

```
#!/bin/bash

function argv {
    for a in ${BASH_ARGV[*]} ; do
        echo -n "$a "
    done
    echo
}

function f {
    echo f $1 $2 $3
    echo -n f ; argv
}

function g {
    echo g $1 $2 $3
    echo -n g; argv
    f
}

f boo bar baz
g goo gar gaz
Save in f.sh
```

```
$ ./f.sh arg0 arg1 arg2
f boo bar baz
farg2 arg1 arg0
g goo gar gaz
garg2 arg1 arg0
f
farg2 arg1 arg0
```

If you want to have your arguments C style (array of arguments + number of arguments) you can use `$@` and `$#`.

`$#` gives you the number of arguments. `$@` gives you all arguments. you can turn this into an array by `args=("$@")`
So for example:

```
args=("$@")
echo $# arguments passed
echo ${args[0]} ${args[1]} ${args[2]}
```

Note that here `${args[0]}` actually is the 1st argument and not the name of your script.

Ravi's comment is essentially the answer. Functions take their own arguments. If you want them to be the same as the command-line arguments, you must pass them in. Otherwise, you're clearly calling a function without arguments.

That said, you could if you like store the command-line arguments in a global array to use within other functions:

```
my_function() {
    echo "stored arguments:"
    for arg in "${commandline_args[@]}"; do
        echo "    $arg"
    done
}
```

```
commandline_args=("$@")
```

```
my_function
```

You have to access the command-line arguments through the `commandline_args` variable, not `$@`, `$1`, `$2`, etc., but they're available. I'm unaware of any way to assign directly to the argument array, but if someone knows one, please enlighten me!

Also, note the way I've used and quoted `$@` - this is how you ensure special characters (whitespace) don't get mucked up.

```
#!/usr/bin/env bash
```

```
echo name of script is $0
echo first argument is $1
echo second argument is $2
echo seventeenth argument is $17
echo number of arguments is $#
```

Edit: please see my comment on question

```
# Save the script arguments
```

```
SCRIPT_NAME=$0
```

```
ARG_1=$1
```

```
ARGS_ALL=$*
```

```
function stuff {
    # use script args via the variables you saved
    # or the function args via $
    echo $0 $*
}
```

```
# Call the function with arguments
```

```
stuff 1 2 3 4
```

ou can use the `shift` keyword (operator?) to iterate through them. Example:

```
#!/bin/bash
```

```
function print()
```

```
{
```

```
    while [ $# -gt 0 ]
```

```
    do
```

```
        echo $1;
```

```
        shift 1;
```

```
    done
```

```
}  
print $*;
```

One can do it like this as well

```
#!/bin/bash  
# script_name function_test.sh  
function argument(){  
  for i in $@;do  
    echo $i  
  done;  
}  
argument $@
```

Now call your script like

```
./function_test.sh argument1 argument2
```

\$1 (or \$2,\$3 ...) is supposed to be the arguments given to some script.

Here's an example script:

```
#!/bin/bash  
  
echo "\$1 is now $1"  
echo "\$2 is now $2"  
echo "\$3 is now $3"
```

And the example output

```
jaba@lappy:/tmp$ ./example.sh  
$1 is now  
$2 is now  
$3 is now  
jaba@lappy:/tmp$ ./example.sh 1 2 3  
$1 is now 1  
$2 is now 2  
$3 is now 3
```

```
GNU nano 2.2.6      File: bin/datecp

#!/bin/bash

# This will copy a file, appending the date and time
# to the end of the file name.

date_formatted=$(date +%m_%d_%y-%H.%M.%S)

cp -iv $1 $2.$date_formatted

yatri@svarga:~$ datecp Trogdor2.mp3 backups/Trogdor_music.mp3
'Trogdor2.mp3' -> 'backups/Trogdor_music.mp3.07_05_11-04.25.45'
yatri@svarga:~$
```

The term “shell scripting” gets mentioned often in Linux forums, but many users aren’t familiar with it. Learning this easy and powerful programming method can help you save time, learn the command-line better, and banish tedious file management tasks.

What Is Shell Scripting?

Being a Linux user means you play around with the command-line. Like it or not, there are just some things that are done much more easily via this interface than by pointing and clicking. The more you use and learn the command-line, the more you see its potential. Well, the command-line itself is a program: the shell. Most Linux distros today use Bash, and this is what you’re really entering commands into.

Now, some of you who used Windows before using Linux may remember batch files. These were little text files that you could fill with commands to execute and Windows would run them in turn. It was a clever and neat way to get some things done, like run games in your high school computer lab when you couldn’t open system folders or create shortcuts. Batch files in Windows, while useful, are a cheap imitation of shell scripts.

```
GNU nano 2.2.6      File: cbr      Modified

RAR_PATH=/usr/bin #Path to rar binary

cd $NAUTILUS_SCRIPT_CURRENT_URI
LABEL=$(gdialog --title "RAR Output Name?" --inputbox "Type output
name here.. defaulted to default so we dont get any errors,
you can change this name to anything you like
BUT DONT HAVE SPACES AND DONT ADD THE .rar
You Just need the name you want " 200 450 default 2>&1)

LABEL1=$(gdialog --title "RAR File Size?" --inputbox "Type size here.
For auto sizing make it blank. LEAVE BLANK IF YOU WANT ALL FILES IN 1
RAR FILE,
1mb files type, 1000000kb
3mb files type, 3000000kb
5mb files type, 5000000kb

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit     ^J Justify  ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell
```

Shell scripts allow us to program commands in chains and have the system execute them as a scripted event, just like batch files. They also allow for far more useful functions, such as command substitution. You can invoke a command, like `date`, and use its output as part of a file-naming scheme. You can automate backups and each copied file can have the current date appended to the end of its name. Scripts aren't just invocations of commands, either. They're programs in their own right. Scripting allows you to use programming functions – such as 'for' loops, if/then/else statements, and so forth – directly within your operating system's interface. And, you don't have to learn another language because you're using what you already know: the command-line.

That's really the power of scripting, I think. You get to program with commands you already know, while learning staples of most major programming languages. Need to do something repetitive and tedious? Script it! Need a shortcut for a really convoluted command? Script it! Want to build a really easy to use command-line interface for something? Script it!

Before You Begin

Before we begin our scripting series, let's cover some basic information. We'll be using the bash shell, which most Linux distributions use natively. Bash is available for Mac OS users and Cygwin on Windows, too. Since it's so universal, you should be able to script regardless of your platform. In addition, so long as all of the commands that are referenced exist, scripts can work on multiple platforms with little to no tweaking required.

Scripting can easily make use of “administrator” or “superuser” privileges, so it’s best to test out scripts before you put them to work. Also use common sense, like making sure you have backups of the files you’re about to run a script on. It’s also really important to use the right options, like `-i` for the `rm` command, so that your interaction is required. This can prevent some nasty mistakes. As such, read through scripts you download and be careful with data you have, just in case things go wrong.

At their core, scripts are just plain text files. You can use any text editor to write them: `gedit`, `emacs`, `vim`, `nano`... This list goes on. Just be sure to save it as plain text, not as rich text, or a Word document. Since I love the [ease of use that nano provides](#), I’ll be using that.

Script Permissions and Names

Scripts are executed like programs, and in order for this to happen they need to have the proper permissions. You can make scripts executable by running the following command on it:

```
chmod +x ~/somecrazyfolder/script1
```

This will allow anyone to run that particular script. If you want to restrict its use to just your user, you can use this instead:

```
chmod u+x ~/somecrazyfolder/script1
```

In order to run this script, you would have to `cd` into the proper directory and then run the script like this:

```
cd ~/somecrazyfolder
./script1
```

To make things more convenient, you can place scripts in a “bin” folder in your home directory:

```
~/bin
```

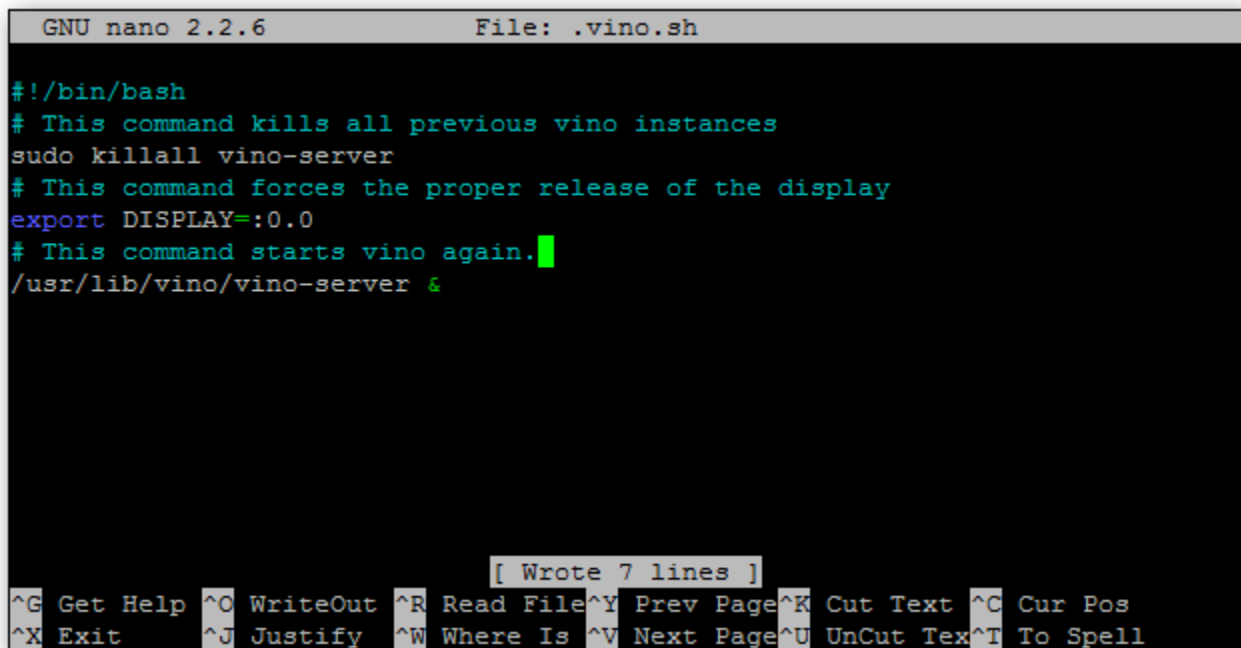
In many modern distros, this folder no longer is created by default, but you can create it. This is usually where executable files are stored that belong to your user and not to other users. By placing scripts here, you can just run them by typing their name, just like other commands, instead of having to `cd` around and use the `./` prefix.

Before you name a script, though, you should the following command to check if you have a program installed that uses that name:

```
which [command]
```

A lot of people name their early scripts “test,” and when they try to run it in the command-line, nothing happens. This is because it conflicts with the test command, which does nothing without arguments. Always be sure your script names don’t conflict with commands, otherwise you may find yourself doing things you don’t intend to do!

Scripting Guidelines



```
GNU nano 2.2.6 File: .vino.sh

#!/bin/bash
# This command kills all previous vino instances
sudo killall vino-server
# This command forces the proper release of the display
export DISPLAY=:0.0
# This command starts vino again.
/usr/lib/vino/vino-server &
```

[Wrote 7 lines]

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell

As I mentioned before, every script file is essentially plain text. That doesn’t mean you can write what you want all willy-nilly, though. When a text file is attempted to be executed, shells will parse through them for clues as to whether they’re scripts or not, and how to handle everything properly. Because of this, there are a few guidelines you need to know.

1. Every script should be with “#!/bin/bash”
2. Every new line is a new command
3. Comment lines start with a #
4. Commands are surrounded by ()

The Hash-Bang Hack

When a shell parses through a text file, the most direct way to identify the file as a script is by making your first line:

```
#!/bin/bash
```

If you use another shell, substitute its path here. Comment lines start with hashes (#), but adding the bang (!) and the shell path after it is a sort of hack that will bypass this comment rule and will force the script to execute with the shell that this line points to.

New Line = New Command

Every new line should be considered a new command, or a component of a larger system. If/then/else statements, for example, will take over multiple lines, but each component of that system is in a new line. Don't let a command bleed over into the next line, as this can truncate the previous command and give you an error on the next line. If your text editor is doing that, you should turn off text-wrapping to be on the safe side. You can turn off text wrapping in nano by hitting ALT+L.

Comment Often with #s

If you start a line with a #, the line is ignored. This turns it into a comment line, where you can remind yourself of what the output of the previous command was, or what the next command will do. Again, turn off text wrapping, or break your comment into multiple lines that all begin with a hash. Using lots of comments is a good practice to keep, as it lets you and other people tweak your scripts more easily. The only exception is the aforementioned Hash-Bang hack, so don't follow #s with !s. ;-)

Commands Are Surrounded By Parentheses

In older days, command substitutions were done with single tick marks (` , shares the ~ key). We're not going to be touching on this yet, but as most people go off and explore after learning the basics, it's probably a good idea to mention that you should use parentheses instead. This is mainly because when you nest – put commands inside other commands – parentheses work better.

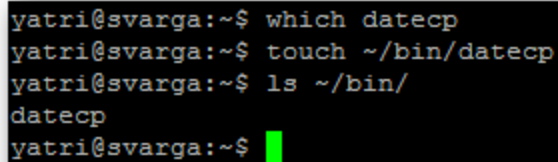
Your First Script

Let's start with a simple script that allows you to copy files and append dates to the end of the filename. Let's call it "datecp". First, let's check to see if that name conflicts with something:

You can see that there's no output of the which command, so we're all set to use this name.

Let's create a blank file in the ~/bin folder:

```
touch ~/bin/datecp
```

A terminal window with a black background and green text. The user 'yatri' is at host 'svarga' in the home directory '~'. They run 'which datecp' and get no output. Then they run 'touch ~/bin/datecp'. Finally, they run 'ls ~/bin/' and see 'datecp' listed. The prompt is currently at 'yatri@svarga:~\$' with a green cursor.

```
yatri@svarga:~$ which datecp
yatri@svarga:~$ touch ~/bin/datecp
yatri@svarga:~$ ls ~/bin/
datecp
yatri@svarga:~$
```

And, let's change the permission now, before we forget:

Let's start building our script then. Open up that file in your text editor of choice. Like I said, I like the simplicity of nano.

```
nano ~/bin/datecp
```

And, let's go ahead and put in the prerequisite first line, and a comment about what this script does.

```
GNU nano 2.2.6      File: bin/datecp      Modified
#!/bin/bash

# This will copy a file, appending the date and time
# to the end of the file name.

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Tex ^T To Spell
```

Next, let's declare a variable. If you've ever taken algebra, you probably know what a that is. A variable allows us to store information and do things with it. Variables can "expand" when referenced elsewhere. That is, instead of displaying their name, they will display their stored contents. You can later tell that same variable to store different information, and any instruction that occurs after that will use the new information. It's a really fancy placeholder.

What will we put in our variable? Well, let's store the date and time! To do this, we'll call upon the date command.

Take a look at the screenshot below for how to build the output of the date command:

```
yatri@svarga:~$ date +%D
07/05/11
yatri@svarga:~$ date +%m
07
yatri@svarga:~$ date +%m_%d_%Y
07_05_2011
yatri@svarga:~$ date +%m_%d_%y
07_05_11
yatri@svarga:~$ date +%m_%d_%y-%l.%M.%S
07_05_11- 3.56.13
yatri@svarga:~$ date +%m_%d_%y-%H.%M.%S
07_05_11-04.00.15
yatri@svarga:~$ █
```

You can see that by adding different variables that start with %, you can change the output of the command to what you want. For more information, you can look at the manual page for the date command.

