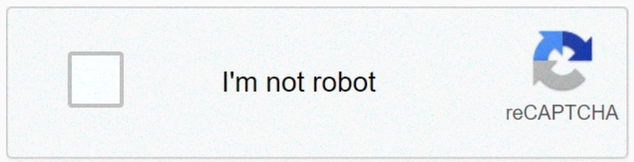

Continue

Awk printf format examples

So far we have used the print AWK and Print function to display data on standard output. But printing is much more powerful than what we've seen before. This function is borrowed in the C language and is very useful while producing output format. Below is the syntax of the print statement − the syntax print fmt, expr-list of the fm above syntax is a string of format specifications and constants. expr-list is a list of arguments corresponding to specified format. Escape Sequences Similar to any string, formats can contain forbidden escape sequences. Discuss below are the escape sequences supported by AWK − This new example line print Hello with World in separate lines using newline character − Example [jerry] $awK 'BEGIN { print HelloWorld }' On execute the following code, you get the following result − Output Hello World Horizontal This Tab uses table horizontally to display different fields − Example [Jerry] $AwK ' START { print Sr No\TSub\tSub\tMarks}' On execute code above , you get that result − Sr No Sub Marks Vertical This Tab uses table vertically after each fill − Example [jerry] $ Start { print Sr No \ vName \ vSub \ vMarks }' On executing this code, you get the following result − From Sr Name Sub Marks Backspace This example printed a backspace after each field except the last one. He deleted the last number from the first three fields. For example, Field 1 is displayed as Field, because the last character is deleted with backspace. However, the last 4 field is displayed as it is, as we didn't have a \ b after Field 4. Example [Jerry]$awk'START { print Field 1\BField 2\bField 3\bField 4}' On execute this code, you get the following result − Out Field Field 4 Carriage returns in this example, after printing each field, we make a carriage return and print next value on top of the current printed value. That is to say, in the final output, you can see only Field 4, as it was the last thing to be printed on top of all the fields before them. Example [Jerry]$awk'BEGIN { print Field 1\RField 2\RField 3\rField 4}' On executing this code, you get the following result − From Field 4 Form Feed this example to form feed after printing each field. Example [jerry]$awk'BEGIN { print Sr No\fName\fSub\fMarks}' On executing this code, you get the following result − From Sr Specific FormatTing Mark Name as in C-language, AWK has also specified format. The AWK version of the print statement accepts the following conversion format specification − c It prints a single character. If the argument used for %c is numeric, it is treated as a character and printed. Otherwise, the argument is supposed to be a string, with the only first character in which string is printed. Example [jerry]$awk'START { print ASCII value 65 = character 'c,65}' From executing this code, you get the following result − ASCII value = character A %d and I it print only the relative integer part of a decimal number. Example [jerry]$awk'START { print percentage = %d,80.66}' On executing this code, you get the following result −Exit Percentage = 80%e and %E to print a floating point number in the form[-] ddddde[+-] dddd. Example [jerry]$awk'BEGIN { print rate = %E,80.66}' On executing this code, you get the following result − From Percentage = 8.066000e +01 The Format E is used instead of e. Example [jerry]$awk'START { print rate = e, 80.66 }' On executing this code, you get the following result − From Percentage = 8.06600E +01 f He printed a floating point number in the form [-] ddd.dddd. Example [jerry]$awk'START { print rate = f, 80.66 }' On executing this code, you get the following result − Output Percentage = 80.660000g and %G% Users e or f %conversion, whatever is shorter, and zero non-significant suppressed. Example [jerry]$awk'BEGIN { print percentage = g, 80.66 }' Output on executing this code, you get the following result − percentage = 80.66% Format of G % used instead of %e. Example [jerry]$awk'BEGIN { print percentage = G, 80.66 }' On executing this code, you get the following result − Output Percentage = 80.66%o He printed out an octal number that doesn't sign. Example [jerry]$awk'BEGIN { print Octal Representation in decimal number 10 = o, 10 }' On executing this code, you get the following result − From Octal representation of decimal number 10 = 12% U print a non decimal number. Example [jerry]$awk'BEGIN { print Non-signing 10 = %u,10}' On executing this code, you get the following result − From non signing 10 = 10% s print it a character string. Example [jerry]$awk'BEGIN { print Name = s, Sherlock Holmes }' On execute this code, you get the following result − Output = Sherlock Holmes %x and %X It print an outstanding non-signed number. The X format uses uppercase letters instead of lowercase. Example [jerry]$awk'START { print hexadecimal representation of decimal number 15 = %x,15 }' On executing this code, You get the following result − Hexadecimal representation of decimal number 15 = f Now let use X and observe the result − Example [jerry]$awk'BEGIN { print Hexadecimal representation of decimal number 15 = X, 15 }' On executing this code, you get the following result − Outstanding Hexadecimal representation of decimal number 15 = F% It prints a single character and does not no argument is converted. Example [jerry]$awk'START { print percentage = %d%,80.66}' On executing this code, you get the following result −Exit Percentage = 80% Parameter if you want and %we can use parameters if you want − The field Width is loaded in the width. By default, the field is loaded with spaces but when 0 flags are used, it is off with zero. Example [jerry]$awk 'START { num1 = 10; num2 = 20; print num1 = 10dNum2 = %10d, num1, num2 }' On execute code, you get the following result − From Num1 = 10 Num2 = 20 Leading Zero A leading zero as a flag, indicating that the output should be pasted with zero instead of space. Please note that this flag only has an effect when the field is wider than the value to be printed. The following example describes the following example − Example [jerry]$awk'Start { num1 = -10; num2=20;print Num1=%05dNum2=%055d,num1,num2}' On executing