Advanced shell scripting Techniques

Starts at 9:05 pm

Agenda

1. Program
2. Associative Arrays
3. Process Control
4. traps
5. trons
6. set -x ✓
7. Process substitution.

→ Program

Given a log file with each line containing a timestamp, process ID (PID), log level (INFO, WARNING, ERROR), and log message, write a one-liner using `grep` and `awk` to filter all `ERROR` logs, showing the timestamp, PID, and log message (excluding the log level).

Data set

2024-10-15 08:23:45 [1234] INFO User logged in

2024-10-15 08:25:13 [5678] ERROR Failed to connect to database

2024-10-15 08:26:02 [9101] WARNING Disk space running low

2024-10-15 08:27:59 [1121] ERROR Timeout while reaching service

2024-10-15 08:30:12 [3141] INFO User logged out

ERROR → X

date time pid message

grep 'ERROR' data | awk '{printf ", $1, $2, $3; for (i=5; i<=NF; i++) printf "%s printf "%s", $i; print ""}'

2024-10-15 08:25:13 [5678] Failed to connect to database

2024-10-15 08:27:59 [1121] Timeout while reaching service

→ Associative Arrays

array [key] = "value"

① declare an Associative array.

declare -A my_array

② Assigning values.

my_array [key] = "value"

→ my_array [fruit 1] = "apple"
fruit 2 Banana

③ Accessing values.

echo ${my_array[fruit1]}

(4) check if key exists or not

-if [[-v]]

Program for associative arrays.

cat associative_arrays.sh

```
#!/bin/bash

declare -A my_array

my_array[fruit1]="apple"
my_array[fruit2]="banana"

echo "This is Fruit1 ${my_array[fruit1]}"

if [[ -v my_array[fruit1] ]]; then
    echo "Key fruit1 exists"
else
    echo "Key fruit1 doesn't exist"
```

```
fi

echo "These are all the keys ${!my_array[@]}"
echo "These are the values ${my_array[@]}"

for i in "${!my_array[@]}"; do
        echo "Key = $i and value = ${my_array[$i]}"
done
echo "Number of elements: ${#my_array[@]}"

unset my_array[fruit1]

echo "Removing fruit1"

echo "Trying to print fruit1 ${my_array[fruit1]}"

echo "All values are ${my_array[@]}"

echo "Number of elements: ${#my_array[@]}"
ubuntu@ip-172-31-41-136:~$ ./associative_arrays.sh
This is Fruit1 apple
Key fruit1 exists
```

These are all the keys fruit2 fruit1

These are the values banana apple

Key = fruit2 and value = banana

Key = fruit1 and value = apple

Number of elements: 2

Removing fruit1

Trying to print fruit1

All values are banana

Number of elements: 1

Output

my -array [hostname] = ''

→ Process Control
  → Background jobs bg
  → foreground jobs fg

[+1] → job id

fg → Bring a background

process to foreground.

Q How to move a foreground job to background.

SIGSTOP

① use ctrl+z

SIGSTOP

↓

ctrl+z

② bg %1 → Run job id 1 in background.

→ set -x used for debugging.

set -x

code in debug mode

set +x

bash -x script → To Run entire script in Debug Mode.

Break → 10:25 pm.

→ Traps

SIGINT → ctrl+c

SIGSTOP → ctrl+z

SIGKILL → kill -9

SIGTERM → kill -15

traps 'rm' 'ctrl+c'

making directory. operations

I

inside the directory

II

Remove the directory

III

Try this out.

ctrl+c

traps

action → Remove the directory.

trap 'command' SIGNAL

cat traps.sh

```
#!/bin/bash

# Define a trap for SIGINT (Ctrl+C)
trap 'echo "You pressed Ctrl+C! Exiting safely..."; exit' SIGINT

echo "Press Ctrl+C to trigger the trap."

# Simulate a long-running task
for i in {1..10}; do
    echo "Iteration $i: Running..."
    sleep 2  # Simulates some work being done
done

echo "Task completed!"
```

output

```
./traps.sh
Press Ctrl+C to trigger the trap.
Iteration 1: Running...
Iteration 2: Running...
Iteration 3: Running...
Iteration 4: Running...
```

^CYou pressed Ctrl+C! Exiting safely...

To disable a trap.

