In His Name

Sharif University of Technology

Department of Computer Engineering

Operating Systems

Working with Linux screen Command

&

Parallel Execution of Processes

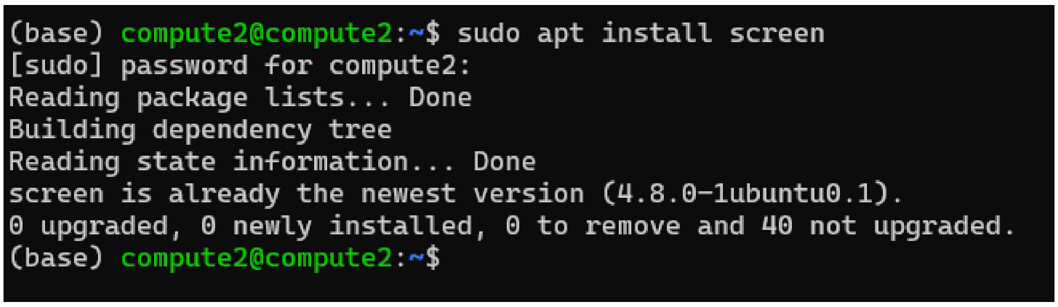
Dr. Hossein Asadi

CE424

Fall 2022

- **Installing screen Command**

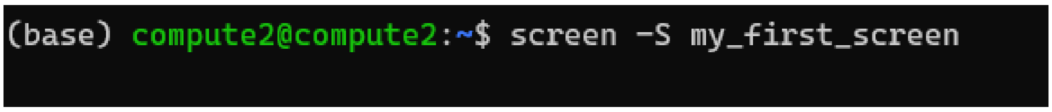
```
~$ sudo apt install screen
```



```
(base) compute2@compute2:~$ sudo apt install screen
[sudo] password for compute2:
Reading package lists... Done
Building dependency tree
Reading state information... Done
screen is already the newest version (4.8.0-1ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 40 not upgraded.
(base) compute2@compute2:~$
```

- **Creating New Screen**

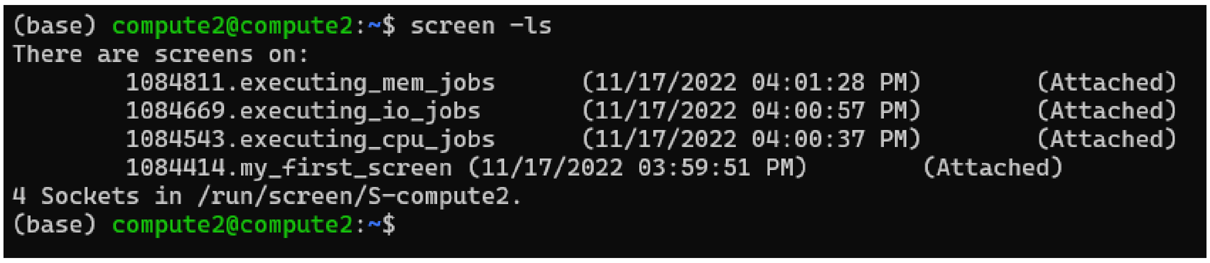
```
~$ screen -S screen_name
```



```
(base) compute2@compute2:~$ screen -S my_first_screen
```

- **List all Screens**

```
~$ screen -ls
```



```
(base) compute2@compute2:~$ screen -ls
There are screens on:
        1084811.executing_mem_jobs      (11/17/2022 04:01:28 PM)        (Attached)
        1084669.executing_io_jobs       (11/17/2022 04:00:57 PM)        (Attached)
        1084543.executing_cpu_jobs      (11/17/2022 04:00:37 PM)        (Attached)
        1084414.my_first_screen (11/17/2022 03:59:51 PM)        (Attached)
4 Sockets in /run/screen/S-compute2.
(base) compute2@compute2:~$
```

- **Attaching (connecting) to a detached Screen**

```
~$ screen -r screen_name
```

- **Detaching (disconnecting) an attached Screen**

```
~$ screen -d screen_name
```

Or

Closing the terminal will detach the screen automatically!

- **Terminating Screen**

```
~$ screen -X -S screen_name quit
```

Or

Simply type 'exit' on the attached screen.

