

WEEK 3

ZEESHAN AHMED

20K-1741

BSE-4B

TASKS: COURSERA

MATH



```
zeeshan@ubuntu: ~  
bc(1) General Commands Manual bc(1)  
NAME  
bc - An arbitrary precision calculator language  
SYNTAX  
bc [ -hlwsqv ] [long-options] [ file ... ]  
DESCRIPTION  
bc is a language that supports arbitrary precision numbers with interactive execution of statements. There are some similarities in the syntax to the C programming language. A standard math library is available by command line option. If requested, the math library is defined before processing any files. bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read. (If a file contains a command to halt the processor, bc will never read from the standard input.)  
  
This version of bc contains several extensions beyond traditional bc implementations and the POSIX draft standard. Command line options can cause these extensions to print a warning or to be rejected. This document describes the language accepted by this processor. Extensions will be identified as such.  
OPTIONS  
-h, --help  
    Print the usage and exit.  
-i, --interactive  
    Force interactive mode.  
-l, --mathlib  
    Define the standard math library.  
-w, --warn  
    Give warnings for extensions to POSIX bc.  
-s, --standard  
    Process exactly the POSIX bc language.  
-q, --quiet  
    Do not print the normal GNU bc welcome.  
-v, --version  
    Print the version number and copyright and quit.  
NUMBERS  
The most basic element in bc is the number. Numbers are arbitrary precision numbers. This precision is both in the integer part and the fractional part. All numbers are represented internally in decimal and all computation is done in decimal. (This version truncates results from divide and multiply operations.) There are two attributes  
Manual page bc(1) line 1/732 6% (press h for help or q to quit)
```

```
math.txt  
1 10+3  
2 2*6  
3 89-14  
4 106/6  
5 23/12
```

```
zeeshan@ubuntu:~$ man bc  
zeeshan@ubuntu:~$ echo "13+89" | bc  
102  
zeeshan@ubuntu:~$ echo "7-3" | bc  
4  
zeeshan@ubuntu:~$ echo "7/2" | bc  
3  
zeeshan@ubuntu:~$ echo "3*11" | bc  
33  
zeeshan@ubuntu:~$ gedit math.txt  
zeeshan@ubuntu:~$ cat math.txt | bc  
13  
12  
75  
16  
1  
zeeshan@ubuntu:~$
```

VARIABLES

```
zeeshan@ubuntu: ~  
zeeshan@ubuntu:~$ var_1=5  
zeeshan@ubuntu:~$ var_2=4  
zeeshan@ubuntu:~$ echo $var_1+$var_2  
5+4  
zeeshan@ubuntu:~$ let var_3=var_1+var_2  
zeeshan@ubuntu:~$ echo $var_3  
9  
zeeshan@ubuntu:~$ var_4=5+9  
zeeshan@ubuntu:~$ var_5=6+4  
zeeshan@ubuntu:~$ let var_6=var_4+var_5  
zeeshan@ubuntu:~$ echo $var_6  
24  
zeeshan@ubuntu:~$
```

```
zeeshan@ubuntu: ~  
zeeshan@ubuntu:~$ str_1="Islamabad"  
zeeshan@ubuntu:~$ str_2=" is the capital of Pakistan"  
zeeshan@ubuntu:~$ echo "$str_1$str_2"  
Islamabad is the capital of Pakistan  
zeeshan@ubuntu:~$ str3=$(cat << EOF  
> $str_1  
> $str_2  
> EOF  
> )  
zeeshan@ubuntu:~$ echo "$str_3"  
  
zeeshan@ubuntu:~$ echo "$str3"  
Islamabad  
 is the capital of Pakistan  
zeeshan@ubuntu:~$
```

```
zeeshan@ubuntu:~$ nano ex.sh  
zeeshan@ubuntu:~$ chmod +x ex.sh  
zeeshan@ubuntu:~$ ./ex.sh 2 3 4  
  
The value of First Variable =  
2  
The value of Second Variable = 2  
3  
The value of Third Variable = 3  
Result=3*  
zeeshan@ubuntu:~$ gedit ex.sh  
zeeshan@ubuntu:~$ nano ex.sh  
zeeshan@ubuntu:~$ ./ex.sh  
2  
The value of First Variable = 2  
3  
The value of Second Variable = 3  
4  
The value of Third Variable = 4  
Total Arguments: 0  
Result = 0  
zeeshan@ubuntu:~$ ./ex.sh 1 2 3  
2  
The value of First Variable = 2  
3  
The value of Second Variable = 3  
4  
The value of Third Variable = 4  
Total Arguments: 3  
Result = 6  
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 ex.sh  
read arg_1  
echo "The value of First Variable = $arg_1"  
read arg_2  
echo "The value of Second Variable = $arg_2"  
read arg_3  
echo "The value of Third Variable = $arg_3"  
echo "Total Arguments: $#"  
let Result=$arg_1  
echo "Result = $Result"
```

