Operating Systems Reflective Journal Template

Note: You **must** provide evidence for your answers, where appropriate, by inserting examples of Practical work you undertook; for eg: solutions attempted, code written.
You **must** also provide, at the end of this document, your answers to the questions from the Practical Handout.
Please answer **all** questions.

Name Danyil Tymchuk	**Student ID** B00167321
Practical Name/Topic *Workshop on Monitoring Processes in Linux*	**Session Dates** 03/04/25
What did I read for this session (apart from the Practical handout)?	Researching Process Monitoring Tools
What were the main points that were new to me in the Practical?	Process Monitoring
What were two main things I learned?	Process Monitoring
In trying to solve the problems, what were the things I tried?	Nothing was wrong
What went wrong in my initial attempts to solve each problem, and why?	Nothing was wrong
What went right in my final attempts to solve each problem, and why?	Nothing was wrong
How did my learning in the Practical relate to what I'm learning in other parts of the module; for example: in the Lecture	Good
What did we not cover that I expected we should?	It's ok

What was new or surprising to me?	Amost nothing
I am still unsure about…	I am fine with this
Issues that interested me a lot, and that I would like to study in more detail	C programming
What I most liked about this Practical was…	All good
What I most disliked about this Practical was…	All good
How did you feel about how you performed in the Practical?	Good
Miscellaneous interesting facts I learned in this Practical…	null

Please insert your answers from the Practical Handout here (use as many pages as you need):

3.2: principal difference (s)

- parentchild is more detailed version of the slide 26 Lecture example:
  - Runs `xeyes` instead of `ls`
  - Has addictional print statements to clarify execution
  - Include sleep(5); to impruve output order
  - implement error handling for fork() failure

4.4: $ ./parentchild

- `Child: I am running...`
- `Parent: I am running...Parent: I am waiting until Child is terminated by the user!`
- `(wait)`

4.: CTRL – Z

- process stopped, but can be resumed later

4.8: $ jobs

- `[1]+ Stopped ./parentchild`

4.11: $ bg ./parentchild

- $ jobs
  - `[1]+ Running ./parentchild &`
- $ fg ./parentchild

- just back to live

4.13: $ ps

- $ ps
  - ```
    PID TTY          TIME CMD
    4160 pts/1    00:00:00 bash
    6033 pts/1    00:00:00 parentchild
    6034 pts/1    00:00:18 xeyes
    8334 pts/1    00:00:00 ps
    ```
- $ ps a
  - ```
    PID TTY      STAT   TIME COMMAND
    1495 tty1     Ssl+   0:49 /usr/libexec/Xorg -nolisten tcp -
    background none -seat seat0 vt1 -auth /var
    2996 pts/0    Ss     0:00 /bin/bash
    4119 pts/0    Sl+    0:16 emacs
    4160 pts/1    Ss     0:00 /bin/bash
    6033 pts/1    S      0:00 ./parentchild
    6034 pts/1    S      0:19 Xeyes
    8385 pts/1    R+     0:00 ps a
    ```

4.15: terminate the process

4.17: $ ps au

- ```
  USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START
  TIME COMMAND
  root      1495  0.7  6.2 415016 254796 tty1    Ssl+ 09:17
  0:54 /usr/libexec/Xorg -nolisten tcp -back
  student   2996  0.0  0.1   9912  5008 pts/0    Ss   09:25
  0:00 /bin/bash
  student   4119  0.2  1.4 134868 57192 pts/0    Sl+  09:39
  0:16 emacs
  student   4160  0.0  0.1   9912  5052 pts/1    Ss   09:39
  0:00 /bin/bash
  student   6033  0.0  0.0   2172   456 pts/1    S    10:30
  0:00 ./parentchild
  student   6034  1.0  0.0  10616  3700 pts/1    S    10:30
  0:27 Xeyes
  student   8766  0.0  0.0   2172   460 pts/1    S+   11:14
  0:00 ./parentchild
  student   8767  0.7  0.0  10616  3924 pts/1    S+   11:14
  0:00 Xeyes
  student   8825  0.0  0.1   9780  4768 pts/2    Ss   11:15
  0:00 /bin/bash
  student   8864  0.0  0.0  10888  2960 pts/2    R+   11:15
  0:00 ps au
  ```

