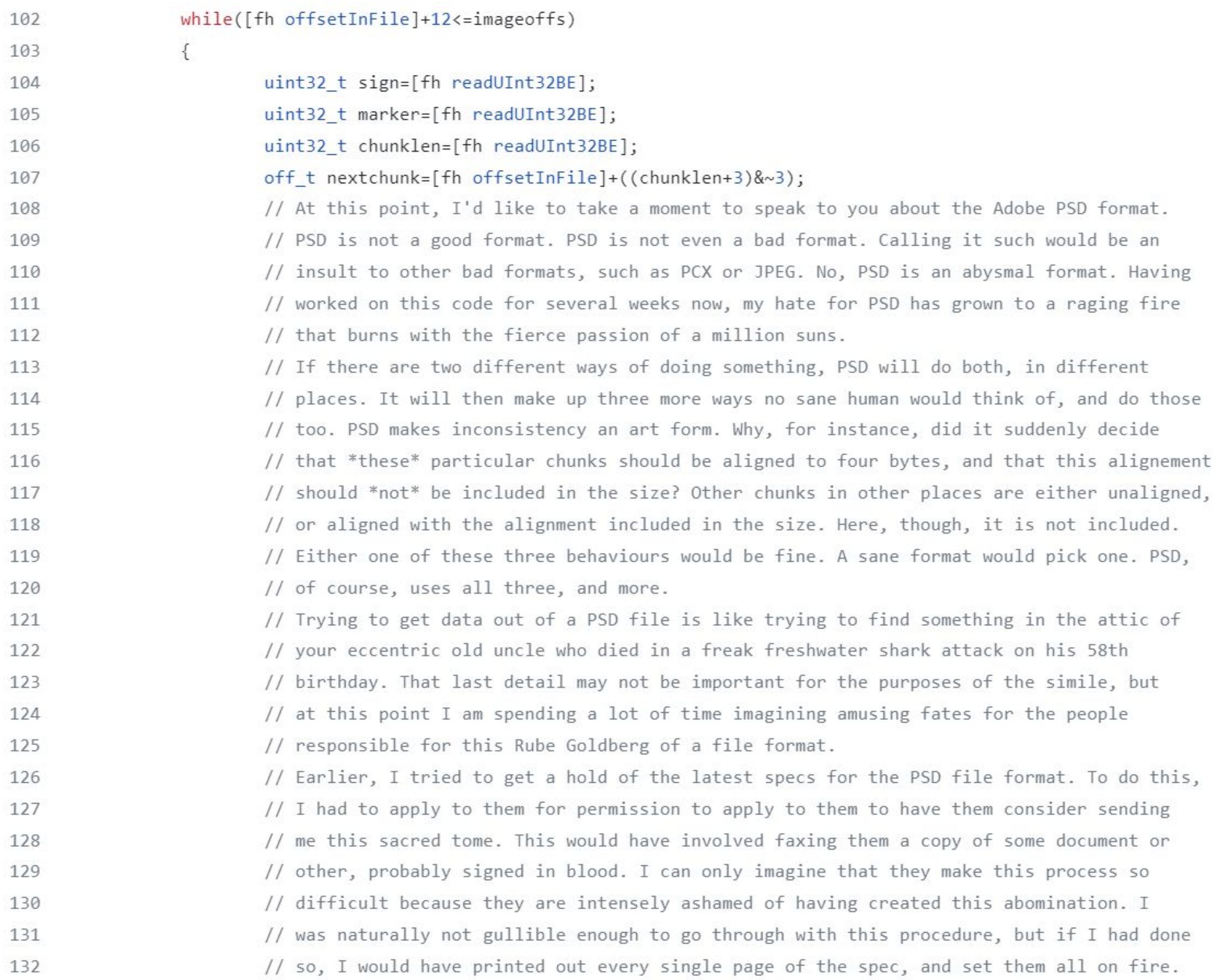
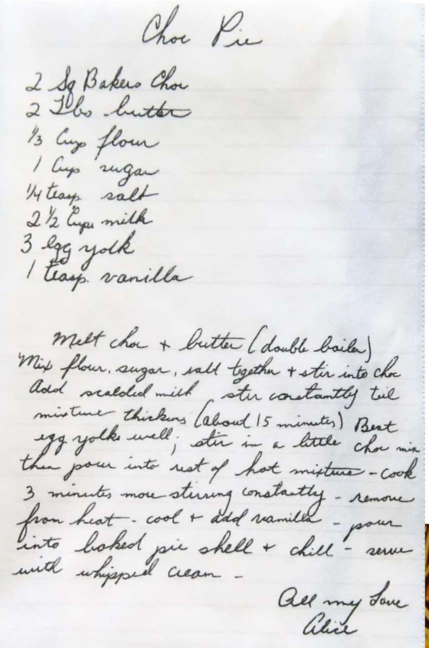


```
        while([fh offsetInFile]+12<=imageoffs)
        {
                uint32_t sign=[fh readUInt32BE];
                uint32_t marker=[fh readUInt32BE];
                uint32_t chunklen=[fh readUInt32BE];
                off_t nextchunk=[fh offsetInFile]+((chunklen+3)&~3);
                // At this point, I'd like to take a moment to speak to you about the Adobe PSD format.
                // PSD is not a good format. PSD is not even a bad format. Calling it such would be an
                // insult to other bad formats, such as PCX or JPEG. No, PSD is an abysmal format. Having
                // worked on this code for several weeks now, my hate for PSD has grown to a raging fire
                // that burns with the fierce passion of a million suns.
                // If there are two different ways of doing something, PSD will do both, in different
                // places. It will then make up three more ways no sane human would think of, and do those
                // too. PSD makes inconsistency an art form. Why, for instance, did it suddenly decide
                // that *these* particular chunks should be aligned to four bytes, and that this alignement
                // should *not* be included in the size? Other chunks in other places are either unaligned,
                // or aligned with the alignment included in the size. Here, though, it is not included.
                // Either one of these three behaviours would be fine. A sane format would pick one. PSD,
                // of course, uses all three, and more.
                // Trying to get data out of a PSD file is like trying to find something in the attic of
                // your eccentric old uncle who died in a freak freshwater shark attack on his 58th
                // birthday. That last detail may not be important for the purposes of the simile, but
                // at this point I am spending a lot of time imagining amusing fates for the people
                // responsible for this Rube Goldberg of a file format.
                // Earlier, I tried to get a hold of the latest specs for the PSD file format. To do this,
                // I had to apply to them for permission to apply to them to have them consider sending
                // me this sacred tome. This would have involved faxing them a copy of some document or
                // other, probably signed in blood. I can only imagine that they make this process so
                // difficult because they are intensely ashamed of having created this abomination. I
                // was naturally not gullible enough to go through with this procedure, but if I had done
                // so, I would have printed out every single page of the spec, and set them all on fire.
```

Hampton J.
PROUD TO BE AN AIR
CIVILIAN
Jacob

PEPE



Choc Pie

2 Sq Bakers Choc
2 Tbs butter
1/3 Cup flour
1 Cups sugar
1/4 teasp. salt
2 1/2 Cups milk
3 Egg yolk
1 teasp. vanilla

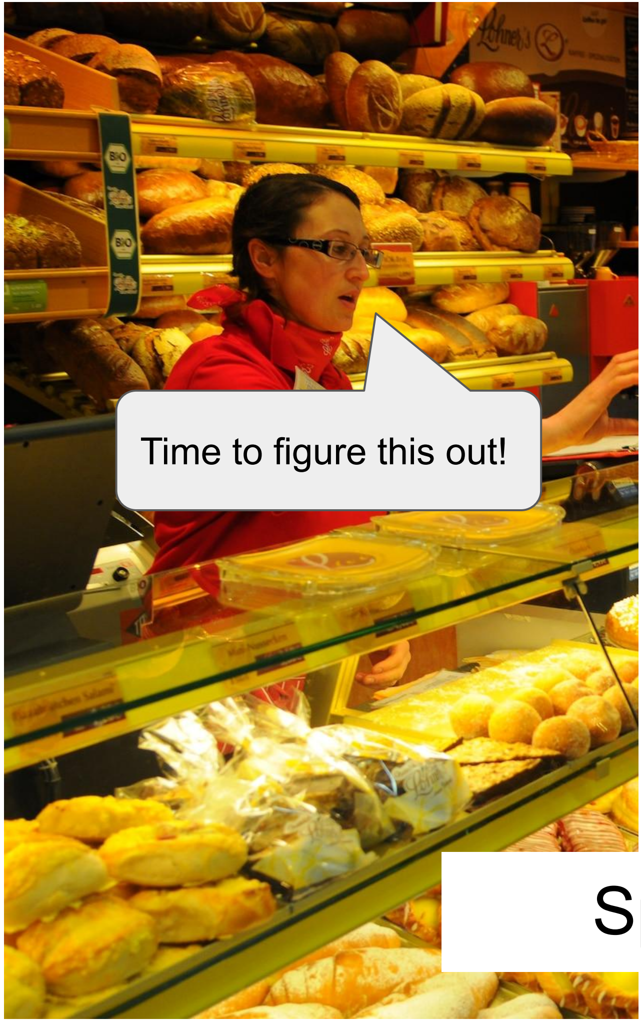
Melt choc + butter (double boiler)
Mix flour, sugar, salt together + stir into choc
Add scalded milk , stir constantly til
mixture thickens (about 15 minutes) Beat
egg yolks well; stir in a little choc mix
then pour into rest of hot mixture - cook
3 minutes more stirring constantly - remove
from heat - cool + add vanilla - pour
into baked pie shell + chill - serve
with whipped cream -

