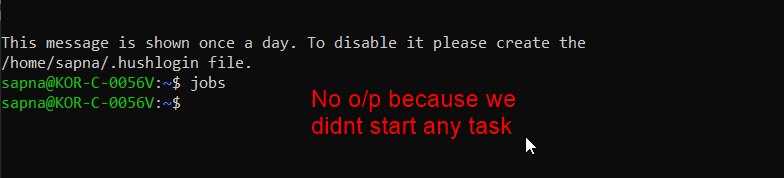
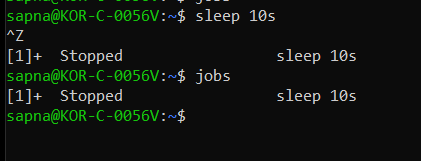
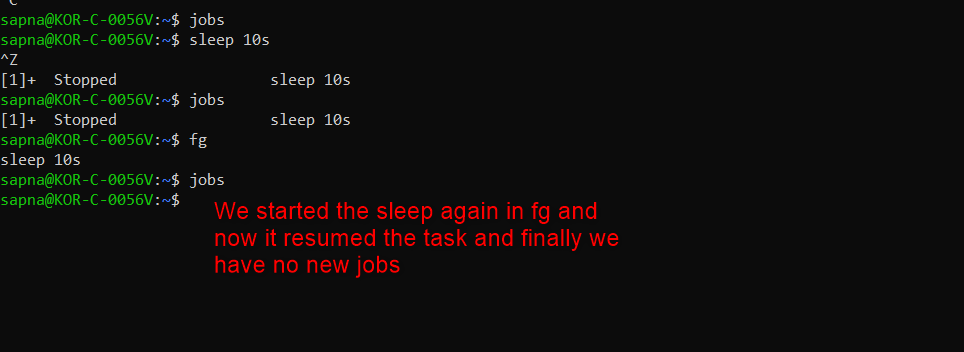
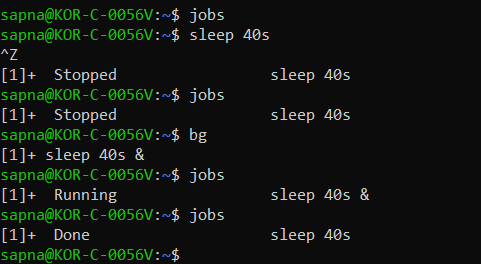
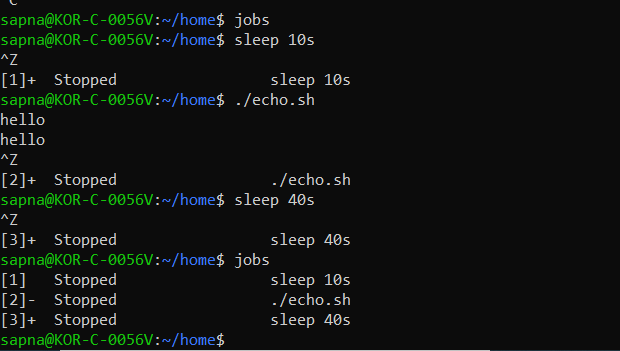
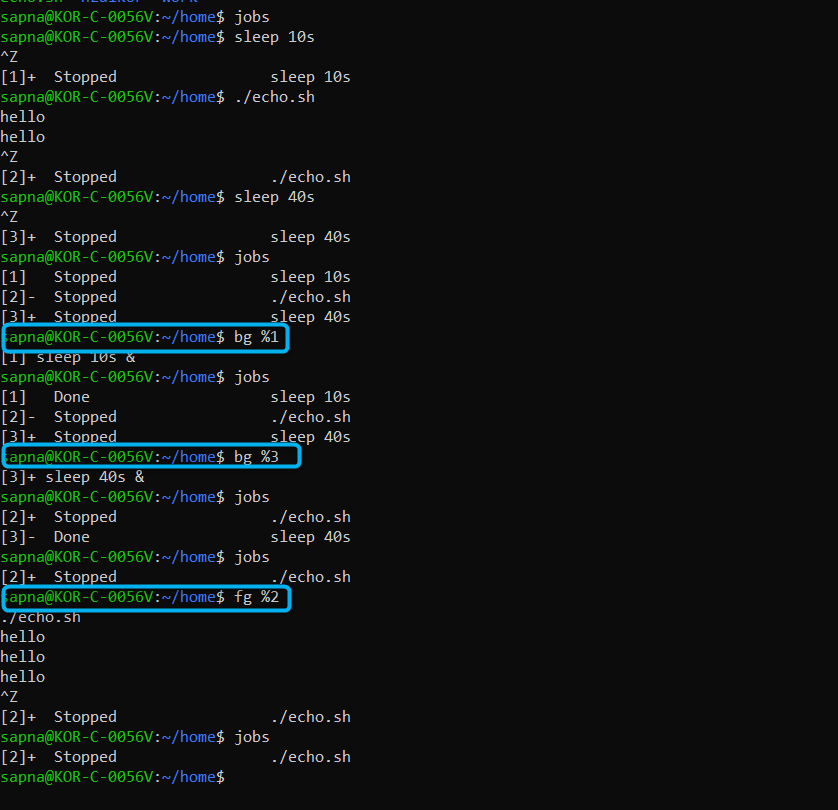
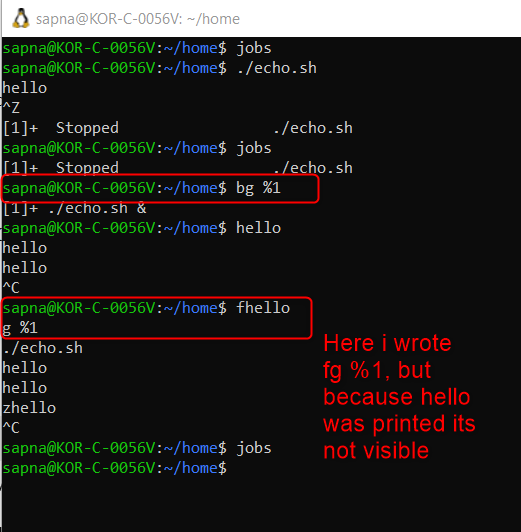
**How to Manage Processes on Linux with nohup, nice, bg, fg, jobs Commands**

* Jobs command is used to show all the active jobs running on a terminal, say I open shell and just run jobs then I wont see any o/p because we didn’t start any task on the terminal
* 
* A very simple example is to use the sleep command
* We give a sleep time of say 10sec, so now the terminal/shell has a task to sleep for 10sec, now say you stop this task in the between using Ctrl + z, and then run jobs, it will show that sleep 10sec job was stopped
* 
* fg command is used to resume jobs to the foreground
* say I start the sleep 10s again and then do a Ctrl + z , now this task, is stopped, say now I want to start it again, I will just run fg, then the stopped sleep task will resume again in the foreground and show us
* 
* bg command is used to resume jobs to the background – lets use the same example here as well, we will start the sleep timer, stop it, then run the bg command, here the only advantage is that this process will now run in the background and our terminal will be free, but we can see that the sleep timer is running in the background with jobs
* 
* Say you have multiple jobs present, like I have started a sleep 10sec, then ran a script echo.sh which print hello after every 2sec, then again a sleep of 40sec, so currently we have 3 jobs.
