Operating Systems (Client)

Lecture 7

User Account Management

Dr. Kevin Farrell

Lecture Overview

Introduction

- Adding and removing users is a bread-and-butter administrative skill
- Administrators need a thorough understanding of the Linux account system in order to manage network services and configure accounts appropriately for the local computing environment.
- Account hygiene is also a key determinant of system security. Infrequently used accounts, and accounts with easily guessed passwords are prime targets for hackers
- There are a number of automated tools to add and remove users, but it's important to understand the underlying changes, which those tools are making.
- In this Lecture, we'll first examine the underlying model that the automated tools implement, then briefly describe the tools themselves (`useradd`, `userdel`, etc.)

The /etc/passwd File: Definition

- This file contains a list of users recognised by the system
- The system consults the file at login time to determine a user’s UID and to verify the user’s password
- Each line in the file represents one user and contains seven fields separated by colons:
  - Login name
  - Encrypted password (unless a shadow password file is used)
  - UID number
  - Default GID number
  - “GECOS” information: full name, office, extension, home phone
  - Home directory
  - Login shell
- For example, the following *two* lines are all valid /etc/passwd entries.

```
root:lga5FjuGpZ2so:0:0:The System,,x6096,:/:/bin/csh
jane:x:100:0:Jim Lane,ECT8-3,,:/staff/jane:/bin/sh
```

The /etc/passwd File: An Old File!

- As computers have become faster => increasingly dangerous to leave encrypted passwords in plain view
- Linux allows you to hide the encrypted passwords by placing them in a separate file that is *not* world-readable
- This is known as a **shadow** password mechanism, and it is the default on most distributions.
- The shadow password system makes more sense when explained as an extension of the traditional **/etc/passwd** (as it historically was), => defer discussion of shadow until later
- The contents of **/etc/passwd** are often shared among systems with a database such as Network Information System (**NIS**) or Lightweight Directory Access Protocol (**LDAP**)
- The following slides discuss the seven **/etc/passwd** fields in more detail

Login Name: Definition + Rules 1

- Login names (also known as usernames) must be unique and no more than 32 characters long. They may contain any characters except colons and newlines.
- Some older versions of UNIX limit the permissible characters to alphanumeric characters, and impose an 8-character length limit. If you use NIS, login names are limited to 8 characters regardless of the operating system.
- It’s a good idea to obey the more restrictive limits even under Linux. Such a policy will avert potential conflicts with older software and will guarantee that users can have the same login name on every machine.
- Login names are case sensitive; however, most mail systems (including `sendmail`) pay no attention to case
- To avoid any possible problems caused by mixed-case login names, use lowercase names. They are traditional and also easier to type!

Login Name: Rules 2

- Login names should be easy to remember, so random sequences of letters do not make good login names.
- Avoid “handles” and cutesy nicknames, even if your organisation is relatively informal. They can have an adverse effect on your site’s credibility
- Since login names are often used as email addresses, it’s useful to establish a standard way of forming them
- It should be possible for users to make educated guesses about each other’s login names. First names, last names, initials, or some combination of these all make reasonable naming schemes.

Login Name: Rules 3

- Any fixed scheme for choosing login names eventually results in duplicate names or names that are too long, so you will sometimes have to make exceptions.
- In the case of a long name, you can use the **/etc/mail/aliases** file to equate two versions of the name, at least as far as mail is concerned.
- For example, suppose you use an employee's first initial and last name as a paradigm. Brent Browning would therefore be "bbrowning", which is 9 characters and therefore too long. Instead, you could assign the user the login "brentb", leaving "bbrowning" as an aliases file entry:
    - bbrowning: brentb
- If your site has a global mail alias file, each new login name must be *distinct* from any alias in this file. If it is not, mail will be delivered to the alias rather than the new user.

Login Name: Rules 4

- Common for large sites to implement a full-name email addressing scheme (e.g., John.Q.Public@mysite.com) that hides login names from the outside world.
- This is a fine idea, but it doesn’t obviate any of the naming advice given above. If for no other reason than the sanity of administrators, it’s best if login names have a clear and predictable correspondence to users’ actual names.

Login Name: Rules 5

- If you have more than one machine, login names should be unique
  - a user should have the same login name on every machine. This rule is mostly for convenience, both yours and the user’s.
  - a particular login name should always refer to the same person. Some commands (e.g., ssh) can be set up to validate remote users based on their login names.
- Even if scott@boulder and scott@refuge were two different people, one might be able to log into the other’ s account without providing a password if the systems were not set up properly.

