LINUX FILE SECURITY

Overview

- Objective.
  - Linux Users Background Information
  - Linux Shell
  - Basic Commands
  - Creating and managing User Account
  - Creating and managing Group Account
  - Linux File System Hierarchy
  - Linux File System Permissions
  - Assigning and Removing Permissions

File Security

- Users uses files and create files
- Files may contain information requiring different security levels.
- Everything in Linux is a file:
  - Kernel modules
  - Device drivers
  - Programs
  - Commands
  - Log files
  - User Files
- The need to ensure files are secure from those accesing the machine.

File Security

- Security Objectives:
  - User Authentication
  - File Integrity
  - Access Authorization

Users Background Information

- Users - background information
  - There are several things to consider before attempting to create/install user accounts on a system.
    - Creating login names
      - How is a login constructed?
        - » **Users last name?**
        - » **Users first name?**
        - » **Combination of names?**
        - » **Random string?**
        - » **Alphanumeric string?**
        - » **It is best if the site has a policy about login names. Otherwise users will request login names that may not be acceptable (to management, society, ...).**

Users Background Information

- Assigning a homedir
  - On large systems there may be many “user” directories. The system administrator needs to think about how users are distributed across these file systems.
    - » Space requirements
    - » Project/Work Unit Association
    - » Other considerations (special needs)

Users Background Information

- Creating UID's
  - Each user must have a unique user-id. Most Unix systems use integers in the range of 0 - 65536.
    - » Are there special (reserved) userids?
    - » What happens at a large company/university where there are more than 65536 employees?
    - » Are UID's reused? For example, if an employee leaves the company, is their userid assigned to the next person hired in?

Users Background Information

- Assigning a shell
  - Once a user is created he/she is assigned a default shell.
    - sh - standard with almost every UNIX
    - csh - standard with almost every UNIX
    - bash - Standard with Linux
    - tcsh - Popular, but not generally shipped with system.
    - ksh - used by many install programs

Users Background Information

- In addition to the items above, the administrator may elect (or be forced) to set constraints like:
  - What workstations can the user access
  - What hours can the user access the workstation
  - Account expiration date
  - How often user must change their password
  - What are acceptable passwords

Users Background Information

- Format of password file
  - The Linux password file conforms to a very strict format:

robert:x:502:503:Finanace Officer:/home/robert:/usr/bin/csh

USER:x:UID:GID:full name :HOMEDIR:SHELL

- If the password file format is incorrect, one of the following situations may occur:
  - » Nobody listed after the error can login.
  - » Nobody can login
  - » The password file is automatically truncated by the system to remove the error.

Users Background Information

- Password file fields
  - User - the login name assigned to the user.
  - X : used to be the Password field
    - [xX] - Look in some alternate location encrypted password
  - UID - the UID assigned to this user
  - GID - the login group this user belongs to
    - Users may be in other groups (see /etc/group)

Users Background Information

- Password file fields
  - Homedir - the home directory assigned to this user
  - shell - the shell program assigned to this user
    - Make sure it is listed in /etc/shells!
- Sorting the password file
  - Some sites want names alphabetically.
    - Problem: What happens if an error occurs somewhere before the letter “r” in the password file?
  - Some sites want ascending UID’s.
    - Not real convenient when searching for a username.
  - Some sites have several password files, and use some tool to create password files for individual systems.

Users Background Information

- The principle method by which an operating system determines the authenticity of a user is by a password.
  - Good passwords are essential to the security of all operating systems.
  - Choosing passwords, educating users in the use of passwords, and choosing and employing one or more tools to enhance password security are tasks a sysadmin will face when creating user accounts.

Users

- Both Windows and UNIX systems employ reusable passwords as their default.
  - Reusable passwords have several problems.
  - First, they are vulnerable, either through manual or automated brute force attacks, to discovery if unchanged for long periods of time..
  - Reusable passwords are vulnerable to their own quality; poorly chosen passwords are more easily guessed.
  - If the user accesses the system using an insecure connection such as *telnet* or *ftp*, the user's password is transmitted over the connection in clear text, which is easily intercepted if an attacker is listening.

Users

- The first approach to improve password securiy is to educate the users of your systems to the dangers of reusable passwords.
  - Education on choosing good passwords and encouragement to change them periodically is universally applicable to all operating systems.
  - Good password construction techniques include assembling passwords from words separated by punctuation characters or numbers, or assembling a password using the first letter of each word in a phrase of the user's choosing.
  - Semester breaks, seasonal changes, and holiday breaks can help provide cues to encourage periodic password changes.

