Revision Notes: Building a Command Line Utility with Bash

Introduction

In this class, we focused on creating a Bash script that behaves as a command-line utility tool for system monitoring and reporting. The main objective was to enable users, particularly developers, to easily fetch system resource data such as CPU, memory, and disk usage using custom commands. This script serves as a DevOps tool to simplify operations typically associated with system health checks【4:0†source】.

Key Concepts

Command-Line Utility in Bash

- **Purpose**: To develop a script (`syslog.sh`) that accepts flags to output CPU, memory, and disk usage data, which can be easily understood and manipulated by developers【4:4†source】.
- **Utility Script**: This script functions as an alias to custom commands which replace the need for remembering complex Linux commands【4:17†source】.

Components of the Script

1. **Script Initialization**
   - Using Shebang (`#!/bin/bash`) to specify the interpreter【4:6†source】.
   - Declaring environment variables for CPU, memory, disk, and log enabled with initial default values【4:6†source】.
2. **Accepting Flags**
   - Flags signify what data (CPU, MEMORY, DISK) to capture, and also include a `--no-log` option to dictate logging behavior【4:18†source】【4:16†source】.

3. **Checking and Processing Flags**
    - The script uses a for-loop to iterate through arguments and a switch case to evaluate which flags are set【4:16†source】.
    - Appropriate steps are taken based on which flags are toggled, modifying the behavior of data capture【4:6†source】【4:13†source】.

Features Included in the Script

- **Timestamp and Logging**
    - Each log entry includes a timestamp at the beginning for record-keeping【4:8†source】.
    - Logs are exported to an environment variable, allowing for flexible redirection【4:12†source】.
- **Data Capture Commands**
    - **CPU Logic**: Utilizes the `top` command in non-interactive mode (`-b` flag) for a static snapshot【4:9†source】.
    - **Memory Logic**: Uses `free -m` to display memory usage【4:10†source】.
    - **Disk Logic**: Employs `df -h` to show disk space usage in a human-readable format【4:11†source】.
- **Error Handling**
    - Implements handling of invalid flags with user messages【4:3†source】.
    - Script writes data both to a log file and to the terminal, ensuring visibility of operation【4:7†source】【4:5†source】.

Advanced Topics

- **Alias and Source Commands**
    - An alias (`sysrep`) to the script is created for simplified command execution, using `alias` and `source` commands【4:0†source】.
- **Skeleton of the Script**
    - Emphasizes the importance of a well-structured skeleton to simplify scripting and debugging【4:11†source】【4:5†source】.

The script created in this class presents a scalable and efficient tool for system monitoring, enhancing the utility of command-line tools in a developer's toolkit. The design leverages Bash's capabilities to interact with system resources directly, offering a clear and customizable interface for users【4:4†source】【4:18†source】.

For more in-depth learning, try experimenting with different flag combinations and expanding script functionalities to cover more system metrics or integrate it further into DevOps pipelines!