**COMP2221 — Sys. Prog.**

# Bash Scripting: The Unix Admin

Practical 01 • 13 October 2025

Instructor: Dr. Stuart James (stuart.a.james@durham.ac.uk)

## Scenario

You have just started using the University's Linux system. Instead of typing the same commands over and over, you can put them into small text files called *shell scripts* that the computer can run for you. In this session, you will write and run your first Bash scripts to work with files, directories, and simple text processing.

## Learning Objectives

- Command line basics
- Creating and running shell scripts
- Communicating diagrams
- Design and combine commands into a sequence to solve a task.

## Prerequisites

- Access to a UNIX or UNIX-like system (Linux, macOS or `mira.dur.ac.uk` (See Ultra for details) – A general tutorial is provided in Appendix B ).
- Reviewed the lab info sheet Appendix A.
- Basic command-line navigation (`cd`, `ls`, `pwd`).

## Changes

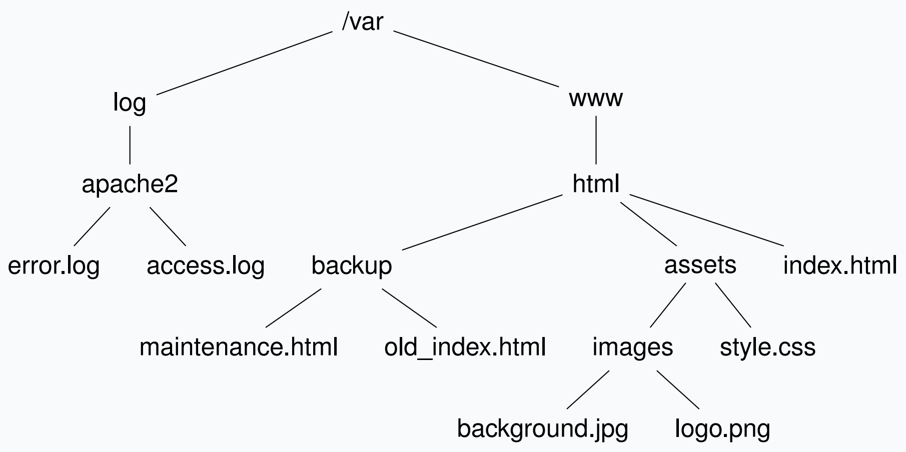
- Added file for uploading Task 6, the translation command.
- Added basic instructions for accessing terminal (Appendix B).
- Added missing output for Task 3, dictionaries (`dictionary.txt`)

## Tasks

### Task 1: Directories

SOLO INTRO 10m

Consider this Apache web server directory structure on a Linux system. You may assume that anything without a file extension is a directory, and files ending in `.conf` are configuration files.



As a systems administrator, you start in the `/var/www/html/` directory and execute the following maintenance commands:

```
mkdir temp_maintenance
mv index.html temp_maintenance/
cp backup/maintenance.html .
mv maintenance.html index.html
cd assets
rm *.css
cd ../backup
rm old_index.html
cd ..
mkdir config
cd config
touch apache_backup.conf
```

▶ **Activity**

**New Structure Diagram:** Draw the directory structure again, as it would be on completion of the above commands.

▶ **Activity**

**Structure Change Diagram:** Now draw a diagram that would indicate how the original structure has changed to the new structure. Consider how this could be communicated efficiently and concisely without excessive explanation.

**? Quiz**

What would the output be if the administrator ran `pwd` immediately after the above commands?

## Task 2: Sketch testing

PARTNER INTRO 10m

Take the **Structure Change Diagram** you did in **Task 1**. Swap them with a partner who is not sat next to you.

**◇ Reflection**

**Reflect:** Individually reflect on each others diagrams. Is it easy to understand? Is there excessive amounts of text? If you did not know the task, would you understand?

**▶ Activity**

**Discuss:** Talk to each other what you found worked, what you found hard to understand. Consider being constructive avoid statements that do not suggest an alternative. Discuss each other points.

**▶ Activity**

**Redraw (Final Structure Diagram):** Based on the reflection and discussion redraw your diagram. Then take a picture save it in your folder for submission call it Final-Structure-Diagram.jpg (remember to convert if you have a different format).

