

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

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

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

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

-----  
( v2.5 )  
-----

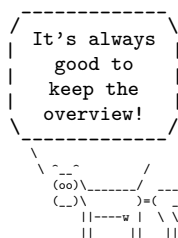
^ ^  
 / (oo) /  
 / (\_\_)  
 / \ (   
 | w----| |  
 | | | |

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

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

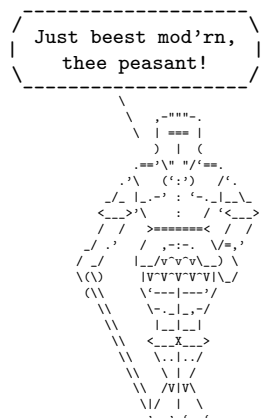
-----  
( Today is 2020-02-01 )  
-----

\ .\|//| | | |  
 \ | / | | / | | / |  
 / . ' | / \ | / | | |  
 o\_\_ \_ | / | / | | | |



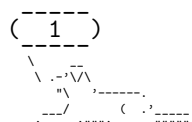
# Contents

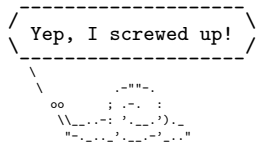
|          |                                          |           |
|----------|------------------------------------------|-----------|
| <b>1</b> | <b>Documentation</b>                     | <b>1</b>  |
| 1.1      | Downward Compatibility Issues            | 2         |
| 1.2      | Shared between versions                  | 2         |
| 1.2.1    | Macros                                   | 2         |
| 1.2.2    | Options                                  | 3         |
| 1.3      | Version 1                                | 6         |
| 1.3.1    | Introductkion                            | 6         |
| 1.3.2    | Macros                                   | 6         |
| 1.3.3    | Options                                  | 6         |
| 1.3.4    | Defects                                  | 7         |
| 1.4      | Version 2                                | 8         |
| 1.4.1    | Introductkion                            | 8         |
| 1.4.2    | Macros                                   | 8         |
| 1.4.3    | Options                                  | 8         |
| 1.5      | Dependencies                             | 14        |
| 1.6      | Available Animals                        | 14        |
| 1.7      | License and Bug Reports                  | 16        |
| <b>2</b> | <b>Implementation</b>                    | <b>17</b> |
| 2.1      | Shared between versions                  | 17        |
| 2.1.1    | Variables                                | 17        |
| 2.1.2    | Regular Expressions                      | 17        |
| 2.1.3    | Messages                                 | 17        |
| 2.1.4    | Key-value setup                          | 17        |
| 2.1.5    | Functions                                | 19        |
| 2.1.6    | Load the Correct Version and the Animals | 22        |
| 2.2      | Version 1                                | 23        |
| 2.2.1    | Functions                                | 23        |
| 2.3      | Version 2                                | 26        |
| 2.3.1    | Messages                                 | 26        |
| 2.3.2    | Variables                                | 26        |
| 2.3.3    | Options                                  | 26        |
| 2.3.4    | Functions                                | 29        |
| 2.4      | Definition of the Animals                | 38        |



## 1 Documentation

This is ducksay! A cowsay for L<sup>A</sup>T<sub>E</sub>X. ducksay is part of T<sub>E</sub>XLive and MiK<sub>T</sub>E<sub>X</sub> since September 2017. If it is not part of your installation it means that your L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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 `grabbbox` for version 2 of `ducksay` that won't be part of your L<sup>A</sup>T<sub>E</sub>X installation, too. Please note that I don't actively support out of date L<sup>A</sup>T<sub>E</sub>X installations, so if loading `l3regex` doesn't fix the issues and you're on an old installation, I won't provide further support.

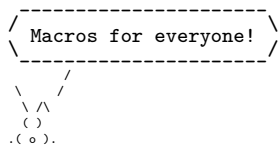




# 1.1 Downward Compatibility Issues

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 1.2.2](#) 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 1.2.1](#) 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 efficient now.

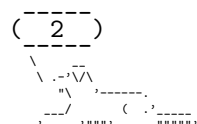


# 1.2 Shared between versions

## 1.2.1 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 1.2.2](#) for the code variants, [subsection 1.3.2](#) and [subsection 1.4.2](#) for descriptions of the two macros).

|                              |                                                                                                                                                                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\DefaultAnimal</code>  | <code>\DefaultAnimal{&lt;animal&gt;}</code><br>use the <code>&lt;animal&gt;</code> if none is given in the optional argument to <code>\ducksay</code> or <code>\duckthink</code> .<br>Package default is <code>duck</code> .                                           |
| <code>\DucksayOptions</code> | <code>\DucksayOptions{&lt;options&gt;}</code><br>set the defaults to the keys described in <a href="#">subsection 1.2.2</a> , <a href="#">subsection 1.3.3</a> and <a href="#">subsection 1.4.3</a> . Don’t use an <code>&lt;animal&gt;</code> 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 1.2.2.1](#) 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.

## 1.2.2 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 slightly different things. If that's the case they will be explained in [subsubsection 1.3.3](#) and [subsubsection 1.4.3](#) 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.

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

```
( 3 )
\ .-'\V\
" \
_/_/ ( .)
)-----)-----)-----)
```

`\animal` One of the animals listed in [subsection 1.6](#) 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.

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.

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 `'<>.'`.

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.

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.

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`.

### 1.2.2.1 Options for \AddAnimal

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.

( 4 )

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

### 1.3.1 Introduction

### 1.3.2 Macros

$$\backslash\text{ducksay}[\langle options \rangle]\{\langle message \rangle\}$$
$$\backslash\text{duckthink}[\langle options \rangle]\{\langle message \rangle\}$$

### 1.3.3 Options

```
\begin{tikzpicture}
\draw [dashed] (-1,-1) rectangle (1,1);
\node at (0,0) {Everyone likes options};
\end{tikzpicture}
```

`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= $\langle column \rangle$`   
 use  $\langle column \rangle$  for alignment of the bubble and the body. It should match a `tabular` column specifier. Default is 1.

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

**ht**= $\langle count \rangle$  you might explicitly set the height (the row count) of the  $\langle message \rangle$ . This only has an effect if you also specify **wd**.

### 1.3.4 Defects

```

{Ohh, no!}
  \ (.)_(.)
  / \_-----\
--\ ( ( ) ) /--
   \ / \_ / \_ /
   )_ / \ \ / \ \ (

```

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

```

( 7 )
  \ .-' \_
   " \_-----
   / \_ ( '-----
   /-----)-----)

```



## 1.4 Version 2

### 1.4.1 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.

### 1.4.2 Macros

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

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

options might include any of the options described in [subsection 1.2.2](#) and [subsection 1.4.3](#) 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 1.4.3](#).

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<sup>A</sup>T<sub>E</sub>X3'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.

### 1.4.3 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`.

```
( 8 )
\ .-'\
" \
- - - - - ( . )
- - - - -
```

**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.

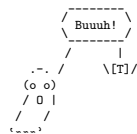
**b** shortcut for `out-v=b`.

