**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙢❖🙠

**TIỂU LUẬN**

**HỆ ĐIỀU HÀNH**

**ĐỀ TÀI: THU THẬP THÔNG TIN TIẾN TRÌNH CỦA CPU SỬ DỤNG NODE.JS PACKAGE 'child\_process'**

Giảng viên hướng dẫn: Đặng Thanh Hải

Sinh viên thực hiện: 2111846 – Lê Trần Anh Khôi

**Đà Lạt, tháng 5 năm 2023**

Mục Lục

[I. Mở đầu 4](#_Toc136536404)

[1.1 Lý do chọn đề tài 5](#_Toc136536405)

[1.2 Phương pháp nghiên cứu 5](#_Toc136536406)

[1.3 Nội dung đề tài 5](#_Toc136536407)

[1.3.1 Mục đích nghiên cứu 5](#_Toc136536408)

[1.3.2 Nhiệm vụ nghiên cứu: 5](#_Toc136536409)

[1.3.3 Đối tượng nghiên cứu: 5](#_Toc136536410)

[1.3.4 Phạm vi nghiên cứu: 5](#_Toc136536411)

[II. Khái niệm và cấu trúc của tiến trình 4](#_Toc136536412)

[2.1 Khái niệm tiến trình 5](#_Toc136536413)

[2.2 Cấu trúc của một tiến trình 5](#_Toc136536414)

[III. Node.js và npm 4](#_Toc136536415)

[3.1 Node.js 5](#_Toc136536416)

[3.1.1 Giới thiệu 5](#_Toc136536417)

[3.1.2 Cài đặt Node.js 5](#_Toc136536418)

[3.1.3 Cú pháp và cấu trúc mã Node.js 5](#_Toc136536419)

[3.1.4 Cấu trúc cơ bản: Một ứng dụng Node.js cơ bản thường bao gồm các thành phần sau: 5](#_Toc136536420)

[3.1.5 Các module quan trọng trong Node.js: 5](#_Toc136536421)

[3.2 Quản lý gói với npm 5](#_Toc136536422)

[3.2.1 Giới thiệu 5](#_Toc136536423)

[3.2.2 Các lệnh quan trọng trong npm: 5](#_Toc136536424)

[IV. Thư viện Node.js cho thu thập thông tin CPU 4](#_Toc136536425)

[V. Cách sử dụng Node.js Package để thu thập thông tin tiến trình CPU 4](#_Toc136536426)

[VI. Cấu trúc chương trình thu thập thông tin trong tiểu luận 4](#_Toc136536427)

[6.1 Tổng quan 4](#_Toc136536428)

[6.1.1 Giới thiệu 5](#_Toc136536429)

[6.1.2 Cấu trúc chung: 5](#_Toc136536430)

[6.2 Phần mã chính 4](#_Toc136536431)

[6.2.1 processes.js 5](#_Toc136536432)

[6.2.2 index.js và các thông tin lấy từ mã 5](#_Toc136536433)

[VII. Ứng dụng của việc thu thập thông tin tiến trình CPU 4](#_Toc136536434)

[VIII. Kết luận 4](#_Toc136536435)

[IX. Tài liệu tham khảo 4](#_Toc136536436)

# Mở đầu

## Lý do chọn đề tài

Khác với trước kia ngôn ngữ lập trình ngày nay đã trở nên đa dạng và phổ biến hơn rất nhiều, nhiều công cụ, thư viện và ngôn ngữ mới tạo ra để thực hiện các công việc cụ thể một cách nhanh chóng, hiệu quả và dễ dàng hơn. Đặc biệt là sự lên ngôi của môi trường web dần trở thành xu hướng phổ biến toàn cầu do sự tiện lợi và có thể truy cập đa nền tảng, truy cập từ xa một cách dễ dàng. Vì vậy em chọn đề tài này để tìm hiểu cách ứng dụng những thư viện và công cụ mới liên quan đến môi trường web để thực hiện dễ dàng những công việc mà trước đây khá phức tạp để thực hiện như lấy thông tin của các tiến trình CPU.

## Phương pháp nghiên cứu

Trong quá trình nghiên cứu em đã dùng những phương pháp sau:

* Phương pháp thu thập dữ liệu.
* Phương pháp phân tích.
* Phương pháp đưa ra kết luận.

## Nội dung đề tài

### 1.3.1 Mục đích nghiên cứu

Đưa ra một mẫu chương trình web cơ bản kết hợp Node.js, Express.js và React có chức năng lấy và hiển thị thông tin của các tiến trình CPU lên môi trường web.

### 1.3.2 Nhiệm vụ nghiên cứu:

tìm hiểu những vấn đề, cách sử dụng cơ bản của Node.js, Express.js và React, thư viện hay npm package để lấy được thông tin

### 1.3.3 Đối tượng nghiên cứu:

Node.js để làm BackEnd, Express.js và React để làm FrontEnd.

### 1.3.4 Phạm vi nghiên cứu:

Các Hàm được xây sẵn của Node.js, Express.js và React.