USER INPUT

```
zeeshan@ubuntu:~$ nano ex2.sh
zeeshan@ubuntu:~$ chmod +x ex2.sh
zeeshan@ubuntu:~$ ./ex2.sh
Enter a Noun:
Zeeshan
Enter an Adjective:
amazing
Enter a verb:
enjoying
Zeeshan is enjoying his amazing life
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 ex2.sh
echo "Enter a Noun: $noun"
read noun
echo "Enter an Adjective: $adj"
read adj
echo "Enter a verb: $verb"
read verb
echo "$noun is $verb his $adj life"
```

LOGIC AND IF/ELSE

```
GNU nano 4.8 ex3.sh
echo "Enter a string: $str_1"
read str_1
if [[ $str_1 -ge A && $str_1 -le Z ]];then
echo "how proper"
else
echo "not upper case"
fi
```

```
zeeshan@ubuntu:~$ nano ex3.sh
zeeshan@ubuntu:~$ chmod +x ex3.sh
zeeshan@ubuntu:~$ ./ex3.sh
Enter a string:
Hello
how proper
zeeshan@ubuntu:~$
```

```
zeeshan@ubuntu:~$ nano ex4.sh
zeeshan@ubuntu:~$ chmod +x ex4.sh
zeeshan@ubuntu:~$ ./ex4.sh
Enter a number:
4
Even
zeeshan@ubuntu:~$ ./ex4.sh
Enter a number:
5
Odd
zeeshan@ubuntu:~$
```

```
GNU nano 4.8                                ex4.sh
echo "Enter a number: $var"
read var
if [[ $var%2 -eq 0 ]];then
echo "Even"
else
echo "Odd"
fi
```

```
zeeshan@ubuntu: ~
zeeshan@ubuntu:~$ nano ex5.sh
zeeshan@ubuntu:~$ chmod +x ex5.sh
zeeshan@ubuntu:~$ ./ex5.sh
Enter values of arg_1 and arg_2: 3 6
9
zeeshan@ubuntu:~$ ./ex5.sh
Enter values of arg_1 and arg_2: r t
0
zeeshan@ubuntu:~$
```

```
GNU nano 4.8                                ex5.sh
read -p "Enter values of arg_1 and arg_2: " x y
if [[ $x -ge 0 && $x -le 9 ]];then
if [[ $y -ge 0 && $y -le 9 ]];then
let z=$x+$y
echo "$z"
else
echo "$x $y"
fi
fi
```

```
zeeshan@ubuntu:~$ nano ex6.sh
zeeshan@ubuntu:~$ chmod +x ex6.sh
zeeshan@ubuntu:~$ ./ex6.sh
Sunday
Not Friday
zeeshan@ubuntu:~$
```

```
GNU nano 4.8                                ex6.sh
date +%A
var=$(date)
var='date'
if [[ $var -eq "Friday" ]];then
echo "Not Friday"
else
echo "Thanks Moses its Friday"
fi
```

ARRAYS

```
GNU nano 4.8                                ex7.sh
array=(zeeshan mustafa ayesha uner muneer ghufan hassan areeba ehtishan)
echo ${array[50]}
```

```
zeeshan@ubuntu: ~$ nano ex7.sh
zeeshan@ubuntu:~$ chmod +x ex7.sh
zeeshan@ubuntu:~$ ./ex7.sh 2
ayesha
zeeshan@ubuntu:~$
```

```
GNU nano 4.8                                ex8.sh
arr1=(a b c d e f g h i j)
arr2=(1 2 3 4 5)
let var=$((arr1[0])+${arr2[0]})
echo $var
```

```
zeeshan@ubuntu:~$ nano ex8.sh
zeeshan@ubuntu:~$ chmod +x ex8.sh
zeeshan@ubuntu:~$ ./ex8.sh
15
zeeshan@ubuntu:~$
```

BRACES

```
zeeshan@ubuntu:~$ nano {0..99} ex9.txt

Use "fg" to return to nano.

[1]+  Stopped                  nano {0..99} ex9.txt
zeeshan@ubuntu:~$

zeeshan@ubuntu:~$ gedit 50ex9.txt
zeeshan@ubuntu:~$
```

LOOPS

```
GNU nano 4.8                                ex10.sh
count=3
while [[ $count -gt 0 ]]
do
let count=$count-1
for i in {1..10}
do
if [[ $i -lt 3 ]] || [[ $i -gt 8 ]];then
echo $i
fi
done
done

```

```
zeeshan@ubuntu:~$ nano ex10.sh
zeeshan@ubuntu:~$ ./ex10.sh
1
2
9
10
1
2
9
10
1
2
9
10
zeeshan@ubuntu:~$ nano ex10.sh
zeeshan@ubuntu:~$
```

```
YES(1)                                     User Commands                               YES(1)

NAME
    yes - output a string repeatedly until killed

SYNOPSIS
    yes [STRING]...
    yes OPTION

DESCRIPTION
    Repeatedly output a line with all specified STRING(s), or 'y'.

    --help display this help and exit
    --version
        output version information and exit

AUTHOR
    Written by David MacKenzie.

REPORTING BUGS
    GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
    Report yes translation bugs to <https://translationproject.org/team/>

COPYRIGHT
    Copyright © 2018 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or
    later <https://gnu.org/licenses/gpl.html>.
    This is free software: you are free to change and redistribute it. There is NO WAR-
    RANTY, to the extent permitted by law.

SEE ALSO
    Full documentation at: <https://www.gnu.org/software/coreutils/yes>
    or available locally via: info '(coreutils) yes invocation'

GNU coreutils 8.30                               September 2019                               YES(1)
Manual page yes(1) line 1/35 (END) (press h for help or q to quit)
```

