SomNOG

Capacity Building Series 2026

Linux 201: System Administration and Management

Hands-On Exercise Sheet

19 May 2026 | Linux 201 | Virtual (Zoom)

somnog.so | info@somnog.so | @SomNOG

This exercise sheet is designed to be worked through after the session. Each exercise builds on the previous one. Read the instructions carefully, run each command, and observe the output. If something doesn't work as expected, that's learning! Check your syntax, re-read the notes, and try again.

Topics Covered

Exercise 1: Vim Editor

Vim is the most widely available text editor on Linux servers. Being comfortable in Vim means you can edit configuration files, scripts, and logs on any server without needing to install anything extra.

Setup

Open a terminal and create a new file to work with:

```
$ vim practice.txt
```

Part A: The Three Essential Modes

1. When Vim opens, you are in NORMAL mode. Try pressing the arrow keys to move around.
2. Press `i` to enter INSERT mode. You will see `-- INSERT --` at the bottom.
3. Type the following lines exactly:

   Somalia's Internet community is growing.
   Linux powers most of the world's servers.
   SomNOG builds local technical capacity.
   Linux 201 - SomNOG
   SomNOG8
4. Press `Esc` to return to NORMAL mode.

Part B: Editing and Navigation

1. Position your cursor on the first line using `gg` (jump to top).
2. Press `dd` to delete the first line.
3. Press `u` to undo - the line comes back.
4. Use `j` and `k` to move down and up.
5. On the last line, press `o` (lowercase) to open a new line below and enter INSERT mode.
6. Type: SomNOG9 Conference 2026 - then press `Esc`.

Part C: Search and Replace

1. In NORMAL mode, type `/Linux` and press Enter - Vim jumps to the word.
2. Press `n` to find the next occurrence.
3. Now type the following command to replace:

```
:s/SomNOG8/SomNOG9/g
```

4. Press Enter. The word should be replaced on that line from SomNOG8 to SomNOG9 now.

Part D: Save and Exit

1. To save without quitting: `:w` then Enter
2. To save and quit: `:wq` then Enter
3. Open the file again with `vim practice.txt` to verify your changes are saved.

> ✓ *Checkpoint: You should be able to open Vim, type text, navigate, search, and save/exit confidently.*

Exercise 2: Users and Groups

Every file and process in Linux belongs to a user. Understanding how to create, manage, and inspect users and groups is essential for any system administrator.

Part A: Inspect the System's User Database

Before creating new users, let's understand the existing structure:

```
$ cat /etc/passwd | head -5
# Observe the 7 fields: username:password:UID:GID:GECOS:home:shell

$ cat /etc/group | head -5
# Observe the 4 fields: groupname:password:GID:members

$ id                          # show your own UID, GID, groups
$ whoami                      # show your current username
```

Part B: Create Users

Use sudo to create two new users. We'll use Somali names as examples:

```
$ sudo useradd -m -c 'Ali Omar' ali
# -m creates the home directory   -c adds a comment (full name)

$ sudo useradd -m -c 'Ruun Hassan' ruun

$ sudo passwd ali             # set a password for ali
$ sudo passwd ruun            # set a password for ruun
```

Part C: Verify the Users Were Created

```
$ cat /etc/passwd | grep ali
Expected output: ali:x:1001:1001:Ali Omar:/home/ali:/bin/bash

$ cat /etc/passwd | grep ruun

$ ls /home/                   # confirm home directories were created
$ id ali                      # show ali's UID, GID, groups
```

Part D: Create a Group and Add Users

```
$ sudo groupadd somnog

$ sudo usermod -aG somnog ali    # -a = append, -G = supplementary group
$ sudo usermod -aG somnog ruun

$ cat /etc/group | grep somnog
Expected output: somnog:x:1002:ali,ruun

$ id ali                         # confirm somnog group appears in ali's groups
```

Part E: Modify and Remove a User

```
$ sudo usermod -c 'Ali Omar Hassan' ali    # update the comment field
$ cat /etc/passwd | grep ali               # verify the change

$ sudo userdel -r ruun                     # -r also removes home directory
$ ls /home/                                # confirm ruun's home is gone
```

> ✓ *Checkpoint: You should be able to create users with full name fields, create groups, add users to groups, and verify everything in /etc/passwd and /etc/group.*

Exercise 3: File Permissions and Ownership

Permissions control who can read, write, or execute a file. There are two methods to change them - symbolic and numeric. Practice both until they feel natural.