# Khái niệm và cấu trúc của tiến trình

## 2.1 **Khái niệm tiến trình**

Một tiến trình của CPU(Bộ xử lý trung tâm) là một chương trình đang được thực thi. Một tiến trình sẽ cần các tài nguyên máy tính nhất định, chẳng hạn như thời gian CPU, bộ nhớ, tệp và thiết bị I/O để hoàn thành nhiệm vụ của mình. Các tài nguyên này thường được phân bổ cho tiến trình trong khi nó đang thực thi.

Một tiến trình là đơn vị công việc trong hầu hết các hệ thống. Hệ thống bao gồm một tập hợp các tiến trình: các tiến trình của hệ điều hành thực thi mã hệ thống và tiến trình của người dùng thực thi mã người dùng. Tất cả các quá trình này có thể thực hiện đồng thời.

Các hệ điều hành hiện đại hỗ trợ các tiến trình có nhiều luồng điều khiển. Trên các hệ thống có nhiều lõi xử lý phần cứng, các luồng này có thể chạy song song.

Một trong những khía cạnh quan trọng nhất của một hệ điều hành là cách nó

lên lịch cho các luồng vào các lõi xử lý có sẵn. Một số lựa chọn cho thiết kế bộ thiết lập lịch cho CPU có sẵn cho các lập trình viên.

## 2.2 Cấu trúc của một tiến trình

* 1. **Phần ngăn xếp(Stack session):** lưu trữ dữ liệu tạm thời khi gọi hàm (chẳng hạn như tham số hàm, địa chỉ trả về và biến cục bộ).
  2. **Phần heap(Heap session):** bộ nhớ được cấp phát động trong thời gian chạy chương trình.
  3. **Phần dữ liệu(Data session):**

các biến toàn cục.

* 1. **Phần văn bản(Text session):**

mã thực thi.

**Hình 2.1** Bố cục của một tiến trình trong bộ nhớ.

kích thước của phần văn bản và dữ liệu là cố định vì kích thước của chúng không thay đổi trong thời gian chạy chương trình. Tuy nhiên, phần ngăn xếp và heap có thể thu lại và mở rộng linh hoạt trong quá trình thực thi chương trình.

Mỗi khi một hàm được gọi, một bản ghi kích hoạt chứa các tham số hàm, biến cục bộ và địa chỉ trả về được đẩy lên ngăn xếp; khi quyền điều khiển được trả về từ hàm, bản ghi kích hoạt sẽ được bật ra khỏi ngăn xếp. Tương tự, heap

sẽ tăng lên khi bộ nhớ được cấp phát động và sẽ co lại khi bộ nhớ được trả về hệ thống.

Mặc dù các phần ngăn xếp và heap mở rộng về phía nhau, nhưng hệ điều hành phải đảm bảo chúng không chồng lên nhau. Bản thân chương trình không phải là một tiến trình. Một chương trình là một thực thể thụ động, chẳng hạn như một tệp chứa danh sách các chỉ dẫn được lưu trữ trên đĩa (thường được gọi là tệp thực thi hay **executable fil**). Ngược lại, một tiến trình là một thực thể đang hoạt động với bộ đếm chương trình chỉ định lệnh tiếp theo sẽ thực thi và một tập hợp các tài nguyên liên quan.

Một chương trình trở thành một tiến trình khi một tập tin thực thi được nạp vào bộ nhớ. Hai kỹ thuật phổ biến để tải các tệp thực thi là nhấp đúp vào biểu tượng đại diện cho tệp thực thi và nhập tên của tệp thực thi trên dòng lệnh (như trong prog.exe hoặc a.out).

# Node.js và npm

## Node.js

### 3.1.1 Giới thiệu

Node.js là một nền tảng phát triển ứng dụng được xây dựng trên Chrome's V8 JavaScript engine. Nó cho phép chúng ta xây dựng các ứng dụng mạng phía máy chủ hiệu quả và mở rộng. Node.js sử dụng mô hình I/O không đồng bộ (asynchronous) và sự kiện (event-driven) giúp xử lý đồng thời hàng ngàn kết nối mà không gây block chương trình.

### 3.1.2 Cài đặt Node.js

**- Tải Node.js:** Node.js có sẵn miễn phí để tải về từ trang web chính thức (https://nodejs.org). Trang web cung cấp các phiên bản Node.js cho các hệ điều hành phổ biến như Windows, macOS và Linux.

**- Cài đặt Node.js:** Sau khi tải xuống gói cài đặt Node.js, chúng ta có thể tiến hành cài đặt bằng cách chạy tệp cài đặt và làm theo hướng dẫn trên màn hình. Quá trình cài đặt sẽ cài đặt Node.js và npm (Node Package Manager) cùng với nó.

### 3.1.3 Cú pháp và cấu trúc mã Node.js

**- Cú pháp:** Mã Node.js được viết bằng JavaScript, do đó, cú pháp của nó giống với JavaScript. Một tệp mã Node.js thường có phần mở rộng là .js.

