Scaler Companion beta

Software Engineering - Revision Notes on Process Management and Scheduling

These notes are compiled from a live class session and are intended to aid in the revision of process management and scheduling concepts. The focus is on commands, their usage, and the underlying concepts.

Background Processes

Definition

Background processes are those that run without user interaction. They are initiated in the background and do not clog the terminal with output that requires immediate attention【4:5†transcript.txt】.

Contextual Analogy

Imagine you are having a conversation (foreground process) while your brain simultaneously processes environmental noises (background processes) without conscious effort【4:17†transcript.txt】.

Commands

- **Use of '&'**: Appending an `&` at the end of a command in Linux runs the process in the background, allowing other tasks to continue without waiting for the current one to finish.

  ```
  sleep 60 &
  ```

  This command will initiate a sleep process for 60 seconds in the background【4:17†transcript.txt】.
- **Retrieving Process ID**: Use `echo $!` to obtain the PID of the last background process【4:3†transcript.txt】.

Scaler Companion beta

Definition

Cron jobs are scheduled tasks that run at specified times or intervals. This is essential for automating repetitive tasks【4:7†transcript.txt】.

Cron Syntax

Cron expressions comprise five fields indicating the minute, hour, day of the month, month, and day of the week:

```
*    *    *    *    *  command-to-be-executed
-    -    -    -    -
|    |    |    |    |
|    |    |    |    +----- day of the week (0 - 6) (Sunday to Saturd
|    |    |    +---------- month (1 - 12)
|    |    +--------------- day of the month (1 - 31)
|    +-------------------- hour (0 - 23)
+------------------------- minute (0 - 59)
```

For instance, `0 0 * * 0` denotes a task running every Sunday at midnight【4:19†transcript.txt】.

Example

To run a shell script every minute:

```
* * * * * /path/to/script
```

Editing and Listing Crontab

- Edit crontab using `crontab -e`.
- List current cron jobs with `crontab -l`【4:19†transcript.txt】【4:19†transcript.txt】.

Process Prioritization

- **Nice values** range from -20 (highest priority) to 19 (lowest priority). The `nice` command adjusts the priority of a running process.
- **Renice** is used to change the priority of an existing process 【4:4†transcript.txt】.

Command Examples

- Adjust priority on start:

  ```
  nice -n 10 your_command
  ```

- Adjust priority of a running process:

  ```
  renice -n 5 -p 1234
  ```

Concept Analogy

Processes with lower nice values receive more CPU time. In a way, less "nice" processes (lower nice value) are favored by the CPU 【4:10†transcript.txt】.

System Monitoring and Commands

Process IDs and Trees

- **PSTree**: Displays the hierarchical view of processes, showing parent-child relationships.

```
pstree
```

This command can illustrate how processes are linked together, often useful for diagnosing process issues【4:18†transcript.txt】.

Using `htop` for Monitoring

```
htop
```

VMStat offers insights into system processes, memory, and CPU activity.

```
vmstat
```

Iostat monitors system input/output device load.

```
iostat
```

Both commands are integral for identifying bottlenecks and system load【4:13†transcript.txt】【4:13†transcript.txt】.

These notes provide a robust foundation for understanding and utilizing process management and scheduling in your endeavors with Linux and system administration. For practical understanding, engage actively with these commands in a test environment.