

CSS 262: Linux Administration

*nix Systems for Cybersecurity

Lecture 1: Course Introduction & The Shell





Welcome to CSS 262

- **What this course is about:** Linux Administration with a security-first approach
- **Who this course is for:** Future cybersecurity professionals
- **What you'll learn:** How to deploy, maintain, and secure Linux environments
- **Why it matters:** Essential skills for DevSecOps and Security Operations roles

💡 **Key Philosophy:** We focus on the "why" and "how" of system internals, not just commands to memorize.



Course Details

Course Code: CSS 262

Duration: 15 Weeks

Credits: 5-6 ECTS

Level: Undergraduate



Workload

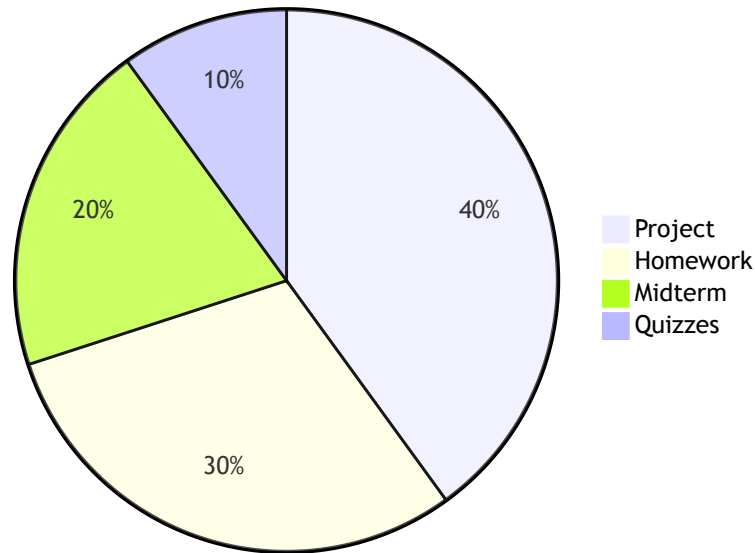
- Lectures: 15 hours
- Labs: 30 hours
- Self-study: 45 hours
- Assignments: 16 hours
- Exams & Project: 32 hours




Total: ~138 Hours



Grading Breakdown

Assessment Distribution



-  **Automated Grading** via GitHub Actions
-  Resubmissions allowed until deadline
-  Code must work on standard environment



Course Learning Outcomes

1. **Administer** fundamental Linux components (users, permissions, storage, systemd)
2. **Develop** automated solutions using Bash scripting
3. **Configure** robust network settings and package management
4. **Harden** server security (SSH, firewalls, service management)
5. **Analyze** system logs and audit trails for security incidents
6. **Construct** production-ready infrastructure using IaC principles



Course Journey

Weeks 1-7: Foundations

- **Week 1:** Shell & VM Setup 🔧
- **Week 2:** Users & Permissions 👤
- **Week 3:** Processes & Systemd ⚙️
- **Week 4:** Storage & Filesystems 💾
- **Week 5:** Bash Scripting 📖
- **Week 6:** Networking Basics 🌐
- **Week 7:** Package Management 📦







Weeks 8-15: Security Focus

- **Week 8:** 🔥 **Midterm Exam**
- **Week 9:** SSH Hardening 🔒
- **Week 10:** Firewalls 🛡️
- **Week 11:** SELinux/MAC 🗝️
- **Week 12:** Logging & Auditing 📊
- **Week 13:** Docker Security 🐳
- **Week 14:** Vuln Scanning 🔍
- **Week 15:** 🎓 **Capstone Project**



Skills You'll Acquire

Academic Skills

-  Problem-solving & troubleshooting
-  Critical thinking & root cause analysis
-  Research skills & documentation
-  Practical coding in Bash/Shell
-  Self-directed learning
-  Security-first mindset

Technical Skills

- Filesystem & permission management
- Security auditing scripts
- Firewall configuration
- Mandatory Access Control (SELinux)
- Log analysis & forensics
- Systemd & LVM administration



Required Resources



Textbooks

1. **The Linux Command Line (2nd Edition)** - William Shotts
2. **UNIX and Linux System Administration Handbook (5th Edition)** - Evi Nemeth et al.
3. **PicoCTF Learning Primer** - For CTF-style challenges



Hardware & Software

- **Laptop:** Minimum 8GB RAM
- **Virtualization:** VirtualBox or VMware
- **Tools:** Git, VS Code
- **Accounts:** GitHub (Free Educational Account)



Important: You MUST have a working VM environment by next week's lab!