### 3.1.4 Cấu trúc cơ bản: Một ứng dụng Node.js cơ bản thường bao gồm các thành phần sau:

1. Khai báo và sử dụng các module: Sử dụng require để nhập các module và sử dụng chúng trong ứng dụng.
2. Khởi tạo server: Tạo một server HTTP hoặc HTTPS để lắng nghe các yêu cầu từ client.
3. Xử lý yêu cầu: Xử lý các yêu cầu từ client bằng cách định nghĩa các route và xử lý logic phù hợp.
4. Gửi phản hồi: Gửi phản hồi từ server đến client bằng cách sử dụng phương thức send hoặc write.

### 3.1.5 Các module quan trọng trong Node.js:

* http: Module http cung cấp các công cụ để tạo và làm việc với server HTTP. Nó cho phép chúng ta tạo server, xử lý yêu cầu và gửi phản hồi.
* fs: Module fs cung cấp các công cụ để làm việc với tệp và thư mục trên hệ thống tệp. Nó cho phép chúng ta đọc, ghi và thực hiện các thao tác khác trên tệp và thư mục.
* path: Module path cung cấp các công cụ để làm việc với đường dẫn tệp và thư mục. Nó cung cấp các phương thức để nối, phân tích và chuyển đổi đường dẫn.

## Quản lý gói với npm

### 3.2.1 Giới thiệu

npm (Node Package Manager) là trình quản lý gói mạnh mẽ trong Node.js. Nó cho phép chúng ta cài đặt, cập nhật và quản lý các gói phụ thuộc trong dự án.

### 3.2.2 Các lệnh quan trọng trong npm:

* **npm init**: Tạo một tệp package.json mới để quản lý dự án.
* **npm install <package>**: Cài đặt một gói phụ thuộc mới vào dự án.
* **npm install**: Cài đặt tất cả các gói phụ thuộc được định nghĩa trong tệp package.json.
* **npm update <package>**: Cập nhật một gói phụ thuộc lên phiên bản mới nhất.
* **npm uninstall <package>**: Gỡ bỏ một gói phụ thuộc khỏi dự án.

# Thư viện Node.js cho thu thập thông tin CPU

Node.js có rất nhiều thư viện được phát triển để thu thập thông tin CPU của hệ thống. Một số thư viện phổ biến bao gồm:

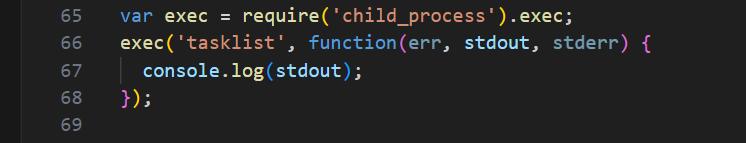
os-utils: Thư viện này cung cấp một API đơn giản để lấy thông tin CPU sử dụng Node.js. Nó cung cấp các phương thức để lấy thông tin về tải CPU, bộ nhớ, tiến trình và hệ thống.

pidusage: Thư viện này cho phép thu thập thông tin tiến trình của CPU bằng cách sử dụng PID (Process ID). Nó cung cấp các phương thức để lấy thông tin về tải CPU, bộ nhớ và xử lý IO của một tiến trình cụ thể.

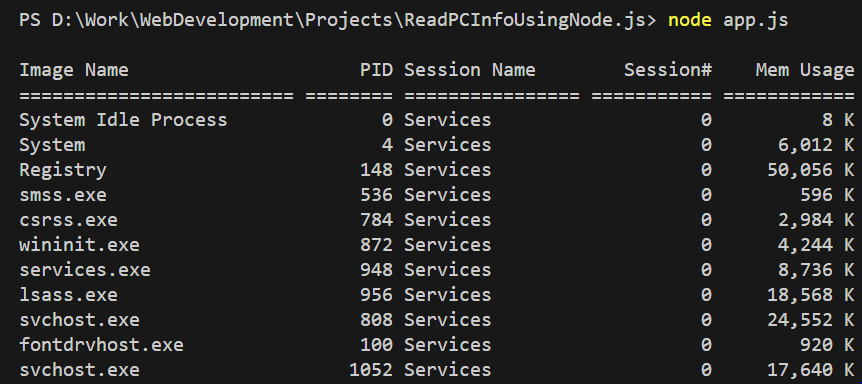
# Cách sử dụng Node.js Package để thu thập thông tin tiến trình CPU

Để sử dụng Node.js Package để thu thập thông tin tiến trình CPU, chúng ta cần tuân thủ các bước sau:

1. Cài đặt Node.js và npm
2. Cài đặt thư viện: Sau khi cài đặt Node.js và npm, chúng ta có thể sử dụng lệnh npm để cài đặt thư viện thu thập thông tin tiến trình CPU như os-utils, pidusage và child process. Ví dụ: **npm install child\_process**
3. Import và sử dụng thư viện: Tiếp theo, chúng ta cần nhập thư viện vào mã nguồn của dự án Node.js và sử dụng các phương thức cung cấp để thu thập thông tin tiến trình CPU. Ví dụ:



Ouput:



Trên đây là một ví dụ đơn giản về cách sử dụng thư viện **child\_process** để lấy thông tin về tải CPU.