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

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

`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]{Buuuh!}
\ducksay[crusader,body-bigger=4,b,out-h=r,no-bubble]{}
```

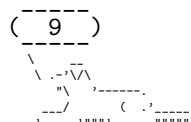
`body-mirrored=<bool>`  
if set true the `<animal>` will be mirrored along its vertical centre axis. Package default is `false`. If you set it `true` you'll most likely need to manually adjust the alignment of the body with one or more of the keys `body-align`, `body-to-msg`, `msg-to-body`, `body-x`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

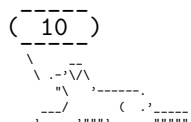
`body-x=<dimen>`  
defines a horizontal offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

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

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



- bubble\*= $\langle font \rangle$**   
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the bubble to  $\langle font \rangle$ . This does not affect the  $\langle message \rangle$  only the bubble put around it. The package default is `\verbatim@font`.
- bubble-bot-kern= $\langle dimen \rangle$**   
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= $\langle token list \rangle$**   
the left delimiter used if only one line of delimiters is needed. Package default is `(`.
- bubble-delim-left-2= $\langle token list \rangle$**   
the upper most left delimiter used if more than one line of delimiters is needed. Package default is `/`.
- bubble-delim-left-3= $\langle token list \rangle$**   
the left delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.
- bubble-delim-left-4= $\langle token list \rangle$**   
the lower most left delimiter used if more than one line of delimiters is needed. Package default is `\`.
- bubble-delim-right-1= $\langle token list \rangle$**   
the right delimiter used if only one line of delimiters is needed. Package default is `)`.
- bubble-delim-right-2= $\langle token list \rangle$**   
the upper most right delimiter used if more than one line of delimiters is needed. Package default is `\`.
- bubble-delim-right-3= $\langle token list \rangle$**   
the right delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.
- bubble-delim-right-4= $\langle token list \rangle$**   
the lower most right delimiter used if more than one line of delimiters is needed. Package default is `/`.
- bubble-delim-top= $\langle token list \rangle$**   
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= $\langle dimen \rangle$**   
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= $\langle dimen \rangle$**   
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`.



`col=<column>` specifies the used column specifier used for the `<message>` enclosing `tabular` for `arg=tab` and `arg=tab*`. Has precedence over `msg-align`. You can also use more than one column this way: `\ducksay[arg=tab,col=cc]{ You & can \\ do & it }` would be valid syntax.

`hpad=<count>` Add `<count>` times more `bubble-delim-top` instances than necessary to the upper and lower border of the bubble. Package default is 2.

`ht=<count>` specifies a minimum height (in lines) of the `<message>`. The lines' count is that of the needed lines of the horizontal bubble delimiters. If the count of the actually needed lines is smaller than the specified `<count>`, `<count>` lines will be used. Else the required lines will be used.

`ignore-body=<bool>` If set `true` the `<animal>`'s body will be added to the output but it will not contribute to the bounding box (so will not take up any space).

`msg=<font>` add `<font>` to the font definitions in use to typeset the `<message>`.

`msg*=<font>` clear any definitions previously made (including the package default) and set the font definitions in use to typeset the `<message>` to `<font>`. The package default is `\verbatim@font`.

`MSG=<font>` same as `msg=<font>`, `bubble=<font>`.

`MSG*=<font>` same as `msg*=<font>`, `bubble*=<font>`.

`msg-align=<choice>` specifies the alignment of the `<message>`. Possible values are `l` for flushed left, `c` for centred, `r` for flushed right and `j` for justified. If `arg=tab` or `arg=tab*` the `j` choice is only available for fixed width contents. Package default is `l`.

`msg-align-c=<token list>` set the `<token list>` which is responsible to typeset the message centred if the option `msg-align=c` 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 `\centering`. It might be useful if you want to use `ragged2e`'s `\Centering` for example.

`msg-align-j=<token list>` set the `<token list>` which is responsible to typeset the message justified if the option `msg-align=j` 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 empty as justification is the default behaviour of contents of a `p` column and of a `\vbox`. It might be useful if you want to use `ragged2e`'s `\justifying` for example.

`msg-align-l=<token list>` set the `<token list>` which is responsible to typeset the message flushed left if the option `msg-align=l` 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 `\raggedright`. It might be useful if you want to use `ragged2e`'s `\RaggedRight` for example.

$$\left( \begin{array}{c} \text{---} \\ 11 \\ \text{---} \end{array} \right)$$

- `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 )
-----
 \ .-'\ \
  " \   \
----- ( .) -----
) -----) HHHH) -----) HHHHHH)

```

`wd*= $\langle dimen \rangle$`  specifies the width of the  $\langle message \rangle$  to be fixed to  $\langle dimen \rangle$ . A value smaller than 0pt 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.

`wd-eq-body=<bool>`  
if this is **true**, `wd` is smaller than 0, and `wd*` is smaller than 0pt the `<message>` will be as wide as the `<animal>`'s body. Note that because the `<animal>` bodies contain white space on their left end and due to the additional horizontal bubble delimiters the bubble will be wider than the `<animal>`'s body. If the **none** option was also used this option has no effect.

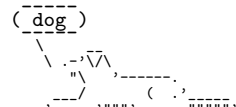
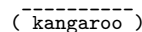
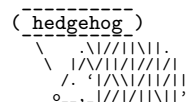
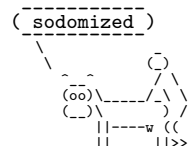
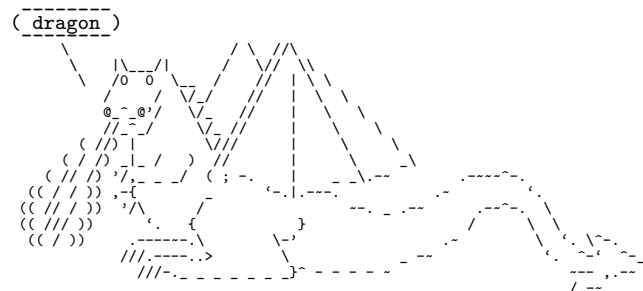
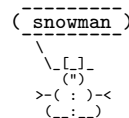
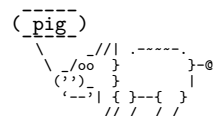
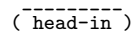
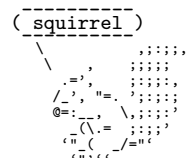
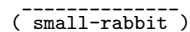
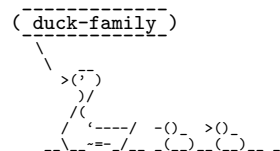
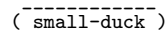
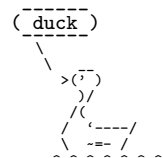


## 1.5 Dependencies

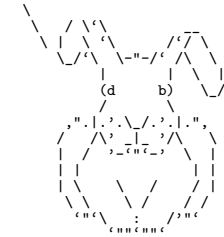
The package depends on the two packages `xparse` and `l3keys2e` and all of their dependencies. Version 2 additionally depends on `array` and `grabbbox`.

## 1.6 Available Animals

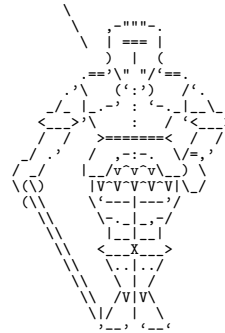
The following animals are provided by this package. I did not create them (but altered some), they belong to their original creators.



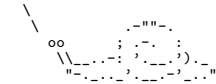
( rabbit )



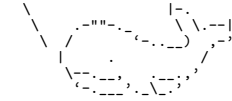
( knight )



( snail )



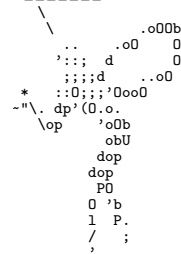
( whale )



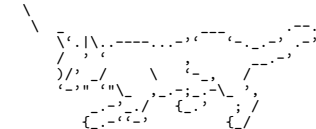
( snake )



( fairy )



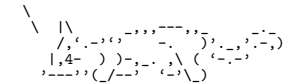
( cat )



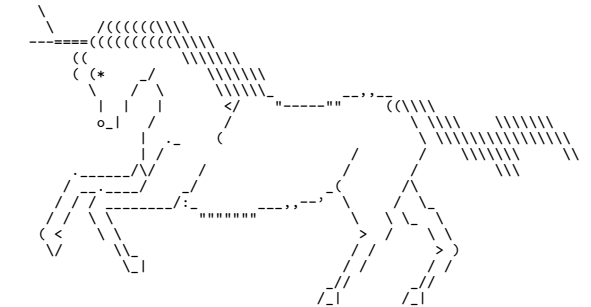
( ghost )



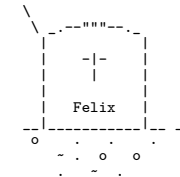
( sleepy-cat )



( unicorn )



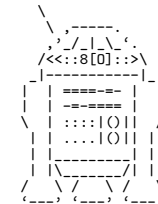
( schroedinger-dead )



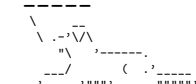
( schroedinger-alive )



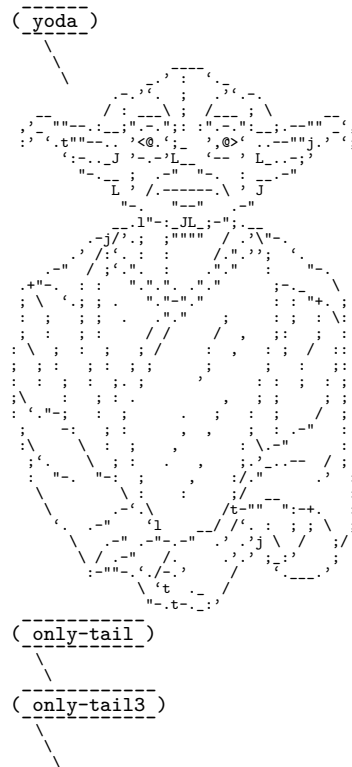
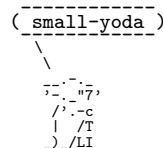
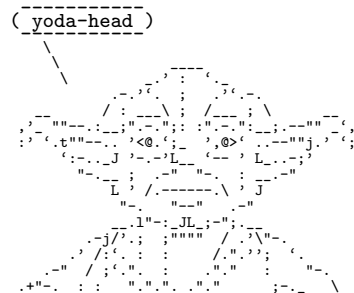
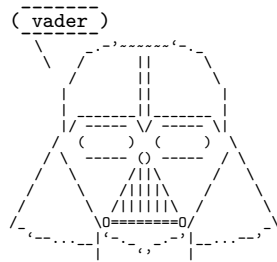
( r2d2 )



( crusader )



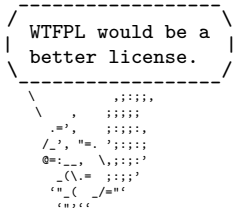




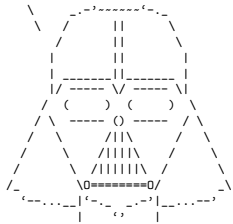
## 1.7 License and Bug Reports

This work may be distributed and/or modified under the conditions of the L<sup>A</sup>T<sub>E</sub>X 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](https://github.com/Skillmon/ltx_ducksay), you might report bugs there.



