TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CƠ SỞ NGÀNH MẠNG

Đề tài hệ điều hành

TÌM HIỂU SỰ TRANH CHẤP TÀI NGUYÊN VÀ XÂY DỰNG GIẢI PHÁP MONITOR CHO BÀI TOÁN CÁC TRIẾT GIA ĂN TỐI

Đề tài mạng

TÌM HIỂU IP HELPER VÀ XÂY DỰNG CHƯƠNG TRÌNH MY IPCONFIG

Giáo viên hướng dẫn:	**ThS. NGUYỄN THẾ XUÂN LY**
Sinh viên thực hiện:	**NGUYỄN NGỌC QUANG NHÂN**
Lớp:	**17T1**
MSSV:	**102170040**

Đà Nẵng, 12/2020

LỜI CẢM ƠN

Nguyên lý hệ điều hành và Lập trình mạng là những kiến thức căn bản là nền tảng mà mỗi lập trình viên phải hiểu rõ để phục vụ cho những kiến thức cao hơn.
Hệ điều hành là tập hợp các chương trình phần mềm chạy trên máy tính, dùng để điều hành, quản lý các thiết bị phần cứng và tài nguyên phần mềm trên máy tính. Hệ điều hành đóng vai trò trung gian trong việc giao tiếp giữa người sử dụng và phần cứng máy tính, cung cấp một môi trường cho phép người sử dụng phát triển và thực hiện các ứng dụng một cách dễ dàng.

Cùng với sự phát triển của mạng Internet hiện nay, thì việc nghiên cứu nắm vững những kiến thức về mạng là rất quan trọng.
Dựa trên những hiểu biết của mình và tìm hiểu tài liệu, em đã nghiên cứu và thực hiện 2 đề tài: Bài toán 5 triết gia ăn tối và Xây dựng chương trình my_ipconfig

Em xin chân thành cảm ơn thầy cô khoa Công Nghệ Thông Tin đã tạo điều kiện để em nghiên cứu kĩ hơn những kiến thức này, và đặc biệt là thầy Nguyễn Thế Xuân Ly đã nhiệt tình theo dõi, hướng dẫn em trong quá trình thực hiện đề tài này.
Vì kiến thức còn hạn hẹp, nên không thể tránh khỏi những sai sót trong quá trình làm đề tài, rất mong nhận được sự góp ý của thầy cô để sản phẩm được hoàn thiện hơn. Em xin chân thành cảm ơn.

Sinh viên thực hiện

Nguyễn Ngọc Quang Nhân

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH TỪ VIẾT TẮT

Từ viết tắt	Diễn giải
JDT	Java Development Toolkit
SDK	Software Development Kit
PDE	Plug-in Development Environment

MỞ ĐẦU

1.Tổng Quan Về Đề Tài:

2. Mục đích và ý nghĩa của đề tài

2.1. Mục đích

- Đi sâu và nắm vững một cách có hệ thống kiến thức đã thu nhận được trong quá trình học lí thuyết, làm bài tập và thực hành.

- Từng bước làm quen với các công tác khoa học có định hướng của giáo viên hướng dẫn và hình thành hành vi nghiên cứu độc lập có sự trợ giúp của tài liệu tham khảo.

- Gắn quá trình học lí thuyết với công tác nghiên cứu thực tế.

- Trình bày rõ ràng và khoa học một vấn đề thuộc lĩnh vực nghiện cứu của mình.

2.2. Ý nghĩa

- Kết quả của công việc phản ánh công sức, tài năng, trí tuệ của người làm đồ án và phải được trình bày bằng văn bản trong đồ án môn học theo những chuẩn mực và yêu cầu của giáo viên hướng dẫn.

- Đồ án môn học phản ánh công sức nghiên cứu nên cần phải được trình bày trước bộ môn và được đánh giá bằng điểm số. Việc trình bày đúng, rõ ràng, ngắn gọn và khoa học chỉ có được ở những sinh viên có quá trình lao động nghiêm túc cho đề tài và có chuẩn bị đầy đủ kiến thức trong lĩnh vực nghiên cứu đề tài.

3. Bố cục của đề tài

Báo cáo đề tài bao gồm các nội dung sau:

Mở đầu

Phần I: Nguyên lý hệ điều hành

Phần II: Lập trình mạng

Tài liệu tham khảo

Phụ lục

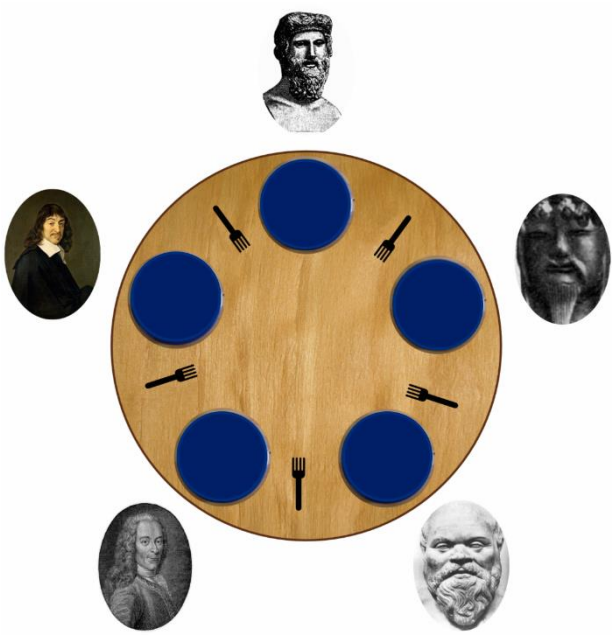
PHẦN I: NGUYÊN LÍ HỆ ĐIỀU HÀNH

Đề tài: Viết chương trình giải quyết bài toán năm triết gia ăn tối. Chương trình phải tạo ra năm tiến trình con mô phỏng hoạt động của năm triết gia. Dùng monitor để đồng bộ hoạt động của năm triết gia này.

Mô tả vấn đề:

Đây là bài toán cổ điển về hệ điều hành. Bài toán bữa tối của cá triết gia được đưa ra bởi nhà toán học E. W. Dijkstra. Bài toán được mô tả như sau:

Có năm triết gia cùng ngồi ăn tối quanh một chiếc bàn tròn, trước mặt mỗi người có một đĩa mì Ý, giữa 2 triết gia có một chiếc nĩa.



Hình 1. Minh họa bài toán 5 triết gia ăn tối

Mỗi triết gia dành toàn bộ thời gian để suy nghĩ hoặc ăn khi đói.

Mối triết gia chỉ có thể ăn khi có được 2 chiếc nĩa bên cạnh mình.

Đói: một triết gia có thể chết đói nếu ông ta không có cách nào để ăn được.

Tắc nghẽn: các triết gia phải đợi lẫn nhau nên không có ai ăn được.

Yêu cầu bài toán:

Phải đặt ra thuật toán sao cho khi một triết gia đói thì ông ta sẽ được ăn và đảm bảo không có triết gia nào bị chết đói.

Bài toán đặt ra vấn đề "đồng bộ giữa các tiến trình", giải quyết vấn đề tắc nghẽn có thể xảy ra.

Thuật toán được đưa ra là thuật toán Monitor.

CHƯƠNG 1: CƠ SỞ LÍ THUYẾT

1. Tiến trình (Proccess)

1.1. Khái niệm:

Tiến trình là một bộ phận của một chương trình đang thực hiện, đơn vị thực hiện tiến trình là Proccesser.

Vì tiến trình là một bộ phận của chương trình nên tương tự như chương trình, tiến trình cũng sở hữu một con trỏ lệnh, một con trỏ Stack, một tập các thanh ghi, một không gian địa chỉ trong bộ nhớ chính và tất cả các thông tin cần thiết khác để tiến trình có thể hoạt động được.

1.2. Định nghĩa tiến trình

Theo Saltzer: Tiến trình là một chương trình do một proccesor logic thực hiện.

Theo Horming& Rendell: Tiến trình là một quá trình chuyển từ trạng thái này sang trạng thái khác dưới tác động của hàm hành động, xuất phát từ một trạng thái ban đầu nào đó.

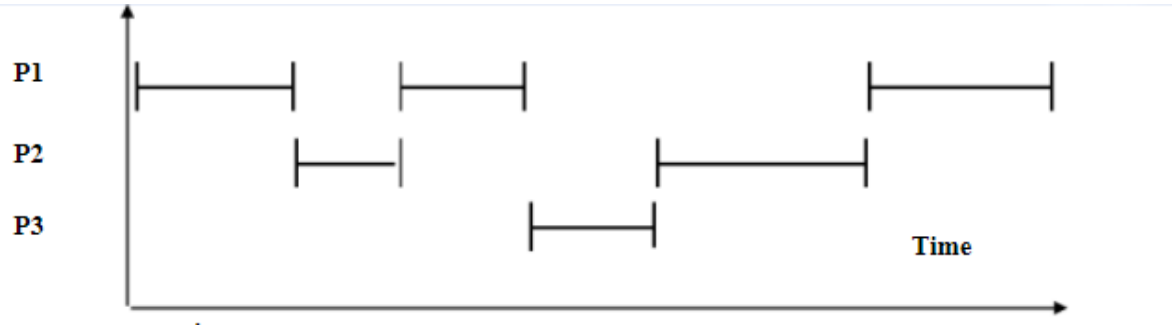
1.3. Các loại tiến trình

Tiến trình tuần tự: là các tiến trình mà điểm khởi tạo nó là điểm kết thúc của tiến trình trước đó.

