1. Khái niệm phần mềm nguồn mở là gì?

-Phần mềm nguồn mở (Open Source Software) là một loại phần mềm mà mã nguồn của nó được công khai, cho phép người dùng xem, sửa đổi và phân phối lại mã nguồn đó. Khái niệm này được xây dựng dựa trên các nguyên tắc của "Giấy phép công cộng GNU" (GNU General Public License) của Tổ chức Free Software Foundation (FSF).

1. Freeware: Phần mềm miễn phí, nhưng không có mã nguồn công khai. Người dùng có thể sử dụng và phân phối miễn phí, nhưng không được phép sửa đổi mã nguồn.
2. Shareware: Phần mềm có thể được tải về và sử dụng miễn phí trong một thời gian giới hạn. Sau đó, người dùng được yêu cầu mua bản quyền để tiếp tục sử dụng.
3. Charityware/Careware: Phần mềm được cung cấp miễn phí, nhưng người dùng được khuyến khích đóng góp tiền cho một tổ chức từ thiện hoặc một mục đích nhân đạo.
4. Public domain: Phần mềm không thuộc bất kỳ quyền sở hữu trí tuệ nào và có thể được sử dụng, sửa đổi và phân phối tự do mà không cần yêu cầu bất kỳ giấy phép nào.
5. Copyleft: Một phần mềm nguồn mở có điều khoản giấy phép đặc biệt, yêu cầu bất kỳ phiên bản mới nào hoặc các tác phẩm phái sinh cũng phải tuân thủ cùng một giấy phép nguồn mở.
6. Proprietary/Locked-in/Non-free: Phần mềm có mã nguồn đóng và được kiểm soát bởi một nhà phát triển hoặc một tổ chức. Người dùng không được phép xem, sửa đổi hoặc phân phối lại mã nguồn của nó mà không có sự cho phép từ chủ sở hữu.
7. Hệ quả của sự tự do phần mềm

Sự tự do phần mềm mang lại nhiều hệ quả quan trọng và có ảnh hưởng sâu rộng. Dưới đây là một số hệ quả chính của sự tự do phần mềm:

1. Tính linh hoạt và tùy biến: Sự tự do phần mềm cho phép người dùng tùy chỉnh và thay đổi phần mềm theo nhu cầu của họ. Người dùng có quyền tiếp cận mã nguồn và sửa đổi nó để phù hợp với mục đích sử dụng cụ thể.
2. Khả năng học hỏi và phát triển: Sự tự do phần mềm khuyến khích việc học hỏi và phát triển vì người dùng có thể xem và tìm hiểu cách phần mềm hoạt động. Điều này thúc đẩy sự truyền đạt kiến thức và khả năng cải thiện phần mềm.
3. Sự cộng tác và chia sẻ: Tự do phần mềm tạo ra một cộng đồng người dùng và nhà phát triển phần mềm tương tác và chia sẻ kiến thức, ý tưởng và tài nguyên. Điều này thúc đẩy sự cộng tác và tạo ra sự tiến bộ chung.
4. Khả năng đảm bảo an ninh và sửa lỗi: Với sự tự do phần mềm, người dùng có thể phát hiện và sửa lỗi bảo mật và lỗi chức năng trong phần mềm. Việc có nhiều người kiểm tra và đóng góp vào mã nguồn mở tăng cường khả năng phát hiện và sửa lỗi, đảm bảo an ninh và độ tin cậy của phần mềm.
5. Tiết kiệm chi phí: Sự tự do phần mềm giúp tránh các chi phí phụ thuộc vào bản quyền phần mềm và giấy phép sử dụng. Người dùng không phải trả phí sử dụng phần mềm và có thể sao chép, phân phối và sửa đổi mà không gặp rào cản pháp lý.
6. Sự công bằng và đạo đức: Tự do phần mềm đặt trọng tâm vào sự công bằng, đạo đức và sự tôn trọng quyền riêng tư. Nó khuyến khích việc chia sẻ, tái sử dụng và phát triển công khai, mang lại lợi ích cho cộng đồng và
7. Ai tham gia phát triển các dự án PMNM, động lực khi họ tham gia là gì?

Các dự án Phần mềm nguồn mở (PMNM) thu hút sự tham gia từ một loạt các cá nhân và tổ chức. Dưới đây là một số người tham gia phổ biến và động lực khi họ tham gia vào các dự án PMNM:

1. Nhà phát triển (Developers): Nhà phát triển là những người chính tham gia vào việc viết mã nguồn và phát triển các tính năng cho dự án PMNM. Động lực của họ có thể là đam mê với lĩnh vực công nghệ thông tin, muốn chia sẻ kiến thức và kỹ năng của mình, hoặc muốn đóng góp vào cộng đồng.
2. Cộng đồng người dùng (User community): Cộng đồng người dùng là những người sử dụng phần mềm nguồn mở và cung cấp phản hồi, ý kiến và báo cáo lỗi. Động lực của họ là muốn sử dụng phần mềm chất lượng và tham gia vào việc phát triển và cải tiến sản phẩm.
3. Các tổ chức phi lợi nhuận: Một số tổ chức phi lợi nhuận như tổ chức từ thiện, tổ chức giáo dục hoặc tổ chức chính phủ tham gia vào các dự án PMNM để đáp ứng mục tiêu và nhiệm vụ của họ. Động lực của họ có thể là cung cấp phần mềm miễn phí cho cộng đồng, đóng góp vào nền kinh tế chia sẻ hoặc thúc đẩy sự phát triển và tiến bộ công nghệ.
4. Doanh nghiệp và tổ chức thương mại: Một số doanh nghiệp và tổ chức thương mại cũng tham gia vào các dự án PMNM. Động lực của họ có thể là muốn sử dụng phần mềm nguồn mở để tiết kiệm chi phí, tận dụng tính linh hoạt và tùy chỉnh của phần mềm, hoặc tạo và phân phối các sản phẩm và dịch vụ dựa trên phần mềm nguồn mở.

