Lịch thực tập tốt nghiệp tại Công ty TNHH phần mềm FPT Miền Trung

Tuần	Thứ	Nội dung thực tập	Thực hiện
1. *Tìm hiểu công ty*	5	Gặp mặt công ty, nhận tài liệu và nhận nhiệm vụ	Sinh viên được giới hiệu về công ty FPT Software Đà Nẵng.
	6	Tìm hiểu tổng quan về các quy định, quy chế bảo mật thông tin của công ty	Sinh viên tự học theo tài liệu đã được cung cấp
2. *Học tập theo syllabus được giao*	2-6	Sinh viên học và làm bài tập	Biến trong C
			Mảng và các câu lệnh điều và vòng lặp
			Hàm trong C
3. *Học tập theo syllabus được giao*	2 -6	Sinh viên học và làm bài tập	Macro và các phép toán với bits
			Quản lý bộ nhớ và con trỏ cơ bản
4. *Học tập theo syllabus được giao*	2-6	Sinh viên học và làm bài tập	Cấu trúc dữ liệu và thuật toán
			Con trỏ nâng cao
			Tối ưu code trong C
5. *Học tập theo syllabus được giao*	2-6	Sinh viên học và làm bài tập	Phát hiện và sửa lỗi trong C
			Xử lý File
			Unit Test
			Sử dụng Git trong dự án
6. *Thực hiện MOCK project*	2	Đọc tài liệu	Nhóm tự trao đổi
	3	Triển khai dự án	Nhóm tự trao đổi
	4		Nhóm tự trao đổi
	5	Báo cáo tiến độ	Báo cáo tiến độ với cán bộ phụ trách
	6	Cán bộ phụ trách kiểm tra nội dung thực tập	Báo cáo nội dung cho cán bộ phụ trách
7. *Thực hiện MOCK project*	2	Điều chỉnh báo cáo cho phù hợp sau khi nhận xét	Sinh viên làm việc theo nhóm, hoàn thành báo cáo
	3	Công ty nhận xét về chấm điểm quá trình học tập	Sinh viên gửi báo cáo cho công ty để nhận xét
	4	Cán bộ phụ trách kiểm tra vấn đáp	Sinh viên báo cáo qua MSTeam

MỤC LỤC

LỜI CẢM ƠN

Em xin chân thành cảm ơn công ty TNHH phần mềm FPT Miền Trung (hay còn gọi là FPT Software Đà Nẵng) đã tạo môi trường, điều kiện thuận lợi cho em thực tập, rèn luyện và sử dụng các kiến thức đã học, các kỹ năng của mình vào thực tế.

Trong quá trình thực hiện em xin chân thành cảm ơn cô Nguyễn Thị Thanh Quỳnh và thầy Giáp Quang Huy, giảng viên Khoa Điện và anh Trần Văn Khơ, cán bộ Công ty TNHH phần mềm FPT Miền Trung đã tận tình hướng dẫn, truyền đạt những kiến thức, hỗ trợ tinh thần trong suốt quá trình thực tập.

Qua đợt thực tập này, em nhận thấy nhiều kiến thức mới và bổ ích cần học hỏi và áp dụng cho công việc sau này của bản thân. Mặc dù đã có nhiều cố gắng nhưng vì kiến thức bản thân còn hạn chế trong quá trình thực tập không thể tránh những thiếu sót, kính mong nhận được những ý kiến đóng góp của các thầy cô cũng như góp ý của cơ quan để có thể hoàn thiện bản thân hơn. Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Quá trình thực tập tốt nghiệp là một cơ hội quan trọng để sinh viên có thể tìm hiểu, tiếp xúc với thực tế hoạt động của một số công ty trong lĩnh vực lập trình nhúng, tìm hiểu các vị trí công việc sẽ làm trong tương lai và các yêu cầu về kỹ năng chuyên môn cũng như kỹ năng mềm để có thể đáp ứng được các vị trí công việc đó. Qua đợt thực tập tốt nghiệp, sinh viên sẽ có dịnh hướng rõ ràng hơn về nghề nghiệp của chính bản thân mình, có động lực học tập tốt hơn, đam mê nghiên cứu, tìm tòi và phục vụ cho đồ án tốt nghiệp sắp tới.

DANH MỤC CÁC TỪ KHOÁ

FPT Software Đà Nẵng	Công ty TNHH phần mềm FPT Miền Trung
Variable	Biến
Pointer	Con trỏ
Function	Hàm
Life time	Thời gian tồn tại của một biến
Scope	Tầm vực (phạm vi nhận diện) của một biến
Floppy disk	Đĩa mềm
Static	Các file khác không extern được và lớp lưu trữ của các biến đó thay đổi
FAT	File Allocation Table

CHƯƠNG 1: GIỚI THIỆU CÔNG TY FPT SOFTWARE ĐÀ NẴNG

1.1 Giới thiệu về công ty.

Công ty FPT Software Đà Nẵng được thành lập vào ngày 13/8/2005, tính đến nay sau 17 năm hoạt động, FPT Software Đà Nẵng đã không ngừng lớn mạnh và trở thành công ty công nghệ thông tin có quy mô lớn nhất miền Trung.

Năm 2015, giá trị xuất khẩu phần mềm của FPT Software Đà Nẵng đạt khoảng 26 triệu USD, trong đó thị trường Nhật Bản chiếm tỷ trọng trên 70% với tốc độ tăng trưởng bình quân khoảng 35%. Dự kiến đơn vị phấn đấu đạt tăng trưởng doanh thu xuất khẩu phần mềm cho thị trường Nhật Bản trên 40% với việc chuyển sang khai thác những linh vực mới như tài chính, ngân hàng, ô tô…

Hình 1.1. Công ty FPT Software Đà Nẵng

FPT Software là công ty chuyên xuất nhập khẩu dịch vụ và phần mềm, cung cấp các dịch vụ phát triển và bảo trì, triển khai ERP, QA, chuyển đổi ứng dụng, hệ thống nhúng, điện toán di động, điện toán dám mây… trong nhiều linh vực như: Tài chính ngân hàng, Viễn thông, Y tế, Chế tạo, Công nghiệp xe hơi, dịch vụ công…

1.2 Các lĩnh vực hoạt động của công ty

Công nghệ đang được xem là công cụ chiến lược giúp các tổ chức, doanh nghiệp nâng cao năng lực cạnh tranh, thúc đẩy sự đổi mới sáng tạo trong hoạt động.

- Dịch vụ công nghệ thông tin: Dịch vụ điện toán đám mây, Công nghệ di động, Internet of Things (IOTs), Chuyển đổi hệ thống, Phát triển ứng dụng và bảo trì, Phát triển phần mềm cho thiết bị, Kiểm thử phần mềm, Quản lý quy trình doanh nghiệp, Cải tiến quy trình doanh nghiệp.

- Tích hợp hệ thống: Các dịch vụ Tích hợp Hệ thống và thiết bị FPT cung cấp gồm: Dịch vụ hạ tầng CNTT; Thiết kế và xây dựng cơ sở dữ liệu cho doanh nghiệp; Hệ thống mạng và bảo mật; Hệ thống thanh toán, giám sát, các thiết bị đặc chủng cho ngành ngân hàng; các sản phẩm chuyên dụng cho viễn thông, giao thông, hải quan; Dịch vụ triển khai các giải pháp ngân hàng, chứng khoán và viễn thông; Lưu trữ máy chủ; Quản trị cơ sở dữ liệu.

- Giải pháp theo ngành:

Chính phủ: FPT phát triển phần mềm ứng dụng cho khối chính phủ hướng đến mục tiêu tăng cường năng lực và thúc đẩy phát triển chính phủ điện tử tại Việt Nam. Hiện hệ thống thông tin chính quyền điện tử của FPT (FPT. eGov) đang được triển khai tại 21 tỉnh thành trên cả nước, đặc biệt đã đưa Thành phố Hồ Chí Minh trở thành địa phương dẫn dầu về chỉ số ứng dụng ICT toàn quốc.

Tài chính công: FPT đã và đang tham gia triển khai nhiều hệ thống Tài chính công có quy mô lớn tại Việt Nam, Campuchia, Bangladesh, Myanmar... Các dự án tiêu biểu như: Hệ thống quản lý Thuế thu nhập cá nhân, Hệ thống kết nối Thuế - Kho bạc - Tài chính - Hải quan, Hệ thống Thông quan điện tử, Hệ thống Quản lý thuế thu nhập tích hợp và Hệ thống thuế VAT cho Bangladesh, Hệ thống Quản lý Kho bạc nhà nước Campuchia. Hệ thống Chuyển mạch tài chính quốc gia Myanmar...

Ngân hàng – Tài chính: Phần mềm ngân hàng thông minh (SmartBank), Phần mềm kết nối ngân hàng và công ty chứng khoán (FPT.SmartConnect), Hệ thống Báo cáo Ngân hàng Nhà nước (FPT.SBRS), Phần mềm nghiệp vụ công ty tài chính (FPT.SmartFinance) ...

