Examples of shell script

1. **echo $SHELL** ---- to see the working shell in SSH
2. ***#!/bin/sh --This line should always be the first line in your script***

***# A simple script***

***who am i***

***date***

***pwd***

1. **Static values for the list after “in” keyword**

In the following example, the list of values (Mon, Tue, Wed, Thu and Fri) are directly given after the keyword “in” in the bash for loop.

$ cat for1.sh

i=1

for day in Mon Tue Wed Thu Fri

do

echo "Weekday $((i++)) : $day"

done

$ ./for1.sh

Weekday 1 : Mon

Weekday 2 : Tue

Weekday 3 : Wed

Weekday 4 : Thu

Weekday 5 : Fri

Note: The list of values should not be separated by comma (Mon, Tue, Wed, Thu, Fri). The comma will be treated as part of the value. i.e Instead of “Mon”, it will use “Mon,”

Try the below given program

1. cat for1-wrong1.sh

i=1

for day in Mon, Tue, Wed, Thu, Fri

do

echo "Weekday $((i++)) : $day"

done

$ ./for1-wrong1.sh

Output: Weekday 1 : Mon,

Weekday 2 : Tue,

Weekday 3 : Wed,

Weekday 4 : Thu,

Weekday 5 : Fri

1. **Caution:** The list of values should not be enclosed in a double quote. (“Mon Tue Wed Thu Fri”). If you enclose in double quote, it will be treated as a single value (instead of 5 different values),

$ cat for1-wrong2.sh

i=1

for day in "Mon Tue Wed Thu Fri"

do

echo "Weekday $((i++)) : $day"

done

$ ./for1-wrong2.sh

Output: Weekday 1 : Mon Tue Wed Thu Fri

### Variable for the list after “in” keyword

Instead of providing the values directly in the for loop, you can store the values in a variable, and use the variable in the for loop after the “in” keyword, as shown in the following example.

$ cat for2.sh

i=1

weekdays="Mon Tue Wed Thu Fri"

for day in $weekdays

do

echo "Weekday $((i++)) : $day"

done

$ ./for2.sh

Weekday 1 : Mon

Weekday 2 : Tue

Weekday 3 : Wed

Weekday 4 : Thu

Weekday 5 : Fri

**Caution**: As a best practice, you should always quote the bash variables when you are referring it. There are few exceptions to this best practice rule. This is one of them. If you double quote the variable in this for loop, the list of values will be treated as single value.

### Don’t specify the list; get it from the positional parameters

If you don’t specify the keyword “in” followed by any list of values in the bash for loop, it will use the positional parameters (i.e the arguments that are passed to the shell script).

$ cat for3.sh

i=1

for day

do

echo "Weekday $((i++)) : $day"

done

$ ./for3.sh Mon Tue Wed Thu Fri

Output : Weekday 1 : Mon

Weekday 2 : Tue

Weekday 3 : Wed

Weekday 4 : Thu

Weekday 5 : Fri

***8.* Caution:** Please be careful if you use this method. You should not include the keyword “in” in the for loop. If you leave the keyword “in” without any values, it will not use the positional parameter as shown below. It will not go inside the loop. i.e for loop will never get executed as shown in the example below.

$ cat for3-wrong.sh

i=1

for day in

do

echo "Weekday $((i++)) : $day"

done

$ ./for3-wrong.sh Mon Tue Wed Thu Fri

### Unix command output as list values after “in” keyword

You can use the output of any UNIX / Linux command as list of values to the for loop by enclosing the command in back-ticks ` ` as shown below.

$ cat for4.sh

i=1

for username in `awk -F: '{print $1}' /etc/passwd`

do

echo "Username $((i++)) : $username"

done

$ ./for4.sh

Output : Username 1 : ramesh

Username 2 : john

Username 3 : preeti

Username 4 : jason

### *10 .* Loop through files and directories in a for loop

To loop through files and directories under a specific directory, just cd to that directory, and give \* in the for loop as shown below.

The following example will loop through all the files and directories under your home directory.

$ cat for5.sh

i=1

cd ~

for item in \*

do

echo "Item $((i++)) : $item"

done

$ ./for5.sh

Item 1 : positional-parameters.sh

Item 2 : backup.sh

Item 3 : emp-report.awk

Item 4 : item-list.sed

Item 5 : employee.db

Item 8 : storage

Item 9 : downloads

Usage of \* in the bash for loop is similar to the file globbing that we use in the linux command line when we use ls command (and other commands).

### *11.* Break out of the for loop

You can break out of a for loop using ‘break’ command as shown below.

$ cat for6.sh

i=1

for day in Mon Tue Wed Thu Fri

do

echo "Weekday $((i++)) : $day"

if [ $i -eq 3 ]; then

break;

fi

done

$ ./for6.sh

Weekday 1 : Mon

Weekday 2 : Tue

### *. 12.* Continue from the top of the for loop

Under certain conditions, you can ignore the rest of the commands in the for loop, and continue the loop from the top again (for the next value in the list), using the continue command as shown below.

The following example adds “(WEEKEND)” to Sat and Sun, and “(weekday)” to rest of the days.

$ cat for7.sh

i=1

for day in Mon Tue Wed Thu Fri Sat Sun

do

echo -n "Day $((i++)) : $day"

if [ $i -eq 7 -o $i -eq 8 ]; then

echo " (WEEKEND)"

continue;

fi

echo " (weekday)"

done

$ ./for7.sh

Day 1 : Mon (weekday)

Day 2 : Tue (weekday)

Day 3 : Wed (weekday)

Day 4 : Thu (weekday)

Day 5 : Fri (weekday)

Day 6 : Sat (WEEKEND)

Day 7 : Sun (WEEKEND)

### *12 .* Bash for loop using C program syntax

This example uses the 2nd method of bash for loop, which is similar to the C for loop syntax. The following example generates 5 random number using the bash C-style for loop.

$ cat for8.sh

for (( i=1; i <= 5; i++ ))

do

echo "Random number $i: $RANDOM"

done

$ ./for8.sh

Random number 1: 23320

Random number 2: 5070

Random number 3: 15202

Random number 4: 23861

Random number 5: 23435

### *13.* Range of numbers with increments after “in” keyword

The following example loops through 5 times using the values 1 through 10, with an increment of 2. i.e It starts with 1, and keeps incrementing by 2, until it reaches 10.

$ cat for12.sh

for num in {1..10..2}

do

echo "Number: $num"

done

$ ./for12.sh

Number: 1

Number: 3

Number: 5

Number: 7

Number: 9

### *14.* 11. Range of numbers after “in” keyword

You can loop through using range of numbers in the for loop “in” using brace expansion.

The following example loops through 10 times using the values 1 through 10.

$ cat for11.sh

for num in {1..10}

do

echo "Number: $num"

done

$ ./for11.sh

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5