

**Đơn vị Thực tập: Deha Software**

**GVHD: Nguyễn Thị Hải Yến**

**Nhóm thực hiện: Nhóm 5**

**Lớp/khóa:** **LTMT4 –K9**

**BÁO CÁO THỰC TẬP TỐT NGHIỆP**

**Hà Nội, 13 tháng 2 năm 2020**

|  |  |
| --- | --- |
| **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** | |
|  | *Hà Nội., Ngày 13 tháng 2 năm 2020* |

**PHIẾU NHẬN XÉT THỰC TẬP**

**Họ tên sinh viên (nhóm):** Nhóm 5 **Lớp-khóa:** LTMT4-K9

**Cơ quan / Doanh nghiệp tiếp nhận:** Deha Software

**Địa chỉ:** Tầng 3 tòa nhà Intracom Building , 82 Dịch Vọng Hậu , Cầu Giấy , Hà Nội.

Ngày bắt đầu: 13/2/2020 ngày kết thúc: 8/3/2020

Nội dung thực tập: Giới thiệu , lĩnh vưc, công nghệ của công ty Deha Software.

Bài tập công ty Deha Software cho về nhà tìm hiểu .

1. **Nhận xét của cơ quan / Đơn vị về chất lượng công việc được giao:**

*Các công việc được giao:*

|  |  |  |
| --- | --- | --- |
| * Hoàn thành xuất sắc | * Khá | * Yếu |
| * Tốt | * Trung bình |  |

*Hoàn tất công việc được giao:*

|  |  |  |
| --- | --- | --- |
| * Hoàn thành đúng | * Thỉnh thoảng đúng | * Không đúng thời hạn |

*Tính hữu ích của đợt thực tập với cơ quan*

|  |  |  |
| --- | --- | --- |
| * Giúp ích nhiều | * Giúp ích ít | * Không giúp ích mấy cho cơ quan |

1. **Nhận xét của cơ quan / Đơn vị về bản thân sinh viên:**
   1. *Năng lực chuyên môn được sử dụng vào công việc được giao ở mức:*

|  |  |  |  |
| --- | --- | --- | --- |
| * Giỏi | * Khá | * Trung bình | * Yếu |

* 1. *Tinh thần, thái độ đối với công việc được giao:*

|  |  |  |
| --- | --- | --- |
| * Tích cực | * Bình thường | * Thiếu tích cực |

* 1. *Đảm bảo kỷ luật lao động (giờ giấc lao động, nghỉ làm,...):*

|  |  |  |
| --- | --- | --- |
| * Tốt | * Trung bình | * Kém |

* 1. *Thái độ đới với cán bộ, công nhân viên trong Cơ quan / Đơn vị:*

|  |  |  |
| --- | --- | --- |
| * Hòa đồng | * Không có gì đáng nói | * Rụt rè |

* 1. *Khả năng sử dụng phần mềm văn phòng (office):*

|  |  |  |  |
| --- | --- | --- | --- |
| * Giỏi | * Khá | * Trung bình | * Yếu |

* 1. *Khả năng sử dụng Tiếng Anh:*

|  |  |  |  |
| --- | --- | --- | --- |
| * Giỏi | * Khá | * Trung bình | * Yếu |

1. **Nhu cầu nhân lực của cơ quan (kiến thức, kỹ năng, số lượng):**

1. **Các nhận xét khác (nếu có):**

1. **Đánh giá:**
2. Điểm chuyên cần, tác phong và đạo đức (Đạt/Không đạt):
3. Điểm chuyên môn (Đạt/Không đạt):

|  |  |
| --- | --- |
| **Trưởng Cơ quan/Đơn vị**  (*Ký tên, đóng dấu và ghi rõ họ tên*) | **Người nhận xét**  (*Ký và ghi rõ họ tên*) |

**BẢN KẾ HOẠCH THỰC TẬP TỐT NGHIỆP NHÓM 5**

Họ và tên sv trong nhóm 5:

|  |  |  |  |
| --- | --- | --- | --- |
| Nguyễn Anh Dũng | LTMT4-K9 | 0362719343 | dung.na2410@gmail.com |
| Nguyễn Ngọc Sơn | LTMT4-K9 | 0967455825 | ngocsonab@gmail.com |
| Nguyễn Tiến Long | LTMT4-K9 | 0352771587 | nguyentienlong221@gmail.com |
| Đỗ Quốc Việt | LTMT4-K9 | 0357186998 | doquocviet268@gmail.com |
| Lê Hữu Thắng | LTMT4-K9 | 0362112690 | thangle.lht196@gmail.com |

**1. Nội dung thực tập:**

a) Giới thiệu , lĩnh vưc, công nghệ của công ty Deha Software.

b) Bài tập công ty Deha Software cho về nhà làm và tìm hiểu.