Y tế - Giáo dục: Hướng đến mục tiêu nâng cao chất lượng y tế và giáo dục, FPT đã nghiên cứu và phát triển các phần mềm ứng dụng như Phần mềm quản lý tổng thể bệnh viện (FPT.eHospital), Phần mềm Quản lý phòng khám (FPT.eClinic), Giải pháp quản lý

đào tạo Đại học và Cao đẳng theo hình thức tín chỉ (FPT.EMIS Plus), Hệ thống quản lý thông tin trường quốc tế (FPT.ISM), Hệ thống quản lý đại học (eduprove).

Giao thông vận tải: FPT cung cấp 4 nhóm giải pháp phục vụ Giao thông Vận tải: Pháp luật và Thực thi, Quản lý hạ tầng giao thông, Quản lý thiết bị đầu cuối và Quản lý giao thông (ITS).

Doanh nghiệp: Hệ thống phần mềm quản lý nhân sự và tiền lương (FPT iHRP), Phần mềm quản lý đại hội cổ đông (FPT.DHCD), Hệ thống quản lý khách sạn (FPT iHotel), Hệ thống quản lý hóa đơn (FPT Billing), Giải pháp quản lý hệ thống phân phối (DMS).

1.3 Lĩnh vực văn hóa

Công ty đã tham gia tích cực vào các hoạt động văn hóa xã hội. Ngoài ra, công ty còn tổ chức các câu lạc bộ sở thích: Võ thuật, cuộc thi ảnh chào mừng ngày sinh nhật công ty, các hoạt dộng thể thao, ngày phụ huynh của các nhân viên trong công ty.

1.4 Công việc hằng ngày của kỹ sư khi lên công ty

Hằng ngày, các kỹ sư, nhân viên của FSoft Đà Nẵng sẽ được phân công làm việc trong dự án, làm việc với một khách hàng cụ thể. Mỗi cá nhân khi nhận công việc của dự án phải chịu trách nhiệm về công việc dó. Mỗi dự án diều có một người quản lý để theo dõi quá trình dự án từ đó đánh giá tiến trình dự án có đúng với thời hạn. Từ đó ngăn chặn được các điểm sai, lập tức giúp cho dự án di đúng hướng và hoàn thành đúng kỳ hạn.

Mỗi thành viên trong quá trình làm việc nếu có bất kỳ vấn đề gì thì báo ngay với người quản lý để kịp thời khắc phục. Mỗi cuối tuần hay cuối mỗi giai đoạn sẽ tổ chức họp dự án để đánh giá tiến trình. Cuối mỗi dự án thì sẽ tiến hành dánh giá phê bình thành viên chưa tốt và đồng thời tuyên dương khen thưởng những cá nhân suất xắc.

Các kỹ sư khi mới ra trường có thể làm việc tại các vị trí lập trình viên (Lập trình nhúng, lập trình web …) và nhiều vị trí khác. Yêu cầu cho các kỹ sư khi ra trường phải có kiến thức căn bản về lập trình, khả năng giải quyết vấn đề và suy nghĩ logic, kỹ năng làm việc nhóm tốt, kỹ năng giao tiếp tiếng Anh tốt là một lợi thế, tinh thần học hỏi chịu khó.

1.5. Đề tài nghiên cứu

Để có thể làm việc với các dự án nhúng, sinh viên cần nắm vững các kiến thức căn bản về ngôn ngữ C để làm việc với tầng middle ware như: biến, hàm, con trỏ, danh sách liên kết, các thao tác với File và FAT (File Allocation Table); các kiến thức liên quan đến vi điều khiển: ngắt, timer, ADC, RTOS,…

1.6. Kết luận chương.

Qua quá trình thực tập ở công ty FPT Software Đà Nẵng, em đã được tiếp cận với môi trường làm việc chuyên nghiệp, hiểu được mình sẽ làm như thế nào sau khi ra trường.

Những trải nghiệm ban đầu này giúp em tự tin hơn để chuẩn bị ra trường và tìm công việc cho bản thân. Trong quá trình thực tập, em đã thiết lập dược các mối quan hệ trong nghề nghiệp của mình, điều này cũng rất hữu ích cho em sau khi ra trường.

CHƯƠNG 2: TỔNG QUAN VỀ NGÔN NGỮ C CƠ BẢN

2.1 Tổng quan về biến

2.1.a) Khái niệm về biến

Biến là một định nghĩa giúp người dùng truy xuất vào vùng nhớ thông qua một tên.

Bản chất của khai báo biến là yêu cầu hệ thống cấp phát cho người dùng 1 vùng nhớ để lưu trữ các giá trị đầu vào và đầu ra, trung gian của biến.

2.1.b) Các kiểu dữ liệu cơ bản

Các kiểu dữ liệu cơ bản: int, float, double, char, void.

2.1.c) Các từ khoá dẫn xuất

Từ khoá unsigned áp dụng cho kiểu int, các giá trị được lưu trữ là số nguyên dương.

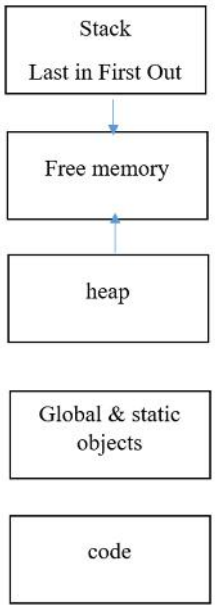
Với từ khoá short, hệ thống sẽ cấp phát nửa vùng nhớ so với kiểu dữ liệu cơ bản. Còn từ khoá long thì sẽ được cấp phát gấp đôi vùng nhớ

2.1.d) Các lớp lưu trữ của 1 biến

Gồm 2 phần là thời gian sống (time line) của 1 biến và tầm vực (scope) của 1 biến.

2.1.e) Các vùng nhớ lưu trữ biến

Hệ thống phân chia vùng nhớ như sau:



Stack là nơi dùng để lưu trữ các biến cục bộ, các biến được khai báo trong một hàm.
Vùng heap dùng để cấp phát động bằng các hàm cấp phát động như relloc, malloc, calloc
Phần lưu trữ các biến global, static, có thể chia thành 2 phần nhỏ: data và bss.
Phần cuối là vùng code, nơi lưu trữ các command, instruction, code.
→Tuỳ vào tính chất của một biến sẽ được định nghĩa bởi 3 thành phần: thời gian sống của một biến, biến đó sẽ được nhận biết ở các phần nào trong chương trình và biến đó được lưu trữ tại vùng nhớ nào trong bộ nhớ của hệ thống.

Hình 2.1. Phân chia vùng nhớ trong RAM

2.1.f) Phân loại biến.
Từ lớp lưu trữ trên, một biến sẽ quy định được lưu trữ ở vùng nào và dựa vào lớp lưu trữ trên để phân biệt được hai loại biến sau: biến global và biến local.

2.1.g) Keyword cho biến trong C
Các từ khoá: static, extern, register, volatile.

2.1.h) Kiểu dữ liệu có cấu trúc trong C
Các bài toán trên thực tế thường không quy ngay về các kiểu dữ liệu cơ bản trong C.

Ví dụ về bài toán quản lý sinh viên của một lớp. Dữ liệu đầu vào là các thành phần của sinh viên như tên tuổi, điểm số, quê quán, giới tính, lúc đó mới là các giá trị có thể quy về các kiểu dữ liệu cơ bản.

Để dễ dàng quản lý các kiểu dữ liệu thực tế, ngôn ngữ lập trình C cung cấp các kiểu dữ liệu có cấu trúc. Các cấu trúc hay sử dụng: structure, union và enum.

2.1.i) Sử dụng Typedef
Ngoài việc định nghĩa một kiểu dữ liệu có cấu trúc mới thì sử dụng từ khoá typedef để định nghĩa 1 kiểu dữ liệu mới thay vì các kiểu dữ liệu cơ bản hay kiểu dữ liệu có cấu trúc, ví dụ: Typedef float deci;

Khi muốn sử dụng kiểu dữ liệu deci thì người dùng chỉ sử dụng kiểu dữ liệu deci để khai báo giống như biến float.

2.1.k) Casting - ép kiểu trong C
Tất cả các đối tượng trong C đều có kiểu dữ liệu cơ bản và các kiểu con trỏ cơ bản.

Trong các biểu thức toán học có thể sử dụng nhiều kiểu dữ liệu cơ bản khác nhau nhưng ở trong ngôn ngữ C thì phải ép kiểu để tính toán được với các kiểu dữ liệu khác.

Có 2 loại ép kiểu là khi biên dịch thì hệ thống sẽ tự ép kiểu cho mình và người dùng chủ động ép kiểu trong biểu thức của mình.

2.2. Cấu trúc điều kiện rẽ nhánh trong C

Các câu lệnh if, else, các câu lệnh switch case.

2.3. Vòng lặp trong C
Vòng lặp trong C là 1 phần mã code chương trình được thực hiện nhiều lần cho đến khi một điều kiện cụ thể được thoả mãn.

Các loại vòng lặp trong C là for, while, do … while.

