

```
-----  
/ This is \  
 \ ducksay! /  
-----
```

```
 \  
  \  
   --  
  >(' )  
   )/  
  /(  
 / '----/  
 \ ~== /  
 ~~~~~
```

```
-----  
( But which Version? )  
-----
```

```
 \  
  \  
   >()_  
   ( )_ _
```

```
-----  
( v2.7 )  
-----
```

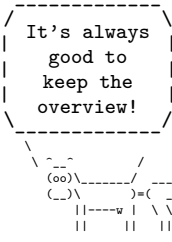
```
  ^--^  
 /-----/(oo) /  
/\(      /(\_) /  
 | w----||  
 ||      ||
```

```
-----  
( by Jonathan P. Spratte )  
-----
```

```
 /  
 .----- /  
 .'_/_|\_\'  
 /<::[0]8::>>\  
 |-----|  
 | | ----- | |  
 | | ===== | |  
 \ ||( )|:>::: | /  
 | ||( )|.... | |  
 | |-----| |  
 | |\_____|/ | |  
 / \ / \ / \ \  
 ( _ ) ( _ ) ( _ )
```

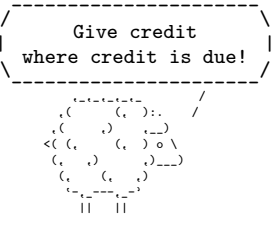
```
-----  
( Today is 2024-03-29 )  
-----
```

```
 \ .X|//|X| |.  
 \ |/\|/|/|/|/|  
 /. '|\X|/|/|/|  
 o--_ |//|/|X| |'
```

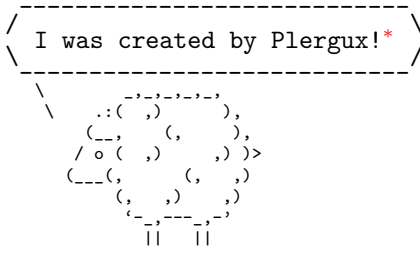


Contents

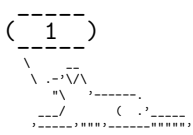
- 1 Acknowledgements** **1**
- 2 Documentation** **2**
 - 2.1 Downward Compatibility Issues 2
 - 2.2 Shared between versions 2
 - 2.2.1 Macros 2
 - 2.2.2 Options 3
 - 2.3 Version 1 6
 - 2.3.1 Introduektion 6
 - 2.3.2 Macros 6
 - 2.3.3 Options 6
 - 2.3.4 Defects 7
 - 2.4 Version 2 8
 - 2.4.1 Introduektion 8
 - 2.4.2 Macros 8
 - 2.4.3 Options 8
 - 2.5 Dependencies 14
 - 2.6 Available Animals 14
 - 2.7 License and Bug Reports 16
- 3 Implementation** **17**
 - 3.1 Shared between versions 17
 - 3.1.1 Variables 17
 - 3.1.2 Regular Expressions 17
 - 3.1.3 Messages 17
 - 3.1.4 Key-value setup 17
 - 3.1.5 Functions 19
 - 3.1.6 Load the Correct Version and the Animals 23
 - 3.2 Version 1 24
 - 3.2.1 Functions 24
 - 3.3 Version 2 27
 - 3.3.1 Messages 27
 - 3.3.2 Variables 27
 - 3.3.3 Options 27
 - 3.3.4 Functions 30
 - 3.4 Definition of the Animals 39



1 Acknowledgements



*<https://chat.stackexchange.com/transcript/message/55986902#55986902>



Just beest mod'rn,
thee peasant!



2 Documentation

This is ducksay! A cowsay for L^AT_EX. ducksay is part of T_EXLive and MiK_TE_X since September 2017. If it is not part of your installation it means that your L^AT_EX installation is *really* out of date, you have two options: Update your installation or try to install ducksay yourself. Chances are that if you opt for the latter, the version of expl3 in your L^AT_EX installation is too old, too, and the l3regex module is not yet part of expl3. In that case you'll get a few undefined control sequence errors. `\usepackage{l3regex}` prior to loading ducksay might fix these issues. Additionally you'll need grabbox for version 2 of ducksay that won't be part of your L^AT_EX installation, too. Please note that I don't actively support out of date L^AT_EX installations, so if loading l3regex doesn't fix the issues and you're on an old installation, I won't provide further support.

2.1 Downward Compatibility Issues

In the following list I use the term “version” to refer to package versions, the same is true if I use an abbreviation like “v2.0” (or anything that matches the regular expression `v\d+(\.d+)?`). For the code variant which can be set using the `version` option I'll use the term “variant” or specify directly that I'm referring to that option (the used font may be a hint, too).

- v2.0
 - Versions prior to v2.0 did use a regular expression for the option `ligatures`, see [subsection 2.2.2](#) for more on this issue.
 - In a document created with package versions prior to v2.0 you'll have to specify the option `version=1` with newer package versions to make those old documents behave like they used to.
- v2.3
 - Since v2.3 `\AddAnimal` and `\AddColoredAnimal` behave differently. You no longer have to make sure that in the first three lines every backslash which is only preceded by spaces is the bubble's tail. Instead you can specify which symbol should be the tail and how many of such symbols there are. See [subsection 2.2.1](#) for more about the current behaviour.
- v2.4
 - The `add-think` key was deprecated in v2.3 and was removed in v2.4 since the output symbols of the bubble tail are handled differently and more efficiently now.

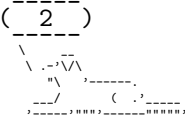
2.2 Shared between versions

2.2.1 Macros

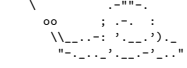
A careful reader might notice that in the below list of macros there is no `\ducksay` and no `\duckthink` contained. This is due to differences between the two usable code variants (see the `version` key in [subsection 2.2.2](#) for the code variants, [subsection 2.3.2](#) and [subsection 2.4.2](#) for descriptions of the two macros).

```
\DefaultAnimal \DefaultAnimal{<animal>}
```

use the `<animal>` if none is given in the optional argument to `\ducksay` or `\duckthink`. Package default is `duck`.



Yep, I screwed up!



Macros for everyone!



`\DucksayOptions` `\DucksayOptions{<options>}`

set the defaults to the keys described in [subsection 2.2.2](#), [subsection 2.3.3](#) and [subsection 2.4.3](#). Don't use an `<animal>` here, it has no effect.

`\AddAnimal` `\AddAnimal* [<options>]{<animal>}{<ascii-art>}`

adds `<animal>` to the known animals. `<ascii-art>` is multi-line verbatim and therefore should be delimited either by matching braces or by anything that works for `\verb`. If the star is given `<animal>` is the new default. One space is added to the begin of `<animal>` (compensating the opening symbol). The symbols signaling the speech bubble's tail (in the `hedgehog` example below the two `s`) can be set using the `tail-symbol` option and only the first `tail-count` occurrences will be substituted (see [paragraph 2.2.2.1](#) for more about these options). For example, `hedgehog` is added with:

```
\AddAnimal[tail-symbol=s]{hedgehog}
{ s .\|//||\||.
  s |/\|/||/|/|/|
  /. '|/\|/||/|/|
  o__._|//|/||\||}'
```

It is not checked whether the animal already exists, you could therefore redefine existing animals with this macro.

`\AddColoredAnimal` `\AddColoredAnimal* [<options>]{<animal>}{<ascii-art>}`

It does the same as `\AddAnimal` but allows three different colouring syntaxes. You can use `\textcolor` in the `<ascii-art>` with the syntax `\textcolor{<color>}{<text>}`. Note that you can't use braces in the arguments of `\textcolor`.

You can also use a delimited `\color` of the form `\bgroup\color{<color>}<text>\egroup`, a space after that `\egroup` will be considered a space in the output, so you don't have to care for correct termination of the `\egroup` (so `\bgroup\color{red}RedText\egroupOtherText` is valid syntax). You can't nest delimited `\colors`.

Also you can use an undelimited `\color`. It affects anything until the end of the current line (or, if used inside of the `<text>` of a delimited `\color`, anything until the end of that delimited `\color's` `<text>`). The syntax would be `\color{<color>}`.

The package doesn't load anything providing those colouring commands for you and it doesn't provide any coloured animals. The parsing is done using regular expressions provided by `LATEX3`. It is therefore slower than the normal `\AddAnimal`.

`\AnimalOptions` `\AnimalOptions*{<animal>}{<options>}`

With this macro you can set `<animal>` specific `<options>`. If the star is given any currently set options for this `<animal>` are dropped and only the ones specified in `<options>` will be applied, else `<options>` will be added to the set options for this `<animal>`. The set `<options>` can set the `tail-1` and `tail-2` options and therefore overwrite the effects of `\duckthink`, as `\duckthink` really is just `\ducksay` with the `think` option.

2.2.2 Options

The following options are available independent on the used code variant (the value of the `version` key). They might be used as package options – unless otherwise specified – or used in the macros `\DucksayOptions`, `\ducksay` and `\duckthink` – again unless otherwise specified. Some options might be accessible in both code variants but do

```
Options.
For every occasion
{
  \_// }
  (')_ }
  '---' { }--{ }
  // / /
```

```
( 3 )
\ .-'√\
"
,-----
,-----
,-----
```

slightly different things. If that's the case they will be explained in [subsection 2.3.3](#) and [subsection 2.4.3](#) for `version 1` and `2`, respectively.

`version=<number>`

With this you can choose the code variant to be used. Currently `1` and `2` are available. This can be set only during package load time. For a dedicated description of each version look into [subsection 2.3](#) and [subsection 2.4](#). The package author would choose `version=2`, the other version is mostly for legacy reasons. The default is `2`.

`<animal>`

One of the animals listed in [subsection 2.6](#) or any of the ones added with `\AddAnimal`. Not useable as package option. Also don't use it in `\DucksayOptions`, it'll break the default animal selection.

`animal=<animal>`

Locally sets the default animal. Note that `\ducksay` and `\duckthink` do digest their options inside of a group, so it just results in a longer alternative to the use of `<animal>` if used in their options.

`ligatures=<token list>`

each token you don't want to form ligatures during `\AddAnimal` should be contained in this list. All of them get enclosed by grouping `{` and `}` so that they can't form ligatures. Giving no argument (or an empty one) might enhance compilation speed by disabling this replacement. The formation of ligatures was only observed in combination with `\usepackage[T1]{fontenc}` by the author of this package. Therefore giving the option `ligatures` without an argument might enhance the compilation speed for you without any drawbacks. Initially this is set to `'<>, '-`.

Note: In earlier releases this option's expected argument was a regular expression. This means that this option is not fully downward compatible with older versions. The speed gain however seems worth it (and I hope the affected documents are few).

`no-tail`

Sets `tail-1` and `tail-2` to be a space.

`random=<bool>`

If `true` a random animal will be used instead of the default one on each usage of `\ducksay` or `\duckthink`. The initial value is `false` and it defaults to `true`.

`say`

Sets `tail-1` and `tail-2` as backslashes.

`schroedinger`

Sets randomly either `animal=schroedinger-alive` or `animal=schroedinger-dead` at the time of use. Both have the same size, so this doesn't affect the needed space.

`tail-1=<token list>`

Sets the first tail symbol in the output to be `<token list>`. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be `0`.

`tail-2=<token list>`

Sets every other tail symbol except the first one in the output to be `<token list>`. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be `o`.

`think`

Sets `tail-1=0` and `tail-2=o`.