Only rebel scum reads  
documentation!  
Join the dark side,  
read the implementation.



## 2 Implementation

1 `<*pkg>`

### 2.1 Shared between versions

#### 2.1.1 Variables

##### 2.1.1.1 Integers

2 `\int_new:N \l_ducksay_msg_width_int`  
3 `\int_new:N \l_ducksay_msg_height_int`  
4 `\int_new:N \l_ducksay_tail_symbol_count_int`

##### 2.1.1.2 Sequences

5 `\seq_new:N \l_ducksay_msg_lines_seq`  
6 `\seq_new:N \l_ducksay_defined_animals_seq`

##### 2.1.1.3 Token lists

7 `\tl_new:N \l_ducksay_align_tl`  
8 `\tl_new:N \l_ducksay_msg_align_tl`  
9 `\tl_new:N \l_ducksay_animal_tl`  
10 `\tl_new:N \l_ducksay_body_tl`  
11 `\tl_new:N \l_ducksay_bubble_tl`  
12 `\tl_new:N \l_ducksay_tmpa_tl`  
13 `\tl_new:N \l_ducksay_tail_symbol_out_one_tl`  
14 `\tl_new:N \l_ducksay_tail_symbol_out_two_tl`  
15 `\tl_new:N \l_ducksay_tail_symbol_in_tl`

##### 2.1.1.4 Boolean

16 `\bool_new:N \l_ducksay_version_one_bool`  
17 `\bool_new:N \l_ducksay_version_two_bool`  
18 `\bool_new:N \l_ducksay_random_animal_bool`

##### 2.1.1.5 Boxes

19 `\box_new:N \l_ducksay_tmpa_box`

### 2.1.2 Regular Expressions

Regular expressions for `\AddColoredAnimal`

20 `\regex_const:Nn \c_ducksay_textcolor_regex`  
21 `{ \c0(?:\\textcolor\{(.*)\}\{(.*)\}) }`  
22 `\regex_const:Nn \c_ducksay_color_delim_regex`  
23 `{ \c0(?:\\bgroup\\color\{(.*)\}(.*))\\egroup }`  
24 `\regex_const:Nn \c_ducksay_color_regex`  
25 `{ \c0(?:\\color\{(.*)\}) }`

### 2.1.3 Messages

26 `\msg_new:nnn { ducksay } { load-time-only }`  
27 `{ The~'#1'~key-is-to-be-used-only-during-package-load-time. }`

### 2.1.4 Key-value setup

28 `\keys_define:nn { ducksay }`  
29 `{`  
30  `,bubble .tl_set:N = \l_ducksay_bubble_tl`  
31  `,body .tl_set:N = \l_ducksay_body_tl`

( 17 )  
-----  
\\ .-'\V\\  
"\\  
----- ( . )-----  
)-----)-----)-----)



```

84 \keys_define:nn { ducksay }
85 {
86   version .code:n = \msg_error:nnn { ducksay } { load-time-only } { version }
87 }

```

#### 2.1.4.1 Keys for \AddAnimal

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

```

88 \keys_define:nn { ducksay / add-animal }
89 {
90   ,tail-symbol .code:n =
91     \tl_set:Nx \l_ducksay_tail_symbol_in_tl { \tl_to_str:n { #1 } }
92   ,tail-symbol .initial:o = \c_backslash_str
93   ,tail-count .int_set:N = \l_ducksay_tail_symbol_count_int
94   ,tail-count .initial:n = 2
95 }

```

### 2.1.5 Functions

#### 2.1.5.1 Generating Variants of External Functions

```

96 \cs_generate_variant:Nn \tl_replace_once:Nnn { NVn }
97 \cs_generate_variant:Nn \tl_replace_all:Nnn { NVn }
98 \cs_generate_variant:Nn \keys_set:nn { nx }

```

#### 2.1.5.2 Internal

\\_ducksay\_everyeof:w

```

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

```

(End definition for \\_ducksay\_everyeof:w.)

\\_ducksay\_scantokens:w

```

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

```

(End definition for \\_ducksay\_scantokens:w.)

\ducksay\_replace\_verb\_newline:Nn

```

101 \cs_new_protected:Npx \ducksay_replace_verb_newline:Nn #1 #2
102 {
103   \tl_replace_all:Nnn #1 { \char_generate:nn { 13 } { 12 } } { #2 }
104 }

```

(End definition for \ducksay\_replace\_verb\_newline:Nn.)

\ducksay\_replace\_verb\_newline\_newline:Nn

```

105 \cs_new_protected:Npx \ducksay_replace_verb_newline_newline:Nn #1 #2
106 {
107   \tl_replace_all:Nnn #1
108     { \char_generate:nn { 13 } { 12 } \char_generate:nn { 13 } { 12 } } { #2 }
109 }

```

(End definition for \ducksay\_replace\_verb\_newline\_newline:Nn.)

\ducksay\_process\_verb\_newline:nnn

```
110 \cs_new_protected:Npn \ducksay_process_verb_newline:nnn #1 #2 #3
111 {
112   \tl_set:Nn \ProcessedArgument { #3 }
113   \ducksay_replace_verb_newline_newline:Nn \ProcessedArgument { #2 }
114   \ducksay_replace_verb_newline:Nn \ProcessedArgument { #1 }
115 }
```

(End definition for \ducksay\_process\_verb\_newline:nnn.)

\ducksay\_add\_animal\_inner:nnnn

```
116 \cs_new_protected:Npn \ducksay_add_animal_inner:nnnn #1 #2 #3 #4
117 {
118   \group_begin:
119   \keys_set:nn { ducksay / add-animal } { #1 }
120   \tl_set:Nn \l_ducksay_tmpa_tl { \ #3 }
121   \int_compare:nNnTF { \l_ducksay_tail_symbol_count_int } < { \c_zero_int }
122   {
123     \tl_replace_once:NVn
124     \l_ducksay_tmpa_tl
125     \l_ducksay_tail_symbol_in_tl
126     \l_ducksay_tail_symbol_out_one_tl
127     \tl_replace_all:NVn
128     \l_ducksay_tmpa_tl
129     \l_ducksay_tail_symbol_in_tl
130     \l_ducksay_tail_symbol_out_two_tl
131   }
132   {
133     \int_compare:nNnT { \l_ducksay_tail_symbol_count_int } >
134     { \c_zero_int }
135     {
136       \tl_replace_once:NVn
137       \l_ducksay_tmpa_tl
138       \l_ducksay_tail_symbol_in_tl
139       \l_ducksay_tail_symbol_out_one_tl
140       \int_step_inline:nnn { 2 } { \l_ducksay_tail_symbol_count_int }
141       {
142         \tl_replace_once:NVn
143         \l_ducksay_tmpa_tl
144         \l_ducksay_tail_symbol_in_tl
145         \l_ducksay_tail_symbol_out_two_tl
146       }
147     }
148   }
149   \tl_map_inline:Nn \l_ducksay_ligatures_tl
150   { \tl_replace_all:Nnn \l_ducksay_tmpa_tl { ##1 } { { ##1 } } }
151   \ducksay_replace_verb_newline:Nn \l_ducksay_tmpa_tl
152   { \tabularnewline\null }
153   \exp_args:NNnV
154   \group_end:
155   \tl_set:cn { l_ducksay_animal_#2_tl } \l_ducksay_tmpa_tl
156   \exp_args:Nnx \keys_define:nn { ducksay }
157   {
158     #2 .code:n =
```

( 20 )  
-----  
 \ .-' \ /  
 " \ '-----  
 / ( '-----  
 )-----)-----)-----)-----)

```

159         {
160             \exp_not:n { \tl_set_eq:NN \l_ducksay_animal_tl }
161             \exp_not:c { l_ducksay_animal_#2_tl }
162             \exp_not:n { \exp_args:NV \DucksayOptions }
163             \exp_not:c { l_ducksay_animal_#2_options_tl }
164         }
165     }
166     \tl_if_exist:cF { l_ducksay_animal_#2_options_tl }
167     { \tl_new:c { l_ducksay_animal_#2_options_tl } }
168     \IfBooleanT { #4 }
169     { \keys_define:nn { ducksay } { default_animal .meta:n = { #2 } } }
170     \seq_if_in:NnF \l_ducksay_defined_animals_seq { #2 }
171     { \seq_push:Nn \l_ducksay_defined_animals_seq { #2 } }
172 }
173 \cs_generate_variant:Nn \ducksay_add_animal_inner:nnnn { nnVn }

```

(End definition for \ducksay\_add\_animal\_inner:nnnn.)

