2. Student Handout

Student Handout: Understanding Processes in Linux

Introduction to Processes

- A **process** is a running instance of a program in Linux.
- Each process has a unique **Process ID (PID)**.

Foreground vs. Background Processes

Foreground Processes

- **Definition**: Runs directly in the terminal and takes control of it.
- **Example 1**: Running `cat largefile.txt` displays the file content and occupies the terminal.
- **Example 2**: Executing `nano myfile.txt` opens a text editor in the terminal.
- **Example 3**: Using `python script.py` runs a Python script and holds the terminal until completion.

Background Processes

- **Definition**: Runs in the background, allowing terminal use for other tasks.
- **Example 1**: `sleep 60 &` runs the sleep command for 60 seconds in the background.
- **Example 2**: `wget http://example.com/file.zip &` downloads a file without blocking the terminal.
- **Example 3**: `find / -name "*.txt" &` searches for text files across the system in the background.

Managing Processes

Bringing a Foreground Process to the Background

1. **Pause the process**: Use `Ctrl + Z`.

2. **Send to background**: Use `bg` command.

- **Example**:

  ```
  $ cat largefile.txt
  # Press Ctrl + Z
  $ bg
  ```

Bringing a Background Process to the Foreground

- Use the `fg` command to bring the most recent background process to the foreground.
- **Example**:

  ```
  $ fg
  ```

Checking Running Processes

- **Command**: `ps` shows processes in the terminal.
- **Command**: `ps aux` lists all system processes.
- **Example 1**: `ps` displays processes with their PIDs.
- **Example 2**: `ps aux | grep python` filters processes related to Python.
- **Example 3**: `ps -ef` provides a full-format listing of all processes.

Killing a Process

- **Command**: `kill <PID>` terminates a process.
- **Force Kill**: `kill -9 <PID>` forcefully kills a process.
- **Example 1**: `kill 1234` terminates the process with PID 1234.
- **Example 2**: `kill -9 5678` forcefully kills the process with PID 5678.
- **Example 3**: `pkill firefox` kills all processes related to Firefox.

Shell Scripting Basics

Creating a Simple Shell Script

1. **Write Commands**: Use a text editor to write commands.
2. **Save File**: Save as `myscript.sh`.
3. **Make Executable**: Use `chmod +x myscript.sh`.
4. **Run Script**: Execute with `./myscript.sh`.

- **Example Script**:

```
#!/bin/bash
echo "Hello, World!"
sleep 5
echo "This is a simple shell script."
```

- **Example 1**: A script to list files:

```
#!/bin/bash
ls -l
```

- **Example 2**: A script to display current date and time:

```
#!/bin/bash
date
```

- **Example 3**: A script to create a directory:

```
#!/bin/bash
mkdir new_directory
```

Conclusion

- **Processes**: Running instances of programs with unique PIDs.
- **Foreground vs. Background**: Foreground takes terminal control; background allows multitasking.
- **Process Management**: Use `ps`, `bg`, `fg`, and `kill` to manage processes.
- **Shell Scripting**: Automate tasks by writing commands in a script.

Feel free to ask questions if you need further clarification on any topic!