2.3.2. Từ khoá trong vòng lặp

Các từ khoá: goto, continue, exit(), break, return.

2.4. Hàm trong C

2.4.1. Funtion scope

Global scope có nghĩa là hàm được nhận biết hoặc sử dụng trong nhiều đơn vị biên dịch.

Internal scope có nghĩa là hàm chỉ được sử dụng trong một đơn vị biên dịch.

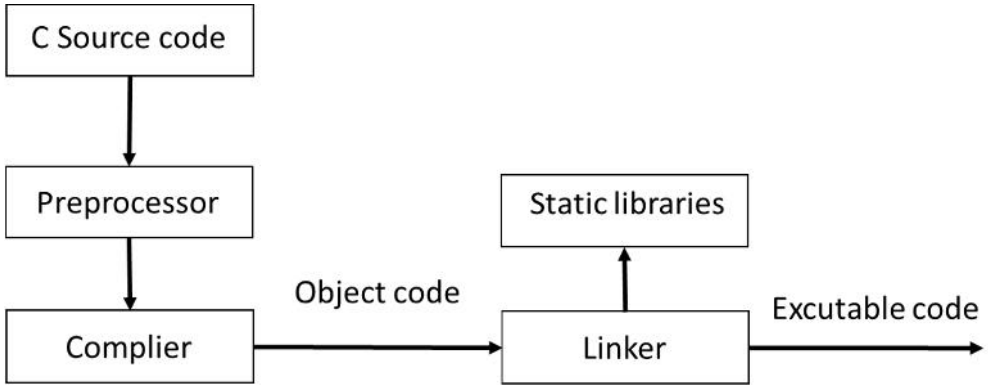
Hàm được biên dịch mà chỉ sử dụng trong file thì chỉ tồn tại internal scope và chỉ được biên dịch nội trong cái đơn vị biên dịch.

Những function nào tồn tại internal scope, tức là chỉ có trong file đó thì ko có symbol vì nó không cần nhảy đến, trỏ đến, không cần con trỏ trỏ đến nó nên cái tên của nó ko mang ý nghĩa là địa chỉ.

2.4.2. Tham trị, tham chiếu hay là pass argument hay reference.

Tham trị là truyền giá trị vào hàm, tham chiếu là truyền địa chỉ vào hàm.

2.5. Bộ tiền xử lý trong C



Hình 2.2. Sơ đồ biên dịch code của trình biên dịch ngôn ngữ C

Bộ tiền xử lý trong C là bộ tiền xử lý macro được sử dụng bởi trình biên dịch C để chuyển đổi source code trước khi nó có thể được biên dịch, là bước đầu tiên tiền xử lý file source code c thành file libraries, excutable code.

Ký hiệu thông báo chỉ thị tiền xử lý là #.

Khi viết một application, một source code.c là đầu vào cho bộ tiền xử lý (preprocessor), có nhiệm vụ xử lý một số chỉ thị tiền xử lý như là #include, #define, … là những line bắt đầu bằng ký hiệu # và bao gồm cả macro.

File source code.c này đã được mở rộng rồi qua bộ tiền xử lý thì nó sẽ làm đầu vào cho complier, complier sẽ biên dịch file mở rộng ấy để tạo ra được những file object code.

Những file object code này thì làm đầu vào cho linker. Nếu application sử dụng 1 số thư viện standard chẳng hạn như là stdio.h hay math. Nếu trình biên dịch, ứng dụng sử dụng những cái thư viện ấy thì nó cũng sẽ include những cái file hay thư viện static libraries vào để link chúng với nhau để tạo ra những file excutable code để nạp vào chip để chip có thể chạy được luôn.

Static libraries chỉ đơn giản là những file tập hợp object code đã được biên dịch. Những phần linker đấy chỉ include vào link thôi.

Static libraries này có thể thư viện do người lập trình tạo ra được.

2.6. Macro

Macro là một mảnh của code, được cấp tên để định danh. Khi quá trình tiền xử lý diễn ra thì nó sẽ xử lý bất kỳ chỗ nào của tên macro đấy xuất hiện thì nó sẽ thay thế giá trị nội dung của macro đó ở file source code trong quá trình tiền xử lý.

Có 2 loại macro: 1 cái là object like macro (để định nghĩa một số dữ liệu hay giá trị) và function like macro (được dùng giống như những cái function con của function, có các tham số).

2.7. Biến con trỏ

Con trỏ là biến dùng để lưu trữ địa chỉ của biến khác hoặc hàm.

Ứng dụng thông dụng nhất của con trỏ hàm (function pointer) là trong việc xây dựng các call back function.

Việc gọi thông qua con trỏ hàm đảm bảo tính độc lập của thư viện với tầng ứng dụng và lập trình viên có thể xây dựng thư viện này mà không cần có code của tầng ứng dụng.

2.8. Cấu trúc dữ liệu.

2.8.a) Phân loại cấu trúc dữ liệu

Cấu trúc dữ liệu chia làm 2 loại, kiểu nguyên thuỷ và ko nguyên thuỷ.

Cấu trúc dữ liệu kiểu nguyên thuỷ bao gồm các kiểu số nguyên, thực, ký tự, các cấu trúc dữ liệu này có thể được thao tác trực tiếp bằng máy tính ví dụ như chúng ta có thẻ cộng trực tiếp 2 số nguyên vào dấu cộng.

Cấu trúc dữ liệu kiểu ko nguyên thuỷ chia làm 2 loại: tuyến tính và ko tuyến tính, cấu trúc dữ liệu được coi là tuyến tính khi các thành phần dữ liệu được xây dựng thành một chuỗi danh sách liên tiếp nhau như mảng (arrays), danh sách liên kết (linked list), ngăn xếp (stack), queues.

Cấu trúc dữ liệu được coi là ko tuyến tính khi các thành phần dữ liệu ko được tổ chức thành 1 chuỗi các thứ tự như dạng cây (tree), đồ hoạ (graph).

Những cấu trúc dữ liệu ko nguyên thuỷ này không thể được thao tác trực tiếp bằng các câu lệnh của máy tính, ví dụ như chúng ta ko thể cộng trực tiếp 2 mảng với nhau bằng dấu cộng.

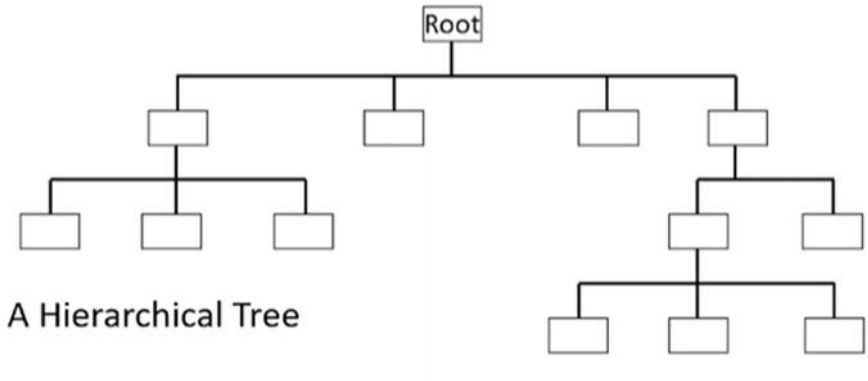
2.8.b) Các cấu trúc dữ liệu thường được sử dụng

- Mảng là 1 cấu trúc dữ liệu lưu trữ 1 tập hợp gồm các phần tử có cùng kiểu có độ dài cố định, có cùng kiểu dữ liệu.Từng phần tử được truy cập theo chỉ mục bằng cách sử dụng 1 loạt các số nguyên liên tiếp nhau gọi là index.

- Danh sách liên kết có cấu trúc là các node liên kết với nhau, mỗi phần tử là một node, mỗi node bao gồm hai phần phần: đối tượng dữ liệu (data item) là phần chứa dữ liệu của phần tử hoặc nó cũng có thể là một con trỏ để trỏ đến một đối tượng dữ liệu và một con trỏ next để trỏ đến node tiếp theo trong danh sách, có tác dụng để liên kết node khác trong danh sách.

Node cuối cùng trong danh sách gọi là tail (đuôi) và có một con trỏ trỏ đến list gọi là head và ban đầu cả hai đều được khởi tạo bằng NULL.

Cấu trúc dữ liệu không tuyến tính là tree (dạng cây), là một tập hợp hữu hạn các phần tử không rỗng, trong đó có một phần tử được gọi là ROOT và các phần tử còn lại được phân chia thành n tập con rời rạc, và mỗi tập con đó tự nó là 1 tree.

