

## 2.7.7 Command Substitution Facts

Command substitution is a feature of the bash shell that substitutes the output of one shell command as the arguments for another shell command.

This lesson covers the following topics:

- Implementing command substitution
- The xargs command

### Implementing Command Substitution

Command substitution is implemented using the **`$(<command>)`** operator. When the shell's command line interpreter encounters the **`$()`** operator:

1. The shell creates a child process that runs the command specified.
2. The stdout from this first command is redirected back to the shell.
3. The shell parses the output from the first command into words separated by white space.



If the command substitution operator **`$()`** appears withing double quotes (""), word parsing is not performed on the output.




4. The shell creates a new command by substituting the parsed output from the first command in place of the **`$(<command>)`** operator.
5. The shell creates another child process that runs the second command.



The backtick (**```**) operator also performs command substitution. Enclosing a command within backticks (**```**) will perform command substitution in the same way as the **`$()`** operator. On newer keyboards, the backtick (**```**) is on the same key as the tilde (**`~`**).

The following examples demonstrate these command substitution concepts.

Example	Explanation
<b><code>printf "The date and time is: \$(date)\n"</code></b>	<p>Command substitution occurs when the shell encounters <b><code>\$(date)</code></b>.</p> <ol style="list-style-type: none"><li>1. The shell creates a child process and runs the <b><code>\$(date)</code></b> command.</li><li>2. The output from the child process is redirected back to the shell, but is not parsed.</li><li>3. The <b><code>\$(date)</code></b> operator in the original command is replaced with the output from the child process.</li></ol>

11/10/22, 9:17 PM	TestOut LabSim
	<p>4. Another new process is created that runs the <b>printf</b> command with the replaced text.</p> <div> The <b>printf</b> command replaces <b>\n</b> within the double quotes with a newline character.</div>
<pre>echo -e "List of logged on users and what they are doing:\n \$(w)"</pre>	<p>Command substitution occurs when the shell encounters <b>\$(w)</b>.</p> <p>1. The shell creates a child process and runs the <b>w</b> command.</p> <div> The <b>w</b> command is short for <i>who</i> and returns a summary of logged on users.</div> <p>2. The output from the child process is redirected back to the shell, but is not parsed.</p> <p>3. The <b>\$(w)</b> operator in the original command is replaced with the output from the child process.</p> <p>4. Another new process is created that runs the <b>echo</b> command with the replaced text.</p> <div> The <b>-e</b> option on the <b>echo</b> command causes the <b>\n</b> within the double quotes to be replaced with a newline character.</div>

## The xargs Command

Linux commands have a maximum length of 128 KB. Command substitution will give an error if the command that is formed after the substitution is over 128 KB. One solution is to pipe the first command to the **xargs** command. The **xargs** command will break down the output that is streamed from the first command into 128 KB chunks. Each chunk is passed one at a time as an argument to the **xargs** command.

The following examples demonstrate the use of the **xargs** command.

Example	Explanation
<pre>find /home - name *~   xargs rm</pre>	<p>This command will delete all the files in all the subdirectories of the /home directory that end with the tilde (~) character.</p> <p>1. The <b>find /home -name *~</b> command returns a list of all the files in all the subdirectories under the /home directory that end with the tilde (~) character.</p>

	<ol style="list-style-type: none"><li>2. The file list is piped as a stream to the <b>xargs</b> command.</li><li>3. The <b>xargs</b> command collects the first 128 KB chunk from the stream.</li><li>4. The <b>rm</b> command is run using the 128 KB chunk as an argument.</li><li>5. The <b>xargs</b> command continues to collect 128 KB chunks from the stream and continues to run the <b>rm</b> command using the chunks until the end of the stream is encountered.</li></ol>
<b>ls -S *.txt   xargs wc</b>	<p>When there are large number of *.txt files in a directory, this command will display a list of all the files along with the number of lines/words/characters in each file. The list will be sorted by size with the largest files shown first.</p> <ol style="list-style-type: none"><li>1. The <b>ls -S *.txt</b> command returns the list of files sorted from largest to smallest.</li><li>2. The file list is piped as a stream to the <b>xargs</b> command.</li><li>3. The <b>xargs</b> command collects the first 128 KB chunk from the stream.</li><li>4. The <b>wc</b> command is run using the 128 KB chunk as an argument.</li><li>5. The <b>xargs</b> command continues to collect 128 KB chunks from the stream and continues to run the <b>wc</b> command using the chunks until the end of the stream is encountered.</li></ol>

---

Copyright © 2022 TestOut Corporation All rights reserved.