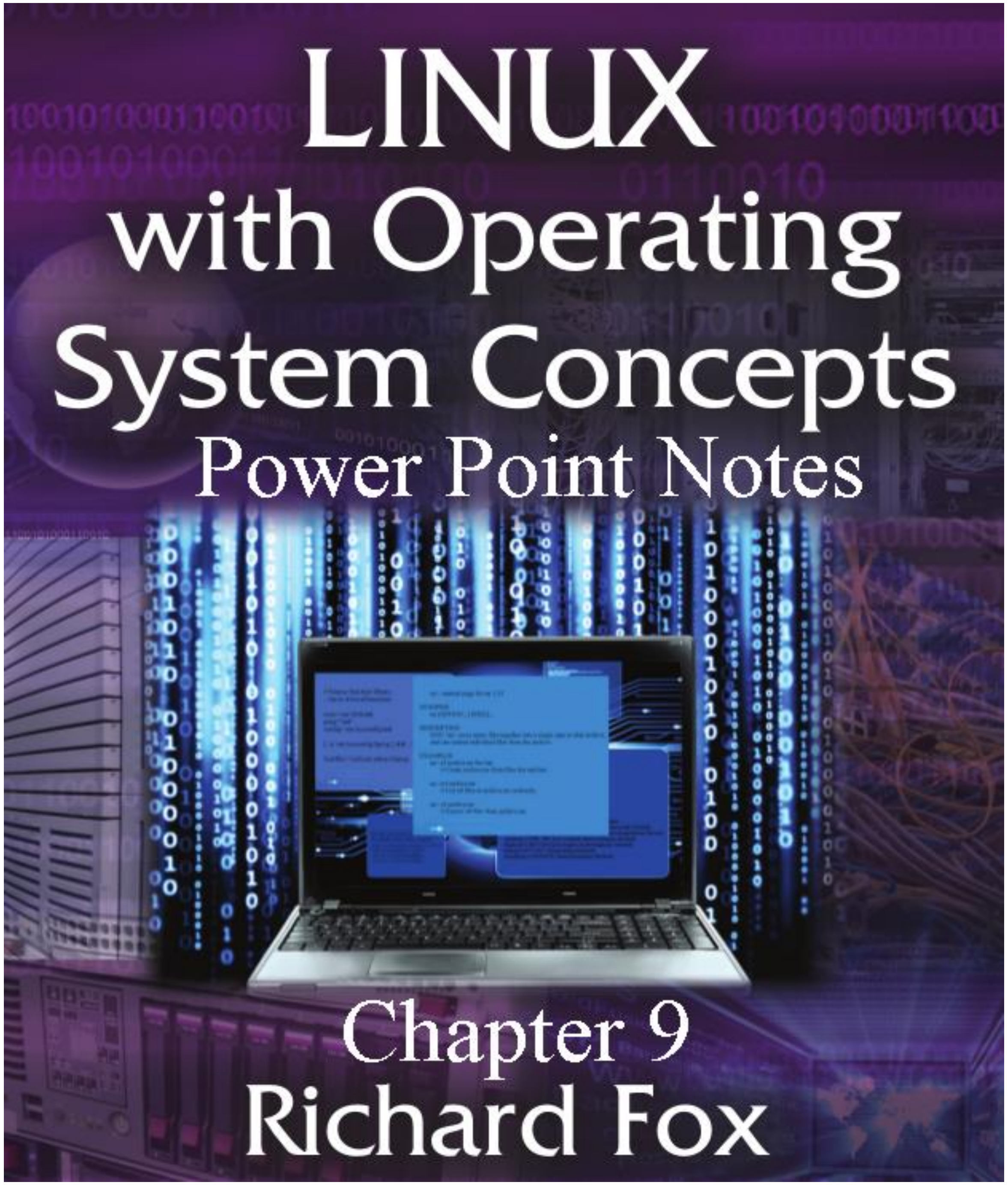
LINUX with Operating System Concepts

Power Point Notes



Chapter 9

Richard Fox

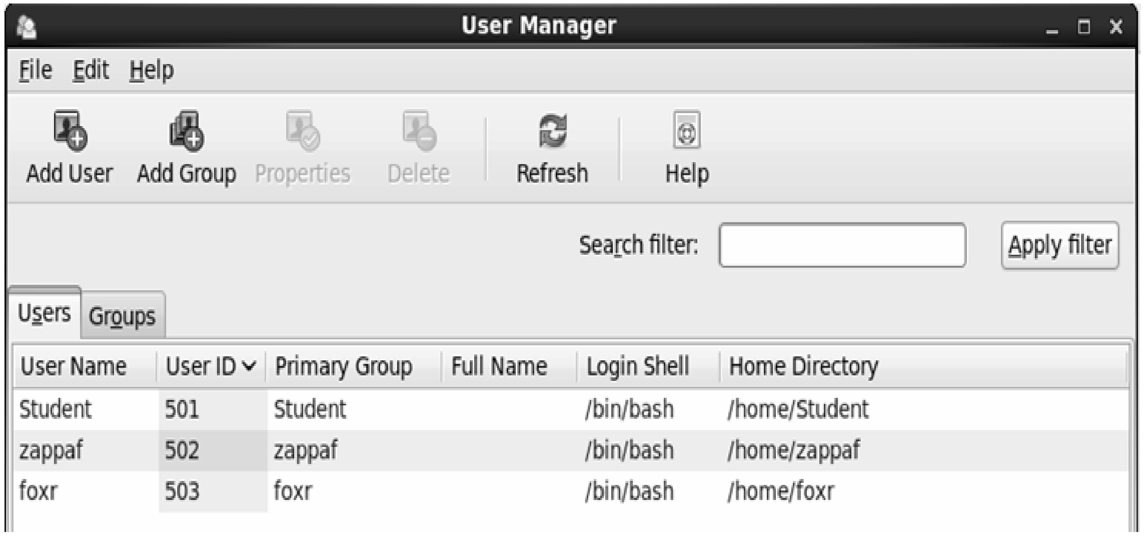
User Accounts

- The OS needs a mechanism to handle protection
  - We use user accounts and permissions
- There are two classes of accounts
  - root – access to everything
  - normal user account – has access to that user’s directory space and files, and public items (world readable, executable)
  - we can further differentiate normal user accounts into accounts for human users and for software

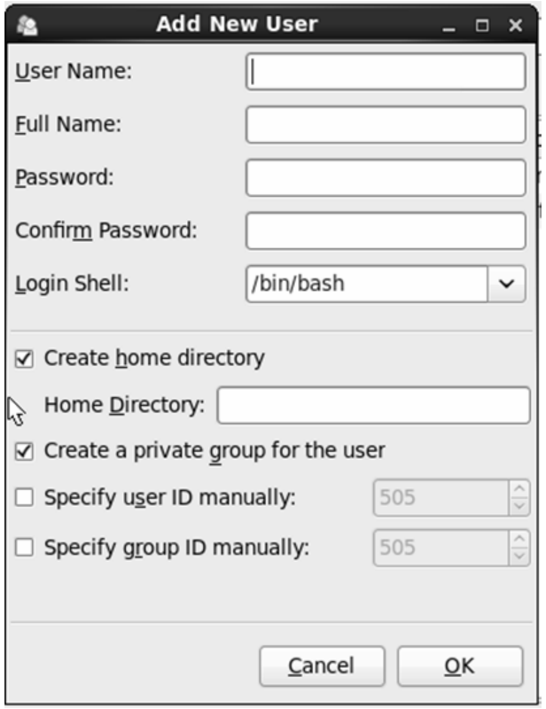
User Accounts: Attributes

- A username, User ID number (UID), password
- Entries in both /etc/passwd and /etc/shadow
- Private group (optional) with a Group ID number (GID), entry in /etc/group
- Initial home directory, by default under the directory /home, with default files
- Log in shell, by default, Bash
  - software may have a directory under another directory (e.g., /usr or /var) and may have a login shell of /sbin/nologin to prevent people from logging in as software (a security violation)

Creating User/Group Accounts: GUI



User Name	User ID	Primary Group	Full Name	Login Shell	Home Directory
Student	501	Student		/bin/bash	/home/Student
zappaf	502	zappaf		/bin/bash	/home/zappaf
foxr	503	foxr		/bin/bash	/home/foxr



