CSET213-Linux and Shell Programming
Odd 2025

Project/Hackathon Guidelines

A. Domain-Centric Focus

Your project must be centred around specific problem domain. You can choose one or more of the following categories:

1. Hardware-level control and monitoring
2. System-level automation and management
3. Application-level scripting and tools
4. Kernel-level utilities and extensions
5. Device driver programming
6. Linux system security and hardening
7. Vulnerability analysis and threat detection
8. Combined Multi-Layer Projects

Some examples, for each category, are provided as a ready reference in this document.

Your project selection should not be limited to the given examples; you can select the project as per your interest. However, it must align with the core theme of the course project.

B. Use Core Linux Tools and Concepts

All projects must demonstrate clear usage of Linux commands, tools, or scripting principles, including but not limited to:

- Bash scripting (bash, sh)
- File descriptors, pipes, redirection
- System files (/proc, /sys, /dev)
- Process and memory utilities (ps, top, kill, free)
- Network tools (netstat, ping, curl, ss, iptables)
- Job scheduling (cron, at)
- Logging and monitoring (journalctl, dmesg, logrotate)

C. Innovation and Impact

Select a project that:

- Solves a real problem (system admin, user tool, security monitoring, etc.)
- Is innovative in how it approaches the solution using Linux capabilities
- Demonstrates deeper understanding of OS internals, not just surface-level scripting

D. Evaluation Details:

The project/hackathon is of ***10 Marks*** with following distribution:

Evaluation 1 (03)					Evaluation 2(07)				
Idea Novelty (1)	Objectives Clarity (1)	Process Clarity (0.5)	Utillity (0.5)	Total (03)	Objectives Mapping (1)	Demo (3)	Report (2)	Linkedin / X Post (1)	Total (10)

Tentative Evaluation Deadlines:

i. Evaluation # 1: 4th week of August 25
ii. Evaluation # 2: 3rd Week of Nov 2025

E. Team Details:
Your project must have at least 1 and at max 3 members. There must be a designated team leader who must act as point of contact.

Sample Project Ideas

1. Hardware-Level Projects (Device Interface & Control)
These projects interact with physical devices or low-level Linux system interfaces.
Examples:

1. USB auto-mount and auto-backup when a device is connected.
2. GPIO-based home automation using shell scripts on Raspberry Pi.
3. Battery health monitoring using acpi and power logs.
4. Detect and log new hardware events using udev rules.
5. Serial communication logger using minicom or screen.
6. Monitor and alert for overheating using /sys/class/thermal.
7. Log disk temperature and status using smartctl.
8. Real-time LED blinking control using shell scripts and GPIO.
9. External sensor data collector (e.g., DHT11 temperature) + cron logging.
10. Auto-disable USB ports when unauthorized device is inserted.

2. System-Level Projects (Automation & Monitoring)
These automate or monitor system behaviour using shell scripting and core Linux utilities.
Examples:

1. Disk usage monitor that sends alerts when usage exceeds threshold.
2. Automated system update and patch installer.
3. User permission checker and fixer tool.
4. System load balancer: logs top processes and manages systemd services.
5. Custom script to batch install software and configure dev environments.
6. Backup and restore system logs or configurations.
7. Script to clean up old files and directories automatically.
8. Auto-kill resource-hogging processes based on CPU/RAM usage.
9. Convert and compress media files using ffmpeg CLI in batch mode.
10. Script to analyze and summarize boot time and shutdown logs.

3. Network-Level Projects (Configuration & Analysis)
These focus on networking concepts and shell-based network management.
Examples:

1. IP address and port scanner using nmap or netcat.
2. Bandwidth monitor using vnstat or iftop + log exporter.
3. Auto-block IPs with excessive failed login attempts.
4. DNS resolver and traceroute visualizer (log-based).
5. Network configuration snapshot tool for troubleshooting.
6. Remote system manager via ssh, scp, and shell automation.
7. Monitor open ports and alert on changes.
8. Automatically update firewall rules based on IP threat lists.
9. Proxy switcher: enable/disable or rotate proxy via shell script.
10. Ping sweep utility for LAN discovery and device mapping.

4. Application-Level Projects (Utility & Productivity)
Build small but useful applications using shell scripting.
Examples:

1. Personal file organizer: sort files into folders by extension/date.
2. CLI-based to-do list and reminder system.
3. Automatic wallpaper changer using cron.
4. Script-based email sender for notifications or newsletters.
5. Bulk rename files in a directory with pattern rules.
6. Markdown-to-PDF converter using pandoc or latexmk.
7. Shell-based calculator with history and command recall.
8. Movie/TV show downloader and file renamer using wget.
9. Directory usage analyzer and visualizer using ncdu.
10. Custom launcher menu for your most-used terminal commands.

5. Linux Security Projects (Hardening & Monitoring)
These improve the system's security posture and resilience.
Examples:

1. User audit tool to find weak passwords, SUID files, and open SSH access.
2. Script to disable unused services and ports.
3. Auto-lock terminal when idle for a specific period.
4. Log failed logins and IPs and ban repeat offenders (Fail2Ban-lite).
5. SSH key rotation script for all users.
6. Password policy enforcement tool.
7. Kernel bootloader security check (e.g., check for unsigned modules).
8. File permission checker and fixer for /etc, /home, and /var.
9. Rootkit detection wrapper for chkrootkit.
10. System audit script using Lynis or OpenVAS in automated fashion.

6. Vulnerability & Threat Detection Projects
Focus on scanning, detecting, and responding to security issues.
Examples:

1. CVE scanner: check package versions against known CVEs (using osv.dev or NVD APIs).
2. Log parser to detect brute force attacks (auth.log, SSH, Apache).
3. File integrity checker using sha256sum + daily cron compare.
4. Monitor unauthorized file access using auditd.
5. Build a mini host-based intrusion detection system (HIDS).
6. Monitor new user or group creation and send alerts.
7. Detect privilege escalation vectors (SUID, world-writable, etc.).
8. Threat feed ingestion and IP blocking (e.g., from AbuseIPDB).
9. Shell wrapper for ClamAV that scans, logs, and emails reports.
10. Log behavior comparison before and after system changes.

7. Kernel-Level and Device Driver Projects (Advanced)
Projects involving kernel modules or interacting with kernel-space code.
Examples:

1. Write a basic character device driver (read/write from /dev/mydev).
2. Kernel module to log process creation and termination.
3. Add a simple system call to a custom kernel (educational).
4. Implement a virtual file that returns system info on read.

5. Log open/write calls on a specific file using LKM.
6. Kernel-space ping logger that hooks into ICMP.
7. Device driver for a virtual LED using timers and interrupts.
8. Communicate with userspace via netlink sockets.
9. Loadable kernel module that tracks active users.
10. Modify kernel scheduler behavior for a process class (advanced).

8. Combined Multi-Layer Projects

These involve integration across hardware, OS, and application levels.

Examples:

1. IoT data logger: sensor input → shell logger → cron → report → alert.
2. USB firewall: auto-detect device + scan for threats + disable if unknown.
3. Real-time system dashboard + remote access CLI via SSH.
4. Honeypot + log analyzer + threat response via iptables.
5. Smart shutdown system: monitor sensors and log activity before shutdown.
6. Self-healing service monitor: restart crashed services automatically.
7. Secure backup agent: encrypted backups, logs, and alerts.
8. Kernel hook + bash interface for system call tracking.
9. Boot-time diagnostic tool: logs + file system check + network test.
10. Threat visualization tool: analyze logs and generate HTML reports.