# [[Học nghề] Sử dụng nguồn](https://codegym.vn/blog/2018/09/17/hoc-nghe-su-dung-nguon/)



## *Cách tốt nhất để chuẩn bị làm một người lập trình là viết các chương trình, và nghiên cứu các chương trình tuyệt vời mà những người khác đã viết ra. Trong trường hợp của tôi, tôi đã đi đến thùng rác tại Trung tâm Khoa học Máy tính và tìm ra danh sách các hệ điều hành của họ.*

*-Bill Gates, Programmers at Work-*

## **Bối cảnh**

Những người mới đến với thế giới nguồn mở thường thấy câu hỏi của họ được trả lời bằng cụm từ “Hãy sử dụng nguồn, Luke”. Điều này thể hiện một sự thật cơ bản về phần mềm: mã là trọng tài cuối cùng. Ý định của người lập trình là không thích hợp nếu nhữ mã không tán thành với điều đó. Chỉ bằng cách đọc mã, một người có thể thực sự hiểu được cả một hệ thống.

## **Vấn đề**

Nếu không có những tấm gương về thực hành tốt để học tập và bắt chước, thì mẫu Thực Hành, Thực Hành, Thực Hành chỉ đào sâu những thói quen xấu mà bạn không biết là bạn có. Nếu bạn không bao giờ đi bộ được một dặm trong đôi giày da đanh của người khác, bạn có thể sẽ tin rằng tất cả các đôi giày đều cần phải có đá bên trong chúng. Vì vậy, làm thế nào để bạn nhận ra nếu công việc của bạn là tốt, trong trường hợp những người xung quanh bạn có thể không có khả năng để phân biệt đâu là mã tốt và đâu là mã xấu?

## **Giải pháp**

Tìm kiếm mã của người khác để đọc. Bắt đầu với các ứng dụng và công cụ mà bạn sử dụng hàng ngày. Là một người học việc, một trong những niềm tin có thể cản trở bạn là ý tưởng rằng những người xây dựng các công cụ mà bạn phụ thuộc vào là khác biệt, đặc biệt hoặc tốt hơn bạn. Bằng cách đọc mã của họ, bạn có thể học cách lập trình giống như họ, và quan trọng hơn, bạn có thể bắt đầu hiểu được các quy trình suy nghĩ tạo ra cơ sở hạ tầng bao quanh bạn.

Khi kiểm tra một dự án mã nguồn mở, bạn có thói quen tải về phiên bản hiện tại của mã nguồn (tốt nhất là từ ​​hệ thống kiểm soát nguồn) để bạn có thể kiểm tra lịch sử cũng như theo dõi tiến trình trong tương lai. Kiểm tra cấu trúc của bộ codebase và cố gắng tìm ra lý do tại sao các tệp tin được đặt theo cách đó. Xem liệu có một lý do đằng sau cách các nhà phát triển đơn bộ hóa các codebase và so sánh nó với cách mà nếu là mình thì bạn sẽ có thể làm như thế nào.

Hãy thử cải tiến mã nguồn các codebase để hiểu tại sao các lập trình viên thực hiện các lựa chọn đó, và để xem kết quả sẽ thế nào nếu bạn là một trong những người viết văn bản mã. Điều này không chỉ giúp bạn hiểu rõ hơn về các dự án này; nó cũng đảm bảo rằng bạn có thể xây dựng các dự án đó. Và nếu bạn tìm ra một cách tốt hơn để làm một việc, thì bạn đang ở một vị trí tốt để đóng góp.

Khi bạn làm việc qua codebase, bạn sẽ không tránh khỏi việc trải qua những quyết định mà bạn bất đồng. Hãy tự hỏi mình liệu có phải các nhà phát triển biết điều gì đó mà bạn không biết hay ngược lại. Hãy xem xét khả năng đây là một thiết kế kế thừa cần phải được sắp xếp lại và suy nghĩ xem liệu việc thực hiện một trò chơi liên quan đến tính năng này có thể dạy cho bạn điều gì không.

Ngoài việc đọc mã của người khác (và, khi được yêu cầu, cung cấp phản hồi), cố gắng tìm những người xung quanh quan tâm đến việc đọc mã nguồn của bạn. Nếu bạn có thể học để nắm lấy phản hồi của họ trong khi lọc ra các đặc tính cá nhân, bạn sẽ trở thành một lập trình viên tốt hơn. Và hãy nhớ rằng, để trở thành một người thợ đào tạo bạn phải giúp đỡ những người khác phát triển, vì vậy hãy trở nên cởi mở với việc đọc mã nguồn của người khác.

Cách tiếp cận phổ biến giữa các lập trình viên mà chúng tôi phỏng vấn liên quan đến việc tham gia nhóm hoặc dự án sử dụng các đánh giá về mã hoặc lập trình theo cặp. Những thực tiễn này tạo ra những môi trường trong đó các lập trình viên có thể an toàn dành thời gian đọc mã của người khác, có người khác đọc mã của mình và học hỏi lẫn nhau. Các nhóm này có xu hướng lập trình ra các chương trình cực kỳ mạnh. Các môi trường khác, chẳng hạn như hầu hết các khoa về khoa học máy tính, có khuynh hướng bỏ qua thực tế rằng các lập trình viên tốn nhiều thời gian đọc mã hơn là viết mã. Họ áp dụng cách tiếp cận này vì làm cho tất cả sinh viên tái phát minh ra bánh xe sẽ tạo ra các đồ tạo tác dễ chấm điểm. Tuy nhiên, đào tạo bản thân để trở nên tốt hơn với công việc chiếm nhiều thời gian nhất trong ngày làm việc của bạn chính là việc tối ưu hoá lợi nhuận lớn hơn trong thời gian dài. Điều này đúng ngay cả khi những người không phải là lập trình viên, những người thường xuyên phụ trách trong những môi trường này không đánh giá cao nó.

