Capstone Task – IoT Sensor Data Logger on Raspberry Pi

Samsung Innovation Campus – IoT Training

Duration: 6 Days (1 Phase per Day + Open-Ended Questions) Starting from (Saturday) 30 Aug till (Thursday) 4 Sept.

Deadline Rule: Submit each **Day Progress before 12 AM.**

This is your Capstone Project for Chapter 5 (Linux 101). It combines all the skills you have learned through the chapter.

Each day you will:

- Complete one project Phase.
- Answer open-ended questions about Linux concepts.

Scenario

You are an IoT engineer at a smart home startup. Your boss has asked you to set up a Raspberry Pi to log simulated sensor data (e.g., temperature readings).

You will:

- Collect and store simulated data.
- Organize files and directories.
- Secure access with users and permissions.
- Monitor processes and logs.
- Automate tasks with shell scripting.
- Apply production-like reliability (log rotation, scheduling, archiving).

Day 1 – Phase 1: System Update & Directory Setup

Boss's Request: Make sure the Raspberry Pi OS is secure and organized.

Tasks:

- Refresh package lists and upgrade the system.
- Verify system details: kernel version, user, time.
- Create /home/<username>/iot_logger with subdirectories: logs, scripts, data.

Open-Ended Questions:

- Draw or describe the Linux architecture layers (hardware → kernel → shell → user space). Where do system calls fit?
- Explain the purpose of these directories: /, /bin, /sbin, /usr, /etc, /var.
- Why does Linux treat everything as a file? Explain the difference between a program and a process.

Day 2 – Phase 2: File & Directory Management + Search

Boss's Request: Organize project files and simulate sensor config checks.

Tasks:

- Inside iot_logger, create logs/temperature.log and scripts/sensor_script.py.
- Copy /etc/services into data and search for patterns like ssh or http.
- Use regex to find lines starting with t or containing numbers.
- Locate .txt files in /home/<username> and remove temporary ones if needed.
- Create hard and symbolic links for temperature.log.
- Display directory structure to confirm organization.

Open-Ended Questions:

- Explain the different types of files in Linux (regular, directory, symbolic link, device, etc.) and how to check them with commands.
- What's the difference between a hard link and a symbolic link? Give real examples of when to use each.
- Is rmdir the same as rm -r when deleting directories? Explain.

Day 3 – Phase 3: User, Group, and Permissions Management

Boss's Request: Secure the project and restrict access to authorized users only.

Tasks:

- Create a new group iot_team and add your user to it.
- Create a new developer user, add it to the group.
- Change ownership of iot_logger to the developer + group.
- Set permissions: group can read/write logs, others blocked.
- Test access as new user, then remove test user.

Open-Ended Questions:

- How do Linux file permissions (r, w, x) work for files vs directories? Give an example using ls -l.
- Explain octal notation for permissions and what the umask command does. Give one calculation example.
- What is the difference between the root user and a normal user? Why is root considered dangerous?

Day 4 – Phase 4: Process and Network Monitoring

Boss's Request: Monitor the system while simulating sensor activity.

Tasks:

- Run a background task to simulate sensor polling.
- List processes and filter for the background task.
- Check network states (established connections).
- Try foreground and background switching.
- Kill a process if needed.

Open-Ended Questions:

- What happens step by step when you type a command in bash (e.g., ls) until you see the output?
- Explain the types of processes in Linux: daemon, zombie, orphan. How can you detect them?
- Why do we need Inter-Process Communication (IPC)? List some IPC mechanisms and real-life examples.

Day 5 – Phase 5: Scripting Automation, Redirection & FDs

Boss's Request: Automate logging with Python and check file descriptors.

Tasks:

- Set an environment variable for sensor type.
- Write scripts/sensor_script.py to simulate data logging (timestamps + random values).
- Redirect script output to logs/temperature.log while running as a background process.
- Find the PID of the process, inspect file descriptors in /proc/<pid>/fd.
- Filter log data into another file.
- Use wildcards to copy logs to data/.
- Clear variable when done.

Challenge – Pipes & FD inspection :

- Run a pipeline (e.g., ls -l | grep .py).
- While it's running, inspect the FDs in /proc/<pid>/fd.
- Hint: To give yourself time, put a sleep in one command of the pipeline so the process stays alive long enough for inspection.

Open-Ended Questions:

- What's the difference between ' ' and " " in shell?
- Explain [-f filename] vs [-d dirname].
- Explain stdout/stderr redirection, appending vs overwrite. How can you confirm redirection using file descriptors?

- Show an example of a for loop in bash. Then, write a simple bash calculator that does add/subtract.

Day 6 – Phase 6: Log Rotation, Scheduling, Archiving

Boss's Request: Prepare the system for production use.

Tasks:

- Configure log rotation for temperature.log (rotate at 1 MB, compress).
- Test by forcing a rotation.
- Schedule the Python script to run every 5 minutes with cron.
- Verify log growth over time.
- Compress old logs into .tar.gz in data/.
- Simulate sending archives to /home/<username>/server/ using cp, scp, or rsync. (hint: use can use scp and copy to destination directory in another path on the same machine just for simulation).

Open-Ended Questions:

- How does cron scheduling work? Show a crontab entry to run a script every 5 minutes.
- Why do we need log rotation? Show an example logrotate config for temperature.log.
- Explain the difference between a Virtual Machine and a Container. Must containers use the same OS as the host? Why or why not?
- Reflection: Which actions in this project combined multiple Linux concepts (e.g., redirection + process monitoring)? How does this apply to real IoT systems?

Submission Guidelines

For each day, submit a folder named DayX containing:

- screenshots of executed tasks and bash script containing all day commands (get them from history :D)
- answers.pdf (with fully explained answer (you can use chatgpt :D but for explaining not for writing the answer and you just copy it, write the answer with your way and add examples if needed).