* 
* Now say I want to start a particular job like echo.sh in the fg, then
* 
* Note that even when you start echo.sh in the bg, the prints will still be visible in the terminal, and you will not be able to terminate it, because this script is running in the background, so to stop or terminate the script you will have to bring it to foreground and then do a ctrl + c or a ctrl +z
* 
* Echo.sh script content –

**cat echo.sh**

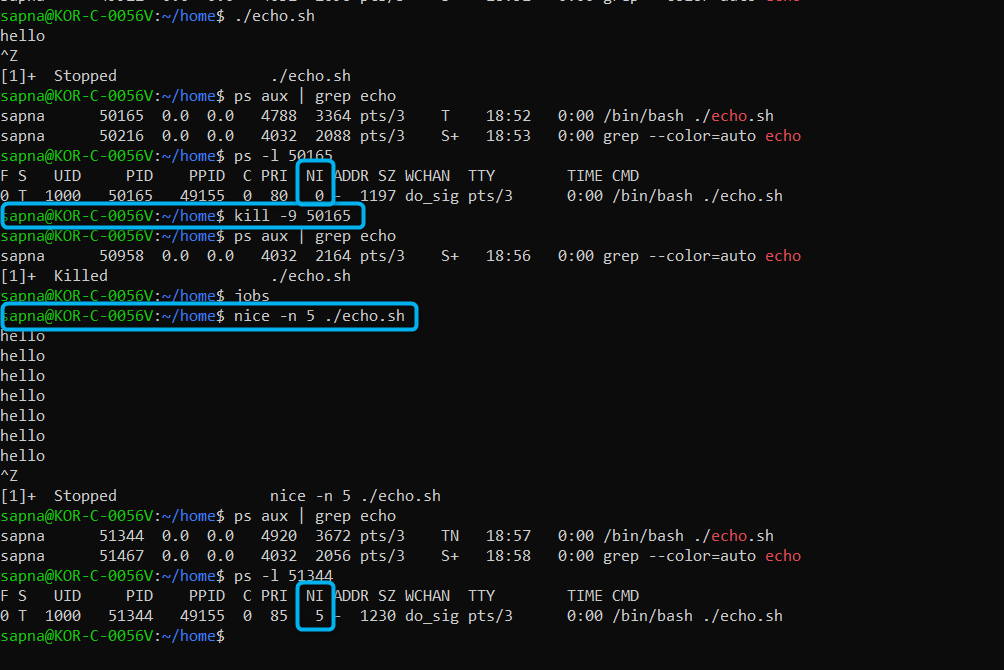
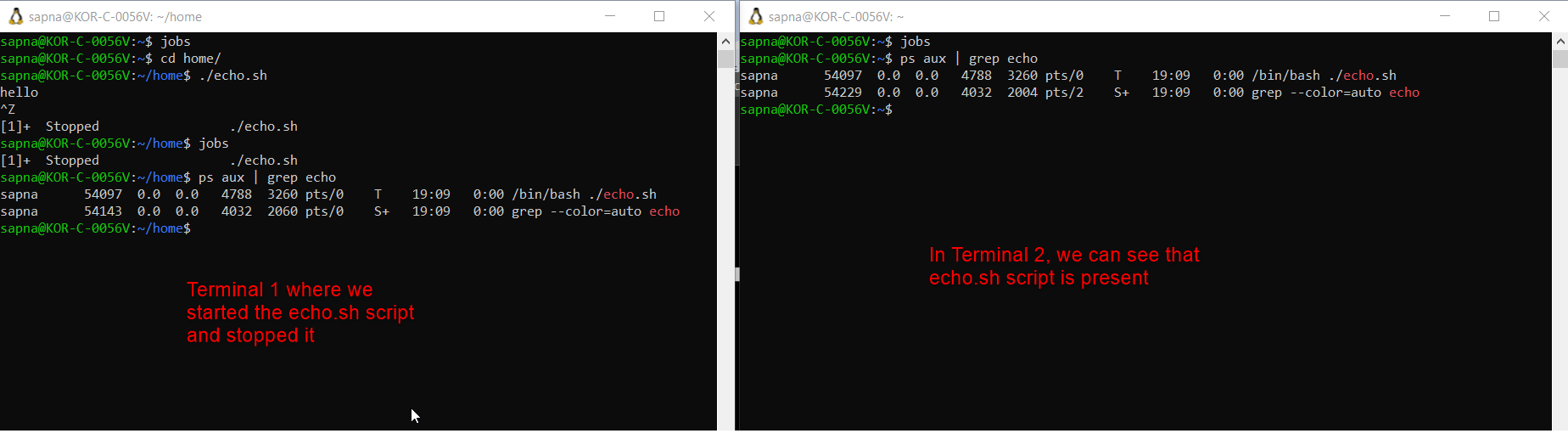
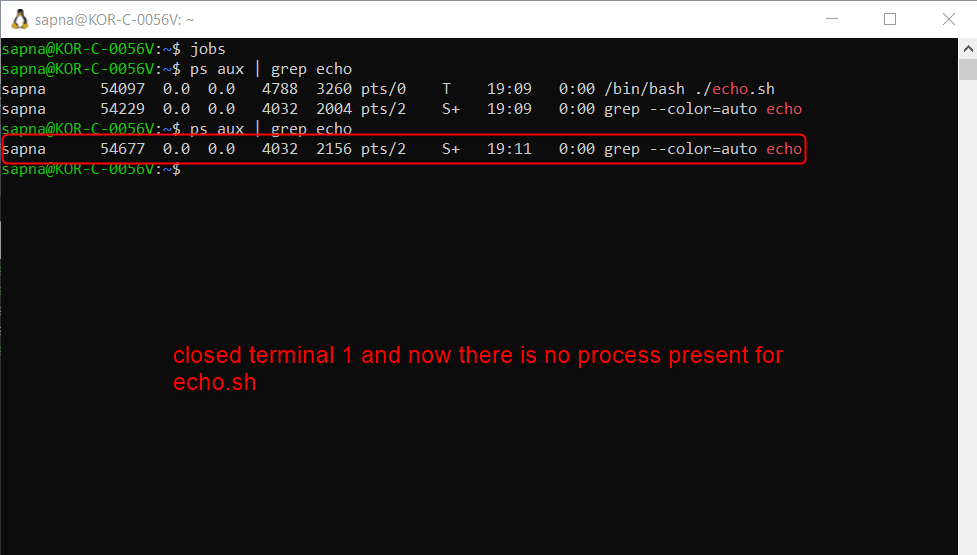
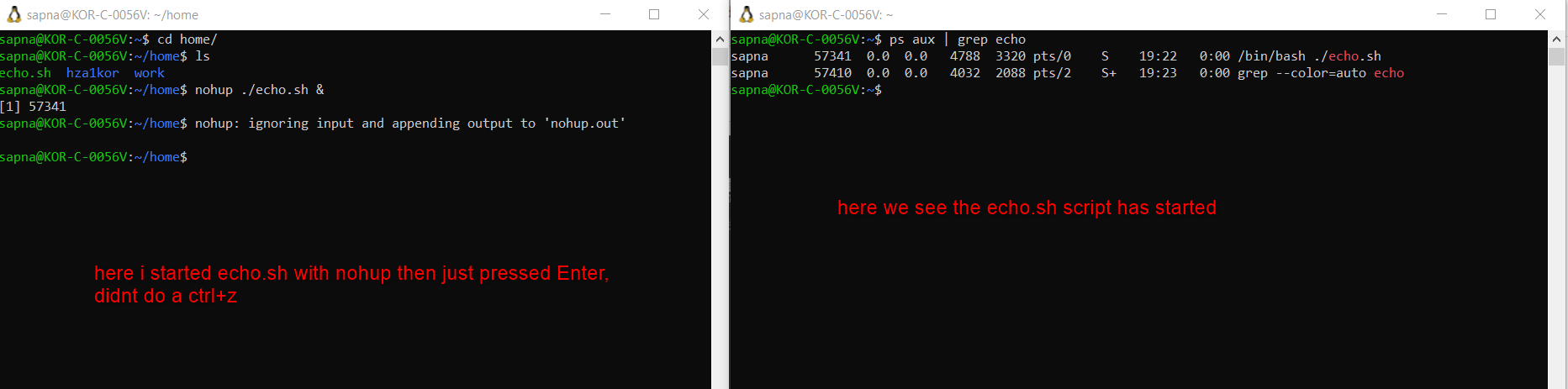
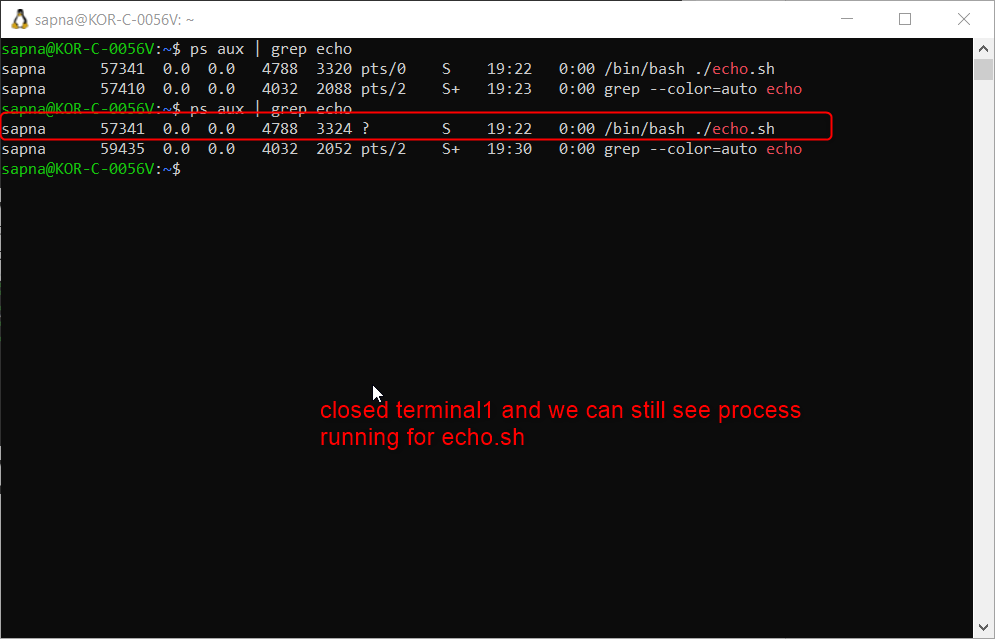
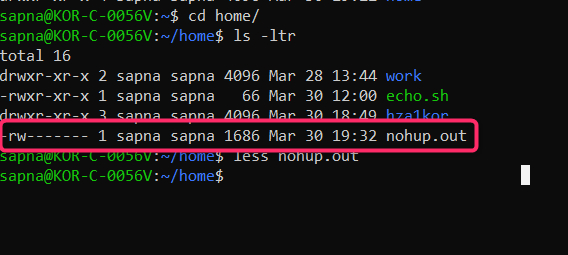
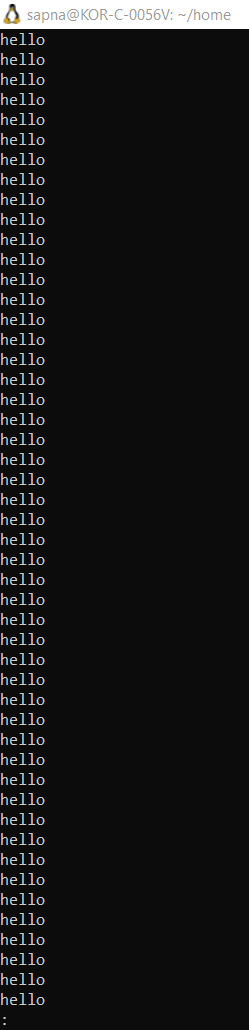
**#!/bin/bash**

**while true; do**

**echo "hello"**

**sleep 2**

**done**

* **Nice value –** niceness scale goes from -20 to 19. The lower the number the more priority the task gets