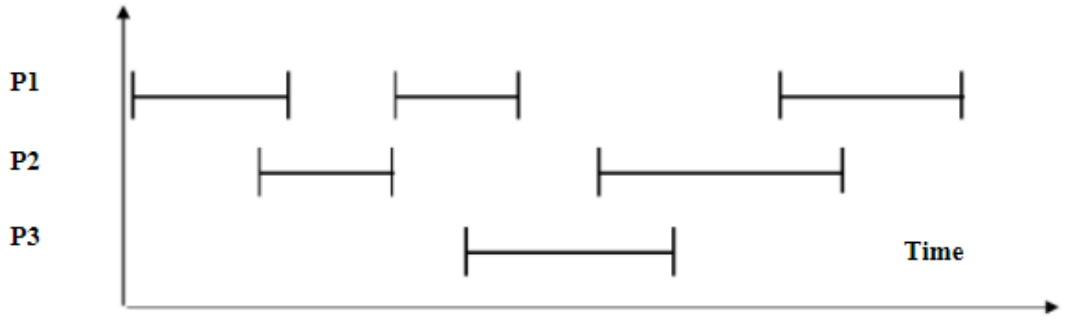
Tiến trình song song: là các tiến trình mà điểm khởi tạo của tiến trình này nằm ở thân của tiến trình khác, tức là có thể khởi tạo một tiến trình mới khi các tiến trình trước đó chưa kết thúc.

Tiến trình tuần tự xuất hiện trong các hệ điều hành đơn nhiệm như hệ điều hành MS_DOS.

Các tiền trình song song xuất hiện trong hệ điều hành đa nhiệm (uniprocessor và multiprocesser).



Hình 2. Tiến trình song song trong hệ thống uniprocessor



Hình 3. Tiến trình song song trong hệ thống multiprocessor

1.4. Tuyến (Thread)

-Tuyến là một thành phần của tiến trình sở hữu ngăn xếp và thực thi độc lập ngay trong mã lệnh của tiến trình. Nếu như hệ điều hành có nhiều tiến trình thì trong mỗi tiến trình hoạt động song song trong hệ điều hành. Ưu điểm của tuyến là chúng hoạt động trong cùng một không gian địa chỉ của tiến trình. Tập hợp một nhóm các tuyến có thể sử dụng chung biến toàn cục, vùng nhớ heap, bằng mô tả file, … của tiến trình, cơ chế liên lạc giữa các tuyến đơn giản hơn cơ chế liên lạc giữa các tiến trình với nhau (nếu hệ điều hành của bạn chạy trên phần cứng nhiều bộ xử lí thì tuyến thực sự chạy song song chứ không phải chạy giả lập kiểu xoay vòng).

-Ưu điểm: sử dụng tuyến trong tiến trình đơn giản hơn lập trình tuần tự. Nhiều thao tác xuất nhập hoặc hiển thị dữ liệu có thể tách rời và phân cho các tuyến chạy được độc lập thực thi. Ví dụ trong môi trường đồ họa, khi bạn copy một file có kích thước lớn, chương trình sẽ được thiết kế sao cho một tuyến thực hiện đọc ghi dữ liệu từ đĩa, tuyến khác sẽ đảm trách việc hiển thị phần trăm hoàn thành công việc cho người dùng theo dõi tiến độ.

-Đối với hệ điều hành chi phí để chuyển đổi giữa ngữ cảnh của tiến trình cao và chậm hơn chi phí chuyển đổi ngữ cảnh đảm bảo cho tuyến (với tiến trình hệ điều hành phải cất thông số môi trường, thanh ghi trạng thái, hoán đổi vùng nhớ, …).

-Tuy nhiên, điểm yếu của việc dùng tuyến đó là **khả năng đổ vỡ của một tuyến sẽ ảnh hưởng đến tất cả các tuyến khác và toàn bộ tiến trình đang hoạt động**. Lí do là các tuyến dùng chung vùng nhớ và không gian địa chỉ của tiến trình. Ngược lại, một tiến trình bị đổ vỡ luôn được hệ điều hành cô lập hoàn toàn không gây ảnh hưởng đến các tiến trình khác. Tiến trình có thể chạy trên nhiều máy khác nhau trong khi tuyến chỉ được thực thi trên một máy và trong một tiến trình.

2. Tài nguyên găng và đoạn găng:

2.1. Tài nguyên găng (Critical Resource)

Trong môi trường hệ điều hành đa nhiệm- đa chương- đa người dùng sử dụng, việc chia sẻ tài nguyên cho các tiến trình của người dùng sử dụng dùng chung là cần thiết, nhưng nếu hệ điều hành không tổ chức tốt việc sử dụng tài nguyên dùng chung của các tiến trình hoạt động đồng thời, thì không những không mang lại hiệu quả khai thác tài nguyên của hệ thống mà còn làm hỏng dữ liệu của các ứng dụng. Và nguy hiểm hơn là việc hỏng dữ liệu này có thể hệ điều hành và ứng dụng không phát hiện được. Việc hỏng dữ liệu của ứng dụng có thể làm sai lệch ý nghĩa thiết kế của nó. Đây là điều mà cả hệ điều hành và người lập trình đều không mong muốn.

Các tiến trình hoạt động đồng thời thường cạnh tranh với nhau trong việc sử dụng tài nguyên dùng chung. Hai tiến trình hoạt động đồng thời cùng ghi vào một không gian nhớ chung (một biến chung) trên bộ nhớ hay hai tiến trình đồng thời cùng ghi dữ kiệu vào một file chia sẻ, đó là những biểu hiện của sự cạnh tranh về việc sử dụng tài nguyên dùng chung của các tiến trình. Để các tiến trình hoạt động đồng thời không cạnh tranh hay xung đột với nhau khi sử dụng tài nguyên dùng chung hệ điều hành phải tổ chức cho các tiến trình này được độc quyên truy xuất/ sử dụng trên các tài nguyên dùng chung này.

Những tài nguyên được hệ điều hành chia sẻ cho nhiều tiến trình hoạt động đồng thời dùng chung, mà có nguy cơ dẫn đến sự tranh chấp giữa các tiến trình này khi sử dụng chúng, được gọi là tàu nguyên găng. Tài nguyên găng có thể là tài nguyên phần cứng hoặc tài nguyên phần mềm, có thể là tài nguyên phân chia được hoặc không phân chia được, nhưng đa số thường là tài nguyên phân chia được như là: các biến chung, các file chia sẻ.

2.2. Đoạn găng (Critical Section)

Đoạn code trong các tiến trình đồng thời, có tác động đến các tài nguyên có thể trở thành tài nguyên găng được gọi là đoạn găng hay miền găng. Tức là, các đoạn code trong các chường trình dùng để truy cập đến các vùng nhớ chia sẻ, các tập tin chia sẻ được gọi là các đoạn găng.

Để hạn chế các lỗi có thể xảy ra do sử dụng tài nguyên găng, hệ điều hành phải điều khiển các tiến trình sao cho, tại một thời điểm chỉ có một tiến trình nằm trong đoạn găng, nếu có nhiều tiến trình cùng muốn vào(thực hiện) đoạn găng thì chỉ có một tiến trình được vào, các tiến trình khác phải chờ, một tiến trình khi ra khỏi(kết thúc) đoạn găng phải báo cho hệ điều hành và/ hoặc các tiến trình khác biết để các tiến trình này vào đoạn găng,vv. Các công tác điều khiển tiến trình được thực hiện đoạn găng

của hệ điều hành được gọi là điều độ tiến trình qua đoạn găng. Để công tác điều độ tiến trình qua đoạn găng được thành công, thì cần phải có sự phối hợp giữa vi xử lí, hệ điều hành và người lập trình. Vi xử lí đưa các chỉ thị, hệ điều hành cung cấp các công cụ để người lập trình cây dựng các sơ đồ điều độ hợp lý, để đảm bảo sự độc quyền trong việc sử dụng tài nguyên găng của tiến trình.

2.3. Yêu cầu đối với đoạn găng

Đoạn găng phải thỏa mãn các yêu cầu sau:

- Tại một thời điểm không thể có hai tiến trình nằm trong đoạn găng.

- Nếu có nhiều tiến trình đồng thời cùng xin được vào đoạn găng thì chỉ có một tiến trình được phép vào đoạn găng, các tiến trình khác phải xếp hàng chờ trong hàng đợi.

- Tiến trình chờ ngoài đoạn găng không được ngăn cản các tiến trình khác vào đoạn găng.

- Không có tiến trình nào được phép ở lâu vô hạn trong đoạn găng và không có tiến trình phải chờ lâu mới được vào đoạn găng (chờ trong hàng đợi).

- Nếu tài nguyên găng được giải phóng thì hệ điều hành có nhiệm vụ đánh thức các tiến trình trong hàng đợi ra để tạo điều kiện cho nó vào đoạn găng.

Các vấn đề có thể gặp phải đối với đoạn găng:

- Có thể dẫn đến tắc nghẽn (Deadlock) trong hệ thống.

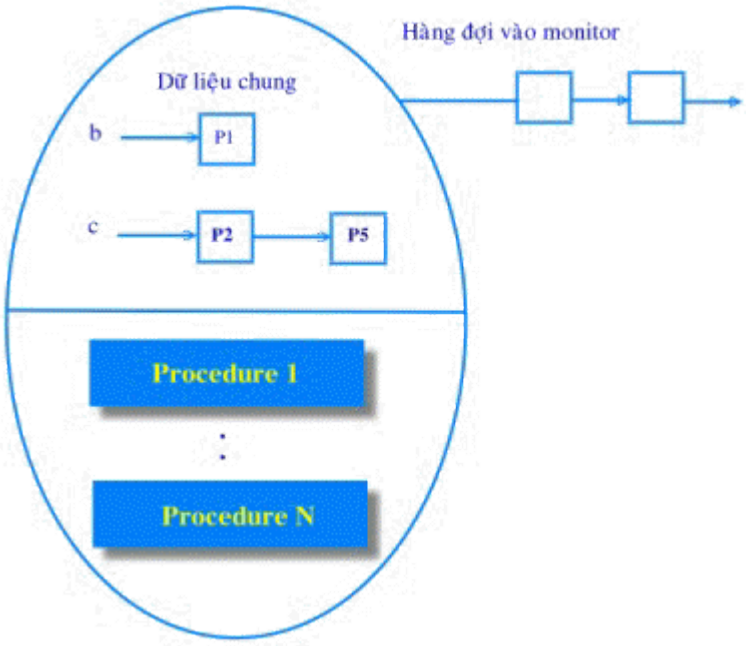
- Các tiến trình có thể bị đói (Stravation) tài nguyên.

