USERS & GROUPS, BACKUPS

Basic System Administration

USERS AND GROUPS

- Users and Groups define access to the operating system through the file permission scheme.
- Root is the super user, and the only user with special permissions
- Every user is a member of at least one group, which is called their <u>primary group</u>. The main purpose of this primary relationship is to define group owner of created files.
- Users can have a <u>secondary group</u> membership in as many groups as needed. These secondary relationships exist to broaden a user’s access to the files on the system.

SIDE NOTE: SU AND SUDO

- Best practice states that a user should always log in as a regular user, then switch to the root user when necessary for a system administration task. There are two tools available to do this:
  - `su`: switch user. As a regular user, this allows you to switch to the root account if you know the root password.
  - `sudo`: "su do". Perform an action as root or another user. If configured for access, you only need your password. Use `visudo` to edit configuration.

CONFIG FILES

- User information is stored in two files:
  - `/etc/passwd`
  - `/etc/shadow`
- Group information is stored in one file:
  - `/etc/group`

/ETC/PASSWD

- List of user records, one per line, with columns separated by colons. Format:
- `login:x:userid:groupid:gecos:homedir:shell`
- Examples:
  - `root:x:0:0:root:/root:/bin/bash`
  - `mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash`

/ETC/SHADOW

- Similar colon-separated-column list of records:
- `login:password:`*password aging fields*
- Aging fields track dates for password resets, locks, etc
- Examples:
  - root:pB8msP1fCbCqc:13904:0:99999:7:::
  - nisburgh:vRoPw6a/jQsp.:14466:0:99999:7:::

/ETC/GROUP

- Same colon-separated-column list of records format
- `groupname:grouppassword:groupid:secondarymembers`
- Group passwords allow temporary access to a group, are rarely used and not set up by default
- Examples:
    - `daemon:x:2:root,bin,daemon`
    - `apache:x:48:jack,nisburgh`

MANAGEMENT

- While it is possible to edit the three files directly, it's easier and safer to use the management commands to create, modify and delete users and groups:
  - `useradd, usermod, userdel`
  - `groupadd, groupmod, groupdel`

USERADD

- `useradd`: Add a new user to the system
- Accepts various arguments to control the settings on the user account. Most common is the -g option to specify the primary group of the user, and the -G option to list secondary group memberships. Examples:
  - `useradd lisa`
  - `useradd -g clowns -G trouble,simpson bart`

USERMOD, USERDEL

- `usermod`: Modify a user's settings. Example:
  - `usermod -G detention bart`
- `userdel`: Remove a user from the system. Main option to consider is `-r`, which tells `userdel` to remove the user's home and spool directories. Example:
  - `userdel moe`

GROUP COMMANDS

- `groupadd`: Adds a new group to the system. Example:
  - `groupadd bullies`
- `groupmod`: Mainly used to rename a group. Example:
  - `groupmod -n mktg mkg`
- `groupdel`: Remove a group. Example:
  - `groupdel microsoft`

PASSWORDS

- `passwd`: Change login password.
- Root can change the password for any user on the system
- Root can also setup password aging, allowing for timed password resets and account disabling
- `passwd` is also the preferred way to lock a user account:
  - `passwd -l mary`

PASSWORD AGING

- To set the maximum lifetime for a user's password:
  - `passwd -x days login`
- When a user's password has expired, you can set the number of days it can remain expired before disabling the account completely:
  - `passwd -i days login`

IMPORTANT USER ENVIRONMENT FILES

`/etc/skel`	default template for a newly-added user's home directory
`/etc/profile`	sets environmental variables used by all users
`/etc/profile.d`	contains scripts specific to certain rpms
`/etc/bashrc`	contains global aliases and system settings
`~/.bashrc`	contains user aliases and functions
`~/.bash_profile`	contains user environment settings and can be set to automatically start programs at login

EXERCISES

- Create a new group ‘dev’. Create a new user ‘alice’ as a member of the ‘dev’ group, with a description of “Alice from Dev” and a default shell of ‘/bin/csh’. Use the passwd command to set a password for alice, then log in as alice and verify her access.

- Lock alice’s account and verify she can’t log in anymore. Unlock her account and verify access once more. Add alice as a secondary member of the ‘gdm’ group.

- Set a maximum password lifetime of 4 weeks for the alice account. Look at the passwd, shadow and group files.

BACKUPS

- Why backup?
    - Hardware failures
    - Software failures
    - [Epic] User failures
    - Disasters

WHAT TO BACKUP?

- At minimum, all user data and intellectual property
- At maximum, entire systems, OS and all
- In reality, many factors drive what gets backed up:
  - budget
  - time
  - resources
  - need

WHERE TO BACKUP?

- Good question - many, many places
  - Local online copies
  - Remote online copies
  - Offline copies - Disk, Tape

HOW TO BACKUP?