## Task 3: Dictionaries

SOLO INTRO 20m

A university librarian has contacted you asking for help with a research project. They need to compute a comprehensive word dictionary from texts in their digital catalogue to assist a literature researcher. As a proof of concept, they have provided Jane Austen's *Pride and Prejudice* and asked you to generate a complete dictionary of unique words from this novel.

**▶ Activity**

You will use a pipe of `tr` commands to extract and process the individual words. The file `1342.txt.utf-8.txt` contains a copy of the novel from the library's digital collection. The steps in the pipe are as follows:

- Convert all newline characters to spaces.
- Delete all non-printable characters.
- Delete all non-word characters: `0-9`, `_`, `"`, `;`, `{`, `}`, `[`, `]`, `(`, `)`, `-`, `!`, `:`, `#`, `$`, `%`, `@`, `^` `&`.
- Convert spaces to newline characters.
- Convert all the letters to lower case.
- Sort the words.
- Keep only the unique words.
- Redirect the output to dictionary.txt

Create a file with your one-line solution called `unique_words.sh`.

## Task 4: Scripts

SOLO INTERMEDIATE 30m

As a systems engineer, you need to automate common maintenance and monitoring tasks. Write the following shell scripts to help with daily operations:

**▶ Activity**

**Log File Analysis Script (`count_chars.sh`):** Write a shell script that analyzes log files in the current directory. For all files in the current directory, the script should print out the string *"The number of characters in the file"* followed by the filename and the character count. This helps systems engineers quickly assess log file sizes for disk space management and identify unusually large log files that might indicate system issues.

**▶ Activity**

**Threshold Monitoring Script (`check_number.sh`):** Write a shell script that takes a single parameter (representing a system metric like CPU usage, memory usage, or disk space percentage) and prints out *"This number is bigger than 10"* or *"This number is not bigger than 10"* depending on whether the parameter exceeds the threshold. This type of script is fundamental for automated system monitoring and alerting.

▶ **Activity**

**Configuration Backup Management Script (`text_files.sh`):** Write a shell script for managing configuration file backups. The script should:

- Take a single parameter and create a directory called `MyTextFiles` if it doesn't exist
- If parameter is `1`: copy all text files (handling case variations: `.txt`, `.Txt`, `.TXT`) from current directory to `MyTextFiles` (backup operation)
- If parameter is not `1`: move all text files from `MyTextFiles` back to current directory (restore operation)
- If restoring and `MyTextFiles` doesn't exist: display a warning
- Generate a directory listing piped to `DirectoryListing.txt` and display the line count

This script simulates common backup and restore operations that systems engineers perform for configuration management.

## Task 5: Optimise

WITH AI OPTIONAL INTERMEDIATE 10m

Your shell scripts from the previous task have been approved for deployment across the organization's server infrastructure. Before releasing them to the production environment where they'll be used by junior administrators and automated systems, you need to ensure they meet enterprise-grade standards for reliability, efficiency, and maintainability.
Using Microsoft Copilot, conduct a professional code review to optimize your scripts for production deployment. This mirrors real-world practices where AI tools assist in code quality assurance before enterprise rollouts.

▶ **Activity**

**Production-Ready Script Analysis:** For each of your three scripts (`count_chars.sh`, `check_number.sh`, `text_files.sh`):

1. Present your script to Copilot with context: *"This shell script will be deployed across multiple production servers and used by junior system administrators. Please critique it for enterprise-grade reliability, efficiency, error handling, and maintainability. Suggest improvements for production deployment."*
2. Demand evidence-based recommendations: *"For each suggestion, provide references to established shell scripting standards, security best practices, or official documentation that justifies the change."*
3. Focus on operational concerns: *"What failure modes, edge cases, or operational issues could cause problems when this script runs in automated environments or is used by less experienced administrators?"*

**▶ Activity**

**Enterprise Optimization Implementation:**

- Choose **one** script for production hardening based on AI feedback
- Implement critical improvements focusing on: error handling, input validation, logging, and operational robustness
- Create the production version: `production_[scriptname].sh`
- Add comprehensive header comments including: purpose, usage, dependencies, and maintenance notes
- Document all modifications with inline comments explaining the production considerations

**▶ Activity**

**AI-Assisted Code Review Reflection:** Write a 75-100 word reflection addressing:

- How did framing the task as "production deployment" change the AI's recommendations?
- Which enterprise-focused suggestions were most valuable for operational reliability?
- What production concerns did you choose not to implement and why?
- How has this process informed your understanding of using AI for enterprise-grade development practices?