**2. Yêu cầu:** Thảo luận và làm bài thật tốt , nộp bài đúng hạn.

**3. Kế hoạch chi tiết**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ngày thực hiện** | **Công việc triển khai cụ thể** | **Người thực hiện** | **Kết quả đạt được *(theo tỷ lệ %)*** | **Ghi chú** |
| 13/2/2020 | Giới thiệu , lĩnh vưc, công nghệ của công ty Deha Software. | Dũng, Sơn ,Long, Việt , Thắng | 100% |  |
| 13/2/2020 | [Quản lý thư mục, file trong Unix/Linux](javascript:void(0)) : sao chép, di chuyển , xóa file và thư mục | Việt | 100% |  |
| 13/2/2020 | [Quyền hạn/Chế độ truy cập file trong Unix](javascript:void(0)) | Sơn | 100% |  |
| 13/2/2020 | Phân quyền tập tin và thư mục trên linux | Dũng | 100% |  |
| 13/2/2020 | Giải thích các flag: “-R” , “-rf” , “-r” trong câu lệnh Linux | Thắng | 100% |  |
| 13/2/2020 | Tìm hiểu về cấu trúc cây thư mục trong Linux ( thư mục /var , /bin , /home chứa gì?) | Long | 100% |  |
| 20/2/2020 | Quản lý phiên bản là gì? Quản lý phiên bản cục bộ, tập trung , phân tán là như thế nào? Git là gì? Lợi ích , ý nghĩa của việc sử dụng Git ? Các câu lệnh cơ bản trong Git. | Long | 100% |  |
| 20/2/2020 | Cách tạo , ý nghĩa và cách sử dụng các branch develop, feature , hotfix , release. | Sơn | 100% |  |
| 20/2/2020 | Các trường hợp dẫn đến conflict? Cách xử lý conflict. | Thắng | 100% |  |
| 20/2/2020 | Git workflow. | Dũng | 100% |  |
| 20/2/2020 | Git successful branching model. | Việt | 100% |  |

*Hà Nội, ngày 13 tháng 2 năm 2020*

**Xác nhận của Cán bộ hướng dẫn Chữ kí nhóm**

*(ký, ghi rõ họ và tên)*

Dũng , Sơn , Việt , Long, Thắng

**Giới thiệu về doanh nghiệp Deha software**

Được thành lập năm 2016, Công ty Cổ phần DEHA Việt Nam phát triển với 3 dịch vụ chính gồm: Phát triển phần mềm, Cung cấp các package và Tư vấn offshore. DEHA nằm trong Top 10 doanh nghiệp có năng lực công nghệ 4.0 tiêu biểu năm 2018 do VINASA bình chọn.

Coi con người là giá trị cốt lõi của công ty, DEHA luôn tự hào là tổ chức đào tạo lập trình viên uy tín cho người mới bắt đầu và các doanh nghiệp cần tìm nhân lực chất lượng cao hoặc muốn nâng cao trình độ lập trình cho nhân viên. Với môi trường làm việc năng động, mọi hoạt động của DEHA là nhằm chuyển giao liên tục, không giới hạn các giá trị cho các thành viên, để cùng nhau học tập, tiến bộ, tạo cơ hội phát triển cho nhân viên.

Thế mạnh của DEHA là

-Phát triển Website : Tham gia tương tác, tạo những trải nghiệm tốt chính là chìa khóa để kết nối với khách hàng. Bạn muốn xây dựng một website, một trang thương mại điện tử mới hay chỉnh sửa, thêm mới vào một site hiện có, chúng tôi luôn đáp ứng để đảm bảo trang web thực hiện đúng theo yêu cầu của bạn. Dịch vụ phát triển web của chúng tôi kết hợp giữa phương thức truyền thống và công nghệ hiện đại để tạo ra các trang web trực quan, dễ dàng mở rộng, thêm mới các chức năng.

-Phát triển Mobile App: Cho dù bạn muốn xây dựng sản phẩm mới, phát triển một ứng dụng song song, hoặc đưa ứng dụng sẵn có sang platform khác, công nghệ Dogfish Software mà DEHA sử dụng sẽ đưa ra các giải pháp linh hoạt, đáng tin cậy cho mọi hệ điều hành, thỏa mãn được mọi nhu cầu của bạn. Chúng tôi hiểu rằng môi trường mobile là luôn luôn thay đổi, vì vậy chúng tôi sẽ tư vấn những giải pháp phù hợp nhất cho mô hình kinh doanh của bạn.

-Bảo trì sản phẩm: Các nhà cung cấp sản phẩm phần mềm có thể sử dụng dịch vụ bảo trì offshore của DEHA một cách hiệu quả để bảo trì phần mềm hiện có, cũng như dịch vụ phát triển các ứng dụng mới mà DEHA cung cấp. DEHA hỗ trợ bạn lan truyền những ý tưởng sáng tạo với mức chi phí thấp và thúc đẩy tăng trưởng kinh doanh trong tương lai.

-Tư vấn Offshore : Dịch vụ tư vấn Offshore của chúng tôi sẽ hỗ trợ các doanh nghiệp trong việc đạt được tăng trưởng bền vững. DEHA đã có trên 10 năm kinh nghiệm trong việc cung cấp dịch vụ tư vấn Offshore cho các khách hàng ở mọi ngành nghề. Chúng tôi mong muốn giúp các khách hàng sử dụng công nghệ một cách sáng tạo để tạo ra các dịch vụ tốt hơn, các doanh nghiệp năng động hơn trong thời đại công nghệ thông tin bùng nổ.

Do đó sáng ngày 13/02/2020, học viên lớp LTMT4 tại trường CDNBKHN đã có chuyến thăm DEHA Software để có thêm những trải nghiệm thực tế từ doanh nghiệp. Đây là dịp để các bạn tìm hiểu về môi trường làm việc, học hỏi kinh nghiệm từ các nhân viên của công ty, nắm bắt cơ hội thực tập và việc làm.

Trong buổi tham quan, phía đại diện DEHA Software đã có những chia sẻ vô cùng cở mở, giới thiệu về công ty, những dự án, cơ hội việc làm và thực tập trong quá trình học đối với sinh viên trường CDNBKHN. Ngoài ra, các bạn cũng được lắng nghe những chia sẻ về văn hóa doanh nghiệp, những kinh nghiệm, bài học quý giá của những người làm trong nghề lập trình.