- **Running Parallel Screens Simultaneously**

Now, we're going to run some parallel commands (programs) on separate screens.
Suppose two python programs that write to their output files and print the status after writing each line. Each program must be running on a separate screen. To do this, we need to follow these steps:

1) Create a bash script file for each program (screen)

For each screen, you should write a two-line script (.sh file) that creates a new screen and then starts this screen in the background, running your desired command.
Write the script files as follows:

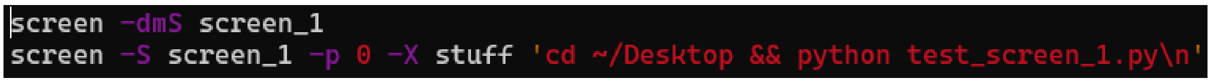
```
nano your_script_file.sh

>> screen -dmS screen_name

>> screen -S screen_name -p 0 -X stuff 'cd your_path && your_command\n'
```

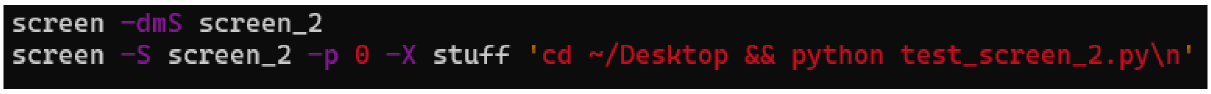
Here, I've to run two parallel programs, so I create 2 scripts:

For the first screen:



```
screen -dmS screen_1
screen -S screen_1 -p 0 -X stuff 'cd ~/Desktop && python test_screen_1.py\n'
```

For the second screen:



```
screen -dmS screen_2
screen -S screen_2 -p 0 -X stuff 'cd ~/Desktop && python test_screen_2.py\n'
```

Then, you must give execution permissions to these script files with the following command:

```
~$ chmod 777 your_script_file.sh
```

2) Run script files for parallel execution

Now that you have created scripts for each screen, it's time to run those programs in parallel. The command for this step is:

```
~$ parallel -u ::: 'bash first_script.sh' ... 'bash last_script.sh'
```

For me, the command looks like this:

```
~$ parallel -u ::: 'bash run_screen_1.sh' 'bash run_screen_2.sh'
```

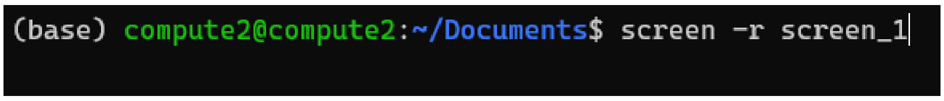
You will notice some screens have been created by the command (check with "screen -ls")

3) Check the execution of each program (screen)

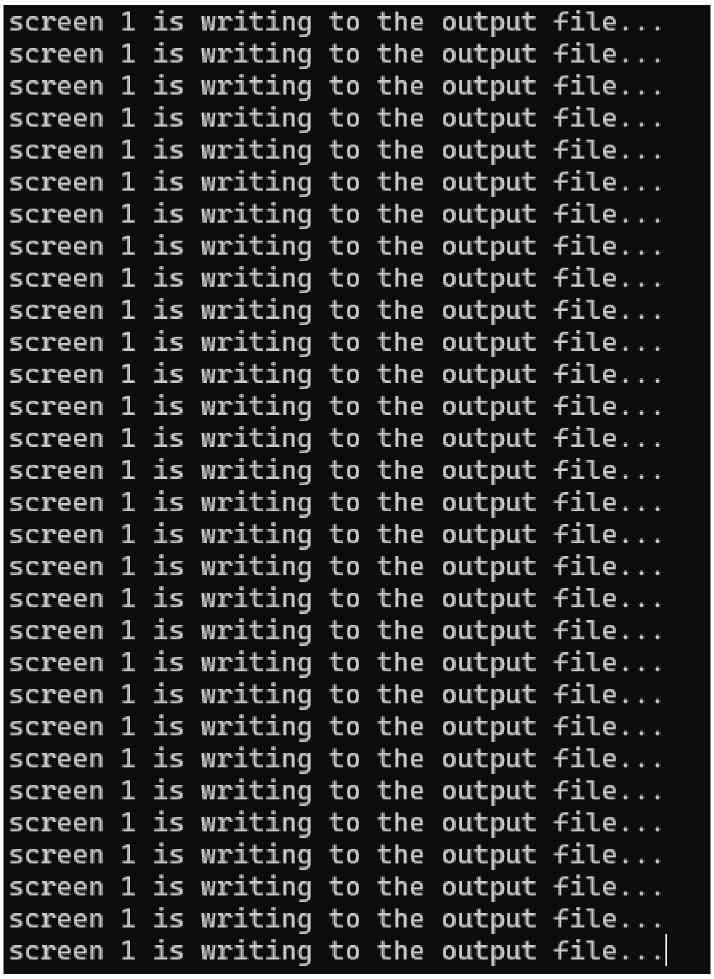
Our programs get started exactly at the same time and are running in the background in parallel. To make sure everything is OK, you can attach to the screen by "screen -r screen_name" on a new terminal and see what's going on!