Part A: Read Permissions

Create a few test files and observe their default permissions:

```
$ mkdir ~/permtest && cd ~/permtest
$ touch file.txt script.sh notes.txt
$ ls -la
# Read the output carefully:
# -rw-r--r--  1 ali  ali   0 May 19 16:45 file.txt
# Columns:  permissions  links  owner  group  size  date  name
```

Part B: Method 1: Symbolic

Symbolic mode uses letters to describe changes. Format: chmod [who][+/-/=][rwx] file

```
$ chmod u+x script.sh          # give the owner execute permission
$ ls -la script.sh             # should now show: -rwxr--r--

$ chmod g+rw notes.txt         # give group read and write
$ ls -la notes.txt             # should show: -rw-rw-r--

$ chmod o-r file.txt           # remove read from others
$ ls -la file.txt              # should show: -rw-r-----

$ chmod a+r notes.txt          # give everyone read
```

Part C: Method 2: Numeric (Octal)

Numeric mode uses three digits: [owner][group][others]. Each digit: r=4, w=2, x=1.

```
# Calculate: rwx = 4+2+1 = 7    rw- = 4+2+0 = 6    r-- = 4+0+0 = 4

$ chmod 755 script.sh         # rwxr-xr-x  (owner full, group+others r-x)
$ ls -la script.sh

$ chmod 644 file.txt          # rw-r--r--  (standard file permissions)
$ ls -la file.txt

$ chmod 600 private.txt       # rw-------  (owner only - for private keys etc.)
$ touch private.txt && chmod 600 private.txt && ls -la private.txt
```

Part D: Ownership with chown

```
$ sudo chown ali file.txt           # change owner to ali
$ sudo chown ali:somnog script.sh   # change owner AND group
$ ls -la                            # verify changes

$ mkdir ~/shared && sudo chown :somnog ~/shared    # change only group
$ sudo chmod 775 ~/shared                          # group can write to shared dir
```

> ✓ *Checkpoint: You should be comfortable with both chmod methods and know when to use each. Symbolic is more readable for small changes; numeric is faster for setting exact permissions in one command.*

Exercise 4: Redirection and Piping

Redirection and pipes are what make the Linux command line so powerful. Instead of viewing output on screen, you can save it, filter it, count it, or feed it into the next command.

Part A: Output Redirection (> and >>)

```
$ ls -la /etc > etc_listing.txt      # save output to file (overwrites)
$ cat etc_listing.txt | head -10     # view first 10 lines

$ echo 'First log entry' > app.log
$ echo 'Second entry'   >> app.log   # >> appends - does NOT overwrite
$ echo 'Third entry'    >> app.log
$ cat app.log                        # should show all three lines

$ ls -la > app.log                   # WARNING: > overwrites - app.log is replaced
$ cat app.log
```

Part B: Error Redirection (2>)

```
$ ls /this/does/not/exist 2> errors.txt
$ cat errors.txt                    # the error message is now in the file

$ ls /real/path /fake/path > out.txt 2> err.txt   # split stdout and stderr
$ ls /real/path /fake/path > all.txt 2>&1         # combine both into one file
```

Part C: Pipes (|)

```
$ cat /etc/passwd | grep somnog      # filter passwd for somnog entries

$ ps aux | grep bash                 # find all bash processes

$ ls -la /etc | wc -l               # count files in /etc

$ cat /var/log/syslog | grep 'error' | head -10   # first 10 error lines

$ ps aux | sort -k3 -rn | head -5   # top 5 processes by CPU usage
```

> ✓ *Checkpoint: Try building a one-liner that: lists all files in /var/log, filters for files ending in .log, and saves the result to ~/log_files.txt*

Exercise 5: Process Management and dd

Every running program is a process. This exercise covers how to find, monitor, signal, and stop processes - and uses dd as a practical process to work with since it runs long enough to observe.