User Name:	
Full Name:	
Password:	
Confirm Password:	
Login Shell:	/bin/bash

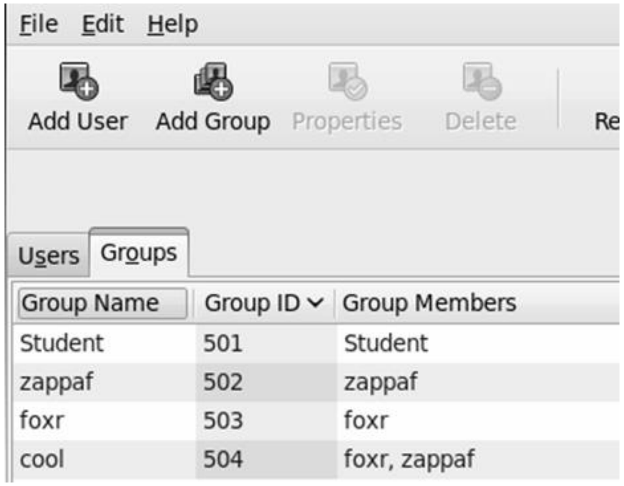
☑ Create home directory

Home Directory:

☑ Create a private group for the user

☐ Specify user ID manually: 505

☐ Specify group ID manually: 505



Group Name	Group ID	Group Members
Student	501	Student
zappaf	502	zappaf
foxr	503	foxr
cool	504	foxr, zappaf

Creating User Accounts: CLI

- useradd instruction
  - Only required parameter is the username to be created
    - many options to override defaults such as default home directory, default shell and default UID
    - default UID/GID is 1 greater than last UID/GID used
  - -m option to create home directory
- The CLI may be more challenging to use but to create numerous accounts, it is easier
  - Type your command
  - Press control+p (or up arrow) and revise the command for the next user
  - Quicker typing the commands than through the cumbersome GUI

Creating User Accounts: useradd Options

Option	Meaning	Example
-c comment	Fills comment field, used to specify user’s full name	“Richard Fox” – quote marks are necessary if the value has a blank space
-d directory	Used to alter the user’s home directory from /home/username to directory	-d /home/faculty/foxr
-D	Print default values to see what defaults are currently set as, including directory, expiration value, default shell, default skeleton directory (see below), default shell, and whether to create an email storage location	
-e date	Set expiration date to date	-e 2014-05-31
-g GID	Alter private group ID to this value, otherwise it defaults to 1 greater than the last issued GID	-g 999
-G groups	Add user to the listed groups; groups are listed by name or GID and separated by commas with no spaces in between	-G faculty,staff,admin
-k directory	Change the default skeleton directory (this is explained in subsection G)	-k /etc/students/skel

Creating User Accounts: useradd Options

Option	Meaning	Example
-l	Do not add this user to the lastlog or faillog log files; this permits an account to go “unnoticed” by authentication logging mechanisms, which constitutes a breach in security	
-m	Create a home directory for this user	
-M	Do not create a home directory for this user (the default case so can be omitted)	
-N	Do not create a private group for this user	
-o	Used in conjunction with –u so that the UID does not have to be unique, see –u	-u 999 –o
-p passwd	Set the user’s initial password; passwd must be encrypted for this to work	
-r	Create a system account for this user	
-s shell	Provide this user the specified shell rather than the default shell; for software, you will often use this to establish the shell as /sbin/nologin	-s /bin/csh
-u UID	Give user the UID of UID rather than the default (one greater than the last UID); can be used with –o so that two users share a UID	-u 999

Creating User Accounts: useradd

- useradd foo1
  - create new user account foo1 with all of the default values except for a home directory (because –m was not used)
- useradd –m foo2
  - create new user account foo2 with all of the default values including a home directory at /home/foo2
- useradd –m –d /home/students/foo3
  - create new user account foo3 with a home directory of /home/students/foo3
- useradd –m –u 1001 foo5
  - create new user account foo5 with UID of 1001
- useradd –m –o –u 1001 foo5jr
  - create new user account foo5jr who will have the same UID as foo5
- useradd –m –e 2015-12-31 –l –r backdoor
  - interested in creating a backdoor account?
- useradd –l –M –N –s /sbin/nologin softwaretitle
  - create an account for softwaretitle that has no group, no login, no home directory and is not logged in lastlog or faillog log files