```
( 4 )
\ .-'\ \
" \
\ /
,-----
```

2.2.2.1 Options for `\AddAnimal`

The options described here are only available in `\AddAnimal` and `\AddColoredAnimal`.

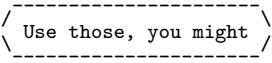
`tail-count=<int>`

sets the number of tail symbols to be replaced in `\AddAnimal` and `\AddColoredAnimal`.
Initial value is 2. If the value is negative every occurrence of `tail-symbol` will be replaced.

`tail-symbol=<str>`

the symbol used in `\AddAnimal` and `\AddColoredAnimal` to mark the bubble's tail. The
argument gets `\detokenized`. Initially a single backslash.

```
( 5 )
 \ .-.\
  " \  ( )
 ,-----
```



Use those, you might

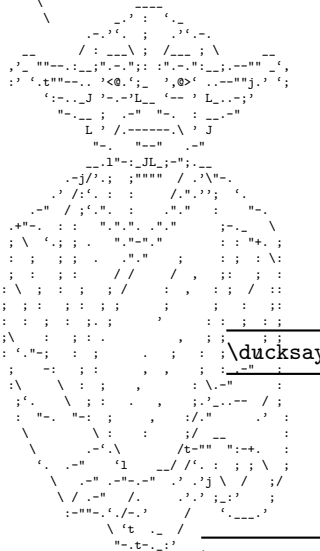
2.3 Version 1

2.3.1 Introduction

This version is included for legacy support (old documents should behave the same without any change to them – except the usage of `version=1` as an option, for a more or less complete list of downward compatibility related problems see [subsection 2.1](#)). For the bleeding edge version of `ducksay` skip this subsection and read [subsection 2.4](#).

2.3.2 Macros

The following is the description of macros which differ in behaviour from those of version 2.

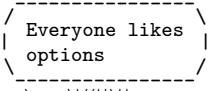


```
\ducksay \ducksay[options]{message}
```

`options` might include any of the options described in [subsection 2.2.2](#) and [subsection 2.3.3](#) if not otherwise specified. Prints an `<animal>` saying `<message>`. `<message>` is not read in verbatim. Multi-line `<message>`s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.

```
\duckthink \duckthink[options]{message}
```

`options` might include any of the options described in [subsection 2.2.2](#) and [subsection 2.3.3](#) if not otherwise specified. Prints an `<animal>` thinking `<message>`. `<message>` is not read in verbatim. Multi-line `<message>`s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.



Everyone likes options

2.3.3 Options

The following options are available to `\ducksay`, `\duckthink`, and `\DucksayOptions` and if not otherwise specified also as package options:

```
bubble=code
```

use `<code>` in a group right before the bubble (for font switches). Might be used as a package option but not all control sequences work out of the box there.

```
body=code use <code> in a group right before the body (meaning the <animal>). Might be used as a package option but not all control sequences work out of the box there. E.g. to right-align the <animal> to the bubble, use body=\hfill.
```

```
align=valign
```

use `<valign>` as the vertical alignment specifier given to the `tabular` which is around the contents of `\ducksay` and `\duckthink`.

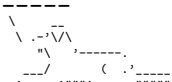
```
msg-align=halign
```

use `<halign>` for alignment of the rows of multi-line `<message>`s. It should match a `tabular` column specifier. Default is `l`. It only affects the contents of the speech bubble not the bubble.

```
rel-align=column
```

use `<column>` for alignment of the bubble and the body. It should match a `tabular` column specifier. Default is `l`.

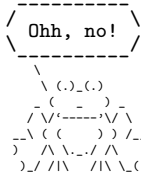
(6)



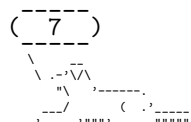
`wd=<count>` in order to detect the width the `<message>` is expanded. This might not work out for some commands (e.g. `\url` from `hyperref`). If you specify the width using `wd` the `<message>` is not expanded and therefore the command *might* work out. `<count>` should be the character count.

`ht=<count>` you might explicitly set the height (the row count) of the `<message>`. This only has an effect if you also specify `wd`.

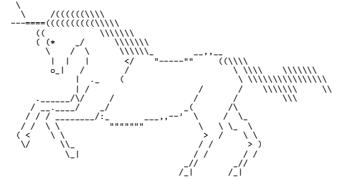
2.3.4 Defects



- no automatic line wrapping
- message width detection based on token count with `\edef` expansion, might fail badly



Here's all the good stuff!



2.4 Version 2

2.4.1 Introduction

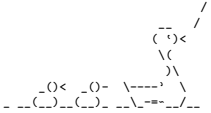
Version 2 is the current version of `ducksay`. It features automatic line wrapping (if you specify a fixed width) and in general more options (with some nasty argument parsing).

If you're already used to version 1 you should note one important thing: You should only specify the `version` and the `ligatures` during package load time as arguments to `\usepackage`. The other keys might not work or do unintended things and only don't throw errors or warnings because of the legacy support of version 1. After the package is loaded, keys only used for version 1 will throw an error.

2.4.2 Macros

The following is the description of macros which differ in behaviour from those of version 1.

Look at those, kids!



`\ducksay` `\ducksay[options]{message}`

options might include any of the options described in [subsection 2.2.2](#) and [subsection 2.4.3](#) if not otherwise specified. Prints an `<animal>` saying `<message>`.

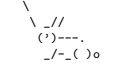
The `<message>` can be read in in four different ways. For an explanation of the `<message>` reading see the description of the `arg` key in [subsection 2.4.3](#).

The height and width of the message is determined by measuring its dimensions and the bubble will be set accordingly. The box surrounding the message will be placed both horizontally and vertically centred inside of the bubble. The output utilizes L^AT_EX₃'s coffin mechanism described in [interface3.pdf](#) and the documentation of `xcoffins`.

`\duckthink` `\duckthink[options]{message}`

The only difference to `\ducksay` is that in `\duckthink` the `<animal>`s think the `<message>` and don't say it.

Fast, use options!



2.4.3 Options

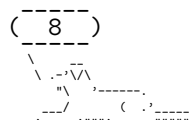
In version 2 the following options are available. Keep in mind that you shouldn't use them during package load time but in the arguments of `\ducksay`, `\duckthink` or `\DucksayOptions`.

`arg=<choice>`

specifies how the `<message>` argument of `\ducksay` and `\duckthink` should be read in. Available options are `box`, `tab` and `tab*`:

`box` the argument is read in either as a `\hbox` or a `\vbox` (the latter if a fixed width is specified with either `wd` or `wd*`). Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work (provided that you don't use `\ducksay` or `\duckthink` inside of an argument of another macro of course).

`tab` the argument is read in as the contents of a `tabular`. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will *not* work. This mode comes closest to the behaviour of version 1 of `ducksay`.



tab*

the argument is read in as the contents of a `tabular`. However it is read in verbatim and uses `\scantokens` to rescan the argument. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work. You can't use `\ducksay` or `\duckthink` as an argument to another macro in this mode however.

b shortcut for `out-v=b`.

`body=` add `` to the font definitions in use to typeset the `<animal>`'s body.

`body**=`

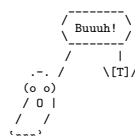
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the `<animal>`'s body to ``. The package default is `\verbatim@font`. In addition `\frenchspacing` will always be used prior to the defined ``.

`body-align=<choice>`

sets the relative alignment of the `<animal>` to the `<message>`. Possible choices are `l`, `c` and `r`. For `l` the `<animal>` is flushed to the left of the `<message>`, for `c` it is centred and for `r` it is flushed right. More fine grained control over the alignment can be obtained with the keys `msg-to-body`, `body-to-msg`, `body-x` and `body-y`. Package default is `l`.

`body-bigger=<count>`

vertically enlarge the body by `<count>` empty lines added to the bottom. This way top-aligning two different body types is easier (by actually bottom aligning the two):



```
\ducksay[ghost,body-x=-7mm,b,body-mirrored]{Buuhh!}
\ducksay[crusader,body-bigger=4,b,out-h=r,no-bubble]{}
```

`body-mirrored=<bool>`

if set true the `<animal>` will be mirrored along its vertical centre axis. Package default is `false`. If you set it `true` you'll most likely need to manually adjust the alignment of the body with one or more of the keys `body-align`, `body-to-msg`, `msg-to-body`, `body-x` and `body-y`.

`body-to-msg=<pole>`

defines the horizontal coffin `<pole>` to be used for the placement of the `<animal>` beneath the `<message>`. See [interface3.pdf](#) and the documentation of `xcoffins` for information about coffin poles.

`body-x=<dimen>`

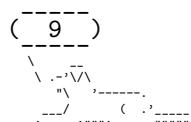
defines a horizontal offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

`body-y=<dimen>`

defines a vertical offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

`bubble=`

add `` to the font definitions in use to typeset the bubble. This does not affect the `<message>` only the bubble put around it.



- `bubble*=`
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the bubble to **. This does not affect the *<message>* only the bubble put around it. The package default is `\verbatim@font`.
- `bubble-bot-kern=<dimen>`
specifies a vertical offset of the placement of the lower border of the bubble from the bottom of the left and right borders.
- `bubble-delim-left-1=<token list>`
the left delimiter used if only one line of delimiters is needed. Package default is `(`.
- `bubble-delim-left-2=<token list>`
the upper most left delimiter used if more than one line of delimiters is needed. Package default is `/`.
- `bubble-delim-left-3=<token list>`
the left delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.
- `bubble-delim-left-4=<token list>`
the lower most left delimiter used if more than one line of delimiters is needed. Package default is `\`.
- `bubble-delim-right-1=<token list>`
the right delimiter used if only one line of delimiters is needed. Package default is `)`.
- `bubble-delim-right-2=<token list>`
the upper most right delimiter used if more than one line of delimiters is needed. Package default is `\`.
- `bubble-delim-right-3=<token list>`
the right delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.
- `bubble-delim-right-4=<token list>`
the lower most right delimiter used if more than one line of delimiters is needed. Package default is `/`.
- `bubble-delim-top=<token list>`
the delimiter used to create the top and bottom border of the bubble. The package default is `{-}` (the braces are important to suppress ligatures here).
- `bubble-side-kern=<dimen>`
specifies the kerning used to move the sideways delimiters added to fill the gap for more than two lines of bubble height. (the left one is moved to the left, the right one to the right)
- `bubble-top-kern=<dimen>`
specifies a vertical offset of the placement of the upper border of the bubble from the top of the left and right borders.
- `c` shortcut for `out-v=vc`.