4.24: ./parentcild &

- ```
  USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START
  TIME COMMAND
  root      1495  0.7  6.2 415400 254980 tty1    Ssl+ 09:17
  0:56 /usr/libexec/Xorg -nolisten tcp -back
  student   2996  0.0  0.1   9912  5008 pts/0    Ss   09:25
  0:00 /bin/bash
  student   4119  0.2  1.4 134868 57192 pts/0    Sl+  09:39
  ```

```
0:16 emacs
student   4160  0.0  0.1   9912  5052 pts/1    Ss+  09:39
0:00 /bin/bash
student   6033  0.0  0.0   2172   456 pts/1    S    10:30
0:00 ./parentchild
student   6034  1.0  0.0  10616  3700 pts/1    S    10:30
0:30 Xeyes
student   8825  0.0  0.1   9780  4776 pts/2    Ss   11:15
0:00 /bin/bash
student   8939  0.0  0.0   2172   460 pts/1    S    11:17
0:00 ./parentchild
student   8941  1.1  0.0  10616  3792 pts/1    S    11:17
0:01 Xeyes
student   8961  0.0  0.0   2172   520 pts/1    S    11:18
0:00 ./parentchild
student   8963  1.0  0.0  10616  3788 pts/1    S    11:18
0:01 Xeyes
student   9007  0.0  0.0   2172   456 pts/1    S    11:19
0:00 ./parentchild
student   9009  0.4  0.0  10616  3820 pts/1    S    11:19
0:00 Xeyes
student   9011  0.0  0.0   2172   456 pts/1    S    11:19
0:00 ./parentchild
student   9013  0.7  0.0  10616  3824 pts/1    S    11:19
0:00 Xeyes
student   9015  0.0  0.0   2172   452 pts/1    S    11:19
0:00 ./parentchild
student   9017  0.4  0.0  10616  3692 pts/1    S    11:19
0:00 Xeyes
student   9026  0.0  0.0  10888  2960 pts/2    R+   11:20
0:00 ps au
```

4.28: kill

- $ kill <PID>
  - kill process by id
- $ killall <programname>
  - kill all programs with that name

Researching Process Monitoring Tools

1. Comand-Line Tool: `**htop**`
   - *How to run*: Open a terminal and type `htop`, then press Enter.
   - *Information provided*: Displays a dynamic, color-coded list of running processes, CPU & memory usage, and system load.
   - *Options*:
     - `F6` – Sort processes by selected column.
     - `F9` – Kill a process.
     - `-u username` – Show processes for specific user.
     - `-p PID` – Monitor a specific process by its PID.
2. GUI Tool: `**KsysGuard**` (System Monitor)
   - *How to run*: Open `KsysGuard` from the application menu or run `ksysguard` in terminal.

- *Information provided*: Graphical overview of CPU, memory, and network usage, along with a list of running processes.
- *Options*:
  - `End Process` – Kill a selected process.
  - `Monitor Remote Sysstem` – Connect to reemote machines.
  - `Sort Processes` – Order by name, CPU, memory, or other parameters.
  - `Graphs` – View resouces usage in real time.

3. Choice-Based Tool: `**ps**` (Command-Line)
   - *How to run*: Open a terminal and type `ps aux`.
   - *Information provided*: Lists all running processes with detailes likee PID, CPU & memory usage, and command execution path.
   - *Options*:
     1. `ps -e` – Show all processes.
     2. `ps aux --sort=-%cpu` – Sort precesses by CPU usage.
     3. `ps -o pid,ppid,cmd,%mem,%cpu` – Custom format forr process display.
     4. `ps --forest` – Show a tree structure of parent-cchild processes.