Creating User Accounts: Defaults

- The default values for useradd can be viewed with the command
  - useradd –D
- The output might look like this
  - GROUP=100 (this is the default group number to use for any user account not given a private group through –N)
  - HOME=/home
  - INACTIVE=-1
  - EXPIRE=
  - SHELL=/bin/bash
  - SKEL=/etc/skel
  - CREATE_MAIL_SPOOL=yes
- To change a default value use useradd –D option value as in useradd –D –s /bin/csh to change from /bin/bash to /bin/csh

Creating Groups: CLI

- The groupadd instruction has fewer options
  - groupadd [options] groupname

Option	Meaning
-f	Force groupadd to exit without error if the specified groupname is already in use, in which case groupadd does not create a new group
-g GID	Use the specified GID in place of the default, if used with –f and the GID already exists, it will cause groupadd to generate a unique GID in place of the specified GID
-o	Used with –g so that two groups can share a GID
-p passwd	Assign the group to have the specified passwd
-r	Create a system group

Creating a Large Number of Accounts

- Enter initial useradd command
  - useradd –c “Mike Keneally” –m keneallym
- Use command line editing to convert the above for new user George Duke (dukeg)
  - control+p – recall the instruction
  - escape+b – move to beginning of user name
  - control+k (or escape+d) – delete username
  - dukeg – enter new user name
  - control+a, escape+f, escape+f, control+f, control+f – move to the “M” in Mike Keneally
  - escape+d, escape+d – delete Mike Keneally (if there were more than two names in quotes, do additional escape+d’s)
  - George Duke – type the new name
  - <enter>
- Repeat for each new user account

Creating Accounts Through a Script

- Assume we have a file storing all of the new users by first name and last name
  - We will give each user an account name of the form lastname firstinitial as in foxr or zappaf

```
#!/bin/bash
while read first last; do
		name="$first $last"
		username="$last${first:0:1}"
		useradd –c "$name" –m $username
done
```

Notice the “” used around $name after -c

What if we have two users who have the same last name and first initial like Tom Fowler and Tim Fowler?

We will modify this script to search /etc/passwd for lastname-firstinit and add a number to the end of the user name

Creating Accounts Through a Script

- Revised script
  - We obtain the number of users who share the same lastname-firstinit and add 1 to it and tack this on to the username

```
#!/bin/bash
while read first last; do
        name= "$first $last"
        username="$last${first:0:1}"
        n=`egrep –c $username /etc/passwd`
        n=$((n+1))
        username=$username$n
        useradd –c "$name" –m $username
done
```

Creating Accounts Through newusers

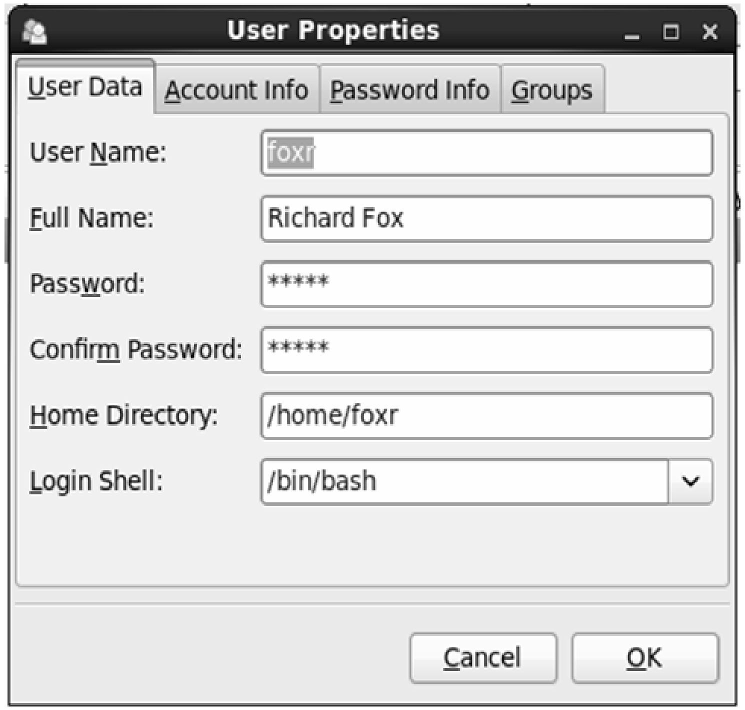
- Another useful command is newusers
  - Given a file of user account information, newusers generates the accounts (unless the information leads to an error)
  - The file must include the following information for every new user
    - username:passwd:uid:gid:comment:dir:shell
  - The UID and GID fields are optional and if omitted will default to 1 greater than the previously generated user account
    - to omit these, use username:passwd:::comment:dir:shell
  - All other fields are required so this approach is somewhat more cumbersome than the script from the previous slide