```

-----
( 10 )
-----
 \ .-.'√\
  "     '-----
  /      ( '-----
-----

```


- `msg-align-r=<token list>`
 set the `<token list>` which is responsible to typeset the message flushed right if the option `msg-align=r` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is `\raggedleft`. It might be useful if you want to use `ragged2e`'s `\RaggedLeft` for example.
- `msg-to-body=<pole>`
 defines the horizontal coffin `<pole>` to be used as the reference point for the placement of the `<animal>` beneath the `<message>`. See [interface3.pdf](#) and the documentation of `xcoffins` for information about coffin poles.
- `no-bubble=<bool>`
 If `true` the `<message>` will not be surrounded by a bubble. Package default is of course `false`.
- `none=<bool>` One could say this is a special animal. If `true` no animal body will be used (resulting in just the speech bubble). Package default is of course `false`.
- `out-h=<pole>`
 defines the horizontal coffin `<pole>` to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See [interface3.pdf](#) and the documentation of `xcoffins` for information about coffin poles.
- `out-v=<pole>`
 defines the vertical coffin `<pole>` to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See [interface3.pdf](#) and the documentation of `xcoffins` for information about coffin poles.
- `out-x=<dimen>`
 specifies an additional horizontal offset of the print out of the complete result of `\ducksay` and `\duckthink`.
- `out-y=<dimen>`
 specifies an additional vertical offset of the print out of the complete result of `\ducksay` and `\duckthink`.
- `strip-spaces=<bool>`
 if set `true` leading and trailing spaces are stripped from the `<message>` if `arg=box` is used. Initially this is set to `false`.
- `t` shortcut for `out-v=t`.
- `vpad=<count>`
 add `<count>` to the lines used for the bubble, resulting in `<count>` more lines than necessary to enclose the `<message>` inside of the bubble.
- `wd=<count>` specifies the width of the `<message>` to be fixed to `<count>` times the width of an upper case M in the `<message>`'s font declaration. A value smaller than 0 is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a p-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.

```

-----
( 12 )
-----
 \ .-'\ \
  " \   \
  /    \
-----
,-----

```


We rely on you

Slowly new animals are added.

2.5 Dependencies

The package depends on the L^AT_EX kernel, for older versions of L^AT_EX the two packages xparse and l3keys2e and all of their dependencies are loaded. Version 2 additionally depends on array and grabbox.

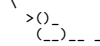
2.6 Available Animals

The following animals are provided by this package. I did not create them (but altered some), they belong to their original creators.

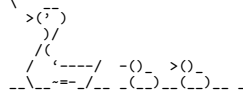
(duck)



(small-duck)



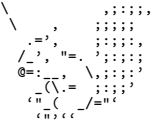
(duck-family)



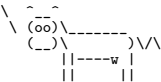
(small-rabbit)



(squirrel)



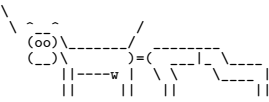
(cow)



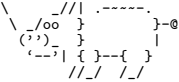
(tux)



(head-in)



(pig)



(frog)



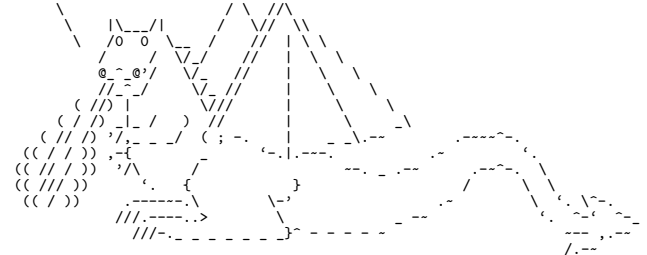
(snowman)



(bunny)



(dragon)



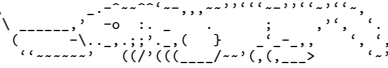
(sodomized)



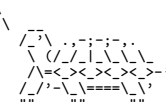
(hedgehog)



(platypus)



(turtle)



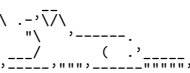
(kangaroo)



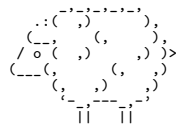
(small-horse)



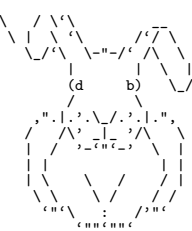
(dog)



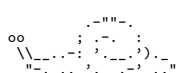
(sheep)



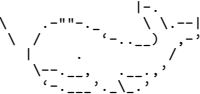
(rabbit)



(snail)



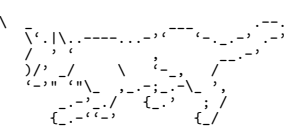
(whale)



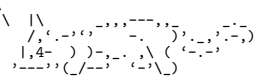
(snake)



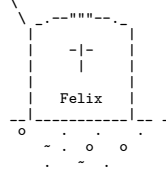
(cat)



(sleepy-cat)



(schroedinger-dead)



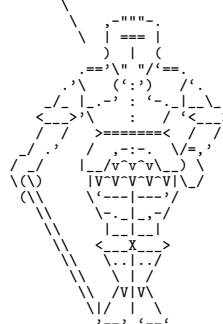
(schroedinger-alive)



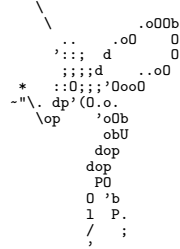
(crusader)



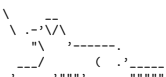
(knight)

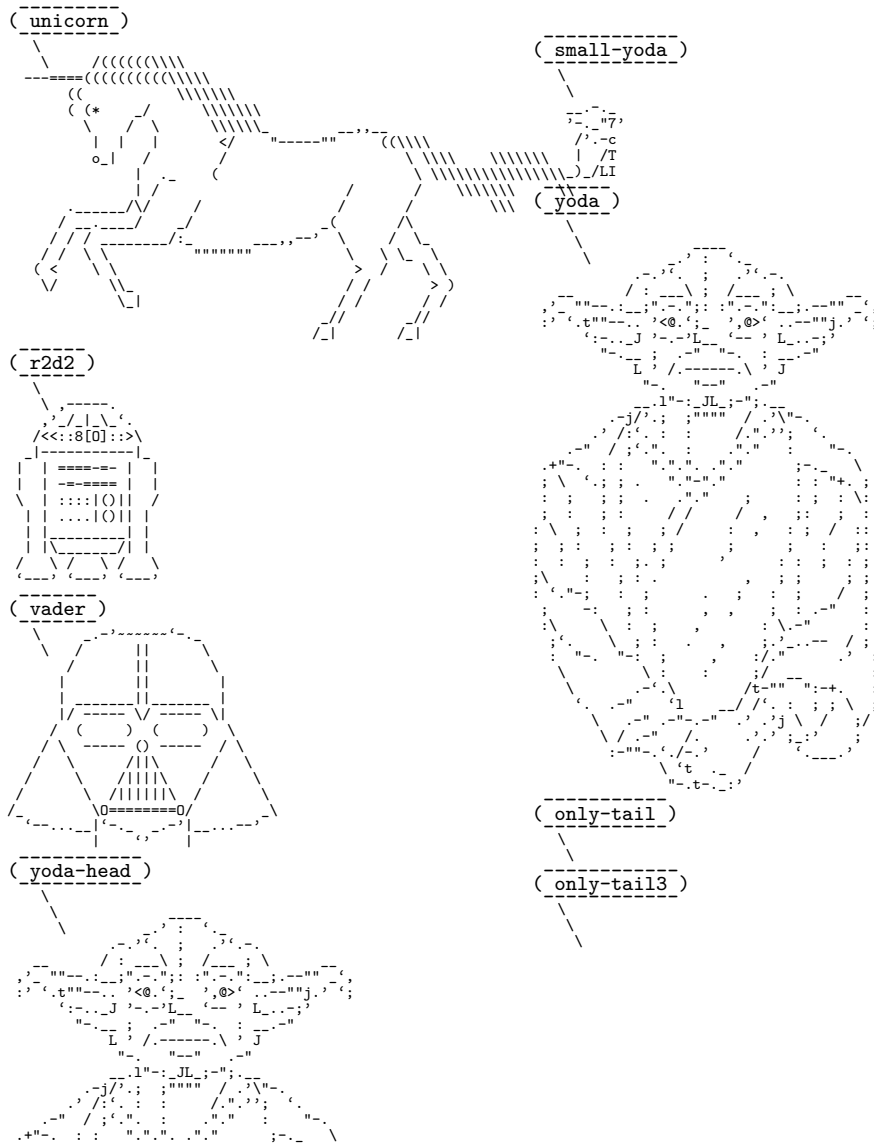


(fairy)



(ghost)

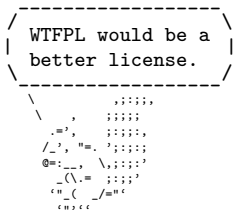




2.7 License and Bug Reports

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License (LPPL), either version 1.3c of this license or (at your option) any later version. The latest version of this license is in the file: <http://www.latex-project.org/lppl.txt>

The package is hosted on https://github.com/Skillmon/ltx_ducksay, you might report bugs there.