Part 2: The Shell



Understanding the command-line interface

What is a Shell?

Definition

The **shell** is a command-line interpreter that provides a user interface for accessing operating system services.

Why Command Line?

- ⚡ **Power & Efficiency:** Automate repetitive tasks
- 🎯 **Precision:** Fine-grained control over system
- 🔄 **Reproducibility:** Script your actions
- 🌐 **Remote Access:** Manage systems over SSH
- 💪 **Professional Standard:** Industry-standard for sysadmin

Shell Types

Common Shells

- **bash** - Bourne Again Shell (Most common)
- **zsh** - Z Shell (Modern, feature-rich)
- **sh** - Original Bourne Shell
- **fish** - Friendly Interactive Shell
- **dash** - Debian Almquist Shell

In this course: We'll focus on **bash** (default on most Linux distributions)

Check Your Shell

```
1  # See current shell
2  echo $SHELL
3
4  # List available shells
5  cat /etc/shells
6
7  # Change shell
8  chsh -s /bin/bash
```

Shell vs Terminal

- **Terminal:** The window/application
- **Shell:** The program running inside
- Think: Terminal = TV, Shell = Channel

The Command Prompt

```
1 user@hostname:~/directory$ command [options] [arguments]
```

Prompt Components

- `user` - Your username
- `hostname` - Computer name
- `~/directory` - Current location
- `$` - Regular user
- `#` - Root/superuser

Special Paths

- `~` - Home directory
- `/` - Root directory
- `.` - Current directory
- `..` - Parent directory
- `-` - Previous directory

💡 ****Pro Tip:**** The prompt can be customized via the `'PS1'` environment variable!

Essential Navigation Commands

Moving Around

```
1  pwd          # Print working directory
2
3  # List contents
4  ls
5  ls -l        # Long format
6  ls -a        # Show hidden files
7  ls -lah      # All options
8
9  # Change directory
10 cd /etc
11 cd ~         # Go home
12 cd ..        # Go up one level
13 cd -         # Previous directory
```

File Operations

```
1  touch file.txt      # Create file
2  cp source.txt dest.txt # Copy
3  mv old.txt new.txt  # Move/rename
4  rm file.txt         # Delete
5  rm -r directory/    # Recursive delete
6
7  mkdir new_folder    # Create dir
8  mkdir -p path/to/folder # With parents
```

Getting Help

1. Manual Pages (man)

```
1  man ls      # Read the manual for ls
2  man man     # Learn about man itself!
```

2. Command Help


```
1  ls --help   # Quick help
2  help cd     # For built-in commands
```

3. Type and Which

```
1  type ls     # Show command type
2  which python # Show command location
```

4. Info Pages

```
1  info coreutils # GNU info system
```

 **Learning Skill:** Reading man pages is critical for your career!

File System Hierarchy

1	/	Root directory
2	— bin/	Essential user binaries
3	— boot/	Boot loader files
4	— dev/	Device files
5	— etc/	System configuration
6	— home/	User home directories
7	— root/	Root user's home
8	— tmp/	Temporary files
9	— usr/	User programs
10	— bin/	User binaries
11	— local/	Locally installed
12	— var/	Variable data (logs, etc.)



Everything in Linux is a file! Including devices, processes, and network sockets.