* **Process priority –** nice value
* To check the nice value of a process, run ps -l PID
* Note that to change the nice value you will have to kill the process, give the nice value in the beginning, then start the script and check for nice value of the PID, you cant directly change the nice value while the process is running, I guess you will have to use renice command for it
* Note that if you want to set the nice value to negative, you need to have root access
* 
* Nohup – if you want your process keep running even after closing your terminal, you can use nohup
* Say I start echo.sh, and then stop it, check the ps aux | grep echo, then open another terminal and run the same command we will see echo process, but when I close the terminal, and again run ps aux | grep echo, on another terminal, echo.sh process will not be present then
* 
* Now say I close the first terminal, then echo.sh will be terminated as well and wont be visible on the terminal
* 
* Now lets try using nohup, to start echo.sh script, so on terminal1, we start echo.sh with nohup, command will be **nohup ./echo.sh &**, then don’t do ctrl+z, just press Enter, then on terminal 2, we can see the process started for echo.sh
* 
* Now the idea is that if I close the first terminal, still I should be able to see that echo.sh process is running
* 
* Now the question that arrised is that if echo.sh is running, then where are all the hello o/p being saved, the answer is its saved in a file called nohup.out(this is at the location where you started the run from)
* 
* And this file will keep updating the o/p of the script echo.sh
* 
* Now say I am not really interested in having the o/p stored in nohup.out, I can directly run – nohup ./echo.sh > /dev/null 2>&1 &
* 