Động lực chung của những người tham gia là mong muốn tham gia vào cộng đồng, chia sẻ kiến thức và kỹ năng của mình, đóng góp vào sự phát triển và cải thiện phần mềm, hoặc tận dụng ưu điểm của phần mềm nguồn mở để đáp

1. So sánh PMNM và PMNĐ, ưu/nhược điểm

Phần mềm nguồn mở (PMNM): Ưu điểm:

1. Mã nguồn công khai: Mã nguồn của PMNM được công khai, cho phép người dùng xem, sửa đổi và phân phối lại. Điều này tạo điều kiện cho sự linh hoạt, tùy chỉnh và đóng góp từ cộng đồng.
2. Cộng đồng phát triển: PMNM thu hút sự tham gia của cộng đồng người dùng và nhà phát triển. Sự cộng tác và chia sẻ thông qua việc kiểm tra mã nguồn, báo cáo lỗi và đóng góp tính năng giúp cải thiện phần mềm.
3. Tiết kiệm chi phí: PMNM thường miễn phí hoặc có chi phí thấp hơn so với phần mềm đóng. Người dùng không phải trả phí bản quyền và có thể sử dụng, sao chép và phân phối miễn phí.
4. Độ tin cậy và an ninh: Với sự tham gia của cộng đồng rộng lớn, lỗi bảo mật và lỗi chức năng có thể được phát hiện và sửa lỗi nhanh chóng, đảm bảo tính tin cậy và an ninh của phần mềm.

Nhược điểm:

1. Hỗ trợ kém: Vì phần mềm nguồn mở thường không đi kèm với sự hỗ trợ chuyên nghiệp, người dùng có thể gặp khó khăn trong việc tìm kiếm giải pháp và hỗ trợ khi gặp vấn đề.
2. Thích nghi phần cứng: Một số phần mềm nguồn mở có thể gặp khó khăn trong việc tương thích với một số phần cứng đặc biệt, vì không có sự tối ưu hóa cho từng môi trường cụ thể.
3. Độ phức tạp và học tập: Sử dụng và hiểu PMNM có thể đòi hỏi một quá trình học tập và làm quen với các công cụ và quy trình phát triển PMNM.

Phần mềm không nguồn đóng (PMNĐ):

1. Hỗ trợ chuyên nghiệp: PMNĐ thường đi kèm với sự hỗ trợ chuyên nghiệp từ nhà cung cấp phần mềm, bao gồm cài đặt, bảo trì, và hỗ trợ kỹ thuật. Điều này giúp người dùng có sự an tâm và hỗ trợ nhanh chóng khi cần.
2. Tính ổn định và đáng tin cậy: PMNĐ thường được kiểm tra, thử nghiệm và chứng nhận bởi nhà cung cấp, đảm bảo tính ổn định và đáng tin cậy. Người dùng có thể tin tưởng vào hiệu suất và tính năng của phần mềm.
3. Tích hợp và tương thích: PMNĐ thường được thiết kế để hoạt động tốt trên nhiều nền tảng và môi trường khác nhau. Các nhà cung cấp thường đảm bảo tích hợp và tương thích tốt với phần cứng và phần mềm khác.
4. Bảo mật và quyền riêng tư: PMNĐ thường được bảo vệ bởi các biện pháp bảo mật và quyền riêng tư chặt chẽ. Nhà cung cấp đảm bảo rằng phần mềm được bảo vệ khỏi các lỗ hổng bảo mật và đảm bảo quyền riêng tư của người dùng.

Nhược điểm:

1. Chi phí cao: PMNĐ thường có giá cả cao hơn so với PMNM, đòi hỏi người dùng phải mua bản quyền sử dụng và trả phí theo giấy phép từ nhà cung cấp phần mềm. Điều này có thể làm tăng chi phí và hạn chế sự tiếp cận đối với một số người dùng.
2. Tính linh hoạt và tùy chỉnh: PMNĐ có giới hạn về tính linh hoạt và khả năng tùy chỉnh. Người dùng không được phép truy cập vào mã nguồn và sửa đổi phần mềm theo ý muốn.
3. Phụ thuộc vào nhà cung cấp: PMNĐ đặt người dùng phụ thuộc vào nhà cung cấp phần mềm, bao gồm cả hỗ trợ và ph

4. Tính khả dụng: PMNĐ thường có sẵn các tính năng và chức năng phong phú, được phát triển và tối ưu hóa bởi các chuyên gia trong lĩnh vực. Người dùng có thể tận dụng ngay các tính năng này mà không cần phải tự phát triển từ đầu.

1. Đảm bảo pháp lý: Với PMNĐ, người dùng có một bản quyền hợp pháp để sử dụng phần mềm, được bảo vệ bởi các quy định pháp lý và giấy phép sử dụng từ nhà cung cấp. Điều này giúp người dùng đảm bảo tuân thủ quyền sở hữu trí tuệ và tránh các vấn đề pháp lý.