Let's use that last iteration of the date command, "date +%m_%d_%y-%H.%M.%S", and use that in our script.

```
GNU nano 2.2.6 File: bin/datecp

#!/bin/bash

# This will copy a file, appending the date and time
# to the end of the file name.

date +%m_%d_%y-%H.%M.%S

[ Wrote 6 lines ]

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

If we were to save this script right now, we could run it and it would give us the output of the date command like we'd expect:

```
yatri@svarga:~$ datecp
07_05_11-04.06.31
yatri@svarga:~$
```

But, let's do something different. Let's give a variable name, like `date_formatted` to this command. The proper syntax for this is as follows:

```
variable=$(command -options arguments)
```

And for us, we'd build it like this:

```
date_formatted=$(date +%m_%d_%y-%H.%M.%S)
```

```
GNU nano 2.2.6      File: bin/datecp      Modified

#!/bin/bash

# This will copy a file, appending the date and time
# to the end of the file name.

date_formatted=$(date +%m_%d_%y-%H.%M.%S)
█

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Tex ^T To Spell
```

This is what we call command substitution. We're essentially telling bash that whenever the variable "date_formatted" shows up, to run the command inside the parentheses. Then, whatever output the commands gives should be displayed instead of the name of the variable, "date_formatted".

Here's an example script and its output:

```
GNU nano 2.2.6      File: bin/datecp

#!/bin/bash

# This will copy a file, appending the date and time
# to the end of the file name.

date_formatted=$(date +%m_%d_%y-%H.%M.%S)

echo "This is the Date and Time: " $date_formatted

[ Read 8 lines ]

^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page^U UnCut Tex^T To Spell
```

Note that there are two spaces in the output. The space within the quotes of the echo command and the space in front of the variable are both displayed. Don't use spaces if you don't want them to show up. Also note that without this added "echo" line, the script would give absolutely no output.

Let's get back to our script. Let's next add in the copying part of the command.

```
cp -iv $1 $2.$date_formatted
```

```
GNU nano 2.2.6      File: bin/datecp      Modified

#!/bin/bash

# This will copy a file, appending the date and time
# to the end of the file name.

date_formatted=$(date +%m_%d_%y-%H.%M.%S)

cp -iv $1 $2.$date_formatted
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

This will invoke the copy command, with the `-i` and `-v` options. The former will ask you for verification before overwriting a file, and the latter will display what is being down on the command-line.

Next, you can see I've added the "`$1`" option. When scripting, a dollar sign (\$) followed by a number will denote that numbered argument of the script when it was invoked. For example, in the following command:

```
cp -iv Trogdor2.mp3 ringtone.mp3
```

The first argument is "Trogdor2.mp3" and the second argument is "ringtone.mp3".

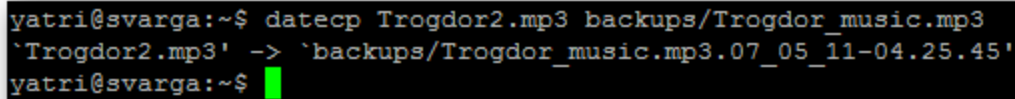
Looking back at our script, we can see that we're referencing two arguments:

```
cp -iv $1 $2.$date_formatted
```

This means that when we run the script, we'll need to provide two arguments for the script to run correctly. The first argument, `$1`, is the file that will be copied, and is substituted as the "`cp -iv`" command's first argument.

The second argument, \$2, will act as the output file for the same command. But, you can also see that it's different. We've added a period and we've referenced the "date_formatted" variable from above. Curious as to what this does?

Here's what happens when the script is run:

A terminal window with a black background and white text. The prompt is 'yatri@svarga:~\$'. The command entered is 'datecp Trogdor2.mp3 backups/Trogdor_music.mp3'. The output is '`Trogdor2.mp3' -> `backups/Trogdor_music.mp3.07_05_11-04.25.45''. The prompt is then 'yatri@svarga:~\$' followed by a green cursor.

```
yatri@svarga:~$ datecp Trogdor2.mp3 backups/Trogdor_music.mp3
`Trogdor2.mp3' -> `backups/Trogdor_music.mp3.07_05_11-04.25.45'
yatri@svarga:~$
```

You can see that the output file is listed as whatever I entered for \$2, followed by a period, then the output of the date command! Makes sense, right?

Now when I run the datecp command, it will run this script and allow me to copy any file to a new location, and automatically add the date and time to end of the filename. Useful for archiving stuff!

Shell scripting is at the heart of making your OS work for you. You don't have to learn a new programming language to make it happen, either. Try scripting with some basic commands at home and start thinking of what you can use this for.

for loop syntax

Numeric ranges for syntax is as follows:

```
for VARIABLE in 1 2 3 4 5 .. N
do
    command1
    command2
    commandN
done
OR
```

```
for VARIABLE in file1 file2 file3
do
    command1 on $VARIABLE
    command2
    commandN
done
OR
```

```
for OUTPUT in $(Linux-Or-Unix-Command-Here)
do
    command1 on $OUTPUT
    command2 on $OUTPUT
    commandN
done
```

Examples

UNIX

This type of for loop is characterized by counting. The range is specified by a beginning (#1) and ending number (#5). The for loop executes a sequence of commands for each member in a list of items. A representative example in BASH is as follows to display welcome message 5 times with for loop:

```
#!/bin/bash
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done
```

Sometimes you may need to set a step value (allowing one to count by two's or to count backwards for instance). Latest **bash version 3.0+** has inbuilt support for setting up ranges:

```
#!/bin/bash
for i in {1..5}
do
    echo "Welcome $i times"
done
```

Bash v4.0+ has inbuilt support for setting up a step value using **{START..END..INCREMENT}** syntax:

```
#!/bin/bash
echo "Bash version ${BASH_VERSION}..."
for i in {0..10..2}
do
    echo "Welcome $i times"
done
```

Sample outputs:

```
Bash version 4.0.33(0)-release...
```

```
Welcome 0 times
```

```
Welcome 2 times
```

```
Welcome 4 times
```

```
Welcome 6 times
```

```
Welcome 8 times
```

```
Welcome 10 times
```

The seq command (outdated)



WARNING! The seq command print a sequence of numbers and it is here due to historical reasons. The following examples is only recommend for older bash version. All users (bash v3.x+) are recommended to use the above syntax.

The [seq command](#) can be used as follows. A representative example in seq is as follows:

```
#!/bin/bash
for i in $(seq 1 2 20)
do
    echo "Welcome $i times"
done
```

There is no good reason to use an external command such as `seq` to count and increment numbers in the `for` loop, hence it is recommended that you avoid using `seq`. The builtin command are fast.

Three-expression `bash` for loops syntax

This type of `for` loop share a common heritage with the C programming language. It is characterized by a three-parameter loop control expression; consisting of an initializer (EXP1), a loop-test or condition (EXP2), and a counting expression (EXP3).

```
for (( EXP1; EXP2; EXP3 ))
do
    command1
    command2
    command3
done
```

A representative three-expression example in `bash` as follows:

```
#!/bin/bash
for (( c=1; c<=5; c++ ))
do
    echo "Welcome $c times"
done
```

Sample output:

```
Welcome 1 times

Welcome 2 times

Welcome 3 times

Welcome 4 times

Welcome 5 times
```

How do I use `for` as infinite loops?

Infinite `for` loop can be created with empty expressions, such as:

```
#!/bin/bash
for (( ; ; ))
do
    echo "infinite loops [ hit CTRL+C to stop]"
done
```

Conditional exit with break

You can do early exit with break statement inside the for loop. You can exit from within a FOR, WHILE or UNTIL loop using break. General break statement inside the for loop:

```
for I in 1 2 3 4 5
do
    statements1          #Executed for all values of 'I', up to a disaster-condition if
any.
    statements2
    if (disaster-condition)
    then
        break           #Abandon the loop.
    fi
    statements3          #While good and, no disaster-condition.
done
```

Following shell script will go through all files stored in /etc directory. The for loop will be abandoned when /etc/resolv.conf file is found.

```
#!/bin/bash
for file in /etc/*
do
    if [ "${file}" == "/etc/resolv.conf" ]
    then
        countNameservers=$(grep -c nameserver /etc/resolv.conf)
        echo "Total ${countNameservers} nameservers defined in ${file}"
        break
    fi
done
```

Early continuation with continue statement

To resume the next iteration of the enclosing FOR, WHILE or UNTIL loop use continue statement.

```
for I in 1 2 3 4 5
do
    statements1          #Executed for all values of 'I', up to a disaster-condition if
any.
    statements2
    if (condition)
    then
        continue       #Go to next iteration of I in the loop and skip statements3
    fi
    statements3
done
```

This script makes backup of all file names specified on command line. If .bak file exists, it will skip the cp command.

```
#!/bin/bash
FILES="$@"
for f in $FILES
```

```

do
    # if .bak backup file exists, read next file
    if [ -f ${f}.bak ]
    then
        echo "Skipping $f file..."
        continue # read next file and skip the cp command
    fi
    # we are here means no backup file exists, just use cp command to copy
file
    /bin/cp $f $f.bak
done

```

Check out related media

This tutorial is also available in a quick video format. The video shows some additional and practical examples such as converting all flac music files to mp3 format, all avi files to mp4 video format, unzipping multiple zip files or tar balls, gathering uptime information from multiple Linux/Unix servers, detecting remote web-server using domain names and much more.

Recommended readings:

- See all sample [for loop shell script](#) in our bash shell directory.
- Bash [for loop syntax and usage page](#) from the Linux shell scripting wiki.
- man bash
- help for
- help {
- help break
- help continue

-
- **Jadu Saikia** November 2, 2008, 3:37 pm
-

Nice one. All the examples are explained well, thanks Vivek.

seq 1 2 20

output can also be produced using jot

jot - 1 20 2

The infinite loops as everyone knows have the following alternatives.

```
while(true)
or
while :
//Jadu
```

REPLY LINK

-
- **Sean** November 4, 2008, 2:20 am
-

The last example can also be produced without the " in \$FILES":

```
#!/bin/sh

for f
do
# For-Loop body

done
```

If the " in ..." is excluded, the loop will run as if "in \$@" was given.

REPLY LINK

-
- **Andreas** November 13, 2008, 4:53 am
-

Nice explanation tutorial.

REPLY LINK

-
- **Manish** November 25, 2008, 6:33 am
-

hey vivek i tried the following syntax for for loop suggested by u but both dint work...

1.

```
#!/bin/bash
```

```
for (( c=1; c<=5; c++ ))
```

```
do
```

```
echo "Welcome $c times..."
```

```
done
```

2.

```
#!/bin/bash
```

```
for i in {1..5}
```

```
do
```

```
echo "Welcome $i times"
```

```
done
```

got error for both the syntax

1. unexpected '('

2. it printed welcome {1..5} times instead repeating it...

help..?

REPLY LINK

o **balamurugan** August 11, 2010, 5:24 am

hi manish your both coding are correct... before execute you must give the execution permission for that file... so you try following steps...

1.goto terminal

2. vim simple

3.then write the following code..

```
for (( c=1; c<=5; c++ ))
```

```
do
```

```
echo "Welcome $c times"
```

```
done
```

4.then save and quit
5.chmod 744 simple
6. ./simple
i hope surely it will help you...

[REPLY LINK](#)

- o **Manish Kumar Mishra** February 3, 2011, 5:16 am

It works properly just check it again..

[REPLY LINK](#)

- o **Dr. Stefan Gruenwald** April 1, 2011, 3:53 am

I can help you on 2. — You were not using the Bash 3.0 or higher. Upgrade your bash and it will work.

[REPLY LINK](#)

- o **Niranjan** May 10, 2011, 1:35 am

Hi All,

I have some '.gif' and '.jpg' files in a directory named Pictures in my home directory. I need to write bash script that would create 2 separate html files such as page1.html and page2.html one for gif files and the other for jpg files. And when i execute the script i need to have the html files in the Pictures directory and should have the contents as follows:

filename.jpg

filename.jpg

filename.gif

filename.gif

Please help me out. Thanks in advance

[REPLY LINK](#)

- **krist0ph3r** May 30, 2011, 2:37 pm

your problem is very easy to solve using the examples on the page.

in addition to the for loops, you will need to use the echo command, the redirection operator >> and a basic knowledge of html.

your script should do:

1. create a html file with the header, opening body tags etc.
2. have a loop for all jpg files
 - 2.1 inside the loop, print one line with the html code for an image, using the image's filename
3. close the loop, add closing html tags
- 4-6. same as 1-3, but for .gif instead of .jpg

REPLY LINK

- **lascost** December 6, 2008, 6:15 pm

i tried the last example but i seen dint work

```
#!/bin/bash
```

```
set -x
```

```
FILES="$@"
```

```
CP=$(which cp)
```

```
for f in $FILES
```

```
do
```

```
if [ -f ${f}.bak ]

then

    echo "skipping $f file"

    continue # read next file and skip cp command

fi

$CP $f $f.bak

done
```

i would like know where is the error

REPLY LINK

- o **Johnny Rosenberg** August 6, 2015, 3:06 pm

Maybe correcting your misspelled variable `FILLES` would be a step forward?

REPLY LINK

- **nixCraft** December 13, 2008, 4:56 pm

Replace

```
FILLES="$@"
```

With

```
FILES="$@"
```

REPLY LINK

- **lo2y** January 28, 2009, 10:10 am

hi guys . can any one help me . i need a script to check the file /var/log/messages every 10 minutes .and if its has the following log :
ext3_orphan_cleanup: deleting unreferenced
to apply the following command
sendsms to wut ever .
thnx alot

REPLY LINK

- **swatkat** February 12, 2009, 2:06 pm

i would like to breakk a csv file depending upon two criteria.

1. Single file should not be more than 100 lines
 2. The third column if has same value on the 100th line as that of the 101th line, the complete line should be included in the 2nd file.
- so., now, 1st file will have 99 lines and 2nd file will have 100 lines, ifthe above 2nd condition does not repeats.,

REPLY LINK

- **archana** March 16, 2009, 11:53 pm

```
for file in $(ls 0902*0010202.TLG); do
day=$(echo $file | cut -c 1-6)
grep ^203 $file | cut -d, -f3 | sort | uniq -c | while read line; do
cnt=$(echo $line | cut -d" " -f1)
acct=$(echo $line | cut -d" " -f2)
echo "Date 20${day} Account ${acct} had ${cnt} 203's" >> Feb_report.txt
```

done

done

when i run it it gives me a syntax error

```
ins@ARTMGA01> ./arc.sh
```

```
./arc.sh: syntax error at line 4: '$' unexpected
```

could you help

REPLY LINK

-
- **Navneet** April 20, 2009, 10:57 am

Good examples!! easily understood

REPLY LINK

-
- **firmit** June 7, 2009, 5:06 pm

Hi

How do I read line by line in a file, and use these in a loop? I have a file I read in (cmd max_cpu):

firefox 15

conky 1

cmds=\$(cat file)

But \$cmds now consist of n items, all being “equal” – it does not split on each line to a new array. I expected that by looping over \$cmds, I’d get a 2D array.... I did not.

Otherwise, excellent tutorial!

REPLY LINK

-
- **nixCraft** June 7, 2009, 5:56 pm
-

Try:

```
FILE=/etc/passwd
```

```
while read line
```

```
do
```

```
    # store field 1
```

```
    F1=$(echo $line|cut -d$FS -f1)
```

```
    # store field 2
```

```
    F2=$(echo $line|cut -d$FS -f6)
```

```
    # store field
```

```
    F3=$(echo $line|cut -d$FS -f7)
```

```
    echo "User \"$F1\" home directory is $F2 and login shell is $F3"
```

```
done < $FILE
```

REPLY LINK

-
- **Andi Reinbrech** November 18, 2010, 7:42 pm

I know this is an ancient thread, but thought this trick might be helpful to someone:

For the above example with all the cuts, simply do

```
set `echo $line`
```

This will split line into positional parameters and you can after the set simply say

```
F1=$1; F2=$2; F3=$3
```

I used this a lot many years ago on solaris with “set `date`”, it neatly splits the whole date string into variables and saves lots of messy cutting :-)

... no, you can't change the FS, if it's not space, you can't use this method

REPLY LINK

- **Dana Good** February 13, 2012, 3:05 am

This *is* helpful – good knowledge to have. I'll go give it a try.

Thanks!

REPLY LINK

- **Alvin** August 17, 2010, 12:09 pm

Hi Vivek,

Please help I was trying to use your code in a script similar issue trying to use a csv file with three columns (loginname,surname,firname) as input for a file that will be executed fileA

loginN,ssn,ffn

ab1pp1,ab1,pp1

bb1oo1,bb1,oo1

cc1qq1,cc1,qq1

#file to be changed cmdch.sh

echo “your login name is \$loginn, your surname is \$ssn and your firname \$ffn”

Program

#!/bin/bash

```

LINNUM=4
while read LINE;
do
LINNUM=`expr $LINENUM + 1`
done < smallops.csv
FILE=fileA.csv
while read LINE;
do
#store field 1
F1=$(echo $line|cut -d$FS -f1)
#store field 2
F2=$(echo $line|cut -d$FS -f6)
#store field 3
F3=$(echo $line|cut -d$FS -f7)
sed '{ $LINNUM s/lgn/$F1/g; $LINNUM s/ssn/$F2/g; $LINNUM s/ffn/$F3/g; }' -i
smallops.csv
done < g.csv
echo

```

REPLY LINK

- **Philippe Petrinko** August 17, 2010, 2:04 pm

@Alvin

First, this code won't work at least because it does not input fileA.csv as intended.