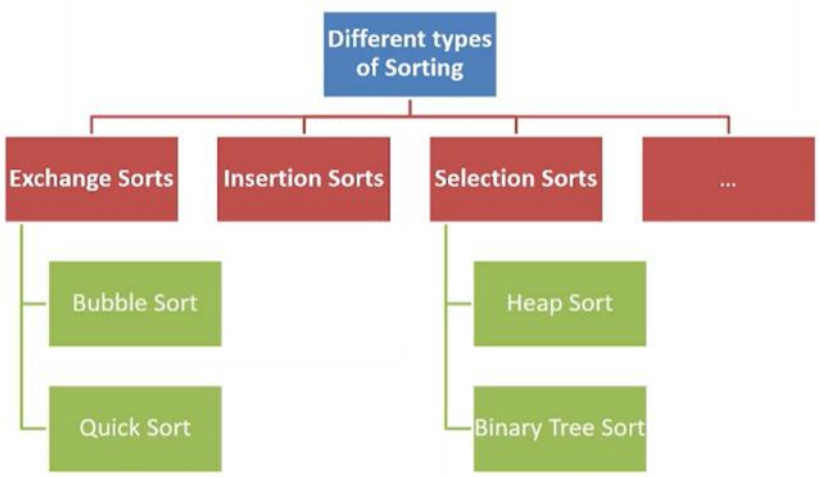


Hình 2.3. Sơ đồ của cấu trúc dữ liệu cây

Các nhánh nhỏ có thể được chia thành các nhánh nhỏ hơn, ta thấy hình dạng tương tự như một cái cây.

Kiểu cấu trúc dữ liệu này cũng được chia thành nhiều dạng nhỏ hơn như binary tree, n – array tree, red – black tree, …

2.9. Các thuật toán cơ bản sắp xếp trong lập trình:



Hình 2.4. Các thuật toán sắp xếp

Gồm có sắp xếp nổi bọt, sắp xếp nhanh, sắp xếp theo kiểu chèn, sắp xếp theo kiểu lựa chọn, sắp xếp bằng heap hoặc cây nhị phân.

2.10. C code Optimization

2.10.a) Giới thiệu

Hầu hết con chip trong dự án có kích thước bộ nhớ Flash và RAM hoặc vừa đủ hoặc lớn hơn so với kích thước chương trình, yêu cầu đáp ứng của mình hay không chặt chẽ về thời gian đáp ứng. Như vậy mình cần optimize code để đáp ứng được các yêu cầu của khách hàng, ví dụ như các yêu cầu về hàm ngắt để tránh trường hợp là mình bị missing ngắt ví dụ mình đang chạy ngắt này thì ngắt khác đến thì ko đáp ứng kịp. nếu thời gian chạy ngắt quá lâu thì sẽ bị bỏ qua một vài ngắt thì khi đó chương trình của mình sẽ bị sai, khi đó khách hàng đưa ra là hàm handle ngắt phải chạy trong bao nhiêu ns, ms, bao nhiêu tick, nếu mình vượt quá thời gian đó thì phải ingroup về speed sao cho cái handle ngắt đó phải nhanh lên, nó ko chạy chậm nữa.

Áp dụng các quy tắc, các thuật toán, phân tích các cái design khác để cho làm code có thể nhanh hơn, nhỏ hơn và hiệu quả hơn.

Khi phần cứng đã đạt đến giới hạn (code size quá lớn hoặc code ko đủ nhanh), để làm cho code chạy nhanh hơn, kích thước code nhỏ hơn và đỡ tốn điện hơn, thường không thể nào đạt được tất cả các mục tiêu cùng lúc mà phải hy sinh một trong các yếu tố đó.

2.10.b) Các phương pháp optimization

Optimize bằng tay (do lập trình viên xử lý), ví dụ như:

- Những biến được đặt trong thanh ghi thì sẽ truy xuất nhanh hơn các biến đặt trong RAM
- Những hàm có chữ inline đằng trước, code của hàm inline sẽ được copy vào nơi gọi hàm đó ra. Việc này sẽ được thực hiện bởi trình biên dịch và code size sẽ lại tăng lên.

Optimize bằng tool

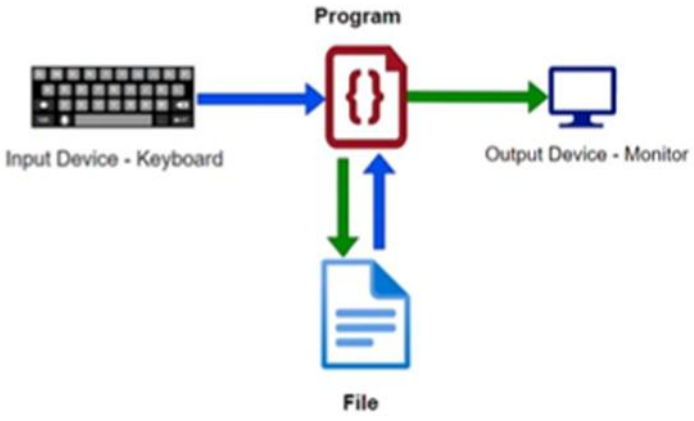
- Bật option về optimize trong trình biên dịch
- Dùng tool profiling để biết hàm nào chạy mất bao nhiêu thời gian, hàm nào được gọi nhiều lần.

Có 2 loại tool profiling là dựa vào xem code của nó có chèn vào code của chúng ta hay không

2.11. File handling

Hầu hết các chương trình đều yêu cầu đọc và ghi dữ liệu vào các hệ thống lưu trữ trên đĩa, các chương trình xử lý văn bản cần lưu các tập tin văn bản, các chương trình xử lý bảng tính cần lưu trữ nội dung vào các ô.

2.11.a) Các thuật ngữ và định nghĩa



Hình 2.5. Ví dụ về stream

Với ngôn ngữ C những người thiết kế muốn xây dựng một cách giao tiếp thống nhất các nguồn dữ liệu tuần tự khác nhau như là file, bàn phím, cổng USB, máy in.

Giao diện có thể áp dụng cho tất cả các chương trình sử dụng thuộc tính và trừu tượng được đặt với tên chung là stream.

Các hàm thư viện trong C hiện nay xử lý các đối tượng kiểu file nên cấu trúc dữ liệu C đại diện cho 1 stream được gọi là file chứ không phải là một stream.

2.11.b) Cấu trúc file

Tên và phần đuôi mở rộng được ngăn cách nhau bằng dấu '.', phần đuôi sau dấu chấm của file được sử dụng để định dạng cho tệp tin đó

2.11.c) Kiểu dữ liệu file và con trỏ file

Là kiểu dữ liệu đại diện cho đối tượng stream. Một đối tượng có kiểu dữ liệu file thì giữ toàn bộ thông tin trạng thái nội bộ về kết nối với tập tin được liên kết

Đối tượng có kiểu dữ liệu file được cấp phát và quản lý bởi chức năng của thư viện vào ra trong C.

Một con trỏ tập tin (file pointer) rất cần thiết cho việc đọc và ghi các tập tin. Nó là một con trỏ trỏ đến một cấu trúc chứa thông tin về tập tin bao gồm tên tập tin, vị trí hiện tại của tập tin. Tập tin đó thì đang được đọc hay ghi và có bất kỳ lỗi nào xuất hiện hay là đã đến cuối tập tin rồi. người dùng không cần thiết phải biết chi tiết.

2.11.d) Làm việc với stream:

- Để mở một stream thì ta mở 1 tệp với hàm fopen để tạo ra một stream mới và thiết lập kết nối giữa stream với tập tin.

- Khi một stream được đóng với hàm fclose, một kết nối giữa stream và file sẽ bị huỷ bỏ. sau khi đóng 1 stream thì ta ko thể thực hiện bất kỳ thêm 1 thao tác bổ sung nào trên stream này nữa

- Stream buffering: các ký tự được ghi vào một stream thường được tích luỹ và truyền không đồng bộ tới tệp trong 1 khối thay vì xuất hiện ngay khi chúng được xuất ra bởi chương trình ứng dụng thì được gọi là 1 vùng đệm. Hiện tại có 3 kiểu vùng đệm: unbuffered, line buffered, fully buffered

- Flusing Buffer: trong quá trình làm việc với buffer thì buffer sẽ liên tục tích luỹ các dữ liệu, việc đẩy vào buffer trên một stream có nghĩa là chúng ta sẽ truyền những dữ liệu data sẽ được tích luỹ dần thành ký tự và được chuyển dần đến file.