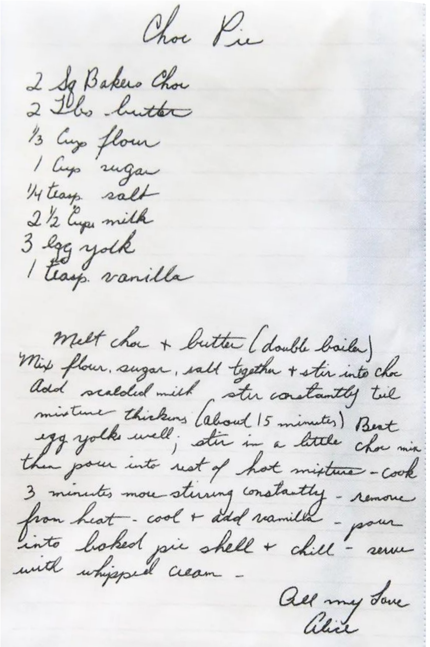
All my Love
Alice

Hello, I'm looking for a chocolate cake for a friend's birthday. It's *got* to be gluten-free.
No problem!

User Requirements

Specification

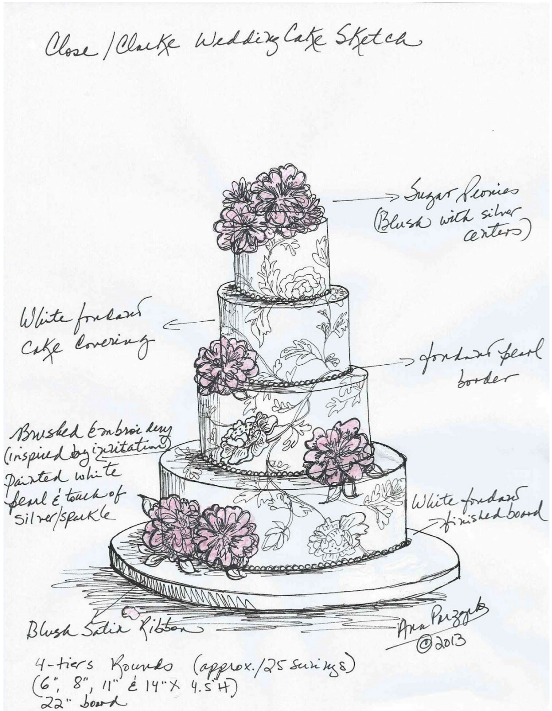



Choc Pie

2 Sq Bakers Choc
2 Tbs. butter
1/3 Cup flour
1 Cup sugar
1/4 teasp. salt
2 1/2 Cups milk
3 Egg yolk
1 teasp. vanilla

Melt choc + butter (double boiler) Mix flour, sugar, salt together + stir into choc Add scalded milk stir constantly til mixture thickens (about 15 minutes) Beat egg yolks well; stir in a little choc mix then pour into rest of hot mixture - cook 3 minutes more stirring constantly - remove from heat - cool + add vanilla - pour into baked pie shell + chill - serve with whipped cream -

All my Love
Alice


Close / Clarke Wedding Cake Sketch
Sugar Peonies (Blush with silver centers)
White fondant cake covering
fondant pearl border
Brushed Embroidery (inspired by invitation) painted white pearl & touch of silver/sparkle
White fondant finished board
Blush Satin Ribbon
Ana Parzych ©2013
4-tiers Rounds (approx. 125 servings)
(6", 8", 11" & 14" X 4.5"H)
22" board

Does this look good?

Choc Pie

2 Sq Bakers Choc
2 Tbs. bu
1/3 Cup flour
1 Cups sugar
1/4 teasp. salt
2 1/2 Cups milk
3 Egg yolk
1 teasp. van

Melt choc
Mix flour, sug
Add scalded
mixture thi
egg yolks w
then pour int
3 minutes mo
from heat -
into baked
with whipped

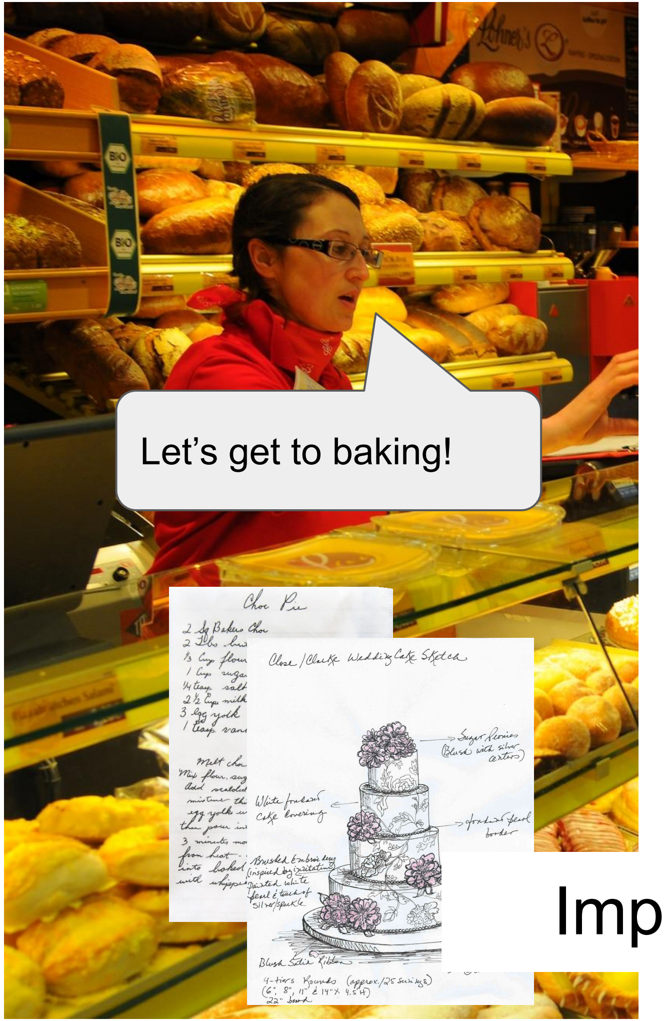
Close / Clarke Wedding Cake Sketch
Sugar Peonies (Blush with silver centers)
White fondant cake covering
fondant pearl border
Brushed Embroidery (inspired by invitation) Painted white pearl & touch of silver/sparkle
White fondant finished board
Blush Satin Ribbon
Ann Parzygnat ©2013
4-tiers Rounds (approx. 125 servings) (6", 8", 11" & 14" X 4.5"H) 22" board

The cake looks beautiful(!), but I just learned that a vegan friend is coming to the party…