3. Giải Pháp Monitor:

3.1. Tiếp cận: Để có thể dễ viết đúng các chương trình đồng bộ hóa hơn, Hoare(1974) và Brinch & Hansen (1975) đã đề nghị một cơ chế cao hơn được cung cấp bởi ngôn ngữ lập trình , là *monitor*. Monitor là một cấu trúc đặc biệt bao gồm các thủ tục, các biến và cấu trúc dữ liệu có các thuộc tính sau:

- Các biến và cấu trúc dữ liệu bên trong monitor chỉ có thể được thao tác bởi các thủ tục định nghĩa bên trong monitor đó. (*encapsulation*).
- Tại một thời điểm, chỉ có một tiến trình duy nhất được hoạt động bên trong một monitor (*mutual exclusive*)
- Trong một monitor, có thể định nghĩa các *biến điều kiện* và hai thao tác kèm theo là **Wait** và **Signal** như sau: gọi ***c*** là biến điều kiện được định nghĩa trong monitor:

- **Wait(c):** chuyển trạng thái tiến trình gọi sang blocked, và đặt tiến trình này vào hàng đợi trên biến điều kiện ***c***.
- **Signal(c)**: nếu có một tiến trình đang bị khóa trong hàng đợi của ***c***, tái kích hoạt tiến trình đó, và tiến trình gọi sẽ rời khỏi monitor.



Hình 4. Monitor và các biến điều kiện

3.2. Cài đặt: trình biên dịch chịu trách nhiệm thực hiện việc truy xuất độc quyền đến dữ liệu trong monitor. Để thực hiện điều này, một semaphore nhị phân thường được sử dụng. Mỗi monitor có một hàng đợi toàn cục lưu các tiến trình đang chờ được vào monitor, ngoài ra, mỗi biến điều kiện ***c*** cũng gắn với một hàng đợi ***f(c)*** và hai thao tác trên đó được định nghĩa như sau:

Wait(c):

```
status(P)= blocked;
enter (P, f(c));
```

Signal(c) :

```
if (f(c)!= NULL){
        exit(Q ,f(c)); //Q là tiến trình chờ trên c
        status(Q) = ready;
        enter(Q,ready-list);
}
```

3.3. Sử dụng:

-Với mỗi nhóm tài nguyên cần chia sẻ, có thể định nghĩa một monitor trong đó đặc tả tất cả các thao tác trên tài nguyên này với một số điều kiện nào đó.

monitor <tên monitor >
condition <danh sách các biến điều kiện>;
<declaration de variables>;

procedure $Action_1$();
{

}
....
procedure $Action_n$ ();
{

}
end monitor;

-Các tiến trình muốn sử dụng tài nguyên chung này chỉ có thể thao tác thông qua các thủ tục bên trong monitor được gắn kết với tài nguyên:

while (TRUE) {
Noncritical-section ();
<monitor>.Actioni; //critical-section();
Noncritical-section ();
}

3.4. Thảo luận: Với monitor, việc truy xuất độc quyền được bảo đảm bởi trình biên dịch mà không do lập trình viên, do vậy nguy cơ thực hiện đồng bộ hóa sai giảm rất nhiều. Tuy nhiên giải pháp monitor đòi hỏi phải có một ngôn ngữ lập trình định nghĩa khái niệm monitor, và các ngôn ngữ như thế chưa có nhiều.

4. Deadlock

4.1. giới thiệu vấn đề

Trong môi trường đa chương, nhiều quá trình có thể cạnh tranh một số giới hạn tài nguyên. Một quá trình yêu cầu tài nguyên, nếu tài nguyên không sẵn dùng tại thời điểm đó, quá trình đi vào trạng thái chờ. Qúa trình chờ có thể không bao giờ chuyển trạng thái trở lại vì tài nguyên chúng yêu cầu bị giữ bởi những quá trình đang chờ khác. Trường hợp này được gọi là deadlock (khóa chết).

4.2. Điều kiện hình thành tắt nghẽn

Trường hợp deadlock có thể phát sinh nếu bốn điều kiện sau xảy ra cùng một lúc trong hệ thống:

- Loại trừ hỗ tương: ít nhất một tài nguyên phải được giữ trong chế độ không chia sẻ. Nghĩa là, chỉ một quá trình tại cùng một thời điểm có thể sử dụng tài nguyên. Nếu một quá trình khác yêu cầu tài nguyên đó, quá trình yêu cầu phải tạm dừng cho đến khi tài nguyên được giải phóng.

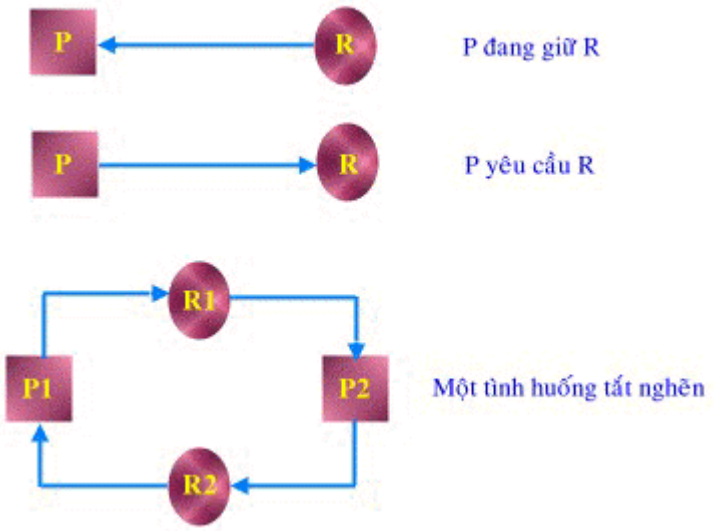
- Giữ và chờ cấp thêm tài nguyên: quá trình phải đang giữ ít nhất một tài nguyên và đang chờ để nhận tài nguyên thêm mà hiện đang được giữ bởi quá trình khác.

- Không đòi lại tài nguyên từ quá trình đang giữ chúng: Các tài nguyên không thể bị đòi lại; nghĩa là, tài nguyên có thể được giải phóng chỉ tự ý bởi quá trình đang giữ nó, sau khi quá trình đó hoàn thành tác vụ

- Tồn tại chu trình trong đồ thị cấp phát tài nguyên: một tập hợp các tiến trình {P0, P1,...,Pn-1} đang chờ mà trong đó P0 đang chờ một tài nguyên được giữ bởi P1, P1 đang chờ tài nguyên đang giữ bởi P2,…,Pn-1 đang chờ tài nguyên đang được giữ bởi quá trình P0.

4.3. Đồ thị cấp phát tài nguyên

Có thể sử dụng một đồ thị để mô hình hóa việc cấp phát tài nguyên. Đồ thị này có 2 loại nút: các tiến trình được biểu diễn bằng hình tròn, và mỗi tài nguyên được hiển thị bằng hình vuông.



Hình 5. Đồ thị cấp phát tài nguyên

4.4. Ngăn chặn tắc nghẽn (Deadlock Prevention)

Để deadlock xảy ra, một trong bốn điều kiện cần phải xảy ra. Bằng cách đảm bảo ít nhất một trong bốn điều kiện này không thể xảy ra, chúng ta có thể ngăn chặn việc xảy ra của deadlock. Chúng ta tìm hiểu chi tiết cách tiếp cận này bằng cách xem xét mỗi điều kiện cần riêng rẻ nhau.:

- Loại trừ hỗ tương: nhìn chung gần như không thể tránh được điều kiện này vì bản chất tài nguyên gần như cố định. Tuy nhiên đối với một số tài nguyên về kết xuất, người ta có thể dùng các cơ chế spooling để biến đổi thành tài nguyên có thể chia sẻ
- Giữ và chờ cấp thêm tài nguyên: phải bảo đảm rằng mỗi khi tiến trình yêu cầu thêm một tài nguyên thì nó không chiếm giữ các tài nguyên khác. Có thể áp đặt một trong hai cơ chế truy xuất sau:
    - Tiến trình phải yêu cầu tất cả các tài nguyên cần thiết trước khi bắt đầu xử lý. Cách này có thể lãng phí tài nguyên hệ thống nếu một tiến trình yêu cầu một tài nguyên sớm mà chưa cần sử dụng ngay.
    - Khi tiến trình yêu cầu một tài nguyên mới và bị từ chối, nó phải giải phóng các tài nguyên đang chiếm giữ, sau đó lại được cấp phát trở lại cùng lần với tài nguyên mới. Phương pháp này làm phát sinh các khó khăn trong việc bảo vệ tính toàn vẹn dữ liệu của hệ thống
- Không đòi lại tài nguyên từ quá trình đang giữ chúng: cho phép hệ thống được thu hồi tài nguyên từ các tiến trình bị khoá và cấp phát trở lại cho tiến trình khi nó thoát khỏi tình trạng bị khóa. Tuy nhiên với một số loại tài nguyên, việc thu hồi sẽ rất khó khăn vì vi phạm sự toàn vẹn dữ liệu.
- Tồn tại một chu trình trong đồ thị cấp phát tài nguyên: một cách để tránh tạo chu trình đồ thị cấp phát tài nguyên là đánh số tất cả các tài nguyên và các tiến trình phải yêu cầu tài nguyên theo thứ tự tăng dần (hoặc giảm dần). Nói cách khác, khi một tiến trình đang chiếm giữ tài nguyên Ri thì chỉ có thể yêu cầu các tài nguyên Rj nếu j > i.