Tóm lại, PMNM và PMNĐ đều có ưu điểm và nhược điểm riêng. PMNM thường tập trung vào tính linh hoạt, sự cộng tác và tiết kiệm chi phí, trong khi PMNĐ tập trung vào tính ổn định, hỗ trợ chuyên nghiệp và tuân thủ pháp lý. Lựa chọn giữa hai mô hình này phụ thuộc vào nhu cầu và ưu tiên của người dùng, cũng như các yếu tố như chi phí, linh hoạt và đáng tin cậy.

1. Sự quan tâm của các tổ chức/chính phủ Việt Nam và trên thế giới với PMNM

ự quan tâm của các tổ chức và chính phủ đối với PMNM (Phần mềm nguồn mở) có xu hướng gia tăng cả ở Việt Nam và trên thế giới. Dưới đây là một số ví dụ về sự quan tâm này:

1. Ứng dụng công nghệ thông tin trong quản lý chính phủ: Các chính phủ trên thế giới, bao gồm cả Việt Nam, đã nhận ra tiềm năng và lợi ích của PMNM trong việc cung cấp các giải pháp công nghệ cho quản lý chính phủ. Sử dụng PMNM có thể giảm chi phí, tăng tính linh hoạt và khả năng tùy chỉnh, cũng như đảm bảo sự an toàn và quản lý dữ liệu hiệu quả.
2. Phát triển và khuyến khích PMNM trong các ngành công nghiệp: Các tổ chức và chính phủ đang đẩy mạnh việc sử dụng PMNM trong các ngành công nghiệp quan trọng như y tế, giáo dục, năng lượng, giao thông, và khai thác tài nguyên. Việc sử dụng PMNM trong các ngành này có thể giúp tăng hiệu suất, giảm chi phí và tạo ra những giải pháp tốt hơn cho cộng đồng.
3. Khuyến khích sáng tạo và khởi nghiệp: Các tổ chức và chính phủ đang tạo điều kiện và khuyến khích sự sáng tạo và khởi nghiệp trong lĩnh vực PMNM. Điều này bao gồm việc cung cấp hỗ trợ tài chính, cơ sở hạ tầng và đào tạo để khuyến khích các nhà phát triển phần mềm và các dự án PMNM mới.
4. Mở rộng quyền truy cập và khả năng sử dụng phần mềm: Một số chính phủ và tổ chức đang quan tâm đến việc mở rộng quyền truy cập và khả năng sử dụng phần mềm cho mọi người, bao gồm cả các khía cạnh như giáo dục, truy cập công cộng và phát triển khu vực nông thôn. PMNM được xem là một giải pháp tiềm năng để đáp ứng nhu cầu này, do tính linh hoạt và chi phí thấp hơn so với phần mềm đóng.

Tuy nhiên, việc quan tâm và khuyến khích PMNM không chỉ giới hạn ở Việt Nam và một số chính phủ trên thế giới. Trên thực tế, có một số tổ chức và chính phủ lớn trên toàn cầu đã thể hiện sự quan tâm đáng kể đối với PMNM, và điều này có một số lợi ích chung:

1. Giảm phụ thuộc vào nhà cung cấp duy nhất: Sử dụng PMNM cho phần mềm quan trọng giúp giảm phụ thuộc vào một nhà cung cấp duy nhất. Điều này giúp tránh tình trạng khóa chéo (vendor lock-in) và tạo điều kiện cho sự lựa chọn và thay đổi nhà cung cấp phần mềm.
2. Tăng tính tương thích và tích hợp: PMNM thường có mã nguồn mở, cho phép các tổ chức và chính phủ có thể tùy chỉnh và tích hợp phần mềm theo nhu cầu cụ thể của họ. Điều này giúp tạo ra các giải pháp phần mềm tương thích và tích hợp tốt hơn với các hệ thống hiện có.
3. Kiểm tra mã nguồn mở và đảm bảo an ninh: Có thể kiểm tra mã nguồn mở của PMNM để phát hiện và khắc phục lỗ hổng bảo mật nhanh chóng. Điều này giúp đảm bảo an ninh thông tin và giảm rủi ro về bảo mật.
4. Cộng đồng phát triển và cộng tác: PMNM thường thu hút cộng đồng phát triển rộng lớn, bao gồm các nhà phát triển, người dùng và các chuyên gia trong lĩnh vực. Các tổ chức và chính phủ có thể tận dụng cộng đồng này để chia sẻ kiến thức, kinh nghiệm và cộng tác để phát triển và nâng cao chất lượng của phần mềm.

Tóm lại, sự quan tâm và khuyến khích PMNM không chỉ giới hạn ở Việt Nam mà còn trên toàn cầu. PMNM mang lại nhiều lợi ích như giảm phụ thuộc vào nhà cung cấp duy nhất, tăng tính tương thích và tích hợp, đảm bảo an ninh và khuyến khích cộng đồng phát triển và cộng tác.