\ducksay\_default\_or\_random\_animal:

```

174 \cs_new_protected:Npn \ducksay_default_or_random_animal:
175 {
176     \tl_if_empty:NT \l_ducksay_animal_tl
177     {
178         \bool_if:NTF \l_ducksay_random_animal_bool
179         {
180             \keys_set:nx { ducksay }
181             { \seq_rand_item:N \l_ducksay_defined_animals_seq }
182         }
183         { \keys_set:nn { ducksay } { default_animal } }
184     }
185 }

```

(End definition for \ducksay\_default\_or\_random\_animal:.)

### 2.1.5.3 Document level

**\DefaultAnimal**

```

186 \NewDocumentCommand \DefaultAnimal { m }
187 {
188     \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } }
189 }

```

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

**\DucksayOptions**

```

190 \NewDocumentCommand \DucksayOptions { m }
191 {
192     \keys_set:nn { ducksay } { #1 }
193 }

```

(End definition for \DucksayOptions. This function is documented on page 2.)

## **\AddAnimal**

```
194 \NewDocumentCommand \AddAnimal { s O{} m +v }
195 {
196   \ducksay_add_animal_inner:nnnn { #2 } { #3 } { #4 } { #1 }
197 }
```

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

## **\AddColoredAnimal**

```
198 \NewDocumentCommand \AddColoredAnimal { s O{} m +v }
199 {
200   \tl_set:Nn \l_ducksay_tmpa_tl { #4 }
201   \regex_replace_all:NnN \c_ducksay_color_delim_regex
202     { \c{bgroup}\c{color}\cB{\1\cE}\2\c{egroup} }
203     \l_ducksay_tmpa_tl
204   \regex_replace_all:NnN \c_ducksay_color_regex
205     { \c{color}\cB{\1\cE}\2\c{egroup} }
206     \l_ducksay_tmpa_tl
207   \regex_replace_all:NnN \c_ducksay_textcolor_regex
208     { \c{textcolor}\cB{\1\cE}\2\c{egroup} }
209     \l_ducksay_tmpa_tl
210   \ducksay_add_animal_inner:nnVn { #2 } { #3 } \l_ducksay_tmpa_tl { #1 }
211 }
```

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

## **\AnimalOptions**

```
212 \NewDocumentCommand \AnimalOptions { s m m }
213 {
214   \tl_if_exist:cTF { l_ducksay_animal_#2_options_tl }
215     {
216       \IfBooleanTF { #1 }
217         { \tl_set:cn }
218         { \tl_put_right:cn }
219     }
220     { \tl_set:cn }
221     { l_ducksay_animal_#2_options_tl } { #3, }
222 }
```

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

## **2.1.6 Load the Correct Version and the Animals**

```
223 \bool_if:NT \l_ducksay_version_one_bool
224   { \file_input:n { ducksay.code.v1.tex } }
225 \bool_if:NT \l_ducksay_version_two_bool
226   { \file_input:n { ducksay.code.v2.tex } }
227 \ExplSyntaxOff
228 \input{ducksay.animals.tex}
229 </pkg>
```

## 2.2 Version 1

```

230 <*code.v1>
231 \ProvidesFile{ducksay.code.v1.tex}
232 [\ducksay@date\space v\ducksay@version\space ducksay code version 1]

```

### 2.2.1 Functions

#### 2.2.1.1 Internal

\ducksay\_longest\_line:n Calculate the length of the longest line

```

233 \cs_new:Npn \ducksay_longest_line:n #1
234 {
235   \int_incr:N \l_ducksay_msg_height_int
236   \exp_args:NNx \tl_set:Nn \l_ducksay_tmpa_tl { #1 }
237   \regex_replace_all:nnN { \s } { \c { space } } \l_ducksay_tmpa_tl
238   \int_set:Nn \l_ducksay_msg_width_int
239   {
240     \int_max:nn
241     { \l_ducksay_msg_width_int } { \tl_count:N \l_ducksay_tmpa_tl }
242   }
243 }

```

(End definition for \ducksay\_longest\_line:n.)

\ducksay\_open\_bubble: Draw the opening bracket of the bubble

```

244 \cs_new:Npn \ducksay_open_bubble:
245 {
246   \begin{tabular}{@{}l@{}}
247     \null\
248     \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 } { ( }
249     {
250       /
251       \int_step_inline:nnn
252       { 3 } { \l_ducksay_msg_height_int } { \\ \kern-0.2em| }
253       \\ \detokenize{ \ }
254     }
255     \\ [-1ex] \null
256   \end{tabular}
257   \begin{tabular}{@{}l@{}}
258     _\\
259     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\ [-1ex]
260     \mbox { - }
261   \end{tabular}
262 }

```

(End definition for \ducksay\_open\_bubble:.)

\ducksay\_close\_bubble: Draw the closing bracket of the bubble

```

263 \cs_new:Npn \ducksay_close_bubble:
264 {
265   \begin{tabular}{@{}l@{}}
266     _\\
267     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\ [-1ex]
268     { - }
269   \end{tabular}

```

```

( 23 )
\ .-'\
" \
----- ( .) -----
) -----) -----) -----) -----)

```



```

270 \begin{tabular}{@{}r@{}}
271 \null\
272 \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 }
273 { ) }
274 {
275 \detokenize {\ }
276 \int_step_inline:nnn
277 { 3 } { \l_ducksay_msg_height_int } { \|\kern-0.2em }
278 \\\
279 }
280 \\\[-1ex]\null
281 \end{tabular}
282 }

```

(End definition for \ducksay\_close\_bubble:.)

\ducksay\_print\_msg:nn Print out the message

```

283 \cs_new:Npn \ducksay_print_msg:nn #1 #2
284 {
285 \begin{tabular}{@{} #2 @{}}
286 \int_step_inline:nn { \l_ducksay_msg_width_int } { _ } \\\
287 #1\|[-1ex]
288 \int_step_inline:nn { \l_ducksay_msg_width_int } { { - } }
289 \end{tabular}
290 }
291 \cs_generate_variant:Nn \ducksay_print_msg:nn { nV }

```

(End definition for \ducksay\_print\_msg:nn.)

\ducksay\_print:nn Print out the whole thing

```

292 \cs_new:Npn \ducksay_print:nn #1 #2
293 {
294 \int_compare:nNnTF { \l_ducksay_msg_width_int } < { 0 }
295 {
296 \int_zero:N \l_ducksay_msg_height_int
297 \seq_set_split:Nnn \l_ducksay_msg_lines_seq { \\\ } { #1 }
298 \seq_map_function:NN \l_ducksay_msg_lines_seq \ducksay_longest_line:n
299 }
300 {
301 \int_compare:nNnT { \l_ducksay_msg_height_int } < { 0 }
302 {
303 \regex_count:nnN { \c { \\\ } } { #1 } \l_ducksay_msg_height_int
304 \int_incr:N \l_ducksay_msg_height_int
305 }
306 }
307 \group_begin:
308 \frenchspacing
309 \verbatim@font
310 \@noligs
311 \begin{tabular}[\l_ducksay_align_tl]{@{}#2@{}}
312 \l_ducksay_bubble_tl
313 \begin{tabular}{@{}l@{}}
314 \ducksay_open_bubble:
315 \ducksay_print_msg:nV { #1 } \l_ducksay_msg_align_tl
316 \ducksay_close_bubble:

```

```

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

```

```

317         \end{tabular}\\
318         \l_ducksay_body_tl
319         \begin{tabular}{@{}l@{}}
320             \l_ducksay_animal_tl
321         \end{tabular}
322     \end{tabular}
323 \group_end:
324 }
325 \cs_generate_variant:Nn \ducksay_print:nn { nV }

```

(End definition for \ducksay\_print:nn.)

\ducksay\_say\_and\_think:nn Reset some variables

```

326 \cs_new:Npn \ducksay_say_and_think:nn #1 #2
327 {
328     \group_begin:
329     \int_set:Nn \l_ducksay_msg_width_int { -\c_max_int }
330     \int_set:Nn \l_ducksay_msg_height_int { -\c_max_int }
331     \keys_set:nn { ducksay } { #1 }
332     \ducksay_default_or_random_animal:
333     \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
334 \group_end:
335 }

```

(End definition for \ducksay\_say\_and\_think:nn.)