Part A: Exploring Running Processes

```
$ ps aux                            # list ALL processes
$ ps aux | wc -l                    # count how many processes are running
$ ps aux | grep bash                # find bash processes
$ ps -ef | grep root | head -10      # show root-owned processes

$ pgrep bash                        # show PIDs of processes named 'bash'
$ pstree                            # show process tree (install if needed: sudo apt
install psmisc)
```

Part B: Using top and htop

```
$ top
# While top is running:
#   Press  P  to sort by CPU usage
#   Press  M  to sort by memory usage
#   Press  q  to quit

$ sudo apt install htop -y          # install htop if not available
$ htop
# Explore the colour-coded display. Press F10 or q to quit.
```

Part C: dd as a Process to Observe

dd is a useful tool for disk operations. It also runs long enough to practice process management:

```
# Run dd in the background to create a 200MB test file
$ dd if=/dev/zero of=~/testfile.img bs=1M count=200 &
# The & puts it in the background and prints the PID

# Find dd's PID
$ pgrep dd
$ ps aux | grep dd

# Send the USR1 signal - this makes dd report progress WITHOUT stopping
$ kill -USR1 $(pgrep dd)

# Watch dd's progress in real time
$ watch -n1 'kill -USR1 $(pgrep dd) 2>/dev/null && echo running || echo done'
# Press Ctrl+C to stop watch

# Stop dd gracefully
$ kill $(pgrep dd)

# Verify it stopped
$ pgrep dd && echo 'still running' || echo 'stopped'

# Clean up
$ rm ~/testfile.img
```

Part D: Signals Quick Reference

Signals are messages sent to processes. The most common ones:

- kill -15 (SIGTERM) - politely ask the process to stop (default kill)
- kill -9 (SIGKILL) - force-kill immediately, no cleanup
- kill -1 (SIGHUP) - reload configuration (commonly used with services)
- kill -USR1 - user-defined signal (dd uses it to print progress)

> ✓ *Checkpoint: You should be able to run a process in the background, find its PID, send signals to it, and confirm it has stopped.*

Exercise 6: Package Management

Package managers handle software installation, dependency resolution, and updates automatically. This exercise uses APT (Ubuntu/Debian). If you're on CentOS/RHEL/Fedora, substitute apt commands with dnf.

Part A: Update and Search

```
$ sudo apt update                    # refresh package index
$ sudo apt list --upgradable         # see what can be updated

$ apt search htop                    # search for a package
$ apt show htop                      # show package details and dependencies
```

Part B: Install, Verify, and Remove

```
$ sudo apt install htop -y          # install htop (-y skips confirmation)
$ which htop                        # confirm it's installed
$ htop --version                    # check version

$ sudo apt install curl tree -y     # install multiple packages at once
$ tree ~                            # use tree to show your home directory

$ dpkg -l | grep htop              # list installed packages matching 'htop'

$ sudo apt remove htop             # remove package (keeps config files)
$ sudo apt purge htop              # remove package AND config files
$ which htop                       # should return nothing
```

> ✓ *Checkpoint: Update the package list, install 'curl' if not already installed, then use it: curl https://api.ipify.org - this returns your public IP address.*

Exercise 7: System Monitoring

A system administrator needs to know what is happening on a system at any given moment. This exercise builds your instinct for quickly checking the health of a Linux system.

Part A: Disk Usage

```
$ df -h                             # disk usage of all filesystems (-h = human
readable)
$ df -h /                           # disk usage of root filesystem only

$ du -sh ~                          # size of your home directory
$ du -sh /var/log/*                 # size of each item in /var/log
$ du -sh /var/log/* | sort -h       # sort by size (smallest first)
```

Part B: Memory Usage

```
$ free -h                           # RAM and swap usage (-h = human readable)
# Output columns:
#   total  = total installed RAM
#   used   = currently in use
#   free   = completely unused
#   available = what's actually available for new processes

$ free -h -s 2                      # refresh every 2 seconds (Ctrl+C to stop)
```