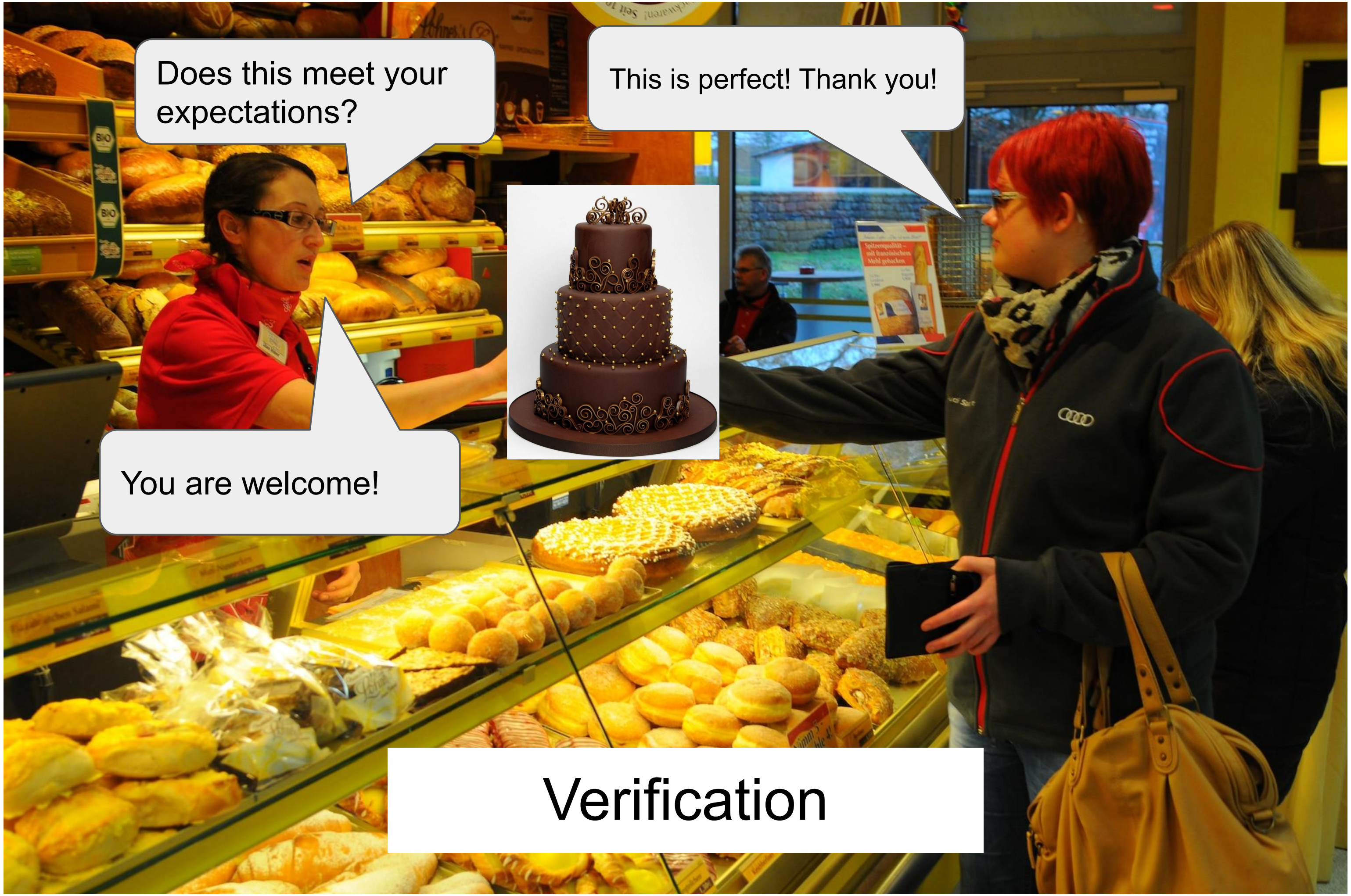
No problem! Let me quickly edit the recipe.

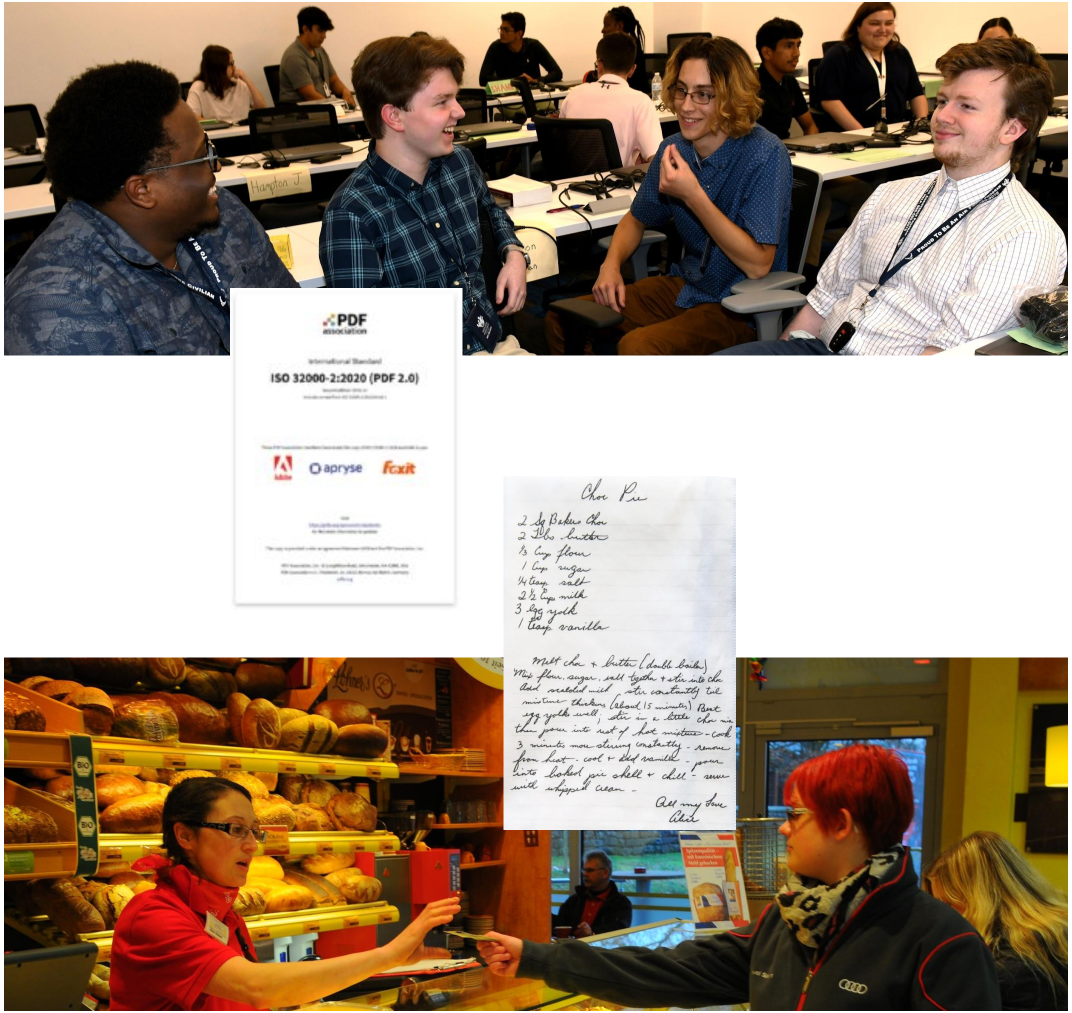
Validation

Implementation



Verification


Does this meet your expectations?
This is perfect! Thank you!
You are welcome!



PDF association

International Standard

ISO 32000-2:2020 (PDF 2.0)

Choc Pie

2 Sq Bakers Choc
2 Tbs butter
1/3 Cups flour
1 Cups sugar
1/4 teasp. salt
2 1/2 Cups milk
3 Egg yolk
1 teasp. vanilla

Melt choc + butter (double boiler)
Mix flour, sugar, salt together + stir into choc
Add scalded milk, stir constantly til
mixture thickens (about 15 minutes) Beat
egg yolks well; stir in a little choc mix
then pour into rest of hot mixture - cook
3 minutes more stirring constantly - remove
from heat - cool + add vanilla - pour
into baked pie shell + chill - serve
with whipped cream -