Users

- Password aging (or password expiration) is another method to improve password security.
  - The aging process allows the system manager to enforce the practice of changing account passwords on a regular basis.
  - The downside to password aging is the psychological factor. Some users dislike changing passwords.
  - Being asked to change with no warning may contribute to a user choosing a simpler, easily guessed password, or simply entering a new password and then changing back to the old password immediately afterward.
  - Password aging is most effective when the account user understands the reasons for periodically changing a password and the elements of a good password, and is given a chance to choose a good password.

Users

- Other features present in some UNIX variants are incorrect password attempt counters and account inactivity timers.
  - These can be employed to reduce the chances of success by an attacker guessing a user's password or of an old unused account being exploited to gain access to a system.
  - A password attempt counter records failed attempts to provide the system with a password. When a user attempts to log in, the number of failed password attempts is checked against a set limit.
  - The user account is disabled if the limit is exceeded. Likewise, an inactivity timer records the last time an account was used.

Users

- In the case of the inactivity timer, when a user attempts to log in, the length of inactivity for the account is compared to a set limit and the account is disabled if it has been inactive for longer than the limit.
- Both of these tools have the downside of preventing access by valid users and of adding additional work for the system administrator as he spends time resetting password attempt counters or inactivity timers triggered by the forgetful or infrequent user.

Users

- The long-term solution to the problems of reusable passwords is passwords that are used just once and not used again.
  - These one-time passwords make use of a shared secret typically held on a secure authentication server and a token held by the user, typically a small calculator or a program on a personal digital assistant (PDA), such as a Palm Pilot.
  - Instead of requesting a password when accessing a system, under one-time passwords the system responds to a log-in request with a challenge in the form of a string of numbers and letters.

Users

- This string is entered into the token that produces another string, the response.
- The response is entered instead of the password to gain access. Both the challenge and the response are a mixture of the shared secret and a continually changing value, usually the current date and time, ensuring that the same combination of challenge and response are never used more than once.

Users

- This sophisticated and secure scheme is not without its own problems. None of the Windows or UNIX variants seem to come with one-time passwords built in.
- They must be added to the system as a separate product. All users making use of the system will be required to carry a token with them, with the resulting problems of loss and damage to the token and the frustration to the user when they are unable to gain access.

Users

- Alternative password token systems are also available.
  - Physical devices such as a smart card or dongle that carry authentication information or provide challenge/response authentication.
  - The devices are read via special readers (smart cards) or are attached to a system via a connection such as a USB port (dongle).
  - Such devices are generally used in highly secure systems for which the cost of the additional hardware for every user can be justified.
  - Another technique is the use of biometric information such as the visual pattern of a fingerprint or the blood vessels in the retina.

Users

- This information is read from the person's finger or eye via a special-purpose reader.
  - Although believed to be very secure, biometrics suffer from problems such as how account revocation is performed, as the information being used for authentication is a permanent feature of the user.
  - As with smart cards and dongles, biometric authentication is only seen in situations for which the cost of the additional hardware can be justified.

Linux Shell

- The 'shell' is the Linux command interpreter
- The shell operates in a command processing loop:
  - Performs various substitutions and expansions on the command line
  - Executes the resulting command and waits for it to finish
  - Displays a 'prompt' and reads a command line
  - Loops back and prompts for another command

Linux Shell

- There are several shells have been written for UNIX and Linux
  - Bourne shell (sh),
  - Korn Shell,
  - C Shell,
  - Bourne Again Shell (bash)
- The core feauture set of all these shells is very similar
- We will use bash, the most popular shell on Linux

Linux Shell

- Example

  [root@localhost ~]# hostname

  localhost.domain

  [root@localhost ~]# (*Display the prompt again once it has displayed the output)*

Basic Commands

- Examples of commands

[root@localhost ~]# date

Fri Apr 16 11:48:33 BST 2004

[root@localhost ~]# id

uid=500(chris) gid=100(users) groups=100(users), 14(uucp)

[root@localhost ~]# cal

June 2014

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30

Basic Commands

Command options

- Command options modify the behavior of a command
- Two types
  - Short option Example (ls -l, cal -y, date -I)
    - prefixed by '-'
  - Long option Example (date –iso-8601)
    - prefixed by '--'

Basic Commands

Command Arguments

- Most commands accept arguments
- Arguments are often operands the commands need to work on.
- The arguments in most cases are the names of files or directories on which to operate.
- The command name, options, and arguments are separated by whitespace (spaces and tabs)
- Examples
- [root@localhost ~]# cal 1995 (prints 1995 car lender)

Basic Commands

- Enter the following commands and observe thier output
- [root@localhost ~]# ls (list the current directory)
- ls /home (list the home directory- displays users home directories)
- ls -l /home - Using Options (-l)and arguments (/home) together

Basic Commands

Command history

- Bash shell remembers the most recent commands you've entered
  - The list survives across logout / login, shared by all instances of bash
  - stored in the file .bash_history in your home directory
- The history command shows your command history
- **Examples**