Managing Users and Groups

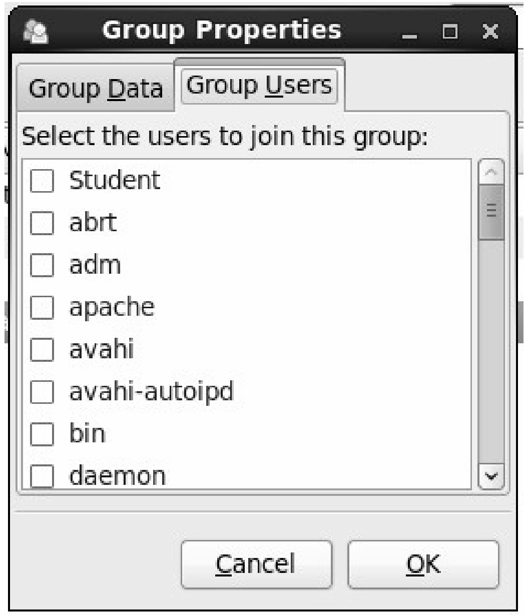
- The id command displays information about a user including UID, GID, other groups the user belongs to and this user’s SELinux context
  - id foxr returns
    - uid=503(foxr) gid=503(foxr) groups=503(foxr),504(cool) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
  - Without the username, id returns the current user’s information

Managing Users and Groups

- We can use the User Manager GUI tool to modify users and groups
- Select the user or group and select the Properties button
  - this brings up a properties window to alter the information about the selected user/group



User Name: foxr
Full Name: Richard Fox
Password: *****
Confirm Password: *****
Home Directory: /home/foxr
Login Shell: /bin/bash



Select the users to join this group:

- [ ] Student
- [ ] abrt
- [ ] adm
- [ ] apache
- [ ] avahi
- [ ] avahi-autoipd
- [ ] bin
- [ ] daemon

Managing Users and Groups

- usermod – similar options to useradd
  - -l newname – changes user’s username to newname
  - -L, -U – lock and unlock the account
  - -m dir – change home directory to dir
- groupmod – similar options to groupadd
  - –n newname – changes group name go newname
- userdel and groupdel – delete users and groups
  - for userdel, -f forces deletion even if user is logged in or has processes running
  - -r – deletes user’s home files (home directory, email directory) but other files owned by the user outside of the home directory remain

Passwords: Automatic Generation

- A third party open source software called apg is available (a password generator)
  - apg's options shown below

Option	Meaning
-a 0 or –a 1	Select between the two embedded random number generation algorithms, defaults to 0
-n num	Change the number of passwords produced from the default of 6 to num
-m min	Change the minimum length of the passwords produced to be a minimum of min characters (defaults to 8)
-x max	Change the maximum length of the passwords produced to be a maximum of max characters (defaults to 8)
-M mode	Change the mode, that is, the types of characters generated; mode can be S (include at least one non-alphanumeric character in every password), N (include at least one digit in every password), C (include at least one capital letter in every password)
-E string	Do not include any characters specified in the given string, not available for algorithm 0
-y	Generate passwords and then encrypt them for output

Passwords: Automatic Generation

- You can write your own script or instruction to generate passwords using /dev/urandom, openssl or use date and a hashing algorithm
  - urandom returns non-printable characters among its output so you would have to cut out anything non-printable
  - let's assume you only want letters and digits, use
    - `tr –cd  '[:alpha:]' < /dev/urandom | head –c8`
  - this generates a password of 8 characters
    - `date %s | sha256sum | head –c8`
    - `openssl rand –base64 6`