### 2.2.1.2 Document level

**\ducksay**

```

336 \NewDocumentCommand \ducksay { 0{} m }
337 {
338     \ducksay_say_and_think:nn { #1 } { #2 }
339 }

```

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

**\duckthink**

```

340 \NewDocumentCommand \duckthink { 0{} m }
341 {
342     \ducksay_say_and_think:nn { think, #1 } { #2 }
343 }

```

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

```

344 \</code.v1>

```

## 2.3 Version 2

```

345 <*code.v2>
346 \ProvidesFile{ducksay.code.v2.tex}
347 [\ducksay@date\space v\ducksay@version\space ducksay code version 2]

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

```

### 2.3.1 Messages

```

349 \msg_new:nnn { ducksay } { justify~unavailable }
350 {
351   Justified~content~is~not~available~for~tabular~argument~mode~without~fixed~
352   width.~'l'~column~is~used~instead.
353 }
354 \msg_new:nnn { ducksay } { unknown~message~alignment }
355 {
356   The~specified~message~alignment~'\exp_not:n { #1 }'~is~unknown.~
357   'l'~is~used~as~fallback.
358 }
359 \msg_new:nnn { ducksay } { v1~key~only }
360 { The~'\l_keys_key_tl'~key~is~only~available~for~'version=1'. }

```

### 2.3.2 Variables

#### 2.3.2.1 Token Lists

```

361 \tl_new:N \l_ducksay_msg_align_vbox_tl

```

#### 2.3.2.2 Boxes

```

362 \box_new:N \l_ducksay_msg_box

```

#### 2.3.2.3 Bools

```

363 \bool_new:N \l_ducksay_eat_arg_box_bool
364 \bool_new:N \l_ducksay_eat_arg_tab_verb_bool
365 \bool_new:N \l_ducksay_mirrored_body_bool
366 \bool_new:N \l_ducksay_msg_eq_body_width_bool

```

#### 2.3.2.4 Coffins

```

367 \coffin_new:N \l_ducksay_body_coffin
368 \coffin_new:N \l_ducksay_bubble_close_coffin
369 \coffin_new:N \l_ducksay_bubble_open_coffin
370 \coffin_new:N \l_ducksay_bubble_top_coffin
371 \coffin_new:N \l_ducksay_msg_coffin

```

#### 2.3.2.5 Dimensions

```

372 \dim_new:N \l_ducksay_hpad_dim
373 \dim_new:N \l_ducksay_bubble_bottom_kern_dim
374 \dim_new:N \l_ducksay_bubble_top_kern_dim
375 \dim_new:N \l_ducksay_msg_width_dim

```

### 2.3.3 Options

```

376 \keys_define:nn { ducksay }
377 {
378   ,arg .choice:
379   ,arg / box .code:n = \bool_set_true:N \l_ducksay_eat_arg_box_bool
380   ,arg / tab .code:n =

```

```

381     {
382         \bool_set_false:N \l_ducksay_eat_arg_box_bool
383         \bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool
384     }
385 ,arg / tab* .code:n =
386     {
387         \bool_set_false:N \l_ducksay_eat_arg_box_bool
388         \bool_set_true:N \l_ducksay_eat_arg_tab_verb_bool
389     }
390 ,arg .initial:n = tab
391 ,wd* .dim_set:N = \l_ducksay_msg_width_dim
392 ,wd* .initial:n = -\c_max_dim
393 ,wd* .value_required:n = true
394 ,wd-eq-body .bool_set:N = \l_ducksay_msg_eq_body_width_bool
395 ,none .bool_set:N = \l_ducksay_no_body_bool
396 ,no-bubble .bool_set:N = \l_ducksay_no_bubble_bool
397 ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
398 ,ignore-body .bool_set:N = \l_ducksay_ignored_body_bool
399 ,body-x .dim_set:N = \l_ducksay_body_x_offset_dim
400 ,body-x .value_required:n = true
401 ,body-y .dim_set:N = \l_ducksay_body_y_offset_dim
402 ,body-y .value_required:n = true
403 ,body-to-msg .tl_set:N = \l_ducksay_body_to_msg_align_body_tl
404 ,msg-to-body .tl_set:N = \l_ducksay_body_to_msg_align_msg_tl
405 ,body-align .choice:
406 ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
407 ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
408 ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
409 ,body-align .initial:n = l
410 ,body-bigger .int_set:N = \l_ducksay_body_bigger_int
411 ,body-bigger .initial:n = \c_zero_int
412 ,msg-align .choice:
413 ,msg-align / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
414 ,msg-align / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
415 ,msg-align / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
416 ,msg-align / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
417 ,msg-align-l .tl_set:N = \l_ducksay_msg_align_l_tl
418 ,msg-align-l .initial:n = \raggedright
419 ,msg-align-c .tl_set:N = \l_ducksay_msg_align_c_tl
420 ,msg-align-c .initial:n = \centering
421 ,msg-align-r .tl_set:N = \l_ducksay_msg_align_r_tl
422 ,msg-align-r .initial:n = \raggedleft
423 ,msg-align-j .tl_set:N = \l_ducksay_msg_align_j_tl
424 ,msg-align-j .initial:n = {}
425 ,out-h .tl_set:N = \l_ducksay_output_h_pole_tl
426 ,out-h .initial:n = l
427 ,out-v .tl_set:N = \l_ducksay_output_v_pole_tl
428 ,out-v .initial:n = vc
429 ,out-x .dim_set:N = \l_ducksay_output_x_offset_dim
430 ,out-x .value_required:n = true
431 ,out-y .dim_set:N = \l_ducksay_output_y_offset_dim
432 ,out-y .value_required:n = true
433 ,t .meta:n = { out-v = t }
434 ,c .meta:n = { out-v = vc }

```

```

435 ,b      .meta:n    = { out-v = b }
436 ,body*   .tl_set:N = \l_ducksay_body_fount_tl
437 ,msg*    .tl_set:N = \l_ducksay_msg_fount_tl
438 ,bubble* .tl_set:N = \l_ducksay_bubble_fount_tl
439 ,body*   .initial:n = \verbatim@font
440 ,msg*    .initial:n = \verbatim@font
441 ,bubble* .initial:n = \verbatim@font
442 ,body    .code:n    = \tl_put_right:Nn \l_ducksay_body_fount_tl { #1 }
443 ,msg     .code:n    = \tl_put_right:Nn \l_ducksay_msg_fount_tl { #1 }
444 ,bubble  .code:n    = \tl_put_right:Nn \l_ducksay_bubble_fount_tl { #1 }
445 ,MSG     .meta:n    = { msg = #1 , bubble = #1 }
446 ,MSG*    .meta:n    = { msg* = #1 , bubble* = #1 }
447 ,hpad    .int_set:N = \l_ducksay_hpad_int
448 ,hpad    .initial:n = 2
449 ,hpad    .value_required:n = true
450 ,vpad    .int_set:N = \l_ducksay_vpad_int
451 ,vpad    .value_required:n = true
452 ,col     .tl_set:N = \l_ducksay_msg_tabular_column_tl
453 ,bubble-top-kern .tl_set:N = \l_ducksay_bubble_top_kern_tl
454 ,bubble-top-kern .initial:n = { -.5ex }
455 ,bubble-top-kern .value_required:n = true
456 ,bubble-bot-kern .tl_set:N = \l_ducksay_bubble_bottom_kern_tl
457 ,bubble-bot-kern .initial:n = { .2ex }
458 ,bubble-bot-kern .value_required:n = true
459 ,bubble-side-kern .tl_set:N = \l_ducksay_bubble_side_kern_tl
460 ,bubble-side-kern .initial:n = { .2em }
461 ,bubble-side-kern .value_required:n = true
462 ,bubble-delim-top .tl_set:N = \l_ducksay_bubble_delim_top_tl
463 ,bubble-delim-left-1 .tl_set:N = \l_ducksay_bubble_delim_left_a_tl
464 ,bubble-delim-left-2 .tl_set:N = \l_ducksay_bubble_delim_left_b_tl
465 ,bubble-delim-left-3 .tl_set:N = \l_ducksay_bubble_delim_left_c_tl
466 ,bubble-delim-left-4 .tl_set:N = \l_ducksay_bubble_delim_left_d_tl
467 ,bubble-delim-right-1 .tl_set:N = \l_ducksay_bubble_delim_right_a_tl
468 ,bubble-delim-right-2 .tl_set:N = \l_ducksay_bubble_delim_right_b_tl
469 ,bubble-delim-right-3 .tl_set:N = \l_ducksay_bubble_delim_right_c_tl
470 ,bubble-delim-right-4 .tl_set:N = \l_ducksay_bubble_delim_right_d_tl
471 ,bubble-delim-top .initial:n = { { - } }
472 ,bubble-delim-left-1 .initial:n = (
473 ,bubble-delim-left-2 .initial:n = /
474 ,bubble-delim-left-3 .initial:n = |
475 ,bubble-delim-left-4 .initial:n = \c_backslash_str
476 ,bubble-delim-right-1 .initial:n = )
477 ,bubble-delim-right-2 .initial:n = \c_backslash_str
478 ,bubble-delim-right-3 .initial:n = |
479 ,bubble-delim-right-4 .initial:n = /
480 ,strip-spaces .bool_set:N = \l_ducksay_msg_strip_spaces_bool
481 }

