# Funlab-processes

### Exercise 1: Listing and Understanding Processes

1. **List All Processes**:
   - **Command**: To list all currently running processes for the current user, use:

```
ps
```

To list all processes running on the system:

```
ps aux
```

or

```
ps -ef
```

   - **Assignment**:
     - Explain which options show all processes.
     - Describe what each column (e.g., `PID`, `USER`, `%CPU`, `%MEM`) represents.

2. **Process Information**:
   - **Command**:

```
ps aux
```

   - **Assignment**:
     - Identify the process consuming the most CPU and memory.
     - Describe the purpose of each column in the output.

3. **Filtering by Process ID**:
   - **Command**: To display information about a specific process:

```
ps -p <PID>
```

   - **Assignment**:
     - Show how to display details for a process with a specific PID (e.g., 1).
     - Describe the information obtained.

### Exercise 2: Parent-Child Process Relationships

1. **Identify Parent Processes**:
   - **Command**: Use the `-o` option to display parent process IDs:

```
ps -o pid,ppid,cmd
```

   - **Assignment**:
     - Find and list the parent process of a specific command.
     - Identify the parent-child relationship based on the `PPID` column.

2. **Tree View of Processes**:
   - **Command**: To view processes in a tree structure:

```
pstree
```

     Or, if `pstree` is not available:

```
ps -ef --forest
```

   - **Assignment**:
     - Analyze the tree structure and locate the parent process of your shell.

### Exercise 3: Job Control in the Shell

1. **Running Jobs in the Background**:
   - **Command**: Start a job in the background:

```
sleep 300 &
```

   - **Assignment**:
     - List all jobs running in the background.

2. **Bringing Jobs to the Foreground**:
   - **Command**: Use `fg` to bring a background job to the foreground:

```
fg %<job_number>
```

   - **Assignment**:
     - Start a job in the background, then bring it to the foreground.

3. **Job Suspension and Resumption**:

  - **Command**: Start a long-running command, suspend it, and then resume it in the background:
    - Start:

```
sleep 1000
```

    - Suspend: `Ctrl+Z`
    - Resume in background:

```
bg
```

  - **Assignment**:
    - Experiment with suspending and resuming a job. Explain how `Ctrl+Z` and `bg` affect the process.

### Exercise 4: Terminating Processes with `kill`

1. **List and Kill a Process**:
  - **Command**: Start a process, find its PID, and terminate it:

```
ps aux | grep sleep
 kill <PID>
```

  - **Assignment**:
    - Identify the process ID of a `sleep` command, then terminate it.

2. **Using Different Kill Signals**:
  - **Command**: Send different signals to a process:

```
kill -SIGINT <PID>
kill -SIGTERM <PID>
kill -SIGKILL <PID>
```

  - **Assignment**:
    - Compare the effects of `SIGINT`, `SIGTERM`, and `SIGKILL`.

3. **Kill by Process Name**:
  - **Command**: Terminate all instances of `sleep` with `pkill`:

```
pkill sleep
```

  - **Assignment**:
    - Discuss the difference between `kill` and `pkill`.

---

1. **Orphan Processes**:
  - **Command**: Start a process and kill its parent:

```
sleep 100 &
exit
```

  - **Assignment**:
    - Observe what happens to the child process and determine its new parent.

3. **Using `nice` and `renice`**:
  - **Command**: Start a high-priority command, then adjust its priority:
    - Start:

```
nice -n 19 yes > /dev/null &
```

    - Change priority:

```
renice 10 <PID>
```

  - **Assignment**:
    - Examine the effect of adjusting process priority using `nice` and `renice`.

---