Lab 5 – Bash/Python Integration

Week 13.5

Integrating Python with Bash Scripts

- Combining Python and Bash: Calling Linux commands from Python using subprocess, and the os modules.
- Passing variables and arguments between Python and Bash.
  - When you run a Bash command from Python using subprocess, you pass arguments to the Bash script or command. These arguments are typically passed as a **list** to **`subprocess.run()`** or similar functions.
- Passing Python Variables to Bash Commands
  - You can format Python variables into strings that form Bash commands, which can then be executed.

Example are coming

The subprocess Module

- Purpose: Spawning and managing new processes in Python.
- Key Function: subprocess.run() for executing shell commands.

```
import subprocess

# Executing the 'ls' command
result = subprocess.run(['ls', '-l'], capture_output=True, text=True

# Printing the standard output
print("Output:")
print(result.stdout)

# Check if there was any error
if result.stderr:
    print("Error:")
    print(result.stderr)
```

Subprocess for User Management

- Implementing Linux commands (useradd, groupadd, passwd) in Python.
- Handling command output and errors.

```
import subprocess

def add_user(username):
    try:
        # Running the useradd command to add a new user
        subprocess.run(['sudo', 'useradd', username], check=True)
        print(f"User '{username}' successfully added.")
    except subprocess.CalledProcessError:
        print(f"Failed to add user '{username}’.”)

# Example usage
add_user("newuser")
```

The os Module

- The os module in Python provides a way of using operating system dependent functionality.
- It includes a wide range of functions to interact with the underlying operating system, filesystems, and processes.
- The os module has not been deprecated, however, certain functions in the module have been replaced.
  - For example, **`os.path`** has been replaced by functions in the **`pathlib`** module.
  - **`os.system()`** is still available but it's generally recommended to use the **`subprocess`** module.

The os Module – File and Directory Manipulation

- os.listdir(path): Returns a list of the entries in the directory given by path.
- os.mkdir(path): Creates a directory named path with default mode 0o777.
- os.makedirs(path): Recursive directory creation function. Like mkdir(), but makes all intermediate-level directories needed to contain the leaf directory.
- os.remove(path): Removes (deletes) the file path.
- os.rename(src, dst): Renames the file or directory from src to dst.

The os Module – Path Manipulation

- os.path.join(path1, path2, ...): Joins one or more path components intelligently.
- os.path.split(path): Splits the pathname path into a pair, (head, tail).
- os.path.exists(path): Returns True if path refers to an existing path.
- os.path.isfile(path): Returns True if path is an existing regular file.

The os Module – Working with Environment Variables

- os.environ: A mapping object representing the string environment.
- For example, os.environ['HOME'] would return the user's home directory.
- os.getenv(key, default=None): Returns the value of the environment variable key if it exists, or default if it doesn’t.

The os Module – Process Parameters

- os.getpid(): Returns the current process’s ID.
- os.getppid(): Returns the parent process’s ID.

The os Module – File Descriptors and Low-Level File I/O

- os.open(file, flags[, mode]): Opens the file file and sets various flags according to flags and possibly its mode according to mode.
- os.read(fd, n): Reads at most n bytes from file descriptor fd.
- os.write(fd, str): Writes the string str to file descriptor fd.

The os Module – Miscellaneous

- Executing System Commands
  - os.system(command): Executes the command (a string) in a subshell.
  - This is a simple way to run a command like in a shell, but subprocess is more versatile.
- System Information
  - os.name: The name of the operating system dependent module imported. The following names have currently been registered: 'posix', 'nt', 'java’.
  - os.uname(): Returns information identifying the current operating system (not available on all platforms).

Reading and Parsing CSV Files

- Using Python's csv module for file reading and parsing.
- Handling data formats and issues in CSV files.

Lab 5 – Wrap Up

- Error Handling and Reporting
  - Importance of robust error handling in scripts.
  - Techniques for error detection and reporting.
- Best Practices
  - Essential practices (shebang lines, executable permissions).
  - Emphasis on clean, readable, and Pythonic code.
- Testing and Debugging
  - Testing script on Rocky 8.
  - Debugging strategies for Python scripts.