**ND ôn tập buổi 1**

**1.**[**Quản lý thư mục, file trong Unix/Linux**](javascript:void(0)) **: sao chép, di chuyển , xóa file và thư mục**

* 1. **Quản lí file:**
     + Sao chép file: sử dụng lệnh **cp**.

Cú pháp: $ cp **source\_file** **destination\_file**

Trong đó, **source\_file** là file chứa thông tin bạn cần sao chép đến **destination\_file.**

* + - Di chuyển file : sử dụng lệnh **mv.**

Cú pháp : $ **mv old\_file new\_file**

Lệnh **mv** sẽ di chuyển tất cả file đang tồn tại(**old\_file)** vào trong file mới(**new\_file).**

* + - Xóa file: sử dụng lệnh **rm.**

Cú pháp: $ rm **filename**

* 1. **Quản lí thư mục:**
     + Xóa thư mục: sử dụng lệnh **rmdir**.

Cú pháp: $rmdir **dirname**

$

* + - Thay đổi thư mục: sử dụng lệnh **cd** để tạo nhiều thay đổi tới thư mục home.

Cú pháp: $cd **dirname**

* + - Đặt lại tên thư mục: sử dụng lệnh **mv.**

Cú pháp: $mv **olddir newdir**

$

|  |  |
| --- | --- |
| cp *file1 file2* | chép *file1* sang *file2* |
| cp *file* /*thưmục* | chép *file* vào thư mục « *thưmục »* |
| cp -r *thưmục1 thưmục2* rsync -a *thưmục1 thưmục2* | chép toàn bộ nội dung của thư mục « *thưmục1 »* sang thư mục « *thưmục2 »* |
| mv *file1 file2* | chuyển tên tập tin *file1* thành tên *file2* |
| mv *thưmục1 thưmục2* | chuyển tên *thưmục1* thành *thưmục2* |
| mv file thưmục | chuyển tập tin *file* vào thư mục *thưmục* |
| mv file1 thưmục/file2 | chuyển *file1* vào thư mục *thưmục* đồng thời đổi tên tập tin thành *file2* |
| mkdir *thưmục* | tạo ra thư mục *thưmục* |
| mkdir -p *thưmục1/thưmục2* | tạo ra thư mục cha *thưmục1* và thư mục con *thưmục2* cùng lúc |
| rm *file* | xóa bỏ tập tin *file* trong thư mục hiện hành |
| rmdir *thưmục* | xóa bỏ thư mục trống mang tên *thưmục* |
| rm -rf *thưmục* | xóa bỏ thư mục mang tên *thưmục* với tất cả các tập tin trong đó *(force)* |

# 2.[Quyền hạn/Chế độ truy cập file trong Unix](javascript:void(0))

# Quyền hạn

Quyền sở hữu file là một thành phần quan trọng của [Unix](https://quantrimang.com/unix-va-linux) mà cung cấp phương thức bảo mật để lưu giữ file. Mọi file trong Unix có các thuộc tính sau để thể hiện quyền hạn truy cập tới nó

Các quyền hạn truy cập được chia thành ba nhóm: user, group, other, mà mỗi vị trí trong nhóm biểu hiện một quyền hạn truy cập cụ thể, theo thứ tự: đọc (r), ghi (w) và thực thi (x)

**Chế độ truy cập vào file trong Unix/Linux**

|  |  |  |
| --- | --- | --- |
|  | Thư mục | File |
| R(read) | Truy cập vào một thư mục với nghĩa là người sử dụng có thể đọc các nội dung. Người sử dụng có thể xem tên các file bên trong một thư mục. | Cho phép đọc nội dung của file. |
| W(write) | Cho phép người sử dụng có thể thêm hoặc xóa file nội dung của thư mục. | Cho phép chỉnh sửa hoặc gỡ bỏ nội dung của file |
| X(Execute) | Chạy một thư mục không thực sự mang nhiều ý nghĩa, vì thế bạn chỉ coi đó là giấy phép để ngang qua.  Một người sử dụng phải có quyền truy cập thực thi tới thư mục bin để chạy lệnh ls hoặc cd chẳng hạn. | Người sử dụng với quyền hạn truy cập thực thi có thể chạy một file như là một chương trình. |

**Thay đổi các quyền hạn truy cập trong Unix/Linux**

Để thay đổi quyền hạn truy cập của file hoặc thư mục, bạn sử dụng lệnh **chmod** (viết tắt của change mode). Có hai cách để sử dụng chmod là: chế độ tượng trưng (Symbolic Mode) và chế độ tuyệt đối (Absolute Mode).

Cú pháp : chmod

**Thay đổi người sở hữu và nhóm trong Unix/Linux**

Hai lệnh có sẵn để thay đổi sở hữu cá nhân và sở hữu nhóm của file là:

1. chown: Lệnh này đại diện cho thay đổi sở hữu cá nhân (change owner).
2. chgrp: Lệnh này đại diện cho thay đổi sở hữu nhóm (change group).