Encrypted Password: Definition

- Reminder: most systems actually keep encrypted passwords in **/etc/shadow** rather than **/etc/passwd**. But, the comments in this section apply regardless of where passwords are actually kept.
- /etc/passwd stores passwords in an encrypted form.
- This means you must either set the contents of this field by using the `passwd` command (`yppasswd` if you use NIS) or by copying an encrypted password string from another account.
- If you edit **/etc/passwd** to create a new account, put a star (*) in the encrypted pass word field.
- The star prevents unauthorized use of the account until you have set a real password.
- Never leave this field empty because this introduces a serious security hole since no password is required to access the account.

Encrypted Password: Encryption

- Most Linux distributions default to using standard **DES** encryption for passwords. This standard limits the length of unencrypted passwords to 8 characters. Longer passwords are accepted, but only the first 8 characters are significant. Extra characters are silently ignored.
- An alternative password encryption system is based on the **MD5** secure hashing algorithm
- This is the default in Red Hat, and an option on most other versions of Linux.
- MD5 isn’t cryptographically “better” than DES, but the MD5 scheme allows passwords of **arbitrary length**. Longer passwords are more secure, if you actually use them.
- MD5 is recommended for all sites.
- All major Linux distributions will recognise MD5 passwords in the **passwd** file if they are present. It isn’t necessary for all passwords on the system to use the same form of encryption.

Encrypted Password: Red Hat

- Under Red Hat, the `authconfig` command accepts the argument `--enablemd5` to turn on the generation of MD5 passwords
- Add `--kickstart` to keep the annoying interactive interface from appearing.
- Encrypted passwords in the **/etc/passwd** file are normally of constant length (34 characters for MD5, 13 for DES) regardless of the length of the unencrypted pass word. Passwords are encrypted in combination with a random "seed" so that a given password can correspond to many different encrypted forms (usually 64).
- If two users happen to select the same password, this fact usually cannot be discovered by inspection of the **passwd** file
- MD5 passwords are easy to spot because they always start with I.

UID Number: Definition

- The UID number is a unique **User ID** number used to identify to each individual user
- UIDs are unsigned 32-bit integers that can represent the values 0 to 4,294,967,296.
- However, because of interoperability issues with older systems, it is suggested to limit the largest UID at your site to 32,767 if possible. This isn’t usually a problem!
- By definition, root has UID 0.
- Most systems also define pseudo-users **bin**, **daemon**, and perhaps some others. It is customary to put such fake logins at the beginning of the **/etc/passwd** file and to give them low UIDs.
- To allow plenty of room for any other “non human users” that you might want to add in the future, it is recommended that you assign UIDs to real users starting at 500 (or higher).

UID Number: Rules

- Never a good idea to have multiple accounts with UID 0. While it might seem convenient to have multiple root logins with different shells and/or passwords, this set-up creates more potential security holes and gives you multiple logins to secure. If people need to have alternate ways to log in as root, you are better off if they use a program like `sudo`.
- Avoid recycling UIDs for as long as possible, even the UIDs of people that have left your organisation and had their accounts permanently removed. This precaution prevents confusion if files are later restored from backups, where users are identified by UID rather than by login name.
- UIDs should be kept unique across your entire organisation. That is, a particular UID should refer to the same login name and the same person on every machine. Failure to maintain such distinct UIDs can result in security problems with systems such as NFS and can also result in confusion when a user moves from one workgroup to another.

UID Number: Maintenance

- It can be hard to maintain unique UIDs when groups of machines are administered by different people or organisations
- The problems are both technical and political!
- There are a number of solutions:
  - Best solution is to have a central database that contains a record for each user and enforces uniqueness.
  - Simpler solution is to assign each group within an organisation a range of UIDs, and let each group manage its own set
- The simple solution keeps the UID spaces separate (a requirement if you are going to use NFS to share filesystems) but does not address the parallel issue of unique login names.