```

32 ,align .tl_set:N = \l_ducksay_align_tl
33 ,align .value_required:n = true
34 ,wd .int_set:N = \l_ducksay_msg_width_int
35 ,wd .initial:n = -\c_max_int
36 ,wd .value_required:n = true
37 ,ht .int_set:N = \l_ducksay_msg_height_int
38 ,ht .initial:n = -\c_max_int
39 ,ht .value_required:n = true
40 ,animal .code:n =
41 { \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } } }
42 ,animal .initial:n = duck
43 ,msg-align .tl_set:N = \l_ducksay_msg_align_tl
44 ,msg-align .initial:n = 1
45 ,msg-align .value_required:n = true
46 ,rel-align .tl_set:N = \l_ducksay_rel_align_tl
47 ,rel-align .initial:n = 1
48 ,rel-align .value_required:n = true
49 ,ligatures .tl_set:N = \l_ducksay_ligatures_tl
50 ,ligatures .initial:n = { '<>,'- }
51 ,tail-1 .tl_set:N = \l_ducksay_tail_symbol_out_one_tl
52 ,tail-1 .initial:x = \c_backslash_str
53 ,tail-2 .tl_set:N = \l_ducksay_tail_symbol_out_two_tl
54 ,tail-2 .initial:x = \c_backslash_str
55 ,no-tail .meta:n = { tail-1 = { ~ }, tail-2 = { ~ } }
56 ,think .meta:n = { tail-1 = { 0 }, tail-2 = { o } }
57 ,random .bool_set:N = \l_ducksay_random_animal_bool
58 ,say .code:n =
59 {
60 \exp_args:Nx \DucksayOptions
61 { tail-1 = { \c_backslash_str }, tail-2 = { \c_backslash_str } }
62 }
63 ,schroedinger .code:n =
64 {
65 \int_compare:nNnTF { \int_rand:n { 2 } } = \c_one_int
66 { \keys_set:nn { ducksay } { animal = schroedinger-dead } }
67 { \keys_set:nn { ducksay } { animal = schroedinger-alive } }
68 }
69 ,schroedinger .value_forbidden:n = true
70 ,version .choice:
71 ,version / 1 .code:n =
72 {
73 \bool_set_false:N \l_ducksay_version_two_bool
74 \bool_set_true:N \l_ducksay_version_one_bool
75 }
76 ,version / 2 .code:n =
77 {
78 \bool_set_false:N \l_ducksay_version_one_bool
79 \bool_set_true:N \l_ducksay_version_two_bool
80 }
81 ,version .initial:n = 2
82 }
83 \cs_if_exist:NTF \ProcessKeyOptions
84 { \ProcessKeyOptions [ ducksay ] }
85 {

```

```

86   \RequirePackage { l3keys2e }
87   \ProcessKeysOptions { ducksay }
88   }

   Undefine the load-time-only keys
89 \keys_define:nn { ducksay }
90   {
91   version .code:n = \msg_error:nnn { ducksay } { load-time-only } { version }
92   }

```

3.1.4.1 Keys for \AddAnimal

Define keys meant for \AddAnimal and \AddColoredAnimal only in their own regime:

```

93 \keys_define:nn { ducksay / add-animal }
94   {
95   ,tail-symbol .code:n =
96   \tl_set:Nx \l_ducksay_tail_symbol_in_tl { \tl_to_str:n { #1 } }
97   ,tail-symbol .initial:o = \c_backslash_str
98   ,tail-count .int_set:N = \l_ducksay_tail_symbol_count_int
99   ,tail-count .initial:n = 2
100  }

```

3.1.5 Functions

3.1.5.1 Generating Variants of External Functions

```

101 \cs_generate_variant:Nn \tl_replace_once:Nnn { NVn }
102 \cs_generate_variant:Nn \tl_replace_all:Nnn { NVn }
103 \cs_generate_variant:Nn \keys_set:nn { nx }

```

3.1.5.2 Internal

_ducksay_everyeof:w

```

104 \cs_set_eq:NN \_ducksay_everyeof:w \tex_everyeof:D

```

(End of definition for _ducksay_everyeof:w.)

_ducksay_scantokens:w

```

105 \cs_set_eq:NN \_ducksay_scantokens:w \tex_scantokens:D

```

(End of definition for _ducksay_scantokens:w.)

\ducksay_replace_verb_newline:Nn

```

106 \IfFormatAtLeastTF{2024-06-01}
107   {
108   \cs_new_protected:Npn \ducksay_replace_verb_newline:Nn #1 #2
109     { \tl_replace_all:Nnn #1 \obeyedline {#2} }
110   }
111   {
112   \cs_new_protected:Npx \ducksay_replace_verb_newline:Nn #1 #2
113     { \tl_replace_all:Nnn #1 { \char_generate:mn { 13 } { 12 } } {#2} }
114   }

```

(End of definition for \ducksay_replace_verb_newline:Nn.)

\ducksay_replace_verb_newline_newline:Nn

```
115 \IfFormatAtLeastTF{2024-06-01}
116 {
117   \cs_new_protected:Npn \ducksay_replace_verb_newline_newline:Nn #1 #2
118     { \tl_replace_all:Nnn #1 { \obeyedline \obeyedline } {#2} }
119 }
120 {
121   \cs_new_protected:Npx \ducksay_replace_verb_newline_newline:Nn #1 #2
122     {
123       \tl_replace_all:Nnn #1
124         { \char_generate:nn { 13 } { 12 } \char_generate:nn { 13 } { 12 } }
125         {#2}
126     }
127 }
```

(End of definition for \ducksay_replace_verb_newline_newline:Nn.)

\ducksay_process_verb_newline:nnn

```
128 \cs_new_protected:Npn \ducksay_process_verb_newline:nnn #1 #2 #3
129 {
130   \tl_set:Nn \ProcessedArgument { #3 }
131   \ducksay_replace_verb_newline_newline:Nn \ProcessedArgument { #2 }
132   \ducksay_replace_verb_newline:Nn \ProcessedArgument { #1 }
133 }
```

(End of definition for \ducksay_process_verb_newline:nnn.)

\ducksay_add_animal_inner:nnnn

```
134 \cs_new_protected:Npn \ducksay_add_animal_inner:nnnn #1 #2 #3 #4
135 {
136   \group_begin:
137   \keys_set:nn { ducksay / add-animal } { #1 }
138   \tl_set:Nn \l_ducksay_tmpa_tl { \ #3 }
139   \int_compare:nNnTF { \l_ducksay_tail_symbol_count_int } < { \c_zero_int }
140     {
141       \tl_replace_once:NVn
142         \l_ducksay_tmpa_tl
143         \l_ducksay_tail_symbol_in_tl
144         \l_ducksay_tail_symbol_out_one_tl
145       \tl_replace_all:NVn
146         \l_ducksay_tmpa_tl
147         \l_ducksay_tail_symbol_in_tl
148         \l_ducksay_tail_symbol_out_two_tl
149     }
150   {
151     \int_compare:nNnT { \l_ducksay_tail_symbol_count_int } >
152       { \c_zero_int }
153     {
154       \tl_replace_once:NVn
155         \l_ducksay_tmpa_tl
156         \l_ducksay_tail_symbol_in_tl
157         \l_ducksay_tail_symbol_out_one_tl
158       \int_step_inline:nnn { 2 } { \l_ducksay_tail_symbol_count_int }
159     }
```

(20)

\\ .-'\V\
")
/ (,

```

160         \tl_replace_once:NVn
161         \l_ducksay_tmpa_tl
162         \l_ducksay_tail_symbol_in_tl
163         \l_ducksay_tail_symbol_out_two_tl
164     }
165 }
166 }
167 \tl_map_inline:Nn \l_ducksay_ligatures_tl
168 { \tl_replace_all:Nnn \l_ducksay_tmpa_tl { ##1 } { { ##1 } } }
169 \ducksay_replace_verb_newline:Nn \l_ducksay_tmpa_tl
170 { \tabularnewline\null }
171 \exp_args:NNnV
172 \group_end:
173 \tl_set:cn { l_ducksay_animal_#2_tl } \l_ducksay_tmpa_tl
174 \exp_args:Nnx \keys_define:nn { ducksay }
175 {
176     #2 .code:n =
177     {
178         \exp_not:n { \tl_set_eq:NN \l_ducksay_animal_tl }
179         \exp_not:c { l_ducksay_animal_#2_tl }
180         \exp_not:n { \exp_args:NV \DucksayOptions }
181         \exp_not:c { l_ducksay_animal_#2_options_tl }
182     }
183 }
184 \tl_if_exist:cF { l_ducksay_animal_#2_options_tl }
185 { \tl_new:c { l_ducksay_animal_#2_options_tl } }
186 \IfBooleanT { #4 }
187 { \keys_define:nn { ducksay } { default_animal .meta:n = { #2 } } }
188 \seq_if_in:NnF \l_ducksay_defined_animals_seq { #2 }
189 { \seq_push:Nn \l_ducksay_defined_animals_seq { #2 } }
190 }
191 \cs_generate_variant:Nn \ducksay_add_animal_inner:nmmm { nnVn }

```

(End of definition for \ducksay_add_animal_inner:nmmm.)

\ducksay_default_or_random_animal:

```

192 \cs_new_protected:Npn \ducksay_default_or_random_animal:
193 {
194     \tl_if_empty:NT \l_ducksay_animal_tl
195     {
196         \bool_if:NTF \l_ducksay_random_animal_bool
197         {
198             \keys_set:nx { ducksay }
199             { \seq_rand_item:N \l_ducksay_defined_animals_seq }
200         }
201         { \keys_set:nn { ducksay } { default_animal } }
202     }
203 }

```

(End of definition for \ducksay_default_or_random_animal:.)

3.1.5.3 Document level

\DefaultAnimal

```

204 \NewDocumentCommand \DefaultAnimal { m }
205 {
206   \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } }
207 }

```

(End of definition for `\DefaultAnimal`. This function is documented on page 2.)

`\DucksayOptions`

```

208 \NewDocumentCommand \DucksayOptions { m }
209 {
210   \keys_set:nn { ducksay } { #1 }
211 }

```

(End of definition for `\DucksayOptions`. This function is documented on page 3.)

`\AddAnimal`

```

212 \NewDocumentCommand \AddAnimal { s O{} m +v }
213 {
214   \ducksay_add_animal_inner:nmmm { #2 } { #3 } { #4 } { #1 }
215 }

```

(End of definition for `\AddAnimal`. This function is documented on page 3.)

`\AddColoredAnimal`

```

216 \NewDocumentCommand \AddColoredAnimal { s O{} m +v }
217 {
218   \tl_set:Nn \l_ducksay_tmpa_tl { #4 }
219   \regex_replace_all:NnN \c_ducksay_color_delim_regex
220     { \c{bgroup}\c{color}\cB{\1\cE}\2\c{egroup} }
221     \l_ducksay_tmpa_tl
222   \regex_replace_all:NnN \c_ducksay_color_regex
223     { \c{color}\cB{\1\cE}\ }
224     \l_ducksay_tmpa_tl
225   \regex_replace_all:NnN \c_ducksay_textcolor_regex
226     { \c{textcolor}\cB{\1\cE}\cB{\2\cE}\ }
227     \l_ducksay_tmpa_tl
228   \ducksay_add_animal_inner:nmVn { #2 } { #3 } \l_ducksay_tmpa_tl { #1 }
229 }

```

(End of definition for `\AddColoredAnimal`. This function is documented on page 3.)

`\AnimalOptions`

