# Linux Fundamentals Project: System Information Automation

**Project by:** Reut Abergel

## Table of Contents

# 1. Introduction

This document serves as a technical guide for the Linux Info Automation Tool. The primary function of this script is to act as a centralized Dashboard for system administrators, automating the collection of critical system data.

Instead of manually executing separate commands to check IP addresses, CPU load, and disk usage, this tool aggregates these functions into a modular, interactive Bash script. It features color-coded outputs for rapid readability.

**Key Features:**

- **Network Recon:** Instantly identifies Public IP, Private IP, and MAC addresses.
- **Resource Monitoring:** Tracks real-time CPU spikes and Memory availability.
- **Service Auditing:** Lists active system services (systemd) to verify operational status.
- **Storage Forensics:** Locates the largest files in the user directory

# 2. Target Audience

This guide and tool are designed for:

- **System Administrators:** a quick health-check script for daily server monitoring.
- **Linux Beginners:** Students learning how to chain commands using pipes (|) and filters (awk, sed).
- **DevOps Engineers:** Users looking to automate status reporting on remote servers.

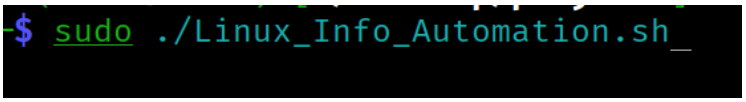
# 3. Requirements

To successfully execute this script, the following requirements must be met:

1. **Operating System:** Any Linux distribution (Kali, Ubuntu).
2. **Shell:** Standard Bash shell (/bin/bash).
3. **Dependencies:** Standard system utilities found in most distros: curl, ifconfig (net-tools), top, free, systemctl.
4. **Internet Connection:** Required for the external IP check (ifconfig.me).