1. history shows your entire command history
2. history 10 shows the last ten commands
3. history -c clears your command history

Basic Commands

Command history

The Previous commands can be selected and re-executed

- **Example**

1. !85 re-execute command 85
2. !string re-execute most recent command that began with string
3. !! re-execute last command

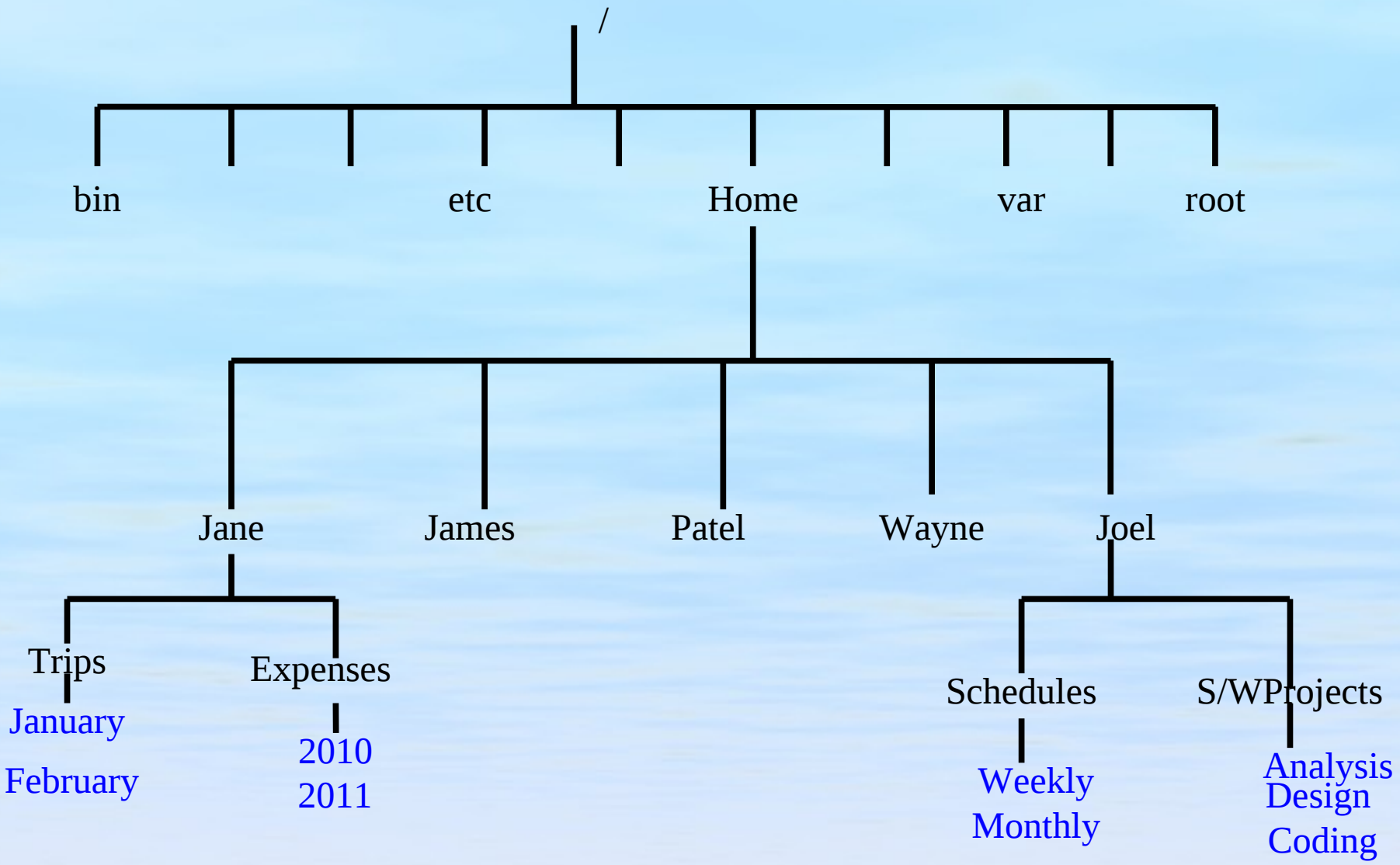
Basic Commands

Command history

A. The following keys are used:

- **↑ scroll back through history**
- **↓ scroll forward through history**
- **← move left along line**
- **→ move right along line**
- **string insert text string into line**
- **<DEL> delete character**
- **<ENTER> execute the command**