Second, to debug, try to break down this program, piece by piece.

For instance, it could read input more easily from fileA.csv this way:

(I have not included your [sed] instruction yet. Go step by step)

```
# backup current IFS (Internal File Separator)
```

```
IFS_backup="${IFS}"
```

```
# change IFS to directly read input file into 3 variables a,b,c

IFS=","

while read a b c

do

echo "LOGIN:$a LASTNAME:$b FIRSTNAME:$c"

done < fileA.csv


# restore IFS

IFS="${IFS_backup}"
```

Third, try to explain what you would like to do with you [sed] instruction.

Why do you start your LINENUM at 4?

What is the content of your smallops.csv? What is it for?

By the way, your [sed] instruction seems to contain a misspelled search pattern, for first field \$F1, should'nt it be [loginn] instead of [lgn] ?

You mention only ONE csv file, but your code contains: fileA.csv, smallops.csv, and g.csv ? What are they? Typos ? Errors in your code?

REPLY LINK

- **Philippe Petrinko** August 17, 2010, 2:05 pm

Sorry, (I forgot a HTML code TAG) Complete code is:

```
# backup current IFS (Internal File Separator)

IFS_backup="${IFS}"

# change IFS to directly read input file into 3 variables a,b,c

IFS=","

while read a b c

do

echo "LOGIN:$a LASTNAME:$b FIRSTNAME:$c"

done < fileA.csv

# restore IFS

IFS="${IFS_backup}"
```

REPLY LINK

- **firmit** June 7, 2009, 6:03 pm

Excellent! Thanks Vivek.

REPLY LINK

-
- **Peko** July 16, 2009, 6:11 pm
-

Hi Vivek,

Thanks for this a useful topic.

IMNSHO, there may be something to modify here

=====

Latest bash version 3.0+ has inbuilt support for setting up a step value:

```
#!/bin/bash
```

```
for i in {1..5}
```

=====

1) The increment feature seems to belong to the version 4 of bash.

Reference: <http://bash-hackers.org/wiki/doku.php/syntax/expansion/brace>

Accordingly, my bash v3.2 does not include this feature.

BTW, where did you read that it was 3.0+ ?

(I ask because you may know some good website of interest on the subject).

2) The syntax is {from..to..step} where from, to, step are 3 integers.

Your code is missing the increment.

Note that GNU Bash documentation may be bugged at this time,

because on GNU Bash manual, you will find the syntax {x..y[incr]}

which may be a typo. (missing the second “..” between y and increment).

see <http://www.gnu.org/software/bash/manual/bashref.html#Brace-Expansion>

The Bash Hackers page

again, see <http://bash-hackers.org/wiki/doku.php/syntax/expansion/brace>

seems to be more accurate,

but who knows ? Anyway, at least one of them may be right... ;-)

Keep on the good work of your own,

Thanks a million.

— Peko

REPLY LINK

- **nixCraft** July 16, 2009, 6:31 pm

@ Peko,

Thanks for pointing out ranges vs step value. I've updated the FAQ.

REPLY LINK

- **Peko** July 16, 2009, 8:04 pm

Yes.

But you misspelled the syntax with an extra dot "." after "START"

not {START...END..INCREMENT}

but {START..END..INCREMENT}

;-)

— Peko

REPLY LINK

- **Michal Kaut** July 22, 2009, 6:12 am

Hello,

is there a simple way to control the number formatting? I use several computers, some of which have non-US settings with comma as a decimal point. This means that

`for x in $(seq 0 0.1 1)` gives 0 0.1 0.2 ... 1 on some machines and 0 0,1 0,2 ...

1 on other.

Is there a way to force the first variant, regardless of the language settings? Can I, for example, set the keyboard to US inside the script? Or perhaps some alternative

to `$x` that would convert commas to points?

(I am sending these as parameters to another code and it won't accept numbers with commas...)

The best thing I could think of is adding `x=`echo $x | sed s/,/./`` as a first line

inside the loop, but there should be a better solution? (Interestingly, the sed command does not seem to be upset by me rewriting its variable.)

Thanks,

Michal

[REPLY LINK](#)

-
- **Peko** July 22, 2009, 7:27 am
-

To Michal Kaut:

Hi Michal,

Such output format is configured through LOCALE settings.

I tried :

```
export LC_CTYPE="en_EN.UTF-8"; seq 0 0.1 1
```

and it works as desired.

You just have to find the exact value for LC_CTYPE that fits to your systems and your needs.

Peko

REPLY LINK

-
- **Peko** July 22, 2009, 2:29 pm
-

To Michal Kaus [2]

Ooops – ;-)

Instead of LC_CTYPE,

LC_NUMERIC should be more appropriate

(Although LC_CTYPE is actually yielding to the same result – I tested both)

By the way, Vivek has already documented the matter

: <http://www.cyberciti.biz/tips/linux-find-supportable-character-sets.html>

— Peko

REPLY LINK

-
- **VIKAS** July 22, 2009, 3:58 pm
-

Excellent stuff... keep up the good work.

REPLY LINK

-
- **Brad Landis** October 26, 2009, 3:26 pm
-

Comment 12 was really helpful. I was trying to split up a log file by date, such as logfile.20091026 , without having to use grep a million times. I'm kind of disappointed I couldn't find a one-liner to do so, but I will take what I can get :).

REPLY LINK

-
- **nixCraft** October 26, 2009, 4:47 pm
-

@Brad,

Try this without grep or cut using bash parameter expansion :

```
file="logfile.20091026"
```

```
log="${file%%.*}"
```

```
date="${file##*.}"
```

```
echo $log
```

```
echo $date
```

HTH

REPLY LINK

-
- **Philippe Petrinko** October 26, 2009, 4:52 pm

H i vivek,

Just wondering why you don't amend the typo I pointed out:

<>

REPLY LINK

-
- **Brad Landis** October 26, 2009, 4:53 pm

I think you misunderstood. I'm going line by line, and converting the dates at the beginning of the line, such as "Sep 12", and copying that line from logfile to logfile.20090912.

My script is *really* slow though, with the conversion of the month name to a number. I've tried using the date command, and my own function, and both take 7 seconds to

process 10,000 lines. It doesn't seem like a long time, but I've got a lot of log files to process on multiple machines.

I don't guess you'd know a faster trick, would you?

REPLY LINK

-
- **nixCraft** October 26, 2009, 6:37 pm
-

@Brad, yes, I did misunderstood your post. If I were you I will try out awk.

@Philippe,

Thanks for the heads up. The faq has been updated.

REPLY LINK

-
- **TheBonsai** October 30, 2009, 6:13 am
-

@Peko:

(I'm the operator of bash-hackers.org/wiki, that's why I found this page):

Regarding Bash documentation for brace expansion (increment syntax), actually I'm right and the documentation is wrong (a rare situation!). I reported it to the list.

REPLY LINK

-
- **Philippe Petrinko** October 30, 2009, 8:35 am
-

To Vivek:

Regarding your last example, that is : running a loop through arguments given to the script on the command line, there is a simpler way of doing this:

instead of:

```
# FILES="$@"  
# for f in $FILES  
# use the following syntax  
for arg  
do  
# whatever you need here – try : echo "$arg"  
done  
Of course, you can use any variable name, not only "arg".
```

To TheBonsai: Welcome Buddy!

Fine! I am happy to see 2 great FOSS web sites now related !

[REPLY LINK](#)

-
- **tdurden** November 10, 2009, 9:32 pm
-

Command line while loop.. Very handy..

Say you wanted to rename all the files in a specific dir..

Create a file with the contents you want to rename

(ls -l | awk '{print \$9}' > asdf or something)

Contents of asdf:

file1

file2

file3

file4

```
cat asdf | while read a ; do mv $a $a.new ; done
```

ls -l

```
asdf file1.new file2.new file3.new file4.new
```

I have used this while command for many things from simply renaming files to formatting and labling new SAN luns..

REPLY LINK

- **Dr. Stefan Gruenwald** April 1, 2011, 4:00 am

There are much easier ways to do this – also it works only for extensions. How do you change the middle of the file name or a few characters on the left?

Here is the regular way of what you just did:

```
for i in *; do mv $i $i.new; done
```

REPLY LINK

- **TheBonsai** November 11, 2009, 7:07 am

There are 2 problems and one optical flaw with your code:

- (1) You should use `read -r` without any variable name given, to use the default `$REPLY` (due to a specific behaviour of `read`, see `manpage`)
- (2) You should quote `$a`
- (3) Useless use of `cat` :)

REPLY LINK

- **Philippe Petrinko** November 11, 2009, 11:25 am

To `tdurden`:

Why wouldn't you use

1) either a `[for]` loop

```
for old in * ; do mv ${old} ${old}.new; done
```

2) Either the `[rename]` command ?

excerpt from “`man rename`” :

RENAME(1) Perl Programmers Reference Guide RENAME(1)

NAME

rename – renames multiple files

SYNOPSIS

```
rename [ -v ] [ -n ] [ -f ] perlexpr [ files ]
```

DESCRIPTION

“rename” renames the filenames supplied according to the rule specified as the first argument. The perlexpr argument is a Perl expression which is expected to modify the `$_` string in Perl for at least some of the filenames specified. If a given filename is not modified by the expression, it will not be renamed. If no filenames are given on the command line, filenames will be read via standard input.

For example, to rename all files matching “*.bak” to strip the extension, you might say

```
rename 's/\.bak$//' *.bak
```

To translate uppercase names to lower, youâ€™d use

```
rename 'y/A-Z/a-z/' *
```

— Philippe

REPLY LINK

-
- **TheBonsai** November 11, 2009, 11:49 am

Note for rename(1): There exist two major variants on Linux system. One non-Perl originating in the RedHat area, and one Perl, originating in the Debian area.

REPLY LINK

-
- **Sean** November 11, 2009, 2:42 pm
-

To tdudden:

I would also replace “ls -l | awk ‘{print \$9}’” with just “ls”. Otherwise you’ll run into issues with files that have spaces in it. As far as using:

```
for i in *;
```

vs

```
for i in $(ls);
```

I personally prefer “\$(ls)” or “\$(find .)”. This provides more control over what files I’m going to be looping through. For instance:

```
$(ls -F | grep -v “V$”)
```

or

```
$(ls -A)
```

[REPLY LINK](#)

-
- o **A.R.Memon** December 4, 2012, 9:58 am

```
#!/bin/bash
```

```
echo “Bash version ${BASH_VERSION}...”
```

```
for i in {0..10..2}
```

```
do
```

```
echo “Welcome to my new script $i times”
```

```
done
```

```
help ..... :)
```

[REPLY LINK](#)

- **Philippe Petrinko** November 11, 2009, 3:03 pm

To Sean:

CMIIAW :

try the following commands:

```
# touch “file with spaces in name”
```

```
# for f in *; do echo "" ;done
```

```
...
```

```
...
```

which shows that there is no need to use [for f in \$(ls)] instead of [for f in *]

Doesn't it ?

— Philippe

[REPLY LINK](#)

-
- **Philippe Petrinko** November 11, 2009, 3:14 pm
-

Sorry Sean, my last post was truncated,

due to limitations of this form used to post comments. (impossible to use Greater_than and Less_than characters)

I meant, use the following:

```
# touch "file with spaces in name"
```

```
# for f in *; do echo "${f}";done
```

```
...
```

:file with spaces in name:

```
...
```

[REPLY LINK](#)

-
- **Sean** November 11, 2009, 8:10 pm
-

Sorry for the confusion, I understand that "for i in *;" will not have any issues with spaces. I was referring to the `ls -l | awk '{print $9}'` having issues with spaces. The reason I choose to use `$(ls)` instead of `*` is for filtering out unwanted files e.g. `$(ls -F | grep -v "\$")`

[REPLY LINK](#)

-
- **Philippe Petrinko** November 11, 2009, 9:16 pm
-

To Sean:

But then, that's wrong.

```
[ for f in $(ls -F|grep -v "V$") ]
```

won't process appropriately spaces in filename.

Check :

```
# touch "file with spaces in name"
```

```
# for f in $(ls -F|grep -v "V$"); do echo "${f}";done
```

:file:

:with:

:spaces:

:in:

:name:

[REPLY LINK](#)

-
- **Philippe Petrinko** November 11, 2009, 9:18 pm
-

The best tool to filter files and process them

is [find] piped to [xargs] (with zero-ended filenames)

[REPLY LINK](#)

-
- **Philippe Petrinko** November 11, 2009, 9:25 pm
-

To sean:

But if you want to exclude files from globbing,

[bash] has the [extglob] option.

Let's say you want to process every file except files ending by a "V", just type

```
# for f in !(*V); do echo "${f}";done
```

REPLY LINK

-
- **Philippe Petrinko** November 11, 2009, 9:27 pm
-

If you set the shell option extglob, Bash understands some more powerful patterns.
Here, a is one or more pattern, separated by the pipe-symbol (|).

?() Matches zero or one occurrence of the given patterns

*() Matches zero or more occurrences of the given patterns

+() Matches one or more occurrences of the given patterns

@() Matches one of the given patterns

!() Matches anything except one of the given patterns

REPLY LINK

-
- **Philippe Petrinko** November 11, 2009, 9:28 pm
-

source: <http://www.bash-hackers.org/wiki/doku.php/syntax/pattern>

REPLY LINK

-
- **Narender** November 12, 2009, 6:47 am
-

I have two files here X.a and y.a Now what i need is i need to substitute
CvfdDisk_sdb/c/d/e in lines of Node CvfdDisk_XXX in the order CvfdDisk_sdb/c/f/g first
word of each line of x.a exists. how can i do in shell scripting i can get the first word of
each line of X.a using awk /cut but to replace these in y.a i am not getting it ... any help
here ?

[Raj]\$ cat x.a

CvfdDisk_sdb /dev/sdb # host 0 lun 1 sectors 4840746976 sector_size 512 inquiry

[AMCC 9550SX-12M DISK 3.08] serial AMCC ZAJBSXJFF92A9D003C6A

CvfdDisk_sdc /dev/sdc # host 0 lun 0 sectors 3906148319 sector_size 512 inquiry

[AMCC 9550SX-12M DISK 3.08] serial AMCC ZAJ8MJKFF92A9D001FEC

CvfdDisk_sdf /dev/sdf # host 0 lun 1 sectors 4840746976 sector_size 512 inquiry

```
[AMCC 9550SX-12M DISK 3.08] serial AMCC ZAJBSXJFF92A9D003C6A
CvfsDisk_sdg /dev/sdg # host 0 lun 0 sectors 3906148319 sector_size 512 inquiry
[AMCC 9550SX-12M DISK 3.08] serial AMCC ZAJ8MJKFF92A9D001FEC
[naren@Beas dxall]$ cat y.a
[StripeGroup Metafiles]
Metadata Yes
Status UP
Read Enabled
Write Enabled
Journal Yes
StripeBreadth 1280K
Node CvfsDisk_sdb 0
[StripeGroup datafiles1]
Metadata Yes
Status UP
Read Enabled
Write Enabled
StripeBreadth 1024K
Node CvfsDisk_sdc 0
[StripeGroup datafiles2]
Metadata Yes
Status UP
Read Enabled
Write Enabled
StripeBreadth 1280K
Node CvfsDisk_sdd 0
[StripeGroup datafiles3]
Metadata Yes
Status UP
Read Enabled
Write Enabled
```

StripeBreadth 1024K

Node CvfsDisk_sde 0

[REPLY LINK](#)

-
- o **nixCraft** November 12, 2009, 1:47 pm

@Narender,

Your post is offtopic. I suggest you use our shell scripting [forum](#) for question.

[REPLY LINK](#)

-
- **Sean** November 12, 2009, 3:26 pm

To Philippe:

You are right,

for i in \$(ls)

will break up files with spaces if IFS isn't set to just the newline character. I don't believe this is consistent across distributions. So the for loop should have

export IFS=\$'\n'

before it. I also use find in for loops when I want to look through the directory contents, but this isn't always desired. Thanks for the info about extglob, I haven't done much with extended globbing in bash.

[REPLY LINK](#)

-
- **Philippe Petrinko** November 12, 2009, 3:44 pm
-

To Sean:

Right, the more sharp a knife is, the easier it can cut your fingers...

I mean: There are side-effects to the use of file globbing (like in [for f in *]) , when the globbing expression matches nothing: the globbing expression is not substituted.