GID Number

- Like a UID, a **group ID** number is an unsigned 32-bit integer
- GID 0 is reserved for the group called “**root**”
- GID 1 is the group “bin” and GID 2 is the group “**daemon**”.
- Groups, (and a user’s membership of groups) are defined in **/etc/group**, with the GID field in **/etc/passwd** providing the default (or “effective”) GID at login time for a particular user
- Default GID is not treated specially when access is determined; relevant only to the creation of new files and directories.
- New files normally owned by the user’s effective group. However, in directories on which the **setgid** bit has been set and on filesystems mounted with the **grpid** option, new files default to the group of their parent directory.
- Linux allows a user to be in up to 32 groups at once, and all groups are considered when access calculations are performed

GECOS Field

- This is commonly used to record personal information about each user. It has no well-defined syntax.
- The GECOS field originally held the login information needed to transfer batch jobs from UNIX systems at Bell Labs to a mainframe running GECOS (the General Electric Comprehensive Operating System)
- Although you can use any formatting conventions you like, `finger` interprets *comma-separated* GECOS entries in the following order:
  - Full name (often the only field used)
  - Office number and building
  - Office telephone extension
  - Home phone number
- The `chfn` command lets users change their own GECOS information.

Home Directory

- Users' shells set the working directory to their home directories when they login.
- If a user's home directory is missing at login time, the system prints a message such as "no home directory."
- Linux systems *generally* allow the login to proceed and put the user in the root directory.
- To disallow logins without a valid home directory, set DEFAULT_HOME to no in **/etc/login.defs**
- Be aware that if home directories are mounted over a network filesystem, they may be unavailable in the event of server or network problems.

Login Shell

- The login shell is normally a command interpreter such as the Bourne shell or the C shell `(/bin/sh` or `/bin/csh`), but it can be any program. bash is the default and is used if **/etc/passwd** does not specify a login shell.
- On Linux systems, `csh` and **`sh`** are really just links to `tcsh` (a superset of the C shell) and `bash` (the GNU "Bourne again" shell), respectively. Most distributions also provide a public-domain version of the Korn shell, `ksh`
- Users can change their shells with the `chsh` command. The file **/etc/shells** contains a list of "valid" shells that `chsh` will permit users to select
- SuSE enforces this list, but Red Hat just warns you if the selected shell is not on the list. If you add entries to the shells file, be sure to use **absolute paths** since `chsh` and other programs expect them.

The /etc/shadow File: Definition

- The **/etc/shadow** file is readable only by the super-user
- The file serves to keep encrypted passwords safe from prying eyes. It also provides account information, which is not available from **/etc/passwd**.
- The use of shadow passwords is standard on some distributions, and configured as an optional package on others. Even when they're optional, it's a good idea to treat them as if they were standard.
- When shadow passwords are in use, the old style password fields in **/etc/passwd** should always contain an **"x"**.
- The **shadow** file is **not** a superset of the **passwd** file, and the passwd file is **not** generated from it
- You must maintain both files (or use tools such as `useradd` that maintain them both on your behalf).

The /etc/shadow File: Contents

- Like /etc/passwd, /etc/shadow contains one line for each user. Each line contains nine fields, separated by colons:
  - Login name
  - Encrypted password
  - Date of last password change
  - Minimum number of days between password changes
  - Maximum number of days between password changes
  - Number of days in advance to warn users about password expiration
  - Number of days after password expiration that account is disabled
  - Account expiration date
  - A reserved field that is currently always empty
- The only fields required to be nonempty are the **username** and **password**.
- Absolute date fields in **/etc/shadow** are specified in terms of days (not seconds) since Jan 1, 1970, which is not a standard way of reckoning time on UNIX systems. Fortunately, you can use the `usermod` program to set the expiration field.

The /etc/shadow File: Fields 1 - 4

Here is a more complete description of each field:

- **Field 1**: The login name is the same as in /etc/passwd. This field simply connects a user’s **passwd** and **shadow** entries.
- **Field 2**: The encrypted password is identical in concept and execution to the one previously stored in **/etc/passwd**
- **Field 3**: The last change field indicates the time at which the user’s password was last changed. This field is generally filled in by **/usr/bin/passwd**
- **Field 4**: sets the number of days that must elapse between password changes. Once users change their password, they cannot change it again until the specified period has elapsed. This feature seems useless, and we think it could be somewhat dangerous when a security intrusion has occurred. It is recommend setting this field to 0.