**Thay đổi quyền sở hữu trong Unix/Linux**

Lệnh chown thay đổi quyền sở hữu của một file. Cú pháp cơ bản là:

$ chown user filelist

## Thay đổi quyền sở hữu nhóm trong Unix/Linux

$ chgrp group filelist

**3.Phân quyền tập tin và thư mục trên linux**

**a.Cấu trúc của một tập tin, thư mục**

Chúng ta sẽ cùng xem cấu trúc của một tập tin, thư mục, chúng ta gõ lệnh sau:

# ls -al

Ý nghĩa của từng chỉ số

-rw-r--r--     1     root   root     0 Mar  6 21:04 .autofsck

-rw-r--r--     1     root   root     0 Dec  2 11:41 .autorelabel

dr-xr-xr-x.    2     root   root  4096 Mar  6 21:06 bin

dr-xr-xr-x.    4     root   root  4096 Dec  2 11:38 boot

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

^ ^  ^  ^      ^      ^      ^    ^      ^-- Ngày tháng tạo ra file và tên file

| |  |  |      |      |      |    |

| |  |  |      |      |      |    \--- Dung lượng của file/folder

| |  |  |      |      |      \-------- Tên group sở hữu

| |  |  |      |      \--------------- Tên user sở hữu

| |  |  |      \---------------------- Hard link, thể hiện nhiều file hoặc thư mục có cùng sử dụng chung inode

| |  |  |

| \--\--\----------------------------- Các chỉ số phân quyền

\------------------------------------- Loại file (chữ d nghĩa là thư mục)

**b.Phân quyền cho tập tin và thư mục**

Linux có 3 quyền cơ bản của 1 user và group đó là

* **r (read)** – quyền đọc file/folder
* **w (write)** – quyền ghi/sửa nội dung file/folder
* **x** **(execute)** – quyền thực thi (truy cập) thư mục. Đối với thư mục thì bạn cần phải có quyền execute thì mới dùng lệnh cd để truy cập vào được
* **–** **(Deny)** – Không có quyền

Các quyền này cũng được đánh số nhất định

* **r (read)** – được biểu diễn bằng số 4.
* **w** **(write)** – được biểu diễn bằng số 2.
* **x** **(execute)** – được biểu diễn bằng số 1.
* **–** **(Deny)** – được biểu diễn bằng số 0

Cấu trúc chỉ số phân quyền

* **owner**: Quyền của user mà chủ sở hữu của file này.
* **group**: Quyền của những users thuộc group mà chủ sở hữu của file này.
* **other**: Quyền của tất cả các user khác trên máy.

Nhìn hình phía trên , các bạn có thể đọc được thông số file test1 như sau

* owner có quyền r+w+- =  4+2+0 = 6
* group có quyền  r+-+- = 4+0+0 = 4
* other có quyền  r+-+- = 4+0+0 = 4

Thông số file test2 như sau

* owner có quyền r+w+x =  4+2+1 = 7
* group có quyền  r+-+x = 4+0+1 = 5
* other có quyền  r+-+x = 4+0+1 = 5

Thông số file test3 như sau

* owner có quyền r+w+x =  4+2+1 = 7
* group có quyền  r+w+- = 4+2+0 = 6
* other có quyền  r+w+- = 4+2+0 = 6

Chỉ số phân quyền của 3 file sẽ là test1=644, test2=755, test3=766

Để có thể thay đổi các chỉ số phân quyền này ta sẽ dử dụng lệnh chmod

# chmod  <tùy chọn> <chỉ số phân quyền> <tên tập tin/thư mục>

Các tùy chọn

* **-v** : Hiển thị báo cáo sau khi chạy lệnh, mỗi lần đổi quyền là hiển thị một lần
* **-c** : Giống như trên, nhưng chỉ hiện khi nó đã làm xong tất cả.
* **-R** : Áp dụng luôn vào các file/folder nằm bên trong folder được phân quyền (chỉ áp dụng cho thư mục)

Ví dụ: Bạn muốn phân quyền cho file test1

# chmod 777 test1 <Cấp quyền truy cập đầy đủ cho mọi đối tượng người dùng.>

# chmod 775 test1 <Cấp quyền truy cập đầy đủ cho chủ hệ thống và nhóm quản trị, đối tượng người dùng chỉ có quyền đọc (read) và chạy (execute) file.>

# chmod 755 test1 <Cấp quyền truy cập đầy đủ cho chủ hệ thống, chỉ cho phép nhóm quản trị và đối tượng người dùng đọc và chạy các file trong thư mục.>

# chmod 700 test1 <Chỉ cấp quyền truy cập đầy đủ cho chủ hệ thống và chặn truy cập với mọi đối tượng khác.>

# chmod 500 test1 <Không cho phép nhóm quản trị và người dùng truy cập vào file trong thư mục, đồng thời giới hạn quyền chủ hệ thống chỉ đọc và chạy để tránh xóa và thay đổi các file trong thư mục này.>

# chmod 660 test1 <Cho phép chủ hệ thống và nhóm quản trị đọc, sửa, xóa và ghi dữ liệu vào file, nhưng không phân quyền truy cập cho những người dùng khác.>

**c.Thay đổi chủ sở hữu cho tập tin và thư mục**

Mặc định  tập tin và thư mục đều có user group riêng của nó, nếu bạn muốn thay đổi bạn sẽ dùng lệnh chown

# chown  <tùy chọn> <tên user>:<tên group>  <tên tập tin/thư mục>

Các tùy chọn

* **-v** : Hiển thị báo cáo sau khi chạy lệnh, mỗi lần đổi chủ sở hữu là hiển thị một lần
* **-c** : Giống như trên, nhưng chỉ hiện khi nó đã làm xong tất cả.
* **-R** : Áp dụng luôn vào các file/folder nằm bên trong folder được đổi chủ (chỉ áp dụng cho thư mục)

