COMPSCI 1XC3

Lab 3-1 — Shell Expansion & Permissions

Before We Start — Important Setup

Your commands must run in a **Unix shell**.

Windows users

Use **Ubuntu (WSL)** in VS Code:

- Terminal → New Terminal, Select **Ubuntu (WSL)**
- Run:

```
cd ~
```

This ensures you are working in the **Linux filesystem**, not the Windows filesystem.

macOS / Linux users

You're good to go 👍

Verify Your Shell (Everyone)

Run this command:

```
echo $SHELL
```

Expected:

- `/bin/bash` → ✅ correct
- `/bin/zsh` (macOS) → ✅ also fine
- `/bin/sh` or blank → ❌ switch shells

Create Your Lab Workspace

Run the following commands:

```
mkdir -p ~/1xc3/lab3-1
cd ~/1xc3/lab3-1
```

Install the C Compiler (Once)

Before compiling C programs, make sure `gcc` is installed.

Run:

```
sudo apt update
sudo apt install gcc
```

Verify installation:

```
gcc --version
```

Create `myCode.c`

```
cat > myCode.c << 'EOF'
#include <stdio.h>

int main() {
    printf("Welcome to 1XC3!\n");
    return 0;
}
EOF
```

Agenda

1. Quick shell concepts
2. Live examples
3. Lab setup
4. TA check-off tasks

Today’s Goal

By the end of this lab, you should be able to:

- Predict how the shell expands text
- Control expansion using quotes
- Explain permission errors
- Debug “why doesn’t this work?”

How the Shell Works (Big Picture)

Before a command runs, the shell may:

- Expand wildcards (`*` , `?` , `[ ]`)
- Expand variables (`$USER`)
- Run commands inside commands (`$(...)`)
- Generate multiple strings (`{}`)

➡️ All of this happens **before** the command runs.

Demo: Pathname Expansion

Try this:

```
touch file{1..3}.txt
echo *.txt
echo "*.txt"
```

What's the difference?

Demo: Multiple Expansions at Once

```
echo ~/$USER
echo $(pwd)
echo file_{1..3}.txt
```

Each line uses a different expansion.

Quoting Controls Expansion

Compare:

```
echo The total is $100
echo "The total is $100"
echo 'The total is $100'
```

Why do these behave differently?

Permissions Preview

```
ls -l myCode.c
chmod 000 myCode.c
cat myCode.c
gcc myCode.c
```

Same file — different behavior. Why?

Restore permissions

```
chmod 644 myCode.c
```

TA Check-off — Task 0 (Warm-up)

Environment Check

Question:

Using **one command**, print:

```
User: <your username> | Dir: <current directory>
```

Rules:

- One command

TA Check-off — Task 1

Brace Expansion

Question:

Using **one command**, create:

```
report_1.txt report_2.txt report_3.txt report_A.txt  report_B.txt  report_C.txt
```

Then verify they exist.

TA Check-off — Task 2

Pathname Expansion (Character Classes)

Question:

With the files you created, using **one command**, show **only the files that contain at least one uppercase letter**.

💡 Hint: POSIX character classes exist.

TA Check-off Task 3

Arithmetic Expansion with Command Substitution

Question:

You have several files in your current directory.

Using **one command**, do the following:

1. Count how many files are in the current directory
2. Add **10** to that number
3. Print the result

💡 Hint:

- You will need **command substitution**
- You will need **arithmetic expansion**

TA Check-off — Task 4

Command Substitution (System Inspection)

Question:

Using **one command**, display the **long listing** of the `cp` executable.

💡 Hint: one command can produce input for another.

TA Check-off — Task 5

Permissions (Concept Check)

You are given `myCode.c`.

A. Remove **all permissions** so that:

- the file still exists
- it cannot be read or written to

B. Restore **read-only** permission.
Show that:

- the file can now be read
- the file still cannot be written to

C. Restore normal permissions, then compile and run the program.