```

230 \NewDocumentCommand \AnimalOptions { s m m }
231 {
232   \tl_if_exist:cTF { l_ducksay_animal_#2_options_tl }
233     {
234       \IfBooleanTF { #1 }
235         { \tl_set:cn }
236         { \tl_put_right:cn }
237     }
238     { \tl_set:cn }
239     { l_ducksay_animal_#2_options_tl } { #3, }
240 }

```

(End of definition for `\AnimalOptions`. This function is documented on page 3.)

3.1.6 Load the Correct Version and the Animals

```
241 \bool_if:NT \l_ducksay_version_one_bool
242   { \file_input:n { ducksay.code.v1.tex } }
243 \bool_if:NT \l_ducksay_version_two_bool
244   { \file_input:n { ducksay.code.v2.tex } }

245 \ExplSyntaxOff
246 \input{ducksay.animals.tex}

247 </pkg>
```


3.2 Version 1

```

248 <*code.v1>
249 \ProvidesFile{ducksay.code.v1.tex}
250 [\ducksay@date\space v\ducksay@version\space ducksay code version 1]

```

3.2.1 Functions

3.2.1.1 Internal

`\ducksay_longest_line:n` Calculate the length of the longest line

```

251 \cs_new:Npn \ducksay_longest_line:n #1
252 {
253   \int_incr:N \l_ducksay_msg_height_int
254   \exp_args:NNx \tl_set:Nn \l_ducksay_tmpa_tl { #1 }
255   \regex_replace_all:nnN { \s } { \c { space } } \l_ducksay_tmpa_tl
256   \int_set:Nn \l_ducksay_msg_width_int
257   {
258     \int_max:nn
259     { \l_ducksay_msg_width_int } { \tl_count:N \l_ducksay_tmpa_tl }
260   }
261 }

```

(End of definition for \ducksay_longest_line:n.)

`\ducksay_open_bubble:` Draw the opening bracket of the bubble

```

262 \cs_new:Npn \ducksay_open_bubble:
263 {
264   \begin{tabular}{@{}l@{}}
265     \null\
266     \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 } { ( }
267     {
268       /
269       \int_step_inline:nnn
270       { 3 } { \l_ducksay_msg_height_int } { \\kern-0.2em| }
271       \\detokenize{\ }
272     }
273     \\[-1ex]\null
274   \end{tabular}
275   \begin{tabular}{@{}l@{}}
276     _\
277     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
278     \mbox { - }
279   \end{tabular}
280 }

```

(End of definition for \ducksay_open_bubble:.)

`\ducksay_close_bubble:` Draw the closing bracket of the bubble

```

281 \cs_new:Npn \ducksay_close_bubble:
282 {
283   \begin{tabular}{@{}l@{}}
284     _\
285     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
286     { - }
287   \end{tabular}

```

```

-----
( 24 )
-----
\ .-'\
" \
-----
-----

```

```

288 \begin{tabular}{@{}r@{}}
289 \null\
290 \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 }
291 { ) }
292 {
293 \detokenize {\ }
294 \int_step_inline:nnn
295 { 3 } { \l_ducksay_msg_height_int } { \\\kern-0.2em }
296 \\\
297 }
298 \\\[-1ex]\null
299 \end{tabular}
300 }

```

(End of definition for `\ducksay_close_bubble:`.)

`\ducksay_print_msg:nn` Print out the message

```

301 \cs_new:Npn \ducksay_print_msg:nn #1 #2
302 {
303 \begin{tabular}{@{} #2 @{} }
304 \int_step_inline:nn { \l_ducksay_msg_width_int } { _ } \\\
305 #1\\\[-1ex]
306 \int_step_inline:nn { \l_ducksay_msg_width_int } { { - } }
307 \end{tabular}
308 }
309 \cs_generate_variant:Nn \ducksay_print_msg:nn { nV }

```

(End of definition for `\ducksay_print_msg:nn`.)

`\ducksay_print:nn` Print out the whole thing

```

310 \cs_new:Npn \ducksay_print:nn #1 #2
311 {
312 \int_compare:nNnTF { \l_ducksay_msg_width_int } < { 0 }
313 {
314 \int_zero:N \l_ducksay_msg_height_int
315 \seq_set_split:Nnn \l_ducksay_msg_lines_seq { \\\ } { #1 }
316 \seq_map_function:NN \l_ducksay_msg_lines_seq \ducksay_longest_line:n
317 }
318 {
319 \int_compare:nNnT { \l_ducksay_msg_height_int } < { 0 }
320 {
321 \regex_count:nnN { \c { \\\ } } { #1 } \l_ducksay_msg_height_int
322 \int_incr:N \l_ducksay_msg_height_int
323 }
324 }
325 \group_begin:
326 \frenchspacing
327 \verbatim@font
328 \@noligs
329 \begin{tabular}[\l_ducksay_align_tl]{@{}#2@{}}
330 \l_ducksay_bubble_tl
331 \begin{tabular}{@{}l@{}}
332 \ducksay_open_bubble:
333 \ducksay_print_msg:nV { #1 } \l_ducksay_msg_align_tl
334 \ducksay_close_bubble:

```

```

( 25 )
\ .-'\
"
\
,-----'-----

```

```

335     \end{tabular}\\
336     \l_ducksay_body_tl
337     \begin{tabular}{@{}l@{}}
338         \l_ducksay_animal_tl
339     \end{tabular}
340 \end{tabular}
341 \group_end:
342 }
343 \cs_generate_variant:Nn \ducksay_print:nn { nV }

```

(End of definition for \ducksay_print:nn.)

\ducksay_say_and_think:nn Reset some variables

```

344 \cs_new:Npn \ducksay_say_and_think:nn #1 #2
345 {
346     \group_begin:
347     \int_set:Nn \l_ducksay_msg_width_int { -\c_max_int }
348     \int_set:Nn \l_ducksay_msg_height_int { -\c_max_int }
349     \keys_set:nn { ducksay } { #1 }
350     \ducksay_default_or_random_animal:
351     \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
352 \group_end:
353 }

```

(End of definition for \ducksay_say_and_think:nn.)

3.2.1.2 Document level

\ducksay

```

354 \NewDocumentCommand \ducksay { 0{ } m }
355 {
356     \ducksay_say_and_think:nn { #1 } { #2 }
357 }

```

(End of definition for \ducksay. This function is documented on page 8.)

\duckthink

```

358 \NewDocumentCommand \duckthink { 0{ } m }
359 {
360     \ducksay_say_and_think:nn { think, #1 } { #2 }
361 }

```

(End of definition for \duckthink. This function is documented on page 8.)

```

362 </code.v1>

```

3.3 Version 2

```

363 <*code.v2>
364 \ProvidesFile{ducksay.code.v2.tex}
365 [\ducksay@date\space v\ducksay@version\space ducksay code version 2]

Load the additional dependencies of version 2.
366 \RequirePackage{array,grabbox}

```

3.3.1 Messages

```

367 \msg_new:nnn { ducksay } { justify~unavailable }
368 {
369   Justified~content~is~not~available~for~tabular~argument~mode~without~fixed~
370   width.~'l'~column~is~used~instead.
371 }
372 \msg_new:nnn { ducksay } { unknown~message~alignment }
373 {
374   The~specified~message~alignment~'\exp_not:n { #1 }'~is~unknown.~
375   'l'~is~used~as~fallback.
376 }
377 \msg_new:nnn { ducksay } { v1~key~only }
378 { The~'\l_keys_key_tl'~key~is~only~available~for~'version=1'. }
379 \msg_new:nnn { ducksay } { zero~baselineskip }
380 { Current~ baselineskip~ is~ 0pt. }

```

3.3.2 Variables

3.3.2.1 Token Lists

```

381 \tl_new:N \l_ducksay_msg_align_vbox_tl

```

3.3.2.2 Boxes

```

382 \box_new:N \l_ducksay_msg_box

```

3.3.2.3 Bools

```

383 \bool_new:N \l_ducksay_eat_arg_box_bool
384 \bool_new:N \l_ducksay_eat_arg_tab_verb_bool
385 \bool_new:N \l_ducksay_mirrored_body_bool
386 \bool_new:N \l_ducksay_msg_eq_body_width_bool

```

3.3.2.4 Coffins

```

387 \coffin_new:N \l_ducksay_body_coffin
388 \coffin_new:N \l_ducksay_bubble_close_coffin
389 \coffin_new:N \l_ducksay_bubble_open_coffin
390 \coffin_new:N \l_ducksay_bubble_top_coffin
391 \coffin_new:N \l_ducksay_msg_coffin

```

3.3.2.5 Dimensions

```

392 \dim_new:N \l_ducksay_hpad_dim
393 \dim_new:N \l_ducksay_bubble_bottom_kern_dim
394 \dim_new:N \l_ducksay_bubble_top_kern_dim
395 \dim_new:N \l_ducksay_msg_width_dim

```

3.3.3 Options

```

396 \keys_define:nn { ducksay }
397 {
398   ,arg .choice:

```

```

( 27 )
\ .-'\
"
/ (
,

```

```

399 ,arg / box .code:n = \bool_set_true:N \l_ducksay_eat_arg_box_bool
400 ,arg / tab .code:n =
401 {
402   \bool_set_false:N \l_ducksay_eat_arg_box_bool
403   \bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool
404 }
405 ,arg / tab* .code:n =
406 {
407   \bool_set_false:N \l_ducksay_eat_arg_box_bool
408   \bool_set_true:N \l_ducksay_eat_arg_tab_verb_bool
409 }
410 ,arg .initial:n = tab
411 ,wd* .dim_set:N = \l_ducksay_msg_width_dim
412 ,wd* .initial:n = -\c_max_dim
413 ,wd* .value_required:n = true
414 ,wd-eq-body .bool_set:N = \l_ducksay_msg_eq_body_width_bool
415 ,none .bool_set:N = \l_ducksay_no_body_bool
416 ,no-bubble .bool_set:N = \l_ducksay_no_bubble_bool
417 ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
418 ,ignore-body .bool_set:N = \l_ducksay_ignored_body_bool
419 ,body-x .dim_set:N = \l_ducksay_body_x_offset_dim
420 ,body-x .value_required:n = true
421 ,body-y .dim_set:N = \l_ducksay_body_y_offset_dim
422 ,body-y .value_required:n = true
423 ,body-to-msg .tl_set:N = \l_ducksay_body_to_msg_align_body_tl
424 ,msg-to-body .tl_set:N = \l_ducksay_body_to_msg_align_msg_tl
425 ,body-align .choice:
426 ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
427 ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
428 ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
429 ,body-align .initial:n = l
430 ,body-bigger .int_set:N = \l_ducksay_body_bigger_int
431 ,body-bigger .initial:n = \c_zero_int
432 ,msg-align .choice:
433 ,msg-align / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
434 ,msg-align / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
435 ,msg-align / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
436 ,msg-align / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
437 ,msg-align-l .tl_set:N = \l_ducksay_msg_align_l_tl
438 ,msg-align-l .initial:n = \raggedright
439 ,msg-align-c .tl_set:N = \l_ducksay_msg_align_c_tl
440 ,msg-align-c .initial:n = \centering
441 ,msg-align-r .tl_set:N = \l_ducksay_msg_align_r_tl
442 ,msg-align-r .initial:n = \raggedleft
443 ,msg-align-j .tl_set:N = \l_ducksay_msg_align_j_tl
444 ,msg-align-j .initial:n = {}
445 ,out-h .tl_set:N = \l_ducksay_output_h_pole_tl
446 ,out-h .initial:n = l
447 ,out-v .tl_set:N = \l_ducksay_output_v_pole_tl
448 ,out-v .initial:n = vc
449 ,out-x .dim_set:N = \l_ducksay_output_x_offset_dim
450 ,out-x .value_required:n = true
451 ,out-y .dim_set:N = \l_ducksay_output_y_offset_dim
452 ,out-y .value_required:n = true

```