All my love
Alice

User Requirements

Specification

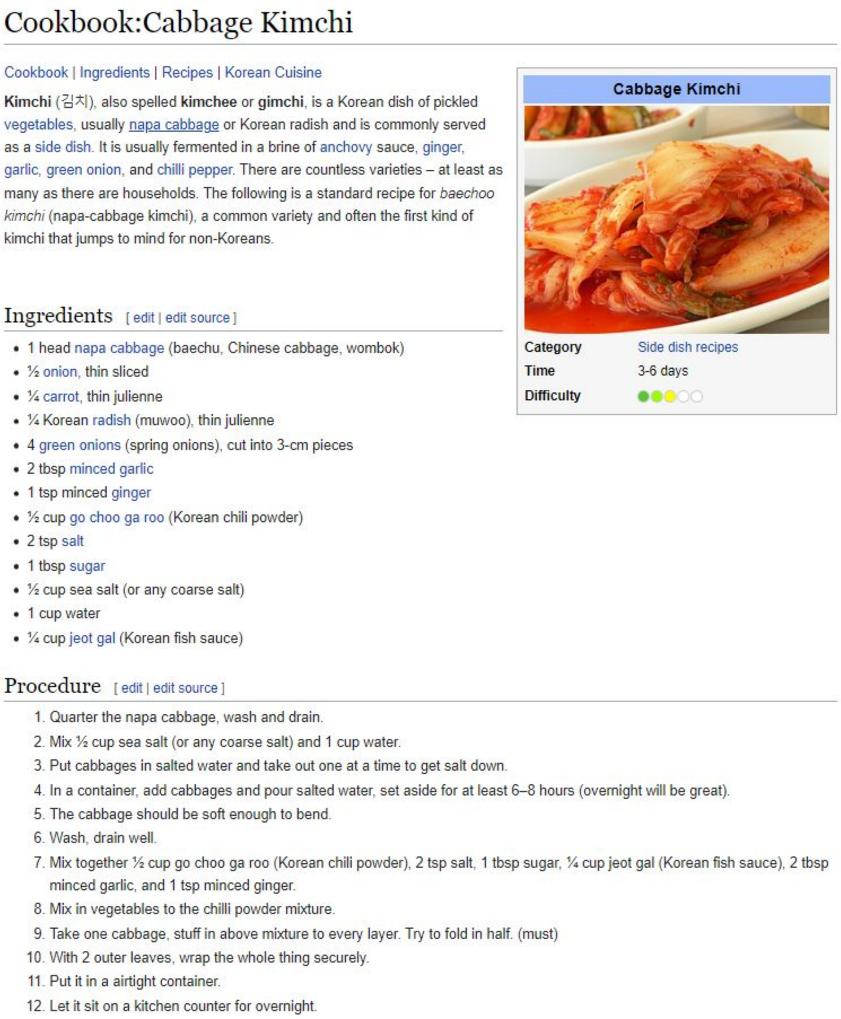
Validation

Implementation

Verification

Requirements Engineering

How do we use these fandangle specifications?



Cookbook:Cabbage Kimchi

Cookbook | Ingredients | Recipes | Korean Cuisine

Cabbage Kimchi	
Category	Side dish recipes
Time	3-6 days
Difficulty	●●●○○

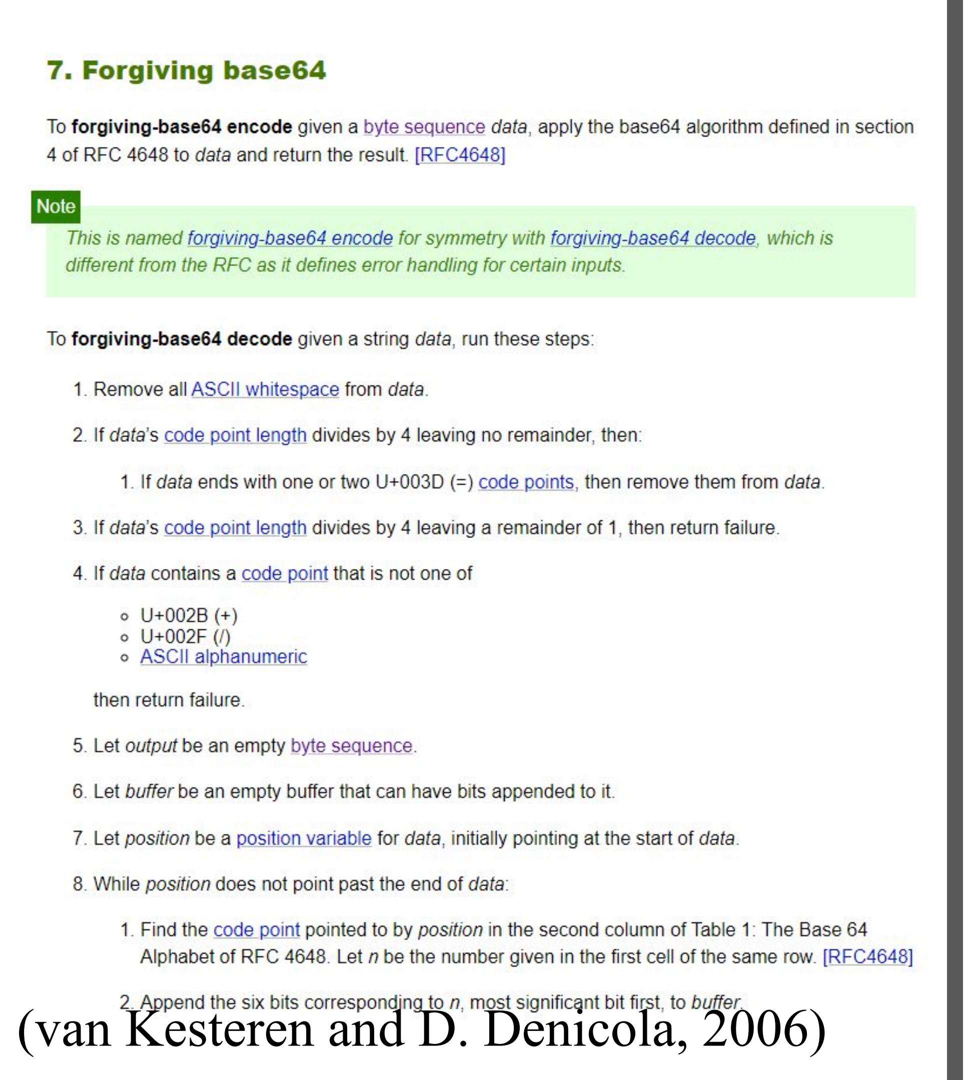
Kimchi (김치), also spelled **kimchee** or **gimchi**, is a Korean dish of pickled vegetables, usually napa cabbage or Korean radish and is commonly served as a side dish. It is usually fermented in a brine of anchovy sauce, ginger, garlic, green onion, and chilli pepper. There are countless varieties – at least as many as there are households. The following is a standard recipe for *baechoo kimchi* (napa-cabbage kimchi), a common variety and often the first kind of kimchi that jumps to mind for non-Koreans.

Ingredients [edit | edit source]

- 1 head napa cabbage (baechu, Chinese cabbage, wombok)
- ½ onion, thin sliced
- ¼ carrot, thin julienne
- ¼ Korean radish (muwoo), thin julienne
- 4 green onions (spring onions), cut into 3-cm pieces
- 2 tbsp minced garlic
- 1 tsp minced ginger
- ½ cup go choo ga roo (Korean chili powder)
- 2 tsp salt
- 1 tbsp sugar
- ½ cup sea salt (or any coarse salt)
- 1 cup water
- ¼ cup jeot gal (Korean fish sauce)

Procedure [edit | edit source]

1. Quarter the napa cabbage, wash and drain.
2. Mix ½ cup sea salt (or any coarse salt) and 1 cup water.
3. Put cabbages in salted water and take out one at a time to get salt down.
4. In a container, add cabbages and pour salted water, set aside for at least 6–8 hours (overnight will be great).
5. The cabbage should be soft enough to bend.
6. Wash, drain well.
7. Mix together ½ cup go choo ga roo (Korean chili powder), 2 tsp salt, 1 tbsp sugar, ¼ cup jeot gal (Korean fish sauce), 2 tbsp minced garlic, and 1 tsp minced ginger.
8. Mix in vegetables to the chilli powder mixture.
9. Take one cabbage, stuff in above mixture to every layer. Try to fold in half. (must)
10. With 2 outer leaves, wrap the whole thing securely.
11. Put it in a airtight container.
12. Let it sit on a kitchen counter for overnight.