```

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

```

482 \clist_map_inline:nn
483 { align, rel-align }
484 {
485   \keys_define:nn { ducksay }
486     { #1 .code:n = \msg_error:nn { ducksay } { v1-key-only } }
487 }

```

### 2.3.4 Functions

#### 2.3.4.1 Internal

```
evaluate_message_alignment_fixed_width_common:
```

```

488 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:
489 {
490   \str_case:Vn \l_ducksay_msg_align_tl
491   {
492     { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
493     { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
494     { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
495     { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
496   }
497 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_common:.)

evaluate\_message\_alignment\_fixed\_width\_tabular:

```

498 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_tabular:
499 {
500   \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
501   {
502     \tl_set:Nx \l_ducksay_msg_tabular_column_tl
503     {
504       >
505       {
506         \ducksay_evaluate_message_alignment_fixed_width_common:
507         \exp_not:N \arraybackslash
508       }
509       p { \exp_not:N \l_ducksay_msg_width_dim }
510     }
511   }
512 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_tabular:.)

```
evaluate message alignment fixed width vbox:
```

```

513 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_vbox:
514 {
515   \tl_set:Nx \l_ducksay_msg_align_vbox_tl
516     { \ducksay_evaluate_message_alignment_fixed_width_common: }
517 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_vbox:.)

```
\ducksay_calculate_msg_width_from_int:
```

```

518 \cs_new:Npn \ducksay_calculate_msg_width_from_int:
519 {
520   \hbox_set:Nn \l_ducksay_tmpa_box { { \l_ducksay_msg_fount_tl M } }
521   \dim_set:Nn \l_ducksay_msg_width_dim
522     { \l_ducksay_msg_width_int \box_wd:N \l_ducksay_tmpa_box }
523 }

```

(End definition for \ducksay\_calculate\_msg\_width\_from\_int:.)

\ducksay\_msg\_tabular\_begin:

```

524 \cs_new:Npn \ducksay_msg_tabular_begin:
525 {
526   \ducksay_msg_tabular_begin_inner:V \l_ducksay_msg_tabular_column_tl
527 }
528 \cs_new:Npn \ducksay_msg_tabular_begin_inner:n #1
529 {
530   \begin { tabular } { @{} #1 @{} }
531 }
532 \cs_generate_variant:Nn \ducksay_msg_tabular_begin_inner:n { V }

```

(End definition for \ducksay\_msg\_tabular\_begin:.)

\ducksay\_msg\_tabular\_end:

```

533 \cs_new:Npn \ducksay_msg_tabular_end:
534 {
535   \end { tabular }
536 }

```

(End definition for \ducksay\_msg\_tabular\_end:.)

\ducksay\_width\_case\_none\_int\_dim:nnn

```

537 \cs_new:Npn \ducksay_width_case_none_int_dim:nnn #1 #2 #3
538 {
539   \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
540   {
541     \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
542     { #1 }
543     { #2 }
544   }
545   { #3 }
546 }

```

(End definition for \ducksay\_width\_case\_none\_int\_dim:nnn.)

\ducksay\_digest\_options:n

```

547 \cs_new:Npn \ducksay_digest_options:n #1
548 {
549   \group_begin:
550   \keys_set:nn { ducksay } { #1 }
551   \ducksay_default_or_random_animal:
552   \bool_if:NF \l_ducksay_no_body_bool
553   {
554     \hcoffin_set:Nn \l_ducksay_body_coffin
555     {
556       \frenchspacing
557       \l_ducksay_body_fount_tl
558       \begin{tabular} { @{} l @{} }
559         \l_ducksay_animal_tl
560         \ducksay_make_body_bigger:
561         \relax
562       \end{tabular}
563     }
564     \bool_if:NT \l_ducksay_msg_eq_body_width_bool

```

```

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

```

```

565     {
566         \bool_lazy_and:nnT
567         { \int_compare_p:nNn \l_ducksay_msg_width_int < \c_zero_int }
568         { \dim_compare_p:nNn \l_ducksay_msg_width_dim < \c_zero_dim }
569         {
570             \dim_set:Nn \l_ducksay_msg_width_dim
571             { \coffin_wd:N \l_ducksay_body_coffin }
572         }
573     }
574 }
575 \bool_if:NTF \l_ducksay_eat_arg_box_bool
576 {
577     \ducksay_width_case_none_int_dim:nnn
578     { \ducksay_eat_argument_hbox:w }
579     {
580         \ducksay_calculate_msg_width_from_int:
581         \ducksay_eat_argument_vbox:w
582     }
583     { \ducksay_eat_argument_vbox:w }
584 }
585 {
586     \ducksay_width_case_none_int_dim:nnn
587     {
588         \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
589         {
590             \str_case:Vn \l_ducksay_msg_align_tl
591             {
592                 { l } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
593                 { c } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
594                 { r } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
595                 { j }
596                 {
597                     \msg_error:nn { ducksay } { justify~unavailable }
598                     \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
599                 }
600             }
601         }
602     }
603     {
604         \ducksay_calculate_msg_width_from_int:
605         \ducksay_evaluate_message_alignment_fixed_width_tabular:
606     }
607     { \ducksay_evaluate_message_alignment_fixed_width_tabular: }
608     \ducksay_eat_argument_tabular:w
609 }
610 }

```

(End definition for \ducksay\_digest\_options:n.)

\ducksay\_set\_bubble\_top\_kern:

```

611 \cs_new:Npn \ducksay_set_bubble_top_kern:
612 {
613     \group_begin:
614     \l_ducksay_bubble_fount_tl

```

```

( 31 )
\ .-'\ \
" \
----- ( '-----
)-----)-----)-----)-----)

```



```

615     \exp_args:NNNx
616     \group_end:
617     \dim_set:Nn \l_ducksay_bubble_top_kern_dim
618     { \dim_eval:n { \l_ducksay_bubble_top_kern_tl } }
619 }

```

(End definition for \ducksay\_set\_bubble\_top\_kern:.)

\ducksay\_set\_bubble\_bottom\_kern:

```

620 \cs_new:Npn \ducksay_set_bubble_bottom_kern:
621 {
622     \group_begin:
623     \l_ducksay_bubble_fount_tl
624     \exp_args:NNNx
625     \group_end:
626     \dim_set:Nn \l_ducksay_bubble_bottom_kern_dim
627     { \dim_eval:n { \l_ducksay_bubble_bottom_kern_tl } }
628 }

```

(End definition for \ducksay\_set\_bubble\_bottom\_kern:.)

\ducksay\_make\_body\_bigger:

```

629 \cs_new:Npn \ducksay_make_body_bigger:
630 {
631     \int_step_function:nN \l_ducksay_body_bigger_int
632     \ducksay_make_body_bigger_aux:n
633 }

```

(End definition for \ducksay\_make\_body\_bigger:.)

\ducksay\_make\_body\_bigger\_aux:n

```

634 \cs_new:Npn \ducksay_make_body_bigger_aux:n #1
635 {
636     \\\
637 }

```

(End definition for \ducksay\_make\_body\_bigger\_aux:n.)

\ducksay\_shipout:

```

638 \cs_new_protected:Npn \ducksay_shipout:
639 {
640     \hcoffin_set:Nn \l_ducksay_msg_coffin { \box_use:N \l_ducksay_msg_box }
641     \bool_if:NF \l_ducksay_no_bubble_bool
642     {
643         \hbox_set:Nn \l_ducksay_tmpa_box
644         { \l_ducksay_bubble_fount_tl \l_ducksay_bubble_delim_top_tl }
645         \int_set:Nn \l_ducksay_msg_width_int
646         {
647             \fp_eval:n
648             {
649                 ceil
650                 (
651                     \box_wd:N \l_ducksay_msg_box / \box_wd:N \l_ducksay_tmpa_box
652                 )
653             }

```