Part C: CPU and Load

```
$ uptime
# Shows: current time, uptime, users logged in, load averages (1m, 5m, 15m)
# Load average: 1.0 on a single-core system = 100% CPU. < 1.0 = fine.

$ nproc                              # how many CPU cores does this system have?

$ top -bn1 | head -20               # one-time snapshot (no interactive mode)

$ vmstat 1 5                         # system stats every 1 second, 5 times
```

> ✓ *Checkpoint: Answer these questions using the commands above: (1) How much disk space is free on your root partition? (2) How much memory is available? (3) What is the 1-minute load average?*

Exercise 8: Log Files

Logs are the primary tool for troubleshooting. When something breaks - a service crashes, a user can't log in, disk fills up - the logs tell you exactly what happened.

Part A: Exploring /var/log

```
$ ls /var/log/                       # list all log files and directories
$ ls -lh /var/log/ | sort -k5 -rh   # sort by size, largest first

$ sudo cat /var/log/auth.log | tail -20    # last 20 auth log entries
$ sudo cat /var/log/syslog | tail -20      # last 20 system log entries
```

Part B: Filtering Logs

```
$ sudo grep 'sudo' /var/log/auth.log | tail -10      # recent sudo usage
$ sudo grep 'Failed' /var/log/auth.log               # failed login attempts

$ sudo grep 'error' /var/log/syslog | wc -l          # count error lines
$ sudo grep 'warning' /var/log/syslog | tail -5      # last 5 warnings
```

Part C: Live Log Monitoring

```
# Open two terminals side by side for this exercise

# Terminal 1: follow the auth log in real time
$ sudo tail -f /var/log/auth.log

# Terminal 2: trigger a login event by running sudo
$ sudo ls /

# Switch back to Terminal 1 - you should see the sudo event appear live
# Press Ctrl+C to stop tail -f
```

Part D: journalctl (systemd logs)

```
$ journalctl -n 20                  # last 20 journal entries
$ journalctl -f                     # follow live (like tail -f)
$ journalctl --since '10 min ago'   # entries from the last 10 minutes
$ journalctl -u ssh                 # logs for the SSH service only
$ journalctl -p err                 # only error-level entries
```

✓ *Checkpoint: Find the last time you (or anyone) used sudo on this system. Use grep on auth.log to locate it.*

Exercise 9: Putting It All Together (Challenge)

This final exercise has no step-by-step instructions. Use everything you've learned to complete the tasks below. Each task is deliberately open-ended, there's often more than one correct way.

This is a challenge exercise. Try each task on your own first. If you get stuck for more than 5 minutes, re-read the relevant section in your notes or ask your facilitator.

1 User and Group Setup

Create three users: asha, omar, and nadra. Create a group called network-team. Add all three users to the group. Verify that all three appear in the group's entry in /etc/group.

2 Project Directory with Correct Permissions

Create a directory called /srv/network-project (use sudo). Set the owner to omar and the group to network-team. Set permissions so that: the owner can read, write, and execute - the group can read and write - others can only read. Use numeric notation. Verify with ls -la.

3 Script, Permissions, and Ownership

Create a shell script called check-system.sh in /srv/network-project. The script should contain three commands that print disk usage, memory, and the current date. Make the script executable by the owner only. Change ownership to asha. Run the script as asha.

4 Capture and Filter Process Information

Run a dd command in the background (write a 100MB file). Capture the full output of ps aux into a file called ~/processes.txt. Pipe that file through grep to filter for dd, and save the result to ~/dd-process.txt. Stop dd. Verify it stopped by checking pgrep.

5 Log Investigation

Answer the following using log commands only (do not guess): How many lines in /var/log/syslog contain the word 'kernel'? What is the most recent entry in /var/log/auth.log? Save both answers into a file called ~/log-report.txt using redirection.

6 Package and Verification

Install the package 'net-tools' (provides ifconfig and netstat). Verify it's installed using dpkg. Run: netstat -tuln > ~/open-ports.txt to save a list of open ports. Use cat and grep together to extract only lines containing ':22' or ':80'.

Well done for completing the exercises.

The skills you've practiced here are used daily by system administrators, DevOps engineers, and network operators worldwide.