CÂU LẠC BỘ AN TOÀN THÔNG TIN PTIT

ISP CLUB

****

BÁO CÁO BÀI TẬP

WEEK 3: Phân quyền

Giảng viên hướng dẫn: **Mùa hè zui zẻ**

Tên sinh viên: **Nguyễn Anh Vũ**

Mã sinh viên: **B21DCAT224**

HÀ NỘI – 2022

MỤC LỤC

LÝ THUYẾT

1. Khái niệm phân quyền:

- Các file permissions, attributes, và ownership quyết định **mức độ truy cập** vào các **files/folders** của các **tiến trình hệ thống** và **người dùng**
- Có **3** nhóm user:

  + Owner (u): chủ sở hữu

  + Group (g): nhóm

  + Others (o): những người dùng khác
- Để có thể xem được **các quyền của 1 file**, dùng *ls -l <filename>*

```
-rw-r--r-- 12 linuxize users 12.0K Apr  28 10:10 file_name
|[-][-][-]-   [------] [---]
| |  |  | |      |       |
| |  |  | |      |       +-----------> 7. Group
| |  |  | |      +-------------------> 6. Owner
| |  |  | +--------------------------> 5. Alternate Access Method
| |  |  +----------------------------> 4. Others Permissions
| |  +-------------------------------> 3. Group Permissions
| +----------------------------------> 2. Owner Permissions
+------------------------------------> 1. File Type
```

❗ Giải thích:

(1) Loại file: có các loại file sau:

+ -: file bình thường (png, txt, rar, etc.) – chứa **dữ liệu, văn bản** (kể cả bin), **ảnh**…

+ d: **thư mục** – chứa các file khác

+ b: block files – chứa thông tin về các **thiết bị khối** (là các thiết bị thực hiện các phép toán dữ liệu input và output dưới dạng khối); là **hardware file**

+ c: character files: là **hardware file** đọc/ghi dữ liệu **từng chữ cái một** trong một file (VD: terminal, serial port, etc.)

+ p: pipe files – **FIFO**

+ l: symbolic link (symlink – tương tự shortcuts trên Windows) – một file **trỏ** tới **file/folder khác**

+ s: **socket** – truyền dữ liệu giữa các app và kích hoạt "giao tiếp" giữa 2 tiến trình

(2) Quyền của owner: bộ 3 đầu

(3) Quyền của group: bộ 3 giữa

(4) Quyền của others: bộ 3 cuối

(5) Alternate access method

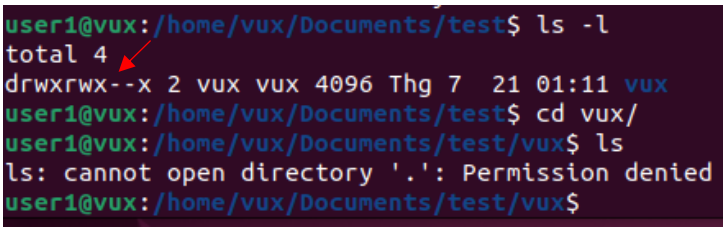
(6) Chủ sở hữu – owner

(7) Nhóm – group

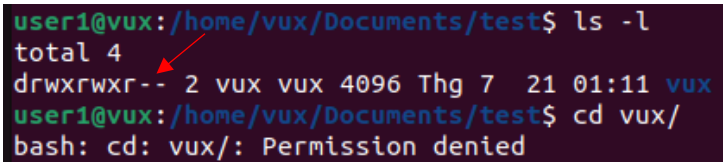
- Tác dụng của cờ quyền lên file và folder:

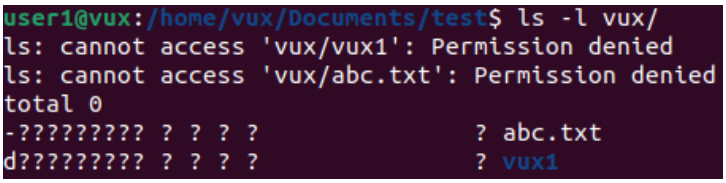
<table>
<tr><th>Các loại cờ</th><th>File</th><th>Directory (Folder)</th></tr>
<tr><td>r</td><td>Xem được nội dung file</td><td>Xem được nội dung folder
(dùng lệnh ls)</td></tr>
<tr><td>w</td><td>Có thể change/modify file</td><td>Có thể thay đổi nội dung folder
(Thêm file, xóa file, etc.)</td></tr>
<tr><td>x</td><td>Có thể thực thi file</td><td>Có thể thay đổi folder
(dùng cd)</td></tr>
<tr><td rowspan="2">s</td><td colspan="2">- Thấy ở trong quyền của owner (setuid) hoặc group (setgid) và thay thế vị trí của x
- Nếu owner/group không có quyền x, s sẽ được thay thành S</td></tr>
<tr><td>- Nếu có cờ này, người dùng hiện tại có thể thực thi file dưới quyền của owner hoặc group</td><td>- Nếu có cờ này (ở quyền của group), người dùng hiện tại khi tạo file mới trong này thì nó kế thừa group id của folder thay vì group id của người dùng hiện tại</td></tr>
<tr><td rowspan="2">t</td><td colspan="2">- Thấy ở trong quyền của other (sticky) và thay thế vị trí của x
- Nếu other không có quyền x, t sẽ được thay thành T</td></tr>
<tr><td>- Không có tác dụng với file</td><td>- Nếu có cờ này, chỉ có file's owner, directory's owner, hoặc người dùng quản trị (root và tương đương) mới có thể xóa hoặc đổi tên file trong nó</td></tr>
</table>

VD:

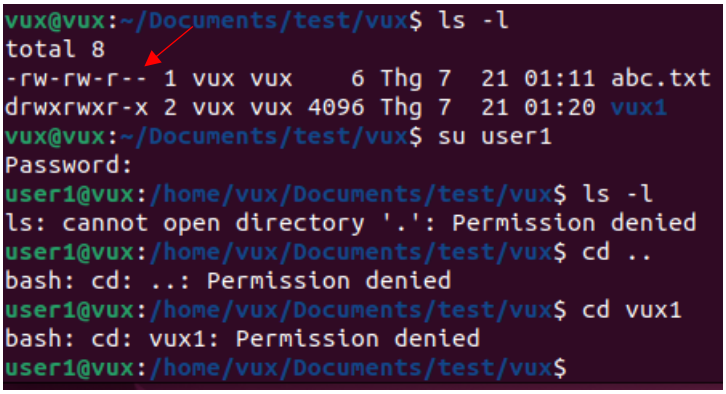


- Folder "vux" không có cờ "r" cho o, user1 vẫn **có thể cd** vào "vux" nhưng **không** thể **xem được** bên trong có gì bằng lệnh *ls*





- Folder "vux" không có cờ "x" cho o, nên user1 **không thể cd** vào "vux" nếu đang không ở bên trong "vux1", nhưng vẫn **có thể ls** để xem nội dung

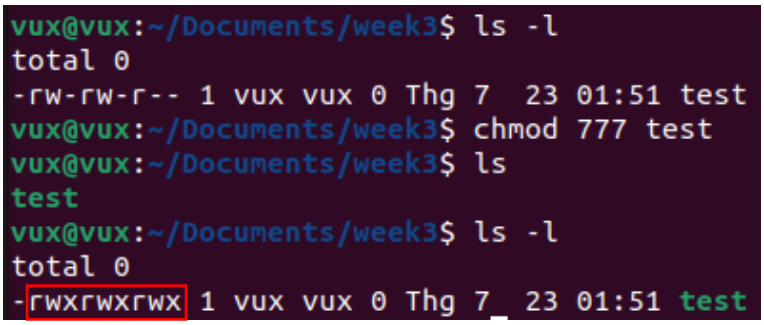


- Folder "vux" vẫn không có cờ "x" cho o, nên dù user 1 có đang ở trong "vux" cũng **không thể cd** ra ngoài hoặc vào trong thư mục khác trong "vux" và cũng **không thể ls** để xem nội dung