FUNCTIONS

```
zeeshan@ubuntu:~$ nano plier.sh
zeeshan@ubuntu:~$ source plier.sh
zeeshan@ubuntu:~$ plier 2 3 4
24
zeeshan@ubuntu:~$ plier 45 6 78
21060
zeeshan@ubuntu:~$ plier 1 23 67
1541
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 plier.sh
function plier
{
    product=1
    for element in $@
    do
        let product=product*$element
    done
    echo $product
}
```

```
zeeshan@ubuntu:~$ nano isiteven.sh
zeeshan@ubuntu:~$ source isiteven.sh
zeeshan@ubuntu:~$ isiteven
Enter an integer: clear
6
1
zeeshan@ubuntu:~$ isiteven
Enter an integer: 6
7
0
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 isiteven.sh
function isiteven {
    echo "Enter an integer: $num"
    read num
    if [[ $num%2 -eq 0 ]];then
        echo "1"
    else
        echo "0"
    fi
}
```

```
zeeshan@ubuntu: ~  
zeeshan@ubuntu:~$ nano nevens.sh  
zeeshan@ubuntu:~$ source nevens.sh  
zeeshan@ubuntu:~$ neven 2 4 5 6 3 7 8  
Total Evens: 4  
zeeshan@ubuntu:~$ nano nevens.sh  
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 nevens.sh  
function neven {  
  cc=0  
  for i in $@  
  do  
    if [[ $i%2 -eq 0 ]];then  
      let cc=cc+1  
    fi  
  done  
  echo "Total Evens: $cc"  
}
```

```
zeeshan@ubuntu:~$ nano fib.sh  
zeeshan@ubuntu:~$ source fib.sh  
zeeshan@ubuntu:~$ fib 4  
0  
1  
1  
2  
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 fib.sh  
function fib {  
  count=1  
  num=$@  
  a=0  
  b=1  
  for (( i=0; i<num; i++ ))  
  do  
    echo "$a"  
    func=$((a + b))  
    a=$b  
    b=$func  
  done  
}
```


WRITING PROGRAMS

```
zeeshan@ubuntu:~$ echo '#!/usr/bin/env bash' > short
zeeshan@ubuntu:~$ echo 'echo "A small program"' >> short
zeeshan@ubuntu:~$ cat short
#!/usr/bin/env bash
echo "A small program"
zeeshan@ubuntu:~$
```

```
zeeshan@ubuntu: ~/Dir1
zeeshan@ubuntu:~/Dir1$ cd ..
zeeshan@ubuntu:~$
zeeshan@ubuntu:~$ mv ex10.sh Dir1
zeeshan@ubuntu:~$ cd Dir1
zeeshan@ubuntu:~/Dir1$ ls -l
total 8
-rwxrwxr-x 1 zeeshan zeeshan 137 Feb 20 08:31 ex10.sh
-rw-rw-r-- 1 zeeshan zeeshan  0 Feb 20 09:02 ex11.sh
-rw-rw-r-- 1 zeeshan zeeshan 16 Feb 20 09:05 ex12.sh
zeeshan@ubuntu:~/Dir1$ ./ex10.sh
1
2
9
10
1
2
9
10
1
2
9
10
zeeshan@ubuntu:~/Dir1$
```

```
zeeshan@ubuntu:~$ nano ex13.sh
zeeshan@ubuntu:~$ chmod +x ex13.sh
zeeshan@ubuntu:~$ ./ex13.sh
1
2
3
4
5
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 ex13.sh
for i in $(seq 5)
do
echo "$i"
done
```

```
zeeshan@ubuntu: ~  
zeeshan@ubuntu:~$ nano extreme.sh  
zeeshan@ubuntu:~$ chmod +x extreme.sh  
zeeshan@ubuntu:~$ ./extreme.sh  
No Arguments  
Max Number is:  
Lowest number is:  
zeeshan@ubuntu:~$ ./extreme.sh 2 4 7  
Max Number is: 7  
Lowest number is: 2  
zeeshan@ubuntu:~$
```

```
GNU nano 4.8 extreme.sh  
if [[ "$#" -eq 0 ]];then  
echo "No Arguments"  
fi  
max=$1  
for arg in "$@"  
do  
if [[ "$arg" -gt "$max" ]];then  
max=$arg  
fi  
done  
echo "Max Number is: $max"  
for arg in "$@"  
do  
if [[ "$arg" -lt "$max" ]];then  
max=$arg  
fi  
done  
echo "Lowest number is: $max"
```