File System Hierarchy



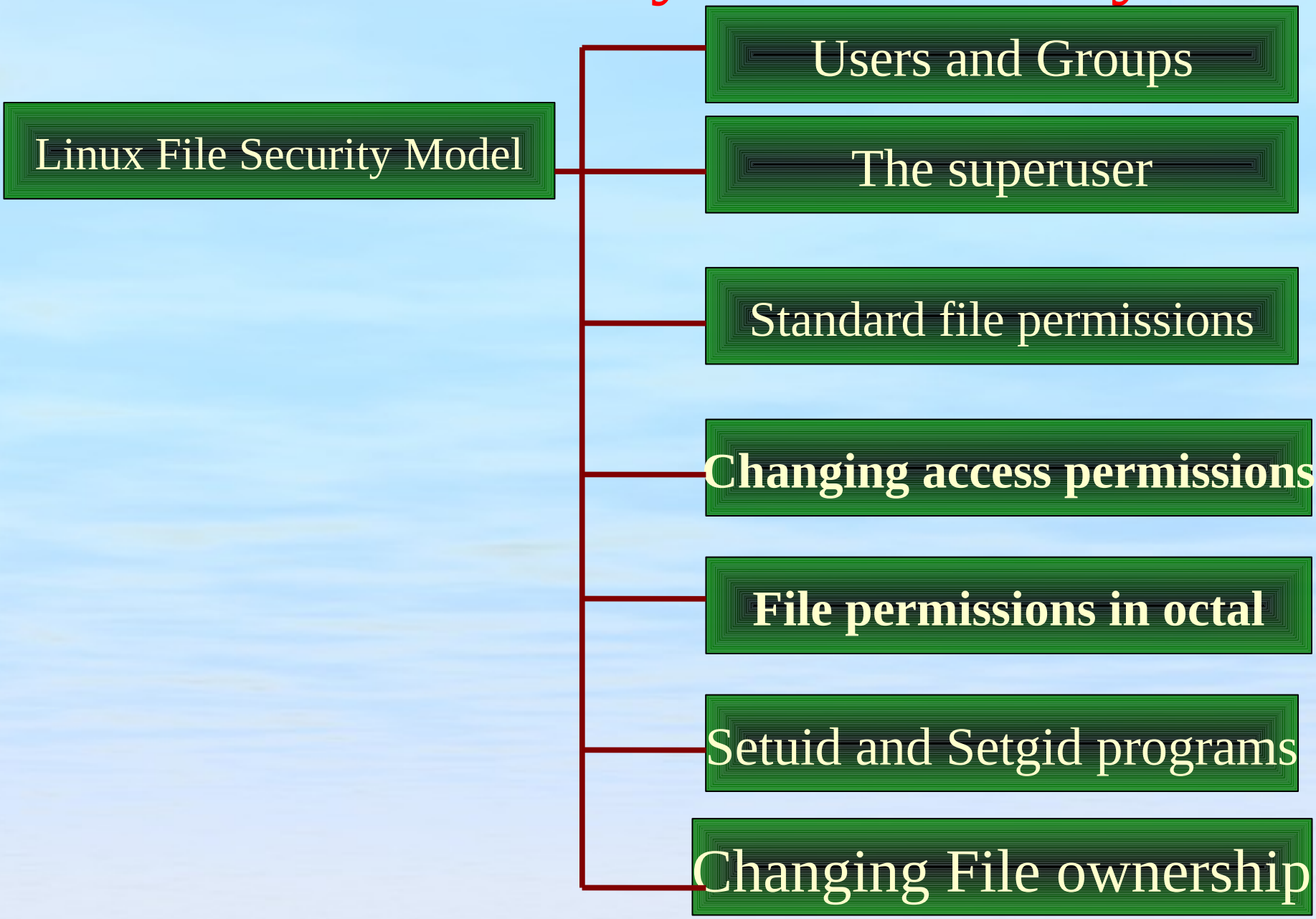
File System Navigation

- Suppose you root user and you are at James(Home directory for James)
- Which command will you use to:
  - Move to the top level
  - Move to the /etc directory
  - Go back to the original directory
  - List the bin content
  - List content for Jane's directory

File System Navigation

- Suppose you root user and you are at James(Home directory for James)
- Which command will you use to:
  - Move to Joel using absolute reference
  - Move to Joel using relative reference
  - List the s/w projects content in long format using absolute reference
  - copy the s/w projects content to my home directory(/root)
  - move the s/w projects content to Wayne home directory
  - List the entire file system using absolute reference

Linux File System Security



Users and Groups

- Every user has an account name (e.g. James) along with an associated numeric user ID (e.g. 500)
- Every user is associated with one named group which is their primary group
  - Groups allow additional flexibility in assigning access permissions to users.
- Users can also be associated with one or more secondary groups
- The command **id** shows your user identity and group memberships

$ id

uid=500(james) gid=100(users) groups=100(users),
14(uucp),16(dialout),17(audio),33(video)

Users and Groups

The user root

- Linux has a priviledged user account called the **super-user**
- The account name is usually root
- The numeric user ID is zero
- The root can access or modify any file or directory and run any command
- To start a new shell as root use the su command
- $ su - *(The '- flag causes root's login Envi. to be created)*
- Password: rootpassword
- Password: rootpassword

Users and Groups

The user root

Security Advice

- Always as the root user use a normal user account when doing non-admin tasks.
- Use the su - command to switch to the root account
- This start a root shell child process with the parent shell
- Once finished you result back to your regular user account environment.