Bằng cách đọc nhiều mã tốt, xấu và trung bình của những người khác, bạn có thể bắt đầu tìm hiểu về những biệt ngữ và sự tinh tế của một cộng đồng sử dụng những ngôn ngữ cụ thể của bạn. Theo thời gian, điều này sẽ phát triển khả năng của bạn để có thể đoán được mục đích của người khác từ mã mà họ viết. Nó cũng sẽ giúp bạn học để đối phó với những dịp khi cả hai phân tách ra. Những kỹ năng này sẽ làm cho bạn thành một phần có giá trị hơn trong một nhóm, bởi vì bạn sẽ có thể làm việc trên mã của người khác mà không phải lúc nào cũng phải viết lại chỉ vì bạn không thể hiểu nó có vai trò gì.

Cuối cùng bạn sẽ có được một nhóm các công cụ về thủ thuật và sự tinh vi thu thập từ mã của người khác. Điều này sẽ giúp bạn gọt giũa khả năng giải quyết các vấn đề nhỏ một cách nhanh chóng và dễ dàng, bởi bạn đã từng thấy một vấn đề tương tự trước đây. Bạn sẽ có thể giải quyết những vấn đề mà những người khác cho là không thể vì họ không có được một nhóm công cụ như của bạn.

Hãy xem mã của Linus Torvalds cho hệ thống điều khiển nguồn phân phối Git hoặc bất cứ thứ gì được viết bởi Daniel J. Bernstein (thường được biết đến là djb). Các lập trình viên như Linus và djb tình cờ sử dụng cấu trúc dữ liệu và các thuật toán mà hầu hết chúng ta chưa bao giờ nghe đến. Họ không phải là ảo thuật gia – họ chỉ dành thời gian để xây dựng bộ công cụ lớn hơn và tốt hơn hầu hết những người khác. Lợi thế của mã nguồn mở là hộp công cụ của họ đã mở ra cho bạn để nghiên cứu, và bạn có thể biến các công cụ đó của họ trở thành của riêng mình.

Một trong những vấn đề trong lĩnh vực phát triển phần mềm là việc thiếu giáo viên. Nhưng nhờ có sự gia tăng của các dự án mã nguồn mở trên các trang web như sourceforge.net, GitHub và Google Code, bạn có thể học hỏi từ mẫu đại diện của cộng đồng các lập trình viên trên toàn cầu. Không giống như trong giảng dạy truyền thống, đây không phải là các dự án đồ chơi được thiết kế để minh họa một điểm, bao gồm các phím tắt và “bài tập cho người đọc” khi mọi thứ trở nên khó khăn. Đây là những dự án thực sự giải quyết các vấn đề thực sự và liên tục thay đổi. Bạn có thể theo dõi một dự án khi mà các nhà phát triển của nó học và thích nghi với các yêu cầu mới. Bằng cách nghiên cứu cách thức phát triển của codebase thực sự, bạn có thể nhận thức rõ hơn những ảnh hưởng của các quyết định thiết kế mà không phải tự mình xây dựng hàng trăm dự án phần mềm. Điều này tạo cơ hội để học hỏi từ những sai lầm của người khác và có được kỹ năng quan trọng hơn là đọc mã: khả năng có thể tự học mà không cần người khác dạy.

Trong cuốn *Programmers at Work*, Bill Gates nói rằng “một trong những bài kiểm tra khả năng lập trình tốt nhất là giao cho lập trình viên khoảng 30 trang mã và xem anh ta có thể đọc và hiểu nó nhanh như thế nào.” Ông ấy đã nhận ra một điều rất quan trọng. Những người có thể nhanh chóng tiếp thu kiến ​​thức trực tiếp từ mã nhanh chóng trở thành những lập trình viên giỏi hơn, bởi vì giáo viên của họ là mọi dòng mã được viết ra bởi tất cả các lập trình viên trên đời này.

Cách tốt nhất để tìm hiểu về các mô hình, biệt ngữ, và các phương pháp hay nhất là đọc mã nguồn mở. Xem cách người khác làm việc đó như thế nào. Đó là cách tuyệt vời để tiếp tục theo kịp với thời đại, và cách này lại còn miễn phí nữa.

-ChrisWanstrath trong Keynote tại Ruby Hoedown 2008-

## **Hành động**

Chọn một dự án nguồn mở phức tạp về mặt thuật toán, như hệ thống kiểm soát nguồn, ví dụ như Subversion, Git, hoặc Mercurial. Duyệt nguồn của dự án, ghi lại các thuật toán, cấu trúc dữ liệu và ý tưởng thiết kế mới đối với bạn. Bây giờ, hãy viết một bài đăng trên blog mô tả kiến ​​trúc của dự án và nhấn mạnh những ý tưởng mới mà bạn đã học được. Bạn có thấy có chỗ nào trong công việc hàng ngày của bạn mà những ý tưởng tương tự có thể được áp dụng không?