Moodle for Instructional Design & System Administration Training

Module 1: File Directory and Management (Linux Fundamentals)

1.1 Understanding the Linux File System (Theory)

Linux uses a **hierarchical directory structure** starting from the root /.

Key directories administrators should know:

- / – Root directory (everything starts here)
- /home – User home directories
- /etc – System configuration files (important for Moodle, Apache, PHP)
- /var – Variable data (logs, Moodle data, web content)
- /opt – Optional software packages
- /tmp – Temporary files

📌 *Tip:* Moodle files are commonly found in:

- Web root: /var/www/html/moodle
- Moodle data: /var/moodledata

1.2 Creating and Viewing Files and Directories

Create Directories

```
mkdir moodle
mkdir -p /var/moodledata/backups
```

- -p creates parent directories if they don't exist

Create Files

```
touch config.php
nano testfile.txt
```

View File Contents

```
cat testfile.txt
less testfile.txt
```

- less allows scrolling (recommended for large files)

View Directory Contents

```
ls
ls -l
ls -lah
```

- -l detailed view
- -a shows hidden files
- -h human-readable sizes

1.3 Copying, Moving, and Deleting Files

Copy Files and Directories

```
cp file1.txt file2.txt
cp -r moodle /backup/moodle
```

- -r is required for directories

Move or Rename Files

```
mv oldname.txt newname.txt
mv moodle /var/www/html/
```

Delete Files and Directories ⚠ (Use Carefully)

```
rm file1.txt
rm -r oldfolder
rm -rf moodle
```

- -f forces deletion without confirmation

📌 *Best Practice:* Avoid using rm -rf unless you are 100% sure.

1.4 Searching and Locating Files

Search by Name (find)

```
find /var/www -name "config.php"
find / -type d -name moodle
```

Fast File Lookup (locate)

```
updatedb
locate moodle
```

- Requires mlocate package

Search Inside Files (grep)

```
grep "www-data" /etc/apache2/apache2.conf
grep -R "moodle" /var/www/html/
```

1.5 Archiving and Compression

Create tar Archive

```
tar -cvf moodle_backup.tar /var/www/html/moodle
```

Compress with gzip

```
tar -czvf moodle_backup.tar.gz /var/www/html/moodle
```

Extract Archive

```
tar -xzvf moodle_backup.tar.gz
```

Zip Files

```
zip -r moodle.zip /var/www/html/moodle
unzip moodle.zip
```

📌 *Real Moodle Use Case:* Backing up Moodle files before upgrades.

Module 2: System Access Controls (Theory + Practice)

Learning Objectives

Participants will:

- Understand Linux user and group concepts
- Apply password policies
- Manage file and directory permissions securely

2.1 Users and Groups (Theory)

Linux is a **multi-user system**.

- Each user has a unique UID
- Users belong to one or more groups
- Permissions control who can read, write, or execute files

Key files:

- /etc/passwd – User accounts
- /etc/shadow – Encrypted passwords
- /etc/group – Group definitions

2.2 User Management (Practical)

Create a User

```
sudo useradd moodleadmin
sudo passwd moodleadmin
```

Create a Group

```
sudo groupadd moodleadmins
sudo usermod -aG moodleadmins moodleadmin
```

Switch User

```
su - moodleadmin
```

2.3 Password Policies (Theory + Practice)

Password policies define **rules that govern how user passwords are created, changed, and managed**. Their goal is to **protect systems from unauthorized access** by reducing the risk of weak, guessable, or reused passwords.

A. Password Policy Concepts (Theory)

1. Minimum Length

Minimum length specifies the **shortest allowed password**.

- Longer passwords are exponentially harder to crack
- Modern best practice recommends **12 characters or more**

🔐 Example:

- Weak: Admin123
- Stronger: Admin@Server2026!

Even a simple password becomes strong when it is long enough.

---

2. Password Complexity

Complexity ensures that passwords use a mix of different character types.