Save this as `production_review_reflection.txt`.

## Task 6: Translating

SOLO INTERMEDIATE 15m

As a systems administrator, you've inherited a legacy hardware inventory system that uses a proprietary data format. The facilities management team needs this data converted to a standard CSV format so it can be imported into their new asset tracking database. The legacy system used special symbols and formatting that need to be standardized.
Input file (`legacy_inventory.dat`):

```
! LEAF 4.5 56 BROWN
@ NEEDLE 3.0 45 SILVER
$ DESK 104.0 453 WHITE
% CHAIR 56.5 124 MAGNOLIA
```

Required output format for database import:

```
1,leaf,4.5,56,brown
2,needle,3.0,45,silver
3,desk,104.0,453,white
4,chair,56.5,124,magnolia
```

Write a pipeline using `tr` commands to convert the legacy format to the required CSV format. Analyze the input and output patterns to determine what transformations are needed. Put the single-line command in a file called `translating.sh`.

## Task 7: System Admin Challenge

ASSESSED HARD 45m

You are the lead systems administrator for a small web hosting company. The company has just acquired another hosting provider and you need to integrate their legacy server monitoring system into your infrastructure. This requires creating shell scripts that process log files, manage backups, and generate reports with exact output specifications for automated evaluation.

### Important

This assessment uses automated grading. Your scripts must produce output that exactly matches the expected format. Use the provided test files to verify your solutions before submission.

### Note on Generative AI

Remember you can use Generative AI (Microsoft Copilot), however, it is suggested to use it to assist you not write the tasks. Think of it as a coding partner providing critical reflection. Demonstrate your understanding and consideration of other people's code by including references where you have used AI assistance.

### ► Activity

**Script 1: Legacy Log Processor (`log_processor.sh`):** Create a script that takes one argument (input log file path) and processes it according to these specifications:
**Input format:** Each line contains: `SYMBOL|TYPE|DATE|TIME|ISSUE|VALUE|SERVER` **Processing requirements:**

1. Convert symbols: `# to 1` (Critical), `* to 2` (Warning), `@ to 3` (Info)
2. Replace all pipe delimiters (`|`) with commas (`,`)
3. Sort entries by time (newest first): compare TIME field as `HH:MM:SS`
4. Add CSV header as first line: `Priority,Type,Date,Time,Issue,Value,Server`
5. Output to file named `processed_logs.csv`

**Usage:** `./log_processor.sh test_data/legacy_server.log`

**► Activity**

**Script 2: Backup File Counter (`backup_counter.sh`):** Create a script that takes one argument (directory path) and generates a backup summary:
**Processing requirements:**

1. Count all files ending in `.conf` and `.log` in the specified directory
2. Calculate total size of these files in bytes
3. Output exactly three lines to `backup_summary.txt`:
   - Line 1: `TOTAL_FILES: X` (where X is the count)
   - Line 2: `TOTAL_SIZE: Y` (where Y is total bytes)
   - Line 3: `SCAN_COMPLETED: YYYY-MM-DD HH:MM:SS` (current timestamp)

**Usage:** `./backup_counter.sh test_data/`

**► Activity**

**Script 3: Error Counter (`error_counter.sh`):** Create a script that takes one argument (CSV file path) and analyzes error patterns:
**Processing requirements:**

1. Read the processed CSV file (skip header line)
2. Count occurrences by Priority level (1, 2, 3)
3. Count occurrences by Issue type
4. Output exactly 6 lines to `error_analysis.txt`:
   - `CRITICAL: X` (count of Priority 1)
   - `WARNING: Y` (count of Priority 2)
   - `INFO: Z` (count of Priority 3)
   - `MOST_COMMON_ISSUE: ISSUE_NAME` (most frequent issue type)
   - `TOTAL_ENTRIES: N` (total non-header lines processed)
   - `ANALYSIS_DATE: YYYY-MM-DD` (current date)