Ví dụ: Bạn muốn thay đổi chủ sở hữu cho file test1

# chown paste:paste test1   <Thay đổi cả user và group cho test1>

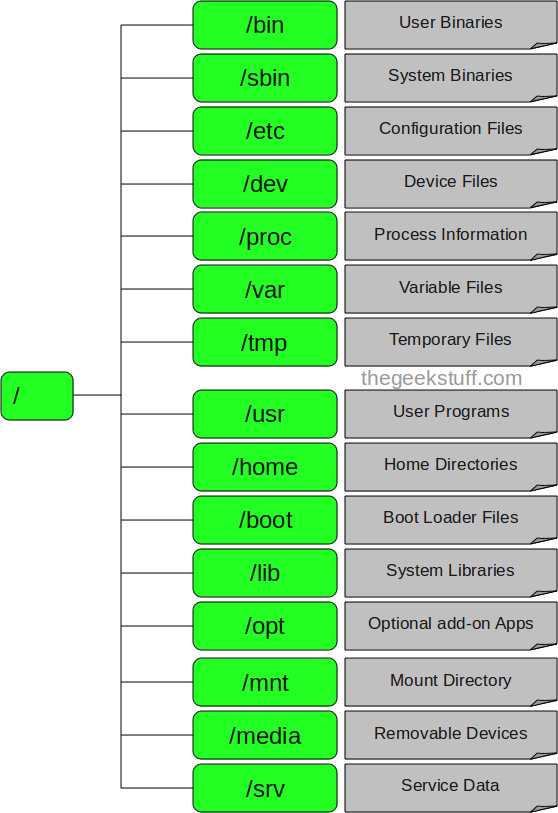
# chown paste test1         <Thay đổi user cho test1>

# chown :paste test1        <Thay đổi group cho test1>

**4.Giải thích các flag: “-R” , “-rf” , “-r” trong câu lệnh Linux**

|  |  |
| --- | --- |
| Option | Ý nghĩa |
| -R | Liệt kê luôn các thư mục con bên trong |
| -r | Liệt kê theo thứ tự đảo ngược |
| -rf | Là kết hợp của –r và –f (-f: disable –lst, enable –aU, không sort ) |

**5.Tìm hiểu về cấu trúc cây thư mục trong Linux ( thư mục /var , /bin , /home chứa gì?)**



**1. / – [Root](https://gocit.vn/bai-viet/tag/root/" \o "Posts tagged with Root)**

Đúng với tên gọi của mình: nút gốc (root) đây là nơi bắt đầu của tất cả các file và thư mục. Chỉ có root user mới có quyền ghi trong thư mục này. Chú ý rằng /root là thư mục home của root user chứ không phải là /.

**2. /bin – Chương trình của người dùng**

Thư mục này chứa các chương trình thực thi. Các chương trình chung của Linux được sử dụng bởi tất cả người dùng được lưu ở đây. Ví dụ như: ps, ls, ping…

**3. /sbin – Chương trình hệ thống**

Cũng giống như /bin, /sbinn cũng chứa các chương trình thực thi, nhưng chúng là những chương trình của admin, dành cho việc bảo trì hệ thống. Ví dụ như: reboot, fdisk, [iptables](https://gocit.vn/bai-viet/tag/iptables/" \o "Posts tagged with Iptables)…

**4. /etc – Các file cấu hình**

Thư mục này chứa các file cấu hình của các chương trình, đồng thời nó còn chứa các shell script dùng để khởi động hoặc tắt các chương trình khác. Ví dụ: /etc/resolv.conf, /etc/logrolate.conf

**5. /dev – Các file thiết bị**

Các phân vùng ổ cứng, thiết bị ngoại vi như USB, ổ đĩa cắm ngoài, hay bất cứ thiết bị nào gắn kèm vào hệ thống đều được lưu ở đây. Ví dụ: /dev/sdb1 là tên của USB bạn vừa cắm vào máy, để mở được USB này bạn cần sử dụng lệnh [mount](https://gocit.vn/bai-viet/tag/mount/" \o "Posts tagged with mount) với quyền root: # [mount](https://gocit.vn/bai-viet/tag/mount/" \o "Posts tagged with mount) /dev/sdb1 /tmp

**6. /tmp – Các file tạm**

Thư mục này chứa các file tạm thời được tạo bởi hệ thống và các người dùng. Các file lưu trong thư mục này sẽ bị xóa khi hệ thống khởi động lại.

**7. /proc – Thông tin về các tiến trình**

Thông tin về các tiến trình đang chạy sẽ được lưu trong /proc dưới dạng một hệ thống file thư mục mô phỏng. Ví dụ thư mục con /proc/{pid} chứa các thông tin về tiến trình có ID là pid (pid ~ process ID). Ngoài ra đây cũng là nơi lưu thông tin về về các tài nguyên đang sử dụng của hệ thống như: /proc/version, /proc/uptime…

**8. /var – File về biến của chương trình**

Thông tin về các biến của hệ thống được lưu trong thư mục này. Như thông tin về log file: /var/log, các gói và cơ sở dữ liệu /var/lib…

**9. /usr – Chương trình của người dùng**

Chứa các thư viện, file thực thi, tài liệu hướng dẫn và mã nguồn cho chương trình chạy ở level 2 của hệ thống. Trong đó

* /usr/bin chứa các file thực thi của người dùng như: at, awk, cc, less… Nếu bạn không tìm thấy chúng trong /bin hãy tìm trong /usr/bin
* /usr/sbin chứa các file thực thi của hệ thống dưới quyền của admin như: atd, cron, sshd… Nếu bạn không tìm thấy chúng trong /sbin thì hãy tìm trong thư mục này.
* /usr/lib chứa các thư viện cho các chương trình trong /usr/bin và /usr/sbin
* /usr/local chứa các chương tình của người dùng được cài từ mã nguồn. Ví dụ như bạn cài [apache](https://gocit.vn/bai-viet/tag/apache/) từ mã nguồn, nó sẽ được lưu dưới /usr/local/[apache](https://gocit.vn/bai-viet/tag/apache/)2