this code, you get the following result − From Num1 = -0010 Num2 = 00020 Left Justification The expression should be left-justified in its field. When the input-string is less than the number of specified characters, and you want it to be left justified, i.e., by adding space to the right, use a minus symbol (-) immediately after the %and before the number. In this example, output to the AWK command is pipe in the chat order to display the END OF LINE ($) character. Example [jerry]$awk 'START { num = 10; print num = %-5d, numer }' | Chat-vte On executing this code, you get the following result − From Num = 10 $ Prefix Sign It always prefix numeric value with a sign, even if the value is positive. Example [jerry]$awk'START { num1 = -10; num2 = 20; print num1 = + dNum2 = %d, num1, num2 }' On executing this code, you get the following result − From Num1 = -10 Num2 = +20 Hash To o, it provides a leading zero. For %x and X, it provides a leading 0x or 0X respectively, only if the result is non-zero. For e, E, F, and F, the result still has a decimal point. For %g and G, trailing zeros are not removed from the result. The following example describes the following example − Example [jerry]$awk'START { print representation Octal = %#oHexadecimal representative = %#X,10,10}' On executing this code, you get the following result − From Octal representation = 012 Representation Hexadecimal = 0XA for more precise control over the output format passed what is provided by printing, using printing. With printer you can specify the width to use for each item, as well as various formatting choices for numbers (such as what base output to use, whether to print an exponent, whether to print a sign, and how many digits to print after the decimal point). This is our third post on AWK scripting. In this post we will cover print statements which many full uses for output formats as the loding spaces before and after the column entries etc. Previously covered topic in this series is AWK scripting: What is an AWK and how to use it? AWK scripting: 14 AWK print instance print is similar to AWK print statement but the advantage is that it can print with the output format in a desired way. So before you learn to print my order suggests you learn about print order and then come through this print statement. print syntax is similar to Bash, C print statement types. Syntax: awk '{print format, argument}' filename For example you want to print decimal value in column 3 then the example will awk '{print d,$3}' e.g. txt Print can do two things that the AWK print order cannot be 1) Define Data type. 2) Padding between columns. AWK printing supports the printer's data type can be useful when specifying data types such as Integer, Decimal, octal etc. Below is the list of some available data types in AWK. I or d -Decimal o --Octal%x-hex%c -ASCII character number s --String f-floating number: Make sure you pass the exact data type when using corresponding format as shown as below. If you pass a string to a decimal format, it will print just zero instead of that string. Lets start with some examples. for this post our test record is Jones 21 78 84 77 Gondrol 23 56 56 58 45 RinRao 25 21 38 37 Edwin 25 87 97 95 Jouan 24 55 30 47 Example 1: Write first column worth of db.tb.txt files. awk '{printf %sn,$1}' db.txt Output: Jones Gonrol RinRao Edwin Dayan Note: print won't have default new line char, so you include assignments when all the time you execute print commands as shown above. Example 2: Try printing a string with decimal formatting and seeing the difference. awk '{print %dn,$1}' db.txt Output: 0 0 0 0 0 0 So must deal with different types of data. Padding between columns using AWK Print Let's explore the available column format and print statement. Type Format: We can format columns to specify the number of carriages each column can use. We followed available padding formats and print statements. -n -Pad n Space on the right side of a column. We-pad our space on the left side of a column. .m - Add zero to left. -n.m - Pad us space right side and add me zero before this number.n.m – Pad we space left side and add me zero before that. Let's start exploring above mentioning the padding and example in detail. Example4: Pad 5 spaces on the right side of each column. With exit loding awk'{print d%d%dn,$2,$3,$4}' db.txt Output: 217884 235658 252138 2587977 245530 And padding 5 space on the right side: awk {printf -5d%-5d%-5d, $2,$3,$4}' db.txt Output: 21 78 84 23 56 58 25 28 28 87 27 25 50 Notes: As for understanding the reasons we provide this example with padding and without padding. We will add | between output columns so that pading can be explicitly seen as shown in below example. awk'{print|%-5d|-5d|-5d|n,$2,$3,$4}' db.txt Output: | The 21 | 78|84|| 23 The | 56 || The 25 | The 25 | 21 || The 25 | The 25 | 97 || 24 The | 24 The | 55 | 30's | Example 5: Pad 5 spaces on the left side of each column. awk'{printf|%5d|5d|5d|n, $2,$3,$4}' db.txt Output: | 21 May 2019 | The 780's | 84 The | 23 May 2019 56 Off III | 58 | 25 May 2019 | 21 May 2019 | The 38 | 25 May 2019 | The 87 ' | 97 | 24 May 2019 | The 55 | 30's | Example 6: Add the zero on the left side of each column element to make it a 5 digit number. awk '{print|. 5d|. 5d|. 5d|n, $2, $3, $4}' db.txt Output: | 00021|00078|00084|| 00023|00056|00058||00025|000221|00038|| 00025|00087|00097|| 00024|00055|00030| Example 7: Make the column inserted with the numbering digit and 7 in length and print the number at left hand awk'{print|%-7.4d|-7.4d|-7.4d|n,$2,$3,$4}' db.txt Output: | 0021 | 0078|0084|| 0023 | 0056 | 0058 || 0025 | 0021 | 0038 ||| 0025 | 0087 |0097|| 0024 | 0025 | 0030 | Example 8: Make the column inserted with the numbering digit and 7 in length and print the number at right side. awk'{print|%7.4d|7.4d|7.4d|n,$2,$3,$4}' db.txt Output: | 0021| 0078 | 0084 | 0023 The | 0056 | 0058 | 0025 | 0021| 0038 | 0025 | 0087 | 0097 | 0024 | 0055 | 0030 | Expect the following examples to help you understand about production formatting and order using print statements in AWK scripting. Keep visiting www.linuxnix.com for more scripting tutorials and tips. These two tabs change content below.