Red Hat System Administration II

Course Outlines

- User Administration
- Standard Permissions
- Advanced Permissions
- Access Control Lists

User Administration

User and Group account databases

- The `/etc/passwd` file

```
username:x:uid:gid:comment:home-directory:login-shell
```

- The `/etc/shadow` file

```
username:encrypted passwd:last
  changed:min:max:warn:??:expire:future-use
```

- The `/etc/group` file

```
groupname:x:gid:comma-separated list of group members
```

The `/etc/gshadow` file ???

Adding a new user account

```
# useradd username
```

- The `useradd` command populates user home directories from the `/etc/skel` directory.
- To view and modify default setting -> `useradd -D`

```
# passwd username
```

Adding multiple user accounts

```
# newusers filename
```

Modifying user accounts

- To change a user's account information, you can:
  - Edit the `/etc/passwd` or `/etc/shadow` files manually
  - Use the `chage` command discussed later
  - Use the `usermod` command:

```
usermod [options] username
```

- `Useful options`
  - To changes the login name use `-l <login name>`
  - To lock the password use `-L`
  - To unlock the password use `-U`

Deleting a user account

- To delete a user account you can
  - Manually remove the user from
    - `/etc/passwd` file
    - `/etc/shadow` file
    - `/etc/group` file
    - remove the user's home directory (`/home/username`)
    - and mail spool file (`/var/spool/mail/username`)
  - Use the `userdel` command.

```
# userdel [-r] username
```

Password Ageing Policies

- The `chage` command sets up password aging

```
# chage [options] username
```

- Options
  - `-m`: to change the min number of days between password changes
  - `-M`: to change the max number of days between password changes
  - `-I`: ???
  - `-E date`: change the expiration date for the account
  - `-W`: change the number of days to start warning before a password change will be required

Describe the user private group scheme

- A traditional problem found in many UNIX/Linux environments is when administrators place all users in the same primary group. When users on such systems use a umask value of 002.

- Red Hat Enterprise Linux solves this problem by assigning user a primary group for which they are the sole members.

- This "private" primary group has the same name as the user's username

fppt.com

Managing Groups

- Creating New Group

```
# groupadd groupname
```

- Modifying an Existing Group

```
# groupmod [options]groupname
```

- Deleting a Certain Group

```
# groupdel groupname
```

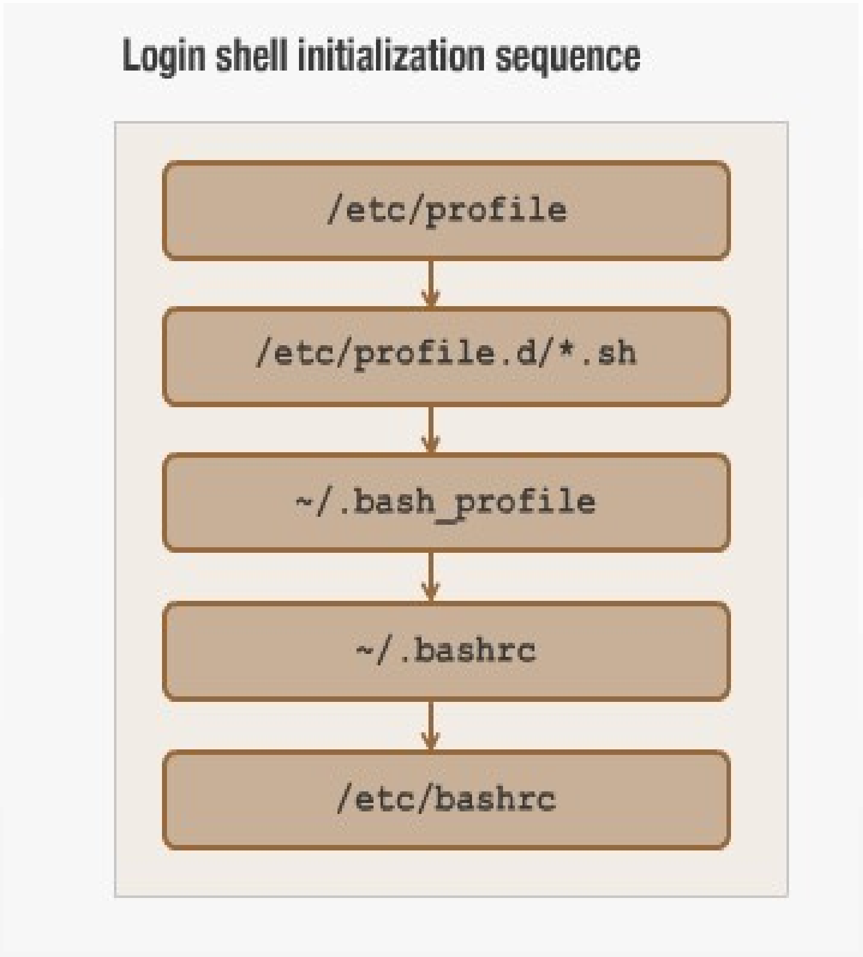
- List all file which are owned by groups not defined in `/etc/group` file

```
# find / -nogroup
```

Managing Groups cont'd

- You can use the `gpasswd` command to define
  - Group members
  - Group administrators
  - And to create or change group passwords
- Use the `-r` option to the `groupadd` command avoids using a GID within the range typically assigned to users and their private groups.