The /etc/shadow File: Fields 5 - 9

- **Field 5**: sets the maximum number of days allowed between password changes. This feature allows the administrator to enforce password aging
- **Field 6**: sets the number of days before password expiration that the login program should begin to warn the user of the impending expiration.
- **Field 7**: specifies how many days after the maximum password age has been reached to wait before treating the login as expired.
- **Field 8**: specifies the day (in days since Jan 1, 1970) on which the user's account will expire. The user may not log in after this date until the field has been reset by an administrator. If the field is left blank, the account will never expire.
- **Field 9**: reserved for future use

The /etc/shadow File: An Example

- A typical shadow entry looks like this:

```
kfarrell:inNO.VRSC1Wn.:11508:0:180:14: :12417:
```

- In eg: the user **kfarrell** last changed his password on July 4, 2001 (Field 3 = 11508 days since 1st January, 1970!).
- The password must be changed again within 180 days (Field 5 = 180)
- **kfarrell** will receive warnings that the password needs to be changed for the last two weeks of this period (Field 6 = 14)
- The account expires on December 31, 2003 (Field 8 = 12417 days since 1st January, 1970!)
- <u>**Note**</u>: You can use the `pwconv` utility to reconcile the contents of the **shadow** file to those of the **passwd** file, picking up any new additions and deleting users that are no longer listed in **passwd**.
- `pwconv` fills in most of the shadow parameters from defaults specified in **/etc/login.defs**

The /etc/group File: Definition

- The /etc/group file contains the names of UNIX groups and a list of each group's members. For example:

```
wheel:*:10:root,evi,garth,trent
csstaff:*:100:lloyd,evi
student:*:200:dotty
```

- Each line represents one group and contains **four** fields:
  - Group name
  - Encrypted password (vestigial and rarely used)
  - GID number
  - List of members, separated by commas (be careful not to add spaces)
- As in /etc/passwd, fields are separated by colons
- Group names should be limited to 8 characters for compatibility reasons, although Linux does not actually require this

The /etc/group File: Rules 1

- It's possible to enter a group password (to allow non-members of a group to change to it using `newgrp` command), but this is rarely done
- Most sites put stars in the password field, but it is safe to leave the password field blank. The `newgrp` command will not change to a group without a password unless the user is already listed as being a member of that group.

The /etc/group File: Rules 2

- As with usernames and UIDs, group names and GIDs should be kept consistent among machines that share files through a network filesystem. Consistency can be hard to maintain in a heterogeneous environment, since different operating systems use different GIDs for the same group names
- If a user defaults to a particular group in **/etc/passwd** but does not appear to be in that group according to **/etc/group**, **/etc/passwd** wins the argument. The group memberships granted at login time are really the union of those found in the **passwd** and **group** files. However, it's a good idea to keep the two files consistent.

The /etc/group File: Rules 3

- To minimise the potential for collisions with vendor-supplied GIDs, suggest starting local groups at GID 100.
- The UNIX tradition has always been to add new users to a group that represents their general category such as “students” or “finance.” However, such a convention increases the likelihood that users will be able to read each others’ files because of slipshod permission-setting, even if that is not the intention of the owner.

The /etc/group File: Rules 4

- To avoid this problem, recommend that you create a unique group for each user. Use the same name for both the user and the group.
- A user' s personal group should contain only that user. If you want to let users share files by way of the group mechanism, create separate groups for that purpose. The idea behind personal groups is to establish a more restrictive default group for each user so that files are not shared inadvertently.
- Red Hat' s version of `useradd` defaults to creating personal groups for new users

Adding Users: Summary of Steps

- The process of adding a new user consists of *three* steps required by the system, *two* steps that establish a useful environment for the new user and several steps for your convenience as an administrator.
- Required:
  - Edit the **passwd** and **shadow** files to define the user's account
  - Set an initial password
  - Create, `chown`, and `chmod` the user's home directory
- For the user for a working environment:
  - Copy default startup files to the user's home directory
  - Set the user's mail home and establish mail aliases
- For you (as system administrator):
  - Add the user to the **/etc/group** file
  - Configure disk quotas
  - Verify that the account is set up correctly
  - Add the user's contact information and account status to your database

Adding Users: Methodolgy

- The `useradd` command and its brethren can perform some of these steps for you, but in the next few slides we' ll go over the steps as you' d execute them by hand.
- This is mostly so that you can see what the supplied tools are doing.
- In real life, it' s generally preferable (faster and less error prone) to run `useradd` or a similar home grown script.
- You must perform each step as root or use a program such as `sudo` that allows you run commands as root.