Creating user accounts

- The useradd Command: Used to create user accounts
- Usernames may only be up to 32 characters long.
- Has the following options
  - -d homedir :Set the home directory to homedir
  - -u 600 :Specifies a user ID of 600
  - -g 120 :Specifies a primary group ID of 120
  - -m Create the home directory for the account. The home directory is initialised with a copy of the files in /etc/skel
  - -c Full Name Specifies the user's full name
  - -s shell Set the path name to the user's login shell

Creating user accounts

- The useradd Command: Used to create user accounts
- Has the following options

-e homedir : The date on which the user account will be disabled. (YYYY-MM-DD).

-G: A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening white space.

-r :Create a system account. System accounts will be created with no aging information in /etc/shadow, and no home directory

  – (Why a system account?)

-M Do not create the user´s home directory, even if the system wide setting from /etc/login.defs (CREATE_HOME) is set to yes.

Creating user accounts

- The useradd Command: Used to create user accounts
- Has the following options

-U, --user-group Create a group with the same name as the user, and add the user to this group.

-G: A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening white space.

-r :Create a system account. System accounts will be created with no aging information in /etc/shadow, and no home directory

  – (Why a system account?)

-M Do not create the user´s home directory, even if the system wide setting from /etc/login.defs (CREATE_HOME) is set to yes.

Creating user accounts

- The passwd Command:
- A utility is used to update user's authentication token(s)
- Has the following options
  - -l :is used to lock the specified account and it is available to root only
  - -c it will unlock the account password by removing the ! prefix. This option is avail able to root only.
  - -d This is a quick way to delete a password for an account. It will set the named account passwordless. Available to root only

Creating user accounts

- The passwd Command:
- A utility is used to update user's authentication token(s)
- Has the following options

-e Expire a password for an account. The user will be forced to change the password during the next login attempt. (root only).

-n set the minimum password lifetime, in days, if the user's account supports password lifetimes. (root only).

-W set the number of days in advance the user will begin receiving warnings that her password will expire

-i set the number of days which will pass before an expired password for this account will be taken to mean that the account is inactive and should be disabled,

-S output a short information about the status of the password for a given account

Creating user accounts

- The passwd Command: Examples

i. passwd -e walter

ii. passwd -n 20 walter set the minimum password lifetime,to 7 days,

iii. passwd -w 3 walater set the 3 to be the number of days in advance the user will be receiving warnings that her password will expire

iv. passwd -i 20 walter set the number of days which will pass before an expired password ccount is inactive

v. passwd -S 20 walter output a short information about the status of the password for a given account

Creating user accounts

- **Exercise**

1. Create the users as shown in the file system hierarchy in slide 14. (Set the password later)
2. Draw and fill in the table as shown:
3. Use the appropriate command options(given) and arguments to implement the security information information

User Name	Pass word	-d	-u	-g	-c	-s

Creating user accounts

- **Exercise..cont..**

1. Draw and fill in the table as shown:
2. Use the appropriate command options(given) and arguments to implement the security information information

User Name	Pass word	-e	-n	-w	-i	

Modifying and Deleting user accounts

- The usermod Command:
- Used by root to modify existing accounts
- Note
- The named user should not be executing any processes when this command is being executed if the user´s numerical user ID, the user´s name, or the user´s home directory is being changed.
- Usermod checks this on Linux, but only check if the user is logged in.
- Has the same options as the usseradd command
- **Example**
- Create a user account robert and password rober123t

Modifying and Deleting user accounts

- The usermod Command:
- **Example**

1. Create a user account robert and password rober123t
2. Run the command
3. #grep robert /etc/passwd
4. Out: robert:x:502:503::/home/robert:/bin/bash
5. #usermod -c "Finance Officer" -s /usr/bin/csh robert
6. Out: robert:x:502:503:Finanace Officer:/home/robert:/usr/bin/csh

The **userdel Command**

7. Root can delete existing accounts using **userdel,** for example:
8. # userdel -r robert // -r means that you remove the home directory too

Modifying and Deleting user accounts

- The chgrp Command:
- Change the group ownership of each FILE to GROUP.
- **-R, --recursive : operate on files and directories recursively**
- **-L traverse every symbolic link to a directory encountered**
- **-P do not traverse any symbolic links (default)**
- EXAMPLES

1. chgrp staff /u : Change the group of /u to "staff".
2. chgrp -hR staff /u: Change the group of /u and subfiles to "staff".