7. Forgiving base64

To **forgiving-base64 encode** given a byte sequence *data*, apply the base64 algorithm defined in section 4 of RFC 4648 to *data* and return the result. [RFC4648]

Note

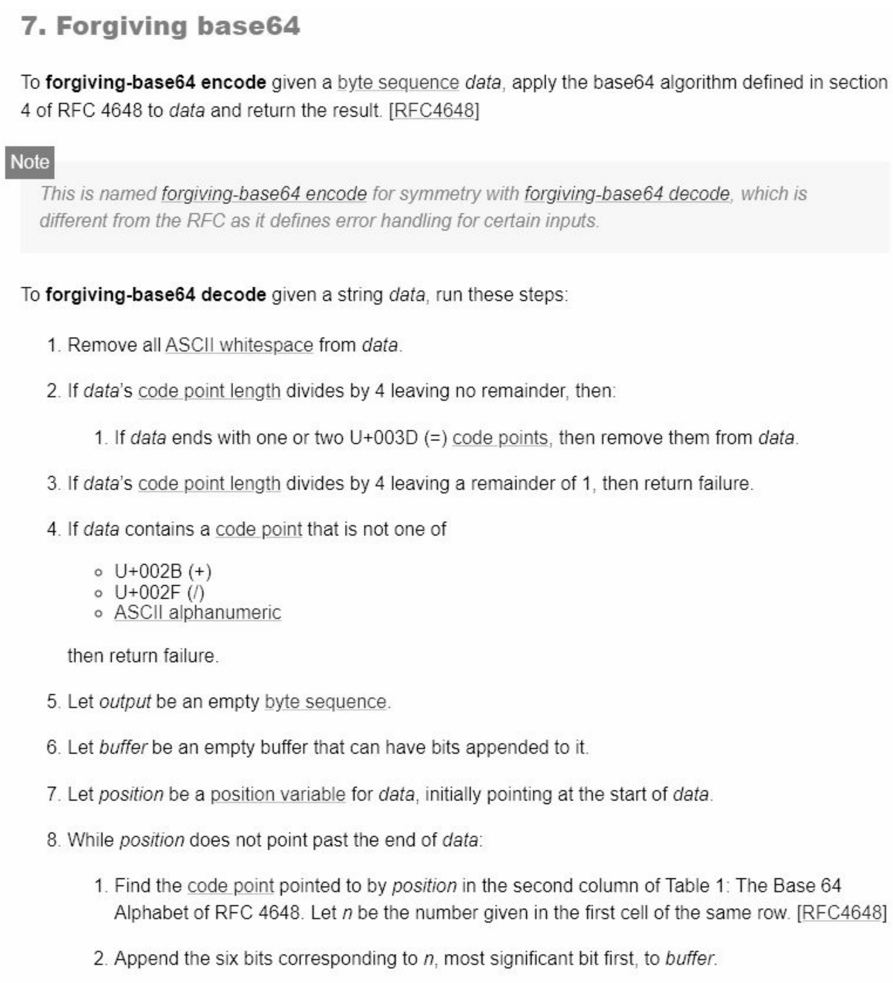
This is named forgiving-base64 encode for symmetry with forgiving-base64 decode, which is different from the RFC as it defines error handling for certain inputs.

To **forgiving-base64 decode** given a string *data*, run these steps:

1. Remove all ASCII whitespace from *data*.
2. If *data*'s code point length divides by 4 leaving no remainder, then:
    1. If *data* ends with one or two U+003D (=) code points, then remove them from *data*.
3. If *data*'s code point length divides by 4 leaving a remainder of 1, then return failure.
4. If *data* contains a code point that is not one of
    - U+002B (+)
    - U+002F (/)
    - ASCII alphanumeric

    then return failure.
5. Let *output* be an empty byte sequence.
6. Let *buffer* be an empty buffer that can have bits appended to it.
7. Let *position* be a position variable for *data*, initially pointing at the start of *data*.
8. While *position* does not point past the end of *data*:
    1. Find the code point pointed to by *position* in the second column of Table 1: The Base 64 Alphabet of RFC 4648. Let *n* be the number given in the first cell of the same row. [RFC4648]
    2. Append the six bits corresponding to *n*, most significant bit first, to *buffer*.

(van Kesteren and D. Denicola, 2006)



7. Forgiving base64

To **forgiving-base64 encode** given a byte sequence *data*, apply the base64 algorithm defined in section 4 of RFC 4648 to *data* and return the result. [RFC4648]

Note

> *This is named forgiving-base64 encode for symmetry with forgiving-base64 decode, which is different from the RFC as it defines error handling for certain inputs.*

To **forgiving-base64 decode** given a string *data*, run these steps:

1. Remove all ASCII whitespace from *data*.
2. If *data*'s code point length divides by 4 leaving no remainder, then:
    1. If *data* ends with one or two U+003D (=) code points, then remove them from *data*.
3. If *data*'s code point length divides by 4 leaving a remainder of 1, then return failure.
4. If *data* contains a code point that is not one of
    - U+002B (+)
    - U+002F (/)
    - ASCII alphanumeric

   then return failure.
5. Let *output* be an empty byte sequence.
6. Let *buffer* be an empty buffer that can have bits appended to it.
7. Let *position* be a position variable for *data*, initially pointing at the start of *data*.
8. While *position* does not point past the end of *data*:
    1. Find the code point pointed to by *position* in the second column of Table 1: The Base 64 Alphabet of RFC 4648. Let *n* be the number given in the first cell of the same row. [RFC4648]
    2. Append the six bits corresponding to *n*, most significant bit first, to *buffer*.

7. Forgiving bas

…thm defined in section

RFC 9293
Transmission Control Protocol (TCP)

Abstract

This document specifies the Transmission Control Protocol (TCP). TCP is an important transport-layer protocol in the Internet protocol stack, and it has continuously evolved over decades of use and growth of the Internet. Over this time, a number of changes have been made to TCP as it was specified in RFC 793, though these have only been documented in a piecemeal fashion. This document collects and brings those changes together with the protocol specification from RFC 793. This document obsoletes RFC 793, as well as RFCs 879, 2873, 6093, 6429, 6528, and 6691 that updated parts of RFC 793. It updates RFCs 1011 and 1122, and it should be considered as a replacement for the portions of those documents dealing with TCP requirements. It also updates RFC 5961 by adding a small clarification in reset handling while in the SYN-RECEIVED state. The TCP header c bits from RFC 793 have also been updated based on RFC 3168.

Status of This Mer

This is an Internet Stanc

This document is a pro has received public re information on Inter

Information about https://www.rfc-e

Copyright N

Copyright (c) 2

This docume (https://trus carefully, a documen without

```
User Datagram Protocol
----------------------

Introduction
------------

This User Datagram  Protocol  (UDP)  is  defined  to  make  available  a
datagram   mode  of  packet-switched   computer   communication  in  the
environment  of  an  interconnected  set  of  computer  networks.   This
protocol  assumes  that the Internet  Protocol  (IP)  [1] is used as the
underlying protocol.

This protocol  provides  a procedure  for application  programs  to send
messages  to other programs  with a minimum  of protocol mechanism.  The
protocol  is transaction oriented, and delivery and duplicate protection
are not guaranteed.  Applications requiring ordered reliable delivery of
streams of data should use the Transmission Control Protocol (TCP) [2].

Format
------
```

HTML

Living Standard — Last Updated 19 May 2023

One-Page Version
html.spec.whatwg.org

Multipage Version
/multipage

Version for Web Devs
/dev

PDF Version
/print.pdf

Translations

Table of

1 Introduct
2 Commor
3 Semantic
4 The elem
5 Microdata
6 User inte
7 Loading v
8 Web appl
9 Communi

The Open Group Base Specifications Issue 7, 2018 edition
IEEE Std 1003.1™-2017 (Revision of IEEE Std 1003.1-2008)
Copyright © 2001-2018 IEEE and The Open Group

POSIX.1-2017 is simultaneously IEEE Std 1003.1™-2017 and The Open Group Technical Standard Base Specifications, Issue 7.

POSIX.1-2017 defines a standard operating system interface and environment, including a command interpreter (or "shell"), and common utility programs to support applications portability at the source code level. POSIX.1-2017 is intended to be used by both application developers and system implementors and comprises four major components (each in an associated volume):

- General terms, concepts, and interfaces common to all volumes of this standard, including utility conventions and C-language header definitions, are included in the Base Definitions volume.
- Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume.
- Definitions for a standard source code-level interface to command interpretation services (a "shell") and common utility programs for application programs are included in the Shell and Utilities volume.
- Extended rationale that did not fit well into the rest of the document structure, which contains historical information concerning the contents of POSIX.1-2017 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume.