# Cấu trúc chương trình thu thập thông tin trong tiểu luận

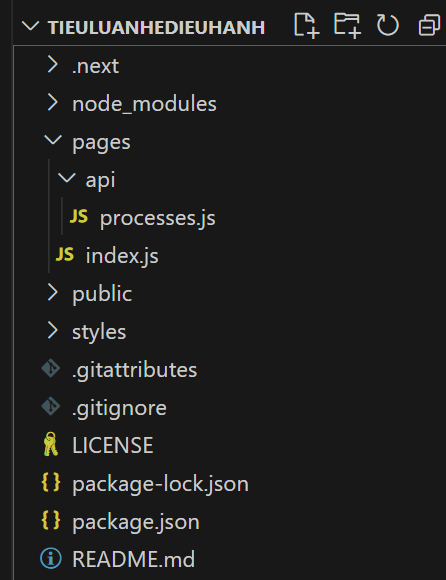
## 6.1 Tổng quan

### 6.1.1 Giới thiệu

Chương trình được xây dựng sử dụng Next.js,

Một framework mã nguồn mở phổ biến để xây dựng các ứng dụng React. Nó được xây dựng dựa trên React và cung cấp các tính năng và khả năng bổ sung để đơn giản hóa việc phát triển các ứng dụng web hiện đại.

### 6.1.2 Cấu trúc chung:



**a) /pages**

Thư mục này là trung tâm của dự án Next.js. Nó chứa các trang trong ứng dụng, về cơ bản là các thành phần React đại diện cho các tuyến khác nhau của trang web. Mỗi tệp trong thư mục này tương ứng với một tuyến cụ thể. Ví dụ: index.js đại diện cho trang chủ, about.js đại diện cho trang giới thiệu, v.v. Next.js sử dụng hệ thống định tuyến dựa trên tệp, trong đó mỗi tệp trang sẽ tự động trở thành một tuyến đường.

**b) index.js**

index.js đại diện cho trang chủ,

**c) /Public**

Thư mục chung được sử dụng để lưu trữ các tệp tĩnh như hình ảnh, phông chữ hoặc bất kỳ nội dung nào khác cần được truy cập công khai. Bất kỳ tệp nào được đặt trong thư mục này đều có thể được tham chiếu bằng / URL.

**d) /components**

Thư mục này được sử dụng để lưu trữ các thành phần React có thể tái sử dụng được sử dụng trên nhiều trang trong ứng dụng. Bằng cách tổ chức các thành phần trong thư mục này, có thể dễ dàng quản lý và sử dụng lại chúng trong các phần khác nhau của ứng dụng.

**e) /styles**

Thư mục này được sử dụng để lưu trữ các tệp CSS hoặc SCSS để tạo kiểu cho các thành phần và trang. Next.js hỗ trợ các tùy chọn tạo kiểu khác nhau, bao gồm các mô-đun CSS, thư viện CSS-in-JS hoặc biểu định kiểu chung. Có thể sắp xếp các biểu định kiểu của mình dựa trên sở thích của mình hoặc phương pháp tạo kiểu cụ thể mà người phát triển chọn tuân theo**.**

**f) /api**

Thư mục này là tùy chọn và được sử dụng để xác định các chức năng serverless hoặc điểm cuối API có thể được gọi từ phía máy khách. Next.js cho phép tạo các tuyến API bằng thư mục /api, trong đó mỗi tệp đại diện cho một điểm cuối API. Các tuyến API này có thể được sử dụng để xử lý việc gửi biểu mẫu, tìm nạp dữ liệu hoặc thực hiện các hoạt động phía máy chủ.

**g) package.json**

Tệp này là tệp cấu hình gói npm tiêu chuẩn. Nó chứa siêu dữ liệu về dự án, cũng như danh sách các thành phần phụ thuộc và tập lệnh cần thiết để xây dựng, chạy và triển khai ứng dụng Next.js.