Việc chuyển đổi tích luỹ này sẽ được tự động đẩy lên khi mà đầu ra của buffer là đầy hoặc là khi chúng ta đóng stream hoặc là chúng ta sẽ gọi kết thúc stream của chúng ta bằng cách gọi hàm exit, …

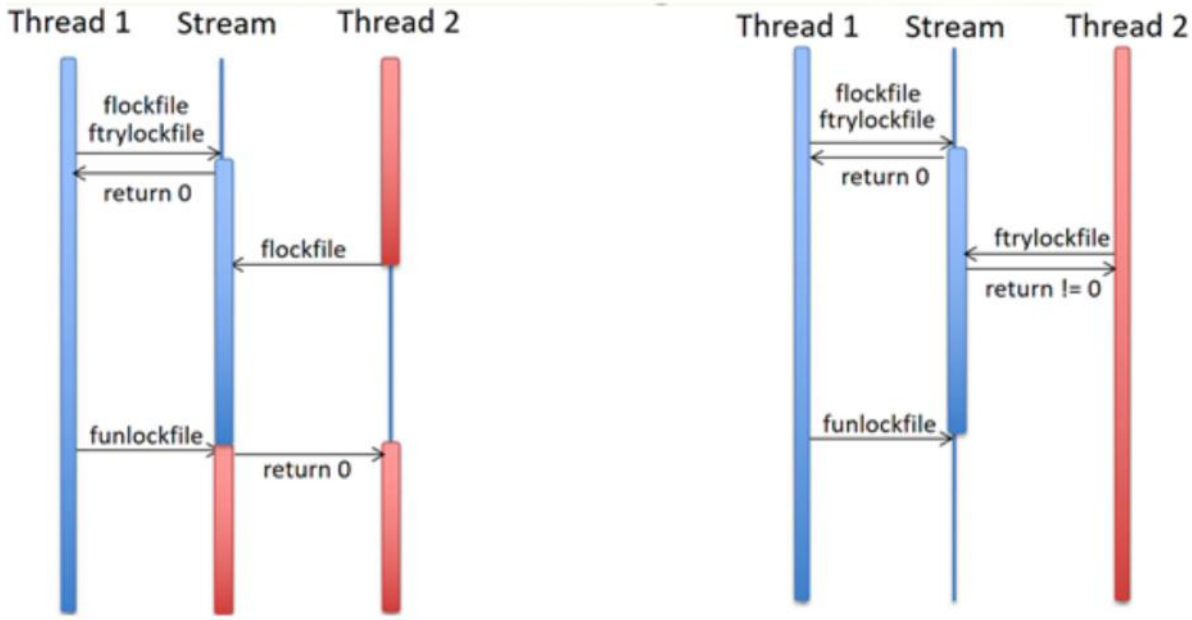
2.11.e) Formatted Input/Output

Đây là chuẩn rất hay làm việc với stream với các hàm fprint và fscanf đều là những hàm dễ sử dụng nhất với input và output trong quá trình chúng ta làm việc với stream.

Một số dữ liệu thì sẽ chuyển đổi thành mã ASCII thay vì formatted binary.

Vấn đề về kích thước file cũng như là tốc độ mà nó xử lý cần được quan tâm thì chúng ta sẽ có các hàm fread fwrite là những lựa chọn tốt hơn trong quá trình input output.

2.11.f) Xung đột khi làm việc với stream.



Hình 2.6. Cách các stream hoạt động

Trong quá trình làm việc của một stream trong các hệ điều hành khác nhau như Linux thì sẽ xảy ra một số vấn đề: khi nhiều chương trình hoặc là nhiều ứng dụng cùng sử dụng stream đó cho việc đọc viết có thể gây ra xung đột vì lúc đó có nhiều luồng cùng vào sửa thay đổi thông tin và dẫn đến làm sai lệch dữ liệu của stream đó.

Để tránh việc đó xảy ra thì ta cần đảm bào khi nhiều luồng ứng dụng cùng làm việc với stream thì sẽ ko gây ra các xung đột trong quá trình làm việc.

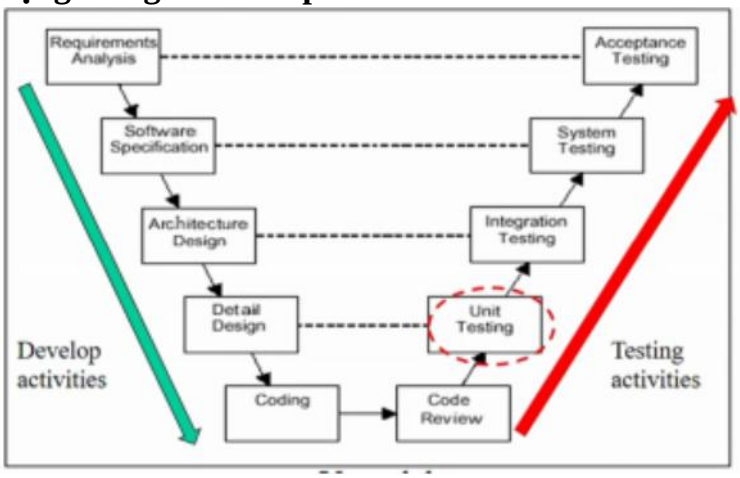
2.12. Unit test

2.12.a) Giới thiệu

Năm 2017, hãng Toyota thu hồi 50 nghìn xe do lỗi điều khiển trợ lực lái cho vô lăng. Cụ thể là trong việc sản xuất sắp xếp công đoạn không đúng dẫn đến tụ bị nứt làm thay đổi giá trị điện áp trong việc điều khiển trợ lực đánh lái dẫn đến gây ra lỗi.

Để tránh các lỗi như vậy xảy ra cần các testing software code (hay còn gọi là unit test) thực hiện kiểm tra các đoạn code cho các đơn vị nhỏ nhất của một phần mềm.

2.12.b) Các hoạt động trong sản xuất phần mềm



Hình 2.7. V model trong sản xuất phần mềm

Bên trái là các hoạt động của dev và bên phải là các hoạt động testing. Với mỗi hoạt động của dev thì tương ứng có một hoạt động của test.

Việc kiểm thử yêu cầu phân tích lập trình viên thu thập và nghiên cứu và phân tích các yêu cầu về kỹ thuật.

Architecure design: với các công cụ cần có, cần có mối quan hệ và sự phụ thuộc giữa chúng là như thế nào. Mối quan hệ và truyền tải link dữ liệu giữa chúng là như thế nào. Với mỗi module cần có detail design các functional feature chính. Với mỗi functional coding như thế nào trong giai đoạn coding.

Sau việc code thì cần phải có code review về coding convention, giải thuật có đúng hay không, các vấn đề về complain có đúng ko, các vấn đề khác thì theo tài liệu cung cấp.

Đầu tiên phải tự review sau đó reivew với các senior.

Unit testing là hoạt động của 1 dev là chủ yếu thực hiện các kỹ thuật testing

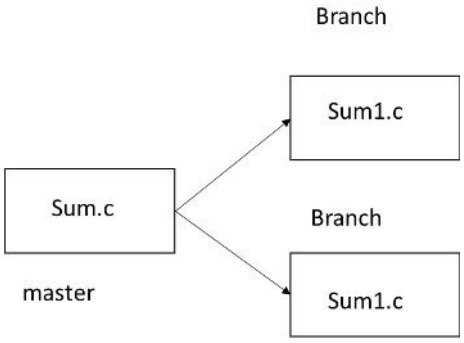
2.13. Sử dụng Git trong dự án

2.13.a) Các khái niệm trong Git

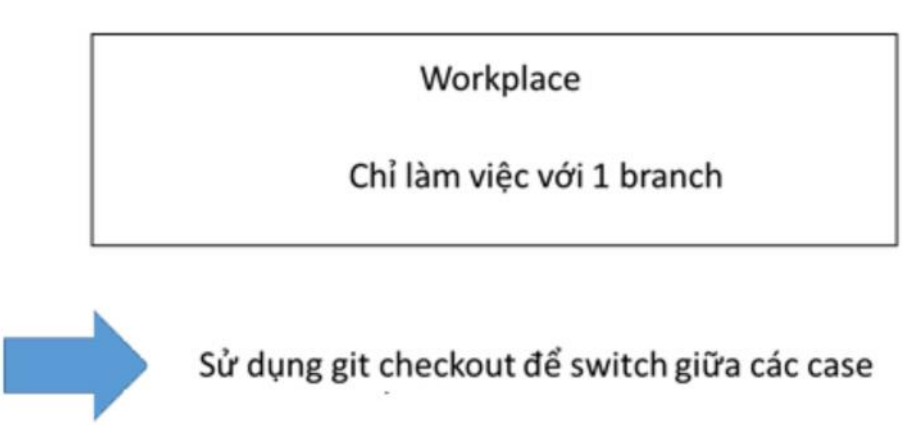
Git hub là 1 server sử dụng giao diện Git, giống như SVN (ít được dùng đến) hay bootlean, …

VCS là trình quản lý phiên bản source code, mỗi một version tương ứng với một commit. Mỗi lần cần trở lại các version thì dùng lệnh git check out.

Snapshot (lưu ảnh của file bị biến đổi) cho tất cả commit.

