SPY Script Documentation

Abraham Reines

March 19, 2024

1 Script Requirements

2 Usage

The script is invoked with the following syntax:

```
./spy.sh [-t tseconds] [-n count]
```

where `-t tseconds` specifies the time interval between each scan, and `-n count` specifies the number of times the scan is performed.

3 Error Checking

The script includes error checking for invalid options, missing argument values, and non-numeric input for time intervals and count.

4 Handling Interrupts

Interrupts, such as Ctrl-C, are gracefully handled by the script to provide a user-friendly exit message.

5 Script Listing

```
#!/bin/bash

# Author: Abraham Reines
# Date: 2024-03-04
# Modified: Tue Mar 19 10:25:45 PDT 2024

# how_do_we_use?: spy [list of patterns] [-t secs] [-n count]
# This script is used to track changes in processes based on patterns.

# preset values
interval=1
count=5
patterns=()
momentary="spy_temp.txt"
prev_momentary="spy_prev_temp.txt"

# display how_do_we_use?
how_do_we_use?() {
    echo "how_do_we_use?: $0 [list of patterns] [-t seconds] [-n count]"
    exit 1
}

# handle interrupts (Ctrl-C); cleaning up temporary files
cleanup_time!() {
    echo
    echo "Cleaning up temporary files..."
```

```
    rm -f "$momentary" "$prev_momentary"
    exit
}

trap cleanup_time! INT

# parsing
while [[ "$#" -gt 0 ]]; do
    case $1 in
        -t) interval="$2"; shift ;;
        -n) count="$2"; shift ;;
        -*) echo "Unknown option: $1"; how_do_we_use? ;;
        *) patterns+=("$1") ;;
    esac
    shift
done

# nested main loop
for ((i=0; i<count; i++)); do
    whats_going_on=$(ps -eo user,pid,args | grep -v grep | grep -v $0)
    if [ ${#patterns[@]} -gt 0 ]; then
        which_are_filtered=$(echo "$whats_going_on" | grep -E "$(IFS=\|;
        ↪ echo "${patterns[*]}")")
    else
        which_are_filtered="$whats_going_on"
    fi

    if [ -f "$momentary" ]; then
        # setup comparison
        mv "$momentary" "$prev_momentary"
    fi
    echo "$which_are_filtered" > "$momentary"

    # comparison without substitution
    if [ -f "$prev_momentary" ]; then
        echo "$(date):"
        beginning=$(comm -13 "$prev_momentary" "$momentary")
        ending=$(comm -23 "$prev_momentary" "$momentary")
        if [ ! -z "$beginning" ]; then
            echo "started:"
            echo "$beginning"
        fi
        if [ ! -z "$ending" ]; then
            echo "ended:"
            echo "$ending"
        fi
        if [ -z "$beginning" ] && [ -z "$ending" ]; then
            echo "No process changes."
        fi
    else
        echo "$(date):"
        echo "Initializing scan. Monitoring beginning."
    fi

    sleep "$interval"
done

# cleanup_time! on exit
rm -f "$momentary" "$prev_momentary"
```

6 Sample Output Listing

For the first 10 lines:

```
Tue Mar 19 01:56:25 PM EDT 2024:
Initial scan. Monitoring started.
Tue Mar 19 01:56:27 PM EDT 2024:
Started:
root         759 [kworker/11:1-mm_percpu_wq]
root      102213 [kworker/89:0-events]
root      139986 [kworker/57:0-mm_percpu_wq]
root      151270 [kworker/65:2-events]
root      191674 [kworker/53:2-events]
root      201935 [kworker/107:2-mm_percpu_wq]
```

For the last 10 lines:

```
root     1339271 [kworker/u226:10-flush-253:0]
colem2cr 1339493 sleep 180
davis7aj 1339602 sleep 180
pegramay 1339618 sleep 180
root     1339742 [kworker/u226:11+rpciod]
root     1339743 [kworker/u226:13-flush-253:0]
davis7aj 1339745 sleep 180
davis7aj 1339758 sleep 180
hargroze 1339927 sleep 180
reinesaj 1340256 ps -eo user,pid,args
```

Full output too long to display in this PDF. The output is intended to be submitted as a .txt file alogn with `psmonitor.sh`

7 Executing the Script

To execute the script, first ensure it has the appropriate permissions set:

```
chmod +x spy.sh
```

Then run the script by providing the desired arguments for time interval and count, e.g.:

```
./spy.sh -t 1 -n 10 > spy_output.txt
```

or use the default values by not providing any arguments.

References

1. Kili, A. (n.d.). *30 Useful 'ps Command' Examples for Linux Process Monitoring.* Tecmint. Retrieved from `https://www.tecmint.com/ps-command-examples-for-linux-process-monitoring/`
2. Bytexd. (n.d.). *Linux Process Monitoring Using the ps, pstree, top Commands.* Retrieved from `https://bytexd.com/linux-process-monitoring/`
3. Upadhyay, K. (2018, October 6). *How to Monitor and Manage Linux Processes.* Open Source For You. Retrieved from `https://www.opensourceforu.com/2018/10/how-to-monitor-and-manage-linux-processes/`

Academic Integrity Pledge

"This work complies with the JMU honor code. I did not give or receive unauthorized help on this assignment."