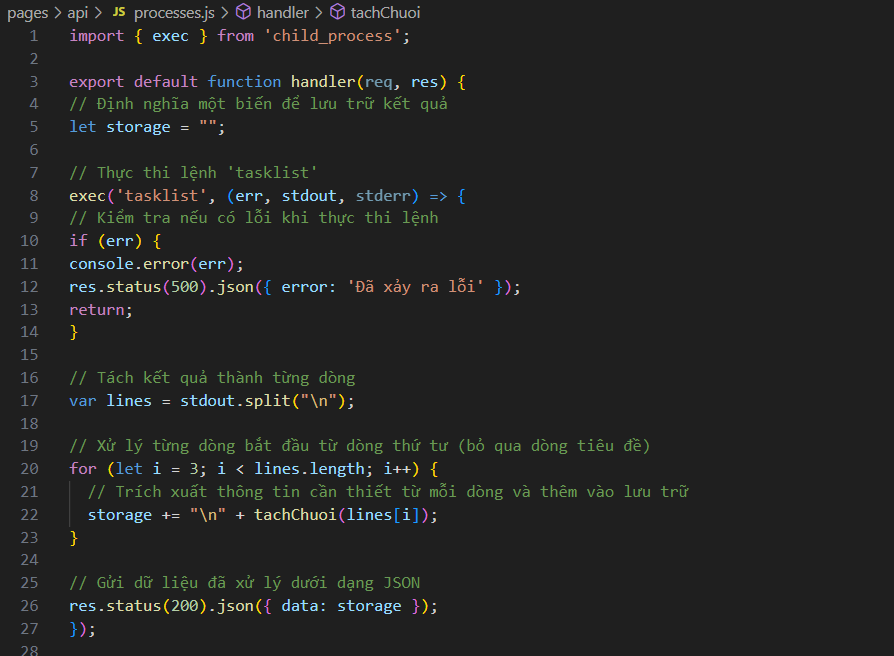
**h) Các tệp khác:**

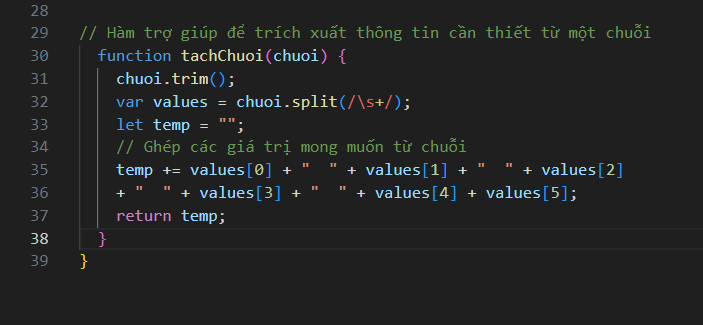
Ngoài các thư mục được đề cập ở trên, dự án Next.js có thể bao gồm các tệp khác, chẳng hạn như .env cho các biến môi trường, .gitignore để chỉ định các tệp bị kiểm soát phiên bản bỏ qua và các tệp cấu hình khác dựa trên các công cụ cụ thể hoặc thư viện sử dụng trong dự án.