Then you might want to consider using [nullglob] shell extension, to prevent this.

see: <http://www.bash-hackers.org/wiki/doku.php/syntax/expansion/globs#customization>

Devil hides in detail ;-)

REPLY LINK

-
- **The_Catalanish** December 4, 2009, 11:22 am

Response to the tip number 12

At this script, It's missing the following line

FS=':'

in the variables declaration

(you forgot the delimiter field, for the cut command)

:-P

The_Catalanish

REPLY LINK

-
- **Rilif** December 10, 2009, 9:27 pm

```
#!/usr/bin/ksh
```

```
for i in `cat /input`
```

```
do
```

```
bdf | grep file_system | grep -vE '^A|B|C' | awk '{ print $4}' | while read output;
```

```
do
```

```
file_system=$(echo $output | awk '{ print $1}' | cut -d'%' -f1 )
```

```
partition=$(echo $output | awk '{ print $2}' )
```

```
if [ $file_system -ge 60 ]; then
echo "don't run the sync $partition ($file_system%) "
else
rsync $i
fi
done
done
```

The problem with the logic I'm having is I do not want the script to exit(as it does now) the loop once the file_system area reaches 60%. I want it to continue to retest bdf and continue the loop once disk usage drops below 60%. Any Ideas?

REPLY LINK

-
- **Philippe Petrinko** December 11, 2009, 12:17 pm
-

To Rilif:

1) I assume you use [bdf] on UNIX system – because Linux equivalent is [df] – and I cannot be of help because I cannot test your script on my Linux boxes.

2) This seems to be a specific programming debugging problem and out of this topic scope – There may be a better place to post that kind of topic – A programmer forum for instance.

Best regards.

REPLY LINK

-
- **dee** December 13, 2009, 10:27 am
-

do any one know how to write this script in steps?

as so `/// ./ test 10 ///` The first argument [1] will ex. to create a multiple users, groups, cn, dn, etc for ldap in one or two scripts but from command line. you would just enter file then the number of attributes to build.

this is a headache for me since i'm new at this.

[REPLY LINK](#)

-
- **Philippe Petrinko** December 13, 2009, 10:54 am

To Dee:

0) The first part of your first sentence is incomprehensible – this may be because text that you entered is altered, it may contain HTML-like syntax that is interpreted by this comment form. (By the way, Vivek Gite would be welcomed to tell us how to prevent this. TIA :-))

1) LDAP syntax is off-topic.

2) You'll find appropriate resources in LDAP forums/sites – just find out.

3) We may be in position to help you to build a [for] loop, assuming you do your part of the job by providing the basic LDAP instructions to create user, for instance. Try to create at least a LDAP

object by yourself on the command-line, then provide us the code, and as much as possible further explanation please.

[REPLY LINK](#)

-
- **nixCraft** December 13, 2009, 10:58 am

There was no html tag or anything else in comment.

@dee, if you need to attach code use `<pre>` tags.

REPLY LINK

- **Philippe Petrinko** December 13, 2009, 11:44 am

Thanks Vivek – But I am afraid I do not get it right – what does “pre” mean ? (I understand you wrote the “less than” tag, and “greater than” tag – but why “pre” ?

And are you sure these are the only ones two use ?

REPLY LINK

- **nixCraft** December 13, 2009, 1:26 pm

@Philippe,

All allowed html tags are displayed below the form itself. It is wordpress that converts those symbol and syntax is

```
<pre>#!/bin/bash
```

```
echo "Hello world!"
```

```
</pre>
```

HTH

REPLY LINK

- **dee** December 13, 2009, 1:41 pm

first file make: create_user_idif.sh.txt

```
#!/bin/bash
```

```
echo "dn: $2 $3,o=dmacc,dc=bad4dee,dc=com" > $1
```

```
echo "changetype: add" >> $1
```

```
echo "objectclass: top" >> $1
```

```
echo "objectclass: person" >> $1
```

```
echo "objectclass: organizationalPerson" >> $1
```

```
echo "objectclass: inetOrgPerson" >> $1
```

```
echo "cn: $2 $3" >> $1
```

```
echo "givenName: $2" >> $1
```

```
echo "sn: $3" >> $1
```

```
echo "uid: ${2:0:1}$3" >> $1
```

```
echo "mail: $4" >> $1
```

```
mass_user.txt
```

```
Scott,brown,sbrown@bad4dee.com\n
```

```
karla,smith,ssmith@bad4dee.com\n
```

```
sam,goodie,sgoodie@bad4dee.com\n
```

```
marge,jones,mjones@bad4dee.com
```

```
mass_add.sh.txt
```

```
#!/bin/bash
```

```
u1=`cat $1 | tr -d '\n'`
```

```
#u2=`echo $u1 | tr \|| '\n'`
```

```
u2=`echo $u1 | sed s/\||/\n/`
```

```
echo "DBG"
```

```
echo -e $u1
```

```
echo "DBG"
```

```
echo -e $u1 | ( IFS=, ; while read fname lname mail;
```

```
do
```

```
echo -e "Creating $fname $lname\n\n"
```

```
./create_user_ldif.sh.txt ./user.ldif $fname $lname $mail
```

```
cat ./user.ldif
```

```
echo -e "\n\n"
```

```
done )
```

```
./mass_add.sh.txt mass_user.txt
```

This ex: will pull from a list but same out come i do not know how to write another script to pull the attributes i need from the command line like ./test 100 and that command will pull only a 100 users id's from idif.txt out of 1000 generated.

This next samples of code will file in the attributes for you.

first file make: create_user_ldif.sh.txt

```
#!/bin/bash
```

```
echo "dn: $2 $3,o=dmacc,dc=badldee,dc=com" > $1
```

```
echo "changetype: add" >> $1
```

```
echo "objectclass: top" >> $1
```

```
echo "objectclass: person" >> $1
```

```
echo "objectclass: organizationalPerson" >> $1
```

```
echo "objectclass: inetOrgPerson" >> $1
```

```
echo "cn: $2 $3" >> $1
```

```
echo "givenName: $2" >> $1
```

```
echo "sn: $3" >> $1
```

```
echo "uid: ${2:0:1}$3" >> $1
```

```
echo "mail: $4" >> $1
```

Then:

mass_add.sh.txt

```
#!/bin/bash
```

```
u1=`cat $1 | tr -d '\n'`
```

```
#u2=`echo $u1 | tr \|| '\n'`
```

```
u2=`echo $u1 | sed s/\||/\n/`
```

```
echo "DBG"
```

```
echo -e $u1
```

```
echo "DBG"
```



```
echo -e $u1 | ( IFS=, ; while read fname lname mail;

do

echo -e "Creating $fname $lname\n\n"

./create_user_ldif.sh.txt ./user.ldif $fname $lname $mail

cat ./user.ldif

echo -e "\n\n"

done )
```

last:

adduserfile1.txt

```
user0,boo,us1@badldee.com\n
```

```
user1,boo,us1@badldee.com\n
```

```
user2,boo,us1@badldee.com\n
```

```
user3,boo,us1@badldee.com\n
```

```
user4,boo,us1@badldee.com\n
```

```
user5,boo,us1@badldee.com\n
```

```
./mass_add.sh.txt adduserfile1.txt
```

This is what i'm working on now ? i still do not know how to tie in C++ or bash script this code to work with command line, so i can control the out come of created users.

```
#!/bin/bash
```

```
Set objRootDSE = GetObject("LDAP://rootDSE")
```

```
Set objContainer = GetObject("LDAP://cn=Users," & _
```

```
objRootDSE.Get("defaultNamingContext"))
```

```
For i = 1 To 1000
```

```
Set objLeaf = objContainer.Create("User", "cn=UserNo" & i)
```

```
objLeaf.Put "sAMAccountName", "UserNo" & i
```

```
objLeaf.SetInfo
```

```
Next
```

```
WScript.Echo "1000 Users created."
```

```
example: [root]#./test 10
```

```
usid:UserNo1 gid:manager cn:Bob dn:bHarison
```

UserNo2

UserNo3

^^^^^^

UserNo10

This script makes a 1000 users. However i can not control the out come. For example: from the command line I would like it to stop at 100 users by typing in ./test 100. Using agrv [1]. so when I type a number after the file name it will create a list and print that record to the screen. I do not know bash that well as C++ and it is not helping because the char.. are diff...

Edited by admin. Reason: *This is not a forum, this is a blog. If you need further help please try our forum @nixcraft.com.*
REPLY LINK

-
- **Philippe Petrinko** December 13, 2009, 2:51 pm
-

To Dee:

- 1) Man, with a 3-users-sample instead of hundreds, we would have figured out, don't you think so?
- 2) Well that's a start. May be Vivek would like to wipe this post out, and create a new topic: "Of mice, LDAP and loops" ;-) ???

REPLY LINK

-
- **Dominic** January 14, 2010, 10:04 am
-

There is an interesting difference between the exit value for two different for looping structures (hope this comes out right):

```
for (( c=1; c<=2; c++ )) do echo -n "inside (( )) loop c is $c, "; done; echo
```

```
"done (( )) loop c is $c"
```

```
for c in {1..2}; do echo -n "inside { } loop c is $c, "; done; echo "done { }
```

```
loop c is $c"
```

You see that the first structure does a final increment of `c`, the second does not. The first is more useful IMO because if you have a conditional break in the for loop, then you can subsequently test the value of `$c` to see if the for loop was broken or not; with the second structure you can't know whether the loop was broken on the last iteration or continued to completion.

[REPLY LINK](#)

-
- **Dominic** January 14, 2010, 10:09 am

sorry, my previous post would have been clearer if I had shown the output of my code snippet, which is:

```
inside (( )) loop c is 1, inside (( )) loop c is 2, done (( )) loop c is 3
```

```
inside { } loop c is 1, inside { } loop c is 2, done { } loop c is 2
```

[REPLY LINK](#)

-
- **Dmitry** March 8, 2010, 5:47 pm

What if I would like to put everything into one line?

[REPLY LINK](#)

-
- **Dominic** March 9, 2010, 5:29 am
-

Dmitry, please give a little more detail about what you are trying to achieve. You can see from my examples above that there is no problem to put a simple loop on one line. Basically you use semicolons (;) instead of line breaks.

[REPLY LINK](#)

-
- **Dmitry** March 9, 2010, 1:56 pm
-

Dominic,

thaks a lot for your quick answer. You have answered on my question but I'm still having problems. Please take a look at this.

```
dmitry@elastix-laptop:~/projects_cg/match_delays/source$ for i in $(seq 1 2 20)
```

```
> do
```

```
> echo "Welcome $i times"
```

```
> done
```

```
Welcome 1 times
```

```
Welcome 3 times
```

```
Welcome 5 times
```

```
Welcome 7 times
```

```
Welcome 9 times
```

```
Welcome 11 times
```

```
Welcome 13 times
```

```
Welcome 15 times
```

```
Welcome 17 times
```

```
Welcome 19 times
```

```
dmitry@elastix-laptop:~/projects_cg/match_delays/source$ for i in $(seq 1 2 20); do;
```

```
echo "Welcome $i times" ; done
```

```
bash: syntax error near unexpected token `;'
```

dmitry@elastix-laptop:~/projects_cg/match_delays/source\$

What am I missing here?

[REPLY LINK](#)

-
- **Philippe Petrinko** March 9, 2010, 2:05 pm
-

@Dmitry

You are missing : Reading The Fantastic Manual. :-)

```
man bash
```

```
...
```

```
for name [ in word ] ; do list ; done
```

```
...
```

```
for (( expr1 ; expr2 ; expr3 )) ; do list ; done
```

```
...
```

There should not be any “;” following the [do].

```
for i in $(seq 1 2 20); do echo "Welcome $i times" ; done
```

Good ol’ one: “When any thing goes wrong – (re) Read the manual”

[REPLY LINK](#)

-
- **Philippe Petrinko** March 9, 2010, 2:34 pm
-

@Dmitry

And, again, as stated many times up there, using [seq] is counter productive, because it requires a call to an external program, when you should Keep It Short and Simple, using only bash internals functions:

```
for ((c=1; c<21; c+=2)); do echo "Welcome $c times" ; done
```

(and I wonder why Vivek is sticking to that old solution which should be presented only for historical reasons when there was no way of using bash internals.

By the way, this historical recall should be placed only at topic end, and not on top of the topic, which makes newbies sticking to the not-up-to-date technique ;-)

REPLY LINK

-
- **Dmitry** March 9, 2010, 2:49 pm

Philippe, thank you, all works perfect know.

Here is the partial excuse that I was reading this thread instead of bash manual

<http://www.google.es/search?q=bash+for+loop&ie=utf-8&oe=utf-8&aq=t&rls=com.ubuntu:en-US:official&client=firefox-a>

REPLY LINK

-
- **Dominic** March 9, 2010, 3:23 pm

It is strange that a do/done loop works if there is CR (end of line) after the do, but not if there is a semi-colon. I can see why this was confusing for you Dmitry, because it's not logical. My guess is that the acceptance of CR after do was added because people wanted to lay out code this way, but the bash coders forgot to allow the semicolon alternative. As Philippe points out, if you follow the manual strictly, it works fine.

REPLY LINK

-
- **Dmitry** March 9, 2010, 3:27 pm
-

Ok. Thank you again!

REPLY LINK

-
- **nixCraft** March 9, 2010, 5:30 pm
-

@Philippe,

I've just updated the faq and also deleted large data sample posted by dee user.

REPLY LINK

-
- **TheBonsai** March 9, 2010, 5:55 pm
-

@ Dominic

Yes, it's not quite intuitive, right. What you mean is the semicolon or the newline as list separator (list as the grammar construct defined in the manual, respectively by ISO9945). After a `do`, a list is expected, but a list isn't **introduced** with a list separator. After a `do`, the shell awaits more input, just like after an opening quote character. In interactive mode, it also displays the continuation prompt `PS2` instead of `PS1` (it would display `PS1` for list continuation).

It's not right that the Bash coders "forgot" it. Inspecting the grammar rules of POSIX XCU Sec. 2.10.2 doesn't show a special rule here (it would have to be a special exceptional rule that extra allows a semicolon here).

REPLY LINK

-
- **TheBonsai** March 9, 2010, 6:06 pm
-

Me again.

From all Bourne-like shells I just “tested”, only ZSH seems to support a semicolon as a start of a list (also in the case after the `do`). This is nice, but that’s all. It’s not a bug to not do so.

REPLY LINK

-
- **Dominic** March 9, 2010, 6:54 pm
-

@ TheBonsai

Interesting. I accept that it is not a bug, but I still think it is confusing. It seems logical to us lesser mortals that in bash semicolon=newline, and in other situations I think this is true, but not here.

```
# this works
```

```
for (( c=1; c<=2; c++ )) do
```

```
#any number of blank lines or even comments
```

```
echo $c; done
```

```
# so does this:
```

```
for (( c=1; c<=2; c++ )) do echo $c; done
```

```
# but this, where we substitute a semi-colon for the blank line(s) above,
```

doesn't:

```
for (( c=1; c<=2; c++ )) do; echo $c; done
```

REPLY LINK

-
- **Philippe Petrinko** March 9, 2010, 11:10 pm
-

@Dominic

You are missing the point.

If you read our Unix pioneers, you will remember:

– Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
[E. Raymond]

see http://en.wikipedia.org/wiki/Unix_philosophy

What's the point of spending hours to code on one line?

– First, It does not give any optimization, it does not save any execution time. Your code will only be more difficult to read, check, debug.

Defensive programming rules include this: Write one instruction per line.

– Second, You still wanna code all on one line ?

Big deal. The manual gave you the right way.

So stick to it, or leave it, and skip to the next real problem, instead of wasting time and energy pointlessly, my dear Linux enthusiast.

REPLY LINK

-
- **Sean** March 9, 2010, 11:15 pm
-

I have a comment to add about using the builtin **for ((...))** syntax. I would agree the builtin method is cleaner, but from what I've noticed with other builtin functionality, I had to check the speed advantage for myself. I wrote the following files:

builtin_count.sh:

```
#!/bin/bash

for ((i=1;i<=1000000;i++))

do

echo "Output $i"

done
```

seq_count.sh:

```
#!/bin/bash

for i in $(seq 1 1000000)

do

echo "Output $i"

done
```

And here were the results that I got:

time ./builtin_count.sh

```
real 0m22.122s
user 0m18.329s
sys 0m3.166s
time ./seq_count.sh
real 0m19.590s
user 0m15.326s
sys 0m2.503s
```

The performance increase isn't too significant, especially when you are probably going to be doing something a little more interesting inside of the for loop, but it does show that builtin commands are not necessarily faster.