Command Structure & Syntax

Basic Anatomy

```
1  command [options] [arguments]
```

- **Command:** The program to run (e.g., `ls` , `cat` , `grep`)
- **Options:** Modify behavior (e.g., `-l` , `--all` , `-R`)
- **Arguments:** What to operate on (e.g., files, directories)

Examples

```
1  ls -l /home/user          # List in long format
2  cp -r source/ destination/ # Copy recursively
3  grep -i "error" /var/log/syslog # Case-insensitive
```

⚠️ ****Remember:**** Options can be combined: ``ls -lah` = `ls -l -a -h``

Keyboard Shortcuts

Navigation & Editing

- `Ctrl + A` / `Ctrl + E` - Line start/end
- `Ctrl + </>` - Jump by word
- `Ctrl + K` - Cut to end of line
- `Ctrl + U` - Cut to start of line
- `Ctrl + W` - Delete previous word

Control

- `Ctrl + C` - Interrupt command
- `Ctrl + D` - Exit/logout
- `Ctrl + Z` - Suspend process
- `Ctrl + L` - Clear screen

History Navigation

- `↑/↓` - Browse history
- `Ctrl + R` - Reverse search
- `history` - Show all commands
- `!!` - Repeat last command
- `!n` - Run command #n

Pro Tips

- Press `Tab` for autocomplete
- Double `Tab` for all options
- `Esc + .` - Last argument

Viewing File Contents

Basic Viewing

```
1  cat file.txt           # Display entire file
2  less file.txt          # Paginated viewer (q to quit)
3  head file.txt          # First 10 lines
4  head -n 20 file.txt    # First 20 lines
5  tail file.txt          # Last 10 lines
6  tail -f /var/log/syslog # Follow in real-time
```

Text Search

```
1  grep "pattern" file.txt # Search for pattern
2  grep -r "error" /var/log/ # Recursive search
3  grep -i "warning" file.txt # Case-insensitive
```

Pipes and Redirection

Redirection Operators

```
1  command > file.txt      # Redirect output (overwrite)
2  command >> file.txt     # Redirect output (append)
3  command < file.txt      # Redirect input
4  command 2> errors.txt   # Redirect stderr
5  command &> all.txt       # Redirect stdout + stderr
```

Pipes (|)

```
1  ls -l | grep ".txt"     # List only .txt files
2  cat /var/log/syslog | grep error | less # Chain commands
3  ps aux | grep apache | wc -l      # Count processes
```



****Unix Philosophy:**** Write programs that do one thing well and work together via pipes.

Wildcards & Pattern Matching

Glob Patterns

```
1  *           # Matches any characters
2  ?           # Matches single character
3  [abc]       # Matches a, b, or c
4  [a-z]       # Matches any lowercase letter
5  [!abc]      # Matches anything except a, b, or c
```

Examples

```
1  ls *.txt           # All .txt files
2  ls file?.txt       # file1.txt, fileA.txt, etc.
3  ls [a-c]*          # Files starting with a, b, or c
4  rm *~              # Delete backup files
5  cp /etc/*.conf ./  # Copy all .conf files
```

⚠️ ****Warning:**** Be careful with `rm *` - it deletes everything!

Environment Variables

What are they?

Variables that affect how processes run on your system.

```
1  env                # View all
2  echo $HOME         # View specific
3  MY_VAR="Hello"     # Set variable
4  export MY_VAR="Hello" # Export for child processes
```

Common Variables

- `$HOME` - Home directory
- `$USER` - Username
- `$PATH` - Command search path
- `$SHELL` - Current shell
- `$PWD` - Current directory

More Variables

- `$HOSTNAME` - Computer name
- `$LANG` - Language setting
- `$EDITOR` - Default editor
- `$PS1` - Prompt string

The PATH Variable

What is PATH?

A colon-separated list of directories where the shell looks for commands.

```
1  echo $PATH
2  # /usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

How it works

When you type `ls`, the shell searches directories in PATH order until found.

Modifying PATH

```
1  # Add directory to PATH (at end)
2  export PATH=$PATH:/home/user/bin
3
4  # Add to beginning (higher priority)
5  export PATH=/home/user/bin:$PATH
```

File Permissions Preview

```
1  $ ls -l myfile.txt
2  -rw-r--r-- 1 user group 1234 Jan 28 10:00 myfile.txt
```

Permission Breakdown

