\_read(ctcp\_state\_t \*state)*: Được gọi khi có input từ STDIN, hàm sẽ đọc dữ liệu và đưa dữ liệu vào các segment, sau đó lưu vào thành phần *linked\_list\_t \*send\_segments*. Hàm sẽ bắt đầu gửi các segment đầu tiên cho đến khi đầy kích thước window gửi.
* Hàm *ctcp\_receive(ctcp\_state\_t \*state, ctcp\_segment\_t \*segment, size\_t len)*: Được gọi khi có một segment nhận được từ kết nối. Tương tự như Lab 1 nhưng hàm này có thêm chức năng khi nhận được segment mà không đúng theo thứ tự, nó sẽ tiến hành đưa segment vào link list *recv\_segments* đã khai báo lúc đầu và khi nhận được segment đúng theo thứ tự bị thiếu đó, hàm sẽ output ra STDOUT segment này cộng với các segment đã được đưa vào bộ đệm.

#### Kết quả thực hiện

Passed 14/14 test cases



*Hình 2‑35: Kết quả Lab 2*

# KẾT QUẢ ĐẠT ĐƯỢC TRONG THỜI GIAN THỬ VIỆC

|  |  |  |
| --- | --- | --- |
| STT | Nội dung công việc đã thực hiện (nêu các đầu việc đã thực hiện, kết quả đạt được) | Đánh giá của người được giao kèm cặp (nếu có) hoặc của người quản lý trực tiếp (đối với từng nội dung tương ứng) |
| I | **Công việc chính** |  |
| 1 | * Tìm hiểu về cơ cấu tổ chức, văn hóa, các giá trị cốt lõi của Viettel. * Tìm hiểu về nhiệm vụ, quan điểm, mục tiêu và mô hình tổ chức của Tổng công ty Công nghiệp – Công nghệ cao Viettel (VPT), Trung tâm Kĩ thuật công nghệ. |  |
| 2 | * Lập trình C cơ bản. * Tìm hiểu và lập trình về internet socket (TCP Socket, UDP Socket). * Tìm hiểu Process và các cơ chế giao tiếp, trao đổi dữ liệu giữa các Process (IPC) * Tìm hiểu thread và các cơ chế đồng bộ threads, viết chương trình multithread. |  |
| 3 | * Thực hiện và hoàn thành Lab 1 (Cardinal TCP theo cơ chế Stop and Wait). |  |
| 4 | * Thực hiện và hoàn thành Lab 2 (Cardinal TCP theo cơ chế Selective Repeat). |  |
| II | **Công việc phụ (hoặc đột xuất)** |  |
| 1 | Không có công việc phụ (hoặc đột xuất). |  |
| III | **Tính phối hợp, tinh thần trách nhiệm, khả năng sáng tạo** |  |
| 1 | Thường xuyên thực hiện các buổi Senimar để chia sẻ, thảo luận những kiến thức với các anh/chị, các bạn trong đội về các vấn đề gặp phải trong quá trình nghiên cứu, thực hiện công việc. |  |
| IV | **Đề xuất, kiến nghị** | |
| 1 | Tiếp tục thực hiện các bài Lab còn lại và tham gia vào dự án OLT sau khi đã hoàn thành các bài Lab đó. | |

# CÁC TÀI LIỆU ĐÃ ĐỌC, HỌC TRONG THỜI GIAN THỬ VIỆC

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên tài liệu** | **Tóm tắt nội dung** | **Tác dụng đối với công việc đang thực hiện** |
| 1 | The C Programming Language | Các kiến thức về lập trình C cơ bản | Tổng kết lại các kiến thức C cơ bản đã biết và tìm hiểu thêm một vài kiến thức mới trên nền tảng Linux |
| 2 | UNIX Network Programming Volume 1 | Lập trình mạng trong môi trường Linux | Các kiến thức về Socket, lập trình mạng căn bản với TCP Socket và UDP Socket |
| 3 | UNIX Network Programming Volume 2 | Các kiến thức về IPC, cơ chế đồng bộ giữa các process, thread | Sử dụng các IPC để trao đổi dữ liệu giữa các Process với nhau, lập trình multithread |
| 4 | A top-down approach featuring the internet | Cơ sở truyền dữ liệu tin cậy, cơ chế của TCP | Các kiến thức dùng đê hỗ trợ, hoàn thành bài Lab 1 và Lab 2 |
| 5 | <https://zinascii.com/2014/a-posix-queue-implementation.html> | Cách hoạt động của message queue POSIX | Hiểu rõ hơn cơ chế message queue |

# NHẬN XÉT CỦA CÁC CÂP

## Nhận xét, đánh giá của người quản lý trực tiếp: (Mặt tích cực, hạn chế, triển vọng)

|  |  |  |
| --- | --- | --- |
| **NGÀY** | **NHẬN XÉT**  Nhận xét:  Đề xuất: | **CHỮ KÝ** |

## Nhận xét của người quản lý chức năng (nếu có)

|  |  |  |
| --- | --- | --- |
| **NGÀY** | **NHẬN XÉT**  Nhận xét:  Đề xuất: | **CHỮ KÝ** |

## Xét duyệt của Ban giám đốc

|  |  |  |
| --- | --- | --- |
| **NGÀY** | **NHẬN XÉT** | **CHỮ KÝ** |