1. Các nhóm giấy phép bản quyền tự do, sự khác nhau giữa các loại giấy phép  
   aGPL  
   b.BSD  
   c.Apache  
   d. MIT/X11

Các nhóm giấy phép bản quyền tự do (free software licenses) cung cấp các điều khoản và điều kiện cho việc sử dụng, phân phối và sửa đổi phần mềm. Dưới đây là các loại giấy phép phổ biến và khác nhau:

a. Giấy phép GPL (General Public License):

* GPL là một giấy phép bản quyền tự do phổ biến nhất, được tạo bởi Hiệp hội Phần mềm Tự do (Free Software Foundation - FSF).
* GPL yêu cầu rằng mọi phiên bản sửa đổi và phân phối phải giữ nguyên giấy phép GPL và phải có sẵn mã nguồn.
* Điều này đảm bảo rằng tất cả các phiên bản sửa đổi cũng phải là phần mềm nguồn mở và giữ miễn phí.

b. Giấy phép BSD:

* BSD là một nhóm các giấy phép bản quyền tự do dựa trên các hệ điều hành BSD (Berkeley Software Distribution).
* Giấy phép BSD cho phép người dùng tự do sử dụng, phân phối và sửa đổi mã nguồn mà không có yêu cầu đặc biệt.
* BSD cho phép mã nguồn mở và độc quyền, điều này cho phép người dùng tích hợp mã nguồn BSD vào các dự án độc quyền.

c. Giấy phép Apache:

* Giấy phép Apache là một giấy phép bản quyền tự do phổ biến cho phần mềm nguồn mở.
* Giấy phép này cho phép người dùng tự do sử dụng, phân phối và sửa đổi phần mềm, miễn là giấy phép và thông báo bản quyền được bảo toàn.
* Ngoài ra, giấy phép Apache cũng đưa ra một số yêu cầu và hạn chế đối với việc sử dụng nhãn thương hiệu và quảng cáo.

d. Giấy phép MIT/X11:

* Giấy phép MIT/X11 (còn được gọi là giấy phép Expat) là một giấy phép bản quyền tự do đơn giản và linh hoạt.
* Giấy phép này cho phép người dùng tự do sử dụng, sao chép, sửa đổi, hợp nhất, phân phối và bán bản sao của phần mềm, miễn là thông báo bản quyền và giấy phép được bảo toàn.

Điểm khác biệt giữa các loại giấy phép này bao gồm:

* Yêu cầu giữ mã nguồn: GPL yêu cầu việc phải giữ mã nguồn và phải cung cấp mã nguồn khi phân phối, trong khi BSD, Apache và MIT/X11 không có yêu cầu này. Điều này có nghĩa là các phiên bản sửa đổi của phần mềm GPL phải cũng là phần mềm nguồn mở, trong khi các giấy phép khác cho phép tích hợp mã nguồn vào các dự án độc quyền.
* Yêu cầu về quyền sở hữu: GPL đặt một ràng buộc chặt chẽ về quyền sở hữu và tác giả phải giữ lại quyền sở hữu của tác phẩm. Trong khi đó, BSD, Apache và MIT/X11 không đặt hạn chế này và cho phép tác giả giữ quyền sở hữu hoặc chuyển giao quyền sở hữu theo ý muốn.
* Yêu cầu phụ thuộc vào giấy phép: Khi sử dụng mã nguồn mở được phân phối theo giấy phép GPL, tất cả các dự án hoặc phần mềm phụ thuộc vào mã nguồn đó cũng phải tuân theo GPL. Trong khi đó, các giấy phép BSD, Apache và MIT/X11 không đặt hạn chế này, cho phép tích hợp mã nguồn vào các dự án độc lập và sử dụng các giấy phép khác.

Tổng quan, GPL là một giấy phép bản quyền tự do có yêu cầu nghiêm ngặt nhất về việc giữ mã nguồn và quyền sở hữu, trong khi BSD, Apache và MIT/X11 cho phép tích hợp mã nguồn vào các dự án độc quyền và không yêu cầu giữ mã nguồn. Sự lựa chọn giấy phép thích hợp phụ thuộc vào mục đích sử dụng và các yêu cầu cụ thể của dự án phần mềm.

1. Tổng quan GNU/Linux
   1. Lịch sử hình thành, quá trình phát triển.
   2. Cấu trúc/kiến trúc của HĐH Linux.
   3. Các lệnh cơ bản trong quá trình sử dụng Linux
   4. So sánh Linux với các HĐH BSD như FreeBSD, NetBSD

a. Lịch sử hình thành và quá trình phát triển của GNU/Linux:

* GNU/Linux được hình thành từ sự kết hợp giữa hệ điều hành GNU (GNU's Not Unix) và kernel Linux.
* GNU là một dự án phần mềm nguồn mở, được khởi xướng bởi Richard Stallman vào năm 1983, với mục tiêu xây dựng một hệ điều hành hoàn toàn tự do và nguồn mở.
* Linux là một kernel (nhân) của hệ điều hành, được phát triển bởi Linus Torvalds vào năm 1991. Linux đã được kết hợp với các thành phần phần mềm của GNU để tạo ra hệ điều hành GNU/Linux.

b. Cấu trúc/kiến trúc của hệ điều hành Linux:

* Hệ điều hành Linux sử dụng kiến trúc hạt nhân (kernel-based architecture), trong đó kernel Linux là trung tâm của hệ thống.
* Cấu trúc hệ điều hành Linux bao gồm kernel, shell, giao diện người dùng, các tiện ích và các ứng dụng phần mềm khác.
* Kernel Linux quản lý các tài nguyên phần cứng và cung cấp các dịch vụ cơ bản cho các tiến trình và ứng dụng.
* Shell là một giao diện dòng lệnh giúp người dùng tương tác với hệ thống bằng cách nhập lệnh và nhận kết quả.
* Giao diện người dùng có thể là giao diện dòng lệnh (command-line interface) hoặc giao diện đồ họa (graphical user interface), như GNOME, KDE, hoặc Xfce.