REPLY LINK

o **Andi Reinbrech** November 18, 2010, 8:35 pm

The reason why the external seq is faster, is because it is executed only once, and returns a huge splurb of space separated integers which need no further processing, apart from the for loop advancing to the next one for the variable substitution.

The internal loop is a nice and clean/readable construct, but it has a lot of overhead. The check expression is re-evaluated on every iteration, and a variable on the interpreter's heap gets incremented, possibly checked for overflow etc. etc.

Note that the check expression cannot be simplified or internally optimised by the interpreter because the value may change inside the loop's body (yes, there are cases where you'd want to do this, however rare and stupid they may seem), hence the variables are volatile and get re-evaluated.

I.e. bottom line, the internal one has more overhead, the "seq" version is equivalent to either having 1000000 integers inside the script (hard coded), or reading once from a text file with 1000000 integers with a cat. Point being that it gets executed only once and becomes static.

OK, blah blah fishpaste, past my bed time :-)

Cheers,

Andi

REPLY LINK

- **Philippe Petrinko** November 18, 2010, 9:06 pm

@Andi

> OK, blah blah fishpaste, past my bed time :-)

Interesting comments anyway!

REPLY LINK

- **Philippe Petrinko** March 10, 2010, 9:13 am

@Sean

1) Again, when your only programming concern that last will be the optimization of your loops, you could invest time into such timings.

This would be when there are no other bugs in your code, which I wish you to enjoy ASAP.

2) But then you may find that the real problem/bottleneck is not a for loop.

As Rob Pike said : "Measure. Do not tune for speed until your performance analysis tool tells you which part of the code overwhelms the rest."

[[http://en.wikipedia.org/wiki/Unix_philosophy#Pike: Notes on Programming in C](http://en.wikipedia.org/wiki/Unix_philosophy#Pike:_Notes_on_Programming_in_C)]

3) I agree with you when you say that your code is not relevant as a timing of real-sized programs.

4) Relating to your benchmark "builtin vs. external" commands

test the builtin [for i in {1..1000000}]

and you will see that it is very close to [for i in \$(seq 1 1000000)]

REPLY LINK

-
- **2012 Doom Day** March 10, 2010, 7:01 pm
-

You are missing : Reading The Fantastic Manual. :-)

A.. uh.. ?

The lack of examples in the bash man page is the main reason to *avoid* man page.

Everyone, knows how to use them. Syntax is all good, only if you know the bash and UNIX in and out. This is the main reason why most people purchase bash and shell scripting books from O'reilly or Amazon. Rest of freeloaders depends upon Google and site like this to get information quickly. man pages are for gurus; for all new user examples are the best way to get started.

[REPLY LINK](#)

-
- **Philippe Petrinko** March 10, 2010, 9:31 pm
-

@ 2012DD

I agree – I use any of resource I need, man page, –help page, info page, web pages, books.

The thing is: you should try to read man page once.

And actually, if he did, he would have find the syntax.

Vivek's web site and contributions do not prevent you of reading the “fantastic” manual.

And, as usual, the ones that issue the bitter critics will not move a finger to enhance the manual.

You say man page lacks good examples?

Did you ever try to contribute to any man pages ?

Hope you did.

[REPLY LINK](#)

-
- **TheBonsai** March 11, 2010, 11:30 am
-

The Bash manual page isn't meant as tutorial. It's a syntax, grammar and behaviour reference. However, it contains the knowledge to explain why a for loop using `seq`, one using brace expansion and one using builtin arithmetics have the performance relations they actually show when you execute them. The point is to make a relation between abstract descriptions and real execution behaviour. If such things really count, however, I suspect you code in the wrong language.

REPLY LINK

-
- **Philippe Petrinko** March 13, 2010, 9:50 pm

@Bonsai

> The Bash manual page isn't meant as tutorial. It's a syntax, grammar and behaviour reference.

Actually, no. A man page can/should contain a EXAMPLE section.

check : http://en.wikipedia.org/wiki/Man_page#Layout

I am pretty happy when the Example section is relevant. And when you want some more, nothing prevents you to try to add new examples... Let's contribute!

REPLY LINK

-
- **TheBonsai** March 14, 2010, 12:34 am

@Philippe

How many examples would you place there, to show the "common usage of Bash"? I agree that manpages usually should contain examples, but I think this would be too much. Huge manpages (huge because they describe a huge set of functionality) contain very small/no examples.

REPLY LINK

-
- **2012 Doom Day** March 14, 2010, 5:39 am
-

@Philippe,

I am learning Bash scripting and that is why I'm here and its wiki. Take a look at Solaris UNIX man page, most of them have good set of examples. Another candidate is FreeBSD, they also have good set of examples. Just discovered that our HP-UX came with printed "Posix Born Shell Scripting Manual".

>Did you ever try to contribute to any man pages ?

No, I'm learning and if I **contribute** anything, I'm dam sure most shell scripting gurus will eat me alive, as I'm not experienced coder. Once I tried to help someone on comp.unix.shell and most of other people on list were so mean to my code that I almost stopped visiting comp.unix.shell. **Beep** those bastards!

Edited by admin

[REPLY LINK](#)

-
- **TheBonsai** March 19, 2010, 5:21 am
-

@2012 Doom Day

Contribution is more than just knowing code. Alone the fact that you write here is a contribution (to the community). Translations, documentation, searching bugs, helping others, sharing expiriences, ...

[REPLY LINK](#)

-
- **Philippe Petrinko** March 19, 2010, 8:35 am
-

Quite right, Bonsai,

and you can contribute to Wikibooks, Wikipedia, and Vivek's Wiki using the books you own and all you have learnt, no one will ever prevent you of doing so, assuming you improve the content. Don't be shy and be confident on your capacities.

See, I wrote loads of questionable comments and Vivek has not banned me [yet] ;-).

[REPLY LINK](#)

-
- **Chris Cheltenham** April 2, 2010, 7:33 pm

Do you know why this doesn't output anything?

```
#!/bin/bash
```

```
for i in $(cat /$HOME/client_list.txt)
```

```
do
```

```
echo $i > /home/$i_file.log
```

```
done
```

[REPLY LINK](#)

-
- **Philippe Petrinko** April 2, 2010, 8:47 pm

@Chris C

If you want a good explanation, first try to ask a good question and explain:

1a) What you want to do with this program

1b) what your program is supposed to do.

Second:

Mainly, it will fail because there will not be variable expansion `$i_file.log` should be

`${i}_file.log`

Read again about variable expansion.

And:

2a) in your " cat ", there should not be a leading slash before `$HOME` (because `$HOME`

contains a leading slash) – anyway this wont prevent it from working – but may come to bugs someday.

2c) if a line in client_list.txt does contains spaces, what do you think this would do? Use quotes.

2d) If it still fails, check your permissions to create and overwrite a file in “/home” directory

2e) As said many times, you do not need to use ” for + cat ” to use the content of a file. Just use a while loop :

```
while read i
```

```
echo ${i}
```

```
done < $HOME/client_list.txt
```

REPLY LINK

-
- **Moihan** May 9, 2010, 11:36 am

Sir,

This is a new post. From a file in unix server with a column of temperature, I want to extract a number if it goes greater than 100. Normally it will be in 60 – 80 range. Can u suggest a bash script?

REPLY LINK

-
- **nixCraft** May 9, 2010, 1:00 pm

Use our shell scripting forum [for questions](#).

REPLY LINK

-
- **Anthony Thyssen** June 4, 2010, 6:53 am
-

The {1..10} syntax is pretty useless as you can use a variable with it!

```
limit=10
```

```
echo {1..${limit}}
```

```
{1..10}
```

You need to eval it to get it to work!

```
limit=10
```

```
eval "echo {1..${limit}}"
```

```
1 2 3 4 5 6 7 8 9 10
```

‘seq’ is not available on ALL system (MacOSX for example)

and BASH is not available on all systems either.

You are better off either using the old while-expr method for computer compatibility!

```
limit=10; n=1;
```

```
while [ $n -le 10 ]; do
```

```
    echo $n;
```

```
    n=`expr $n + 1`;
```

```
done
```

Alternatively use a seq() function replacement...

```
# seq_count 10

seq_count() {

    i=1; while [ $i -le $1 ]; do echo $i; i=`expr $i + 1`; done

}


# simple_seq 1 2 10

simple_seq() {

    i=$1; while [ $i -le $3 ]; do echo $i; i=`expr $i + $2`; done

}


seq_integer() {

    if [ "X$1" = "X-f" ]

    then format="$2"; shift; shift

    else format="%d"

    fi
```

```
case $# in

1) i=1 inc=1 end=$1 ;;

2) i=$1 inc=1 end=$2 ;;

*) i=$1 inc=$2 end=$3 ;;

esac

while [ $i -le $end ]; do

    printf "$format\n" $i;

    i=`expr $i + $inc`;

done

}
```

Edited: by Admin – added code tags.

REPLY LINK

-
- **Philippe Petrinko** June 4, 2010, 7:34 am
-

@Anthony.

Quite right – braces {start..end..step} might not be the best thing in bash.

Nevertheless, I still stick to the old C-like syntax in a for loop, which does accept variable arguments, such as:

```
xstart=10;xend=20;xstep=2
```

```
for (( x = $xstart; x <= $xend; x += $xstep)); do echo $x;done
```

```
10
```

```
12
```

```
14
```

```
16
```

```
18
```

```
20
```

I don't know much of this FOR loop syntax portability, functions you suggest may be the best thing to use for portability concern. (I have to read POSIX reference again :-)

[REPLY LINK](#)

o **TheBonsai** June 4, 2010, 9:57 am

The Bash C-style for loop was taken from KSH93, thus I guess it's at least portable towards Korn and Z.

The seq-function above could use `i=$((i + inc))`, if only POSIX matters. `expr` is obsolete for those things, even in POSIX.

[REPLY LINK](#)

▪ **Philippe Petrinko** June 4, 2010, 10:15 am

Right Bonsai,

(http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_06_04)

But FOR C-style does not seem to be POSIXLY-correct...

Read on-line reference issue 6/2004,

Top is here, <http://www.opengroup.org/onlinepubs/009695399/mindex.html>

and the Shell and Utilities volume (XCU) T.O.C. is here

<http://www.opengroup.org/onlinepubs/009695399/utilities/toc.html>

doc is:

http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap01.html

and FOR command:

http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_09_04_03

REPLY LINK

▪ **Anthony Thyssen** June 6, 2010, 7:18 am

TheBonsai wrote.... "The seq-function above could use `i=$((i + inc))`, if only POSIX matters. `expr` is obsolete for those things, even in POSIX."

I am not certain it is in Posix. It was NOT part of the original Bourne Shell, and on some machines, I deal with Bourne Shell. Not Ksh, Bash, or anything else.

Bourne Shell syntax works everywhere! But as 'expr' is a builtin in more modern shells, then it is not a big loss or slow down.

This is especially important if writing a replacement command, such as for "seq" where you want your "just-paste-it-in" function to work as widely as possible.

I have been shell programming pretty well all the time since 1988, so I know what I am talking about! Believe me.

MacOSX has in this regard been the worse, and a very big backward step in UNIX compatibility. 2 year after it came out, its shell still did not even understand most of the normal 'test' functions. A major pain to write shells scripts that need to also work on this system.

[REPLY LINK](#)

- **TheBonsai** June 6, 2010, 12:35 pm

Yea, the question was if it's POSIX, not if it's 100% portable (which is a difference). The POSIX base more or less is a subset of the Korn features (88, 93), pure Bourne is something "else", I know. Real portability, which means a program can go wherever UNIX went, only in C ;)

[REPLY LINK](#)

- **TheBonsai** June 4, 2010, 5:57 pm

I've read the standard ;-)

[REPLY LINK](#)

- **Phil Goetz** June 17, 2010, 5:38 pm

That {1 .. N} syntax doesn't work with current Linux bash.

```
$for r in {1 .. 15}; do echo $r; done
{1
..
15}
```

[REPLY LINK](#)

- **nixCraft** June 17, 2010, 5:57 pm

Yes, it does work, you need bash version 3.0 or up. Just tested with “GNU bash, version 4.1.5(1)-release (i486-pc-linux-gnu)”. You need to remove white space between 1 and 15, try:

```
for r in {1..15}; do echo $r; done
```

REPLY LINK

-
- **Anthony Thyssen** June 18, 2010, 12:49 am
-

I gave some ‘seq’ alternatives, some simple, some more complex, mostly using shell built-ins only, depending on your needs.

At the very start of the comments “jot” was mentioned as an alternative, though it does not appear to be as wide spread as “seq”. Anyone know if it is on the ‘limited shell support’ MacOSX?

There are also however some other — off the wall — methods of generating a list of number, or just a list for looping ‘N’ times. One of the weirdest ones I came across was using /dev/zero and “dd”!

```
dd 2>/dev/null if=/dev/zero bs=10 count=1 | tr \12 | cat -n | tr -d '\40\11'
```

This gets ‘10’ null characters, converts them to line feeds, uses cat to convert them to numbers, and just to clean up, you can optionally delete the tabs and spaces.

As I said real odd ball.

Like I often say...

There are lots of ways to skin a cat, and what method you use depends on what you want that skin for, and how messy you like the results!

REPLY LINK

- **Philippe Petrinko** June 18, 2010, 7:01 am

Yes,

but the code we see won't work.

I think it needs some syntax enhancement in your first [tr], such as:

```
dd 2>/dev/null if=/dev/zero bs=10 count=1 | tr '00' '12' | cat -n | tr -d
```

```
'\40\11'
```

REPLY LINK

- **Philippe Petrinko** June 18, 2010, 7:02 am

Hell ! missed again – I hate that \$%@ Wordpress text entry

REPLY LINK

- **Paddy** July 20, 2010, 8:54 pm

Hi guys Ive been reading this thread cos i need some advice on a script. I need to rename some (lots) of files in a directory. they are named..

file_name.001.01

file_name.002.01

file_name.003.01... etc

How can I change the names of the files to remove the '.01' at the end of each filename? Im useing Ubuntu Lynx.... Ive been playing with a few examples from this thread, but cant seem to make it work. Any help is appreciated

REPLY LINK

- **nixCraft** July 20, 2010, 9:12 pm

Try [rename command](#):

```
rename 's/\.01$//' *.01
```

If you must use for:

```
for i in *.01; do mv "$i" "${i%.01}"; done
```

HTH

[REPLY LINK](#)

▪ **Dr. Stefan Gruenwald** April 1, 2011, 4:53 am

In general, if you want to add a suffix to your files, do this (.txt in this example):

```
ls
```

```
file1 file2 file3
```

```
for i in *; do mv $i $i.txt; done
```

```
ls
```

```
file1.txt file2.txt file3.txt
```

If you want to take it back off (.txt in this example again)

```
for i in *.txt; do mv $i ${i%.*}; done
```

```
ls
```

```
file1 file2 file3
```

[REPLY LINK](#)

▪ **maxy-millianne** January 24, 2012, 6:18 am

Of course, if you want to worry about files with spaces in (and other things?), put quote around the arguments to mv, as in the gp.

[REPLY LINK](#)

▪ **Anthony Thyssen** January 27, 2012, 7:48 am

Note that is you want to append a string that does not start with a ‘.’ (for example the string “_info.txt”) then you need to delimit the variable name....

```
for i in *; do mv "$i" "${i}_info.txt"; done
```

REPLY LINK

- **Sapia** July 29, 2010, 6:20 am

Hi,

I have two files that contain different columns. Both files have matching one column but row order is different. I want to combine these two files as below.

File 1: file 2

x 2 7 123 r 3 5 9

y 3 -8 124 y 4 6 20

z 4 -2 34 q 3 5 70

q 5 -9 5 z 5 4 10

r 6 1 6 x 50 3 40

I want to combine each row considering the common values in the first column

ex:

x 2 7 123 50 3 40

q 5 -9 5 3 5 70

I would be grateful if you could help me with this problem.

Thank you.

Sapia

REPLY LINK

- **Philippe Petrinko** July 29, 2010, 11:09 am

@Sapia: As Vivek may say,

"Your post is off-topic. I suggest you use our shell scripting forum for question."

Go <http://nixcraft.com/>

REPLY LINK

- **borleand** September 1, 2010, 8:23 pm

bash = GPL

GPL = GNU

GNU = Gnu is Not Unix

UNIX logo in your web page = ???

REPLY LINK

-
- o **nixCraft** September 2, 2010, 11:45 am

Does it really matters?

Bash runs on both certified UNIX and UNIX like (*BSD & various Linux distros) operating systems.

