Linux Job and Process Control Cheat Sheet

Job and Process Control

When Linux runs a program or service, it runs it in a *process*. Sometimes a service spawns several processes. It's important for a Linux administrator to know how to view and control processes so they can manage what software is running on their server.

Listing Processes

The following commands are useful when you want to see what processes are running on a server.

Command	Description
`ps -ef`	Detailed listing of all running processes
`pgrep` *`pattern`*	Search the list of running processes
`jobs`	List jobs associated with the current shell

The `ps` command offers a lot of functionality, but the options you'll use most often are `-ef`, which tell `ps` to list all running processes with full details. The output is usually piped to `grep` or to a pager like `less` to make it easier to sift through.

The `pgrep` command is a specialized `grep` command that searches all running processes. It can limit the results to a specific user with the `-u` option, as in `pgrep -u username pattern`.

The `jobs` command lists the processes associated with the current shell that are running in the background.

Controlling Processes

If you want to control the active command in the current shell (for example, if you typed the command in yourself and are watching it run), you can cancel or suspend the process with control-C or control-Z, respectively.

Command	Description
Control-C	Cancel the active foreground process
Control-Z	Suspend the active foreground process

A suspended process can be designated as a job to run in the background with the `bg` command and can be resumed with the `fg` command.

Processes running independently of a shell or being run by other users can be halted with the `kill` command.

Command	Description
`kill` *`process-id`*	Tell a process to halt
`kill -9` *`process-id`*	Halt a process by force
`pkill` *`pattern`*	Search for a process then kill it

Running `kill` with no options will tell a process to end execution. That usually works.

Sometimes a regular `kill` doesn't work - if that happens, adding `-9` to the options tells `kill` to terminate the process with extreme prejudice. This should be used as a last resort, as it can prevent programs from running cleanup operations when they exit.

The `pkill` command searches the process list for a pattern then kills the processes it finds. Use it with caution - `pkill` can be handy when the program you need to end is running in multiple processes.

Background Processes

If a script or executable is taking longer than you'd like to complete, you can suspend the process with control-Z. You can make a suspended process run in the background with the `bg` command or bring it back to the fore with the `fg` command.

When you exit the active shell, suspended and backgrounded jobs will terminate.

Command	Description
`bg`	Run a suspended process in the background
`fg`	Have a suspended or backgrounded job run in the active shell

The `jobs` command will list suspended or backgrounded jobs. The list identifies each job with a number. You can use that number as an argument to tell `bg` and `fg` what job to act on (like `fg 2` to bring the second job to the foreground).

If you know that you want it to run in the background, put `&` after the command to run it as a background job for the current shell. If it needs to keep running after you log off, use the `nohup` command to tell the system to detach it from the active shell.

Command	Description
`command` `&`	Run a command in the background of the active shell
`nohup` *`command`*	Run a command and detach it from the active shell

Putting a `&` symbol after a command will have it start as a background job. For example, `ping localhost &` will start a series of pings running in the background. You can bring the job to the foreground with the `fg` command.

The `nohup` command tells the shell to run the command but detach its process from the active shell. That makes the process run independently - when you exit the active shell, the command will keep running as if it were a system service. Use `nohup` when you know you want a command to keep running after you log off.