```

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

```

```

654     }
655     \group_begin:
656     \l_ducksay_bubble_fount_tl
657     \exp_args:NNNx
658     \group_end:
659     \int_set:Nn \l_ducksay_msg_height_int
660     {
661         \int_max:nn
662         {
663             \fp_eval:n
664             {
665                 ceil
666                 (
667                     (
668                         \box_ht:N \l_ducksay_msg_box
669                         + \box_dp:N \l_ducksay_msg_box
670                     )
671                     / ( \arraystretch * \baselineskip )
672                 )
673             }
674             + \l_ducksay_vpad_int
675         }
676         { \l_ducksay_msg_height_int }
677     }
678     \hcoffin_set:Nn \l_ducksay_bubble_open_coffin
679     {
680         \l_ducksay_bubble_fount_tl
681         \begin{tabular}{@{}l@{}}
682             \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
683             {
684                 \l_ducksay_bubble_delim_left_a_tl
685             }
686             {
687                 \l_ducksay_bubble_delim_left_b_tl\\
688                 \int_step_inline:nnn
689                 { 3 } { \l_ducksay_msg_height_int }
690                 {
691                     \kern-\l_ducksay_bubble_side_kern_tl
692                     \l_ducksay_bubble_delim_left_c_tl
693                     \\
694                 }
695                 \l_ducksay_bubble_delim_left_d_tl
696             }
697         \end{tabular}
698     }
699     \hcoffin_set:Nn \l_ducksay_bubble_close_coffin
700     {
701         \l_ducksay_bubble_fount_tl
702         \begin{tabular}{@{}r@{}}
703             \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
704             {
705                 \l_ducksay_bubble_delim_right_a_tl
706             }
707             {

```

```

708         \l_ducksay_bubble_delim_right_b_tl \\
709         \int_step_inline:nnn
710         { 3 } { \l_ducksay_msg_height_int }
711         {
712             \l_ducksay_bubble_delim_right_c_tl
713             \kern-\l_ducksay_bubble_side_kern_tl
714             \\
715         }
716         \l_ducksay_bubble_delim_right_d_tl
717     }
718     \end{tabular}
719 }
720 \hcoffin_set:Nn \l_ducksay_bubble_top_coffin
721 {
722     \l_ducksay_bubble_fount_tl
723     \int_step_inline:nn
724     { \l_ducksay_msg_width_int + \l_ducksay_hpad_int }
725     { \l_ducksay_bubble_delim_top_tl }
726 }
727 \dim_set:Nn \l_ducksay_hpad_dim
728 {
729     (
730         \coffin_wd:N \l_ducksay_bubble_top_coffin
731         - \coffin_wd:N \l_ducksay_msg_coffin
732     ) / 2
733 }
734 \coffin_join:NnnNnnnn
735     \l_ducksay_msg_coffin          { l } { vc }
736     \l_ducksay_bubble_open_coffin { r } { vc }
737     { - \l_ducksay_hpad_dim } { \c_zero_dim }
738 \coffin_join:NnnNnnnn
739     \l_ducksay_msg_coffin          { r } { vc }
740     \l_ducksay_bubble_close_coffin { l } { vc }
741     { \l_ducksay_hpad_dim } { \c_zero_dim }
742 \ducksay_set_bubble_top_kern:
743 \ducksay_set_bubble_bottom_kern:
744 \coffin_join:NnnNnnnn
745     \l_ducksay_msg_coffin          { hc } { t }
746     \l_ducksay_bubble_top_coffin { hc } { b }
747     { \c_zero_dim } { \l_ducksay_bubble_top_kern_dim }
748 \coffin_join:NnnNnnnn
749     \l_ducksay_msg_coffin          { hc } { b }
750     \l_ducksay_bubble_top_coffin { hc } { t }
751     { \c_zero_dim } { \l_ducksay_bubble_bottom_kern_dim }
752 }
753 \bool_if:NF \l_ducksay_no_body_bool
754 {
755     \bool_if:NT \l_ducksay_mirrored_body_bool
756     {
757         \coffin_scale:Nnn \l_ducksay_body_coffin
758         { -\c_one_int } { \c_one_int }
759         \str_case:Vn \l_ducksay_body_to_msg_align_body_tl
760         {
761             { l } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { r } }

```

```

762         { r } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { l } }
763     }
764 }
765 \bool_if:NTF \l_ducksay_ignored_body_bool
766 { \coffin_attach:NVnNVnnn }
767 { \coffin_join:NVnNVnnn }
768 \l_ducksay_msg_coffin \l_ducksay_body_to_msg_align_msg_tl { b }
769 \l_ducksay_body_coffin \l_ducksay_body_to_msg_align_body_tl { t }
770 { \l_ducksay_body_x_offset_dim } { \l_ducksay_body_y_offset_dim }
771 }
772 \coffin_typeset:NVVnn \l_ducksay_msg_coffin
773 \l_ducksay_output_h_pole_tl \l_ducksay_output_v_pole_tl
774 { \l_ducksay_output_x_offset_dim } { \l_ducksay_output_y_offset_dim }
775 \group_end:
776 }

```

(End definition for \ducksay\_shipout:.)

**2.3.4.1.1 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.

\ducksay\_eat\_argument\_tabular:w

```

777 \cs_new:Npn \ducksay_eat_argument_tabular:w
778 {
779     \bool_if:NTF \l_ducksay_eat_arg_tab_verb_bool
780     { \ducksay_eat_argument_tabular_verb:w }
781     { \ducksay_eat_argument_tabular_normal:w }
782 }

```

(End definition for \ducksay\_eat\_argument\_tabular:w.)

\ducksay\_eat\_argument\_tabular\_inner:w

```

783 \cs_new:Npn \ducksay_eat_argument_tabular_inner:w #1
784 {
785     \hbox_set:Nn \l_ducksay_msg_box
786     {
787         \l_ducksay_msg_fount_tl
788         \ducksay_msg_tabular_begin:
789         #1
790         \ducksay_msg_tabular_end:
791     }
792     \ducksay_shipout:
793 }

```

(End definition for \ducksay\_eat\_argument\_tabular\_inner:w.)

\ducksay\_eat\_argument\_tabular\_verb:w

```

794 \NewDocumentCommand \ducksay_eat_argument_tabular_verb:w
795 { >{ \ducksay_process_verb_newline:nnn { ~ } { ~ \par } } +v }
796 {
797     \ducksay_eat_argument_tabular_inner:w
798     {
799         \group_begin:

```

```

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

```

```

800         \_ducksay_everyeof:w { \exp_not:N }
801         \exp_after:wN
802         \group_end:
803         \_ducksay_scantokens:w { #1 }
804     }
805 }

```

(End definition for \ducksay\_eat\_argument\_tabular\_verb:w.)

\ducksay\_eat\_argument\_tabular\_normal:w

```

806 \NewDocumentCommand \ducksay_eat_argument_tabular_normal:w { +m }
807 { \ducksay_eat_argument_tabular_inner:w { #1 } }

```

(End definition for \ducksay\_eat\_argument\_tabular\_normal:w.)

\ducksay\_eat\_argument\_hbox:w

```

808 \cs_new_protected_nopar:Npn \ducksay_eat_argument_hbox:w
809 {
810     \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
811     { \@grabbox }
812     { \@grabbox* }
813     {} \l_ducksay_msg_box \l_ducksay_msg_fount_tl \hbox {} \ducksay_shipout:
814 }

```

(End definition for \ducksay\_eat\_argument\_hbox:w.)

\ducksay\_eat\_argument\_vbox:w

```

815 \cs_new_protected_nopar:Npn \ducksay_eat_argument_vbox:w
816 {
817     \ducksay_evaluate_message_alignment_fixed_width_vbox:
818     \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
819     { \@grabbox }
820     { \@grabbox* }
821     {
822         \hsize \l_ducksay_msg_width_dim
823         \linewidth \hsize
824         \l_ducksay_msg_align_vbox_tl
825         \@afterindentfalse
826         \@afterheading
827     }
828     \l_ducksay_msg_box \l_ducksay_msg_fount_tl \vbox {} \ducksay_shipout:
829 }

```

(End definition for \ducksay\_eat\_argument\_vbox:w.)

### 2.3.4.1.2 Generating Variants of External Functions

```

830 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
831 \cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
832 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
833 \cs_generate_variant:Nn \str_case:nn { Vn }

```

### 2.3.4.2 Document level

**\ducksay**

```
834 \NewDocumentCommand \ducksay { 0{} }
835 {
836   \ducksay_digest_options:n { #1 }
837 }
```

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

**\duckthink**

```
838 \NewDocumentCommand \duckthink { 0{} }
839 {
840   \ducksay_digest_options:n { think, #1 }
841 }
```

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