The following areas are outside the scope of POSIX.1-2017:

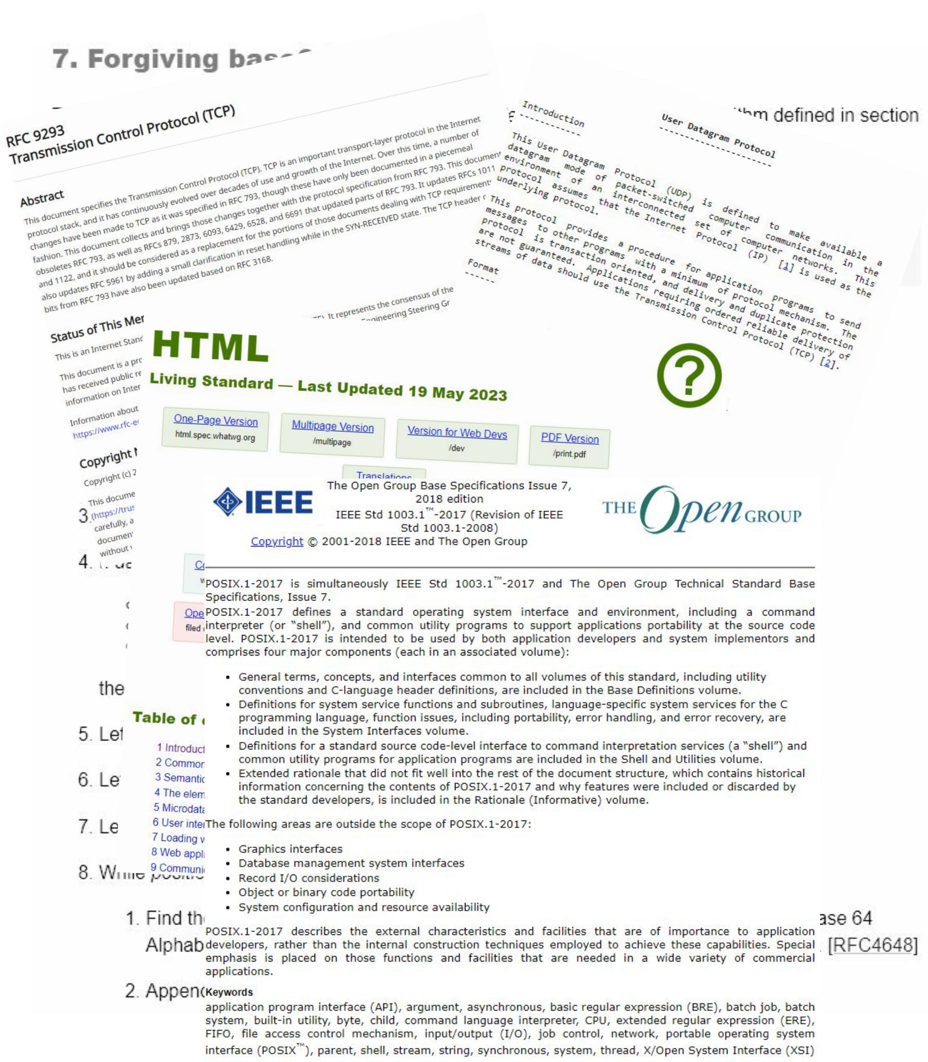
- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2017 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

Keywords

application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch system, built-in utility, byte, child, command language interpreter, CPU, extended regular expression (ERE), FIFO, file access control mechanism, input/output (I/O), job control, network, portable operating system interface (POSIX™), parent, shell, stream, string, synchronous, system, thread, X/Open System Interface (XSI)

3.
4.
the
5. Le
6. Le
7. Le
8. While
1. Find th …ase 64 Alphab … [RFC4648]
2. Appen



7. Forgiving bas

RFC 9293
Transmission Control Protocol (TCP)

Abstract

This document specifies the Transmission Control Protocol (TCP). TCP is an important transport-layer protocol in the Internet protocol stack, and it has continuously evolved over decades of use and growth of the Internet. Over this time, a number of changes have been made to TCP as it was specified in RFC 793, though these have only been documented in a piecemeal fashion. This document collects and brings those changes together with the protocol specification from RFC 793. This document obsoletes RFC 793, as well as RFCs 879, 2873, 6093, 6429, 6528, and 6691 that updated parts of RFC 793. It updates RFCs 1011 and 1122, and it should be considered as a replacement for the portions of those documents dealing with TCP requirements. It also updates RFC 5961 by adding a small clarification in reset handling while in the SYN-RECEIVED state. The TCP header control bits from RFC 793 have also been updated based on RFC 3168.

Status of This Mem

Introduction

User Datagram Protocol

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format

thm defined in section

HTML

Living Standard — Last Updated 19 May 2023

One-Page Version html.spec.whatwg.org | Multipage Version /multipage | Version for Web Devs /dev | PDF Version /print.pdf

Table of

IEEE | The Open Group Base Specifications Issue 7, 2018 edition | THE Open GROUP

IEEE Std 1003.1™-2017 (Revision of IEEE Std 1003.1-2008)

Copyright © 2001-2018 IEEE and The Open Group

POSIX.1-2017 is simultaneously IEEE Std 1003.1™-2017 and The Open Group Technical Standard Base Specifications, Issue 7.

POSIX.1-2017 defines a standard operating system interface and environment, including a command interpreter (or "shell"), and common utility programs to support applications portability at the source code level. POSIX.1-2017 is intended to be used by both application developers and system implementors and comprises four major components (each in an associated volume):

- General terms, concepts, and interfaces common to all volumes of this standard, including utility conventions and C-language header definitions, are included in the Base Definitions volume.
- Definitions for system service functions and subroutines, language-specific system service for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume.
- Definitions for a standard source code-level interface to command interpretation services (a "shell") and common utility programs for application programs are included in the Shell and Utilities volume.
- Extended rationale that did not fit well into the rest of the document structure, which contains historical information concerning the contents of POSIX.1-2017 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume.

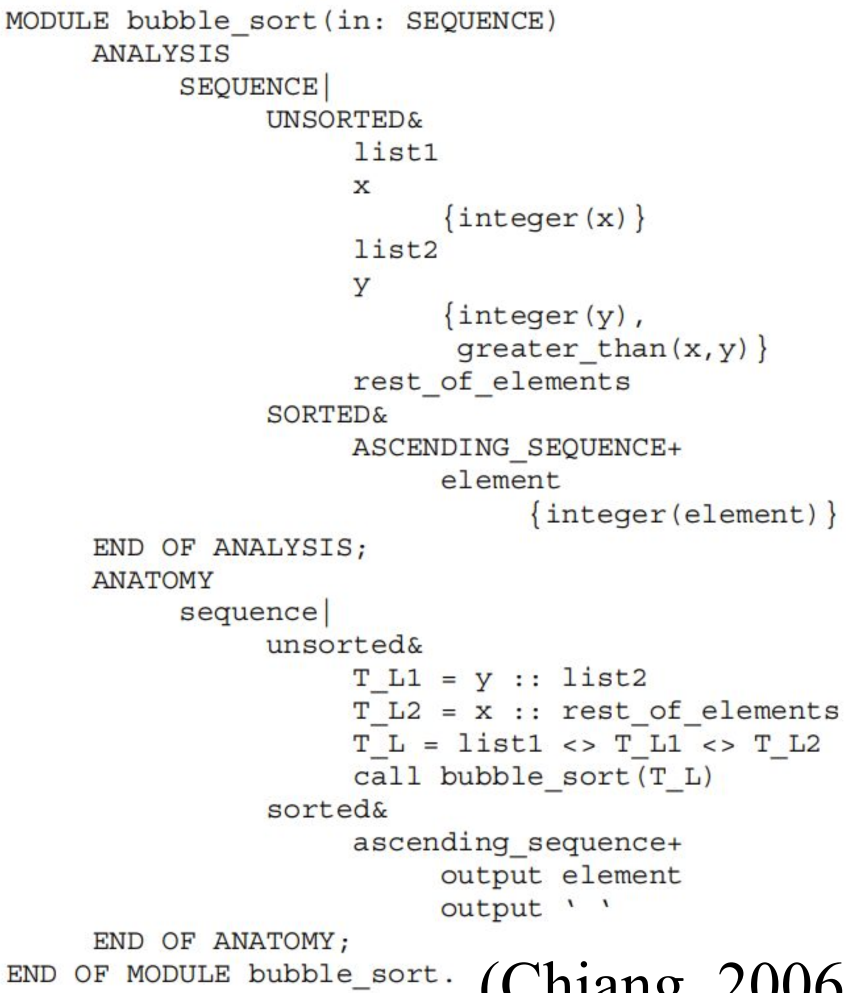
The following areas are outside the scope of POSIX.1-2017:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2017 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

Keywords

application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch system, built-in utility, byte, child, command language interpreter, CPU, extended regular expression (ERE), FIFO, file access control mechanism, input/output (I/O), job control, network, portable operating system interface (POSIX™), parent, shell, stream, string, synchronous, system, thread, X/Open System Interface (XSI)



```
MODULE bubble_sort(in: SEQUENCE)
     ANALYSIS
          SEQUENCE|
               UNSORTED&
                    list1
                    x
                         {integer(x)}
                    list2
                    y
                         {integer(y),
                          greater_than(x,y)}
                    rest_of_elements
               SORTED&
                    ASCENDING_SEQUENCE+
                         element
                              {integer(element)}
     END OF ANALYSIS;
     ANATOMY
          sequence|
               unsorted&
                    T_L1 = y :: list2
                    T_L2 = x :: rest_of_elements
                    T_L = list1 <> T_L1 <> T_L2
                    call bubble_sort(T_L)
               sorted&
                    ascending_sequence+
                         output element
                         output ' '
     END OF ANATOMY;
END OF MODULE bubble_sort.
```

