10.2.5 Process Management Facts

When a process is running in the foreground, no other commands can be run from that terminal until the foreground process terminates.
Therefore, it is important to know how to move processes to the background to allow addition commands to be run.

This lesson covers the following topics:

- Moving processes between the foreground and the background
- Adjusting process priorities

Moving Processes between the Foreground and the Background

The following table describes ways to move processes between the foreground and the background.

Command	Function	Examples
command **&**	Starts a process in the background, leaving the shell available for other commands. When running a process in the background, the shell displays the following information: - The job ID in brackets - The process ID (PID)	**gedit &** starts the gedit process in the background.
jobs	Views background jobs and their associated job ID numbers. The job ID number is specific to the terminal session. Each opened terminal has its own set of jobs and job ID numbers. Jobs from one terminal cannot be managed from a second terminal using job ID numbers. **jobs's** options function as follows: **-c** prints the command name for each process in jobs. **-g** only print the group id of each job. **-l** only the last job to be started is printed. **-p** print the process id for each process in all jobs.	**jobs -p %p** display the process ID or jobs for the job whose name begins with "p".
bg	Sends a job to the background.	**bg 3** sends the job with job ID 3 to the background.
fg	Brings a job to the foreground.	**fg 1** brings the job with job ID 1 to the foreground.
Ctrl+z	Pauses a running process and sends it to the background. A job ID number is assigned to the process.	

Adjusting Process Priorities

The following table describes ways to adjust process priorities.

Command	Function	Examples

nice	Starts a process and customize its priority on the system by manipulating the process's nice value. Be aware of the following:
	
	- Each process running on a Linux system has a priority (PR) and nice (NI) value associated with it.
	  - The PR value represents the process's kernel priority. The higher the PR value, the lower the priority of the process. The lower the PR value, the higher the process's priority.
	  - The NI value is factored into the kernel calculations that determine the PR value of a process. Changing the NI value associated with a process also changes its PR value.
	- NI values range from **19** (lowest priority), to **-20** (highest priority). The higher the NI value, the lower the process's priority.The lower the NI value, the higher the process's priority.
	- Use **-n** to specify the priority value. If no value is specified, the process starts with a nice value of 10 by default.
	- Zero (0) is the default nice value for processes not executed with the **nice** command.
		nice -n 7 gedit starts gedit with a nice value of 7. This runs the process with a lower priority than the default.
		
		nice -n -9 gedit starts gedit with a nice value of -9. This runs the process with a higher priority than the default.

	To keep users from abusing the nice command, standard users are not allowed to specify a nice value less than **0**. Only root can assign a nice value that is a negative number.	
renice	Assigns a new nice value to a running process using the PID of the process. The command can contain multiple PIDs separated by a space to give them all the same priority number. The nice values assigned are identical to those used with the **nice** command. You can use several options with **renice**: • **-n** specifies a priority. • **-u** specifies a user. • **-g** specifies a group.	**renice -5 3346** assigns a nice value of -5 to the process with a PID of 3346. This raises the priority of the process from the default. **renice 7 2266 3902** assigns a nice value of 7 to both processes. This lowers the priority of both processes from the default. **renice 5 -u userbob** sets the nice value of all processes owned by userbob to 5.