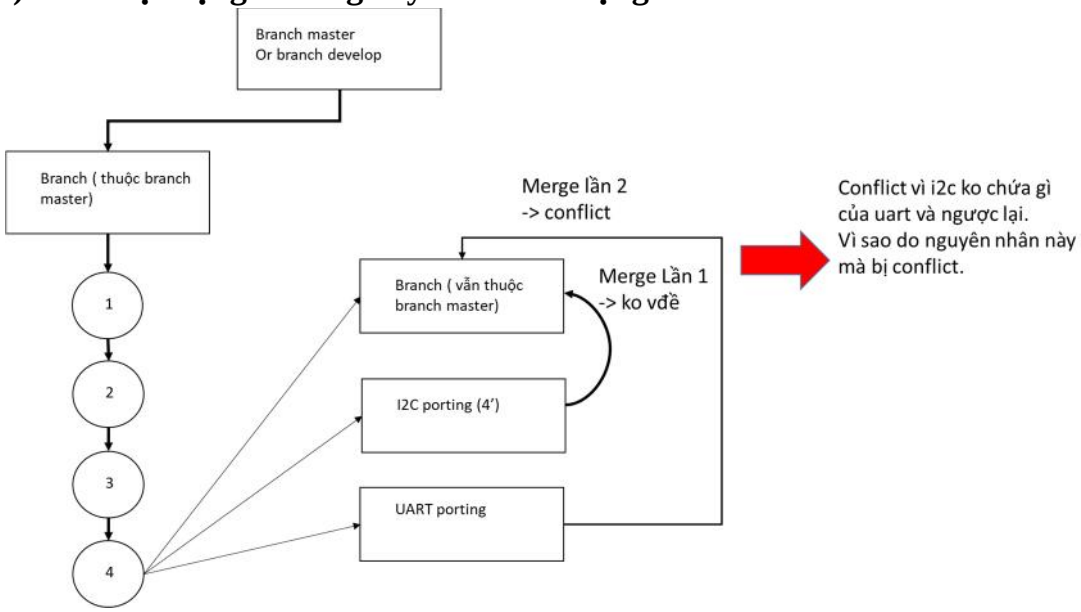


Một master được coi là một root, trong đó các brach là phần tử con của root.

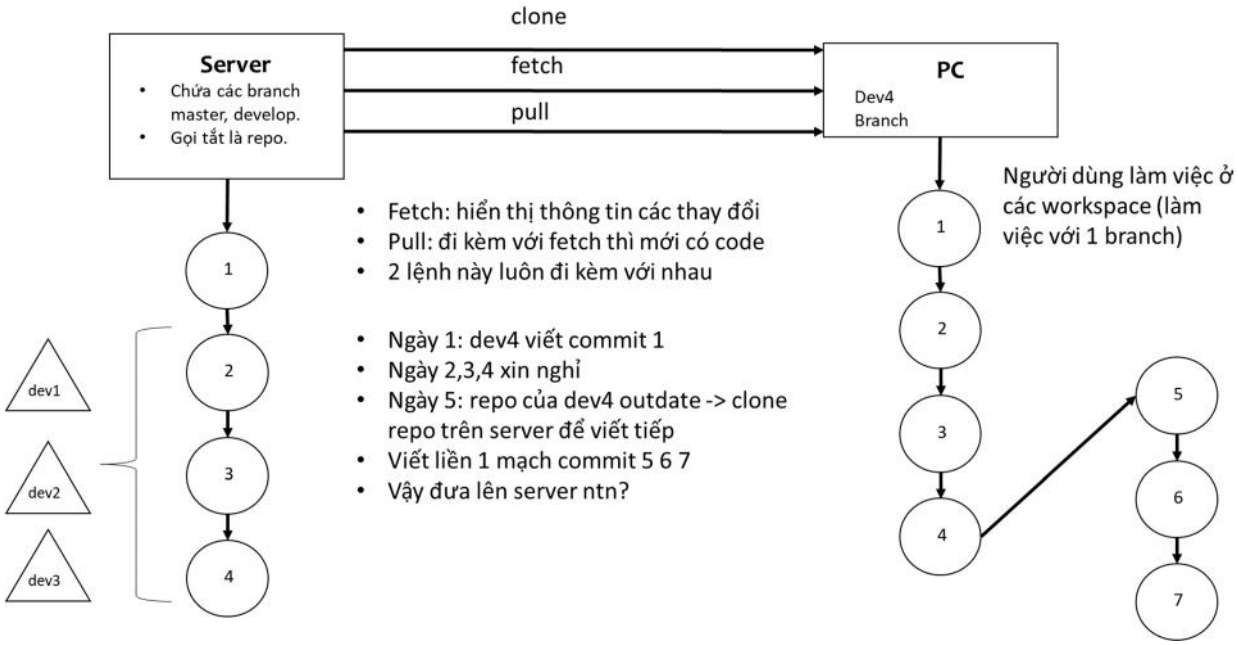


Do một workplace (PC) chỉ làm việc được với một branh, cho nên sử dụng câu lệnh git checkout để chuyển qua các branh khác (giống như bật một tab khác).

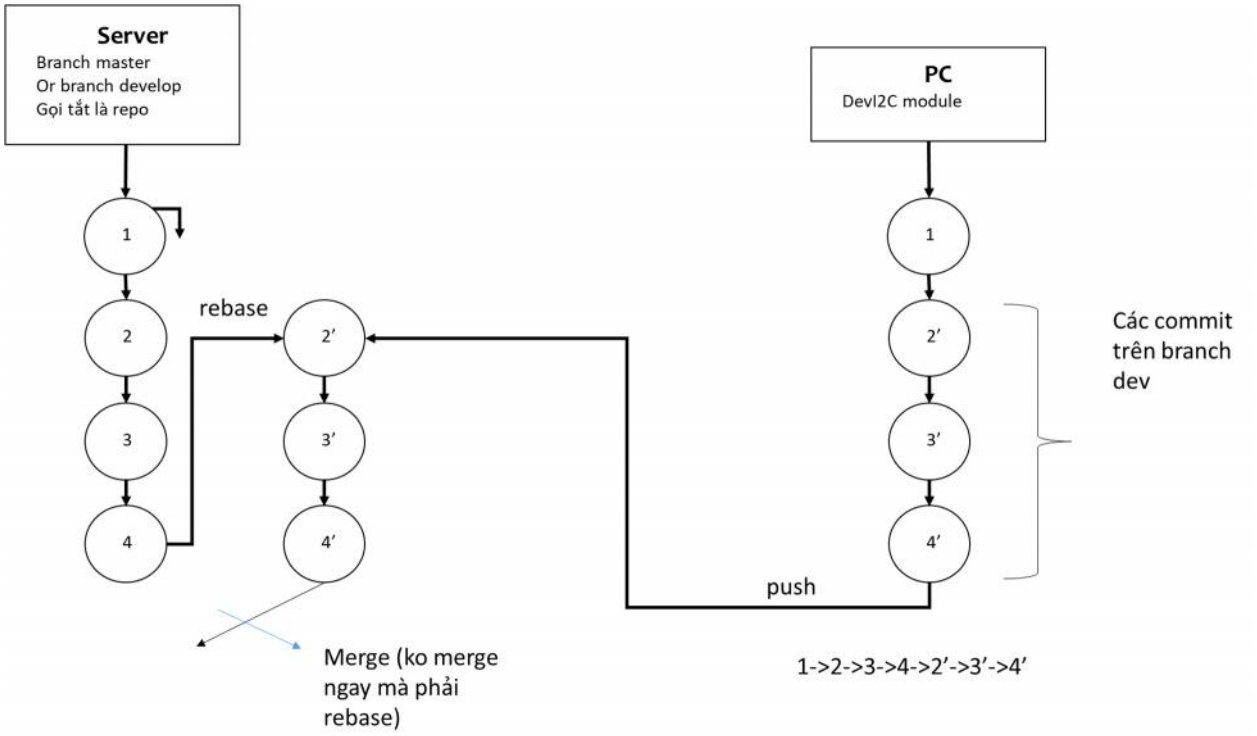
2.13.b) Các hoạt động thường xảy ra khi sử dụng Git



Hình 2.8. Xung đột khi merge các branch



Hình 2.9. Các hoạt động sử dụng của một dev trong dự án



Hình 2.10. Một commit của một dev được merge lên hệ thống

2.2. Kết luận chương

Trong chương này đã hệ thống lại một số khái niệm cơ bản về ngôn ngữ lập trình C, giúp cho sinh viên nắm bắt được các kiến thức trước khi bắt đầu làm quen với các dự án hệ thống nhúng sau này.

CHƯƠNG 3: DỰ ÁN MOCK

3.1. Tổng quan về dự án MOCK

Để sinh viên thực tập nắm rõ các kiến thức về ngôn ngữ C cơ bản, dự án MOCK là một bài kiểm tra tổng hợp để đánh giá khả năng lập trình ngôn ngữ C của sinh viên.

Sinh viên được cung cấp một file image của bảng FAT (FAT12). Sinh viên sẽ thao tác với file này thay vì thao tác với ổ đĩa thực.

3.2. Nội dung dự án

3.2.a) Phần basic

- Viết các hàm để thao tác đến lớp vật lý (tạo file HAL.h và HAL.c).

- Bình thường, người ta sẽ chia file system thanh các layer. Layer giao tiếp với lớp vật lý gọi là Hardware abtraction layer (HAL).
- Lớp này sẽ được thay đổi tùy theo cấu trúc vật lý của hệ thống.
- Các lớp trên sẽ chỉ gọi đến các hàm của lớp này mà không truy xuất trực tiếp đến phần cứng của hệ thống.
- Đối với bài toán trên, HAL sẽ thao tác đến file image thay vì ổ đĩa thực. Tất cả các chương trình khác sẽ gọi các hàm ở lớp này.
- Sau này, khi muốn đọc từ ổ đĩa thực, lập trình viên chỉ việc thay đổi file này mà không cần đả động gì đến các file chương trình khác.
- HAL sẽ cần ít nhất là các hàm sau:

  int32_t kmc_read_sector(uint32_t index, uint8_t *buff);

  Hàm này read sector thứ index vào buff, trả về số byte được đọc.