c. Các lệnh cơ bản trong quá trình sử dụng Linux:

* Lệnh ls: Hiển thị danh sách các tệp tin và thư mục.
* Lệnh cd: Di chuyển đến thư mục khác.
* Lệnh pwd: Hiển thị đường dẫn thư mục hiện tại.
* Lệnh mkdir: Tạo thư mục mới.
* Lệnh rmdir: Xóa thư mục trống.
* Lệnh cp: Sao chép tệp tin và thư mục.
* Lệnh rm: Xóa tệp tin và thư mục.
* Lệnh mv: Di chuyển và đổi tên tệp tin và thư mục.
* Lệnh cat: Hiển thị nội dung của tệp tin.
* Lệnh grep: Tìm kiếm và lọc các dòng trong tệp tin dựa trên mẫu.

d. So sánh Linux với các hệ điều hành BSD như FreeBSD, NetBSD:

* Linux và BSD (Berkeley Software Distribution) là hai hệ điều hành nguồn mở phổ biến trong cộng đồng phần mềm tự do. Tuy có nhiều điểm tương đồng, nhưng cũng có một số khác biệt quan trọng:

1. Nguyên gốc và lịch sử: Linux bắt nguồn từ kernel Linux và kết hợp với các thành phần phần mềm GNU, trong khi BSD xuất phát từ mã nguồn mở của hệ điều hành Unix tại Đại học California, Berkeley.
2. Giấy phép: Linux sử dụng chủ yếu giấy phép GPL, trong khi BSD sử dụng các giấy phép BSD (với các biến thể khác nhau). Điều này tạo ra một số khác biệt trong việc sử dụng, phân phối và tương tác với các dự án phần mềm khác.
3. Quản lý dự án: Linux có một mô hình phát triển phân tán, với nhiều nhà phát triển đóng góp từ khắp nơi trên thế giới. BSD có mô hình phát triển tập trung hơn, với một số dự án BSD chủ yếu được duy trì bởi một nhóm nhỏ các nhà phát triển.
4. Tiêu chuẩn và tương thích: Linux có một số phiên bản và phân phối khác nhau, trong khi BSD có các phiên bản chính như FreeBSD, NetBSD, OpenBSD. Các phiên bản BSD thường được coi là có tính ổn định cao và tuân thủ tiêu chuẩn Unix, trong khi Linux có tính linh hoạt cao và hỗ trợ nhiều phần cứng.
5. Sử dụng và ứng dụng: Linux được sử dụng rộng rãi trên máy tính cá nhân, máy chủ và các thiết bị nhúng. BSD thường được ưu tiên sử dụng trong các môi trường máy chủ và hệ thống mạng, nơi tính ổn định và bảo mật được đánh giá cao.

Tóm lại, Linux và BSD đều là hệ điều hành nguồn mở phổ biến, mỗi loại có những ưu điểm riêng và phù hợp với các tình huống sử dụng khác nhau. Lựa chọn giữa Linux và BSD phụ thuộc vào nhu cầu cụ thể và sự ưu tiên của người dùng hoặc tổ chức.

1. HĐH Linux Ubuntu

a. Tổ chức Ubuntu. Cách đánh số phiên bản Ubuntu

b. Quản lý tệp tin, làm việc với thư mục trên Ubuntu: hệ thống tệp tin gồm  
những thư mục nào? (bin, boot, dev,…) bên trong chứa những gì?

a. Tổ chức Ubuntu và cách đánh số phiên bản Ubuntu:

* Ubuntu là một hệ điều hành Linux dựa trên Debian, được phát triển bởi công ty Canonical Ltd.
* Ubuntu có một cộng đồng lớn và hoạt động dựa trên các nguyên tắc tự do và mã nguồn mở.
* Cách đánh số phiên bản Ubuntu được thể hiện bằng hai số, ví dụ: Ubuntu 20.04. Đầu số đầu tiên là năm phát hành và đầu số thứ hai là tháng phát hành. Ví dụ: phiên bản 20.04 được phát hành vào tháng 4 năm 2020.

b. Quản lý tệp tin và làm việc với thư mục trên Ubuntu:

* Hệ thống tệp tin trên Ubuntu tuân theo cấu trúc tiêu chuẩn của hệ điều hành Linux.
* Các thư mục cơ bản trong hệ thống tệp tin Ubuntu bao gồm:
  1. /bin: Chứa các tệp lệnh thực thi cần thiết cho người dùng và hệ thống.
  2. /boot: Chứa các tệp tin khởi động và thông tin về hạt nhân (kernel).
  3. /dev: Chứa các tệp thiết bị (devices) ảo và vật lý trên hệ thống.
  4. /etc: Chứa các tệp cấu hình hệ thống và tệp cấu hình ứng dụng.
  5. /home: Thư mục chứa các thư mục người dùng cá nhân.
  6. /lib và /lib64: Chứa các thư viện hỗ trợ cho các chương trình thực thi.
  7. /media và /mnt: Được sử dụng để gắn kết các thiết bị lưu trữ (đĩa CD, đĩa USB, ổ cứng...) vào hệ thống tệp tin.
  8. /opt: Chứa các ứng dụng cài đặt từ bên thứ ba.
  9. /root: Thư mục người dùng root (quyền quản trị cao nhất trên hệ thống).
  10. /tmp: Thư mục tạm thời dùng để lưu trữ các tệp tin tạm thời.
  11. /usr: Chứa các tệp tin, thư mục và tài nguyên chung của người dùng.
  12. /var: Chứa các tệp tin biến đổi và dữ liệu thay đổi trong quá trình chạy của hệ thống.

