

Lab: Intro to variables and parameters

1. Every script should have a usage statement. Let's get that into your template. Modify your template program. Inside there, assign a variable called USAGE. Set it equal to a statement of how the program should be run. Hint: most usage statements are of this form:

Usage: <Name of script> -<list of options> <list of parameters>
Right here include a one or two-line overview of what this program should do and how to use it.

Another hint: Don't forget about \$0.

2. Now change your template to say that if the number of command-line parameters is not exactly equal to 0, then we should exit with an error code of 1 and print an error message.

You can now use your template for all future programs and test to see if they have the right number of parameters.

Creating a web page with a table

3. Your instructor will give you a file with a list of customers. Our customers, in fact. Copy that into your working directory. Set its permissions to 666 so that anyone can read and write it.
4. Write a script called Customers.ksh that creates a web page. Run and test it in a browser.
5. In the main part of the web page, put this code:

```
print "<table><tbody>"
print "<tr><th>Name</th><th>Address</th><th>City</th><th>Region</th>"
print "<th>Postal Code</th><th>Phone</th><th>Email</th><th></th></tr>"
for LINE in PUT_PEOPLE_HERE
do
    # DO THINGS HERE
done
print "</tbody></table>"
```

6. Run and test. You should see a table header on your page.

Reading each line from your database file

7. Alright, now the fun begins. See where it says "PUT_PEOPLE_HERE"? Replace that with the contents of the people.txt file. (Hint: You'll need to use command substitution and the *cat* command.)
8. Now put the contents of the \$LINE variable between "<td>" and "</td>" tags. if done right, you should see the contents of the the people file, but it won't be formatted well yet. The problem is that the shell is breaking up its for-loop list by a variable called IFS which is set to spaces, tabs and newlines. We need to set it to newlines only. Here's how ...
9. Before the for loop, save the value of IFS in a variable called OLDIFS.
10. Then change IFS like this:

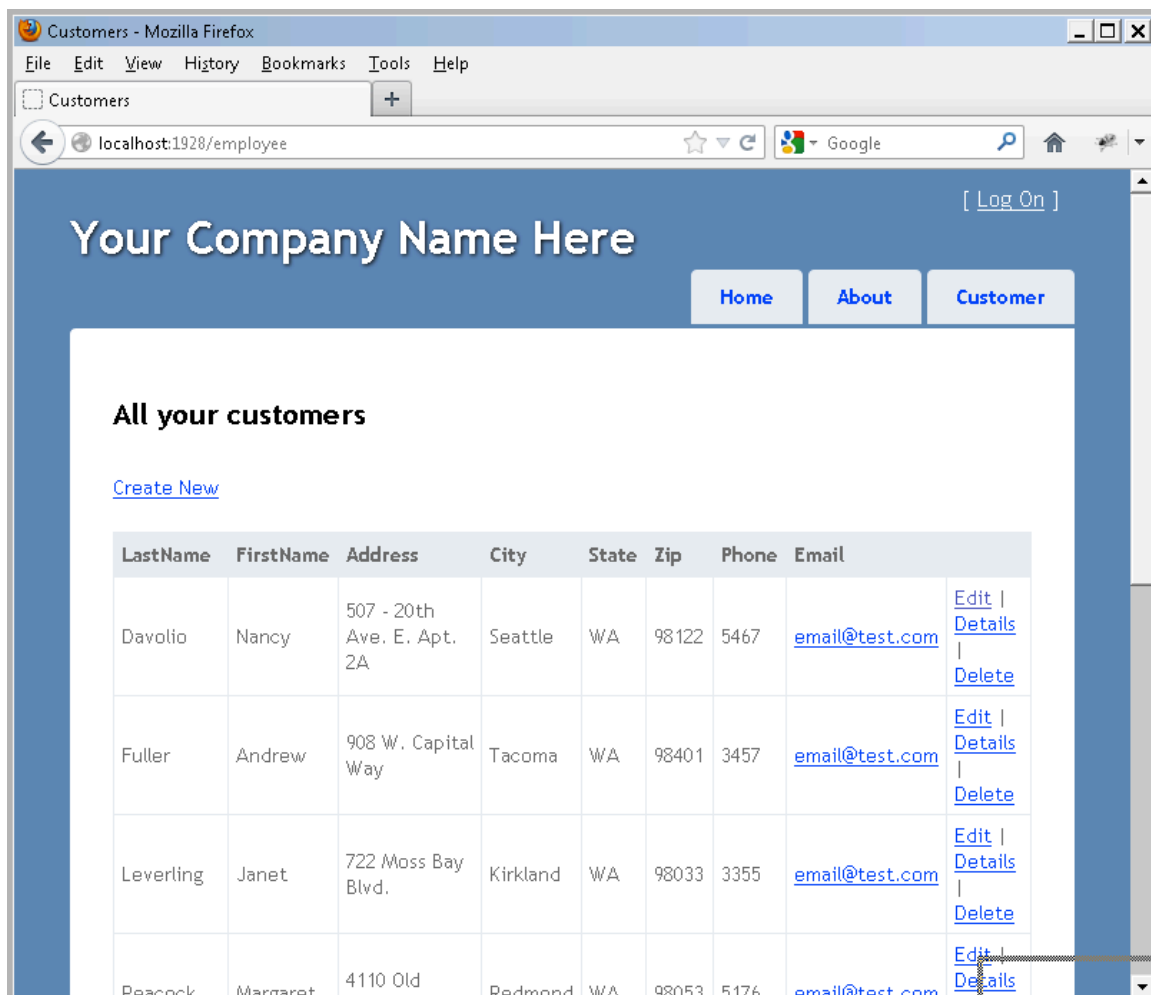
```
IFS='
'
```

11. What you just did there was store only a newline character in IFS.

12. After the for loop, restore the value of IFS from that OLDIFS variable you saved a few steps before.
13. Run and test. Now you should see the entire line in the table. Once you do, you can move on.

Parsing the fields into variables

14. Inside the loop, put each field from the line in a separate variable. Call the variables \$ID, \$LAST, \$FIRST, \$PHONE, and so on. (Hint: Again, you'll use command substitution. You'll also want to use redirection and the cut command).
15. Put each variable between "<td>" and "</td>" tags.
16. Run and test. When it looks something like this you can be finished:



The screenshot shows a web browser window titled "Customers - Mozilla Firefox". The address bar displays "localhost:1928/employee". The page has a blue header with the text "Your Company Name Here" and a "[Log On]" link. Below the header are three buttons: "Home", "About", and "Customer". The main content area is titled "All your customers" and includes a "Create New" link. A table lists customer information with columns: LastName, FirstName, Address, City, State, Zip, Phone, Email, and a column for actions (Edit, Details, Delete). The table contains four rows of customer data.

LastName	FirstName	Address	City	State	Zip	Phone	Email	
Davolio	Nancy	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98122	5467	email@test.com	Edit Details Delete
Fuller	Andrew	908 W. Capital Way	Tacoma	WA	98401	3457	email@test.com	Edit Details Delete
Leverling	Janet	722 Moss Bay Blvd.	Kirkland	WA	98033	3355	email@test.com	Edit Details Delete
Peacock	Margaret	4110 Old	Redmond	WA	98053	5176	email@test.com	Edit Details Delete