```

453 ,t      .meta:n    = { out-v = t }
454 ,c      .meta:n    = { out-v = vc }
455 ,b      .meta:n    = { out-v = b }
456 ,body*  .tl_set:N  = \l_ducksay_body_fount_tl
457 ,msg*   .tl_set:N  = \l_ducksay_msg_fount_tl
458 ,bubble* .tl_set:N  = \l_ducksay_bubble_fount_tl
459 ,body*  .initial:n  = \verbatim@font
460 ,msg*   .initial:n  = \verbatim@font
461 ,bubble* .initial:n  = \verbatim@font
462 ,body   .code:n    = \tl_put_right:Nn \l_ducksay_body_fount_tl { #1 }
463 ,msg    .code:n    = \tl_put_right:Nn \l_ducksay_msg_fount_tl { #1 }
464 ,bubble .code:n    = \tl_put_right:Nn \l_ducksay_bubble_fount_tl { #1 }
465 ,MSG    .meta:n    = { msg = #1 , bubble = #1 }
466 ,MSG*   .meta:n    = { msg* = #1 , bubble* = #1 }
467 ,hpad   .int_set:N  = \l_ducksay_hpad_int
468 ,hpad   .initial:n  = 2
469 ,hpad   .value_required:n = true
470 ,vpad   .int_set:N  = \l_ducksay_vpad_int
471 ,vpad   .value_required:n = true
472 ,col    .tl_set:N  = \l_ducksay_msg_tabular_column_tl
473 ,bubble-top-kern .tl_set:N  = \l_ducksay_bubble_top_kern_tl
474 ,bubble-top-kern .initial:n  = { -.5ex }
475 ,bubble-top-kern .value_required:n = true
476 ,bubble-bot-kern .tl_set:N  = \l_ducksay_bubble_bottom_kern_tl
477 ,bubble-bot-kern .initial:n  = { .2ex }
478 ,bubble-bot-kern .value_required:n = true
479 ,bubble-side-kern .tl_set:N  = \l_ducksay_bubble_side_kern_tl
480 ,bubble-side-kern .initial:n  = { .2em }
481 ,bubble-side-kern .value_required:n = true
482 ,bubble-delim-top .tl_set:N  = \l_ducksay_bubble_delim_top_tl
483 ,bubble-delim-left-1 .tl_set:N  = \l_ducksay_bubble_delim_left_a_tl
484 ,bubble-delim-left-2 .tl_set:N  = \l_ducksay_bubble_delim_left_b_tl
485 ,bubble-delim-left-3 .tl_set:N  = \l_ducksay_bubble_delim_left_c_tl
486 ,bubble-delim-left-4 .tl_set:N  = \l_ducksay_bubble_delim_left_d_tl
487 ,bubble-delim-right-1 .tl_set:N  = \l_ducksay_bubble_delim_right_a_tl
488 ,bubble-delim-right-2 .tl_set:N  = \l_ducksay_bubble_delim_right_b_tl
489 ,bubble-delim-right-3 .tl_set:N  = \l_ducksay_bubble_delim_right_c_tl
490 ,bubble-delim-right-4 .tl_set:N  = \l_ducksay_bubble_delim_right_d_tl
491 ,bubble-delim-top .initial:n  = { { - } }
492 ,bubble-delim-left-1 .initial:n  = (
493 ,bubble-delim-left-2 .initial:n  = /
494 ,bubble-delim-left-3 .initial:n  = |
495 ,bubble-delim-left-4 .initial:n  = \c_backslash_str
496 ,bubble-delim-right-1 .initial:n  = )
497 ,bubble-delim-right-2 .initial:n  = \c_backslash_str
498 ,bubble-delim-right-3 .initial:n  = |
499 ,bubble-delim-right-4 .initial:n  = /
500 ,strip-spaces .bool_set:N  = \l_ducksay_msg_strip_spaces_bool
501 }

```

Redefine keys only intended for version 1 to throw an error:

```

502 \clist_map_inline:nn
503 { align, rel-align }
504 {
505   \keys_define:nn { ducksay }

```

```

506     { #1 .code:n = \msg_error:nn { ducksay } { v1-key-only } }
507   }

```

3.3.4 Functions

3.3.4.1 Internal

luate_message_alignment_fixed_width_common:

```

508 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:
509   {
510     \str_case:Vn \l_ducksay_msg_align_tl
511     {
512       { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
513       { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
514       { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
515       { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
516     }
517   }

```

(End of definition for \ducksay_evaluate_message_alignment_fixed_width_common:.)

uate_message_alignment_fixed_width_tabular:

```

518 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_tabular:
519   {
520     \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
521     {
522       \tl_set:Nx \l_ducksay_msg_tabular_column_tl
523       {
524         >
525         {
526           \ducksay_evaluate_message_alignment_fixed_width_common:
527           \exp_not:N \arraybackslash
528         }
529         p { \exp_not:N \l_ducksay_msg_width_dim }
530       }
531     }
532   }

```

(End of definition for \ducksay_evaluate_message_alignment_fixed_width_tabular:.)

valuate_message_alignment_fixed_width_vbox:

```

533 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_vbox:
534   {
535     \tl_set:Nx \l_ducksay_msg_align_vbox_tl
536     { \ducksay_evaluate_message_alignment_fixed_width_common: }
537   }

```

(End of definition for \ducksay_evaluate_message_alignment_fixed_width_vbox:.)

\ducksay_calculate_msg_width_from_int:

```

538 \cs_new:Npn \ducksay_calculate_msg_width_from_int:
539   {
540     \hbox_set:Nn \l_ducksay_tmpa_box { { \l_ducksay_msg_fount_tl M } }
541     \dim_set:Nn \l_ducksay_msg_width_dim
542     { \l_ducksay_msg_width_int \box_wd:N \l_ducksay_tmpa_box }
543   }

```

```

(-----)
( 30 )
\ .-'√\
"
\
,-----)
,-----)

```

(End of definition for \ducksay_calculate_msg_width_from_int:.)

\ducksay_msg_tabular_begin:

```

544 \cs_new:Npn \ducksay_msg_tabular_begin:
545   {
546     \ducksay_msg_tabular_begin_inner:V \l_ducksay_msg_tabular_column_tl
547   }
548 \cs_new:Npn \ducksay_msg_tabular_begin_inner:n #1
549   {
550     \begin { tabular } { @{} #1 @{} }
551   }
552 \cs_generate_variant:Nn \ducksay_msg_tabular_begin_inner:n { V }

```

(End of definition for \ducksay_msg_tabular_begin:.)

\ducksay_msg_tabular_end:

```

553 \cs_new:Npn \ducksay_msg_tabular_end:
554   {
555     \end { tabular }
556   }

```

(End of definition for \ducksay_msg_tabular_end:.)

\ducksay_width_case_none_int_dim:nnn

```

557 \cs_new:Npn \ducksay_width_case_none_int_dim:nnn #1 #2 #3
558   {
559     \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
560     {
561       \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
562       { #1 }
563       { #2 }
564     }
565     { #3 }
566   }

```

(End of definition for \ducksay_width_case_none_int_dim:nnn.)

\ducksay_digest_options:n

```

567 \cs_new:Npn \ducksay_digest_options:n #1
568   {
569     \group_begin:
570     \keys_set:nn { ducksay } { #1 }
571     \ducksay_default_or_random_animal:
572     \bool_if:NF \l_ducksay_no_body_bool
573     {
574       \hcoffin_set:Nn \l_ducksay_body_coffin
575       {
576         \frenchspacing
577         \l_ducksay_body_fount_tl
578         \begin{tabular} { @{} l @{} }
579           \l_ducksay_animal_tl
580           \ducksay_make_body_bigger:
581           \relax
582         \end{tabular}
583       }

```

(31)

```

\ .-'\ \
"
/ \ ( '
,-----'-----'

```



```

584     \bool_if:NT \l_ducksay_msg_eq_body_width_bool
585     {
586         \bool_lazy_and:nnT
587         { \int_compare_p:nNn \l_ducksay_msg_width_int < \c_zero_int }
588         { \dim_compare_p:nNn \l_ducksay_msg_width_dim < \c_zero_dim }
589         {
590             \dim_set:Nn \l_ducksay_msg_width_dim
591             { \coffin_wd:N \l_ducksay_body_coffin }
592         }
593     }
594 }
595 \bool_if:NTF \l_ducksay_eat_arg_box_bool
596 {
597     \ducksay_width_case_none_int_dim:nnn
598     { \ducksay_eat_argument_hbox:w }
599     {
600         \ducksay_calculate_msg_width_from_int:
601         \ducksay_eat_argument_vbox:w
602     }
603     { \ducksay_eat_argument_vbox:w }
604 }
605 {
606     \ducksay_width_case_none_int_dim:nnn
607     {
608         \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
609         {
610             \str_case:Vn \l_ducksay_msg_align_tl
611             {
612                 { l } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
613                 { c } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
614                 { r } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
615                 { j }
616                 {
617                     \msg_error:nn { ducksay } { justify~unavailable }
618                     \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
619                 }
620             }
621         }
622     }
623     {
624         \ducksay_calculate_msg_width_from_int:
625         \ducksay_evaluate_message_alignment_fixed_width_tabular:
626     }
627     { \ducksay_evaluate_message_alignment_fixed_width_tabular: }
628     \ducksay_eat_argument_tabular:w
629 }
630 }

```

(End of definition for \ducksay_digest_options:n.)

\ducksay_set_bubble_top_kern:

```

631 \cs_new:Npn \ducksay_set_bubble_top_kern:
632 {
633     \group_begin:

```

```

-----
( 32 )
-----
\ .-'\
"
-----
-----

```

```

634 \l_ducksay_bubble_fount_tl
635 \exp_args:NNNx
636 \group_end:
637 \dim_set:Nn \l_ducksay_bubble_top_kern_dim
638   { \dim_eval:n { \l_ducksay_bubble_top_kern_tl } }
639 }

```

(End of definition for \ducksay_set_bubble_top_kern:.)

\ducksay_set_bubble_bottom_kern:

```

640 \cs_new:Npn \ducksay_set_bubble_bottom_kern:
641   {
642     \group_begin:
643     \l_ducksay_bubble_fount_tl
644     \exp_args:NNNx
645     \group_end:
646     \dim_set:Nn \l_ducksay_bubble_bottom_kern_dim
647       { \dim_eval:n { \l_ducksay_bubble_bottom_kern_tl } }
648   }

```

(End of definition for \ducksay_set_bubble_bottom_kern:.)

\ducksay_make_body_bigger:

```

649 \cs_new:Npn \ducksay_make_body_bigger:
650   { \prg_replicate:nn \l_ducksay_body_bigger_int \ \ }

```

(End of definition for \ducksay_make_body_bigger:.)

\ducksay_baselineskip: This is an overly cautious way to get the current baselineskip. Inside of tabular the baselineskip is Opt, so we fall back to \normalbaselineskip, or issue an error and fall back to some arbitrary value not producing an error if that one is also Opt.