9) Các câu lệnh trong Linux:

a, Các lệnh quản lý user (useradd, userdel, …)

1. useradd: Tạo một người dùng mới trên hệ thống. Ví dụ: **useradd username**
2. userdel: Xóa một người dùng khỏi hệ thống. Ví dụ: **userdel username**
3. passwd: Thay đổi mật khẩu cho một người dùng. Ví dụ: **passwd username**
4. usermod: Thay đổi thông tin của một người dùng. Ví dụ: **usermod -c "New User Name" username**
5. chage: Thay đổi các thuộc tính mật khẩu của người dùng, ví dụ như hạn chế tuổi hạn mật khẩu. Ví dụ: **chage -M 90 username** (thiết lập hạn chế thay đổi mật khẩu trong 90 ngày)
6. su: Chuyển đổi người dùng hiện tại sang một người dùng khác. Ví dụ: **su username** (chuyển đổi sang người dùng có tên "username")
7. id: Hiển thị thông tin về người dùng được chỉ định. Ví dụ: **id username**
8. finger: Hiển thị thông tin chi tiết về người dùng, bao gồm cả tên và thông tin liên hệ. Ví dụ: **finger username**
9. whoami: Hiển thị tên người dùng hiện tại đang đăng nhập. Ví dụ: **whoami**
10. w: Hiển thị danh sách người dùng đang đăng nhập vào hệ thống và các tiến trình liên quan. Ví dụ: **w**

b, Các lệnh quản lý thư mục tệp tin (ls, cd, pwd, mkdir, rmdir, cp, rm, …)

1. ls: Liệt kê nội dung của thư mục hiện tại hoặc thư mục được chỉ định. Ví dụ: **ls** (liệt kê nội dung của thư mục hiện tại), **ls /path/to/directory** (liệt kê nội dung của thư mục đã cho)
2. cd: Thay đổi thư mục làm việc hiện tại sang một thư mục khác. Ví dụ: **cd /path/to/directory** (thay đổi sang thư mục đã cho), **cd ..** (thay đổi sang thư mục cha)
3. pwd: Hiển thị đường dẫn tuyệt đối của thư mục làm việc hiện tại. Ví dụ: **pwd**
4. mkdir: Tạo một thư mục mới. Ví dụ: **mkdir directoryname**
5. rmdir: Xóa một thư mục trống. Ví dụ: **rmdir directoryname**
6. cp: Sao chép tệp tin và thư mục. Ví dụ: **cp sourcefile destination** (sao chép tệp tin), **cp -r sourcedirectory destination** (sao chép thư mục)
7. rm: Xóa tệp tin và thư mục. Ví dụ: **rm filename** (xóa tệp tin), **rm -r directoryname** (xóa thư mục)
8. mv: Di chuyển (đổi tên) tệp tin và thư mục. Ví dụ: **mv oldname newname** (đổi tên tệp tin/thư mục), **mv filename directory** (di chuyển tệp tin vào thư mục)
9. touch: Tạo một tệp tin mới hoặc cập nhật thời gian sửa đổi của tệp tin hiện có. Ví dụ: **touch filename**
10. find: Tìm kiếm tệp tin và thư mục dựa trên các tiêu chí nhất định. Ví dụ: **find /path/to/directory -name filename** (tìm kiếm tệp tin có tên "filename" trong thư mục đã cho)

c,Các lệnh hiển thị nội dung file (cat, less, more,…)

1. cat: Hiển thị toàn bộ nội dung của tệp tin. Ví dụ: **cat filename**
2. less: Hiển thị nội dung của tệp tin một cách trang trí và cho phép cuộn lên xuống. Ví dụ: **less filename**
3. more: Hiển thị nội dung của tệp tin dưới dạng từng trang. Ví dụ: **more filename**
4. head: Hiển thị một số dòng đầu tiên của tệp tin. Ví dụ: **head -n 10 filename** (hiển thị 10 dòng đầu tiên)
5. tail: Hiển thị một số dòng cuối cùng của tệp tin. Ví dụ: **tail -n 5 filename** (hiển thị 5 dòng cuối cùng)
6. grep: Hiển thị các dòng trong tệp tin chứa một chuỗi cụ thể. Ví dụ: **grep "keyword" filename** (hiển thị các dòng chứa "keyword")
7. awk: Xử lý và hiển thị nội dung của tệp tin theo các quy tắc xác định. Ví dụ: **awk '{print $1}' filename** (hiển thị cột đầu tiên trong mỗi dòng)
8. sed: Hiệu chỉnh và hiển thị nội dung của tệp tin dựa trên các quy tắc xác định. Ví dụ: **sed 's/old/new/g' filename** (thay thế tất cả các chuỗi "old" bằng chuỗi "new")

d, Các lệnh quản lý tiến trình (top, ps, kill, …)