**Usage:** `./error_counter.sh processed_logs.csv`

**▶ Activity**

**Script 4: System Integration (`run_all.sh`):** Create a master script that executes all three scripts in sequence:

**Processing requirements:**

1. Execute: `./log_processor.sh test_data/legacy_server.log`
2. Execute: `./backup_counter.sh test_data/sample_files`
3. Execute: `./error_counter.sh processed_logs.csv`
4. Output success message: `ALL_SCRIPTS_COMPLETED: YYYY-MM-DD HH:MM:SS`

**Usage:** `./run_all.sh`

**Deliverables:** Submit all four scripts (`.sh` files) that pass automated testing against provided test data.

## Submission Checklist

Submission Checklist

- ☐ `Final-Structure-Diagram.jpg` is included and clear.
- ☐ `unique_words.sh` outputs unique words from Pride and Prejudice.
- ☐ `count_chars.sh` counts characters for each file.
- ☐ `check_number.sh` makes the correct threshold decision.
- ☐ `text_files.sh` creates folder, copies files, lists them.
- ☐ `production_[scriptname].sh` - hardened version of one script (optional).
- ☐ `production_review_reflection.txt` - AI code review reflection (optional).
- ☐ `translating.sh` - translation command
- ☐ `log_processor.sh` - processes legacy logs to CSV format.
- ☐ `backup_counter.sh` - counts backup files and calculates sizes.
- ☐ `error_counter.sh` - analyzes error patterns in CSV data.
- ☐ `run_all.sh` - master script executing all assessment scripts.

Place all the above files in a single folder named `Practical1` and submit it via the coursework submission system.

The weekly structure can be assumed to be:

```
Practical1/
|-- unique_words.sh
|-- count_chars.sh
|-- check_number.sh
|-- text_files.sh
|-- production_count_chars.sh
|-- production_review_reflection.txt
|-- translating.sh
|-- log_processor.sh
|-- backup_counter.sh
|-- error_counter.sh
\-- run_all.sh
Practical2/
Practical3/
Practical4/
Practical5/
Practical6/
Practical7/
Practical8/
Practical9/
README.md
```

## Appendix A Task Tag Reference

Task Tag Guide

Each task in this worksheet includes small coloured *tags* that give you extra context about the task requirements and expectations:

**Approach:**

| | |
|---|---|
| **SOLO** | This task is intended to be attempted individually. Work independently to develop your own understanding and solutions. |
| **PARTNER** | This task is designed for collaboration with a partner. Share ideas, discuss approaches, and learn from each other. |
| **WITH AI** | This task explicitly incorporates AI assistance (e.g., Microsoft Copilot). Use AI as a coding partner for critique and optimization. |
| **ASSESSED** | This task contributes to your course grade. Follow requirements precisely and ensure high-quality submissions. |
| **OPTIONAL** | This task is optional and can be skipped if time is limited. Recommended for students wanting extra practice or deeper understanding. |

**Difficulties:**

| | |
|---|---|
| **INTRO** | Introductory level - focuses on basic concepts and simple implementations. |
| **INTERMEDIATE** | Intermediate level - requires combining concepts and problem-solving skills. |
| **HARD** | Advanced level - involves complex integration and production-ready considerations. |

**Guidance:**

| | |
|---|---|
| **10m** | Estimated completion time in minutes. Use this to plan your session effectively. |

All tags are displayed on the same line as the task heading, aligned to the right for quick visual scanning and session planning.

# Appendix B Using the Terminal Across Operating Systems

In this module, you will use the Unix terminal to run commands, write shell scripts, and connect to the University's remote Linux system. While the practicals assume a Linux-style environment, there are several ways to access this from your own computer depending on your operating system.

## Accessing Mira (Durham Remote Linux Server)

Durham University provides a remote Linux environment known as **mira**. You can log in securely using the `ssh` (Secure Shell) command, which is available on all modern systems.