Common character classes:

- **Uppercase letters** → A–Z
- **Lowercase letters** → a–z
- **Numbers** → 0–9
- **Special symbols** → !@#$%^&*

Using multiple classes:

- Prevents dictionary attacks
- Increases entropy (randomness)

🔐 Example:

- Weak: passwordkenya
- Strong: Kenya@Sys2026

---

3. Expiry Control

Password expiry forces users to **change passwords periodically**.

Why this matters:

- Limits damage if a password is compromised
- Encourages regular credential hygiene

Typical policy:

- Change every **60–90 days**
- Warn users before expiration

---

4. Password Reuse Control

Prevents users from:

- Reusing old passwords
- Cycling between the same few passwords

This is enforced using:

- PAM history modules (pam_pwhistory.so)
- Minimum days before password change

B. Configure Password Aging Policy (Ubuntu)

Password aging policies are controlled in:

```
sudo nano /etc/login.defs
```

Key Settings Explained

```
PASS_MAX_DAYS   90
```

- Maximum number of days a password is valid
- User must change password after 90 days

```
PASS_MIN_DAYS   7
```

- Minimum days before password can be changed again
- Prevents users from quickly changing passwords to reuse old ones

```
PASS_WARN_AGE   14
```

- Number of days before expiration that the system warns the user

📌 Example user experience:

"Your password will expire in 14 days."

Apply Policy to Existing Users

Changes in login.defs affect **new users only**.

To apply to an existing user:

```
sudo chage -M 90 -m 7 -W 14 username
```

To verify:

```
sudo chage -l username
```

C. Enforce Strong Passwords (PAM)

Ubuntu uses **PAM (Pluggable Authentication Modules)** to enforce password quality.

Step 1: Install pwquality module

```
sudo apt install libpam-pwquality
```

Step 2: Edit password quality configuration

```
sudo nano /etc/security/pwquality.conf
```

Key Settings Explained

minlen = 12

- Password must be **at least 12 characters**
- Applies to all users (except root if configured)

minclass = 3

- Password must include **at least 3 character classes**
- Classes include:
  - uppercase
  - lowercase
  - digits
  - symbols

How pwquality Evaluates Passwords

Password	**Length**	**Classes**	**Result**
password123	11	2	❌ Reject
Admin2026	9	2	❌ Reject
Admin@2026	10	3	❌ Reject

Password	Length	Classes	Result
Admin@Kenet2026	15	4	✅ Accept

2.4 User Permissions (Theory)

Linux permissions are based on:

- **Owner (u)**
- **Group (g)**
- **Others (o)**

Permission types:

- r – Read
- w – Write
- x – Execute

Example:

-rwxr-x---

2.5 File Permissions and Linux Editors

File Permissions (Theory)

Linux uses file permissions to control access to files and directories. Each file or directory has three permission sets:
- User (Owner)
- Group
- Others

Permission Types:
Read (r): View file contents or list directory contents
Write (w): Modify file contents or create/delete files in a directory
Execute (x): Run files or access directories

Symbolic Representation Example:
-rwxr-xr--
User: rwx | Group: r-x | Others: r--

Numeric Representation:
r = 4, w = 2, x = 1
Example:
rwx = 7
r-x = 5
r-- = 4
So rwxr-xr-- = 754

Common Commands:

```
ls -l
chmod 755 filename
chmod u+rwx,g+rx,o-rwx filename
chown user:group filename
```

Linux Text Editors (Theory)

Text editors are essential for system administration tasks such as editing configuration files.

vi/vim:
- Modal editor (command and insert modes)
- Available on almost all Linux systems

- Ideal for servers and SSH access

nano:
- Beginner-friendly
- Uses visible keyboard shortcuts
- Good for quick edits

emacs:
- Highly extensible editor
- Can function as a full IDE
- Suitable for advanced users

gedit:
- Graphical text editor
- Easy to use
- Suitable for desktop environments

Class Exercises

Exercise 1

- Create a directory called moodle_lab
- Create 3 files inside it
- Change permissions so only the owner can edit