1. ps: Liệt kê các tiến trình đang chạy trên hệ thống. Ví dụ: **ps aux** (liệt kê tất cả các tiến trình)
2. top: Hiển thị danh sách các tiến trình đang chạy và thông tin thống kê về tài nguyên hệ thống. Ví dụ: **top**
3. htop: Giao diện tương tác đồ họa cho việc xem và quản lý các tiến trình đang chạy. Ví dụ: **htop**
4. kill: Gửi một tín hiệu để kết thúc một tiến trình. Ví dụ: **kill process\_id** (kết thúc tiến trình bằng ID tiến trình)
5. killall: Kết thúc tất cả các tiến trình với một tên cụ thể. Ví dụ: **killall process\_name** (kết thúc tất cả các tiến trình có tên cụ thể)
6. pkill: Kết thúc tiến trình dựa trên tên hoặc mô tả tiến trình. Ví dụ: **pkill -f process\_name** (kết thúc tiến trình dựa trên tên tiến trình)
7. bg: Chuyển một tiến trình vào chế độ nền (background). Ví dụ: **bg job\_id** (chuyển tiến trình vào chế độ nền)
8. fg: Chuyển một tiến trình từ chế độ nền về chế độ tiếp tục (foreground). Ví dụ: **fg job\_id** (chuyển tiến trình vào chế độ tiếp tục)
9. nice: Thay đổi độ ưu tiên của một tiến trình. Ví dụ: **nice -n value command** (thiết lập độ ưu tiên của tiến trình)
10. renice: Thay đổi độ ưu tiên của một tiến trình đang chạy. Ví dụ: **renice value -p process\_id** (thiết lập độ ưu tiên của tiến trình bằng ID tiến trình)

e, Các lệnh quy định quyền truy cập thư mục, tập tin

1. chmod: Thay đổi quyền truy cập của thư mục và tệp tin. Ví dụ: **chmod permissions filename** (thay đổi quyền truy cập của tệp tin)
2. chown: Thay đổi chủ sở hữu của thư mục và tệp tin. Ví dụ: **chown ownername filename** (thay đổi chủ sở hữu của tệp tin)
3. chgrp: Thay đổi nhóm sở hữu của thư mục và tệp tin. Ví dụ: **chgrp groupname filename** (thay đổi nhóm sở hữu của tệp tin)
4. umask: Thiết lập giá trị umask để quy định quyền mặc định cho các tệp tin và thư mục mới. Ví dụ: **umask 022** (thiết lập quyền mặc định là 644 cho tệp tin và 755 cho thư mục)
5. suid: Đặt quyền setuid cho tệp tin, cho phép người dùng thực thi tệp tin với quyền của chủ sở hữu. Ví dụ: **chmod u+s filename** (đặt quyền setuid cho tệp tin)
6. sgid: Đặt quyền setgid cho thư mục, khi một tệp tin được tạo trong thư mục đó, nó sẽ kế thừa nhóm sở hữu của thư mục. Ví dụ: **chmod g+s directoryname** (đặt quyền setgid cho thư mục)
7. sticky bit: Đặt sticky bit cho thư mục, chỉ cho phép chủ sở hữu tệp tin xóa tệp tin trong thư mục. Ví dụ: **chmod +t directoryname** (đặt sticky bit cho thư mục)

f, Các lệnh quy định quyền sở hữu thư mục, tập tin

1. chown: Thay đổi chủ sở hữu của thư mục và tệp tin. Ví dụ: **chown ownername filename** (thay đổi chủ sở hữu của tệp tin)
2. chgrp: Thay đổi nhóm sở hữu của thư mục và tệp tin. Ví dụ: **chgrp groupname filename** (thay đổi nhóm sở hữu của tệp tin)
3. chown -R: Thay đổi chủ sở hữu của thư mục và tất cả các tệp tin bên trong. Ví dụ: **chown -R ownername directoryname** (thay đổi chủ sở hữu của thư mục và tất cả các tệp tin bên trong)
4. chgrp -R: Thay đổi nhóm sở hữu của thư mục và tất cả các tệp tin bên trong. Ví dụ: **chgrp -R groupname directoryname** (thay đổi nhóm sở hữu của thư mục và tất cả các tệp tin bên trong)
5. sudo: Sử dụng lệnh sudo để thực hiện các lệnh quy định quyền sở hữu với quyền root (quyền quản trị). Ví dụ: **sudo chown ownername filename** (thay đổi chủ sở hữu của tệp tin với quyền root)

I, Các lệnh lấy thông tin về user (who, groups, finger,…)

1. who: Hiển thị danh sách người dùng đã đăng nhập vào hệ thống. Ví dụ: **who**
2. w: Hiển thị thông tin chi tiết về người dùng đã đăng nhập và các tiến trình đang chạy của họ. Ví dụ: **w**
3. id: Hiển thị thông tin về người dùng hiện tại, bao gồm UID (User ID), GID (Group ID) và các nhóm người dùng mà người dùng thuộc về. Ví dụ: **id**
4. groups: Hiển thị danh sách các nhóm người dùng mà người dùng thuộc về. Ví dụ: **groups**
5. finger: Hiển thị thông tin chi tiết về một người dùng cụ thể, bao gồm tên, địa chỉ email, thời gian đăng nhập và thông tin khác (nếu có). Ví dụ: **finger username**
6. whoami: Hiển thị tên người dùng hiện tại. Ví dụ: **whoami**
7. getent: Truy xuất thông tin người dùng và nhóm người dùng từ các cơ sở dữ liệu hệ thống như **/etc/passwd** và **/etc/group**. Ví dụ: **getent passwd username** (lấy thông tin về người dùng)

g, Các lệnh tìm kiếm thư mục, tệp tin (find, locate,…)