- int32_t kmc_read_multi_sector(uint32_t index, uint32_t num, uint8_t *buff);

  Đọc num sector liên tiếp, bắt đầu từ sector thứ index, vào mảng trỏ bởi buff.

  Hàm này trả về số byte được đọc

- Ngoài các hàm được cho có thể thêm hàm nào khác để code thuận tiện hơn thì tự thêm vào.

- Viết chương trình để tạo file ví dụ: read_file.h và read_file.c

- Hiển thị tất cả các folder và file trong thư mục gốc.
- Cho phép hiển thị các thư mục con và file ở trong 1 thư mục
- Hiển thị nội dung của 1 file được lựa chọn lên màn hình.

- Chú ý: các yêu cầu trên chỉ cần thao tác với tên file ngắn mà không cần chú ý đến Long File Name

3.2.b) Phần Advance

- Xử lý cả 3 loại FAT12/FAT16/FAT32 và Long file name

- Sau khi đã làm được phần basic ở trên, hãy viết FAT file system library. Library này cho phép mở file, đọc file, ghi file... Giống như các hàm fopen, fread, fwrite...

- Để tạo file image của ổ đĩa:

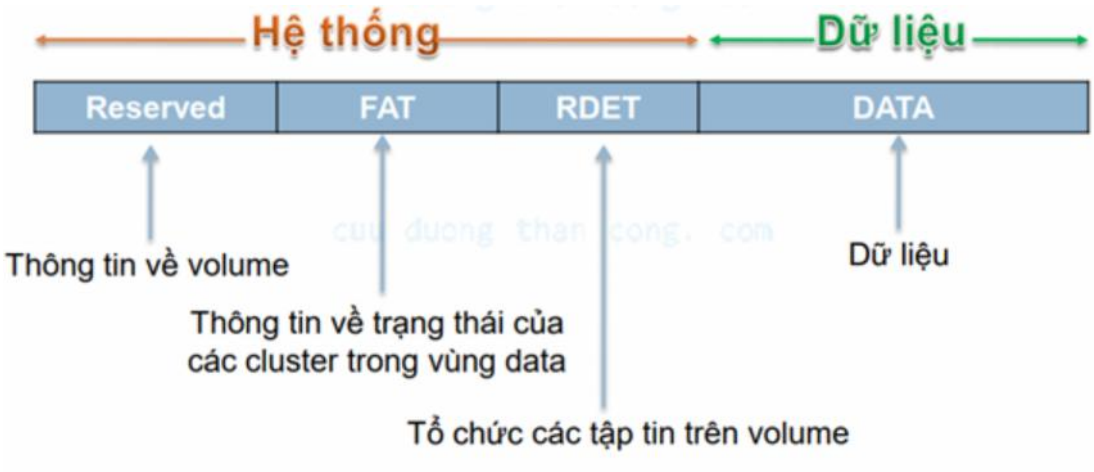
- ❖ Tạo 1 partition bé nhất có thể trên ổ USB hoặc harddisk... Bằng partion magic. Nếu có ổ đĩa mềm thì dùng luôn.
- ❖ Format ổ đĩa đó.
- ❖ Dùng chương trình HxDen để tạo 1 image của ổ đĩa đó.
- ❖ Download HxDen ở http://www.mh-nexus.de
- ❖ Vào menu Extras/Open disk, chọn ổ đĩa cần tạo image. Sau đó vào File/Save As.

3.3. Nội dung tìm hiểu

3.3.a) Tổng quan về hệ thống FAT

Bảng phân bổ tệp (File Allocation Table) được lưu trữ trên đĩa mềm (floppy disk) hoặc đĩa cứng (hard disk) và mô tả cluster nào thuộc về file nào trên đĩa đó.
Xuất hiện vào cuối những năm 1970 và đầu những năm 1980. Các phiên bản của FAT là FAT12, FAT16, FAT32 tương ứng với 12,16, 32 là số bit dùng để đánh số thứ tự các khối dữ liệu (cluster).

Cấu tạo của hệ thống Fat gồm hai vùng: vùng hệ thống, vùng dữ liệu



Hình 3.1. Phân vùng hệ thống FAT

3.3.b) Tóm tắt về sector, cluster

Sector là đơn vị chứa dữ liệu nhỏ nhất. Theo các chuẩn thông thường thì một sector chứa 512 byte.

Cluster là một đơn vị lưu trữ gồm một hoặc nhiều sector. Khi hệ điều hành lưu trữ một file vào disk, nó ghi tập tin đó vào các cluster liền nhau. Nếu không sẵn cluster liền nhau, hệ điều hành sẽ tìm kiếm cluster còn trống ở kế đó và ghi tiếp tập tin lên đĩa. Quá trình cứ thế tiếp tục cho đến khi toàn bộ dữ liệu được cất giữ hết. Nếu quá trình trên diễn ra trong thời gian dài thì sự phân mảnh sẽ xảy ra.

3.3.c) Boot Sector và BPB

Trong FAT, cấu trúc dữ liệu quan trọng nhất đó là BPB (BIOS Parameter Block) vì tất cả các thông tin liên quan đến FAT đều nằm ở đây. BPB nằm bên trong Boot Sector, là Sector đầu tiên trong disk.

Các vùng bôi đậm màu xanh là các vùng cần lưu ý khi làm việc với FAT.

Name	Offset Hex	Size (bytes)	Description	Ký hiệu
BS_jmpBoot	0	3	Lệnh nhảy đến đoạn boot code.	
BS_OEMName	3	8	Version/tên HĐH	
BPB_BytsPerSec	**B**	**2**	**Số bytes/sector** **Ví dụ: 512, 1024, 2048 hoặc 4096**	
BPB_SecPerClus	**D**	**1**	**Số sectors/cluster**	S_C
BPB_RsvdSecCnt	**E**	**2**	**Số sector để dành (khác 0) (Số sector trước bảng FAT)**	S_B
BPB_NumFATs	**10**	**1**	**Số bảng FAT**	N_F
BPB_RootEntCnt	**11**	**2**	**FAT12, FAT16: số entry trong bảng RDET** **FAT32: có giá trị là 0**	N_{RDET}
BPB_TotSec16	**13**	**2**	**FAT12, FAT16: tổng số sector của Volume** **FAT32: có giá trị là 0**	S_V
BPB_Media	15	1	Loại Voulme	
BPB_FATSz16	**16**	**2**	**FAT12, FAT16: số sector trong 1 bảng FAT** **FAT32: có giá trị là 0 (BPB_FATSz32)**	S_F
BPB_SecPerTrk	18	2	Số sectors/track	
BPB_NumHeads	1A	2	Số heads	
BPB_HiddSec	1C	4	Số sector ẩn trước Volume	
BPB_TotSec32	**20**	**4**	**Số sector trong Volume.** **Nếu bằng 0, BPB_TotSec16 phải khác 0**	N_V

Hình 3.2. Các thông tin quan trọng khi đọc bảng FAT

3.3.d) Công thức tính kích thước các vùng.

Từ các thông số của BPB thì ta có thể tính chính xác được vị trí của các vùng và kích thước mỗi vùng trên ổ đĩa như sau:

Vùng FAT là vùng ngay sau vùng dự trữ, vì thế kích thước và vị trị của vùng FAT sẽ được tính như công thức dưới đây:

FatStartSector = BPB_ResvdSecCnt;

FatSectors = BPB_FATSz * BPB_NumFATs;

Tiếp theo, kích thước và vị trí của vùng Root Directory:

RootDirStartSector = FatStartSector + FatSectors;

RootDirSectors = (32 * BPB_RootEntCnt + BPB_BytsPerSec – 1) / BPB_BytsPerSec;

32 ở đây là kích thước một Directory Entry (đơn vị Byte).

Ở công thức trên, kết quả đã đã bao gồm đủ Sector để chứa toàn bộ Root Directory rồi.

Ở FAT32, thì vì trường BPB_RootEntCnt luôn là 0 nên kích thước của vùng Root Directory cũng luôn là 0.

Cứ thế, vùng dữ liệu được tính theo công thức dưới đây;

DataStartSector = RootDirStartSector + RootDirSectors;

DataSectors = BPB_TotSec – DataStartSector;

3.3.e) FAT và Cluster

FAT là một vùng rất quan trọng, chức năng của vùng FAT là ghi lại, quản lý vị trí (một dãy các cluster) trên vùng Data của các File trong hệ thống.

Vùng Data được chia ra đều ra thành các đơn vị lưu trữ gọi là cluster. Cluster không phải là có kích thước tùy ý mà phải bằng một số lần cố định (BPB_SecPerClus) Sector.