- Small scenario:
  - rsync, tar, burning software, tape drive
- Large scenario:
  - rsync, tar, enterprise backup software, tape libraries

FLATTENING HIERARCHIES

- How to backup a *directory*? The directory represents an entire *tree* of files and directories? How can you put all of the information necessary to recreate the tree into one file?
- tar!
- Originally the Tape Archive tool. Used to backup directory trees to tape. Nowadays more commonly used to "flatten" a tree into one file.

CREATING A TAR ARCHIVE

- To create a tar archive:
    - `tar cf <tarfile.tar> <file> [file]...`
- The `c` option tells `tar` to <u>c</u>reate an archive. The `f` option is critical - it tells tar to put the archive in a <u>f</u>ile on disk, rather than on a tape device.
- You can add the `v` option (`tar cvf`) to get verbose output. Tar will report every file added to the archive.

VIEWING AN ARCHIVE

- To view an archive (a table of contents):
  - `tar tf tarfile.tar`
- The `t` option asks `tar` to print a <u>t</u>able of contents of the archive. If you add the verbose flag (`tvf`), tar will report detailed information on each file, similar to the long output of the `ls` command.

EXTRACTING AN ARCHIVE

- This is the tricky part of `tar`, and getting it right requires an understanding of how `tar` stores file in the archive.
- When an archive is created, the pathnames are stored into the archive. When you view the table of contents, you're viewing the relative pathnames as they would be created on extraction.
- This can sometimes confuse the user, and is best illustrated with an example...

EXTRACTING AN ARCHIVE

- If `tar tvf file.tar` reports:
  - `memo.txt`
  - `report/`
  - `report/data`
- Then when the archive is extracted, the resulting files will be:
  - `CWD/memo.txt`
  - `CWD/report/`
  - `CWD/report/data`
- Where `CWD` represents the current working directory

EXTRACTING AN ARCHIVE

- To extract an archive:
    - `tar xf tarfile.tar`
- `tarfile.tar` will be extracted to the current working directory, so be careful! Make sure you understand the contents of the tar file to be sure you don't accidentally overwrite existing files.

TAR EXAMPLES

Help remove the mud

EXERCISES

- From your home directory, create a tar backup of the test folder. Name the tar file ‘test.tar’. Verify it is correct by viewing the table of contents.
- Create a new directory in your home folder called ‘temp’. Change into this directory and extract your test.tar backup file. Can you see the ‘test’ folder and it’s contents?
- Browse through the man page for ‘diff’. Use ‘diff -r’ to compare the original ‘test’ folder with the newly extracted ‘test’ folder. Are there any differences?

COMPRESSION

- Tar files can get quite large, and storing/sharing them uncompressed wastes a large amount of storage space and bandwidth.
- Enter: compression.
- Compression uses complex algorithms to rewrite the contents of a file in a way that takes up less space, but can be reversed back to the original contents

COMPRESS

- One of the original compression algorithms: the Adaptive Lempel-Ziv. Not used very much any more, especially in Linux environment
- Achieves 40-50% compression on average
- Extension: `.Z`
- Compress: `compress`
- Decompress: `decompress`

GZIP

- Updated algorithm: Limpel-Ziv 77 (LZ77)
- Achieves 60-70% compression on average
- Extension: `.gz`
- Compress: `gzip`
- Decompress: `gunzip`

BZIP2

- Powerful algorithm: Burrows-Wheeler Block Sorting Huffman Coding
- Achieves 50-75% compression on average
- Extension: `.bz2`
- Compress: `bzip2`
- Decompress: `bunzip2`

TAR + COMPRESSION

- Once a tarball has been created, it's generally compressed with `gzip` or `bzip2`:
    - `gzip -9 tarfile.tar`
    - `bzip2 -9 tarfile.tar`
- The `-9` option tells the compression tool to maximize compression efficiency (taking longer). 1-9 are acceptable values, with `-1` indicating minimal efficiency and maximum speed.

ZIP FILES

- Zip files, originally put forward in the DOS/Windows world via the pkzip tools, and now winzip, are actually a combination of hierarchy archiving and compression.
- Basically, zip files include the features of tar and compression in one format! Advantages and disadvantages, of course.
- There are open source tools which allow access to creating, viewing and extracting zip files in the Linux environment.

ZIP

- Lots of algorithms implemented
- Varying compression ratio depending on algorithms used
- Extension: `.zip`
- Compress: `zip`
- Decompress: `unzip`

ZIP

- Remember, zip files are not just compressed files. The zip archive actually contains files and directories within it, so the interface is closer to `tar` than `gzip` or `bzip2`.
- Generally, zip files are only encountered in the Linux world when interacting with the Windows world. Within Linux, everything is a compressed tarball.

EXERCISES

- Make several copies of test.tar and use gzip to compress them. Try once with compression level 9 and once with compression level 2. Check the sizes of each.
- Use bzip2 to compress one of the copies. Compare it's size with the gzip sizes.

slideshow.end();