## 6.2 Phần mã chính

### 6.2.1 processes.js

**Đây là file lập trình backend của web app:**

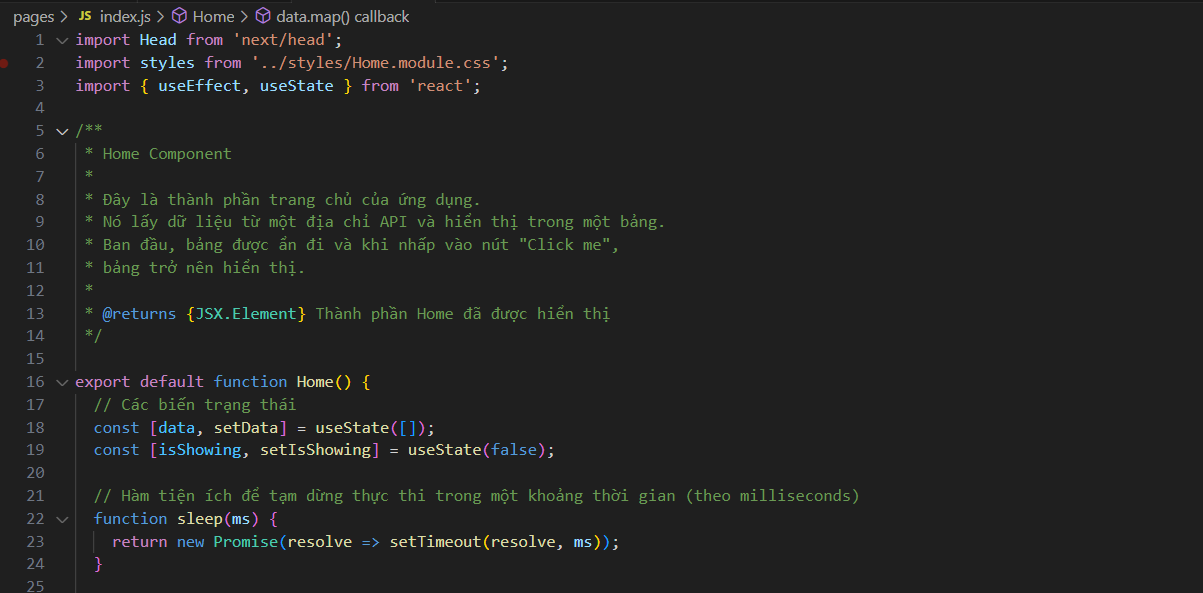


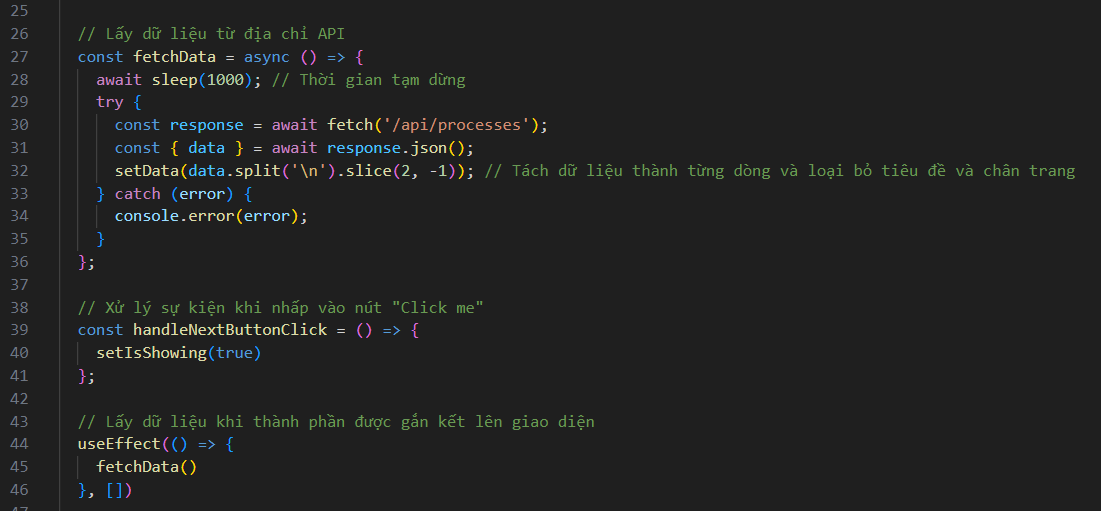


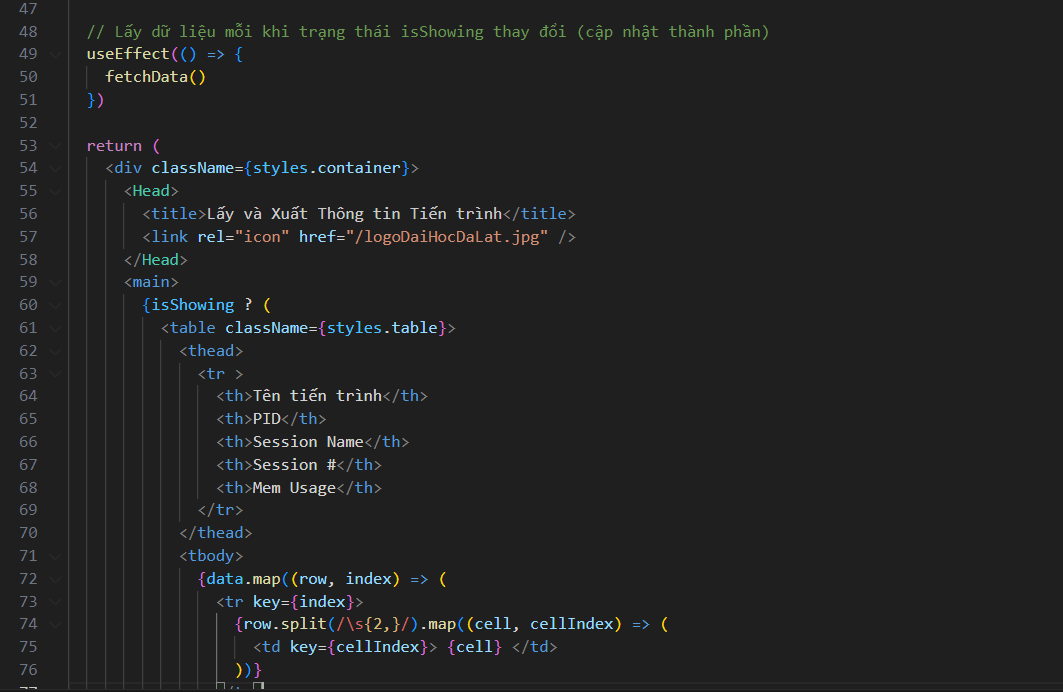
### 6.2.2 index.js và các thông tin lấy từ mã