Vùng FAT sẽ quản lý một danh sách các số và ánh xạ 1:1 đến các cluster này. Mỗi giá trị được quản lý trong vùng FAT tương ứng với một Cluster trong vùng Data.

Lưu ý 2 Entry đầu trong vùng FAT, không sử dụng hay nói cách khác là dự trữ. Nó không ánh xạ đến một cluster nào hết, vậy nên chỉ số Cluster được tính từ 2.

Thông thường khi có thao tác chỉnh sửa, hệ thống FAT sẽ copy và cập nhật ở cả 2 FAT.

3.3.f) Thông số quyết định loại FAT

Được quyết định dựa trên số cluster của ổ đĩa.

3.3.g) Truy cập FAT Entry

Mỗi Cluster thứ N thì sẽ có 1 entry tương ứng (1-1) ở đâu đó trong vùng FAT.

3.3.h). Mối liên hệ giữa File và các cluster

Các File nằm trễn ổ đĩa được quản lý theo đơn vị là Directory. Trong Directory chứa tên file, cluster đầu tiên trong dãy cluster của File, những thông tin này là cửa vào để truy cập FAT. Dữ liệu mỗi File là một dãy các cluster nối nhau (linked list).

Cluster 0, 1 chỉ dùng cho dự trữ, vì thế Cluster sẽ được bắt đầu từ 2. Một ổ đĩa có N Cluster có nghĩa là các Cluster đó có chỉ số từ 2 đến N+1 và số FAT Entry là N+1. Những File có kích thước là 0 thì sẽ không được cấp phát Cluster nào hết. Từ chỉ số hợp lệ của một Cluster, ta có thể tính được vị trí của Sector đầu tiên trong Cluster đó.

FirstSectorofCluster = DataStartSector + (N - 2) * BPB_SecPerClus;

Hay nói cách khác, từ dữ liệu được lưu trữ từ Cluster số 2 trở đi. Khi kích thước File lớn, nó có thể được lưu trữ trên nhiều Cluster. FAT Entry tương ứng cho mỗi Cluster đó sẽ chưa chỉ số của Cluster tiếp theo. Vì vậy, cứ đi theo thứ tự đó sẽ truy cập đến bất cứ Sector nào của File và không thể theo thứ tự ngược lại được.

FAT Entry của Cluster cuối cùng trong dãy Cluster mô tả dữ liệu một File sẽ có dấu hiệu đặc biệt để nhận biết (EOC). Giá trị này sẽ không trùng với bất cứ Cluster nào đang có để đảm bảo tính đúng đắn. Tùy theo mỗi loại FAT sẽ sẽ sử dụng các giá trị khác nhau:

FAT12: 0xFF8～0xFFF (thường đại diện là 0xFFF)

FAT16: 0xFFF8～0xFFFF (thường đại diện là 0xFFFF)

FAT32: 0x0FFFFFF8～0x0FFFFFFF (thường đại diện là 0x0FFFFFFF)

Thêm một giá trị đặc biệt nữa, đó là dấu hiệu dành cho Cluster lỗi (hay ta vẫn gọi là Bad Cluster). Cluster lỗi nghĩa nói đến Cluster chưa Sector không sử dụng được nữa do hỏng hóc của thiết bị nhớ. Trong quá trình Format, phục hồi, hay dọn dẹp, khi một Cluster được phát hiện là hỏng, thì ta phải dấu lại để sau không đưa dữ liệu lên đó nữa. Các giá trị dùng để đánh dấu Cluster hỏng lần lượt như sau, FAT12 là 0xFF7, FAT16 là 0xFFF7, FAT32 là 0x0FFFFFF7.

3.3.i) Cấu trúc Directory

Giả định File đều là Short File Name. Trong ổ đĩa FA, Folder là một loại File đặc biệt, người ta dựa vào một thuộc tính đặc biệt của File để xác định nó là Folder hay không. Thuộc tính này được lưu trữ bằng cấu trúc trong bảng Directory Table, có kích thước 2 byte. Vì thế, sẽ lưu trữ tối đa 65536 Entry.

Folder luôn tồn tại trong ổ đĩa là Root Directory, là Folder ở tầng cao nhất. Trong FAT12/FAT16, nó nằm ngay sau vùng FAT, và có kích thước không thay dổi. Độ dài của bảng Directory chính là giá trị của trường BPB_RootEntCnt. Sector bắt đầu của Root Directory được xác định bởi giá trị BPB_RootClus.

Vùng này được chia thành nhiều entry, mỗi entry chiếm 32 byte. Byte đầu tiên chứa thông tin trạng thái của entry về 1 tập tin/thư mục. Các entry chứa thông tin của file/folder như tên, thuộc tính

Sub Directory chứa thông tin các file/folder trong một folder nằm trên vùng data, tổ chức thành các entry như RDET và luôn luôn có 2 entry "." là thông tin folder của chính nó và ".." là thông tin folder cha.

3.3.k) Phân loại Entry

Gồm hai loại là Entry chính: chứa thông tin của tập tin và Entry phụ: chứa tên tập tin (dành cho Long File Name).

Giá trị	Ý nghĩa
0x00	Entry trống
0x05	Initial character is actually 0xE5
0x2E	'Dot' entry; hoặc '.' or '..'
0xE5	Entry đã bị xoá.

Hình 3.3. Các trạng thái của Entry.

OFFSET	ĐỘ DÀI (byte)	NỘI DUNG
0h (0)	8	Tên chính của tập tin
8h (8)	3	Tên mở rộng
Bh (11)	1	Thuộc tính (0-0-A-D-V-S-H-R) Nếu có giá trị là 0x0F thì entry này sủ dụng cho LFNs
Ch (12)	10	Không dùng
16h (22)	2	Giờ cập nhật tập tin
18h (24)	2	Ngày cập nhật tập tin
1Ah (26)	2	Cluster bắt đầu
1Ch (28)	4	Kích thước tập tin

Hình 3.4. Cấu trúc entry chính.

8n	Mask	Mô tả
0	0x01	Read Only
1	0x02	Hidden
2	0x04	System
3	0x08	Volume Label
4	0x10	Subdirectory
5	0x20	Archive
6	0x40	Device (internal use only, never found on disk)
7	0x80	Unused

Hình 3.5. Các thuộc tính của tập tin

Byte Offset	Length	Description
0	1	Số thứ tự của entry
1	10	Các ký tự của tên file (5 ký tự UTF-16)
B	1	Attributes (luôn luôn có giá trị là 0x0F)
C	1	Reserved (luôn luôn có giá trị là 0x00)
D	1	Checksum của tên file MS-DOS
E	12	Các ký tự của tên file (6 ký tự UTF-16)
1A	2	Cluster đầu tiên (luôn luôn có giá trị là 0x0000)
1C	4	Các ký tự của tên file (2 ký tự UTF-16)

Hình 3.6. Cấu trúc Entry phụ

3.3.l) Định dạng nhãn thời gian

Trong Directory Entry, có các trường liên quan đến thời gian: DIR_WrtTime, DIR_WrtDate. Các trường khác không được hỗ trợ. Các trường không được hỗ trợ sẽ được giữ giá trị 0.

3.4) Giải quyết nội dung

Chia code thành 3 tầng:

Tầng HAL dùng để mở file, đóng file floppy.img và đọc sector.

Tầng FATFS gọi các hàm ở tầng HAL để đọc các entry bằng danh sách liên kết.

Tầng APP gọi các hàm ở tầng FATFS và hiển thị menu: hiển thị bảng thông số Boot Sector, hiển thị Root Directory, Sub Directory và các option để người dùng chọn.

Hình 3.7. Đọc từ ổ đĩa ra những ký tự mà người dùng đọc được.

Để đọc được nội dung trong ổ đĩa, ta cần giải quyết các vấn đề sau:

Tìm kiếm tập tin

- Xuất phát từ RDET.
- Duyệt qua từng entry trong RDET.
- Duyệt qua các SDET (nếu có).

Đọc nội dung tập tin

- Dựa trên RDET/SDET ➔ tìm entry chính tập tin
- Dựa trên entry chính:
  - Sector bắt đầu + Kích thước.
  - FAT: tìm danh sách các cluster chứa nội dung tập tin
  - DATA: đọc nội dung tập tin tương ứng từng cluster.
- Lưu ý: cluster cuối cùng chỉ đọc đúng số byte còn lại (vì có thể nó không chiếm trọn toàn bộ nội dung cluster cuối cùng)

3.5. Kết luận chương

Trong chương này đã giới thiệu về hệ thống phân bổ tập tin FAT và phương pháp giúp người dùng có thể đọc được một ổ đĩa mềm.

CHƯƠNG 4: TÀI LIỆU THAM KHẢO

1. http://en.wikipedia.org/wiki/File_Allocation_Table

2. http://www.pjrc.com/tech/8051/ide/fat32.html

3. http://www.win.tue.nl/~aeb/linux/fs/fat/fat.html#toc2

4. http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc

5. Một số tài liệu mật công ty FPT Software đưa để tham khảo.