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1. Cách giải quyết bài toán:

-Vấn đề Bữa ăn tối của các triết gia: 5 nhà triết học cùng ngồi ăn tối với món spaghetti nổi tiếng. Mỗi nhà triết học cần dùng 2 cái nĩa để có thể ăn spaghetti. Nhưng trên bàn chỉ có tổng cộng 5 cái nĩa để xen kẽ với 5 cái đĩa. Mỗi nhà triết học sẽ suy ngẫm các triết lý của mình đến khi cảm thấy đói thì dự định lần lượt cầm 1 cái nĩa bên trái và 1 cái nĩa bên phải để ăn. Nếu cả 5 nhà triết học đều cầm cái nĩa bên trái cùng lúc, thì sẽ không có ai có được cái nĩa bên phải để có thể bắt đầu thưởng thức spaghetti. Đây chính là tình trạng tắc nghẽn.

-Giải pháp Monitor được sử dụng để kiểm soát quyền truy cập vào các biến trạng thái và biến điều kiện.Nó chỉ cho biết khi nào cần vào và thoát ra khỏi phân đoạn và giải pháp này đưa ra hạn chế là một triết gia chỉ được gắp đũa khi cả hai đều có sẵn.

-Để mã hóa giải pháp này chúng ta cần phân biệt ba trạng thái mà chúng ta có thể tìm thấy ở một triết gia

- **SUY NGHĨ**: Khi triết gia không muốn tiếp cận với hai chiếc đũa
- **ĐÓI:** Khi triết gia muốn sử dụng đũa (bước vào critical section)
- **ĂN:** Khi triết gia đã có cả hai chiếc đũa, tức là anh ta đã bước vào critical section

-Nhà triết học thứ i có thể đặt biến trạng thái state[i] = EATING chỉ khi hai người hàng xóm của cô ấy không ăn (state [(i + 4)% 5]! = EATING) và (state [(i + 1)% 5]! = EATING).

```
monitor DP
{
   status state[5];
   condition self[5];

   // Lấy đũa lên
   Pickup(int i)
   {
      //chỉ ra trạng thái của triết gia thứ i là đang đói
      state[i] = hungry;
      //thiết lập trạng thành ăn trong hàm test() chỉ khi cả hai người hàng xóm đều
      //đang không ăn
      test(i);
      // Nếu không thử ăn thì đợi cho đến khi đươc đánh thức
      if (state[i] != eating)
         self[i].wait;
   }
```

```
   // Đặt đũa xuống
   Putdown(int i)
   {
      // chỉ ra trạng thái của triết gia thứ i là đang suy nghĩ
      state[i] = thinking;
      //Nếu người hàng xóm bên phải R=(i+1)%5 là đói và cả hai người hàng của
      // người bên phải R là đang không ăn thì thiết lập trạng thái của R là ăn
      // và đánh thức R bằng hàm signal
      test((i + 1) % 5);
      test((i + 4) % 5);
   }

   test(int i)
   {

      if (state[(i + 1) % 5] != eating && state[(i + 4) % 5] != eating
      && state[i] == hungry) {
         // chỉ ra trạng thái của triết gia thứ i là đang ăn
         state[i] = eating;
         // signal() không có tác dung trong Pickup(),
         // nhưng nó là quan trọng trong việc đánh thức triết học gia
         // trong hàng đợi trong Putdown()
         self[i].signal();
      }
   }
   init()
   {
      //Thực hiện Pickup(), Putdown() và test()
      // một cách riêng biệt
      // nghĩa là chỉ 1 hàm có thể thực hiện trong 1 thời gian nhất định
    For i = 0 to 4
      // Xác mình giải pháp monitor là
      // loại trừ deadlock
      // không có 2 người hàng xóm nào có thể đồng thời ăn
      state[i] = thinking;
   }
} //kết thúc monitor
```

Ta cần phải khai báo:
condition self[5];

Điều này cho phép triết gia i trì hoãn bản thân khi cô ấy đói nhưng không thể lấy được đôi đũa mà cô ấy cần. Việc phân phối đũa được kiểm soát bởi giám sát Dining Philosophers. Mỗi nhà triết học, trước khi bắt đầu ăn, phải gọi phép toán pickup (). Hành động này có thể dẫn đến việc đình chỉ quá trình của triết gia. Sau khi hoàn thành hoạt động thành công, triết gia có thể ăn. Sau đó, nhà triết học gọi phép toán putdown (). Do đó, nhà triết học i phải gọi các phép toán pickup () và putdown () theo trình tự sau:

DiningPhilosophers.pickup(i);
...
eat
...
DiningPhilosophers.putdown(i);

Dễ dàng cho thấy rằng giải pháp này đảm bảo rằng **không có hai người hàng xóm** nào ăn đồng thời vì:

```
if ((state[(i + 4) % 5] != EATING) &&
   (state[i] == HUNGRY) &&
   (state[(i + 1) % 5] != EATING)) {
      state[i] = EATING;
      self[i].signal();
}
```

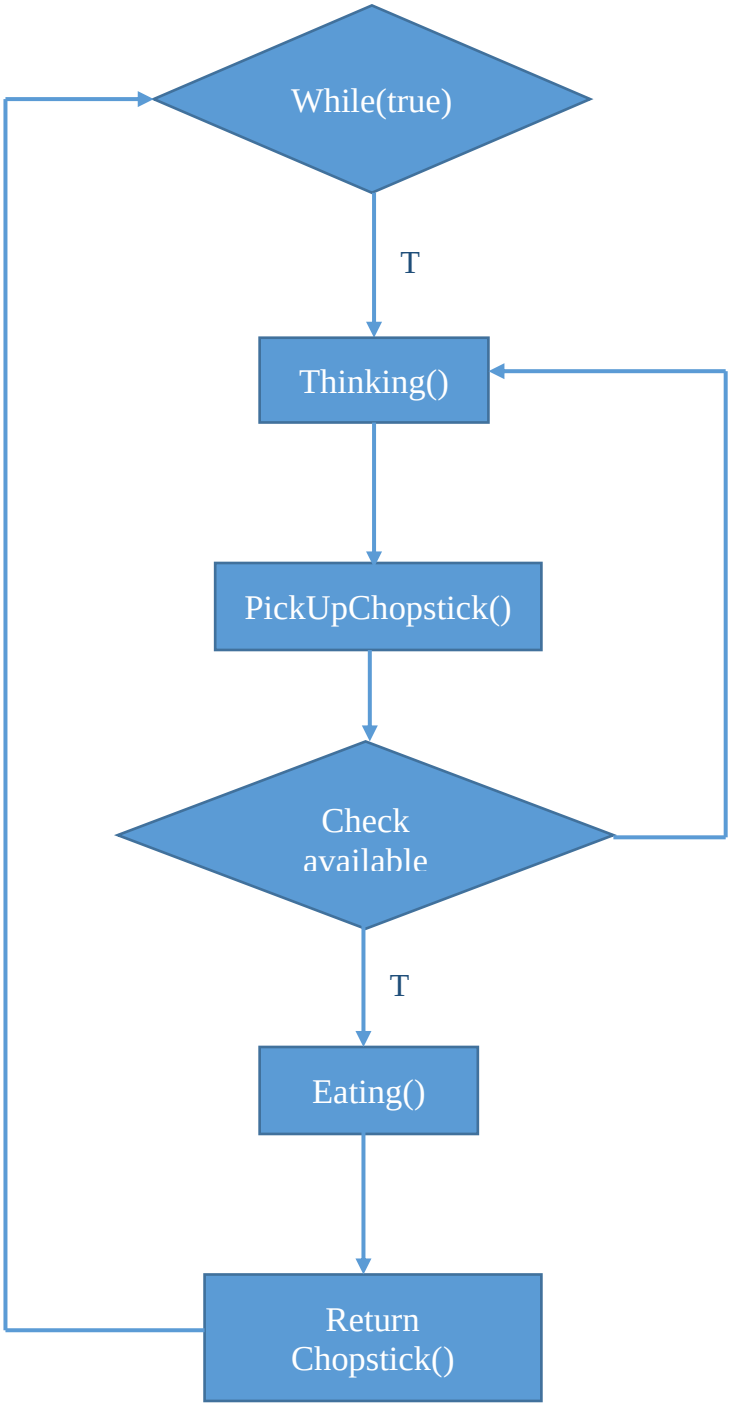
và mỗi lần trước khi ăn triết học gia phải cần đến 2 chiếc đũa điều này chỉ ra rằng nó sẻ không rơi vào tình trạng deadlock. Tuy nhiên, chúng tôi lưu ý rằng một triết gia có thể chết đói nếu như một trong những hàng xóm của cô ấy luôn ăn. Một ý tưởng đơn giản cho việc tránh chết đói là không cho phép một triết gia ăn hai lần trong khi một triết gia lân cận vẫn đói, đây được gọi là thuật toán “Lịch Sự”. Mã nguồn cài đặt thuật toán lịch sự (cải tiến của giải pháp Monitor truyền thống) để giải quyết việc tránh chết đói (các sữa đổi thể hiện bằng chữ in đậm)

```
monitor DP
{
   status state[5];
   condition self[5];
```

```
    boolean[] leftHungry = new boolean[5];
    boolean[] rightHungry = new boolean[5];
   // Lấy đũa lên
   Pickup(int i)
   {
      //chỉ ra trạng thái của triết gia thứ i là đang đói
      state[i] = hungry;
      //thiết lập trạng thành ăn trong hàm test() chỉ khi cả hai người hàng xóm đều
      //đang không ăn
      test(i);
      // Nếu không thử ăn thì đợi cho đến khi đươc đánh thức
      if (state[i] != eating) {
         self[i].wait;
        rightHungry[left(philosopher_number)] = false;
        leftHungry[right(philosopher_number)] = false;
        }
   }
   // Đặt đũa xuống
   Putdown(int i)
   {
      // chỉ ra trạng thái của triết gia thứ i là đang suy nghĩ
      state[i] = thinking;
      //Nếu người hàng xóm bên phải R=(i+1)%5 là đói và cả hai người hàng của
      // người bên phải R là đang không ăn thì thiết lập trạng thái của R là ăn
      // và đánh thức R bằng hàm signal
      test((i + 4) % 5);
      if (state[left(philosopher_number)] == States.HUNGRY)
                        leftHungry[philosopher_number] = true;
      test((i + 1) % 5);
      if (state[right(philosopher_number)] == States.HUNGRY)
                        rightHungry[philosopher_number] = true;
   }

   test(int i)
   {

      if (state[(i + 1) % 5] != eating && state[(i + 4) % 5] != eating
      && state[i] == hungry && !leftHungry[philosopher_number] &&
!rightHungry[philosopher_number] ) {
          // chỉ ra trạng thái của triết gia thứ i là đang ăn
```

```
            state[i] = eating;
            // signal() không có tác dung trong Pickup(),
            // nhưng nó là quan trọng trong việc đánh thức triết học gia
            // trong hàng đợi trong Putdown()
            self[i].signal();
        }
    }
    init()
    {
        //Thực hiện Pickup(), Putdown() và test()
        // một cách riêng biệt
        // nghĩa là chỉ 1 hàm có thể thực hiện trong 1 thời gian nhất định
     For i = 0 to 4
        // Xác mình giải pháp monitor là
        // loại trừ deadlock
        // không có 2 người hàng xóm nào có thể đồng thời ăn
        state[i] = thinking;

          leftHungry[i] = false;
          rightHungry[i] = false;
    }
} //kết thúc monitor
```