1. find: Tìm kiếm thư mục và tệp tin dựa trên các tiêu chí nhất định, như tên, kích thước, quyền truy cập, v.v. Ví dụ: **find /path/to/search -name "filename"** (tìm kiếm tệp tin có tên là "filename")
2. locate: Tìm kiếm thư mục và tệp tin bằng cách tra cứu trong cơ sở dữ liệu mặc định đã được tạo trước. Ví dụ: **locate filename** (tìm kiếm tệp tin có tên là "filename")
3. which: Tìm kiếm đường dẫn của một chương trình thực thi trong các thư mục được liệt kê trong biến môi trường PATH. Ví dụ: **which command** (tìm đường dẫn của chương trình "command")
4. whereis: Tìm kiếm đường dẫn của các tệp tin thực thi, mã nguồn và tài liệu liên quan đến một chương trình. Ví dụ: **whereis command** (tìm đường dẫn liên quan đến chương trình "command")
5. grep: Tìm kiếm một chuỗi cụ thể trong nội dung của các tệp tin. Ví dụ: **grep "pattern" file** (tìm kiếm chuỗi "pattern" trong tệp tin)
6. rgrep: Tìm kiếm một chuỗi cụ thể trong nội dung của các tệp tin, bao gồm các thư mục con. Ví dụ: **rgrep "pattern" directory** (tìm kiếm chuỗi "pattern" trong thư mục "directory" và các thư mục con)
7. mlocate: Tìm kiếm thư mục và tệp tin bằng cách tra cứu trong cơ sở dữ liệu mlocate, một cơ sở dữ liệu tìm kiếm nhanh hơn locate. Ví dụ: **mlocate filename** (tìm kiếm tệp tin có tên là "filename")

h, Các lệnh sử dụng trong trình soạn thảo VI/VIM

1. Khởi động vi/vim: **vi filename** hoặc **vim filename**: Mở tệp tin "filename" để chỉnh sửa.
2. Chế độ lệnh (Command mode):
   * **i**: Chuyển sang chế độ chèn (Insert mode).
   * **x**: Xóa ký tự tại vị trí con trỏ.
   * **dd**: Xóa dòng hiện tại.
   * **yy**: Sao chép dòng hiện tại.
   * **p**: Dán nội dung đã sao chép (paste).
   * **:w**: Lưu (ghi) tệp tin.
   * **:q**: Thoát khỏi vi/vim.
3. Chế độ chèn (Insert mode):
   * Nhấn phím "i" để chuyển sang chế độ chèn và bắt đầu nhập văn bản.
   * Nhấn phím "Esc" để quay lại chế độ lệnh.
4. Chế độ tìm kiếm:
   * Nhấn phím **/** để chuyển sang chế độ tìm kiếm.
   * Nhập từ khóa tìm kiếm và nhấn "Enter" để tìm kiếm tiếp theo.
   * Nhấn phím "n" để tìm kiếm tiếp theo và "N" để tìm kiếm ngược lại.
5. Di chuyển trong tệp tin:
   * **h**: Di chuyển sang trái.
   * **j**: Di chuyển xuống dòng dưới.
   * **k**: Di chuyển lên dòng trên.
   * **l**: Di chuyển sang phải.
   * **G**: Di chuyển tới cuối tệp tin.
   * **gg**: Di chuyển tới đầu tệp tin.
   * **<line\_number>G**: Di chuyển tới dòng số "line\_number".
6. Thao tác với khối văn bản:
   * **v**: Bắt đầu chế độ chọn văn bản (Visual mode).
   * Di chuyển con trỏ để chọn khối văn bản.
   * **d**: Xóa khối văn bản đã chọn.
   * **y**: Sao chép khối văn bản đã chọn.

K, Các lệnh cấu hình mạng (ifconfig, route, ping,…)

1. ifconfig: Hiển thị và cấu hình các thông tin liên quan đến giao diện mạng. Ví dụ: **ifconfig** hoặc **ifconfig eth0**
2. ip: Hiển thị và cấu hình các thông tin liên quan đến giao diện mạng (phiên bản mới hơn của ifconfig). Ví dụ: **ip addr show** hoặc **ip addr show eth0**
3. route: Hiển thị và quản lý bảng định tuyến. Ví dụ: **route** hoặc **route -n**
4. netstat: Hiển thị các kết nối mạng, bảng định tuyến và thống kê mạng. Ví dụ: **netstat -a** (hiển thị tất cả các kết nối mạng)
5. ping: Gửi gói tin ICMP echo request đến một địa chỉ IP để kiểm tra kết nối mạng và đo thời gian đáp lại. Ví dụ: **ping IP\_address** (ping đến địa chỉ IP)
6. traceroute: Hiển thị tuyến đường mà gói tin mạng đi qua từ nguồn đến đích, đồng thời đo độ trễ trên mỗi nút. Ví dụ: **traceroute IP\_address** (hiển thị tuyến đường đến địa chỉ IP)
7. nslookup: Truy vấn thông tin DNS của một tên miền hoặc địa chỉ IP. Ví dụ: **nslookup domain\_name** hoặc **nslookup IP\_address**
8. hostname: Hiển thị hoặc cấu hình tên máy chủ (hostname). Ví dụ: **hostname** hoặc **hostname new\_hostname** (thiết lập tên máy chủ)
9. systemctl: Quản lý các dịch vụ hệ thống, bao gồm cả dịch vụ mạng. Ví dụ: **systemctl start service\_name** (bắt đầu dịch vụ)