**10. /home – Thư mục người của dùng**

Thư mục này chứa tất cả các file cá nhân của từng người dùng. Ví dụ: /home/john, /home/marie

**11. /boot – Các file khởi động**

Tất cả các file yêu cầu khi khởi động như initrd, vmlinux. grub được lưu tại đây. Ví dụ vmlixuz-2.6.32-24-generic

**12. /lib – Thư viện hệ thống**

Chứa cá thư viện hỗ trợ cho các file thực thi trong /bin và /sbin. Các thư viện này thường có tên bắt đầu bằng ld\* hoặc lib\*.so.\*. Ví dụ như ld-2.11.1.so hay libncurses.so.5.7

**13. /opt – Các ứng dụng phụ tùy chọn**

Tên thư mục này nghĩa là optional (tùy chọn), nó chứa các ứng dụng thêm vào từ các nhà cung cấp độc lập khác. Các ứng dụng này có thể được cài ở /opt hoặc một thư mục con của /opt

**14. /mnt – Thư mục để mount**

Đây là thư mục tạm để mount các file hệ thống. Ví dụ như # mount /dev/sda2 /mnt

**15. /media – Các thiết bị gắn có thể gỡ bỏ**

Thư mục tạm này chứa các thiết bị như CdRom /media/cdrom. floppy /media/floopy hay các phân vùng đĩa cứng /media/Data (hiểu như là ổ D:/Data trong [Windows](https://gocit.vn/bai-viet/tag/windows/))

**16. /srv – Dữ liệu của các dịch vụ khác**

Chứa dữ liệu liên quan đến các dịch vụ máy chủ như /srv/svs, chứa các dữ liệu liên quan đến CVS.

**ND ôn tập buổi 2**

**1.Quản lý phiên bản là gì? Quản lý phiên bản cục bộ, tập trung , phân tán là như thế nào?**

**1.1 Quản lý phiên bản là gì?**

Hệ thống quản lý phiên bản - Version Control System (VCS) là một hệ thống ghi nhận và lưu lại sự thay đổi của các file theo thời gian, từ hệ thống đó một file có thể phục hồi quay về trạng thái (phiên bản) ở một thời điểm trước đó. Ngoài ra bạn có thể theo dõi sự thay đổi của một file theo thời gian, ai đã thay đổi, thay đổi vào lúc nào .... Có nhiều hệ thống VCS mà bạn có thể chọn sử dụng như: Concurrent Versions System, Subversion, Git, Mercurial

**1.2 Quản lý phiên bản cục bộ, tập chung, phân tán là như thế nào?**

Để giúp bạn thực hiện các điều trên dễ dàng hơn, bạn nên sử dụng một hệ quản trị phiên bản (VCS) chạy trên máy, lúc này VCS sẽ tạo ra một **Cơ sở dũ liệu** lưu giữ những thay đổi của các file và giúp bạn phục hồi về một phiên bản cũ một cách dễ dàng. Hệ thống đó bạn có thể gọi là **VCS cục bộ**, mô hình hóa hoạt động như hình

Khi làm việc nhóm các developer thiết lập một hệ **thống quản lý phiên bản trung tâm (CVCS)** ở một máy chủ, ở đó chứa các phiên bản của file, các lập trình viên kết nối và lấy về sửa đổi rồi lại cập nhật lại ... Tuy nhiên CVCS vẫn có nhược điểm như Server bị dừng thì không thể kết nối để các thành viên làm việc, Database ở Server bị lỗi (ví dụ do đĩa) thì khó phục hồi file đã mất ...

 VCS trung tâm

Khắc phục nhược điểm của CVCS thì một loại VCS nữa được đề xuất đó là DVCS (Distributed Version Control Systems ) - **Hệ thống quản lý phiên bản phân tán**, nó giống với CVCS là có một máy chủ ở đó có Database lưu giữ các phiên bản của file, tuy nhiên khác biệt đó là các máy khách (các developer) kết nối vào thỉ nó không chỉ lấy file mà nó lấy luôn cả hệ thống Database. Điều này có nghĩa là khi Server bị ngắt, các máy khách vẫn làm việc bình thường trên Database ở máy trạm, sau đó commit (chuyển) lên Server sau, hoặc Database ở Server bị lỗi thì máy khách bất kỳ đều cỏ thể phục hồi lại cho Server

**2.Git là gì? Lợi ích , ý nghĩa của việc sử dụng Git ? Các câu lệnh cơ bản trong Git.**

**Git** chính là hệ thống quản lý phiên bản phân tán (DVCS),

Lợi ích khi sử dụng Git:

* Sắp xếp công việc tốt hơn: tập trung giải quyết từng task mà không phải bận tâm những task liên quan.
* Linh hoạt hơn khi phải làm cùng lúc nhiều task: có thể cấu trúc công việc dễ dàng hơn.
* Tự tin hơn khi thử nghiệm những ý tưởng mới: có thể tách biệt việc thử nghiệm với dự án chính. Giúp nâng cao chất lượng code cũng như tính sáng tạo.
* Tốc độ, đơn giản, phân tán, phù hợp với dự án lớn nhỏ.