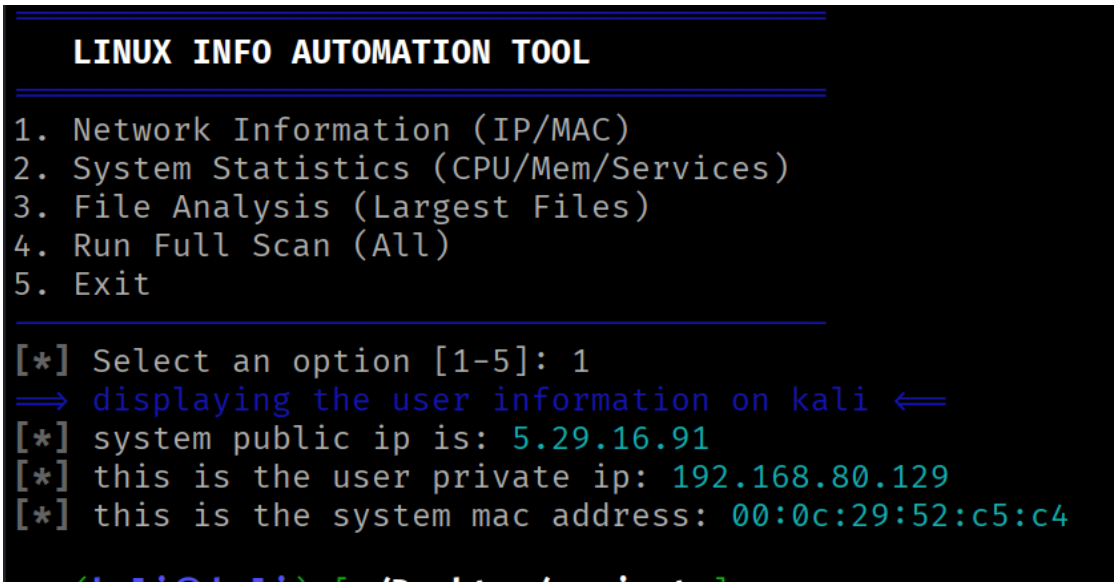
# 4. Operational Guide

**Step 1 Execution & Menu Selection:** The script launches an interactive menu driven by a case statement.



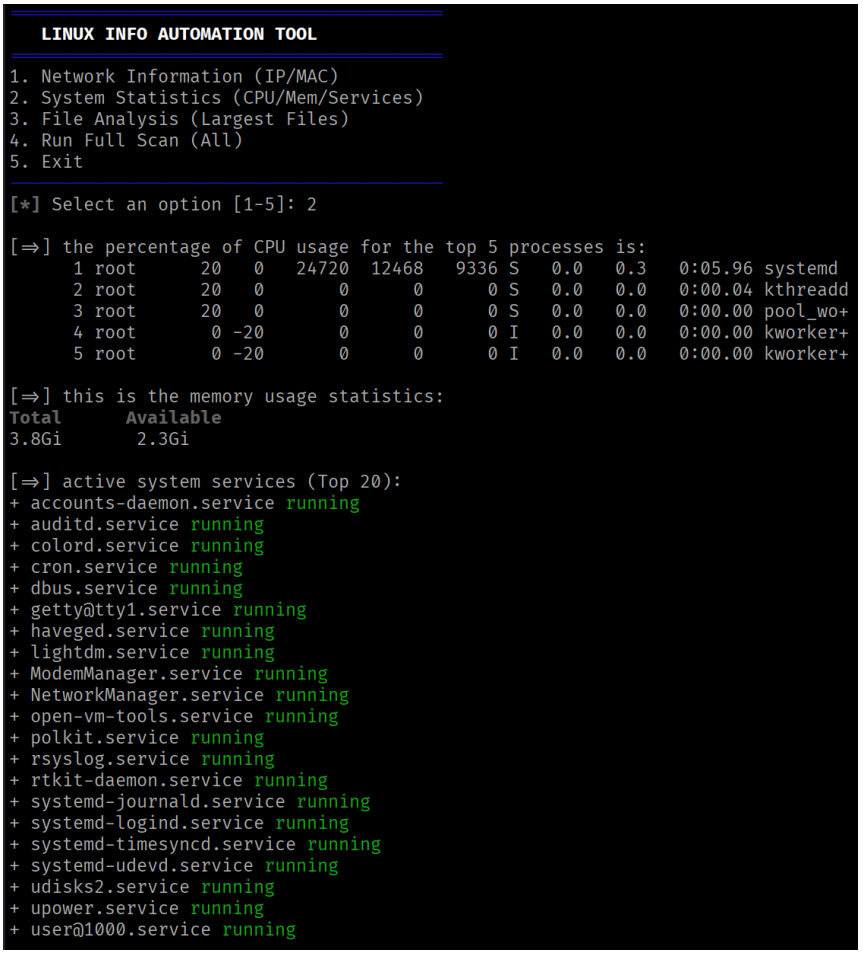
**Step 2 Network Information Gathering:** Selecting **Option 1** triggers the NET_INFO function..

- It reports the internal **Private IP** and the physical **MAC Address** of the interface.




**Step 3 System Health Monitoring:** Selecting **Option 2** triggers the SYS_STATS function.

- **CPU:** Displays the top 5 processes consuming the most CPU power.
- **Memory:** Shows a formatted table of Total vs. Available RAM.
- **Services:** Lists the top 20 active system units (services) that are currently running.



**Step 4 Storage Analysis:** Selecting **Option 3** triggers the FILE_ANALYSIS function.

- The script scans the /home directory recursively.
- It sorts all found files by size and displays the **Top 10 largest files**, allowing the user to quickly identify what is taking up space.

```
   LINUX INFO AUTOMATION TOOL

1. Network Information (IP/MAC)
2. System Statistics (CPU/Mem/Services)
3. File Analysis (Largest Files)
4. Run Full Scan (All)
5. Exit

[*] Select an option [1-5]: 3

[⇒] top 10 largest files in the /home directory:
2.1G    /home/kali/Desktop/tools/vol3_files/m4.vmem
641M    /home/kali/.cache/vmware/drag_and_drop/sYjgSp/HDD/File.e01
641M    /home/kali/.cache/vmware/drag_and_drop/FeDxQe/HDD/File.e01
513M    /home/kali/networkproj/memdump.mem
513M    /home/kali/Desktop/projects/memdump.mem
477M    /home/kali/Desktop/tools/TheFatRat/.git/objects/pack/pack-c6a386353d6e3f03371209d952bf9b148560e58a.pack
437M    /home/kali/.BurpSuite/burpbrowser/139.0.7258.127/chrome
400M    /home/kali/.cache/vmware/drag_and_drop/sYjgSp/HDD/File.e02
400M    /home/kali/.cache/vmware/drag_and_drop/FeDxQe/HDD/File.e02
117M    /home/kali/Desktop/projects/forensics_results_2026-02-06_11-15-04.zip
```

## 5. Purpose & Automation

The goal is to replace repetitive manual typing with automated efficiency.

1. **Visual Clarity:** Raw Linux command output (like top or find) can be overwhelming. This script uses text processing tools (sed, awk) to strip away the noise and display only the relevant lines.
2. **Modular Architecture:** The script is built on **Functions** (e.g., NET_INFO, SYS_STATS). This makes the code cleaner, easier to debug, and allows the user to run specific modules without waiting for a "full scan.