Editing the **passwd** & **shadow** files (1)

- To edit safely the **passwd** file, run `vipw` to invoke a text editor on a temporary **copy** of it
- The default editor is `vi`, but you can specify a different editor by setting the value of you EDITOR environment variable.
- The existence of the temporary edit file serves as a lock; `vipw` allows only one person to edit the **passwd** file at a time, and it prevents users from changing their passwords while the **passwd** file is checked out. When the editor terminates, `vipw` replaces the original **passwd** file with your edited copy.

Editing the **passwd** & **shadow** files (2)

- For example, adding the following line to /etc/passwd would define an account called “tyler”:

```
tyler:*:103:100:Tyler Soap, Room 27, Ext 119,:/home/staff/tyler:/bin/tcsh
```

- We’d also add a matching entry to **/etc/shadow** by running `vipw -s`:

```
        tyler:*::::::12417:
```

- This **shadow** line for “tyler” has no encrypted password or password ageing and sets the account to expire on December 31, 2003.

Setting an Initial Password (1)

- root can change any user’s password with the `passwd` command:

```
# passwd user
```

- `passwd` prompts you to enter a new password and asks you to repeat it. If you choose a short, all-lowercase, or otherwise obviously unsuitable password, `passwd` will complain and ask you to use something more complex. Red Hat and SuSE check prospective passwords against a dictionary for added security.

Setting an Initial Password (2)

- The `mkpasswd` utility that comes with Don Libes's `expect` package helps generate random passwords for new users.
- The assignment of a random password "forces" new users to change their passwords immediately, as random ones are difficult to remember. Don't confuse `expect`'s `mkpasswd` with the standard `mkpasswd` command, which simply encodes a given string as a password.
- Never leave a new account—or any account that has access to a shell—without a password.

Creating the User’s Home Directory

- Any directory created as root is initially owned by root => must change its owner and group with the `chown` command.
- The following sequence of commands would create a home directory appropriate for our example user:

```
# mkdir /home/staff/tyler
# chown tyler.staff /home/staff/tyler
# chmod 700 /home/staff/tyler
```

- Permission 700 = 111 000 000 = rwx --- ---
- Here permissions are represented as three octal numbers (base 8) by converting each of the three 3-bit permissions

Copying in Default Start-up Files 1

- You can customise some commands and utilities by placing configuration files in a user's home directory.
- Start-up files traditionally begin with a dot and end with the letters rc, short for "run command," a relic of the CTSS operating system.
- The initial dot causes is to omit these files from directory listings unless the `-a` option is used with the `ls` command
- If you don't already have a set of good default start-up files, **/usr/local/lib/skel** is a reasonable place to create and put them.
- Copy in some files to use as a starting point and modify them with a text editor. You may wish to start with vendor-supplied files from the **/etc/skel** directory.
- Be sure to set a reasonable value for the a new user's **umask** (we suggest 077, 027, or 022, depending on the friendliness and size of your site).

Copying in Default Start-up Files 2

- Depending on the user’s shell, **/etc** may contain system-wide start-up files that are processed before the user’s own start-up files.
- For example, `sh` reads **/etc/profile** before processing **~/.profile**
- For other shells, see the **man page** for that shell for details.

Copying in Default Start-up Files 3

- The command sequence for installing start-up files for the new user tyler would look something like this:

```
# cp /usr/local/lib/skel/.[a-zA-Z]*
  ~tyler
# chmod 644 ~tyler/.[a-zA-Z]*
# chown tyler ~tyler/.[a-zA-Z]*
# chgrp staff ~tyler/.[a-zA-Z]*
```

- Note that we cannot use

```
# chown tyler ~tyler/.*
```

- because tyler would then own not only his own files but also the parent directory ".." **(/home/staff**) as well. This is a very common and dangerous sysadmin mistake!

Setting the user' s mail home

- It is convenient for each user to receive email on only one machine.
- This scheme is often implemented with an entry in the global aliases file **/etc/mail/ aliases** or the `sendmail` userDB on the central mail server.

Editing the /etc/group File

- To continue the processing of the new user tyler, we should add his login name to the list of users in group 100, since that was the default group to which we assigned him in the **/etc/passwd** file.
- Strictly speaking, tyler will be in group 100 whether he is listed in **/etc/group** or not, because his **passwd** entry has already given him this membership.
- However, this information should be entered in **/etc/group** so that you always know exactly which users belong to which groups.

