Simulated UPS Script Setup

Overview

This guide demonstrates how to simulate a UPS using the NUT (Network UPS Tools) suite. The steps include installing NUT, configuring it to simulate a UPS, writing a monitoring script in Python, scheduling it with cron, and managing the logs with logrotate.

Step-by-Step Instructions

1. Install NUT

Run the following commands to update your system and install the NUT package:

```
apt update
apt install nut
```

2. Configure NUT

Edit configuration files as follows:
/etc/nut/nut.conf
Set the mode to netserver:

```
MODE=netserver
```

/etc/nut/ups.conf
Define a dummy UPS for simulation:

```
[fakeups]
driver = dummy-ups
port = /dev/null
desc = "UPS simulation"
```

/etc/nut/upsd.conf
Allow connections from local and master IP addresses:

```
LISTEN 0.0.0.0 3493
```

/etc/nut/upsd.users
Define a user for monitoring:

```
[upsmon]
password = mipass #choose your own
upsmon master
```

/etc/nut/upsmon.conf
Specify UPS monitoring configuration:

```
MONITOR fakeups@localhost 1 upsmon mipass master
SHUTDOWNCMD "/sbin/shutdown -h now"
```

3. Activate NUT Services

Enable and restart the NUT services:

```
systemctl enable nut-server
systemctl enable nut-monitor
systemctl restart nut-server
systemctl restart nut-monitor
```

4. Manually Start Dummy UPS Driver

For simulated UPS operation, manually launch the driver:

```
 /lib/nut/dummy-ups -a fakeups
```

Where:

- `-a fakeups`: Uses the `[fakeups]` configuration from `/etc/nut/ups.conf`

  Verify operation in a new terminal:

```
/usr/sbin/upsd
upsc fakeups
```

Expected status:

```
Init SSL without certificate database
device.mfr: Dummy Manufacturer
device.model: Dummy UPS
device.type: ups
driver.name: dummy-ups
driver.parameter.mode: dummy
driver.parameter.pollinterval: 2
driver.parameter.port: /dev/null
driver.parameter.synchronous: auto
driver.version: 2.8.0
driver.version.internal: 0.15
ups.mfr: Dummy Manufacturer
ups.model: Dummy UPS
ups.status: OL
```

Also, try

```
upsc fakeups@192.168.0.147
```

In your slave server, the same output as earlier should be obtained. If everything worked, create a script that will run the manual start of the driver on reboot using cron.

First, create the script:

```
#!/bin/bash
/lib/nut/dummy-ups -a fakeups &
sleep 2
/usr/sbin/upsd &
```

Make it executable:

```
chmod +x /usr/local/bin/start-nut-sim.sh
```

Add a Cron job:

```
crontab -e
```

Add line:

```
@reboot /usr/local/bin/start-nut-sim.sh
```

5. The Python Monitoring Script

This script checks if the master device is online. If unreachable, it assumes a power outage and initiates a shutdown.

```
#!/usr/bin/env python3

import subprocess
import time
import logging
import sys

# Configuration
IP_CHECK = "192.168.0.1" # IP to check power state
TIME_TO_CHECK = 60 # Seconds between checks
LOG_FILE = "/root/crl_ups/logs/ups_logs.log"

# Setup logging
logging.basicConfig(
    filename=LOG_FILE,
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s"
)

def do_ping(ip):
    try:
        subprocess.run(['ping', '-c', '1', '-W', '2', ip],
            stdout=subprocess.DEVNULL,
            stderr=subprocess.DEVNULL,
            check=True)
        return True
    except subprocess.CalledProcessError:
        return False

def activate_shutdown_ups():
    logging.warning("Blackout detected, executing 'upsmon -c fsd'")
    try:
        subprocess.run(["upsmon", "-c", "fsd"], check=True)
    except Exception as e:
        logging.error(f"Error while executing 'upsmon -c fsd': {e}")
        sys.exit(1)

def main():
    logging.info("Starting check...")
    if not do_ping(IP_CHECK):
        logging.warning(f"First ping failed to {IP_CHECK}. Waiting {TIME_TO_CHECK} seconds...")
        time.sleep(TIME_TO_CHECK)
        if not do_ping(IP_CHECK):
            activate_shutdown_ups()
        else:
            logging.info("Second ping successful, everything OK.")
    else:
        logging.info("Ping successful, everything OK.")

if __name__ == "__main__":
    main()
```

6. Make the Script Executable

Run the following command:

```
chmod +x /root/crl_ups/simulated_ups.py
```

7. Add a Cron Job

This cron job runs the script every 2 minutes:

```
sudo crontab -e
```

Add line:

```
*/2 * * * * /usr/bin/python3 /root/crl_ups/simulated_ups.py
```

To find the path to Python 3:

```
which python3
```

8. Configure Log Rotation

Prevent logs from consuming too much disk space:

```
vim /etc/logrotate.d/simulated_ups
```

Insert the following content:

```
/var/log/ups_logs.log {
    weekly
    rotate 4
    compress
    missingok
    notifempty
    create 640 root adm
}
```

Conclusion

This setup allows for automatic shutdown of servers without a proper UPS, enabling dumb UPS units (without USB/Ethernet) to provide protection. Note this **won't power on systems** but prevents hardware damage. Customise IP checks, timing values, and second-check intervals for your environment. If you want to make your computers turn on, on power return, I recommend having a look at AC recovery setting in BIOS or using WoL.