Passwords: /etc/shadow

- chage controls the data stored in the /etc/shadow file about a user’s password expiration
  - /etc/shadow stores the username and encrypted password for each user followed by
    - days since January 1, 1970 that the password was last changed
    - days before the password may change again
    - days before the password must be changed
    - days before warning is issued
    - days after the password expires that the account is disabled
    - days since January 1, 1970 that the account will become disabled
  - as in the following where :: means no entry for entries that have no values is empty or appears between ::
    - zappaf:…:15558:1:35:25:20:365:
    - foxr:…:15558:1:28:21:10::

Passwords: chage

- To alter the information in /etc/shadow that control a user's password (other than changing the password itself), use
  - chage [options] username

Option	Meaning
-d day	Set number of days when password was last changed (automatically set once password is changed, if never changed, this date is the number of days since the epoch)
-E day	Set day on which user's account will become inactive (will expire), specified as a date (YYYY-MM-DD) or the number of days since the epoch
-I day	Set number of days of inactivity after a password has expired before the account becomes locked. Using –I -1 removes any previously established inactivity date.
-l	Show this user's password date information
-M days	Number of days remaining before user must change password, use with –W.
-m days	Minimum number of days between which a user is allowed to change passwords, 0 means user free to change password at any time
-W days	Number of days prior to when a password must be changed that a warning is issued

Passwords: passwd

- The passwd command is used to change a user's password
  - passwd [options] username
- Many options overlap what chage does

Option	Meaning
-d	Disable the password (make the account password-less)
-i day	Same as chage –I day
-k	Only modify the password if it has expired
-l	Lock the account (user cannot login until unlocked)
-n day	Same as chage –m day
-S	Output status of password for the given account, similar to chage -l
-u	Unlock the locked account
-x day	Same as chage –M day
-w day	Same as chage –W day

Passwords: Add to Our Script

- We would probably want to also generate passwords when we generate user accounts
  - We modify our script (in this case, we use apg but we could also generate the passwords using one of the other approaches discussed earlier)
  - We store each username/password in a file under /root so that the system administrator can inform the new user of his or her initial password
  - Notice that the passwd command is interactive, we use --stdin to override that so that the password can be passed by redirection

Passwords: Add to Our Script

```
#!/bin/bash
while read first last; do
        name= "$first $last"
        username="$last${first:0:1}"
        n=`egrep –c $username /etc/passwd`
        n=$((n+1))
        username=$username$n
        useradd –c "$name" –m $username
        password=`apg –n 1`
        # or use
        # password=`tr –cd  '[:alpha:]' < /dev/urandom | head –c8`
        echo $password | passwd --stdin $username
        echo "$username $password" >> /root/tempPasswords
done
```

PAM

- Pluggable Authentication Module
  - Allows an administrator to specify, using a variety of modules, how authentication programs will perform their activities
  - An extensible service in that you can add modules and configure/reconfigure authentication programs
- Example: passwd
  - User inputs initial password
  - Program authenticates user
  - User inputs new password
  - Program updates password storage (/etc/shadow)

PAM: Configuration

- You configure PAM for a specific authentication program
  - Place the configuration information into a textfile stored in the directory /etc/pam.d
    - files exist for such programs as atd, crond, sshd, login, passwd, reboot, halt, poweroff, su
  - Files contain directives where each directive has three parts: module type, control flag, module filename
  - Module type will be one of auth (obtain user password and authenticate), account (restriction on account usage), session (maintains a session after logging in), password (changing passwords)

PAM: Configuration

- The configuration can include several directives of the same module type
  - This is known as a stack
  - Within a stack, we have to control the interpretation of these directives – for instance, do all directives have to succeed or just one?
- The interpretation is handled by control flags
  - requisite – the given module must succeed for access to be permitted, if it fails, the application must deal with the failure
  - required – all required directives must succeed
  - sufficient – any one of the directives must succeed
  - optional – success or failure does not impact the application itself but instead is used to determine if some other task (e.g., logging) should take place
- Another directive is include to "piggy-back" on other files' directives (re-use other configuration)