- **On macOS or Linux:** Open your Terminal and type:

  ```
  ssh ab12cd@mira.dur.ac.uk
  ```

  Replace `ab12cd` with your CIS username.
- **On Windows (with WSL):** Open your WSL (e.g., Ubuntu) terminal and run the same command:

  ```
  ssh ab12cd@mira.dur.ac.uk
  ```

- **On other systems:** If you are unable to use a native terminal, you can connect using an SSH client such as **PuTTY** (available for Windows).

After entering your CIS password, you will likely be asked for your authenticator code, which you can get from your normal Microsoft Authenticator app. After which, you will be logged into Mira's Linux environment. This is the same setup used in the labs and by all teaching materials. Files saved on Mira remain available between sessions and can be accessed both on and off campus.

### Uploading and Downloading Files with Mira

Sometimes you will want to transfer files between your own computer and Mira — for example, uploading a script you have written locally or downloading output from your experiments. There are two easy command-line methods to do this: `scp` (secure copy) and `sftp` (secure file transfer).

**Using `scp`** The `scp` command lets you copy files directly over SSH. You can use it in either direction.

- **Upload a file to Mira:**

  ```
  scp myscript.sh ab12cd@mira.dur.ac.uk:~/COMP2221/
  ```

  This uploads `myscript.sh` from your current directory to the folder `COMP2221` in your Mira home directory.
- **Download a file from Mira:**

```
    scp ab12cd@mira.dur.ac.uk:~/results/output.txt .
```

The final dot (`.`) means "save it to my current local directory".

You can also copy entire directories using the `-r` (recursive) flag:

```
scp -r ab12cd@mira.dur.ac.uk:~/data ./data_copy
```

**Using `sftp`** If you prefer an interactive approach, `sftp` works like a remote file browser within the terminal.

```
sftp ab12cd@mira.dur.ac.uk
```

Once connected, you can use simple commands:

```
ls          # list files on Mira
cd lab02    # change remote directory
get file.txt    # download a file
put script.sh   # upload a file
bye         # exit
```

**Tip: Graphical Clients** If you prefer a graphical interface, you can use software such as:

- **FileZilla** or **Cyberduck** (macOS/Windows) — connect via `sftp://mira.dur.ac.uk`
- ForkLift (macOS) — connect via `sftp://mira.dur.ac.uk`
- **VS Code Remote SSH** extension — edit and transfer files directly within VS Code.

All of these methods use the same secure SSH connection, so whichever you choose, your data remains protected.

## Using the Terminal on macOS

macOS includes a built-in Unix terminal. You can find it under:

`Applications → Utilities → Terminal`

By default, macOS uses the **Z shell (zsh)**. Our lecture examples and practical tasks are written for the **Bourne Again Shell (bash)** for simplicity and consistency. To switch temporarily to `bash`, type:

```
bash
```

To make `bash` your default shell (not recommended), you can run:

```
chsh -s /bin/bash
```

Once set, opening a new terminal window will place you directly into the `bash` shell, matching the environment used in lectures and on Mira.

## Using the Terminal on Windows

Windows users can obtain a Linux-style terminal by installing the **Windows Subsystem for Linux (WSL)**. This provides an environment very similar to Mira's, allowing the same commands and scripts to run.

- Install WSL by running the following in PowerShell (as Administrator):

```
wsl --install
```

- After restarting, open the "Ubuntu" application from the Start Menu.
- Inside this terminal, you can use the same commands as in the practicals, including:

```
ssh ab12cd@mira.dur.ac.uk
```

**Compatibility note:** While WSL provides good compatibility, certain Linux system behaviours (such as process management or file permissions) may differ slightly. You should therefore test your scripts on **Mira** before submission to ensure they behave as expected in the assessment environment.

## Summary

| **System** | **Recommended Shell** | **Notes** |
|---|---|---|
| Mira (Durham) | bash | Fully supported Linux environment |
| macOS | bash (instead of zsh) | Switch shell for alignment with teaching |
| Windows | bash via WSL | May have minor compatibility differences |

Regardless of your operating system, using a **bash-compatible shell** will ensure your experience matches the lecture content and Mira environment.