```

651 \cs_new_protected_nopar:Npn \ducksay_baselineskip:
652   {
653     \the\dimexpr
654     \ifdim \baselineskip = \c_zero_dim
655       \ifdim \normalbaselineskip = \c_zero_dim
656         \msg_expandable_error:nn { ducksay } { zero-baselineskip } { 12pt }
657         12pt
658       \else
659         \normalbaselineskip
660       \fi
661     \else
662       \baselineskip
663     \fi
664     \relax
665   }

```

(End of definition for \ducksay_baselineskip:.)

\ducksay_measure_msg:

```

666 \cs_new_protected_nopar:Npn \ducksay_measure_msg:
667   {
668     \hbox_set:Nn \l_ducksay_tmpa_box
669       { \l_ducksay_bubble_fount_tl \l_ducksay_bubble_delim_top_tl }
670     \int_set:Nn \l_ducksay_msg_width_int

```

```

671     {
672       \fp_eval:n
673       {
674         ceil
675         ( \box_wd:N \l_ducksay_msg_box / \box_wd:N \l_ducksay_tmpa_box )
676       }
677     }
678   \group_begin:
679   \l_ducksay_bubble_fount_tl
680   \exp_args:NNNx
681   \group_end:
682   \int_set:Nn \l_ducksay_msg_height_int
683   {
684     \int_max:nn
685     {
686       \fp_eval:n
687       {
688         ceil
689         (
690           (
691             \box_ht:N \l_ducksay_msg_box
692             + \box_dp:N \l_ducksay_msg_box
693           )
694           / ( \arraystretch * \ducksay_baselineskip: )
695         )
696       }
697       + \l_ducksay_vpad_int
698     }
699     { \l_ducksay_msg_height_int }
700   }
701 }

```

(End of definition for \ducksay_measure_msg:.)

\ducksay_set_bubble_coffins:

```

702 \cs_new_protected_nopar:Npn \ducksay_set_bubble_coffins:
703 {
704   \hcoffin_set:Nn \l_ducksay_bubble_open_coffin
705   {
706     \l_ducksay_bubble_fount_tl
707     \begin{tabular}{@{}l@{}}
708       \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
709       {
710         \l_ducksay_bubble_delim_left_a_tl
711       }
712       {
713         \l_ducksay_bubble_delim_left_b_tl\
714         \int_step_inline:nnn
715         { 3 } { \l_ducksay_msg_height_int }
716         {
717           \kern-\l_ducksay_bubble_side_kern_tl
718           \l_ducksay_bubble_delim_left_c_tl
719         }
720       }

```

```

721         \l_ducksay_bubble_delim_left_d_tl
722     }
723     \end{tabular}
724 }
725 \hcoffin_set:Nn \l_ducksay_bubble_close_coffin
726 {
727     \l_ducksay_bubble_fount_tl
728     \begin{tabular}{@{}r@{}}
729         \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
730     {
731         \l_ducksay_bubble_delim_right_a_tl
732     }
733     {
734         \l_ducksay_bubble_delim_right_b_tl \\
735         \int_step_inline:nnn
736             { 3 } { \l_ducksay_msg_height_int }
737         {
738             \l_ducksay_bubble_delim_right_c_tl
739             \kern-\l_ducksay_bubble_side_kern_tl
740             \\
741         }
742         \l_ducksay_bubble_delim_right_d_tl
743     }
744     \end{tabular}
745 }
746 \hcoffin_set:Nn \l_ducksay_bubble_top_coffin
747 {
748     \l_ducksay_bubble_fount_tl
749     \int_step_inline:nn
750         { \l_ducksay_msg_width_int + \l_ducksay_hpad_int }
751         { \l_ducksay_bubble_delim_top_tl }
752 }
753 }

```

(End of definition for \ducksay_set_bubble_coffins:.)

\ducksay_join_bubble_to_msg_coffin:

```

754 \cs_new_protected_nopar:Npn \ducksay_join_bubble_to_msg_coffin:
755 {
756     \dim_set:Nn \l_ducksay_hpad_dim
757     {
758         (
759             \coffin_wd:N \l_ducksay_bubble_top_coffin
760             - \coffin_wd:N \l_ducksay_msg_coffin
761         ) / 2
762     }
763     \coffin_join:NnnNnnnn
764         \l_ducksay_msg_coffin          { l } { vc }
765         \l_ducksay_bubble_open_coffin { r } { vc }
766         { - \l_ducksay_hpad_dim } { \c_zero_dim }
767     \coffin_join:NnnNnnnn
768         \l_ducksay_msg_coffin          { r } { vc }
769         \l_ducksay_bubble_close_coffin { l } { vc }
770         { \l_ducksay_hpad_dim } { \c_zero_dim }

```

```

(-----)
( 35 )
\ .-'\
"
\ /
,-----)

```

```

771 \coffin_join:NnnNnnnn
772   \l_ducksay_msg_coffin      { hc } { t }
773   \l_ducksay_bubble_top_coffin { hc } { b }
774   { \c_zero_dim } { \l_ducksay_bubble_top_kern_dim }
775 \coffin_join:NnnNnnnn
776   \l_ducksay_msg_coffin      { hc } { b }
777   \l_ducksay_bubble_top_coffin { hc } { t }
778   { \c_zero_dim } { \l_ducksay_bubble_bottom_kern_dim }
779 }

```

(End of definition for \ducksay_join_bubble_to_msg_coffin:.)

\ducksay_shipout:

```

780 \cs_new_protected:Npn \ducksay_shipout:
781 {
782   \hcoffin_set:Nn \l_ducksay_msg_coffin { \box_use:N \l_ducksay_msg_box }
783   \bool_if:NF \l_ducksay_no_bubble_bool
784   {
785     \ducksay_measure_msg:
786     \ducksay_set_bubble_coffins:
787     \ducksay_set_bubble_top_kern:
788     \ducksay_set_bubble_bottom_kern:
789     \ducksay_join_bubble_to_msg_coffin:
790   }
791   \bool_if:NF \l_ducksay_no_body_bool
792   {
793     \bool_if:NT \l_ducksay_mirrored_body_bool
794     {
795       \coffin_scale:Nnn \l_ducksay_body_coffin
796       { -\c_one_int } { \c_one_int }
797       \str_case:Vn \l_ducksay_body_to_msg_align_body_tl
798       {
799         { l } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { r } }
800         { r } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { l } }
801       }
802     }
803     \bool_if:NTF \l_ducksay_ignored_body_bool
804     { \coffin_attach:NVnNVnnn }
805     { \coffin_join:NVnNVnnn }
806     \l_ducksay_msg_coffin \l_ducksay_body_to_msg_align_msg_tl { b }
807     \l_ducksay_body_coffin \l_ducksay_body_to_msg_align_body_tl { t }
808     { \l_ducksay_body_x_offset_dim } { \l_ducksay_body_y_offset_dim }
809   }
810   \coffin_typeset:NVVnn \l_ducksay_msg_coffin
811   \l_ducksay_output_h_pole_tl \l_ducksay_output_v_pole_tl
812   { \l_ducksay_output_x_offset_dim } { \l_ducksay_output_y_offset_dim }
813   \group_end:
814 }

```

(End of definition for \ducksay_shipout:.)

3.3.4.1.1 Message Reading Functions Version 2 has different ways of reading the message argument of \ducksay and \duckthink. They all should allow almost arbitrary content and the height and width are set based on the dimensions.

```

-----
( 36 )
-----
\ .-'\
"
-----
-----

```

\ducksay_eat_argument_tabular:w

```

815 \cs_new:Npn \ducksay_eat_argument_tabular:w
816 {
817   \bool_if:NTF \l_ducksay_eat_arg_tab_verb_bool
818     { \ducksay_eat_argument_tabular_verb:w }
819     { \ducksay_eat_argument_tabular_normal:w }
820 }

```

(End of definition for \ducksay_eat_argument_tabular:w.)

\ducksay_eat_argument_tabular_inner:w

```

821 \cs_new:Npn \ducksay_eat_argument_tabular_inner:w #1
822 {
823   \hbox_set:Nn \l_ducksay_msg_box
824     {
825       \l_ducksay_msg_fount_tl
826       \ducksay_msg_tabular_begin:
827         #1
828       \ducksay_msg_tabular_end:
829     }
830   \ducksay_shipout:
831 }

```

(End of definition for \ducksay_eat_argument_tabular_inner:w.)

\ducksay_eat_argument_tabular_verb:w

```

832 \NewDocumentCommand \ducksay_eat_argument_tabular_verb:w
833 { >{ \ducksay_process_verb_newline:nnn { ~ } { ~ \par } } +v }
834 {
835   \ducksay_eat_argument_tabular_inner:w
836   {
837     \group_begin:
838     \__ducksay_everyeof:w { \exp_not:N }
839     \exp_after:wN
840     \group_end:
841     \__ducksay_scantokens:w { #1 }
842   }
843 }

```

(End of definition for \ducksay_eat_argument_tabular_verb:w.)

\ducksay_eat_argument_tabular_normal:w

```

844 \NewDocumentCommand \ducksay_eat_argument_tabular_normal:w { +m }
845 { \ducksay_eat_argument_tabular_inner:w { #1 } }

```

(End of definition for \ducksay_eat_argument_tabular_normal:w.)

\ducksay_eat_argument_hbox:w