Creating and Deleting Groups

The groupadd and groupdel Commands

- The groupadd creates a new group account using the values specified on the command line plus the default values from the system.
- It has the following options:
  - -g, --gid GID The numerical value of the group ´s ID.
  - This value must be unique, and non-negative.
  - The default is to use the smallest ID value
  - greater than 999 and greater than every other group. Values between 0 and 999 are typically reserved for system accounts.

Creating and Deleting Groups

- The groupadd and groupdel Commands
  - Example

    #groupadd -g 1445 finance
  - By default if no gid is specified the next available ID is allocated

  The groupdel Commands
  - Used by root to delete groups
  - # groupdel finance
  - Note
  - One cannot delete a group which is someone's primary group

Users

- Setting Up a Workgroup Directory
- To create a useful workgroup directory for a small team
- The workgroup is to be called sales and has members jdoe, bsmith, and jbrown.
- The directory is /home/sales.
- Only the creators of files in /home/sales should be able to delete them.
- Members shouldn’t worry about file ownership, and all group members require full access to files.
- Nonmembers should have no access to any of the files.
- The following steps will satisfy the goals:

Users Account Configuration File

- The following configuration variables in /etc/login.defs determine the behavior of login process.
- CREATE_HOME (boolean) :Indicate if a home directory should be created by default for new users.
  - This setting does not apply to system users, and can be overriden on the command line.
- GID_MAX (number), GID_MIN (number) : Range of group IDs used for the creation of regular groups by useradd, groupadd, or newusers.
- MAIL_DIR (string) The mail spool directory.
  - This is needed to manipulate the mailbox when its corresponding user account is modified or deleted. If not specified, a compile-time default is used.

Users Account Configuration File

- The following configuration variables in /etc/login.defs determine the behavior of login process.
- MAIL_FILE (string) Defines the location of the users mail spool files relatively to their home directory.
- The MAIL_DIR and MAIL_FILE variables are used by useradd, usermod, and userdel to create, move, or delete the user´s mail spool.
- MAX_MEMBERS_PER_GROUP (number) Maximum members per group entry.
  - When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).
  - The default value is 0, meaning that there are no limits in the number of members in a group.

Users Account Configuration File

- configuration variables in /etc/login.defs
- PASS_MAX_DAYS (number) The max number of days a password may be used. If the password is older than this, a password change will be forced. If not specified, -1 will be assumed (which disables the restriction).
- PASS_MIN_DAYS (number) The minimum number of days allowed between password changes. Any password changes attempted sooner than this will be rejected. If not specified, -1 will be assumed (which disables the restriction).
- PASS_WARN_AGE (number) The number of days warning given before a password expires.
  - A zero means warning is given only upon the day of expiration, a negative value means no warning is given. If not specified, no warning will be provided.
- SYS_GID_MAX (number), SYS_GID_MIN (number) Range of group IDs used for the creation of system groups by useradd, groupadd, or newusers.

Users Account Configuration File

- configuration variables in /etc/login.defs
- SYS_UID_MAX (number), SYS_UID_MIN (number) :Range of user IDs used for the creation of system users by useradd or newusers.
- UID_MAX (number), UID_MIN (number): Range of user IDs used for the creation of regular users by useradd or newusers.
-

Access Control Mechanism

- Linux filesystem access control is implemented on a file by file basis.
- Every file has a set of properties collectively called the access mode'
- The mode is a part of the file's inode,
- The inode is the information retained in the filesystem that describes the file.
- A file's mode controls access by these three classes of users:
  - User :The user who owns the file
  - Group: The group that owns the file
  - Other: All other users on the system

Access Control Mechanism

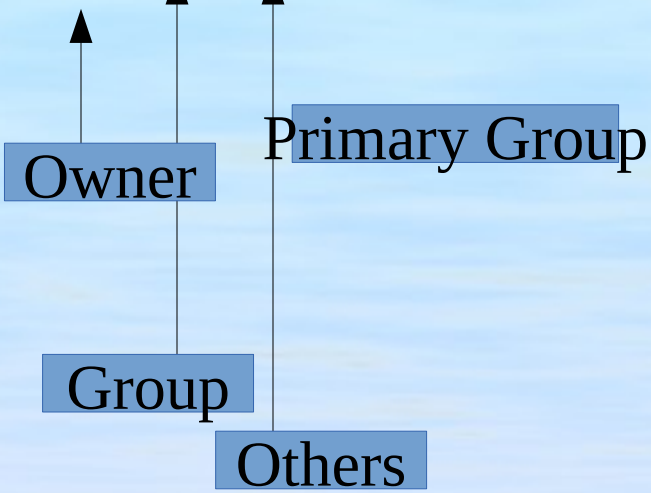
- Linux File system Security
- For each category the are a maximum of three access settings. rwx- **read, write and execute**
- The permissions are shown as a group of nine characters.