Exercise 2

- Create a user trainer1
- Assign them to group moodleadmins

Exercise 3

- Archive the moodle_lab directory
- Compress it and extract it in /tmp

Exercise 4:

1Create a file permissions_lab.txt

2. Set permissions to 644

3. Verify permissions using ls -l

Exercise 5:

1. Create a file editor_lab.txt using nano
2. Edit the same file using vi
3. Add a line using emacs
4. Review the file using gedit

Exercise 6: Password policy

Exercise: Configure Password Policy

1. Edit password aging policy:

```
sudo nano /etc/login.defs
```

2. Set:

```
PASS_MAX_DAYS   90

PASS_MIN_DAYS   7

PASS_WARN_AGE   14
```

3. Install and configure pwquality:

```
sudo apt install libpam-pwquality

sudo nano /etc/security/pwquality.conf
```

4. Set:

minlen = 12

minclass = 3

Exercise: Test the Policy

1. Create a test user:

sudo useradd testuser

sudo passwd testuser

2. Try setting:

- Weak password → should fail

Strong password → should succeed

Exercise: Verify Password Aging

sudo chage -l testuser

Lab Answer Key (Step-by-Step Solutions)

Exercise 1: File and Permission Management

Task:

- Create a directory called moodle_lab
- Create 3 files inside it
- Change permissions so only the owner can edit

Step-by-Step Solution:

```
mkdir moodle_lab
cd moodle_lab
touch file1.txt file2.txt file3.txt
```

Check files:

```
ls -l
```

Set permissions so:

- Owner: read & write
- Group/Others: read-only (or no access)

```
chmod 600 file1.txt file2.txt file3.txt
```

Verify:

```
ls -l
```

Expected output pattern:

```
-rw------- 1 user user file1.txt
```

Exercise 2: User, Group, and Access Control

Task:

- Create a user trainer1
- Assign them to group moodleadmins

Step-by-Step Solution:

Create user:

```
sudo useradd trainer1
sudo passwd trainer1
```

Add user to existing group:

```
sudo usermod -aG moodleadmins trainer1
```

Confirm group membership:

```
groups trainer1
```

Exercise 3: Archiving and Extraction

Task:

- Archive the moodle_lab directory
- Compress it
- Extract it in /tmp

Step-by-Step Solution:

From parent directory:

```
tar -czvf moodle_lab.tar.gz moodle_lab
```

Confirm archive:

```
ls -lh moodle_lab.tar.gz
```

Move archive to /tmp:

```
mv moodle_lab.tar.gz /tmp
cd /tmp
```

Extract archive:

```
tar -xzvf moodle_lab.tar.gz
```

Verify extraction:

```
ls -l /tmp/moodle_lab
```

Exercise 4: File Permissions

Commands:

```
touch permissions_lab.txt
chmod 644 permissions_lab.txt
ls -l permissions_lab.txt
```

Expected Output:

```
-rw-r--r-- 1 user user permissions_lab.txt
```

Exercise 5: Linux Text Editors

```
nano editor_lab.txt
```

(Add initial content, save and exit)

```
vi editor_lab.txt
```

(Add a second line, save and exit)

```
emacs editor_lab.txt
```
(Add a third line, save and exit)

```
gedit editor_lab.txt
```
(Review and confirm content)

Expected Outcome:
File contains three lines edited using different editors.

Exercise 6: Password policy

Exercise 1: Configure Password Policy

2. Edit password aging policy:

```
sudo nano /etc/login.defs
```

3. Set:

```
PASS_MAX_DAYS   90

PASS_MIN_DAYS   7

PASS_WARN_AGE   14
```

4. Install and configure pwquality:

```
sudo apt install libpam-pwquality

sudo nano /etc/security/pwquality.conf
```

5. Set:

```
minlen = 12

minclass = 3
```

Exercise 2: Test the Policy

2. Create a test user:

```
sudo useradd testuser
sudo passwd testuser
```

3. Try setting:

- Weak password → should fail
- Strong password → should succeed

Exercise 3: Verify Password Aging

```
sudo chage -l testuser
```