2. Sơ đồ bài toán:



Hình 6. Sơ đồ hoạt động của mỗi triết gia

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

1. Môi trường

1.1. Eclipse IDE

Eclipse là phần mềm miễn phí, được các nhà phát triển sử dụng để xây dựng những ứng dụng J2EE, sử dụng Eclipse nhà phát triển có thể tích hợp với nhiều công cụ hỗ trợ khác để có được một bộ công cụ hòan chỉnh mà không cần dùng đến phần mềm riêng nào khác. Eclipse SDK bao gồm 3 phần chính: Platform, Java Development Toolkit (JDT), Plug-in Development Environment (PDE). Với JDT, Eclipse được xem như là một môi trường hỗ trợ phát triển Java mạnh mẽ. PDE hỗ trợ việc mở rộng Eclipse, tích hợp các Plug-in vào Eclipse Platform. Eclipse Platform là nền tảng của toàn bộ phần mềm Eclipse, mục đích của nó là cung cấp những dịch vụ cần thiết cho việc tích hợp những bộ công cụ phát triển phần mếm khách dưới dạng Plug-in, bản thân JDT cũng có thể được coi như là một Plug-in làm cho Eclipse như là một Java IDE (Integrated Development Enviroment).

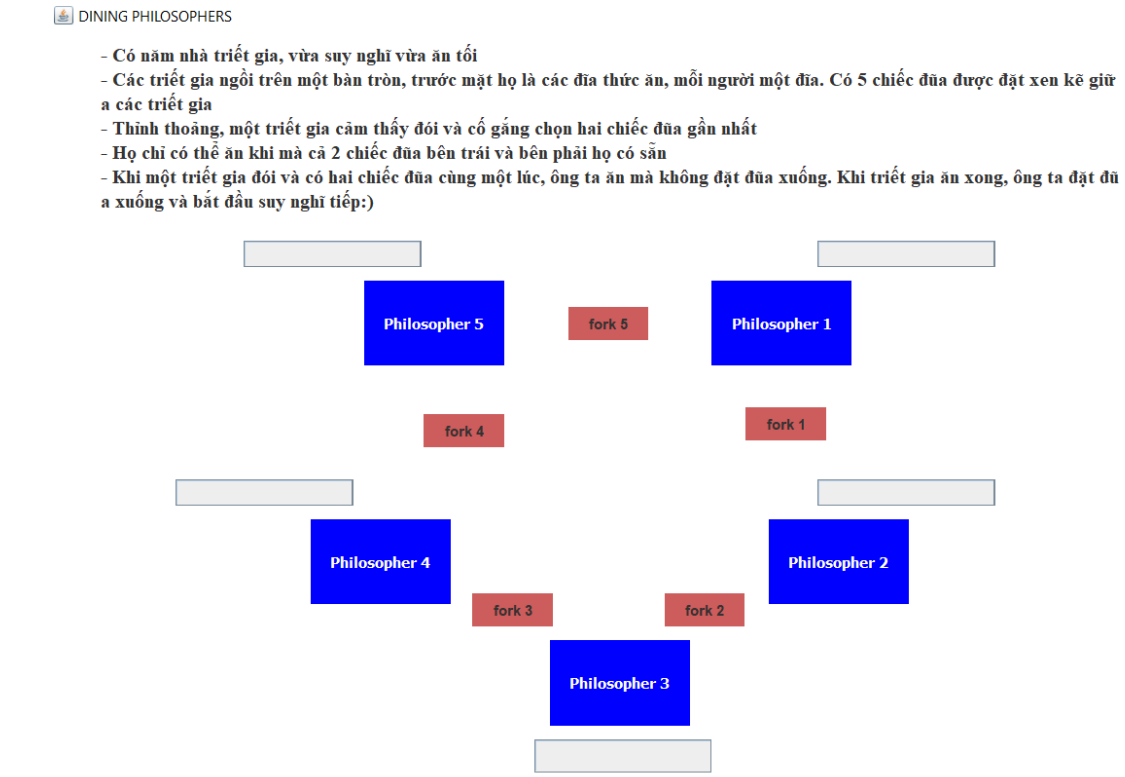
1.2. JRE

JRE (là viết tắt của Java Runtime Environment) được sử dụng để cung cấp môi trường runtime. Nó là trình triển khai của JVM. JRE bao gồm tập hợp các thư viện và các file khác mà JVM sử dụng tại runtime. Trình triển khai của JVM cũng được công bố bởi các công ty khác ngoài Sun Micro Systems.

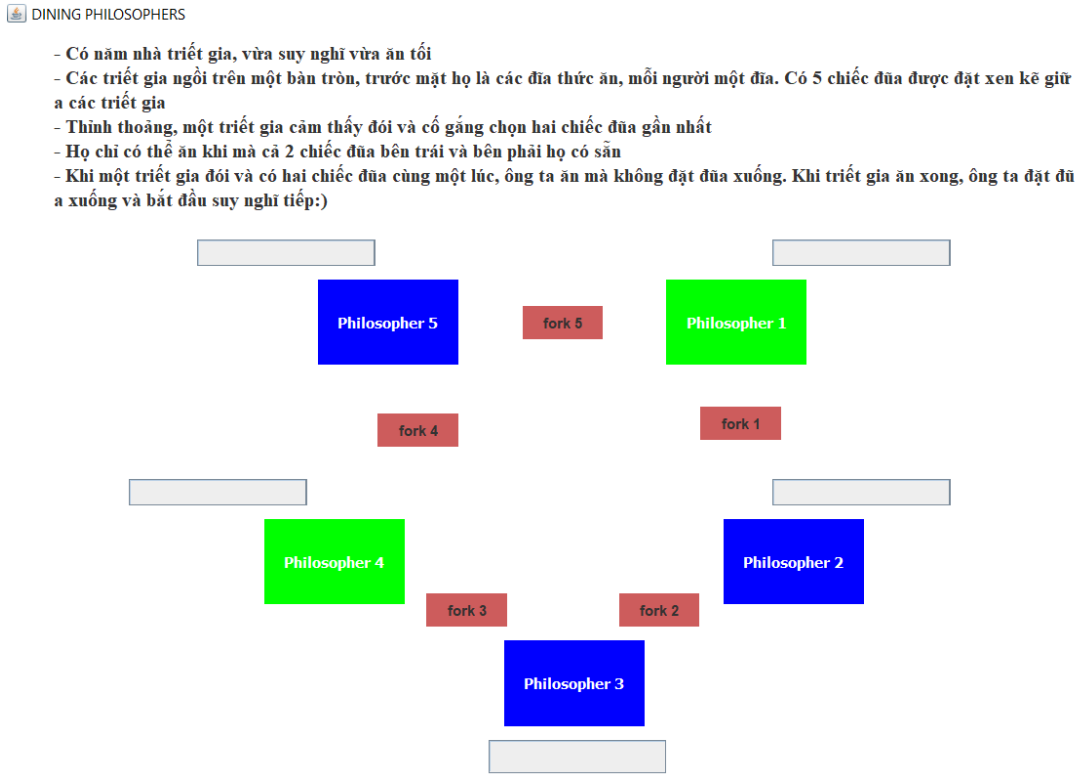
2. Triển khai

- Class Monitor.java: class xây dựng giải pháp Monitor.
- Class GUI.java: class xây dựng giao diện.
- Class Philosopher.java: class tạo đối tượng chạy chương trình

3. Kết Quả:



Hình 7. Màn hình khởi tạo bài toán



Hình 8. Các tiến trình đang chạy

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết Quả Đạt Được

Thông qua đồ án này em đã hiểu được kĩ hơn về giải pháp Monitor cũng như việc tranh chấp tài nguyên dùng chung giữa các tiến trình và cách giải quyết những tranh chấp đó.

Bên cạnh những mặt đã đạt được ở trên, trong quá trình làm đồ án chúng em đã học thêm được nhiều kiến thức bổ ích về hệ điều hành cũng như việc áp dụng giải pháp Monitor phát triển dựa trên ngôn ngữ lập trình Java.