#### a) giải thích mã

**Đây là phần mã front end chính xữ lý dữ liệu nhận được từ back end và hiển thị kết quả lên trang web:**

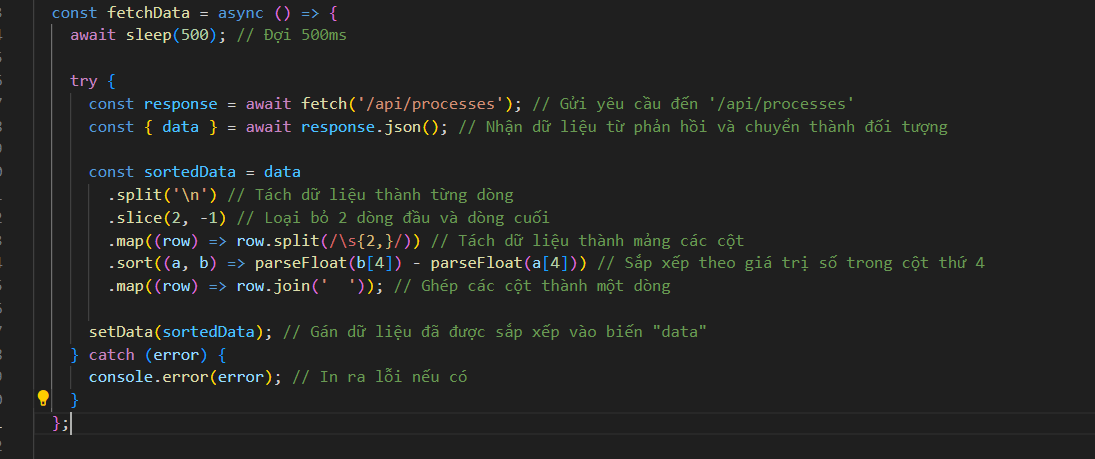








**Sắp xếp dữ liệu theo dung lượng bộ nhớ sử dụng:**



#### b) Các thông tin được hiển thị qua mã:

##### Tên tiến trình:

Tên tiến trình (process name) là một thuộc tính quan trọng của một tiến trình trong hệ điều hành. Nó đại diện cho tên định danh duy nhất của một chương trình hoặc ứng dụng đang chạy trong hệ thống.

Mỗi tiến trình được gắn với một tên tiến trình duy nhất để phân biệt với các tiến trình khác. Tên tiến trình thường được đặt bởi hệ điều hành hoặc ứng dụng khi tiến trình được khởi tạo. Đôi khi tên tiến trình được chỉ định rõ ràng trong mã nguồn chương trình, hoặc nó có thể được tùy chỉnh bởi người dùng hoặc quản trị viên hệ thống.

Tên tiến trình giúp xác định chính xác các tiến trình cụ thể nào đang chạy trên hệ thống. Nó có thể được sử dụng để quản lý, tương tác và kiểm soát các tiến trình trong hệ thống, bao gồm khởi động, dừng, theo dõi và gỡ lỗi chúng.

Một số ví dụ về tên tiến trình có thể bao gồm "chrome.exe" cho trình duyệt Google Chrome, "firefox.exe" cho trình duyệt Mozilla Firefox, "explorer.exe" cho trình quản lý tệp và thư mục Windows Explorer, và "word.exe" cho ứng dụng Microsoft Word.

##### PID:

PID là viết tắt của Process ID (Process Identifier), hay còn gọi là số định danh tiến trình. Mỗi tiến trình trong hệ điều hành được gắn với một PID duy nhất để phân biệt nó với các tiến trình khác.

PID được hệ điều hành sử dụng để quản lý, kiểm soát và theo dõi các tiến trình trong hệ thống. Nó là một số nguyên không âm và thường có giá trị duy nhất cho mỗi tiến trình. Khi một tiến trình được khởi chạy, hệ điều hành tạo một PID cho tiến trình đó và sử dụng nó để xác định và tham chiếu tiến trình trong quá trình thực thi.

PID được sử dụng trong nhiều hoạt động liên quan đến quản lý tiến trình, chẳng hạn như khởi động, dừng, tạm dừng, tiếp tục, kiểm tra trạng thái và gỡ lỗi tiến trình. Khi bạn muốn tương tác với một tiến trình cụ thể, bạn có thể sử dụng PID để xác định tiến trình đó và thực hiện các thao tác tương ứng.

Một PID có thể có giá trị từ 0 đến một giới hạn tối đa tùy thuộc vào hệ điều hành. Thông thường, các PID nhỏ hơn thường được sử dụng cho các tiến trình hệ thống và các tiến trình đặc biệt, trong khi các PID lớn hơn thường được sử dụng cho các tiến trình người dùng thông thường.

##### Session Name:

Phiên làm việc (session) là một khái niệm trong hệ điều hành để đại diện cho một phiên làm việc của một người dùng trên máy tính. Nó thường được tạo ra khi người dùng đăng nhập vào hệ thống và bắt đầu sử dụng các tài nguyên và ứng dụng trên máy tính.

Mỗi phiên làm việc có một số phiên làm việc duy nhất (session number) để phân biệt với các phiên làm việc khác trên cùng một máy tính. Số phiên làm việc thường được sử dụng để quản lý và xác định quyền truy cập của người dùng trong hệ thống.

Phiên làm việc đại diện cho một môi trường làm việc riêng biệt cho người dùng, trong đó các ứng dụng và tác vụ của người dùng được thực thi và quản lý. Mỗi phiên làm việc có thể có các tài nguyên và quyền truy cập khác nhau, và thông tin về phiên làm việc như tên và số phiên làm việc có thể được sử dụng để xác định người dùng và quản lý phiên làm việc trên hệ thống.