HTH

REPLY LINK

-
- **Philippe Petrinko** September 2, 2010, 12:10 pm

@Vivek

Quite right – who cares ? Nobody.

I just hope that those guys [borleand] would have better working and helping for FOSS community instead of posting those high-quality (;-P) comments.

Thanks for such a great web site, Vivek.

REPLY LINK

-
- **Anthony Thyssen** September 3, 2010, 2:02 am

Of bigger concern between UNIX, GNU, GPL, Linux, Solaris, MacOSX, or whatever else, is what extra support programs are also available.

I use to use 'seq' all the time in shell loops. I don't any more because 'seq' is not available on MacOSX. In fact a lot of simple and what I would have though universal support programs are not available on MacOSX.

REPLY LINK

- **naveen** September 16, 2010, 4:53 am

how do i run 100 iteration using bash shell script.. i want to know how long will it take to execute one command(start and end time). I want to keep track which iteration is currently running. i want to log each iteration. I have one automated script i need to run and log it.

```
for i in 1 2 3
do
command1
done
```

But i want to know how long it takes to complete one iteration. and writes a log... help me please

REPLY LINK

- **Philippe Petrinko** September 16, 2010, 7:15 am

Hi naveen,

1) You should at least read thoroughly this topic. Iteration can be done with this syntax

```
for x in {start..end}
```

Look above for explanation.

- 2) You could use [date] command and output it to a file
- as first command of the iteration (echoing something to mark start)
 - as last command of the iteration (echoing something to mark end)

3) Log

What kind of log? To know what? What for?

REPLY LINK

- **naveen** September 16, 2010, 3:03 pm

i want to write a output in text file(.txt)(log). I want to have a report saying that test ran for 100 iteration. My question i have one automated script that run 100 test file. i want to know how long it takes to complete the one iteration. And i want to keep a copy of the test result in .txt file.

i know this is to log for one iteration.....

command > log.txt

But when you are running iteration for 100. How wil you log it??. this is my question.

Just one automated script (command) but there will be 100 test results for 100 iteration.

How wil you log all this???

REPLY LINK

- **Philippe Petrinko** September 16, 2010, 3:20 pm

You really seem to be a beginner in shell – the best way to start would be to study Vivek's Wiki first.

Go http://bash.cyberciti.biz/guide/Main_Page

To answer you iteration question:

1) First write the appropriate [for] loop that calls you commands and show us your code. All the information you need is located above in this page. If you cannot write this loop, you'd better learn [bash] on Vivek's Wiki and come back when you can write a [for] loop with 100 iteration. Good luck! :)

REPLY LINK

- **Anthony Thyssen** November 19, 2010, 3:22 am

The problem with this is that csv files can contain quoted strings. which makes just comma separation useless.. For example

LoginN,ssn,ffn

"ab1,pp1","ab1","pp1"

"bb1,oo1","bb1","oo1"

"cc1,qq1","cc1","qq1"

REPLY LINK

-
- o [nixCraft](#) November 19, 2010, 7:58 am
-

In that case use " as delimiter. This can be done with sed or cut or any other shell built-in or utility:

```
echo '"ab1,pp1","ab1","pp1"' | cut -d '"' -f2
```

```
echo '"ab1,pp1","ab1","pp1"' | cut -d '"' -f4
```

```
echo '"ab1,pp1","ab1","pp1"' | cut -d '"' -f6
```

for loop can be used:

```
for i in {2..6..2}; do echo '"ab1,pp1","ab1","pp1"' | cut -d '"' -f${i}; done
```

You can skip for and use sed too.

HTH

REPLY LINK

-
- **Anthony Thyssen** November 22, 2010, 1:16 am
-

It isn't quite that simple. But then any CSV that is more complex is getting a bit beyond **simple** shell parsing.

REPLY LINK

- **Anthony Thyssen** November 22, 2010, 1:17 am

Nifty solution though, especially without access to a multi-character field separator.

REPLY LINK

- **Philippe Petrinko** November 22, 2010, 7:58 am

Here is another way to do it,

- in a generic way, that is without having to know number of internal fields,
- without IFS manipulation
- without external function (only builtin):

Let's say your CSV file contains 3 fields per record (per line):

"aaa bbb ccc","ddd eee fff","ggg hhh iii","jjj kkk lll"

"mmm nnn ooo","ppp qq q rr","sss ttt uuu","vvv www xxx"

"yyy zzz 111","222 333 444","555 666 777","888 999 000"

To break it in a one-liner, try:

```
while read; do record=${REPLY}; echo ${record}|while read -d ","; do echo
```

```
${REPLY}; done; done<data
```

The same code in a script is:

```
#!/bin/bash
```

```
while read
```

```
do
```

```
record=${REPLY}
```

```
echo ${record}|while read -d ,
```

```
do
```

```
echo ${REPLY}
```

```
done
```

```
done<data
```

REPLY LINK

- **Philippe Petrinko** November 22, 2010, 8:23 am

And if you want to get rid of double-quotes, use:

one-liner code:

```
while read; do record=${REPLY}; echo ${record}|while read -d ", "; do
```

```
field="${REPLY#\"}"; field="${field%\"}"; echo ${field}; done; done<data
```

script code, added of some text to better see record and field breakdown:

```
#!/bin/bash
```

```
while read
```

```
do
```

```
echo "New record"
```

```
record=${REPLY}
```

```
echo ${record}|while read -d ,
```

```
do
```

```
field="${REPLY#\}"
```

```
field="${field%\}"
```

```
echo "Field is :${field}:"
```

```
done
```

```
done<data
```

Does it work with your data?

— PP

REPLY LINK

- **Philippe Petrinko** November 22, 2010, 9:01 am

Of course, all the above code was assuming that your CSV file is named “data”.

If you want to use anyname with the script, replace:

```
done<data
```

With:

```
done
```

And then use your script file (named for instance “myScript”) with standard input redirection:

```
myScript < anyFileNameYouWant
```

Enjoy!

REPLY LINK

- **Philippe Petrinko** November 22, 2010, 11:28 am

well no there is a bug, last field of each record is not read – it needs a workout and may be IFS modification ! After all that’s what it was built for... :O)

REPLY LINK

- **Anthony Thyssen** November 22, 2010, 11:31 pm

Another bug is the inner loop is a pipeline, so you can’t assign variables for use later in the script. but you can use ‘<<<’ to break the pipeline and avoid the echo.

But this does not help when you have commas within the quotes! Which is why you needed quotes in the first place.

In any case It is a little off topic. Perhaps a new thread for reading CVS files in shell should be created.

REPLY LINK

- **Philippe Petrinko** November 24, 2010, 6:29 pm

Anthony,

Would you try this one-liner script on your CSV file?

This one-liner assumes that CSV file named [data] has __every__ field double-quoted.

```
while read; do r="${REPLY#\}";echo "${r//\"/,\"/\}";while read -d \";do echo
```

```
"Field is :${REPLY}:";done;done<data
```

Here is the same code, but for a script file, not a one-liner tweak.

```
#!/bin/bash
```

```
# script csv01.sh
```

```
#
```

```
# 1) Usage
```

```
# This script reads from standard input
```

```
# any CSV with double-quoted data fields
```

```
# and breaks down each field on standard output
```

```
#
```

```
# 2) Within each record (line), _every_ field MUST:
```

```
# - Be surrounded by double quotes,
```

```
# - and be separated from preceeding field by a comma
```

```
# (not the first field of course, no comma before the first field)
```

```
#
```

```
while read
```

```
do
```

```
echo "New record" # this is not mandatory-just for explanation
```

```
#
```

```
#
```

```
# store REPLY and remove opening double quote
```

```
record="${REPLY#\}"
```

```
#
```

```
#
```

```
# replace every ", " by a single double quote
```

```
record=${record//\",\"/\\"}
```

```
#
```

```
#
```

```
echo ${record}|while read -d \"
```

```
do
```

```
# store REPLY into variable "field"
```

```
field="${REPLY}"
```

```
#
```

```
#
```

```
echo "Field is :${field}:" # just for explanation
```

```
done
```

```
done
```

This script named here [cvs01.sh] must be used so:

```
cvs01.sh < my-cvs-file-with-doublequotes
```

REPLY LINK

- **Philippe Petrinko** November 24, 2010, 6:35 pm

@Anthony,

By the way, using [REPLY] in the outer loop _and_ the inner loop is not a bug.
As long as you know what you do, this is not problem, you just have to store [REPLY]
value conveniently, as this script shows.

:-P

REPLY LINK

- **James Wilkinson** December 14, 2010, 10:03 am

Thanks for writing this article Vivek – it is very useful. In particular, I didn't know about
bash's built-in 'help' command and was getting frustrated with the lack of detail in 'man
for'

REPLY LINK

- **Sammeta** December 15, 2010, 4:48 pm

Hi i need help in sorting some of the raw data actually on the unix machine.

The raw data is some thing like this:

```
TOP;0004796;T0698;3,8;2,2;1,6;19384;1560;28;0;304;12;0;httpd
```

```
TOP;0004796;T0699;0,2;0,1;0,1;12984;1524;28;0;304;1;0;httpd
```


TOP;0004806;T0145;40,3;38,8;1,5;427904;32648;8;0;1000;62;0;java

TOP;0004806;T0146;30,6;29,7;0,8;433152;37684;8;0;1036;29;0;java

TOP;0004806;T0147;16,8;16,3;0,6;433808;39152;8;0;1048;11;0;java

TOP;0004806;T0148;14,2;13,4;0,7;434132;40300;8;0;1056;11;0;java

TOP;0004806;T0149;13,1;12,5;0,6;434196;41644;8;0;1080;7;0;java

TOP;0004806;T0150;18,1;17,3;0,8;434276;42556;8;0;1092;18;0;java

TOP;0004806;T0151;8,3;7,8;0,5;434316;42624;8;0;1096;1;0;java

TOP;0004806;T0152;4,4;4,1;0,4;434328;42916;8;0;1096;9;0;java

TOP;0004806;T0153;4,7;4,3;0,4;434352;43204;8;0;1104;9;0;java

TOP;0004806;T0154;3,1;2,8;0,3;434404;43260;8;0;1104;1;0;java

...

VM;T1439;-1;8;-1;-1;-

1;41;0;0;0;7346;0;23;0;2312;2312;0;0;0;0;0;0;0;21;21;0;12;-2633;0;0;0;356;-
3;0;0;0;0;0

VM;T1440;-1;6;-1;-1;-1;8893;0;0;0;10808;0;12178;-36;865179;-

34821;0;0;8895;0;0;0;0;11758;11758;0;-21;2613;0;0;0;-752;3;0;2;0;0;7

ZZZZ;T0001;00:00:02;18-NOV-2010

ZZZZ;T0002;00:01:02;18-NOV-2010

ZZZZ;T0003;00:02:02;18-NOV-2010

....

ZZZZ;T0140;02:19:04;18-NOV-2010

ZZZZ;T0141;02:20:04;18-NOV-2010

ZZZZ;T0142;02:21:04;18-NOV-2010

ZZZZ;T0143;02:22:04;18-NOV-2010

ZZZZ;T0144;02:23:04;18-NOV-2010

ZZZZ;T0145;02:24:04;18-NOV-2010

ZZZZ;T0146;02:25:04;18-NOV-2010

ZZZZ;T0147;02:26:04;18-NOV-2010

So now it has to check for the time stamp T0145 is the timestamp in the below code :

```
TOP;0004806;T0145;40,3;38,8;1,5;427904;32648;8;0;1000;62;0;java< ?pre>
```

now this has to be mapped with the same time stamp in ZZZZ.....Which is some thing like this from above.

ZZZZ;T0145;02:24:04;18-NOV-2010

so finally the output should look like this for every occurrence the T value:

TOP;0004806;T0145;40,3;38,8;1,5;427904;32648;8;0;1000;62;0;java
ZZZZ;T0145;02:24:04;18-NOV-2010,

REPLY LINK

-
- o **nixCraft** December 15, 2010, 6:43 pm
-

See sort command man page or our sort tutorial related FAQs:

[Linux / UNIX Shell: Sort Date](#)

[Linux / UNIX Shell: Sort IP Address](#)

REPLY LINK

-
- **Sammeta** December 16, 2010, 3:52 pm
-

Hi Vivek,

Thanks for your reply,

but this is just not only sorting but recording on of the value in the above code and then match this with other lines in the code .then display both together.

REPLY LINK

-
- **Philippe Petrinko** December 16, 2010, 6:10 pm
-

@Sammeta,

Just trying to help, and not to being rude or anything:

We are not willing to do your work,

anyone would help people who really try to help themselves first,

which you may have been trying already, I suppose.

So, would you either submit a first version of your code,

or at least a main Algorithm you could think of?

You may want to read first [awk] or [join] unix utilities man pages that you could find anywhere.

–P

REPLY LINK

- **mohan** January 5, 2011, 12:29 am

what does FILES="@@" do? Please explain i am new to unix.

REPLY LINK

- **nixCraft** January 5, 2011, 7:03 am

See [http://bash.cyberciti.biz/guide/\\$@](http://bash.cyberciti.biz/guide/$@) and http://bash.cyberciti.biz/guide/Bash_special_parameters

REPLY LINK

- **John** January 5, 2011, 1:24 am

One good reason to use seq instead of the {start..end..increment} idiom:

```
#!/bin/bash
START=1
END=10
INCR=2
echo — use seq —
for x in $(seq $START $INCR $END)
do
echo $x
done
echo — bash idiom —
for x in {$START..$END..$INCR}
do
```

```
echo $x
done
```

— use seq —

```
1
3
5
7
9
```

— bash idiom —

```
{1..10..2}
REPLY LINK
```

-
- **TheBonsai** January 5, 2011, 7:09 am
-

Wrong conclusion. This is not a reason for the seq idiom, it's a reason to use arithmetically driven (C-like) for loops.

```
START=1
```

```
END=10
```

```
INCR=2
```

```
for ((x = START; x <= END; x += INCR)); do
```

```
    echo $x
```

```
done
```

http://wiki.bash-hackers.org/syntax/ccmd/c_for

REPLY LINK

- **Philippe Petrinko** February 3, 2011, 9:09 am

@Bonsai:

The example you gave is weird regarding shell variable usage: It works, but I thought it should not work!

Within the `for (())` instruction, you omitted “\$” sign to allow variable expansion, but it works! That looks very strange to me.

I wrote in an example previously (look above):

```
xstart=1;xend=10;xstep=2
```

```
for (( x = $xstart; x <= $xend; x += $xstep)); do echo $x;done
```

In your example, you wrote `[for(())]` without “\$” :

```
for ((x = START; x <= END; x += INCR))
```

I am astonished that `for(())` works both with and without “\$” for variable expansion!

```
xstart=1;xend=10;xstep=2
```

```
for (( x = xstart; x <= xend; x += xstep)); do echo $x;done
```

Does anyone know why?

–PP

REPLY LINK

- **nixCraft** February 3, 2011, 12:15 pm

I think it is ksh93 compatibility feature; so “START / END / INCR” will work with no “\$”.
See ksh93 man page.

REPLY LINK

- **Philippe Petrinko** February 3, 2011, 1:56 pm

Sorry, I cannot find any evidence of such syntax on ksh93 man page – and I use Bash shell.

On what URI + what chapter do you think there is an explanation of such behavior?

REPLY LINK

- **TheBonsai** February 3, 2011, 5:01 pm

It's not KSH (or at least not KSH-unique). It's how arithmetic environments (of any kind) in Bash work.

Also it's related to what POSIX specifies for the environment inside arithmetic expansion (the only a. environment POSIX knows):

If the shell variable x contains a value that forms a valid integer constant, then the arithmetic expansions “\${(x)}” and “\${(\$x)}” shall return the same value.

REPLY LINK

- **nixCraft** February 3, 2011, 5:29 pm

I think it was mentioned in TLDP.org's advanced bash guide.

REPLY LINK

- **Philippe Petrinko** February 3, 2011, 9:05 pm

Quite right Vivek.

<http://tldp.org/LDP/abs/html/arithexp.html>

REPLY LINK

- **toto** February 16, 2011, 5:01 am

hi,

good tutorial.

thank you

REPLY LINK

- **VÃ°ir Valberg GuÃ°mundsson** March 8, 2011, 3:22 am

I might be going out on a limb due to a bad case of TL;DR, but I noticed the seq warning.