1. Vấn đề còn tồn tại

– Giao diện và hình ảnh còn đơn giản.

2. Hướng phát triển

- Tiếp tục tìm hiểu nghiên cứu sâu hơn về Thread và các giải pháp tránh tắc nghẽn.
- Chỉnh sửa giao diện cho thuận tiện và logic hơn.

PHẦN II: LẬP TRÌNH MẠNG

Đề tài: Tìm hiểu IP HELPER và xây dựng chương trình MY IPCONFIG.

CHƯƠNG 1: CƠ SỞ LÍ THUYẾT

1. IP HELPER:

1.1. Giới thiệu

- Trình trợ giúp Giao thức Internet (IP Helper) hỗ trợ quản trị mạng của máy tính cục bộ bằng cách cho phép các ứng dụng truy xuất thông tin về cấu hình mạng của máy tính cục bộ và sửa đổi cấu hình đó. IP Helper cũng cung cấp các cơ chế thông báo để đảm bảo rằng một ứng dụng được thông báo khi các khía cạnh nhất định của cấu hình mạng máy tính cục bộ thay đổi.
- IP Helper cung cấp các khả năng trong các lĩnh vực sau:
    - Truy xuất thông tin về cấu hình mạng
    - Quản lí Network Adapters
    - Quản lí Giao Diện
    - Quản lí địa chỉ IP
    - Sử dụng giao thức phân giải địa chỉ
    - Truy xuất thông tin về Internet Protocol và Internet Control Message Protocol
    - Quản lí định tuyến (Routing)
    - Nhận thông báo về các sự kiện mạng
    - Truy xuất thông tin về giao thức điều khiển truyền (Transmission Control Protocol) và giao thức sơ đồ người dung (User Datagram Protocol)

1.2. Truy xuất thông tin về cấu hình mạng:

- IP Helper cung cấp thông tin về cấu hình mạng của máy tính cục bộ. Để truy xuất thông tin cấu hình chung, sử dụng hàm GetNetworkParams. Hàm này trả về các thông tin không dành riêng của adapter hoặc interface như : trả về danh sách các máy chủ DNS được sử dụng bởi máy tính cục bộ, ...

1.3. Quản Lí Network Adapter:

- IP Helper cung cấp khả năng quản lý bộ điều hợp mạng.
- Sử dụng các hàm sau để truy xuất thông tin về bộ điều hợp mạng trong máy tính cục bộ.
    - Hàm **GetAdapterInfo** trả về một mảng cấu trúc **IP_ADAPTER_INFO** cho mỗi bộ điều hợp trong máy tính cục bộ
    - Các **IP_ADAPTER_INFO** cấu trúc chứa thông tin về một bộ chuyển đổi mạng cụ thể trên máy tính cục bộ.
    - Các **GetAdaptersAddresses** chức năng cho phép bạn để lấy địa chỉ IP kết hợp với một bộ chuyển đổi cụ thể. Chức năng này hỗ trợ cả định địa chỉ IPv4 và IPv6.

1.4. Quản Lí Giao Diện:

- IP Helper mở rộng khả năng của bạn để quản lý các giao diện mạng
- Để quản lý các giao diện trên máy tính cục bộ, IP Helper cung cấp các hàm có thể trả về số lượng giao diện trên máy tính cục bộ, một bảng chứa tên và chỉ mục tương ứng cho các giao diện trên máy tính cục bộ, lấy một chỉ mục giao diện và trả về một chỉ mục giao diện tương thích ngược, tức là chỉ sử dụng 24 bit thấp hơn, ...

1.5. Quản Lí địa chỉ IP:

- IP Helper có thể hỗ trợ quản lý các địa chỉ IP được liên kết với các giao diện trên máy tính cục bộ.
- Để quản lý địa chỉ IP, IP Helper cung cấp các hàm có thể truy xuất một bảng chứa ánh xạ địa chỉ IP tới các giao diện, để thêm địa chỉ IP vào một giao diện cụ thể, xóa các địa chỉ IP đã được thêm trước đó
- Ngoài ra các hàm IpReleaseAddress và IpRenewAddress yêu cầu máy tính cục bộ để được sử dụng Dynamic Host Configuration Protocol (DHCP). Hàm **IpReleaseAddress** giải phóng một địa chỉ IP đã lấy trước đó từ DHCP. Hàm **IpRenewAddress** gia hạn hợp đồng thuê DHCP trên một địa chỉ IP cụ thể. Hợp đồng thuê DHCP cho phép máy tính sử dụng địa chỉ IP trong một khoảng thời gian nhất định.

1.6. Sử dung giao thức phân giải địa chỉ:

- IP Helper có thể thực hiện các hoạt động Giao thức phân giải địa chỉ (ARP) cho máy tính cục bộ. Sử dụng các chức năng để truy xuất và sửa đổi bảng ARP (Bảng ARP chứa ánh xạ địa chỉ IP thành địa chỉ vật lý).

1.7. Truy xuất thông tin về Internet Protocol và Internet Control Message Protocol:

- IP Helper cung cấp khả năng truy xuất thông tin hữu ích cho việc quản trị mạng của máy tính cục bộ. Các chức năng sau đây truy xuất thống kê cho Giao thức Internet (IP) và Giao thức thông báo điều khiển Internet (ICMP). Bạn cũng có thể sử dụng các chức năng sau để đặt các thông số cấu hình nhất định cho IP:
    - Chức năng lấy số liệu thống kê chỉ IP hiện tại cho máy tính cục bộ
    - Chức năng lấy số liệu thống kê ICMP hiện hành.

1.8. Quản Lí Định Tuyến:

- IP Helper cung cấp các tính năng để quản lý định tuyến mạng. Sử dụng các chức năng sau để quản lý bảng định tuyến IP và lấy thông tin định tuyến khác.
    - Truy xuất nội dung của bảng định tuyến IP

- Thao tác các mục nhập cụ thể trong bảng định tuyến IP: Thêm mục nhập bảng định tuyến, xóa mục nhập hiện có và sửa đổi mục nhập hiện có.

- Khả năng quản lý bộ định tuyến của IP Helper có thể được sử dụng để truy xuất thông tin về cách các sơ đồ được định tuyến qua mạng

- Truy xuất tuyến đường tốt nhất đến một địa chỉ đích được chỉ định
- Truy xuất chỉ mục của giao diện được sử dụng bởi tuyến đường tốt nhất đến một địa chỉ đích được chỉ định.
- Truy xuất thời gian khứ hồi (RTT) và số bước nhảy đến một địa chỉ đích được chỉ định.

1.9. Nhận Thông Báo Về Các Sự Kiện Mạng

- IP Helper sử dụng các chức năng sau để đảm bảo rằng ứng dụng nhận được thông báo về các sự kiện mạng nhất định.
    - yêu cầu thông báo về bất kỳ thay đổi nào xảy ra trong ánh xạ giữa địa chỉ IP và giao diện trên máy tính cục bộ.
    - cho phép ứng dụng yêu cầu thông báo về bất kỳ thay đổi nào xảy ra trong bảng định tuyến IP.

1.10. Truy xuất thông tin về giao thức điều khiển truyền (Transmission Control Protocol) và giao thức sơ đồ người dung (User Datagram Protocol)

- IP Helper có thể truy cập thông tin về các giao thức mạng được sử dụng trên máy tính cục bộ. Sử dụng các chức năng được mô tả trong các đoạn sau để truy xuất thông tin về Giao thức điều khiển truyền (TCP) và Giao thức sơ đồ người dùng (UDP) trên máy tính cục bộ.
    - truy xuất số liệu thống kê hiện tại cho TCP
    - truy xuất số liệu thống kê hiện tại cho UDP.
    - truy xuất bảng kết nối TCP và UDP

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1. Bài toán:

- Viết chường trình my_ipconfig để truy xuất các thông tin về mạng của máy tính cục bộ:

2. Thiết kế:

a) GetNetworkParams function (iphlpapi.h)

- Hàm **GetNetworkParams** được sử dụng để truy xuất các thông số mạng cho máy tính cục bộ. Các tham số mạng được trả về trong cấu trúc **FIXED_INFO**
- Cú Pháp:

```
IPHLPAPI_DLL_LINKAGE DWORD GetNetworkParams(
        PFIXED_INFO pFixedInfo,
        PULONG      pOutBufLen

);
```

trong đó:

- **pFixedInfo** :Một con trỏ tới bộ đệm chứa cấu trúc **FIX_INFO** nhận các tham số mạng cho máy tính cục bộ, nếu chức năng này thành công. Bộ đệm này phải được cấp phát bởi người gọi trước khi gọi hàm **GetNetworkParams** .
- **pOutBufLen:** Một con trỏ đến một biến ULONG chỉ định kích thước của cấu trúc FIXED_INFO . Nếu kích thước này không đủ để chứa thông tin, GetNetworkParams sẽ điền vào biến này với kích thước được yêu cầu và trả về mã lỗi **ERROR_BUFFER_OVERFLOW**. Nếu hàm thành công, giá trị trả về là **ERROR_SUCCESS**

- Cấu Trúc FIX_INFO:

```
typedef struct {
    char        HostName[MAX_HOSTNAME_LEN + 4];
    char        DomainName[MAX_DOMAIN_NAME_LEN + 4];
    PIP_ADDR_STRING   CurrentDnsServer;
    IP_ADDR_STRING    DnsServerList;
    UINT        NodeType;
```

```
    char        ScopeId[MAX_SCOPE_ID_LEN + 4];
    UINT        EnableRouting;
    UINT        EnableProxy;
    UINT        EnableDns;
} FIXED_INFO_W2KSP1, *PFIXED_INFO_W2KSP1;
```

b) GetAdaptersInfo function (iphlpapi.h)

Hàm GetAdaptersInfo truy xuất thông tin bộ điều hợp cho máy tính cục bộ.

