**Implementing Software Design Principles Using Linux Utilities and Shell Scripting**

**1. Introduction**

This report details the implementation of software design principles in a Linux-based automation system. The chosen problem statement is an **Automated File Organization System**, which sorts and organizes files in a specified directory based on their type, reducing clutter and improving accessibility.

**2. Software Design Principles Applied**

**a) Abstraction**

The system is divided into key functionalities:

* **File scanning**: Identifies files in a target directory.
* **Sorting mechanism**: Moves files into categorized subdirectories (e.g., Documents, Images, Videos, Archives, Executables).
* **Logging**: Maintains logs of file movements and errors.
* **Error handling**: Ensures fault tolerance with exception handling.
* **Scheduling**: Automates execution using cron jobs.

**b) Encapsulation**

Related tasks are grouped into separate functions and scripts:

* file\_scanner.sh: Scans the directory and categorizes files.
* file\_mover.sh: Moves files to appropriate folders.
* setup.sh: Configures the system for execution.
* logger.sh: Logs all activities and errors.

**c) Modularity**

Each script operates independently, allowing easy maintenance and reuse in different workflows.

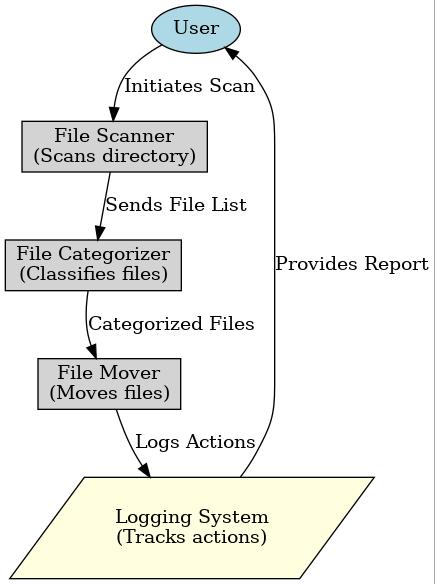
**d) Cohesion & Coupling**

* **High Cohesion**: Each script performs a well-defined function.
* **Low Coupling**: Minimal dependencies among scripts, allowing independent modifications.

**3. Software Architecture Documentation**

**a) Data Flow Diagram (DFD)**

|  |
| --- |
|  |



**b) Class Diagram (for Object-Oriented Implementation in Python)**

If the system were implemented in Python, it could use classes as follows:

|  |
| --- |
| FileOrganizer |
| - target\_dir  - log\_file |
| + scan\_files()  + categorize()  + move\_files()  + log\_action() |

**c) Deployment Design**

**Installation Steps**

1. Clone the repository: git clone https://github.com/user/repo.git
2. Set permissions: chmod +x \*.sh
3. Configure the cron job: crontab -e and add \*/10 \* \* \* \* /path/to/file\_scanner.sh
4. Verify logs using cat $HOME/file\_organizer.log

**4. Shell Script Implementation**

**a) file\_scanner.sh**

#!/bin/bash

TARGET\_DIR="$HOME/Downloads"

DOC\_DIR="$HOME/Documents"

IMG\_DIR="$HOME/Pictures"

VID\_DIR="$HOME/Videos"

ARCHIVE\_DIR="$HOME/Archives"

EXE\_DIR="$HOME/Executables"

LOG\_FILE="$HOME/file\_organizer.log"

log\_activity() {

echo "[$(date)] $1" >> "$LOG\_FILE"

}

organize\_files() {

mkdir -p "$DOC\_DIR" "$IMG\_DIR" "$VID\_DIR" "$ARCHIVE\_DIR" "$EXE\_DIR"

mv $TARGET\_DIR/\*.pdf $DOC\_DIR/ 2>/dev/null && log\_activity "Moved PDFs to Documents."

mv $TARGET\_DIR/\*.jpg $IMG\_DIR/ 2>/dev/null && log\_activity "Moved Images to Pictures."

mv $TARGET\_DIR/\*.mp4 $VID\_DIR/ 2>/dev/null && log\_activity "Moved Videos to Videos."

mv $TARGET\_DIR/\*.zip $ARCHIVE\_DIR/ 2>/dev/null && log\_activity "Moved Archives to Archive Folder."

mv $TARGET\_DIR/\*.sh $EXE\_DIR/ 2>/dev/null && log\_activity "Moved Executables to Executable Folder."

}

organize\_files

**b) Logging and Error Handling**

* Uses log\_activity function to track all file movements.
* Redirects errors to /dev/null to prevent script termination.
* Uses trap to handle script interruptions and unexpected failures.

**5. Software Configuration Management (SCM)**

**a) Git Version Control**

* Repository: https://github.com/user/repo
* Branching strategy: main, feature-scanner, feature-mover, feature-logger
* Versioning: v1.0, v1.1, v1.2 with CHANGELOG.md

**6. Performance Testing and Risk Management**

**a) Performance Evaluation**

* Used time ./file\_scanner.sh to measure execution time.
* Used top, htop, and vmstat to analyze system performance.

**b) Risk Identification and Mitigation**

* **Technical Risks**: Incorrect permissions → Solution: chmod +x \*.sh
* **Operational Risks**: Files moved incorrectly → Solution: Test in a sample directory before deployment.
* **Security Risks**: Unauthorized script execution → Solution: Restrict execution permissions to the owner.

**7. Conclusion**

This project successfully applies software design principles in a Linux automation task. The modular, encapsulated, and well-documented scripts improve maintainability and performance.