Correct me if I'm wrong but using for instance

```
for i in $(seq -w 1 1 20); do echo $i; done
```

is in my opinion quite an useful way of using seq in bash, at least when you want leading zeros before 1-9 and not from 10<

Or is there a better way of doing this all "bashy"? ;)

-vÃÃ°ir

[REPLY LINK](#)

-
- o **TheBonsai** March 8, 2011, 6:26 am

```
for ((i=1; i<=20; i++)); do printf "%02d\n" "$i"; done
```

[REPLY LINK](#)

- o **nixCraft** March 8, 2011, 6:37 am

+1 for printf due to portability, but you can use bashy .. syntax too

```
for i in {01..20}; do echo "$i"; done
```

[REPLY LINK](#)

-
- **TheBonsai** March 8, 2011, 6:48 am

Well, it isn't portable per se, it makes it portable to pre-4 Bash versions.

I think a more or less "portable" (in terms of POSIX, at least) code would be

```
i=0
```



```
while [ "$(i >= 20)" -eq 0 ]; do
```

```
    printf "%02d\n" "$i"
```

```
    i=$((i+1))
```

```
done
```

REPLY LINK

- **ankit gulati** March 12, 2011, 2:44 pm

Hi

Can anybody help for this.

I have two text file viz gem1.txt and gem2.txt

EX. Content of gem1.txt

activerecord (2.3.5, 2.2.2)

activerecord-oracle_enhanced-adapter (1.1.9)

activerecord-sqlserver-adapter (2.3.4)

activeresource (2.3.5, 2.2.2)

Now i have to put all the data of these two .txt file on gem.csv. column wise.

REPLY LINK

- **Philippe Petrinko** March 12, 2011, 10:58 pm

@ankit gulati

This is not a place :-P

for such questions, go and see Forum. <http://nixcraft.com/>

Anyway an answer is [paste] command. :-)

<http://linux.die.net/man/1/paste>

REPLY LINK

- **PiCarre** March 20, 2012, 9:15 am

Simply use the “paste” command.

REPLY LINK

- **Rupender** March 23, 2011, 3:48 pm

Hi

can some one help in ceating a loop

the problem is

i have 3 folder and name of the folder changes.

i want a loop which enter these folder one by one echo the files inside the folder

REPLY LINK

- **Philippe Petrinko** March 23, 2011, 7:27 pm

Hi

following this topic,

you may try to write something down first,

and then we may help you ... if you help yourself first.

Read and use this topic,

and the following material: http://bash.cyberciti.biz/guide/Main_Page

However, such request would be best posted into the forum <http://nixcraft.com/>

Anyway, a simple solution would be to use [ls] command with appropriate arguments.

REPLY LINK

- **Philip Ratzsch** April 20, 2011, 5:53 am

I didn't see this in the article or any of the comments so I thought I'd share. While this is a contrived example, I find that nesting two groups can help squeeze a two-liner (once for each range) into a one-liner:

```
for num in {{1..10},{15..20}};do echo $num;done
```

Great reference article!

[REPLY LINK](#)

-
- **Philippe Petrinko** April 20, 2011, 8:23 am

@Philip

Nice thing to think of, using brace nesting, thanks for sharing.

[REPLY LINK](#)

- **David** May 4, 2011, 10:10 am

Hi guys.

I was wondering whether it is possible to run a for loop through rsh (now before you start protesting that I use rsh instead of ssh; I know that it is extremely insecure.

Thanks.) like this:?

```
rsh -l dev 192.168.x.x "for file in /PackSun/distills/*; do if [[ ${file} =~ "3\.6\.76" ]]; echo $file; fi; done"
```

Because when I type this it doesn't work, \$file is always the same file and it comes from the pwd on the local machine, not on the remote.

Thanks in advance.

D

[REPLY LINK](#)

-
- **David** May 4, 2011, 10:11 am

damn, sorry, didn't close the code tag:

```
rsh -l dev 192.168.x.x "for file in /PackSun/distills/*; do if [[ ${file} =~ "3\.6\.76" ]]; echo $file; fi; done"
```

REPLY LINK

- **Anthony Thyssen** May 5, 2011, 2:57 am

First — you should not use rsh. It is an old and insecure network protocol. ssh is its replacement.

As for your question YES it is posible, Wrap it in a 'shell' command.

```
rsh -l dev 192.168.x.x 'sh -c \"for file in /PackSun/distills/*; do if [[ ${file} =~ \"3\\.6\\.76\" ]];  
echo $file; fi; done\"'
```

Watch the quoting as you want to use single quotes for both the outside and inside commands. If you use " quotes you will need to escape the \$ characters instead! And that may in turn lead to escapes of escapes (not pretty).

I have actually done this to execute a very very large shell script on a remote server. However I don't recommend it for things beyond what you have. If need be copy (update) a shell script on the remote server (scp, rsync, etc) then execute that. It works better, you do not have constant quote handling problems, and not command line length limits.

I do this for a incremental home backup program (written for my own use, using rsync hardlinks backup directories), that can backup to a remote account. You are free to download and look at the copy/execute remote script that the main script performs for its sub-scripts (like backup cycle rolls).

REPLY LINK

- **Philippe Petrinko** May 5, 2011, 1:10 pm

Anthony,

You forgot to include URI for you script, your > a < tag is empty.

REPLY LINK

-
- **Anthony Thyssen** May 5, 2011, 3:01 am
-

PS: the reason for the 'sh -c ...' is because you may not have control of the login shell of the remote account. If you did not have it and the remote account used tcsh or zsh, you will have problems.

PPS; you are missing a then. I tested it with this ssh version (a shell-builtin only 'ls' of the remote account).

```
ssh remote_machine 'sh -c \"for file in *; do echo $file; done\"'
```

[REPLY LINK](#)

-
- **Sanya** May 6, 2011, 6:02 am
-

Hello

I am using bash V4+ and this loop works fine:

```
for i in {1..10}; do echo $i; done
```

But when I put variables in the loop

```
max=10; for i in {1..$max}; do echo $i; done
```

I see one line as output:

```
{1..10}
```

Could anybody explain me what's wrong ?

Cheers. Sanya

REPLY LINK

- **Philippe Petrinko** May 6, 2011, 10:13 am

Hello Sanya,

That would be because brace expansion does not support variables. I have to check this.

Anyway, Keep It Short and Simple: (KISS) here is a simple solution I already gave above:

```
xstart=1;xend=10;xstep=1
```

```
for (( x = $xstart; x <= $xend; x += $xstep )); do echo $x;done
```

Actually, POSIX compliance allows to forget \$ in for quotes, as said before, you could also write:

```
xstart=1;xend=10;xstep=1
```

```
for (( x = xstart; x <= xend; x += xstep )); do echo $x;done
```

REPLY LINK

- **Philippe Petrinko** May 6, 2011, 10:48 am

Sanya,

Actually brace expansion happens __before__ \$ parameter expansion, so you cannot use it this way.

Nevertheless, you could overcome this this way:

```
max=10; for i in $(eval echo {1..$max}); do echo $i; done
```

REPLY LINK

- **Sanya** May 6, 2011, 11:42 am

Hello, Philippe

Thanks for your suggestions

You basically confirmed my findings, that bash constructions are not as simple as zsh

ones.

But since I don't care about POSIX compliance, and want to keep my scripts "readable" for less experienced people, I would prefer to stick to zsh where my simple for-loop works

Cheers, Sanya

REPLY LINK

- **Philippe Petrinko** May 6, 2011, 12:07 pm

Sanya,

First, you got it wrong: solutions I gave are not related to POSIX, I just pointed out that POSIX allows not to use \$ in for (()), which is just a little bit more readable – sort of.

Second, why do you see this less readable than your [zsh] [for loop]?

```
for (( x = start; x <= end; x += step )) do
```

```
echo "Loop number ${x}"
```

```
done
```

It is clear that it is a loop, loop increments and limits are clear.

IMNSHO, if anyone cannot read this right, he should not be allowed to code. :-D

BFN

REPLY LINK

- **Anthony Thyssen** May 8, 2011, 11:30 pm

If you are going to do... `$(eval echo {1..$max});`

You may as well use "seq" or one of the many other forms.

See all the other comments on doing for loops.

REPLY LINK

-
- **Tom P** May 19, 2011, 12:16 pm
-

I am trying to use the variable I set in the for line on to set another variable with a different extension. Couldn't get this to work and couldn't find it anywhere on the web... Can someone help.

Example:

```
FILE_TOKEN=`cat /tmp/All_Tokens.txt`
```

```
for token in $FILE_TOKEN
```

```
do
```

```
A1_$token=`grep $A1_token /file/path/file.txt | cut -d ":" -f2`
```

my goal is to take the values from the ALL Tokens file and set a new variable with A1_ in front of it... This tells me that A1_ is not a command...

Edited by admin: Added pre tag

[REPLY LINK](#)

-
- **atef fawzy** May 23, 2011, 10:27 am
-

could you please help me in write a bash script do the following:

- 1- read an input file contains IP Port
- 2- check the connectivity for every IP and its Port via telnet
- 3- if the telnet not connected send email to alert me.

thanks in advance

atef fawzy

[REPLY LINK](#)

- **Philippe Petrinko** May 23, 2011, 11:34 am

No problem.

Would you first send me 1000USD payment?

Thanks in advance. ;-)

REPLY LINK

- **atef fawzy** May 23, 2011, 12:37 pm

dear Philippe Petrinko

thank you a lot for you value site

the below is my script and i don't know what is the wrong?can you help me

```
#!/bin/sh
```

```
for ip in $(cat iplist);
```

```
do
```

```
# check for open ports #
```

```
connTest=`echo " " | telnet $ip`
```

```
if [ "`echo $ip | awk '{ print $3 }' = "Connected" ]
```

```
then
```

```
echo "$ip is up"
```

```
else
```

```
#echo "$connTest port is down"
```

```
echo $ip is down
```

```
fi
```

```
done
```

REPLY LINK

- **Philippe Petrinko** May 23, 2011, 12:25 pm

Lucky You! Atef fawzy, Vivek has already written necessary training material for you:

Read http://bash.cyberciti.biz/guide/Main_Page

REPLY LINK

- **atef fawzy** May 23, 2011, 12:41 pm

please tell me what is the wrong?

```
#!/bin/sh
for ip in $(cat iplist);
do
# check for open ports #
connTest=`echo " " | telnet $ip`

if [ "`echo $ip | awk '{ print $3 }'`" = "Connected" ]
then
echo "$ip is up"

else

echo $ip is down

fi
done
REPLY LINK
```

- **atef fawzy** May 23, 2011, 12:33 pm

dear Philippe Petrinko
thank you a lot for your value site
the below is my script and i don't know what is the wrong? can you hep me please

```
#!/bin/sh
for ip in $(cat iplist);
do
# check for open ports #
connTest=`echo " " | telnet $ip`

#if [ "`echo $connTest | awk '{ print $3 }'`" = "Connected" ]
if [ "`echo $ip | awk '{ print $3 }'`" = "Connected" ]
then
```

```
#echo "$connTest port is up"
```

```
echo "$ip is up"
```

```
else
```

```
#echo "$connTest port is down"
```

```
echo "$ip is down"
```

```
fi
```

```
done
```

REPLY LINK

- **Philippe Petrinko** May 23, 2011, 1:39 pm

Your request is off-topic.

Your [for] loop works fine, so this is not a point to be discussed here,

Your [if] test is buggy – not to be discussed here but there <http://nixcraft.com/>

REPLY LINK

- **atef fawzy** May 24, 2011, 10:20 am

please send me the full URL to post the error i got

REPLY LINK

- **Philippe Petrinko** May 24, 2011, 11:14 am

You can go where I already offered you => <http://nixcraft.com/>

You can go there and figure out what topic is appropriate to your needs.

REPLY LINK

- **loop how** May 26, 2011, 12:49 am

```
#!/bin/sh
```

How can I do this using infinite loops?

```
for disk in $(ls /dev/disk/by-id/ | grep -v part | grep scsi-35000I);  
  
do  
  
echo $disk && \  
  
dd if=/dev/zero of=/dev/disk/by-id/$disk bs=8M oflag=direct count=250 && \  
  
dd if=/dev/disk/by-id/$disk of=/dev/null bs=8M count=250;  
  
done
```

REPLY LINK

-
- **Chu-Siang** July 21, 2011, 9:07 pm

thanks for you example, It`s help me very much.

REPLY LINK

-
- **Faizanul Islam** July 26, 2011, 6:04 am

hey,

Thanks a lot. Example are in the way they should be...some of the examples are very new to me. It has increased by knowledge.

keep posting these things and let us increase our knowledge.

REPLY LINK

-
- **brian** August 5, 2011, 7:58 am

Hi,

I have to read a file line by line and do the iteration and apply the condition parameter for each input . Please share your ideas

REPLY LINK

- **Philippe Petrinko** August 5, 2011, 9:56 am

@Brian

You could google “awk” to proceed a file line by line, of use example in Vivek’s blog.

<http://bash.cyberciti.biz/file-management/read-a-file-line-by-line/>

<http://www.cyberciti.biz/faq/unix-howto-read-line-by-line-from-file/>

REPLY LINK

- **Sam McAllister** August 10, 2011, 11:10 am

Hi, I’m a beginner and I was writing a very simple script :

```
#!/bin/sh
clear
echo "Enter username:"
read username
if [ "$username" = "newbay" ]
then
echo "Username correct,"
else
echo "Incorrect username, try again:"
fi
```

```
echo "Now enter password:"
read password
if [ "$password" = "welcome" ]
then
echo "You are now logged in."
```

else

echo "Sorry, incorrect password. Please try again."

fi

and I was wondering how to loop the incorrect username try again part ? Can anyone help me, if they understand my awful script ..

REPLY LINK

- o **Anthony Thyssen** August 11, 2011, 7:25 am

Start a new thread!

REPLY LINK

- **Sam McAllister** August 11, 2011, 7:47 am

Haha ok thanks :)

REPLY LINK

- o **Philippe Petrinko** August 11, 2011, 8:48 am

You could start learning shell scripting:

About [for] loops: http://bash.cyberciti.biz/guide/For_loop

(from that excellent http://bash.cyberciti.biz/guide/Main_Page)

Enjoy!

REPLY LINK

- **ITtuition.com** August 11, 2011, 1:30 pm

Spot on. Included all examples that form a good base. Thanks for sharing.

REPLY LINK

- **Nagesh** September 14, 2011, 10:21 am

This post is 3 yrs old but still RockS.\m/

REPLY LINK

- **ptrac3** October 24, 2011, 5:43 pm

Very handfult tutorial! :) But i've got a problem on my Ubuntu linux box, a strange problem.. Executing the example code

```
#!/bin/bash
```

```
for i in {1..5}
```

```
do
```

```
echo "Welcome $i times"
```

```
done
```

i get as output Welcome {1..5} times. So the bash doesn't understand the range {1..5}...Have any idea? My bash version is GNU bash, version 4.2.8(1)-release (x86_64-pc-linux-gnu). Thank u :)

REPLY LINK

- **Philippe Petrinko** October 25, 2011, 9:01 am

This code has to be written into a text file, which must be made executable.

Did you do that?

REPLY LINK

- **nixCraft** October 25, 2011, 11:19 am

GNU/Bash v4.2.8 does supports {1..5} syntax. What is the output of the following commands?

```
ls -l /bin/bash
```

```
/bin/bash --version
```

REPLY LINK

-
- **Sonia McMath** November 18, 2011, 6:52 am
-

Great weblog right here! Additionally your web site quite a bit up very fast! What host are you the usage of? Can I get your associate hyperlink for your host? I want my website loaded up as fast as yours lol

[REPLY LINK](#)

- **Hannah** December 5, 2011, 11:21 pm
-

Hi all... I have a question about using for loops. Basically, I have a file containing a list of protein ID numbers. what I want to do is create a for loop that can go into this file, grab each protein ID number, and then search the NCBI database to get the fasta sequence of each protein... and then create another file containing all of the fasta sequences....

In general, my problem is that I can't figure out how to get the protein ID numbers from the output file (ex. \$1 in file1) into the for loop script.

This is what I have so far:

```
for gi in file1
do
fastacmd -d /data/nr -s gi
done
```

but I need to specify that the gi (protein ID number) is the first column (\$1) of file1.

Does this make sense?

[REPLY LINK](#)

-
- **Akshit** December 7, 2011, 12:36 pm
-

Hi...

Thanks for the article, it is helpful really.

I want to know one thing.

Let's say there is file with content

1

2

3

4

5

how can I use for loop and print these numbers?

If you are understanding what I am trying to tell. I want to use the contents of these files and stored in the variable.

[REPLY LINK](#)

- **andreea** January 5, 2012, 4:21 pm
-

Hello!

i am beginner and i have to make a bash script in which i have to show the number of lines that were added by the analized commits(in git).can you help me?Thanks.