Example: rwxr-xr-x

Access Control Mechanism

- These attributes are shown in a long directory listing:
- Example.

```
[root@localhost ~]# touch myfile
[root@localhost ~]# ls -l myfile
-rw-r--r--. 1 root root 0 Jun  9 17:10 myfile
```



Access Control Mechanism

Permiss	Meaning to a file			Meaning to a	
r (read)	Able to see the contents of the Able to list the contents of the file			Able to list the contents of the file directory	
w (write)	Able to change the contents of the file			Able to create or delete files or subdirectories	
x (execute)	Able to run the file as a program or a script			Able to make the directory “current” or use it in a path name	

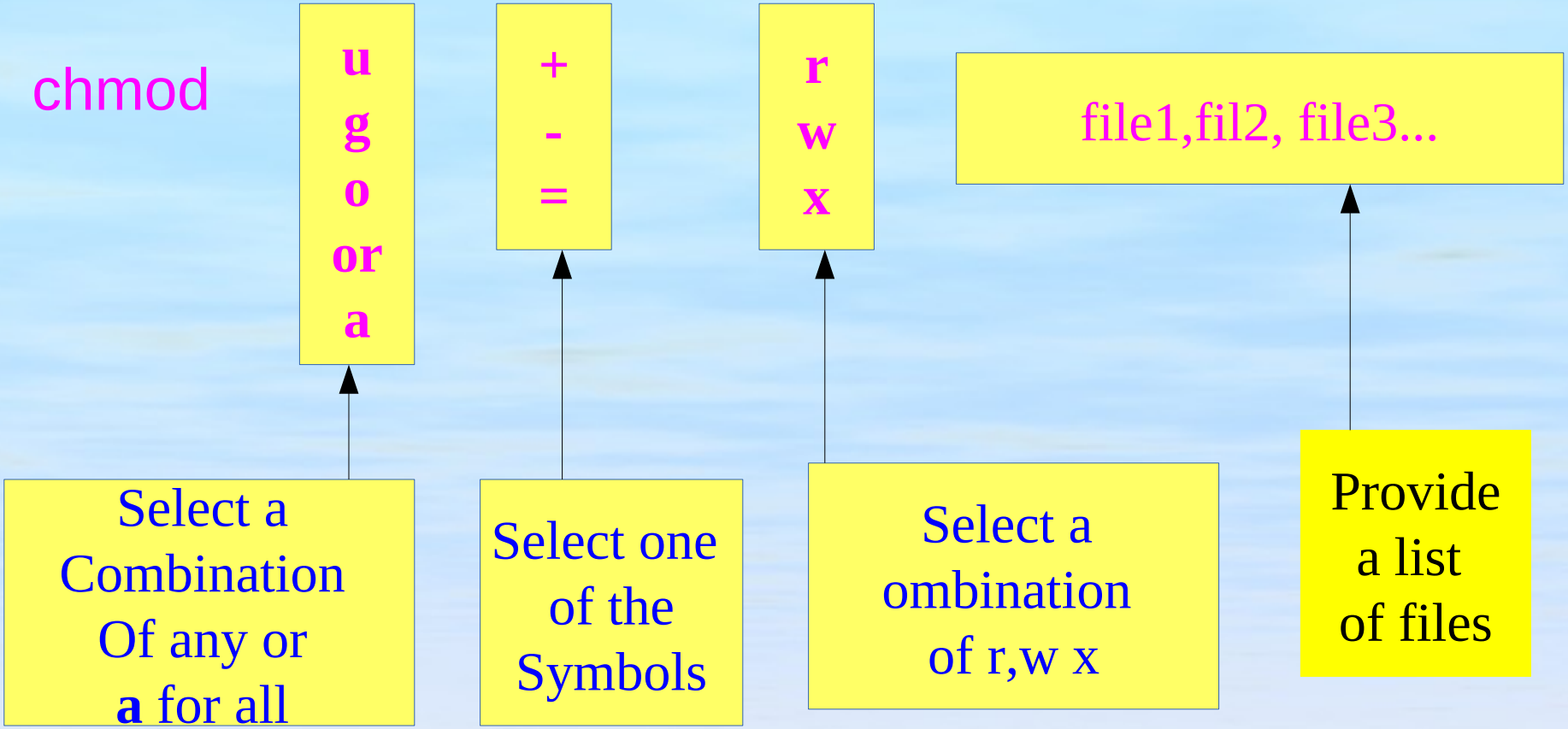
Access Control Mechanism

- Linux File system Security
- For each category the are a maximum of three access settings. rwx- **read, write and execute**
- The permissions are shown as a group of nine characters.