In my case, 2 programs are running in parallel:

First Screen:

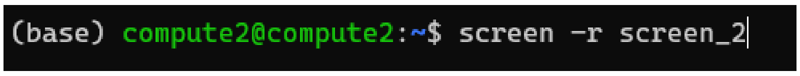


```
(base) compute2@compute2:~/Documents$ screen -r screen_1
```

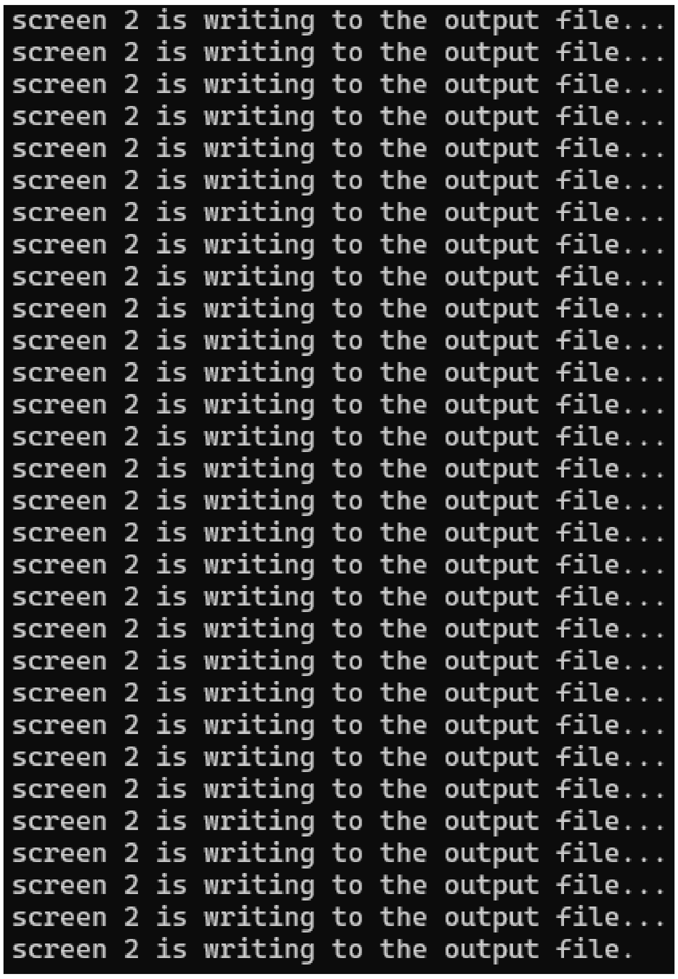


```
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
screen 1 is writing to the output file...
```

Second Screen:

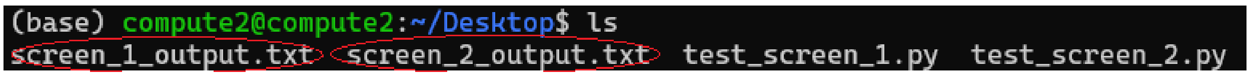


```
(base) compute2@compute2:~$ screen -r screen_2
```



```
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file...
screen 2 is writing to the output file.
```

And files are created successfully!



```
(base) compute2@compute2:~/Desktop$ ls
screen_1_output.txt  screen_2_output.txt  test_screen_1.py  test_screen_2.py
```

Good Luck!