Các câu lệnh cơ bản trong Git :

* 1 **git init** khởi tạo một **Repository** mới
* 2 **git status** hiện thị thông tin trạng thái của Repository, **git status -s** hiện thị thông tin ngắn gọn hơn
* 3 **git add** cập nhật nội dung thay đổi của các file, file mới ... vào index, đưa chúng vào **staged** để chuẩn bị cho việc **commit** tiếp theo, một số cách sử dụng tham số với lệnh này như:
  + **git add filename** thêm file có tên chỉ ra trong tham số
  + **git add \*.c** tất cả các file có phần mở rộng .c
  + **git add -A** thêm mọi thứ có sự thay đổi (file mới, xóa file, nội dung thay đổi ...)
  + **git add .** thêm mọi thứ trừ loại xóa file
  + **git add -u** thêm mọi thứ trừ file mới
* 4 **git commit**
  + **git commit -m "Thông báo ..."** thực hiện commit, lưu snapshot vào DB Git
  + **git commit --amend -m "Thông báo ..."** commit, nhưng cập nhật thông tin vào commit cuối
* 5 **git clone path** sao chép một Repository có địa chỉ là **path** (có thể là địa mạng như https://github.com/ichte/xtlab.git hoặc SSH git@github.com:ichte/xtlab.git, có thể là ở chính máy của bạn như C:\Path\projectcode)
* 6 **git log** xem lại lịch sử commit. Nếu muốn hiện thị thông tin 4 commit cuối thôi thì thêm tham số -4 vào **git log -4**, nếu muốn hiện thị nhưng thông tin gì thay đổi trong từng commit thêm tham số **-p**, **git log --pretty=oneline** hiện thị thông tin mỗi commit trên một dòng
* 7 **git diff** xem sự thay đổ giữa nội dung ở thư mục làm việc và Staged nếu có, hoặc giữa thư mục làm việc với commit cuối. **git diff --staged** xem sự thay đổi của nội dung trong staged và commit cuối
* 8 **git rm filename** xóa file khỏi thư mục làm việc
* 9 **git reset HEAD filename** hủy bỏ việc thay đổi trạng thái của file, ví dụ hủy đưa và Staged
* 10 **git checkout -- filename** khôi phục thay đổi nội dung của filename, việc khôi phục ở trạng thái hiện tại có thể là là khôi phục từ Staged nếu có, nếu không thì khôi phục từ commit cuối. Nếu muốn khôi phục file từ một commit cũ có mã hash hash thì làm như sau
  + **git checkout [hash] filename** khôi phục lại filename
  + **git checkout [hash] .** khôi phục các file về phiên bản theo commit cũ

**3.Cách tạo , ý nghĩa và cách sử dụng các branch develop, feature , hotfix , release.**

**Cách tạo branch:**

$ git branch <tên>

**Feature**

Cách tạo: git flow feature start [tên tính năng]

Feature:dùng để phát triển tính năng bắt nguồn từ develop

**Release**

Cách tạo**:**git flow release start RELEASE [BASE]

Tùy chọn [BASE] cho phép commit được một giá trị hash sha-1. Trường hợp không chỉ định tùy chọn này thì HEAD sẽ được sử dụng.

Releach: dùng để chuần bị phát hành

Cho phép sửa các nỗi nhỏ và chuẩn bị meta-data

**Hotfix**

Cách tạo: git flow hotfix start VERSION [BASENAME]

Tham số VERSION ở đây chỉ định tên của phiên bản release hotfix này. Còn tham số tùy chọn basename sẽ chỉ định tên nhánh cơ bản thay cho hotfix thay vì nhánh 'master'

Hotfix: Sử dụng cho các trường hợp khẩn cấp cần phải sửa đổi ngay trên bản sản phẩm (nhánh master).

Có thể tạo nhánh giải quyết khẩn cấp từ tag của nhánh 'master'.

**4.Các trường hợp dẫn đến conflict? Cách xử lý conflict.**

**Các trường hợp dẫn tới xung đột(conflict):**

**\* Theo tính chất lợi hại**

- Xung đột chức năng: Là sự đối đầu giữa các phía mà sự đối đầu này ảnh hưởng tích cực đến việc thực hiện nhiệm vụ.

Nó xuất phát từ những bất đồng về năng lực. Khi có quá ít xung đột và mâu thuẫn cũng là bất lợi, vì người ta trở nên tự mãn. Khi đó sẽ có rất ít hoặc chẳng có chút sáng tạo nào.

Nhà quản trị cần phải biết phân biệt các xung đột và mâu thuẫn giữa các cá nhân, giữa các nhóm, giữa các tổ chức và ở chính cá nhân.

 - Xung đột phi chức năng: Là sự đối đầu giữa các phía mà kết cục là sẽ cản trở việc hoàn thành mục tiêu trong công việc.

Xung đột này có thể về tình cảm và liên quan đến việc không hợp nhau nhưng mang tính tàn phá. Ðây là bản chất dẫn tới nhiều khả năng thất bại khi giải quyết các xung đột này.

**\* Theo tính bộ phận**

- Xung đột giữa các bộ phận.

- Xung đột giữa các nhà quản trị và nhân viên

- Xung đột giữa các nhân viên.

- Xung đột nhóm: nguyên nhân thông thường nhất là xung đột giữa các nhóm trong tổ chức do nguồn lực khan hiếm, cần phải có thêm nguồn lực và nhu cầu này mở ra xung đột.

**Cách xử lý conflict trong Git:**

- Để giải quyết xung đột conflict trong git, rất đơn giản là chỉnh sửa nội dung trên file mà nó dẫn tới xung đột, sau đó thực hiện add, commit cho file đó.  
- Để giảm thiểu xung đột trong một nhóm có nhiều thành viên, bạn nên xác định một quy trình ngay từ đầu để tất cả các thành viên trong nhóm làm việc trên cùng một dự án được biết khi sử dụng Git. Dưới đây là một số gợi ý để làm việc hiệu quả hơn:  
+ Cam kết thường xuyên: đừng đợi cho đến khi bạn có một lượng lớn các tập lệnh được tạo để commit và push chúng vào kho lưu trữ từ xa Git. Tập lệnh nhỏ hơn được push, bạn càng dễ giải quyết xung đột.  
+ Nên pull các thay đổi từ kho lưu trữ từ xa Git trước khi làm việc trên các tập lệnh mới và trước khi commit.  
+ Mỗi thành viên nên làm việc trên từng tính năng trong các nhánh riêng biệt tại một thời điểm.

