# MDEV-21978 Test Report

| Test Case | Sub Test | Configuration | Expected Output | Real Output | Notes |
|---|---|---|---|---|---|
| %sQ | ASCII Non-Printable U+0001 - u+0031 | Print as a loop that sets the buffer via {int} directly | Value matches output for backtick flag. | As expected | |
| | ASCII Printable Characters 32-0x7F | Print as a loop that sets the buffer via {int} directly | Characters are printed correctly | As expected | |
| | ASCII NUL 0000 \0 | set buffer[0]=0; | Ends string (doesn't actually print anything) | As expected | |
| | ASCII > U+007F (end 0xFF) | Print as a loop that sets the buffer via {int} directly | Nothing breaks, and the character representation for an unknown char is output | As expected | |
| | Multiple %sQ can be specified together | | Multiple %sQ can be specified together | As expected | |
| | Output matches behavior of old ‘`’ output for all above tests | | Output matches behavior of old ‘`’ output for all above tests | As expected | |
| %sB | Test simple characters | | Simple characters are printed | As expected | |
| | 0x0 does not end the string, and %sB should print passed it if <precision> specifies so | Using a char array of valid chars, set one index to 0 (end of string) | The entire array of characters should be stored | As expected | |
| | Escapes don't do anything | Escape a '\0' and make sure everything is still printed | All values used in the input sB should be retained as-is | As expected | |
| | | \0 can start a string | The whole string is still retained | As expected | |
| | | \0 can end a string | The 0 at the end is retained | As expected | |
| | Multiple %sB can be specified together | | | As expected | |
| | Output matches behavior of old ‘%b’ output for all above tests | | Output matches behavior of old ‘%b’ output for all above tests | As expected | |
| %iE | Test valid error code ranges | Loop through i from 1 to 201, and output the %iE | Each iteration prints the correct error number, as well as message description for that error number | As expected | |
| | Test invalid error code ranges | Negative values and 0 | Should mention that the error | As expected | The message for 0 vs. negative |

| | | | | | |
|---|---|---|---|---|---|
| | | | code number is not valid | | values is different. For 0 it is:<br><br>0 "Internal error/check (Not system error)"<br><br>And for general negative numbers it is:<br><br>-5 "Internal error < 0 (Not system error)" |
| | Test invalid error code ranges > 201 | | Display message about an unknown error number | As expected | 202 "Unknown error 202" |
| | Test multiple %iE invocations | | Each invocation display its error information correctly given the number in the varargs. | As expected | |
| | Give the input number weird values | Memory address | Value should be treated as a number, and the memory address should translate to an invalid error number that is displayed | As expected | |
| | | Character literal | Value should be treated as a number that is interpreted according to the valid/invalid value range behavior. | As expected | |
| | | NUL (‘\0’) character | Value should be treated as 0 and result in an invalid error number message | As expected | |
| | Output matches behavior of old ‘%M’ output for all above tests | | Output matches behavior of old ‘%M’ output for all above tests | As expected | |
| %sT | A string of size less than the truncation length should not be truncated | Truncation length = 5 | abcd -> abcd | As expected | |
| | A string of size equal to the truncation length should not be truncated | | abcde -> abcde | As expected | |
| | A string of size 1 over the truncation length should truncate the 3 characters at and before the index of the truncation length. | | abcdef -> ab… | As expected | |
| | A string of size 2 over the truncation length should truncate | | abcdefg -> ab… | As expected | |

| | | | | | |
|---|---|---|---|---|---|
| | the 3 characters at and before the index of the truncation length. | | | | |
| | A string of size 3 over the truncation length should truncate the 3 characters at and before the index of the truncation length. | | abcdefgh -> ab… | As expected | |
| | A string of size 5 over the truncation length should truncate the 3 characters at and before the index of the truncation length. | | abcdefghijk -> ab… | As expected | |
| | Two truncated strings in the same call should both truncate, correctly | | “abcdefghijk” and “uvwxyz” -> “ab…” and “uv…” | As expected | |
| | A string that is longer than the truncation length, but has a ‘\0’ before the truncation length inside it should not truncate, but rather end before the ‘\0’ | | ab\0defghijk -> ab | As expected | |
| | %sT should match the old %T behavior for all above tests | | %sT should match the old %T behavior for all above tests | As expected | |
| %sS | %sS should print a string | %sS | 123abcde\0fg -> 123abcde | As expected | |
| | %sS should adhere to precision | %.6sS | 123abcde\0fg -> 123abc | As expected | |
| | Multiple %sS should be ok | | %.3sS %sS % “abcdef” “ghi” -> “abc ghi” | As expected | |