[REPLY LINK](#)

- **Philippe Petrinko** January 5, 2012, 5:33 pm
-

Dear andreea

1) I quote Vivek: "You are free to use our shell scripting forum for questions."

<http://nixcraft.com/shell-scripting/>

2) You can learn for free shell scripting with the infamous Vivek's collaborative Linux Shell Scripting Tutorial

http://bash.cyberciti.biz/guide/Main_Page

Enjoy!

— Philippe

[REPLY LINK](#)

- **Philippe Petrinko** January 5, 2012, 6:23 pm

“infamous” was a private joke-understatement

Sorry, no offense intended!

Vivek website roxxxxs !

— Philippe

[REPLY LINK](#)

- **andreea** January 6, 2012, 6:33 pm

Thanks for your quick answer.

I have another problem.i have to use this script by using git.can you explain me or give some advices how can i use it?thank you very much.

[REPLY LINK](#)

- **andreea** January 8, 2012, 8:54 pm

I have this code that shows the first column which represents the number of insertions of a commit and calculate the sum of all numbers of each line.

```
if [ $# -eq 2 ]; then
```

```
if [ $2 = "added_lines" ]; then
```

```
tmpfile=$(mktemp)
```

```
sum=0
```

```
git log --pretty=tformat: --numstat | tr -s "\n" | cut -f 1 > $tmpfile
```

```
for i in $(cat $tmpfile); do
```

```
sum=$((sum + $i))
```

```
done
```

```
echo "$sum"
```

```
rm $tmpfile
```

But with ‘cut -f 1’ it takes too much time to calculate and to show the result.Can you help me saying how can i do this with ‘awk’ or ‘gawk’?

Please help me.Thanks

[REPLY LINK](#)

- **Ram** January 10, 2012, 6:57 pm

Hi Vivek,

i am trying to run the below code as ram.sh in server as \$ sh ./ram.sh
code:

```
#!/bin/bash
for i in 1 2 4
do
echo "Welcome $i times"
done
```

It fails saying

ram.sh: line 3: syntax error near unexpected token `do

ram.sh: line 3: `do

is that something i need to check which version is the sh and bash used in the server.

how to check that.

it is a pretty simple code and it is not workign.

i am running this script in Linux server 64 bit server.

[REPLY LINK](#)

o **the dsc** October 23, 2012, 2:54 am

What's the easiest way to have a loop from N to N, when the numbers are dictated by variables?

Something like what one could/would expect from "for i in {\${a}..\${b}}", but something that actually works of course.

What I do is:

```
a=0 ; b=5 ; until ((a==b)) ; do echo $a ; a=$((a+1)) ; done
```

But it seems kind of dumb, somewhat like echoing a huge set of spaces instead of “clear”, to clear the screen. But perhaps it’s the only way to do it. And it works anyway, so perhaps it’s good enough and worthy as an addition rather than a question.

Funnily enough you can make an alphabetic countdown with “for i in {a..z}”

REPLY LINK

- **Philippe Petrinko** January 27, 2012, 10:07 am

Hi Ram,

Since you are a real beginner, you would really take great benefit self-training on this course free of charge by vivek & contributors:

http://bash.cyberciti.biz/guide/Main_Page

–P

REPLY LINK

- **shahzad** March 10, 2012, 9:04 am

Hi,

I have this code in NetBeens:

```
#include
#include
using namespace std;
int main() {
int a,i,j;
cout << "inter your number of row " <>a;
for (i=0; i<=a; i++)
{
for(j=0; j<=i; j++)
{
```

```
cout << "*";  
}  
cout << '\n';  
}  
return 0;  
}
```

but I want to run it in ubuntu so at first I made a " nano file.sh" after that I wrote this code in it:

```
#!/bin/sh  
echo "Enter your number of rows"  
Read a  
for((i=0; i<=a; i++))  
do  
for((j=0; j<=i; j++))  
do  
echo " "*  
done  
echo "  
done  
exit 0
```

after that " chmod +x file.sh", but my code didn't run, my code in ubuntu has problem, I don't know how can I solve it, Thanks a lot if you answer my question,

Regards

[REPLY LINK](#)

-
- **regan** May 5, 2012, 2:26 pm
-

hi, am having problem to write a program using For statement. Please help if you know something about using For statement

[REPLY LINK](#)

-
- **Excellent_Test** August 23, 2012, 11:57 am
-

This is what I was looking for. TLDP advanced guide is also good one.

[REPLY LINK](#)

-
- **Philippe Petrinko** November 12, 2012, 7:38 pm
-

Vivek,

It seems that Wordpress was hungry, it has eaten your text ;-)

[for] code sample is broken after sentence: "A representative three-expression example in bash as follows"

—P

[REPLY LINK](#)

-
- **Youshi Patel** February 20, 2013, 3:14 pm
-

nice write up.

[REPLY LINK](#)

-
- **m.vinothini** March 7, 2013, 6:17 am
-

very useful

[REPLY LINK](#)

-
- **OSuKaRu** June 6, 2013, 8:54 pm
-

i try the infinite variable, but it tells me that:

Syntax error: Bad for loop variable

:<

REPLY LINK

-
- **K J Ramana Rao** June 20, 2013, 2:26 pm
-

Hello,

Nice post.

Please help me in below code. we are getting syntax error. Please resolve the syntax issue.

```
if [ $1 ] ; then
```

```
    LIB=$1/lib
```

```
    if -d ${_LIB} ; then
```

```
        for jar in (cd ${_LIB}; ls *.jar)
```

```
        do
```

```
            EXISTS=echo ${CP} | grep "${jar}"
```

```
            if [ "${_EXISTS}" != "" ]; then
```

```
                logger "WARN: Classpath will contain multiple files named  
${jar}"
```

```
            else
```

```
CP=${CP}${CP_DELIM}${LIB}/${jar}

fi

done

fi

cd $1

fi

}
```

Error:

```
syntax error near unexpected token `('
```

```
line 15: `for jar in (cd ${_LIB}; ls *.jar)'
```

Thanks

Ramana

[REPLY LINK](#)

o **Anthony Thyssen** June 24, 2013, 12:43 am

You are a 'help troll'.. You posted in the wrong section.

you would have been better of starting a new topic.

[REPLY LINK](#)

• **foolar** June 21, 2013, 1:20 pm

try with


```
if [ $1 ] ; then
LIB=${1}/lib
if -d ${_LIB} ; then
for jar in $(cd ${_LIB}; ls *.jar)
do
EXISTS=$(echo ${CP} | grep “/${jar}”)
if [ “${_EXISTS}” != “” ]; then
logger “WARN: Classpath will contain multiple files named ${jar}”
else
CP=${CP}${CP_DELIM}${LIB}/${jar}
fi
done
fi
cd $1
fi
REPLY LINK
```

-
- **abr** June 27, 2013, 12:51 am
-

hi if I use it with a input variable like:

echo “enter the numbers of repetitions :”

read variable

the variable represent the number example 15 times to repeat !!!???

how could it be ???

REPLY LINK

-
- **dennis bernaerts** July 4, 2013, 1:19 pm
-

nice and clear thx a lot.

What I am dreaming of is this clear kind of webpage in which each box has a tab in which you can select any language.... I'm sure it exists already but where ???
and I mean it needs to be clear and simple !!

[REPLY LINK](#)

-
- **Tring Tring** September 24, 2013, 9:27 am

Thank you so Much!

[REPLY LINK](#)

-
- **sana** October 2, 2013, 10:40 am

hi i have a problem i want to write a shell script for siesta.

i want to make diffrent directories and want to change lattice constants n then run it with siesta n want to chck total energy

cn u tell m how cn i do this

[REPLY LINK](#)

-
- **Philippe Petrinko** October 2, 2013, 12:32 pm

no w cnt

sry w dnt 1drstnd w U sd

[REPLY LINK](#)

-
- **sana** October 3, 2013, 6:26 am

I want to write a shell script

for diffrent values of lattice constants .

If i do manually i first make a directorie i.e for 3.80 then i copy *.psf and fdf file there open the fdf file and change lattice constant to 3.80 and the execute file with siesta and note the total energy.

n again i repeat the same process for let say 3.90,4.10.4.20 etc.

now i want to write a code in shell so that i dnt need to make directory every time and

change lattice constant.

I want to use looping for this purpose.....but how??

anybody have the little code for this?or anything?

[REPLY LINK](#)

- **sana** October 3, 2013, 6:31 am

```
#!/bin/bash
# changing lattice constants
for 3.80 3.90 in $inp3.70.fdf
do
echo 3.80 3.90
mkdir
mv inp3.70.fdf inp*.fdf.old
mv *.psf *.psf.old
sed 's/inp*.fdf/&, lattice constant/' inp*.fdf
done
# Now comes the commands to be executed
~/code/sanabin/siesta 3.70.out &
~

as i write this but not working..... ../
```

[REPLY LINK](#)

- **Philippe Petrinko** October 3, 2013, 7:48 am

Well, it seems that you do not understand the basics of shell scripting, even a simple [mkdir] command.

Anyway, you are very lucky, because this web site and Internet offer you free training material.

The good news is : all you have to do is learn and work, using for instance these links.

http://bash.cyberciti.biz/guide/Main_Page

<http://bash.cyberciti.biz/script/for-loop/>

and of course this page <http://www.cyberciti.biz/faq/bash-for-loop/>

REPLY LINK

-
- **matthias** October 10, 2013, 5:19 am
-

nice you used “c++” in “A representative three-expression example in bash” but that is a programming language.

REPLY LINK

-
- **Philippe Petrinko** October 10, 2013, 9:18 am
-

@matthias :

What’s your point?

Is it a humorous play of words on “C++” language and increment of c variable in this script?

In that case it would have been more convenient to append a smiling smiley, because otherwise, there is no coding issue in “A representative three-expression example in bash” .

:-/ wondering...

— Philippe

REPLY LINK

-
- **ravindra** December 3, 2013, 3:00 pm
-

Hi.. check it

```
for (( c=1; c<=5; c++ ))
```

```
do
```

```
    echo "Welcome $c times"
```

```
done
```

why came this error in ubuntu please tell me

reply please

./for3: 1: ./for3: Syntax error: Bad for loop variable

[REPLY LINK](#)

-
- o **Vamsi** December 1, 2014, 11:18 am

Hi,

It works for BASH only.

Check you have given the right interpreter at the top of the script.

You can give it by placing

`#!/bin/bash` at the top.

–Vamsi

[REPLY LINK](#)

- **RH User** January 3, 2014, 9:12 pm

Thanks. Solved my coding issue.

[REPLY LINK](#)

-
- **dee** January 17, 2014, 8:27 pm
-

Can anyone advise how to write a for loop statement to run commands from a .sh when an event comes up like users uploading new files to the server.

My issue is that I am using a soft link to mirror an external disk drive in the .www/ and the soft link never updates when a new content is added to the drive within a session.

REPLY LINK

-
- **Dai** June 18, 2014, 2:16 pm

Hi. Nice overview of for loops. I have a question, however:

Using the old “seq” command, one could perform zero padding via “seq -f “%05g” 1 100”. How would this work with the {1..100} (or other) syntax?

REPLY LINK

-
- **Philippe Petrinko** June 19, 2014, 7:00 am

Hi Dai,

simple:

1)

```
for x in {0001..10} ; do echo “padding :$x:”; done
```

2)

Actually it works specifying padding on first argument (here 0001). No need to specify padding on second argument, but it will either work.

```
for x in {0001..0010} ; do echo “padding :$x:”; done
```

3) but beware: you can specify different padding on both arguments, but only the `_longest_` will be used !

so this will use 6 digits padding, not 3 !

```
for x in {001..000010} ; do echo "padding :$x:"; done
```

ok ?

Vivek, would improve this topic on for loop adding this information on padding? It seems to be useful and at least informative and relevant to this topic.

— Philippe

REPLY LINK

- **Dai** June 19, 2014, 7:33 am

Marvelous. Cheers!

REPLY LINK

- **dronkit** April 1, 2015, 8:55 pm

You cannot completely abandon seq for the new bash syntax. Apparently, all variables in bash are integers. I tried using the new syntax with negative and float numbers with disastrous results. Seq, in turn, does it wonderfully.

REPLY LINK

- **Johnny Rosenberg** August 6, 2015, 3:07 pm

Ooops, forget it, someone else said the same thing. Didn't see that, I'm not sure whyâ€¦

REPLY LINK

- **wt** September 24, 2015, 1:06 pm

1 line loop, for example curling:

for i in {1..500}; do curl "https://localhost:8040"; done;

[REPLY LINK](#)

-
- **onews** November 3, 2015, 2:05 pm

How do we do something like:

for file in /etc/* /bin/*

[REPLY LINK](#)

-
- **Carlos Luna** November 14, 2015, 3:32 am

Useful information! thank you!

Can somebody explain me what this loop “for i in circles[0,:].” means?

[REPLY LINK](#)

-
- **Sasha** December 22, 2015, 2:41 pm

Hi:

I use seq when I need variables in the loop limits, because as far as I remember

for y in {\$x1..\$x2}... is not allowed.

Cheers

[REPLY LINK](#)

-
- **nezabudka** January 3, 2016, 4:39 pm
-

Sasha

```
for i in $(seq 1 2 $max)
```

```
for i in $(eval echo {1..$max..5})
```

REPLY LINK

-
- **Emma Edwards** March 18, 2016, 2:22 pm

Hi i would like to know how to loop numbers using terminal but i cant seem to find how to so it would be great if one of you can help me

REPLY LINK

-
- **Emma Edwards** March 18, 2016, 2:23 pm

sorry i ment i cant seem to find....

REPLY LINK

-
- **Golla praveen** June 14, 2016, 1:59 pm

```
#!/bin/bash
```

```
for ((i=1;i<=100;i++));
```

```
do
```

```
echo $i
```

while excute the above program the below errors comes please check revert

REPLY LINK

-
- **mito** July 1, 2016, 8:42 am
-

hi

i would like to know how i write a shell script to search a 100 user home directory by certain file

can you help me?

thx

[REPLY LINK](#)

-
- **mito** July 1, 2016, 8:45 am

output is username und the file

[REPLY LINK](#)

-
- **Chad** November 23, 2016, 3:41 pm

Hey I hope you can help me here. I'm stuck.

down vote

favorite

Trying to bind 20 x /24 subnets in Ubuntu 14 Server but I'm stuck here. These are diversified /24 subnets.

I tried the below method via script called addips.sh

```
for i in $(seq 3 254); do echo "auto em1:$i
iface em1:$i inet static
address xxx.xxx.16.$i
netmask 255.255.255.0
" >> virthosts; done
```

But, I noticed that this happened now:

Each /24 subnet it outputted started off at #1 again instead of 255, 256, 257, etc all the way through consecutively for all 20 x /24 subnets

```
auto em1:1
```

```
auto em1:254
```

Then again...

```
auto em1:1
```

...

How can I properly change this so it does reset at #1? em1 is the NIC Ethernet port (primary) and only port used too.

I also want to make sure these are permanent, they stick after a server reboot.

REPLY LINK

-
- **Jignesh** December 19, 2016, 7:36 am
-

Hello There,

I want to make one script which shall change multiple file name as per below example

- 1) if file name contains *abc* then file name should be change to *xyz*
- 2) if file name contains *def* then file name should be change to *iop*
- 3) if file name contains *(any file name except above name) then file name should be change to *qwe*

Kindly help me to make the script as mentioned above requirement

Thanks!!

i am running my application in linux by providing inputs as command line. My input field contain an argument which contains ";"(semicolon) internally.(For example:123;434;5464). This will be parsed using UTF8String encode and send. But when i am using like this, in initial itself i am getting,

```
bash: 434: command not found
bash: 5464: command not found
```

And when i capture traffic the output contains only 123 instead 123;434;5464 But if i give without semicolon (Ex:123:434:5464),not getting any problem output coming properly as 123:434:5464

Point me how to give command line input by using semicolon as to come output. Is there any particular syntax to use while doing with semicolon. I am running like below

```
./runASR.sh -ip 10.78.242.4 -port 3868 -sce 10.78.241.206 -id 85;167838865;1385433280
```

where -id field contain that value with issue.

;
; is treated an *end of command* character. So 123;456;5464 to bash is in fact 3 commands. To pass such meta-characters escape it with escape character \.

```
./command 123\;456\;5464
```

Or Just quote it with single quote ([double quote evaluates the inner string](#)) (Thanks Triplee, I forgot to mention this)

```
./command '123;456;5464'
```