```
# User Audit Script - Revision Notes

## Introduction
In this class, we learned how to write a bash script that could audi

### Key Concepts Covered:
1. **Script Requirements and Use Cases**
2. **Error Handling in Bash with Traps**
3. **Functions and Their Usage in Bash**
4. **Shell Scripting Commands and Techniques**

---

## Bash Script Design

### Script Overview
The script is designed to perform the following functions:
- Accept a username or "all" as an argument.
- Generate a report for individual users or all users based on the i
- Provide UID, home directory, shell, and last login information for
- Handle various error scenarios such as missing arguments and non-e
- Output results to standard output or an optional file specified by

### Key Bash Commands Used
- **`getent`:** Used to fetch information about users from databases
- **`id`:** Retrieves user and group IDs.
- **`lastlog`:** Provides the last login details.
- **`set -euo pipefail`:** A strategy to handle errors and undefined

### Script Components

#### Global Setup
1. **Shebang and Initial Setup:**
   ```bash
   #!/bin/bash
   set -euo pipefail
   trap 'echo "Error occurred at line $LINENO"; exit 1' ERR
```

2. **Variables:**
   - `USER_INPUT` for storing the username or "all".
   - `OUTPUT_FILE` to store the file path for the output, defaulting to `stdout` if not provided .
```

1. **`print_help`:** Displays usage information.
2. **`get_user_info`:** Retrieves and prints information for a specific user, including checking if the user exists using the `id` command.
3. **`get_report`:** Iterates over all users when "all" is specified, making use of a `while` loop to process entries in `/etc/passwd`.

Best Practices

- **Error Handling with `trap`:** Critical for capturing unexpected errors and providing meaningful messages.
- **Function Usage:** Encapsulates repetitive tasks and aids in maintaining clean and modular code【4:10†transcript.txt】【4:13†transcript.txt】.

Example Usage

- To audit a specific user: `./script.sh username`
- To audit all users: `./script.sh all`
- To direct output to a file: `./script.sh username|all output_filename`

Additional Notes

- The script includes mechanisms to handle non-existent users and missing command scenarios using `trap` and conditional checks【4:18†transcript.txt】.
- Comments are included within the script to document the use of built-in variables like `$0` and `$LINENO`【4:14†transcript.txt】.

Conclusion

This script demonstrates fundamental shell scripting practices through real-world applications like user audit and information retrieval. Functions, error handling, and input validation were central to this class, providing a robust foundation for writing efficient and reliable scripts in bash. Remember, practice and iteration are key to mastering shell scripting.

Scaler Companion beta