Setting Disk Quotas

- If your site uses disk quotas, you should set quota limits for each new account with the `edquota` command.
- `edquota` can be used interactively, but it is more commonly used in “prototype” mode to model the quotas of the new user after those of someone else. For example, the command

```
# edquota -p proto-user new-user
```

- sets new-user’ s quotas to be the same as proto-user’ s.
- This way of using `edquota` is especially useful in scripts.

Verifying the New login

- To verify that a new account has been properly configured, first log out, then log in as the new user and execute the following commands:

```
$ pwd
```

- to verify the home directory

```
$ ls -la
```

- to check owner/group of startup files
- You will need to notify new users of their login names and initial passwords... But not by e-mail!
- Be sure to remind new users to change their passwords immediately. You can enforce this by setting the password to expire within a short time.

Recording a user’ s Status and Contact Information

- If you manage a large and changeable user base, need a formal way to keep track of accounts.
- Maintaining a database of contact info and account statuses helps to determine who someone is and why they have an account once the act of adding them has faded from memory.
- It’ s a good idea to keep complete contact info on hand so that you can reach users in the event of problems or misbehaviour.

Removing Users: Checklist

- When a user leaves an organisation, their login account and files should be removed from the system
- This procedure involves removing all references to the login name that were added by you or your `useradd` program. The automated counterpart to `useradd` is `userdel`. If you remove a user by hand, you may want to use the following checklist:
  - Set the user’s disk quota to zero if quotas are in use.
  - Remove the user from any local user databases or phone lists.
  - Remove the user from the aliases file or add a forwarding address.
  - Remove the user’s **crontab** file and any pending **at** jobs.
  - Kill any of the user’s processes that are still running.
  - Remove any temporary files owned by the user in **/var/tmp** or **/tmp**
  - Remove the user from the **passwd**, **shadow**, and **group** files.
  - Remove the user’s home directory.
  - Remove the user’s mail spool.

Removing Users: Cleanup (1)

- Before removing a user’s home directory, relocate any files that are needed by other users. Since one often can’t be sure which files those might be, it’s a good idea to make an extra tape backup of the user’s home directory and mail spool before deleting them.

Removing Users: Cleanup (2)

- Having removed a user, may want to verify that the user's old UID owns no more files on the system. To find the paths of orphaned files, can use the `find` command with the `-nouser` argument. To prevent `find` "escaping" onto network servers, it's usually best to check filesystems individually with `-xdev`:

```
 # find f -xdev -nouser
```

Removing Users: Cleanup (3)

- If the organisation assigns individual workstations to users, it’s generally simplest and most efficient to reinstall the entire system from a master template before turning the system over to a new user. Before performing reinstallation, it’s a good idea to back up any local files on the system’s hard disk in case they are needed in the future.

Disabling Logins (1)

- Sometimes, a user's login must be temporarily disabled
- Before networking invaded the computing world, one could just put a star in front of the encrypted password, disabling the local login account. However, users could still log in across the network without entering a password, so this technique no longer works very well.
- Nowadays, replace the user's shell with a program that prints a message explaining why the login has been disabled and providing instructions for rectifying the situation.

Disabling Logins (2)

- This pseudo-shell should not be listed in **/etc/shells**; many daemons that provide nonlogin access to the system (e.g., `ftpd`) check to see if a user's login shell is listed in **/etc/shells** and will deny access if it is not (which is the behaviour you want). Unfortunately, this message will probably not be seen if logins at your site typically occur through a window system.

Account Management Utilities

- The `useradd` command adds users to the **passwd** file (and to the shadow file if applicable).
- It provides a command-line-driven interface that is easy to run by hand or to call from a home-grown `adduser` script.
- The `usermod` command changes the **passwd** entries of existing users.
- The `userdel` command removes a user from the system, optionally deleting the user's home directory.
- The `groupadd`, `groupmod`, and `groupdel` commands also operate on the /etc/group file
- See the man pages or the books listed in the references for instructions on how to use these!

References

- “Linux Administration Handbook”, by Nemeth, Snyder and Hein, (Prentice Hall, 2002).
- “Linux System Administration”, by Stanfield and Smith, (Sybex, Craig Hunt Linux Library, 2001)