- Cú Pháp:

```
IPHLPAPI_DLL_LINKAGE ULONG GetAdaptersInfo(
    PIP_ADAPTER_INFO AdapterInfo,
    PULONG          SizePointer
);
```

Trong đó:

- **AdapterInfo**: Một con trỏ tới bộ đệm nhận danh sách liên kết của cấu trúc **IP_ADAPTER_INFO.**
- **SizePointer:** Một con trỏ đến một biến ULONG chỉ định kích thước của bộ đệm được trỏ tới bởi tham số **pAdapterInfo** . Nếu kích thước này không đủ để chứa thông tin bộ điều hợp, **GetAdaptersInfo** sẽ điền vào biến này với kích thước được yêu cầu và trả về mã lỗi **ERROR_BUFFER_OVERFLOW** .

- Cấu Trúc **IP_ADAPTER_INFO:**

```
typedef struct _IP_ADAPTER_INFO {
    struct _IP_ADAPTER_INFO *Next;
    DWORD               ComboIndex;
    char AdapterName[MAX_ADAPTER_NAME_LENGTH +
4];
    char
Description[MAX_ADAPTER_DESCRIPTION_LENGTH + 4];
    UINT    AddressLength;
    BYTE    Address[MAX_ADAPTER_ADDRESS_LENGTH];
    DWORD               Index;
    UINT                Type;
```

```
    UINT            DhcpEnabled;
    PIP_ADDR_STRING     CurrentIpAddress;
    IP_ADDR_STRING      IpAddressList;
    IP_ADDR_STRING      GatewayList;
    IP_ADDR_STRING      DhcpServer;
    BOOL            HaveWins;
    IP_ADDR_STRING      PrimaryWinsServer;
    IP_ADDR_STRING      SecondaryWinsServer;
    time_t          LeaseObtained;
    time_t          LeaseExpires;
} IP_ADAPTER_INFO, *PIP_ADAPTER_INFO;
```

c) GetInterfaceInfo function (iphlpapi.h)

- Hàm GetInterfaceInfo lấy danh sách các bộ điều hợp giao diện mạng với IPv4 được kích hoạt trên hệ thống cục bộ.
- Cú Pháp:

```
IPHLPAPI_DLL_LINKAGE DWORD GetInterfaceInfo(
        PIP_INTERFACE_INFO pIfTable,
        PULONG         dwOutBufLen
    );
```

Trong đó:

- **pIfTable**: Một con trỏ tới bộ đệm chỉ định cấu trúc **IP_INTERFACE_INFO** nhận danh sách bộ điều hợp. Bộ đệm này phải được cấp phát bởi người gọi.
- **dwOutBufLen**: Một con trỏ đến một biến **DWORD** chỉ định kích thước của bộ đệm được trỏ tới bởi tham số pIfTable để nhận cấu trúc **IP_INTERFACE_INFO** . Nếu kích thước này không đủ để chứa thông tin giao diện IPv4, GetInterfaceInfo sẽ điền vào biến này với kích thước được yêu cầu và trả về mã **ERROR_INSUFFICIENT_BUFFER**.
  Nếu hàm thành công, giá trị trả về là **NO_ERROR** .

- Cấu trúc **IP_INTERFACE_INFO**:

```
typedef struct _IP_INTERFACE_INFO {
        LONG                    NumAdapters;
        IP_ADAPTER_INDEX_MAP    Adapter[1];
} IP_INTERFACE_INFO, *PIP_INTERFACE_INFO;
```

d) GetAdaptersAddresses function (iphlpapi.h)

- Hàm GetAdaptersAddresses truy xuất các địa chỉ được liên kết với bộ điều hợp trên máy tính cục bộ.
- Cú Pháp:

```
IPHLPAPI_DLL_LINKAGE ULONG GetAdaptersAddresses(
        ULONG                 Family,
        ULONG                 Flags,
        PVOID                 Reserved,
        PIP_ADAPTER_ADDRESSES AdapterAddresses,
        PULONG                SizePointer
);
```

Trong đó:

- **Family**: Họ địa chỉ của các địa chỉ cần truy xuất. Tham số này phải là một trong các giá trị sau.

Giá trị	Ý nghĩa
AF_UNSPEC	Trả lại cả địa chỉ IPv4 và IPv6 được liên kết với bộ điều hợp có bật IPv4 hoặc IPv6.
AF_INET	Chỉ trả lại các địa chỉ IPv4 được liên kết với bộ điều hợp có bật IPv4.
AF_INET6	Chỉ trả lại các địa chỉ IPv6 được liên kết với bộ điều hợp có bật IPv6.

- **Flags**: Loại địa chỉ cần truy xuất
- **Reserved**: Tham số này hiện không được sử dụng, nhưng được dành riêng cho việc sử dụng hệ thống trong tương lai. Ứng dụng gọi phải truyền NULL cho tham số này.
- **AdapterAddresses**: Một con trỏ đến bộ đệm có chứa danh sách liên kết các cấu trúc **IP_ADAPTER_ADDRESSES** khi trả về thành công.
- **SizePointer**: Một con trỏ đến một biến chỉ định kích thước của bộ đệm được chỉ ra bởi AdapterAddresses .

- Cấu trúc **IP_ADAPTER_ADDRESSES_LH:**

```
typedef struct _IP_ADAPTER_ADDRESSES_LH {
    union {
        ULONGLONG Alignment;
        struct {
            ULONG    Length;
            IF_INDEX IfIndex;
        };
    };
    struct _IP_ADAPTER_ADDRESSES_LH    *Next;
    PCHAR                              AdapterName;
    PIP_ADAPTER_UNICAST_ADDRESS_LH
        FirstUnicastAddress;
    PIP_ADAPTER_ANYCAST_ADDRESS_XP
        FirstAnycastAddress;
    PIP_ADAPTER_MULTICAST_ADDRESS_XP
        FirstMulticastAddress;
    PIP_ADAPTER_DNS_SERVER_ADDRESS_XP
        FirstDnsServerAddress;
    PWCHAR                             DnsSuffix;
    PWCHAR                             Description;
    PWCHAR                             FriendlyName;
    BYTE
    PhysicalAddress[MAX_ADAPTER_ADDRESS_LENGTH];
    ULONG                              PhysicalAddressLength;
```

```
                union {
                    ULONG Flags;
                    struct {
                        ULONG DdnsEnabled : 1;
                        ULONG RegisterAdapterSuffix : 1;
                        ULONG Dhcpv4Enabled : 1;
                        ULONG ReceiveOnly : 1;
                        ULONG NoMulticast : 1;
                        ULONG Ipv6OtherStatefulConfig : 1;
                        ULONG NetbiosOverTcpipEnabled : 1;
                        ULONG Ipv4Enabled : 1;
                        ULONG Ipv6Enabled : 1;
                        ULONG Ipv6ManagedAddressConfigurationSupported
: 1;
                    };
                };
                ULONG                   Mtu;
                IFTYPE                  IfType;
                IF_OPER_STATUS          OperStatus;
                IF_INDEX                Ipv6IfIndex;
                ULONG                   ZoneIndices[16];
                PIP_ADAPTER_PREFIX_XP   FirstPrefix;
                ULONG64                 TransmitLinkSpeed;
                ULONG64                 ReceiveLinkSpeed;
                PIP_ADAPTER_WINS_SERVER_ADDRESS_LH
                    FirstWinsServerAddress;
                PIP_ADAPTER_GATEWAY_ADDRESS_LH                ‘
                    FirstGatewayAddress;
                ULONG                   Ipv4Metric;
                ULONG                   Ipv6Metric;
                IF_LUID                 Luid;
                SOCKET_ADDRESS          Dhcpv4Server;
                NET_IF_COMPARTMENT_ID   CompartmentId;
                NET_IF_NETWORK_GUID     NetworkGuid;
                NET_IF_CONNECTION_TYPE  ConnectionType;
                TUNNEL_TYPE             TunnelType;
                SOCKET_ADDRESS          Dhcpv6Server;
                BYTE
                    Dhcpv6ClientDuid[MAX_DHCPV6_DUID_LENGTH]
            ;
```

```
    ULONG               Dhcpv6ClientDuidLength;
    ULONG               Dhcpv6Iaid;
    PIP_ADAPTER_DNS_SUFFIX      FirstDnsSuffix;
} IP_ADAPTER_ADDRESSES_LH, *PIP_ADAPTER_ADDRESSES_LH;
```

e) IpReleaseAddress function (iphlpapi.h)

- Hàm IpReleaseAddress giải phóng một địa chỉ IPv4 có được trước đó thông qua Giao thức cấu hình máy chủ động (DHCP).
- Cú Pháp:

```
IPHLPAPI_DLL_LINKAGE DWORD IpRenewAddress(
        PIP_ADAPTER_INDEX_MAP AdapterInfo
);
```

Trong đó:

- ❖ **AdapterInfo**: Con trỏ đến cấu trúc **IP_ADAPTER_INDEX_MAP** chỉ định bộ điều hợp được liên kết với địa chỉ IPv4 để giải phóng.

- Cấu Trúc **IP_ADAPTER_INDEX_MAP:**

```
typedef struct _IP_ADAPTER_INDEX_MAP {
    ULONG Index;
    WCHAR Name[MAX_ADAPTER_NAME];
} IP_ADAPTER_INDEX_MAP, *PIP_ADAPTER_INDEX_MAP;
```

f) IpRenewAddress function (iphlpapi.h)

- Các IpRenewAddress chức năng canh tân một thuê trên một địa chỉ IPv4 thu được trước đó thông qua Dynamic Host Configuration Protocol (DHCP).