**5.Git workflow. Git successful branching model.**

\*Git workflow

Gitflow Workflow là một thiết kế dòng công việc Git được xuất bản lần đầu tiên và phổ biến lần đầu tiên và được phổ biến Vincent Driessen tại nvie. Luồng công việc Gitflow định nghĩa một mô hình phân nhánh nghiêm ngặt được thiết kế xung quanh bản phát hành dự án lớn hơn.

Gitflow phù hợp lý tưởng cho các dự án có chu kỳ phát hành theo lịch trình.Quy trình công việc này không thêm bất kỳ khái niệm hoặc lệnh mới nào ngoài những gì cần thiết cho quy trình chi nhánh tính năng. Thay vào đó , nó gán các vai trò rất cụ thể cho các nhánh khác nhau và xác định cách thức và thời điểm chúng nên tương tác .Ngoài feature các nhánh ,nó sử dụng các nhánh riêng lẻ để chuẩn bị, duy trì và ghi lại các bản phát hành. Tất nhiên , bạn cũng có thể tận dụng tất cả các lợi ích của dòng công việc tính năng , yêu cầu kéo , thử nghiệm tách biệt và cộng tác hiểu quả hơn.

Gitflow là một trong nhiều phong cách làm việc của Git mà bạn và nhóm của bạn có thể sử dụng.

Một số điều quan trọng cần biết về Gitflow là :

* + Quy trình công việc là tuyệt vời cho quy trình công việc phần mềm dựa trên phát hành.
  + Gitflow cung cấp một kênh dành riêng cho các hotfix để sản xuất.

Lưu lượng chung của Gitflow là :

1. Một develop nhánh được tạo từ master
2. Một release nhánh được tạo từ develop
3. Feature các nhánh được tạo ra từ develop
4. Khi feature hoàn thành, nó được sáp nhập vào develop nhánh
5. Khi release chi nhánh được thực hiện , nó được hợp nhất vào develop và master
6. Nếu một vấn đề trong master được phát hiện , một hotfix nhánh được tạo từ master
7. Khi hotfix hoàn thành , nó được hợp nhất với cả hai develop và master

\*Git successful branching model

Git Flow là tên gọi của 1 tool (command) hỗ trợ branch model gọi là A successful Git branching model do Vincent Driessen đề xuất ra. Người ta thường gọi đó là model hay tool, và trong git-flow có 5 kiểu với mỗi vai trò khác nhau (tùy trường hợp mà có lúc là 6 kiểu), switch các kiểu với nhau để phát triển Bằng việc set trước các branch, những rule khi merge, dù có bao nhiêu developer cùng thời điểm phát triển vẫn có thể quản lý branch dễ dàng, và tránh được những vấn đề do merge.

- Mô hình gồm 5 nhánh:

* 2 nhánh chính:
  + master
  + develop:
    - tạo từ master
    - merge về master
* 3 nhánh phụ
  + feature:
    - Quy ước đặt tên: feature-têntínhnăng hoặc feature/têntínhnăng
    - Dùng khi phát triển tính năng mới
    - Tạo từ develop
    - Merge về develop
  + release:
    - Quy ước đặt tên: release-version
    - Chức năng: chuẩn bị phát hành phiên bản mới và sửa lỗi nhỏ
    - Tạo từ develop
    - Merge về develop và master
  + hotfix:
    - hotfix-version
    - Chức năng: fix lỗi lớn, nghiêm trọng
    - Tạo từ master
    - Merge về develop và master

- Thứ tự thực hiện:

* Tạo kho chứa trên local:

**git init**

=> nhánh master đã được tự động khởi tạo ngay sau đó.

* Code, commit trên master, sau đó gắn tag v1.0.0

**git tag v1.0.0**

* Tạo nhánh develop
* Tạo nhánh feature/\* từ develop để code và commit thêm 1 tính năng nào đó.
* Lần lượt chuyển sang các nhánh master và develop để merge nhánh feature

**git merge feature/\* develop**

* Tạo nhánh release/v1.1.0 từ develop để chuẩn bị phát hành phiên bản mới, tiếp tục code commit và merge. Sau khi merge về master thì gắn tag v1.1.0

**git merge release/v1.1.0 develop**

**git merge release/v1.1.0 master**

**git tag v1.1.0**

* Tạo nhánh hotfix/v1.1.1 từ master để fix các lỗi lớn, sau đó merge về nhánh chính, sau khi merge về master thì gắn tag v1.1.1

**git merge hotfix/v1.1.1 develop**

**git merge hotfix/v1.1.1 master**

**git tag v1.1.1**

- Chú ý:

Nếu cần tạo Merge Request trên sever như github, gitlab mà không tạo merge ở local như trên, thì cần phải pull nhánh develop, sau đó mới được push các nhánh cần merge.

*Ví dụ*:

Muốn tạo Merge Request nhánh feature/login sau khi đã hoàn thiện code, commit xong

* **git pull origin develop**
* fix conflict nếu có, sau đó commit lại.
* **git push origin feature/login develop**
* tạo merge request trên sever và đợi review để được merge