```
========= macro test ==================
old register value: |macro:->This is the WRONG text.|
define env and execute: || =!= ||
new register value: |macro:->This is the right text.|
meaning env: |macro:->| =!= |macro:->|
expand env: || =!= ||
new register value (again): |macro:->This is the right text.|

========= toks test ==================
old register value: |This are NOT the toks I want.|
define env and execute: || =!= ||
new register value: |This are the toks I want.| =!= ||
meaning env: |macro:->| =!= |macro:->|
expand env: || =!= ||
new register value (again): |This are the toks I want.|

========= counts test ==================
old register value:|11|
define env and execute: || =!= ||
new register value: |22| =!= |22|
meaning env: |macro:->| =!= |macro:->|
expand env: || =!= ||
new register value (again): |22| =!= |22|

========= \@defaultunits test ==================
old register value: |0.0pt|
define env and execute (1): || =!= ||
new register value (1): |28.45274pt| >! |28pt|
meaning env (1): |macro:->| =!= |macro:->|
expand env (1): || =!= ||
define env and execute (2): || =!= ||
new register value (2): |1.0pt| =!= |1.0pt|
meaning env (2): |macro:->| =!= |macro:->|
expand env (2): || =!= ||

========= \relax test ==================
old register value:|11|
define env and execute: || =!= ||
new register value: |22| =!= |22|
meaning env: |macro:->| =!= |macro:->|
expand env: || =!= ||
new register value (again): |22| =!= |22|

========= \immediate\openout test ==================
define env and execute: ||
meaning env: |macro:->| =!= |macro:->|
expand env: || =!= ||

========= \immediate\write test ==================
define env and execute: ||

meaning env: |macro:->| =!= |macro:->|

expand env: || =!= ||


========= \immediate\closeout test ==================

define env and execute: ||

meaning env: |macro:->| =!= |macro:->|

expand env: || =!= ||


========= \openin test ==================

define env and execute: ||

meaning env: |macro:->| =!= |macro:->|

expand env: || =!= ||


========= \read test ==================

define env and execute: ||

meaning lualine: |macro:->Lua: I'm text immediately written to the .aux file with \command \%\a . |

Note that TeX appends the current \endlinechar at the end of every read line. TeX then converts (an
\endlinechar which is) <return> == ^^M == 13 to a space token.

meaning env: |macro:->\endlinechar=13| =!= |macro:->|

expand env: |\endlinechar=13| =!= ||


========= \readline test ==================

define env and execute: ||

meaning texline: |macro:->TeX: I'm text immediately written to the .aux file with \command \%\a .'|

Note that TeX appends the current \endlinechar at the end of every read line. eTeX NOT converts the appended
<return> == ^^M == 13 to a space token. As with other read characters, except the space, eTeX assigns
catcode 12 == other to these characters. And other characters are like letters simply typeset. In the (O)T1
font encoding at slot/code point 13 == ‘015 with the Computer Modern Typewriter font cmttxx is the glyph of
an apostrophe.

This code after this text in the test TeX file illustrates the issue.'

meaning env: |macro:->\endlinechar=13| =!= |macro:->|

expand env: |\endlinechar=13| =!= ||


========= \closein test ==================

define env and execute: ||

meaning env: |macro:->| =!= |macro:->|

expand env: || =!= ||


========= latex-env test ==================

old register value:|macro:->This is the FALSE text.|

define env and execute: ||

new register value (\meaning): |macro:->This is the right text: percent <\%>, space <\ >, backslash <\\>|

Note that we \def'ine the macro \todef␣and not \edef'ine it. Though the control symbols that are macros are
not yet expanded.

new register value (not \meaning): |This is the right text: percent <%>, space <␣>, backslash <\>|

meaning env: |macro:->% \|

expand env: |%␣\|

new register value (again): |macro:->This is the right text: percent <\%>, space <\ >, backslash <\\>|


========= \filedate test ==================
old register value:|macro:->This is NOT the current date.|

define env and execute: ||

new register value: |macro:->The current date is 2023/02/19.|

meaning env: |macro:->|

expand env: ||

new register value (again): |macro:->The current date is 2023/02/19.|


========= \expandafter test ==================

|| =!= ||

2 =!= 2


========= \begingroup & \endgroup test ==================

define env and execute: ||

\meaning\env: |macro:->(The next token is the letter "A".)A|


========= \bgroup & \egroup test ==================

define env and execute: ||

\meaning\env: |macro:->(The next token is NOT the letter "A".)B|


========= remaining braces test ==================

define env and execute: ||

\meaning\env: |macro:->{argument text}|


========= lots of text & \loop test ==================

|This is many text. This is many text. This is many text. This is many text. This is many text. This is many
text. This is many text. This is many text. This is many text. This is many text. This is many text. This is
many text. This is many text. This is many text. This is many text. This is many text. This is many text.
This is many text. This is many text. This is many text. This is many text. This is many text. This is many
text. This is many text. This is many text. This is many text. This is many text. This is many text. This is
many text. This is many text. This is many text. This is many text. This is many text. This is many text.
This is many text. This is many text. This is many text. This is many text. This is many text. This is many
text. |

Note that the last space before | is the same space as between every two sentences.


========= \par test ==================

define \todef and execute: ||

\meaning\todef: |macro:->first line\par second line|

simply expand and execute \todef: |first line

second line|


========= \protected tokens ==================

define \env and execute: || =!= ||

\meaning\env: |macro:->\protect \protectedTok | =!= |macro:->\protect \protectedTok |


========= test if group toks can be executed inside tex.runtoks() ==================

macro:->outside definition of \string \test

macro:->inside definition of \string \test

macro:->outside definition of \string \test
```