- Có thể **đổi quyền** bằng lệnh *chmod* và chỉ có **root**, **owner**, hoặc người dùng có thể dùng ***sudo*** mới có thể thay đổi quyền

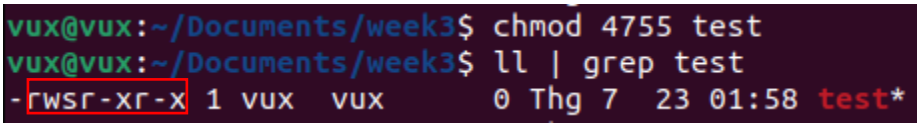
  + Phân quyền bằng **số**:

- Là một số gồm 3 hoặc 4 chữ số, mỗi chữ số sẽ đại diện cho các quyền của mỗi nhóm user và là tổng của các quyền dưới dạng số cộng lại
- Nếu là **3** chữ số thì **mỗi chữ số** từ trái sang phải đại diện cho quyền của u-g-o
- r = 4, w = 2, x = 1
- VD:



(777 là full quyền cho tất cả các nhóm)

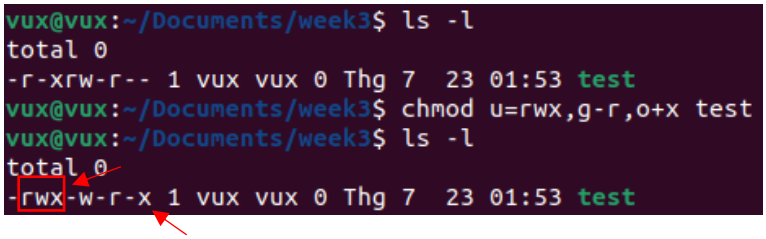
- Nếu là **4** chữ số thì **chữ số đầu tiên** đại diện cho các **quyền đặc biệt**, ba số cuối giống trên
- setuid = 4, setgid = 2, sticky = 1
- VD:



(4755: setuid, full quyền cho u, r và x cho g và o)

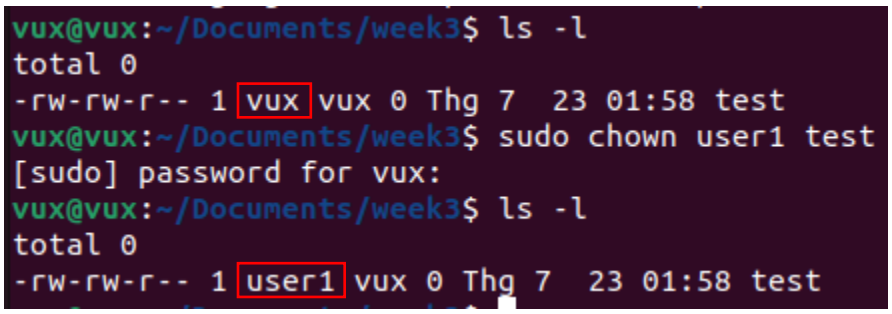
+ Phân quyền bằng **chữ**:

- Sử dụng các ký tự u-g-o-a (all) cùng các toán tử "+" – gán thêm cờ, "-" – bỏ cờ, "=" – thay đổi cờ thành cờ khác
- VD:



(u=rwx: set rwx cho u; g-r: bỏ r ở group; o+x: thêm x cho others)

- Có thể **đổi chủ sở hữu** file bằng *chown* (cú pháp *chown <user hoặc uid> <filename>*) hoặc **đổi nhóm** bằng *chgrp* (cú pháp *chgrp <group hoặc gid> <filename>*) hoặc *chgrp <user>:<group> <filename>* để **đổi cả owner và group** VD:



2. UID, RUID, EUID, OWN và MOD:

- *Privileged processes*: tiến trình có euid là 0
- *uid*: user id – id của user. Mỗi user khi **được tạo** sẽ có một id **riêng**
- *ruid*: real user id – id của user **bắt đầu process**
- *euid*: effective user id – id mà hệ thống thấy khi quyết định một process **có những quyền gì**. Hầu hết mọi trường hợp, euid giống ruid, nhưng khi một file có **setuid bit** ở owner thì sẽ khác: khi đó euid sẽ là **owner** của file
- Để xem các uid của tiến trình*: ps -o pid,euid,ruid,suid,egid,rgid,sgid,cmd*

```
vux@vux:/etc$ ps -o pid,euid,ruid,suid,egid,rgid,sgid,cmd
    PID  EUID  RUID  SUID  EGID  RGID  SGID CMD
   4385  1000  1000  1000  1000  1000  1000 bash
   4611  1000  1000  1000  1000  1000  1000 bash
   5150  1000  1000  1000  1000  1000  1000 ps -o pid,euid,ruid,suid,egid,rgid,s
```

- So sánh giữa *setuid(uid_t uid)* và *seteuid(uid_t uid)*:

  + Giống: cùng để set euid cho tiến trình

  + Khác: set_uid có thêm 1 chi tiết nữa là nếu tiến trình đó có quyền CAP_SETUID (cái này em không rõ :<) thì set luôn cả cho ruid và saved uid

- **own và mod**

THỰC HÀNH

1. Kiến thức cơ bản:

- Include thư viện cho C (GNU C Library):

  + <*sys/types.h*>

  + <*unistd.h*>

  + <*pwd.h*>: chứa **cấu trúc** *passwd* được defined như sau (các biến tương ứng với các trường trong /etc/passwd)

```
struct passwd {
     char   *pw_name;       /* username */
     char   *pw_passwd;     /* user password */
     uid_t  *pw_uid;        /* user ID */
     gid_t  *pw_gid;        /* group ID */
     char   *pw_gecos;      /* user information */
     char   *pw_dir;        /* home directory */
     char   *pw_shell;      /* shell program */
};
```

- Các data types:

  + pid_t: kiểu dữ liệu cho **pid**

  + uid_t: kiểu dữ liệu cho **uid**

  + gid_t: kiểu dữ liệu cho **gid**

- Các hàm:

  + getuid(): **trả về ruid** của process

  + geteuid(): **trả về euid** của process

  + setuid(*uid_t* <**uid**>): **set ruid** của process thành <**uid**>

  + seteuid(*uid_t* <**uid**>): **set euid** của process thành <**uid**>

  + getpid(void): trả về **pid** process

  + getppid(void): trả về **pid** của **parent** process

  + getpwnam(username): trả về **con trỏ** tới cấu trúc trong password database (Linux thì là /etc/passwd) mà **trùng với username**

- Để xem kích thước của pid_t, uid_t, gid_t: dùng *sizeof()* và đối số truyền vào là *%zu* (thực ra là *%d* cũng không sai nhưng chưa phải chính xác nhất và *%zu* dùng để in các kiểu dữ liệu *size_t*)

THỰC HÀNH

Bài 1: Set password cho 1 user bất kỳ bằng quyền người dùng thường:

```cpp
#include <iostream>
#include <pwd.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>

using namespace std;

int main(){
    // cout << getuid() << endl;
    // cout << geteuid() << endl;
    passwd *p_entry;
    char username[100];
    cout << "Enter username to change pass: ";
    cin >> username;
    p_entry=getpwnam(username);
    setreuid(p_entry->pw_uid, p_entry->pw_uid);
    // cout << getuid() << endl;
    // cout << geteuid() << endl;
    char command[100]="passwd ";
    strcat(command, username);
    // cout << command << endl;
    system(command);
}
```