PAM: Example

- /etc/pam.d/su – the configuration file for how su works

```
auth        sufficient    pam_rootok.so
auth        include       system-auth
account     sufficient    pam_succeed_if.so uid=0  use_uid quiet
account     include       system-auth
password    include       system-auth
session     include       system-auth
session     optional      pam_xauth.so
```

System-auth is shown here:

```
auth        required      pam_env.so
auth        sufficient    pam_fprintd.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so
```

Common User Resources: /etc/skel

- The /etc/skel directory stores any files that will be duplicated into a new user's home directory
  - The system administrator will set this up, placing files that either the sysadmin wants all users to have (.bashrc) or files that might be useful (.mozilla subdirectory for Firefox browser)
  - You will probably find files like .bashrc, .bashrc_profile, .bash_logout and directories like .gnome and .mozilla

Common User Resources: /etc/bashrc

- When a user opens a Bash shell, the /etc/bashrc is executed
  - This allows the system administrator to establish default settings for all users' Bash settings
- There is also /etc/profile for all user logins irrelevant of shell
  - These scripts will establish environment variables like
    - EDITOR – for default editor (/bin/vi or /bin/vim)
    - PATH – an initial PATH variable
    - PS1 – user prompt
    - umask – initial umask value for file and directory permissions

Common User Resources: login.defs

- The file /etc/login.defs is another tailorable resource for the system administrator
  - The file establishes common user defaults
  - Some of the directives are listed below and others are shown on the next slide
    - DEFAULT_HOME – specifies whether a user is allowed to login if the user’s home directory is not currently available
    - MAX_MEMBERS_PER_GROUP – establishes a maximum number of users allowed in any one group, should you add more members than this maximum, a new group of the same name is created to handle the overflow
    - UMASK – default umask value (if none has been specified in another location like /etc/profile)

Directive	Usage	Possible Values
CREATE_HOME	When creating a user account, does useradd default to automatically creating a home directory or not creating a home directory?	yes, no
ENCRYPT_METHOD	Encryption algorithm used to store encrypted passwords	SHA512, DES (default), MD5
ENV_PATH, ENV_SUPATH	To establish the initial PATH variable for logged in users, for root	PATH=/bin:/usr/bin
FAIL_DELAY	Number of seconds after a failed login that the user must wait before retrying	0, 5, etc
MAIL_DIR, MAIL_FILE	Default directory, filename of user mail spool files	/var/spool/mail .mail or username
PASS_ALWAYS_WARN	Warn about weak passwords (even though they are still allowed)	yes, no
PASS_CHANGE_TRIES	Maximum number of attempts to change a password if password is rejected (too weak)	0 (unlimited), 3
PASS_MAX_DAYS, PASS_MIN_DAYS, PASS_MIN_LEN, PASS_WARN_AGE	Maximum, minimum number of days a password may be used, minimum password length, default warning date (as with chage –W)	Numeric value, 99999 for max, 0 for min are common defaults
UID_MIN, UID_MAX, GID_MIN, GID_MAX	Range of UID, GID available for useradd, groupadd	500, 60000

Common User Resources: ulimit

- The ulimit instruction places default restrictions on a user’s shell interaction
  - ulimit –a   list all establish limits
  - ulimit option value set option to given value (some of the options are shown below)

Option	Meaning
-c	Maximum core file size, in blocks
-e	Scheduling priority for new processes
-f	Maximum size of any newly created file, in blocks
-m	Maximum memory size useable by a new process, in kbytes
-p	Maximum depth of a pipe (pipe size)
-r	Real-time priority for new processes
-T	Maximum number of threads that can be run at one time
-v	Maximum amount of virtual memory usage, in kbytes
-x	Maximum number of file locks (number of open files allowable for the shell)

sudo

- The sudo command allows a user to execute an instruction as another user
  - The most common use is to let a user execute root-level commands
  - This is of course dangerous so we must be careful with who we provide sudo access with and what they can do with it
- The sudo command looks like this:
  - sudo [-u username|uid] [-g groupname|gid] command
  - Thus, you specify the command you want to execute and the user that the command should execute under
  - If you omit user/group, the default is to execute it as root
  - In order to use sudo, the user must have an entry in the /etc/sudoers file (see next slide)