```

846 \cs_new_protected_nopar:Npn \ducksay_eat_argument_hbox:w
847 {
848   \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
849     { \@grabbox }
850     { \@grabbox* }
851   { } \l_ducksay_msg_box \l_ducksay_msg_fount_tl \hbox {} \ducksay_shipout:
852 }

```

(End of definition for \ducksay_eat_argument_hbox:w.)

\ducksay_eat_argument_vbox:w

```

853 \cs_new_protected_nopar:Npn \ducksay_eat_argument_vbox:w
854 {
855   \ducksay_evaluate_message_alignment_fixed_width_vbox:
856   \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
857     { \@grabbox }
858     { \@grabbox* }
859     {
860       \hspace \l_ducksay_msg_width_dim
861       \linewidth \hspace
862       \l_ducksay_msg_align_vbox_tl
863       \@afterindentfalse
864       \@afterheading
865     }
866   \l_ducksay_msg_box \l_ducksay_msg_fount_tl \vbox {} \ducksay_shipout:
867 }

```

(End of definition for \ducksay_eat_argument_vbox:w.)

3.3.4.1.2 Generating Variants of External Functions

```

868 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
869 \cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
870 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
871 \cs_generate_variant:Nn \str_case:nn { Vn }

```

3.3.4.2 Document level

\ducksay

```

872 \NewDocumentCommand \ducksay { 0{ } }
873 {
874   \ducksay_digest_options:n { #1 }
875 }

```

(End of definition for \ducksay. This function is documented on page 8.)

\duckthink

```

876 \NewDocumentCommand \duckthink { 0{ } }
877 {
878   \ducksay_digest_options:n { think, #1 }
879 }

```

(End of definition for \duckthink. This function is documented on page 8.)

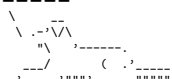
880 </code.v2>

3.4 Definition of the Animals

```

881 (*animals)
882 \ProvidesFile{ducksay.animals.tex}
883 [\ducksay@date\space v\ducksay@version\space ducksay animals]
884 %^A some of the below are from http://ascii.co.uk/art/
885 \AddAnimal{duck}%>>=
886 { \
887   \
888     >(')
889     )/
890     /(
891     / '----/
892     \ ~-- /
893     ~~~~~~}%<<
894 \AddAnimal{small-duck}%>>=
895 { \
896   \
897     >()_
898     (__)__ _}%<<
899 \AddAnimal{duck-family}%>>=
900 { \
901   \
902     >(')
903     )/
904     /(
905     / '----/ -()_ >()_
906     __\__~=-/_/_ _()_ _()_ __ _}%<<
907 \AddAnimal{cow}%>>=
908 { \
909   \ ^--^
910     \ (oo)\_____
911     (__)\_____)\ \
912     ||----w |
913     ||    ||}%<<
914 \AddAnimal{head-in}%>>=
915 { \
916   \ ^--^
917     \ (oo)\_____ /
918     (__)\_____ )=( _____ | \_____
919     ||----w | \ \ \ \ \_____ |
920     ||    ||    ||    || }%<<
921 \AddAnimal{sodomized}%>>=
922 { \
923   \
924     \ ^--^
925     \ (oo)\_____ / \
926     (__)\_____ ) /
927     ||----w ((
928     ||    ||>>}%<<
929 \AddAnimal{tux}%>>=
930 { \
931   \ .--.
932     |o_o |
933     |\_ / |

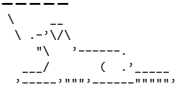
```




```

987 \AddAnimal{bunny}%>>=
988 { \
989     \      /
990     /\ /
991     ( )
992     .( o ).}%<<=
993 \AddAnimal{small-rabbit}%>>=
994 { \
995     \ _//
996     (')---.
997     _/-( )o}%<<=
998 \AddAnimal[tail-symbol=s,tail-count=3]{dragon}%>>=
999 {
1000     s      | \_ _ / |      / \ \ // \
1001     s /0 0 \_ _ /      // | \ \
1002     / _ _ / \ \_ /      // | \ \
1003     @_~@' / \_ _ //      | \ \
1004     //_~_ / \_ _ //      | \ \
1005     ( // ) |      \ //      | \ \
1006     ( / / ) _ | _ / ) //      | \ \
1007     ( // / ) ' / , _ _ / ( ; - .      | \ \
1008     (( // / )) , - {      | \ \
1009     (( // / )) ' \ /      | \ \
1010     (( // / )) ' . {      | \ \
1011     (( / ))      .----- \      | \ \
1012     ///.-----..>      | \ \
1013     ///-.-.-.-.-}^-----~
1014     /.-~}%<<=
1015 %^A http://www.ascii-art.de/ascii/def/dogs.txt
1016 \AddAnimal{dog}%>>=
1017 { \
1018     \ .-' \ \
1019     " \ '-----.
1020     _ _ / ( ' '-----
1021     ,-----, " " " ,-----" " " " ,}%<<=
1022 %^A http://ascii.co.uk/art/squirrel
1023 \AddAnimal{squirrel}%>>=
1024 { \
1025     \      , ;;;;,
1026     .=' , ;;;;,
1027     /_ ' , "= . ' ;;;;,
1028     @=: _ _ , \ , ;;;;, '
1029     _ (\ . = ;;;;, '
1030     " _ ( _ / = " '
1031     ' " ' , ' } %<<=
1032 \AddAnimal{snail}%>>=
1033 { \
1034     \      .-"-."
1035     oo      ; .- . :
1036     \ \ _ _ . . : ' . _ _ ' ) . _
1037     "- . _ _ ' . _ _ - ' _ _ " } %<<=
1038 %^A http://www.ascii-art.de/ascii/uvw/unicorn.txt
1039 \AddAnimal{unicorn}%>>=
1040 { \

```




```

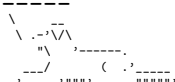
1149      |          ||          |
1150      |-----||-----|
1151      |/  ----- \  ----- \|
1152      /  (      )  (      )  \
1153      / \  ----- ( ) ----- / \
1154      / \          /||\
1155      / \          /||||\
1156      / \          /||||||\
1157 /_  \          \0=====0/  \_
1158 '---..._|'-'-'-'-'|_---...-'
1159 |      '      |}%=<<
1160 \AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>>=
1161 { |
1162 \[T]/}
1163 \csname bool_if:cT\endcsname {l_ducksay_version_one_bool}
1164   {\AnimalOptions{crusader}{tail-1=|,rel-align=c}}
1165 \csname bool_if:cT\endcsname {l_ducksay_version_two_bool}
1166   {\AnimalOptions{crusader}{tail-1=|,body-align=c}}%=<<
1167 %^A http://ascii.co.uk/art/knights
1168 \AddAnimal[tail-count=3]{knight}%>>=
1169 {
1170   \ ,-" "-.
1171   \ | === |
1172   ) | (
1173   .==\' " "/\'==.
1174   .' \ (\' :') /\' .
1175   _/_ |_.-' : '-._|_ \_
1176   <--->\' : / \'<--->
1177   / / >===== < / /
1178   _/_ .' / ,--:-- \/=,'
1179   /_/_ |_/v^v^v^v\_) \
1180   \(\) |v^v^v^v^v\| \_/
1181   (\ \ \ \'---|---'/
1182   \ \ \ \ \ -._|_,-/
1183   \ \ \ \ \ |__|__|
1184   \ \ \ \ \ <__X__>
1185   \ \ \ \ \ \..|..|
1186   \ \ \ \ \ \ | /
1187   \ \ \ \ \ /v|v\
1188   \ / \ | \
1189   ,--' '---'}%=<<
1190 %^A https://www.asciiart.eu/mythology/ghosts
1191 \AddAnimal{ghost}%>>=
1192 {
1193   \ .-.
1194   (o o)
1195   | 0 \
1196   \ \ \
1197   '~~~'}%=<<
1198 %^A https://asciiart.website/index.php?art=creatures/fairies
1199 \AddAnimal{fairy}%>>=
1200 {
1201   \ .o00b
1202   .. .o0 0

```

```

1203     ';;; d      0
1204     ;;;;d    ..o0
1205 *   :;0;;;'0oo0
1206 ~"\. dp'(0.o.
1207   \op   'o0b
1208         obU
1209         dop
1210         dop
1211         P0
1212         0 'b
1213         1 P.
1214         / ;
1215         '}%=<<<
1216 \AddAnimal[tail-symbol=s]{only-tail}%>>=
1217 { s
1218   s}%=<<<
1219 \AddAnimal[tail-symbol=s,tail-count=3]{only-tail3}%>>=
1220 { s
1221   s
1222   s}%=<<<
1223 %^A head taken from https://www.asciiart.eu/animals/reptiles/snakes
1224 \AddAnimal[tail-symbol=s,tail-count=3]{snake}% >>=
1225 { s
1226   s   /\~/^
1227   s _|_|| 0 |
1228   /'  \_/ \
1229 \ / |-----/ \
1230   \_/ \----- \
1231         ' | |
1232         / / _--- | |
1233         / / / _- " _ " |
1234         | " _ " / " _ " _ " |
1235         " _ _ _ " " _ _ _ " }% =<<<
1236 %^A http://www.ascii-art.de/ascii/c/cat.txt
1237 \AddAnimal{cat}% >>=
1238 + \
1239   \
1240   \'.|\..-----.' ' ' _ _ _ ' _ _
1241   / ' ' ' ' ' ' ' ' ' ' ' ' ' ' '
1242   )/' _/ \ \ ' _ _ /
1243   ' _ " " \ _ , _ . ; _ - \ _ ' ,
1244   _ . - ' _ / { _ . ' ; /
1245   { _ . - ' ' - ' ' _ / + % =<<<
1246 %^A https://www.asciiart.eu/animals/cats
1247 \AddAnimal{sleazy-cat}% >>=
1248 { \
1249   \ | \
1250   / , ' _ ' ' ' _ . ' ) ' _ , ' _ , )
1251   | , 4 - ) ) - , _ , \ ( ' - , - '
1252   , _ _ _ ' ( _ / - - ' ' - ' \ _ ) }% =<<<
1253 \AddAnimal{schroedinger-dead}% >>=
1254 { \
1255   \ _ . - - " " - - _
1256   |

```




```
1311 /_/'-\_\\=====\\_\'
1312 "" "" "" ""}
1313 </animals>
```


Who's gonna use it anyway?

```
0
  o
    --
  >(' )
    )/
  /(
  / '-----/
  \ ~=- /
~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^
```

Hosted at
https://github.com/Skillmon/ltx_ducksay
it is.

```
--'---
'-'-'7'
/'-c
| /T
_)_/_LI
```