- Cú Pháp:

```
IPHLPAPI_DLL_LINKAGE DWORD IpRenewAddress(
        PIP_ADAPTER_INDEX_MAP AdapterInfo
);
```

Trong đó:

- **AdapterInfo**: Một con trỏ đến cấu trúc **IP_ADAPTER_INDEX_MAP** chỉ định bộ điều hợp được liên kết với địa chỉ IP để gia hạn.

- Cấu Trúc **IP_ADAPTER_INDEX_MAP:**

```
typedef struct _IP_ADAPTER_INDEX_MAP {
        ULONG Index;
        WCHAR Name[MAX_ADAPTER_NAME];
} IP_ADAPTER_INDEX_MAP, *PIP_ADAPTER_INDEX_MAP;
```

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

1. Môi trường

1.1. Visual Studio:

- Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý**.**
- Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

2. Triển khai

- Ipconfig.cpp: chứa chương trình my_ipconfig gồm có các chức năng :
    - Ipconfig
    - Ipconfig/all
    - Ipconfig /renew
    - Ipconfig /release

3. Kết quả:

```
Local Area Connection* 1
       Connection-specific DNS Suffix  .:
       IPv6 Address  . . . . . . . . . . . . : fe80::7585:b5b3:460d:3804
       IPv4 Address  . . . . . . . . . . . .: 169.254.56.4
       Subnet Mask . . . . . . . . . . . . : 0.0.0.0
       Default Gateway . . . . . . . . . . : 0.0.0.0


Local Area Connection* 2
       Connection-specific DNS Suffix  .:
       IPv6 Address  . . . . . . . . . . . . : fe80::408b:5c3c:e5b0:6659
       IPv4 Address  . . . . . . . . . . . .: 169.254.102.89
       Subnet Mask . . . . . . . . . . . . : 0.0.0.0
       Default Gateway . . . . . . . . . . : 0.0.0.0


VMware Network Adapter VMnet1
       Connection-specific DNS Suffix  .:
       IPv6 Address  . . . . . . . . . . . . : fe80::68:e815:f87b:16e9
       IPv4 Address  . . . . . . . . . . . .: 169.254.22.233
       Subnet Mask . . . . . . . . . . . . : 255.255.0.0
       Default Gateway . . . . . . . . . . : 0.0.0.0


VMware Network Adapter VMnet8
       Connection-specific DNS Suffix  .:
       IPv6 Address  . . . . . . . . . . . . : fe80::42c:41ad:9853:2f2a
       IPv4 Address  . . . . . . . . . . . .: 169.254.47.42
       Subnet Mask . . . . . . . . . . . . : 255.255.0.0
       Default Gateway . . . . . . . . . . : 0.0.0.0


VMware Network Adapter VMnet0
       Connection-specific DNS Suffix  .:
       IPv6 Address  . . . . . . . . . . . . : fe80::5c13:3d10:6c64:f911
       IPv4 Address  . . . . . . . . . . . .: 169.254.249.17
       Subnet Mask . . . . . . . . . . . . : 255.255.0.0
       Default Gateway . . . . . . . . . . : 0.0.0.0


Wi-Fi
       Connection-specific DNS Suffix  .:
       IPv6 Address  . . . . . . . . . . . . : fe80::4944:4525:bd9d:96b1
       IPv4 Address  . . . . . . . . . . . .: 192.168.1.147
       Subnet Mask . . . . . . . . . . . . : 255.255.255.0
       Default Gateway . . . . . . . . . . : 192.168.1.1


Menu:
       Option 1: ipconfig
       Option 2: ipconfig /all
       Option 3: ipconfig /renew
       Option 4: ipconfig /release
press 0 to exit
Option: _
```

Hình 9. Giao Diện IpConfig

```
        NetBIOS over Tcpip. . . . . : Enable



VMware Network Adapter VMnet0
        Connection-specific DNS Suffix  .:
        Description. . . . . .: VMware Virtual Ethernet Adapter for VMnet0
        Physical Address. . . . . : 00-50-56-C0-00-00
        DHCP Enabled. . . . . : YES
        IPv6 Address  . . . . . : fe80::5c13:3d10:6c64:f911
        IPv4 Address  . . . . . .: 169.254.249.17
        Subnet Mask . . . . . : 255.255.0.0
        Default Gateway . . . . . : 0.0.0.0
        DHCP Server . . . . . : 0.0.0.0

        Lease Obtained: Thu Jan  1 07:00:00 1970
        Lease Expires:  Thu Jan  1 07:00:00 1970
        DNS Servers . . . . . :192.168.1.1
        NetBIOS over Tcpip. . . . . : Enable



Wi-Fi
        Connection-specific DNS Suffix  .:
        Description. . . . . .: Realtek RTL8821CE 802.11ac PCIe Adapter
        Physical Address. . . . . : 70-66-55-81-CE-A5
        DHCP Enabled. . . . . : YES
        IPv6 Address  . . . . . : fe80::4944:4525:bd9d:96b1
        IPv4 Address  . . . . . .: 192.168.1.147
        Subnet Mask . . . . . : 255.255.255.0
        Default Gateway . . . . . : 192.168.1.1
        DHCP Server . . . . . : 192.168.1.1

        Lease Obtained: Thu Jan  1 07:00:00 1970
        Lease Expires:  Invalid Argument to _localtime32_s
        DNS Servers . . . . . :192.168.1.1
        NetBIOS over Tcpip. . . . . : Enable



Menu:
        Option 1: ipconfig
        Option 2: ipconfig /all
        Option 3: ipconfig /renew
        Option 4: ipconfig /release
press 0 to exit
Option:
```

Hình 10. Giao diện IpConfig/all

```
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::68:e815:f87b:16e9
        IPv4 Address  . . . . . . . . . . . .: 169.254.22.233
        Subnet Mask . . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0



VMware Network Adapter VMnet8
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::42c:41ad:9853:2f2a
        IPv4 Address  . . . . . . . . . . . .: 169.254.47.42
        Subnet Mask . . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0



VMware Network Adapter VMnet0
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::5c13:3d10:6c64:f911
        IPv4 Address  . . . . . . . . . . . .: 169.254.249.17
        Subnet Mask . . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0



Wi-Fi
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::4944:4525:bd9d:96b1
        Subnet Mask . . . . . . . . . . . . : 0.0.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0




ipconfig /release succeeded.
Menu:
        Option 1: ipconfig
        Option 2: ipconfig /all
        Option 3: ipconfig /renew
        Option 4: ipconfig /release
press 0 to exit
Option:
```

Hình 11. Giao Diện IpConfig/ release

```
VMware Network Adapter VMnet1
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::68:e815:f87b:16e9
        IPv4 Address  . . . . . . . . . . . .: 169.254.22.233
        Subnet Mask . . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0


VMware Network Adapter VMnet8
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::42c:41ad:9853:2f2a
        IPv4 Address  . . . . . . . . . . . .: 169.254.47.42
        Subnet Mask . . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0


VMware Network Adapter VMnet0
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : fe80::5c13:3d10:6c64:f911
        IPv4 Address  . . . . . . . . . . . .: 169.254.249.17
        Subnet Mask . . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . . : 0.0.0.0


Wi-Fi
        Connection-specific DNS Suffix  .:
        IPv6 Address  . . . . . . . . . . . . : 2405:4800:309f:3888:4944:4525:bd9d:96b1
        Temporary IPv6 Address. . . . . . . . .: 2405:4800:309f:3888:e50f:f087:30f9:dc37
        Link-local IPv6 Address  . . . . . . . .: fe80::4944:4525:bd9d:96b1
        IPv4 Address  . . . . . . . . . . . .: 192.168.1.147
        Subnet Mask . . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . . : 192.168.1.1



ipconfig /renew succeeded.
Menu:
        Option 1: ipconfig
        Option 2: ipconfig /all
        Option 3: ipconfig /renew
        Option 4: ipconfig /release
press 0 to exit
```

Hình 12. Giao Diện IpConfig/ renew

CHƯƠNG 4 :KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết Quả Đạt Được

Trong thời gian tìm hiểu, nghiên cứu cơ sở lý thuyết và triển khai ứng dụng công nghệ, đề tài đã đạt được những kết quả sau:

Về mặt lý thuyết, hiểu được cách hoạt động và sử dụng thư viện IP Helper

Về mặt thực tiễn ứng dụng, ứng dụng vào việc viết chương trình my_ipconfig

Tuy nhiên, đề tài còn tồn tại các vấn đề như sau:

- Vấn đề 1: Vẫn còn thiếu một vài thông số của Adapter và Interface
- Vấn đề 2: Giao diện còn đơn giản

2. Hướng Phát Triển

Một số số hướng nghiên cứu và phát triển của đề tài như sau:

- Thiết kế giao diện cho cho chương trình my_ipconfig

KẾT LUẬN CHUNG

Tìm hiểu về IP HELPER và đã xây dựng tương đối đầy đủ chươ trình My ipconfig

Hiểu được kĩ hơn về giải pháp Monitor và việc tranh chấp tài nguyên dùng chung giữa các tiến trình và cách giải quyết những tranh chấp đó.

Xây dựng được chương trình đảm bảo đáp ứng cơ bản những yêu cầu của bài toán

Về mặt thực thi chương trình, chương trình chạy đúng theo mục đích đề ra, đã tạo được giao diện thực thi.

Biết cách trình bày báo cáo, slide, phục vụ cho việc học tập cũng như làm việc sau này.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Đặng Vũ Tùng, *Giáo trình nguyên lý hệ điều hành*, NXB Hà Nôi, 2005.
[2] Trần Hồ Thuỷ Tiên, *Bài giảng Nguyên lý hệ điều hành*, Khoa CNTT trường Đại học Bách khoa Đà nẵng.

Internet

[1] *https://docs.microsoft.com/en-us/windows/win32/iphlp/*

[2] https://www.geeksforgeeks.org/dining-philosophers-solution-using-monitors/l

[3] *https://stackoverflow.com*