sudo: /etc/sudoers

- This file, accessible only by root, lists all of the users who have sudo access along with the commands that they have access to
  - Format: username(s) host=command
    - where username(s) is the list of users or groups, groups are indicated using %group as in %users for all users, or the word ALL
    - host is the hostname which can be localhost, the specific machine's host, or the word ALL
    - command is the Linux command including any options or parameters that the command needs

sudo: Example

- Let’s imagine that we want all users to have the ability to add groups to the system
  - %users   localhost=/usr/sbin/groupadd
    - notice the need for the full path because sudo operates as root and we are not assured that root will have /usr/sbin in its PATH
- Now a user can create a new group through
  - sudo groupadd my_new_group
  - The user is asked to log in using their own password
  - An error message will arise warning the user that this occurrence is being logged if groupadd is not available to this user in the sudoers file

sudo: Discussion

- The sudo command can be very powerful
  - You will want to restrict its usage as you don’t want just anyone to have access to root commands
  - One command that users of a workstation might need is shutdown
  - You can also give users access to view files that they might otherwise not have access to
    - %users   localhost=/bin/ls /usr/local/protected
- To edit the sudoers file, use visudo
  - This command launches vi and lets you edit sudoers (only root can edit suoders)
  - This is better than directly editing sudoers in vi because visudo will syntactically check your sudoers file before you exit

Account Policies

- We should ask questions about user accounts before generating them
  - Will we have different levels (types) of users?
  - Will users have different types of software that they will need to access and different files that go along with them?
  - Will different levels of users require different resources, for instance different amounts of hard disk space, web server space? different amounts of processing?

Account Policies

- We should also ask questions about user access to resources
  - Will we need to enforce disk quotas? If so, on all partitions or on specific partitions? Can different users have different quota limits? (see chapter 10 for a discussion on disk quotas)
  - Will we need to establish priority levels for user processes?
  - Will users have sole access to their workstations; will workstations be networked so that users could potentially log into other workstations; will resources be sharable?

Account Policies

- We will also have questions about the accounts themselves
  - Will accounts exist for a limited amount of time or be unlimited? For instance, in a university setting, do we delete student accounts once the student graduates? If so, how long after graduation will we eliminate them? For a company, how long should an account remain in existence after the employee has left the company?
  - What password policy will we enact? No organization should ever use anything other than strong passwords. How often should passwords be changed? Can passwords be changed to something similar to previous passwords?

Account Policies

- Any organization should have policies
- As a system administrator, you might be asked to either define the policies or advise on their definitions
- Policies should be established with respect to
  - User accounts
  - Passwords
  - Disk space utilization
  - Miscellany

Account Policies

- User account questions
  - Does every user get an account?
    - this would be the most likely case.
  - Should users share accounts?
    - this can be a security violation and is generally not encouraged.
  - How long do user accounts remain active after the user is no longer with the organization?
    - companies may disable such accounts immediately
    - other organizations might wait a few weeks
    - universities often keep student accounts active for some time after graduation, possibly even permanently
    - if accounts are disabled or deleted, do the users get some notification?
    - what happens to any files owned by deleted accounts?
  - What resources come with an account?
    - file space? web server space? remote access? access to one workstation or all workstations? access printers?

Account Policies

- Password questions
  - Will users be given an initial password?
  - Will the organization enforce strong passwords?
  - How often will passwords require changing?
  - Can passwords be reused? If so, at what frequency? If not, can password variations be permitted?

Account Policies

- Disk utilization questions
  - Will files be placed locally or on a file server? If the latter, is file sharing allowed?
  - Will users have disk quotas?
  - Should files be encrypted?
  - Does file space also exist for the web server?
  - Are the users allowed to store anything in their file space?
  - Are there policies giving system administrators permission to search user file space for files that should not be there (e.g., illegal downloads)?

Account Policies

- Miscellany
  - How do we implement and enforce protection
    - passwords, biometric measures, keycards, other?
  - How do we implement and enforce security
    - will we use a VPN?
  - How often are backups performed and on what partitions?
  - What disaster planning do we have?
  - How do we replace resources? Do we have an account of money for replacements? What priorities might be established on how this money is spent?