9.1. The for loop

9.1.1. How does it work?

The **for** loop is the first of the three shell looping constructs. This loop allows for specification of a list of values. A list of commands is executed for each value in the list.

The syntax for this loop is:

for `NAME` **[in LIST]; do COMMANDS; done**

If **[in LIST]** is not present, it is replaced with **in** `$@` and **for** executes the **COMMANDS** once for each positional parameter that is set (see Section 3.2.5 and Section 7.2.1.2).

The return status is the exit status of the last command that executes. If no commands are executed because `LIST` does not expand to any items, the return status is zero.

`NAME` can be any variable name, although `i` is used very often. `LIST` can be any list of words, strings or numbers, which can be literal or generated by any command. The **COMMANDS** to execute can also be any operating system commands, script, program or shell statement. The first time through the loop, `NAME` is set to the first item in `LIST`. The second time, its value is set to the second item in the list, and so on. The loop terminates when `NAME` has taken on each of the values from `LIST` and no items are left in `LIST`.

9.1.2. Examples

9.1.2.1. Using command substitution for specifying LIST items

The first is a command line example, demonstrating the use of a **for** loop that makes a backup copy of each `.xml` file. After issuing the command, it is safe to start working on your sources:

```
[carol@octarine ~/articles] ls *.xml
file1.xml  file2.xml  file3.xml

[carol@octarine ~/articles] ls *.xml > list

[carol@octarine ~/articles] for i in `cat list`; do cp "$i" "$i".bak ; done

[carol@octarine ~/articles] ls *.xml*
file1.xml  file1.xml.bak  file2.xml  file2.xml.bak  file3.xml  file3.xml.bak
```

This one lists the files in `/sbin` that are just plain text files, and possibly scripts:

```
for i in `ls /sbin`; do file /sbin/$i | grep ASCII; done
```

9.1.2.2. Using the content of a variable to specify LIST items

The following is a specific application script for converting HTML files, compliant with a certain scheme, to PHP files. The conversion is done by taking out the first 25 and the last 21 lines, replacing these with two PHP tags that provide header and footer lines:

```
[carol@octarine ~/html] cat html2php.sh
#!/bin/bash
# specific conversion script for my html files to php
```

```
LIST="$(ls *.html)"
for i in "$LIST"; do
     NEWNAME=$(ls "$i" | sed -e 's/html/php/')
     cat beginfile > "$NEWNAME"
     cat "$i" | sed -e '1,25d' | tac | sed -e '1,21d'| tac >> "$NEWNAME"
     cat endfile >> "$NEWNAME"
done
```

Since we don't do a line count here, there is no way of knowing the line number from which to start deleting lines until reaching the end. The problem is solved using **tac**, which reverses the lines in a file.

The basename command

Instead of using **sed** to replace the `html` suffix with `php`, it would be cleaner to use the **basename** command. Read the man page for more info.

Odd characters

You will run into problems if the list expands to file names containing spaces and other irregular characters. A more ideal construct to obtain the list would be to use the shell's globbing feature, like this:

```
for i in $PATHNAME/*; do
        commands
done
```

Prev Home Next
Repetitive tasks Up The while loop