**Variable Scope**

A confusing property of environment variables is that setting them in one shell applies only to that shell and not parents, children, or other shells. For example, if you had a program that looked at an environment variable called WORK\_FACTOR to see how much work it could do, this wouldn’t work:

$ WORK\_FACTOR=3  
$ ./my\_command

This is because running **./my\_command** starts a child process with a new environment. Variables you set aren’t passed to children. You need to export that variable:

$ WORK\_FACTOR=3  
$ export WORK\_FACTOR  
$ ./my\_command

The **export** command marks the variable as something that gets passed to children. There are less verbose ways of accomplishing this: **export WORK\_FACTOR=3**.

The **export** keyword can be used on the same line as the assignment. Alternatively, if you just want to pass the variable to the child and not set it in the current environment, you can set it on the same line as the command:

[**Click here to view code image**](ch11_images.html#p316pro01a)

$ WORK\_FACTOR=3 ./my\_command  
$ echo $WORK\_FACTOR  
  
$

The **echo** shows that even though you set WORK\_FACTOR to 3 for the child, the current process didn’t keep it around.

By itself, **export** prints a list of variables currently exported to child environments.

###### Setting Variables from a Child

A common pattern is to put the assignments in a separate configuration file and use that file in a script or the command line. This adds consistency and makes commands easier to type. However the variables would be set in the child environment, which is thrown away when the process exits.

Given a configuration file called **config** containing an assignment; **WORK\_FACTOR=3**.

If you were to run that, your current shell wouldn’t know about the variable:

$ ./config  
$ echo $WORK\_FACTOR  
  
$

What you need is to make the assignments in the current environment. This is the job of **source**:

$ source ./config

$ echo $WORK\_FACTOR  
3

[*Sourcing*](gloss01.html#gloss_393) a file executes it in the current shell environment instead of creating a separate child environment. Thus variables set in the script are available in the current environment. If the script didn’t **export** the variable, it will not be available to child processes later on.

There is an alternate way to source a file, which is to replace the word **source** with a period:

$ . ./config