Giải thích:

- Tạo con trỏ cấu trúc *passwd* (trong thư viện *pwd.h*) *p_entry* để sau trỏ tới địa chỉ có giá trị là tên *username*
- Sau khi nhập tên user, gán giá trị *p_entry* bằng hàm *getpwnam*
- Sau đó đổi **ruid** và **euid** của tiến trình bằng **uid của username**
- Sau khi compile file code, cần đổi owner và group sang **root** và thêm cờ **setuid**

```
vux@vux:~/Documents/week3$ ll | grep b1
-rwsrwxr-x 1 root  root 16936 Thg 7  24 20:31 b1*
-rw-rw-r-- 1 vux   vux    576 Thg 7  24 20:31 b1.cpp
```

Kết quả:

```
vux@vux:~/Documents/week3$ ./b1
Enter username to change pass: vux
Changing password for vux.
Current password:
New password:
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: password updated successfully
```

Bài 2: Có 2 user thường, chạy tiến trình bằng quyền user 1 nhưng thực hiện lệnh id thì in ra thông tin user 2:

```
1   #include <iostream>
2   #include <pwd.h>
3   #include <unistd.h>
4   #include <string.h>
5   #include <sys/types.h>
6
7   using namespace std;
8
9   int main(){
10      // cout << getuid() << endl;
11      // cout << geteuid() << endl;
12      struct passwd *p_entry;
13      char username[100];
14      cout << "Enter username to check id: ";
15      cin >> username;
16      p_entry=getpwnam(username);
17      setreuid(p_entry->pw_uid,p_entry->pw_uid);
18      // cout << getuid() << endl;
19      // cout << geteuid() << endl;
20      system("id");
21      int a; cin >> a;
22  }
```

Giải thích:

- Từ đầu tới dòng 17 giống bài 1
- Gọi trực tiếp hàm *id* bằng *system()* ra để lấy kết quả
- Dòng 21 có tác dụng để tiến trình không bị **terminate** (em không hiểu sao không dùng được system(“pause”) :<)
- Sau khi compile file code, cần đổi owner và group sang **root** và thêm cờ **setuid**

Kết quả:

```
vux@vux:~/Documents/week3$ ./b2
Enter username to check id: user1
uid=1001(user1) gid=1001(user1) groups=1001(user1),27(sudo)
vux@vux:~/Documents/week3$ ./b2
Enter username to check id: user2
uid=1002(user2) gid=1002(user2) groups=1002(user2)
vux@vux:~/Documents/week3$ ./b2
Enter username to check id: vux
uid=1000(vux) gid=1000(vux) groups=1000(vux),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),122(lpadmin),134(lxd),135(sambashare)
```

```
vux@vux:~/Documents/week3$ ps -e | grep b2
  13836 pts/0    00:00:00 b2
```

```
vux@vux:~/Documents/week3$ ps -eo pid,uid,euid | grep 13836
  13836  1002  1002
```

Có thể thấy tuy chạy ./b2 bằng user *vux* nhưng khi **kiểm tra** ruid và euid của tiến trình thì lại là uid của user *user2*

TÀI LIỆU THAM KHẢO

- Phân quyền:

  + *https://linuxize.com/post/understanding-linux-file-permissions/* (tiếng Anh, khá đầy đủ, dễ hiểu nếu hiểu tiếng Anh)

  + *https://www.geeksforgeeks.org/how-to-find-out-file-types-in-linux/* (các loại file, tiếng Anh)

  + *https://viblo.asia/p/phan-quyen-trong-linux-yMnKMbDNZ7P* (tiếng Việt)

- RUID, EUID, SUID, UID:

  + *https://book.hacktricks.xyz/linux-hardening/privilege-escalation/euid-ruid-suid#:~:text=*uid,a%20case%20where%20they%20differ*.

  + *https://unix.stackexchange.com/questions/191940/difference-between-owner-root-and-ruid-euid*