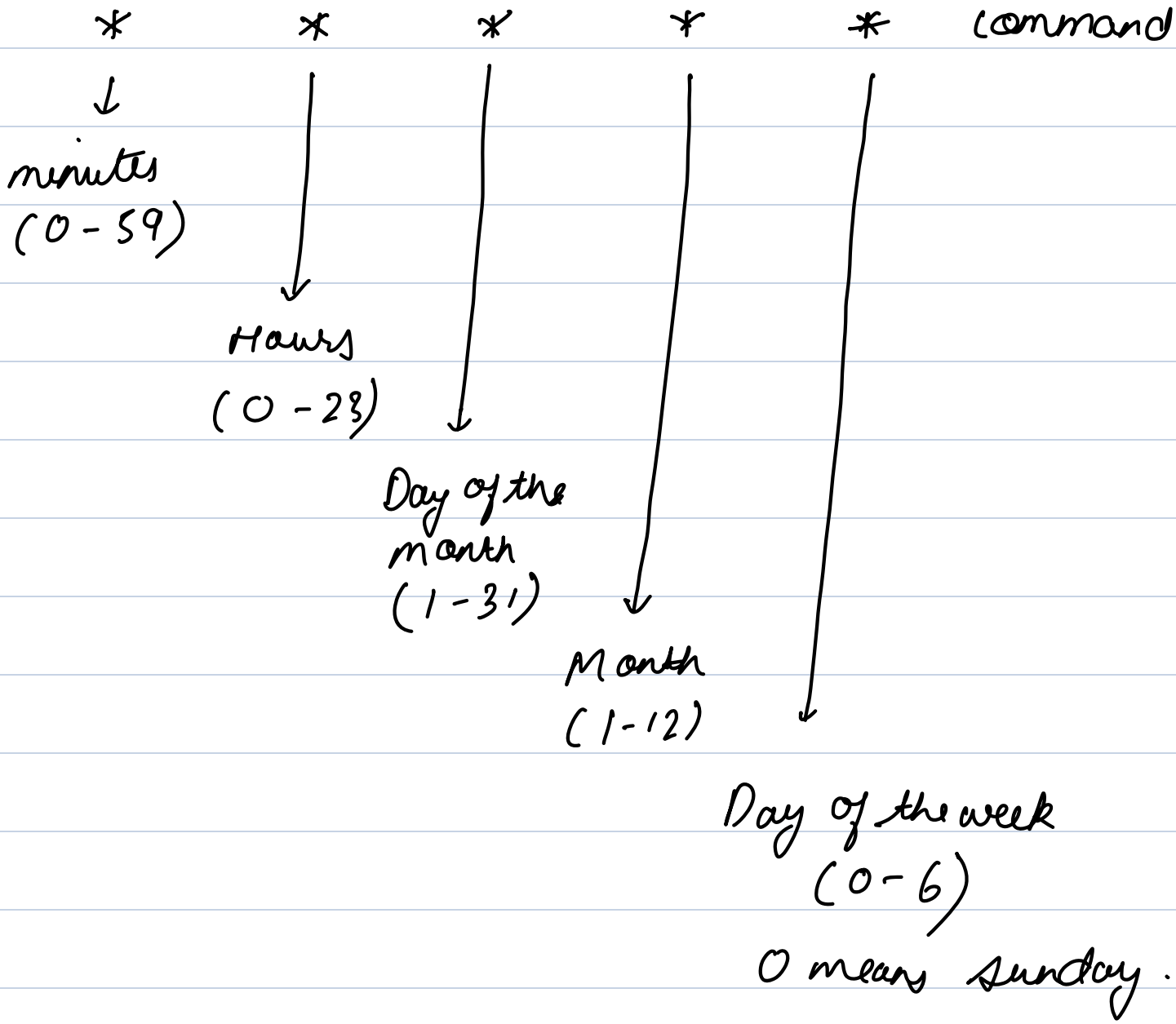
trap - SIGNAL

cat reset_trap.sh

```
#!/bin/bash

trap 'echo "Caught SIGINT! Exiting..."; exit' SIGINT

echo "Trap is active. Press Ctrl+C now."

sleep 5

trap - SIGINT

echo "Trap is removed. Press Ctrl+C now, and the script will exit normally."

sleep 10

echo "Finished."
```

→ Cron Jobs.

Time Based scheduler in Linux



→ Schedule a job at 3:30 am.

30 3 * * * /path/to/script

→ run every minute.

* * * * * /path

→ run every 5 minutes

*/5 * * * * /path

→ Run at reboot

@reboot /path

→ Run daily.

@daily /path

↓

0 0 * * * /path

crontab -e for editing

crontab -l for listing.

→ Process Substitution.

cat ?

cat file

any command

treated as a file

cat <("command")