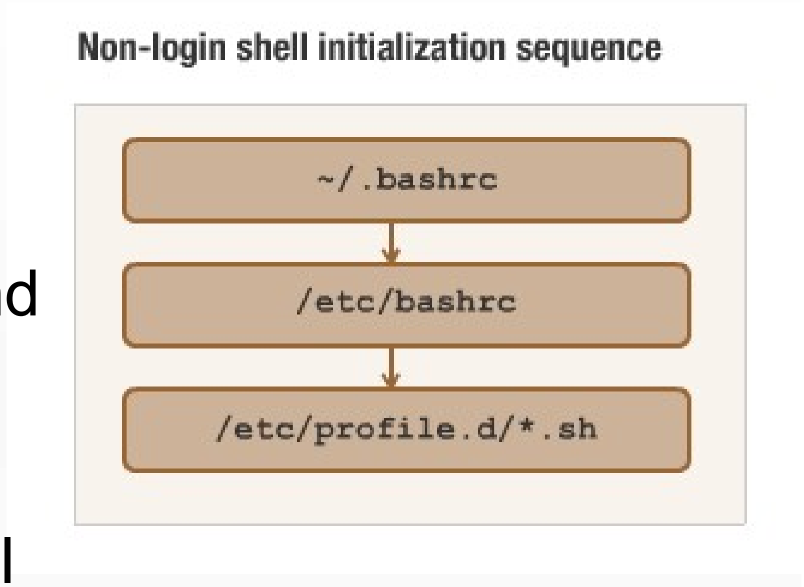
Login Shells



Non-Login Shells

- A user can open a new shell through
  - Starting a terminal
  - Executing the `bash` command

    Or
  - executing a command that naturally opens a new shell (like the `newgrp` command)



Switching Accounts

```
# su [-] [username]
# su [-] [username] -c command
```

Using `sudo` command

- `sudo` is more secure
- `sudo` access is controlled by the `/etc/sudoers`.
  - This file is edited by `visudo`, an editor and syntax checker.
  - To give a specific group of users limited root privileges

```
User_Alias   LIMITEDTRUST=st1,st2
Cmnd_Alias MINIMUM=/etc/rc.d/init.d/httpd
Cmnd_Alias SHELLS=/bin/sh,/bin/bash
LIMITEDTRUST ALL=MININUM
user5 ALL=ALL,!SHELLS
%development station1=ALL, !SHELLS
```

Standard Permissions

File Ownership and Permissions

- Every file and directory has both user and group ownership. A newly-created file will be owned by:
  - The user who creates it
  - That user's primary group (unless the file is created in a set group ID (SGID) directory; more on this file in the next lesson)

- File ownership can be changed using `chown` command.

- Example

```
# chown user1 file1
# chown user1:group1 file1
# chown :group1 file1
```

File Permission

- Access level
  - User (u)
  - Group (g)
  - Others (o)
- Access mode
  - Read (r)
  - Write (w)
  - Execute (x)
- Example

```
# chmod u=rw file1
# chmod g+x file1
# chmod o-w file1
# chmod u+x,go-w file1
# chmod 742 file1
```

Advanced Permissions

Advanced Permissions

- SUID on an executable file
- SGID on an executable file
- SGID on a directory
- Sticky bit on a directory
- Example

```
# chmod u+s executable1
# ls -l
    -rwsr-xr-x ...
# chmod g+s executable1
# ls -l
    -rwxr-sr-x ...
# chmod g+s directory1
  # ls -l
      drwxr-sr-x ...
         # chmod o+t directory1
         # ls -l
             drwxrwxrwt ...
```

Default Permissions

- The `umask` command sets the default permissions for files and directories

- Example

```
# umask 002
# umask
    022
```

Access Control Lists

Access Control Lists

- Use ACLs to generate customized permission sets
- To enable ACLs on a filesystem, the filesystem must be mounted with the acl option.
- Example

```
# mount -o remount,acl /home
# getfacl /home/sherine/fname
….
u::rwx                  # applies to owner
u:ahmed:rw-             # applies to user ahmed
u:3142:---              # applies to user id 3142

     g::rw-       applies to file group
     g:telecom:rwx applies to telecom group
       g:10:rw-       applies to group id 10

       O:rwx        applies to everyone else
```

Access Control Lists Cont'd

```
# setfacl -m u:user1:rw file1      # grants rw to user1
# setfacl -m g:group1:rx directory1 # grants rx to group group1
# setfacl -x u:user1 file1        # remove user1 acl entry
```

- Permission Precedence

1. if owner, then the file's owner permission apply
2. else if user has an acl entry, then the user acl applies as long as it is permitted by **mask**
3. else if matches file's group or acl group entry then it applies
4. otherwise the file's other permissions apply.

Access Control Lists Cont'd

- The ACL mask
  - Mask limits the maximum permissions that both the group that owns the file and that supplementary users and groups in ACLs can have.
  - getfacl command displays the current mask **mask::perm**

*** changing group permissions on a file with an ACL by using chmod, it actually changes the mask, which limits the max. permissions of all groups and supplementary users**

Access Control Lists Cont'd

- Default ACLs (Inheritance)
  - A directory can have a default ACL entries that are automatically set on new files created in that directory

```
# setfacl -m d:u:ahmed:rw directory
# setfacl -m d:u:ahmed:rx directory
```

Thanks ☺

SBAHADER@GMAIL.COM