##### Session #:

Session # (Session Number) là một thuộc tính được sử dụng để định danh và phân biệt các phiên làm việc trong hệ điều hành. Nó được sử dụng để xác định số phiên làm việc đang được sử dụng bởi mỗi người dùng trong hệ thống.

Trên một hệ điều hành, khi một người dùng đăng nhập, một phiên làm việc mới được tạo ra và được gắn kết với người dùng đó. Mỗi phiên làm việc được gán một số phiên làm việc duy nhất, được gọi là Session #. Số phiên làm việc thường được sử dụng để xác định và quản lý phiên làm việc của mỗi người dùng, đồng thời giúp phân biệt các phiên làm việc khác nhau trên cùng một hệ thống.

Session # có thể được sử dụng để theo dõi và quản lý quyền truy cập của người dùng trong hệ thống, cũng như xác định các tài nguyên và quyền hạn liên quan đến từng phiên làm việc cụ thể. Điều này giúp hệ thống phân biệt và quản lý các tác vụ, ứng dụng và tài nguyên được sử dụng bởi từng phiên làm việc riêng biệt của người dùng.

##### Mem Usage:

Mem Usage (Memory Usage) là một thuộc tính được sử dụng để chỉ ra lượng bộ nhớ mà một tiến trình đang sử dụng trong hệ thống.

Trên hệ điều hành, mỗi tiến trình khi chạy sẽ sử dụng một phần bộ nhớ để lưu trữ dữ liệu và các tài nguyên cần thiết. Mem Usage là một thông số đo lường lượng bộ nhớ mà tiến trình hiện tại đang sử dụng. Thông thường, Mem Usage được tính bằng đơn vị kilobyte (KB), megabyte (MB) hoặc gigabyte (GB) để đo lường lượng bộ nhớ.

Mem Usage cho thấy khối lượng bộ nhớ thực tế đang được tiến trình sử dụng và thường bao gồm bộ nhớ thực và bộ nhớ ảo. Bộ nhớ thực đại diện cho phần bộ nhớ vật lý thực tế đã được cấp phát cho tiến trình, trong khi bộ nhớ ảo đại diện cho không gian bộ nhớ ảo mà tiến trình có thể sử dụng, bao gồm cả bộ nhớ vật lý và bộ nhớ trên ổ đĩa cứng.

Thông tin về Mem Usage có thể hữu ích để giám sát và quản lý việc sử dụng bộ nhớ của các tiến trình trong hệ thống. Nó có thể giúp người dùng và quản trị viên đánh giá tải trọng bộ nhớ, tìm hiểu về hiệu suất hoạt động của tiến trình và tìm kiếm các vấn đề liên quan đến việc sử dụng bộ nhớ quá mức.

# Ứng dụng của việc thu thập thông tin tiến trình CPU

Việc thu thập thông tin tiến trình CPU có nhiều ứng dụng hữu ích, bao gồm:

**- Giám sát hiệu suất hệ thống:** Thu thập thông tin tiến trình CPU giúp chúng ta đánh giá hiệu suất của hệ thống và xác định các khâu xử lý có thể được tối ưu hóa.

**- Phân tích sự cố và gỡ lỗi:** Thông tin về tải CPU có thể giúp chúng ta xác định các tiến trình hoặc quá trình đang sử dụng nhiều tài nguyên CPU và gây ra sự cố hoặc gây ra sự chậm trễ trong hệ thống.

**- Quản lý tài nguyên:** Thông tin về tải CPU có thể giúp chúng ta quản lý tài nguyên hệ thống một cách hiệu quả và phân phối tài nguyên sao cho công việc cần thiết được thực hiện một cách tốt nhất.

# Kết luận

Trong bài tiểu luận này, chúng ta đã khám phá cách sử dụng Node.js Package để thu thập thông tin tiến trình CPU. Việc thu thập thông tin tiến trình CPU đóng vai trò quan trọng trong việc giám sát hiệu suất hệ thống và quản lý tài nguyên. Các thư viện Node.js như os-utils, pidusage và child process… cung cấp các phương thức thuận tiện để thu thập thông tin tiến trình CPU một cách dễ dàng và linh hoạt.

# Tài liệu tham khảo

[1] Giáo trình Hệ điều hành, Đặng Thanh Hải, Khoa Công nghệ thông tin, Trường Đại học Đà Lạt

[2] Abraham Silberschatz, Peter B. Galvin, Greg Gagne (2018). *Wiley.Operating.System.Concepts.10th.Edition*, Wiley, 222 Rosewood Drive, Danvers.

[3] Tài liệu về child\_process <https://nodejs.org/api/child_process.html#child-process>

[4] Tài liệu Node.js: <https://nodejs.org/en/docs>

[5] Tài liệu npm: <https://docs.npmjs.com/>

[6] Tài liệu next.js: <https://nextjs.org/docs>