```
1  -  rw-  r--  r--
2  |   |   |   |
3  |   |   |   | Others: read only
4  |   |   |   | Group: read only
5  |   |   |   | Owner: read + write
6  |   |   |   | File type (- = regular file)
```



Note: We'll dive deep into permissions in Week 2!

Lab 1 Preview: VM Setup

What you'll do:

1. 🖥️ **Install VirtualBox/VMware** on your laptop
2. 📄 **Download Linux distribution** (Ubuntu Server recommended)
3. ⚙️ **Create and configure VM** (minimum specs provided)
4. 🚀 **First boot and initial setup**
5. 🐉 **Practice shell navigation commands**

Deliverables:

- Screenshot of successful VM boot
- Output of basic commands (`pwd` , `ls` , `whoami`)
- Create a directory structure using `mkdir`

Time Estimate: 2 hours

✅ **Goal:** Have a working Linux environment ready for Week 2!

Tips for Success

Learning Strategies

- 📖 Read error messages carefully
- 🔍 Use `man` pages before Googling
- 💻 Practice daily, even 15 minutes
- 🎯 Type commands, don't copy-paste
- 📝 Keep a command journal
- 🤝 Form study groups



Common Pitfalls

- ❌ Ignoring case sensitivity
- ❌ Forgetting `sudo` when needed
- ❌ Not reading documentation
- ❌ Running `rm -rf` carelessly
- ❌ Hardcoding values in scripts
- ❌ Not testing before submission



🎓 **Remember:** The goal isn't memorization—it's understanding how the system works!

Automated Grading System


How It Works

- All assignments submitted via **GitHub**
- **GitHub Actions** runs automated tests
- Tests use **dynamic inputs** (no hardcoding!)
- You'll see results instantly:  or 

Grading Philosophy

- 1  Green Checkmark = Full Points
- 2  Red Cross = Zero Points (but you can resubmit!)

Key Rule: Code must work on the grading environment, not just "on your machine"

 **Anti-Cheat:** Midterm flags are cryptographically tied to YOUR GitHub username!

Important Dates & Deadlines

Homework Deadlines

- **Homework 1:** Week 3
- **Homework 2:** Week 6
- **Homework 3:** Week 11
- **Homework 4:** Week 14

Quizzes

- **Quiz 1:** Week 7
- **Quiz 2:** Week 12

Major Assessments

- **Midterm Exam:** Week 8
 - "Broken VM" Challenge
 - 4 hours
 - 20% of grade
- **Capstone Project:** Week 15
 - Infrastructure deployment
 - Defense presentation
 - 40% of grade



No Extensions: Late work not accepted. Plan accordingly!

Resources & Support



Course Materials

- **LMS:** All lectures, labs, and assignments
- **GitHub Org:** Starter code and submissions
- **Discussion Forum:** Ask questions, help peers



Getting Help

1. **Office Hours:** TBD (to be announced)
2. **Lab Sessions:** Hands-on assistance during lab time
3. **Discussion Forum:** Community support
4. **Documentation:** Man pages, official docs



Useful Links

- Course GitHub: `github.com/your-org/css262`
- Linux docs: `linux.die.net`
- Bash guide: `mywiki.woledge.org/BashGuide`

Week 1 Action Items

Before Next Lecture:

1. Read **Chapter 1-3** of "The Linux Command Line"
2. Set up **GitHub account** (if you don't have one)
3. Join course **GitHub Organization** (link on LMS)
4. Review VM requirements (8GB RAM minimum)

For Lab This Week:

1. Install **VirtualBox** or VMware
2. Download **Ubuntu Server ISO** (link provided)
3. Bring laptop with **at least 20GB free space**
4. Complete **Lab 1: VM Setup & Shell Navigation**



Optional:

- Explore `man` pages for basic commands
- Try shell navigation on your own

Questions?



Remember: There are no stupid questions!

The shell can be intimidating at first, but you'll be comfortable in no time.

Next Week: Users, Groups & Permissions  

Thank You!



See you in the lab!

Start your Linux journey today 🚀

CSS 262 - Linux Administration & *nix Systems for Cybersecurity