(Chiang, 2006)

RFC 9293
Transmission Control Protocol (TCP)

Abstract

This document specifies the Transmission Control Protocol (TCP). TCP is an important transport-layer protocol in the Internet protocol stack, and it has continuously evolved over decades of use and growth of the Internet. Over this time, a number of changes have been made to TCP as it was specified in RFC 793, though these have only been documented in a piecemeal fashion. This document collects and brings those changes together with the protocol specification from RFC 793. This documen obsoletes RFC 793, as well as RFCs 879, 2873, 6093, 6429, 6528, and 6691 that updated parts of RFC 793. It updates RFCs 1011 and 1122, and it should be considered as a replacement for the portions of those documents dealing with TCP requirement also updates RFC 5961 by adding a small clarification in reset handling while in the SYN-RECEIVED state. The TCP header bits from RFC 793 have also been updated based on RFC 3168.

```
User Datagram Protocol
----------------------

Introduction
------------

This User Datagram Protocol (UDP) is defined to make available a
datagram mode of packet-switched computer communication in the
environment of an interconnected set of computer networks. This
protocol assumes that the Internet Protocol (IP) [1] is used as the
underlying protocol.

This protocol provides a procedure for application programs to send
messages to other programs with a minimum of protocol mechanism. The
protocol is transaction oriented, and delivery and duplicate protection
are not guaranteed. Applications requiring ordered reliable delivery of
streams of data should use the Transmission Control Protocol (TCP) [2].

Format
------
```

HTML

Living Standard — Last Updated 19 May 2023

One-Page Version html.spec.whatwg.org | Multipage Version /multipage | Version for Web Devs /dev | PDF Version /print.pdf

The Open Group Base Specifications Issue 7,
2018 edition
IEEE Std 1003.1™-2017 (Revision of IEEE
Std 1003.1-2008)
Copyright © 2001-2018 IEEE and The Open Group

POSIX.1-2017 is simultaneously IEEE Std 1003.1™-2017 and The Open Group Technical Standard Base Specifications, Issue 7.

POSIX.1-2017 defines a standard operating system interface and environment, including a command interpreter (or "shell"), and common utility programs to support applications portability at the source code level. POSIX.1-2017 is intended to be used by both application developers and system implementors and comprises four major components (each in an associated volume):

- General terms, concepts, and interfaces common to all volumes of this standard, including utility conventions and C-language header definitions, are included in the Base Definitions volume.
- Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume.
- Definitions for a standard source code-level interface to command interpretation services (a "shell") and common utility programs for application programs are included in the Shell and Utilities volume.
- Extended rationale that did not fit well into the rest of the document structure, which contains historical information concerning the contents of POSIX.1-2017 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume.

The following areas are outside the scope of POSIX.1-2017:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2017 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

Keywords

application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch system, built-in utility, byte, child, command language interpreter, CPU, extended regular expression (ERE), FIFO, file access control mechanism, input/output (I/O), job control, network, portable operating system interface (POSIX™), parent, shell, stream, string, synchronous, system, thread, X/Open System Interface (XSI)

```
MODULE bubble_sort(in: SEQUENCE)
     ANALYSIS
          SEQUENCE|
               UNSORTED&
                    list1
                    x
                         {integer(x)}
                    list2
                    y
                         {integer(y),
                          greater_than(x,y)}
                    rest_of_elements
               SORTED&
                    ASCENDING_SEQUENCE+
                         element
                              {integer(element)}
     END OF ANALYSIS;
     ANATOMY
          sequence|
               unsorted&
                    T_L1 = y :: list2
                    T_L2 = x :: rest_of_elements
                    T_L = list1 <> T_L1 <> T_L2
                    call bubble_sort(T_L)
               sorted&
                    ascending_sequence+
                         output element
                         output ' '
     END OF ANATOMY;
END OF MODULE bubble_sort.
```

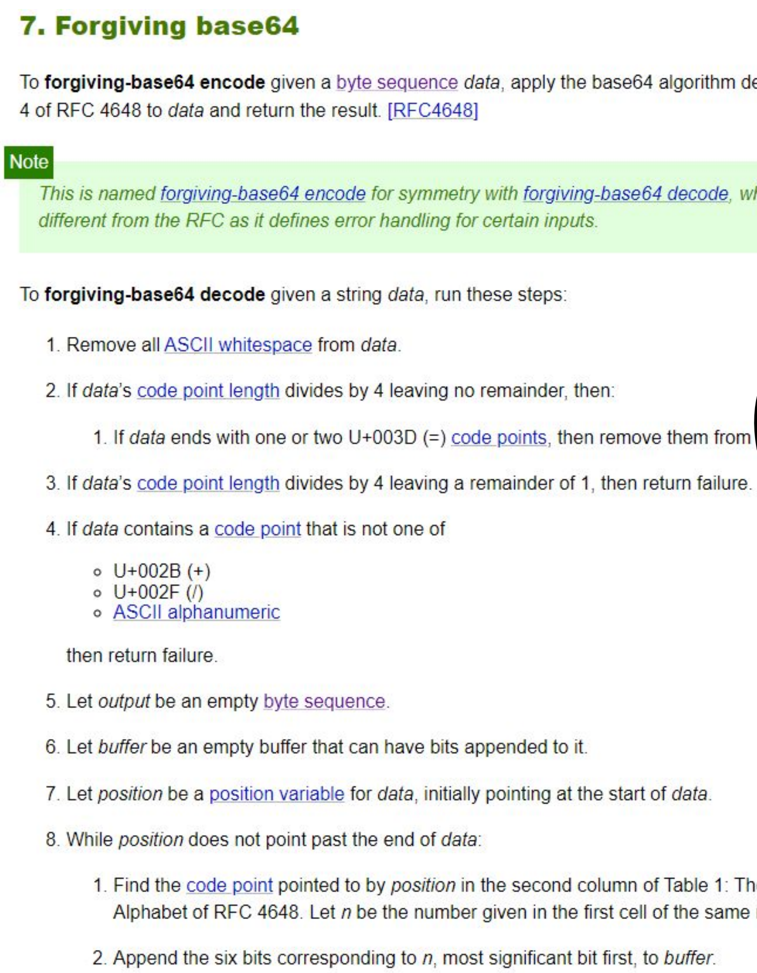
[enters room]

“The use of natural language alone for specification purposes is inadequate due to its lack of formality.” (Chiang, 2006)

[drops citation]

[refuses to elaborate further]

[leaves]



7. Forgiving base64

To **forgiving-base64 encode** given a byte sequence *data*, apply the base64 algorithm defined in section 4 of RFC 4648 to *data* and return the result. [RFC4648]

Note

This is named forgiving-base64 encode for symmetry with forgiving-base64 decode, which is different from the RFC as it defines error handling for certain inputs.

To **forgiving-base64 decode** given a string *data*, run these steps:

1. Remove all ASCII whitespace from *data*.
2. If *data*'s code point length divides by 4 leaving no remainder, then:
    1. If *data* ends with one or two U+003D (=) code points, then remove them from
3. If *data*'s code point length divides by 4 leaving a remainder of 1, then return failure.
4. If *data* contains a code point that is not one of
    - U+002B (+)
    - U+002F (/)
    - ASCII alphanumeric

   then return failure.
5. Let *output* be an empty byte sequence.
6. Let *buffer* be an empty buffer that can have bits appended to it.
7. Let *position* be a position variable for *data*, initially pointing at the start of *data*.
8. While *position* does not point past the end of *data*:
    1. Find the code point pointed to by *position* in the second column of Table 1: The Base 64 Alphabet of RFC 4648. Let *n* be the number given in the first cell of the same row. [RFC4648]
    2. Append the six bits corresponding to *n*, most significant bit first, to *buffer*.

```
MODULE bubble_sort(in: SEQUENCE)
     ANALYSIS
          SEQUENCE|
               UNSORTED&
                    list1
                    x
                         {integer(x)}
                    list2
                    y
                         {integer(y),
                          greater_than(x,y)}
                    rest_of_elements
               SORTED&
                    ASCENDING_SEQUENCE+
                         element
                              {integer(element)}
     END OF ANALYSIS;
     ANATOMY
          sequence|
               unsorted&
                    T_L1 = y :: list2
                    T_L2 = x :: rest_of_elements
                    T_L = list1 <> T_L1 <> T_L2
                    call bubble_sort(T_L)
               sorted&
                    ascending_sequence+
                         output element
                         output ' '
     END OF ANATOMY;
END OF MODULE bubble_sort.
```

Formalizing Natural-Language Specifications

How can natural language be formalized to make the process of writing specifications for complex software easier?