## 6. Technical Analysis

The script relies on four core functions to manage the workflow:

**1. Function NET_INFO (Connectivity):** Aggregates internal and external network identity.

- Uses curl -s (silent mode) to fetch the external IP without showing the progress bar.
- Uses grep and awk to filter the ifconfig output, isolating just the MAC address string.

```
#Network Information
function NET_INFO()
{
    echo -e "${BOLD}${BLUE}==> displaying the user information on kali <==${NC}"
    #Identify the system's public IP
    echo -e "${INFO} system public ip is: ${CYAN}$(curl -s ifconfig.me)${NC}"
    #Identify the private IP address
    echo -e "${INFO} this is the user private ip: ${CYAN}$(hostname -I)${NC}"
    #display the mac address
    echo -e "${INFO} this is the system mac address: ${CYAN}$(ifconfig eth0 | grep ether | awk '{print $2}')${NC}"
}
```

**2. Function SYS_STATS (Resource Usage):** Filters real-time system data into static reports.

- **Services:** Queries systemctl for running services, then uses awk to add a green "+" indicator next to active services for readability.
- **CPU:** Runs top in batch mode (-b) for a single iteration (-n 1). It pipes the output to sed, printing only lines 8 through 12 (the top processes).

```
#System Statistics
function SYS_STATS()
{
    #Display the percentage of CPU usage for the top 5 processes.
    echo -e "\n${BOLD}${WHITE}[=>] the percentage of CPU usage for the top 5 processes is:${NC}"
    top -b -n 1 -o %CPU | sed -n '8,12p'

    #Display memory usage statistics: total and available memory
    echo -e "\n${BOLD}${WHITE}[=>] this is the memory usage statistics:${NC}"
    free -h | awk -v gray="$GRAY" -v white="$WHITE" -v nc="$NC" 'NR==1{print gray "Total      Available" nc} NR==2{print white $2 "       " $7 nc}'

    #List active system services with their status
    echo -e "\n${BOLD}${WHITE}[=>] active system services (Top 20):${NC}"
    systemctl list-units --type=service --state=running | awk -v green="$GREEN" -v nc="$NC" '{print "+ " $1 " " green $4 nc}' | tail -27 | head -21
}
```

**3. Function FILE_ANALYSIS (Disk Usage):** Identifies storage hogs.

- **Find:** Searches /home for files (-type f).
- **-Exec:** Runs du -h (disk usage human-readable) on every file found.
- **Sort:** Orders the results by human-readable size (-hr), putting the biggest files (GB/MB) at the top.
- **Head:** Limits the output to the top 10 results.

```
#File Analysis
function FILE_ANALYSIS()
{
    #Locate the Top 10 Largest Files in /home
    echo -e "\n${BOLD}${WHITE}[=>] top 10 largest files in the /home directory:${NC}"
    find /home -type f -exec du -h {} + 2>/dev/null | sort -hr | head -n 10
}
```

**4. Function SHOW_MENU (User Interface):** Controls the flow of the application and handles user input.

- **Case Statement:** Utilizes a case structure to evaluate the user's variable $CHOICE and execute the corresponding function (NET_INFO, SYS_STATS…).
- **Exit Strategy and Run Full Scan (All):** Executes all three analysis modules sequentially. This "View All" strategy provides a complete system report. And Exit (Option 5) provides a clean exit point.

```
# INTERACTIVE MENU
function SHOW_MENU()
{
    clear
    echo -e "${BLUE}=========================================${NC}"
    echo -e "${BOLD}   LINUX INFO AUTOMATION TOOL   ${NC}"
    echo -e "${BLUE}=========================================${NC}"
    echo -e "${WHITE}1.${NC} Network Information (IP/MAC)"
    echo -e "${WHITE}2.${NC} System Statistics (CPU/Mem/Services)"
    echo -e "${WHITE}3.${NC} File Analysis (Largest Files)"
    echo -e "${WHITE}4.${NC} Run Full Scan (All)"
    echo -e "${WHITE}5.${NC} Exit"
    echo -e "${BLUE}-----------------------------------------${NC}"

    read -p "$(echo -e ${INFO}" Select an option [1-5]: "${NC})" CHOICE

    case $CHOICE in
        1)
            NET_INFO
            ;;
        2)
            SYS_STATS
            ;;
        3)
            FILE_ANALYSIS
            ;;
        4)
            NET_INFO
            SYS_STATS
            FILE_ANALYSIS
            ;;
        5)
            echo -e "${OK} Exiting tool. Goodbye!"
            exit 0
            ;;
        *)
            echo -e "${RED}[!] Invalid option. Please try again.${NC}"
            sleep 1
            SHOW_MENU
            ;;
    esac
```

## 7. Technical Skills Gained

Through the development of this project, I developed the following technical skills:

- **Bash Scripting Proficiency:** Mastered variables, custom functions, and case statements for control flow.
- **Advanced Text Processing:** Utilized awk, sed, and grep to parse raw command output into clean, formatted reports.
- **System Administration:** Gained a deeper understanding of how Linux handles processes (top), memory (free), and service management (systemctl).

## 8. Summary

This project resulted in a fully automated Linux Information tool. By wrapping complex system commands into simple, reusable functions, it significantly speeds up the reconnaissance and monitoring process. It demonstrates the ability to write clean, modular code suitable for real-world administration tasks.