```
842 \</code.v2>
```

## 2.4 Definition of the Animals

```

843 <*animals>
844 \ProvidesFile{ducksay.animals.tex}
845 [\ducksay@date\space v\ducksay@version\space ducksay animals]
846 %^~A some of the below are from http://ascii.co.uk/art/
847 \AddAnimal{duck}%>>=
848 { \
849   \
850     >(')
851     )/
852     /(
853     / '----/
854     \ ~-- /
855     ~~~~~~}%<<
856 \AddAnimal{small-duck}%>>=
857 { \
858   \
859     >()_
860     (__)__ _}%<<
861 \AddAnimal{duck-family}%>>=
862 { \
863   \
864     >(')
865     )/
866     /(
867     / '----/ -()_ >()_
868     __\__~--/_/_ _(_)_(_)_ _}%<<
869 \AddAnimal{cow}%>>=
870 { \
871   \ ^__^
872     \ (oo)\_____
873       (__)\       )\/\
874         ||----w |
875         ||     ||}%<<
876 \AddAnimal{head-in}%>>=
877 { \
878   \ ^__^
879     \ (oo)\_____
880       (__)\       )\/\
881         ||----w |
882         ||     ||}%<<
883 \AddAnimal{sodomized}%>>=
884 { \
885   \ ^__^
886     \ (oo)\_____
887       (__)\       )\/\
888         ||----w ((
889         ||     ||>>}%<<
890 \AddAnimal{tux}%>>=
891 { \
892   \ .--.
893     |o_o |
894     |\\_ |

```

```

895     //  \ \
896     (|    | )
897     /'\_  _/'\
898     \___)=(___/}%=<<
899 \AddAnimal{pig}%>>=
900 + \    _//| .-~~~~-.
901     \ _/oo }      }-@
902     ('')_ }      |
903     '---| { }--{ }
904     //_/_/_/+}%=<<
905 \AddAnimal{frog}%>>=
906 { \
907     \ (.)_(.)
908     _ ( _ ) _
909     / \/'-----'\ \
910 --\ ( ( ) ) /--
911 )   /\ \_./ /\ (
912 )_/_/ /\ \_/_/ \%=<<
913 \AddAnimal{snowman}%>>=
914 { \
915     \[_]_
916     (")
917     >-( : )-<
918     ( _ : _ )}%=<<
919 \AddAnimal[tail-symbol=s]{hedgehog}%>>=
920 { s    .\|//||\|||.
921     s  |/\|//|/\|//|
922     /. '|\|//|/\|//|
923     o__._|//|//|//|'}%=<<
924 \AddAnimal{kangaroo}%>>=
925 { \
926     \_-, '
927     <--\__/_ \
928     \_ / \_ \
929     \, \ / \ \
930     // \ \
931     ,/' ' \_,}%=<<
932 %^~A http://chris.com/ascii/index.php?art=animals/rabbits
933 \AddAnimal[tail-symbol=s,tail-count=3]{rabbit}%>>=
934 { s
935     s / \'\
936     s | \ \ ' \ /' / \
937     \_/' \ \ -"/' / \ \
938     |      | \ \ |
939     (d      b) \_/_/
940     /      \
941     ,".|.'. \_/.'.|. ",
942     / \/' _|_ ' \ \
943     | / ' ' " " ' \ \
944     | |      | |
945     | \ \ \ / / |
946     \ \ \ / / /
947     " " \ : /' " '
948     " " " " " }%=<<

```





```

1003      \      /((((((\\
1004  -----((((((((\\
1005      ((      \\\
1006      ( (*      _/      \\\
1007      \      / \      \\\
1008      | | |      </      "-----"      --,--
1009      o_|      /      /      \\\
1010      |      .-      (      \\\
1011      |      /      /      \\\
1012      .-----/ \      /      \\\
1013      /      /      /      \\\
1014      /      /      /      \\\
1015      ( <      \ \      "-----"      --,--
1016      \      \ \      >      \ \
1017      \      \ \      /      /
1018      \      \ \      /      /
1019      \      \ \      /      /
1020      /_ |      /_ |}%=<<
1021 %^~A https://asciiart.website//index.php?art=animals/other%20(water)
1022 \AddAnimal[tail-count=3,tail-symbol=s]{whale}%>>=
1023 { s      |-.
1024      s      .-"-.-      \ \.-|
1025      s /      '---)      ,-'
1026      |      .      /
1027      \---.-      .---.-
1028      '---.-      .\_-'}%=<<
1029 %^~A from http://www.ascii-art.de/ascii/s/starwars.txt :
1030 \AddAnimal[tail-count=3]{yoda}%>>=
1031 { \
1032      \
1033      .-.' ;      .-.'
1034      / : --- \      / --- ; \
1035      ,-' "----.-;" : "----.-;" : "----.-;" : "----.-;"
1036      : ' .t"---.-.' <@.' ;      ,@> ' .t"---.-.' j.' ;
1037      ' : -.-J ' -.-L_ ' -.-L_ .-.-'
1038      " -.- ;      " -.- :      " -.-
1039      L ' / .-----.\ ' J
1040      " -.-      " -.-      " -.-
1041      ---.l"-:-JL_:-";      .-j/' ;      ; " " " / .'\ " -.-
1042      .-' / : ' :      / .'" ;      '
1043      .-" / ;      " :      " :      " -.-
1044      .+"-.- :      " " " " ;      -.- \
1045      ; \ ' ;      " " " " :      "+. ;
1046      : ; ;      " " " ;      : ; \
1047      ; : ;      / / / , ;      :
1048      : \ ;      ; / : , ;      / :
1049      : ; ;      ; ; ;      ; ; ;
1050      : ; ;      ; ; ;      ; ; ;
1051      : ; ;      ; ; ;      ; ; ;
1052      ; \ : ;      , ; ;      ; ;
1053      : ' "- ;      ;      ;      ; /
1054      ;      - : ;      , , ;      .-" :
1055      : \      \ : ;      , : \.-" :

```



```

1111      |      ||      |
1112      |-----||-----|
1113      | / ----- \ / ----- \ |
1114      / ( ) ( ) \
1115      / \ ----- () ----- / \
1116      / \      /||\
1117      / \      /||||\
1118      / \      /|||||\
1119      /- \0=====0/ \-
1120      '---..._|'-'-'-'|_---...-'
1121      |      '      |}%=<<
1122 \AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>=>=
1123 { |
1124 \[T]/}
1125 \csname bool_if:cT\endcsname {l_ducksay_version_one_bool}
1126 { \AnimalOptions{crusader}{tail-1=|,rel-align=c}}
1127 \csname bool_if:cT\endcsname {l_ducksay_version_two_bool}
1128 { \AnimalOptions{crusader}{tail-1=|,body-align=c}}}%=<<
1129 %^A http://ascii.co.uk/art/knights
1130 \AddAnimal[tail-count=3]{knight}%>=>=
1131 {
1132 \
1133 \      ,-"-".
1134 \      |===|
1135 \      ) | (
1136      .==\' " "/\'==.
1137      .\' \ (\' :) /\'
1138      _/_ |_-\' : \'_-|_/_
1139      <--->\' \ : / \'<--->
1140      / / >=====< / /
1141      _/_ .\' / ,--:-- \/=,\'
1142      /_/_ |_/v^v^v^v\_) \
1143      \(\) |V^V^V^V^V|\_/
1144      (\ \ \ \'---|---\'/
1145      \ \ \ \ \'_-|_-,-/
1146      \ \ \ \ |__|__|
1147      \ \ \ \ <---X--->
1148      \ \ \ \ \..|..|
1149      \ \ \ \ \ | /
1150      \ \ \ \ /V|V\
1151      \ | / | \
1152      ,--\' \'--\'}%=<<
1153 %^A https://www.asciiart.eu/mythology/ghosts
1154 \AddAnimal{ghost}%>=>=
1155 {
1156 \
1157 \      .-.
1158 \      (o o)
1159 \      | 0 \
1160 \      \ \
1161 \      \'~~~\'}%=<<
1162 %^A https://www.asciiart.eu/mythology/fairies
1163 \AddAnimal{fairy}%>=>=
1164 {
1165 \
1166 \      .o00b
1167 \      ..      .o0      0

```



```

1219   |   -|-   |
1220   |     |   |
1221   |     |   |
1222   |   Felix   |
1223  --|-----|--
1224   o   .   .   .
1225   ~ . o o
1226   . ~ .}
1227 %^A https://www.asciart.eu/animals/cats
1228 \AddAnimal{schroedinger-alive}
1229 { \
1230   \ ,_-
1231   |\\_-,'/
1232   / _ _ | ,--.
1233   ( @ @ ) / ,-'
1234   \ _T_/_._( (
1235   /      ' \
1236   |      - \ |
1237   \ \ , / - |
1238   || | _\ _ /
1239   ((_/ '(____,-'}
1240 </animals>

```

Who's gonna use it anyway?

0

o

>( ' )

) /

/ (

/ ' ----- /

\ ~ = - /

~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^

Hosted at  
[https://github.com/Skillmon/ltx\\_ducksay](https://github.com/Skillmon/ltx_ducksay)  
it is.

--.-.-  
'-.\_"7'  
/'.-c  
| /T  
\_)\_/LI