Example: rwxr-xr-x

Changing File Access Permissions

- The chmod Command
- Used to change a file access permission
-
- chmod



Changing File Access Permissions

- Only the owner of a file (or the superuser) can change the file's permissions

Examples: rwxr-xr-x

chmod	u+x	hello.txt
chmod	go-w	display.object
chmod	a-wx	cat.txt exam.txt
chmod	u=rw	open*
chmod	u=rwx,go=r	index

Changing File Access Permissions

- File's permissions in Octal
- Permissions can also be represented using octal notations
- Let

r	w	x
1	1	1

- In octal rwx is equivalent to $1X2^0 +1X2^1+1X2^2=7$
- In r-- is equivalent to $0+0+1X2^2=4$
- In r-x is equivalent to $1X2^0+1X2^2=4$
- In --x is equivalent to $1X2^0 +0+0=1$
- In rw- is equivalent to $0 +1X2^1+1X2^2=6$
-

Changing File Access Permissions

- File’s permissions in Octal
- Examples
- chmod 764 file2.txt is equivalent to
- chmod u=rwx,g=rw,o=rx file2.txt
-
- Exercise

1.

Changing File Access Permissions

- Setting Up a Workgroup Directory

1. Create the new group:

groupadd sales

2. Add the existing users to the group:e.g

usermod –a -G sales Jane

3. Create a directory for the group:

mkdir /home/sales

4. Set the ownership of the new directory:

chgrp sales /home/sales

5. Protect the directory from others:

chmod 770 /home/sales

Changing File Access Permissions

- Special file permissions
- 1.Linux provide three more bits in a file's 'access mode':
- The bits further provide some functionality,
- These includes:

  Sticky bit: shown as t as the 'other execute' permission

  Setgid:shown as s in the 'group execute' permission

  Setuid: shown as s in the 'user execute' permission

Changing File Access Permissions

Special file permissions

Sticky bit:

- On a file Originally this bit indicated that executable programs should stay in memory after they have terminated. This meaning is now obsolete
- On a directory You can only delete files if you own the file or you own the directory.
- Often used for communal directories such as /tmp

Setgid:shown as s in the 'group execute' permission

Setuid: shown as s in the 'user execute' permission

Changing File Access Permissions

Special file permissions

Setgid:

- On a file
  - If the file is executed, the effective group ID is set to the group of the file.
- On a directory
  - Files created in the directory belong to the directory's group and not to the primary group of the user

Changing File Access Permissions

Special file permissions

setuid:

- On a file
  - If the file is executed, the effective user ID is set to the owner of the file
  -
- On a directory
  - Not set

Changing File Access Permissions

Special file permissions

setuid:

- Some programs reserved for the root are also required by the users to carry out some tasks
- Example is the passwd command
  - Required by users to change their own password
  - To make it possible for that to happen the passwd file has the setuid bit set.
  - It grants the user different priviledges(root) for the duration of the command(Security Provision)

Changing File Access Permissions

Special file permissions

setuid:

- Note
- User's (encrypted) passwords are stored in the file /etc/shadow;
- Only root has write permission on this file
- However because of setuid bit setting on
  - Users can change their passwords using the command /usr/bin/passwd

Changing File Access Permissions

Special file permissions

setuid:

- [root@localhost ~]# ls -l /etc/shadow
- -rw-r-----. 1 root root 1528 Jun 8 23:35 /etc/shadow
- [root@localhost ~]# ls -l /usr/bin/passwd
- -rwsr-xr-x. 1 root root 30768 Feb 22 2012 /usr/bin/passwd
-

Notice the s symbol in place of x in user

In most cases such Files are indicated red

Changing File Access Permissions

Special file permissions

setuid:Security Precaution

- It is possible for an adversary to gain access to the root user workstation in his absence and put in an executable that has the setuid bit set.
- This is a security threat as the program can be invoked to cause damage while everybody thinks it was written by the administrator
-

Changing ownership with chown

- The chown command
- Used to change the ownership and group of a file
- Security Requirements

1. Only root can change a file's owner
2. Ordinary users can change a file's group only if they are members of both the original and the new group

chown owner.group file1 file1

Example

chown root.sales file1 file1

3. You can change just the owner:

chown root file1 file1

2. you can change just the group

chown .sales file1 file1

Users Account & File Security Files

- Files Used to Implement User and File Security

1. /etc/passwd: User account information.
2. /etc/shadow: Secure user account information.
3. /etc/group: Group account information.
4. /etc/gshadow: Secure group account information.
5. /etc/default/useradd: Default values for account creation.
6. /etc/skel/ Directory containing default files.
7. /etc/login.defs Shadow password suite configuration.

-