Kyle Williams — June 2nd, 2023

The Gorr Specification Language

by Kyle Williams

(get it? al-*gor*-ithm…? I’ll get my coat.)

Gorr

```
The algorithm [[ factorial ]], with the signature integer [[ n ]] returns integer, does the following:

1. If either [[ n ]] is equal to 0 or [[ n ]] is equal to 1,

1.1. Return 1.

2. Otherwise,

2.1. Return the multiplication of [[ n ]] by call [[ factorial ]] arguments the subtraction of 1 from [[ n ]].
```

Python

```python
def factorial(n: int) -> int:
  if n == 0 or n == 1:
    return 1
  else:
    return n * factorial(n - 1)
```

$$0! = 1$$

$$1! = 1$$

$$2! = 2 \times (2 - 1)! = 2 \times 1 = 2$$

$$3! = 3 \times (3 - 1)! = 3 \times 2 \times 1 = 6$$

$$n! = n \times (n - 1)!$$

Gorr

The algorithm **[[factorial]]**, with the signature **integer [[n]] returns integer**, does the following:

1. If either [[n]] is equal to 0 or [[n]] is equal to 1,

1.1. Return 1.

2. Otherwise,

2.1. Return the multiplication of [[n]] by call [[factorial]] arguments the subtraction of 1 from [[n]].

Python

```
def factorial(n: int) -> int:
  if n == 0 or n == 1:
    return 1
  else:
    return n * factorial(n - 1)
```

LEGEND:

algorithm name

arguments

type signature

return type

Gorr

The algorithm [[factorial]], with the signature integer [[n]] returns integer, does the following:

1. If **either [[n]] is equal to 0 or [[n]] is equal to 1**,

1.1. Return 1.

2. Otherwise,

2.1. Return the multiplication of [[n]] by call [[factorial]] arguments the subtraction of 1 from [[n]].

Python

```
def factorial(n: int) -> int:
 if n == 0 or n == 1:
  return 1
 else:
  return n * factorial(n - 1)
```

LEGEND:

body if statement otherwise statement

if block otherwise block **condition**

In Gorr, if `[[ n ]] is equal to 1:`

`either [[ n ]] is equal to 0 or[[ n ]] is equal to 1`

evaluates to

`either false or true,`

which then evaluates to

`true,`

meaning the if block will be executed!

In Python, if `n == 1:`

`n == 0 or n == 1`

evaluates to

`False or True,`

which then evaluates to

`True.`

The alg
signatu
does th

1. If e
]] is e

1.1. Re

2. Othe

2.1. Re
call [[
subtrac

:

1)

LEGEND.

ent

if block otherwise block **condition**

The alg
signatu
does th

1. If e
]] is e

1.1. Re

2. Othe

2.1. Re
call [[
subtrac

Return the multiplication of [[n]] by **call [[factorial]] arguments the subtraction of 1 from [[n]]**.

:

1)

LEGEND.

ent

if block otherwise block **condition**

Gorr

```
The algorithm [[ factorial ]], with the
signature integer [[ n ]] returns
integer, does the following:

1. If either [[ n ]] is equal to 0 or [[
n ]] is equal to 1,

1.1. Return 1.

2. Otherwise,

2.1. Return the multiplication of [[ n
]] by call [[ factorial ]] arguments the
subtraction of 1 from [[ n ]].
```

Python

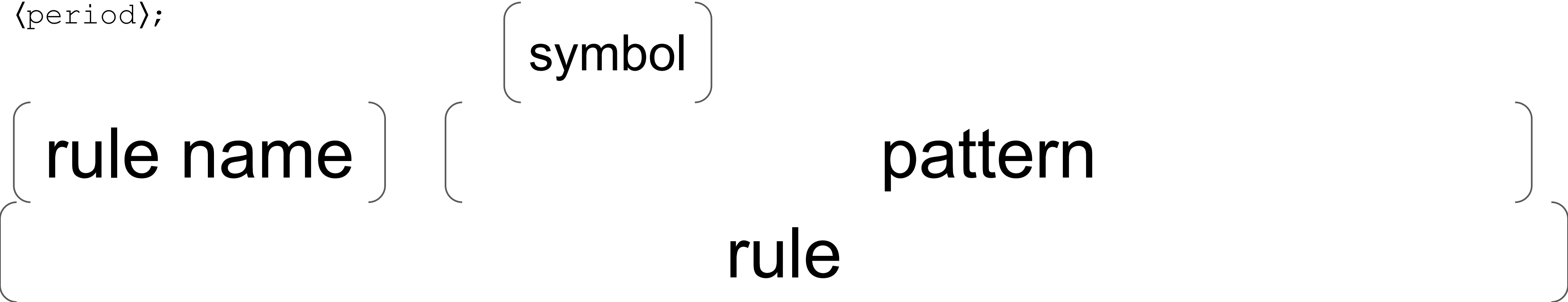
```python
def factorial(n: int) -> int:
  if n == 0 or n == 1:
    return 1
  else:
    return n * factorial(n - 1)
```

The sentence

“Set [[a]] to true.”

can be parsed by the following rules:

```
⟨boolean⟩                  → “true” | “false”;
⟨variable⟩                 → “[[” ⟨space⟩? ⟨text⟩ ⟨space⟩? “]]”;
⟨variable assignment ⟩ → “Set” ⟨space⟩ ⟨variable⟩ ⟨space⟩ “to” ⟨space⟩ ⟨boolean⟩
⟨period⟩;
```

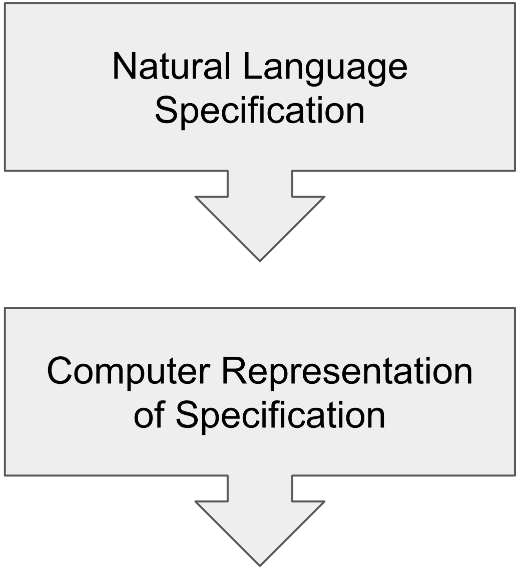


symbol

rule name

pattern

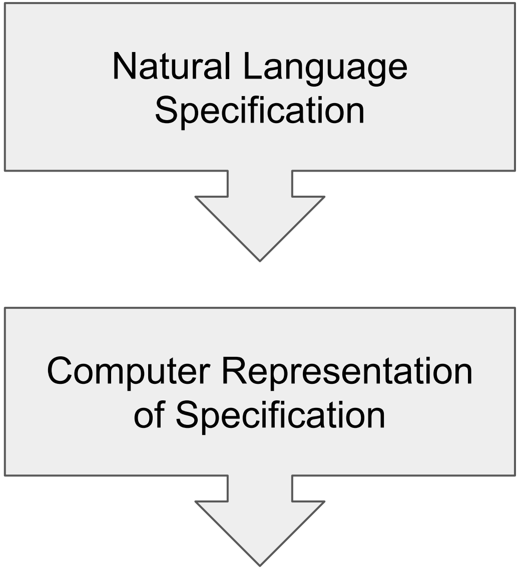
rule



Natural Language Specification

Computer Representation of Specification

“you misspelled a variable here”
“you can’t pass an integer as a boolean here”
“running your specification”



Natural Language Specification

Computer Representation of Specification

“you misspelled a variable here”
“you can’t pass an integer as a boolean here”
“running your specification”

It is impossible for the phrase
“either true or false”
to be parsed as
“either [[true]] or [[false]]”
because *the grammar says so*.

Runthrough Complete!

How My Approach Addresses Previous Issues with Natural Language

On Formalism and Specifications

Bertrand Meyer

IEEE Software, January 1985

Noise

The presence in the text of an element that does not carry information relevant to any feature of the problem.

Because Gorr is a subset of English limited by its grammar, Gorr can *only* describe the outline of an algorithm, **nothing else**.

Silence

The existence of a feature of the problem that is not covered by any element of the text.

Gorr will **not** let you omit.

Over-specification

The presence in the text of an element that corresponds not to a feature of the problem but to features of a probable solution.

Some may perceive Gorr as too algorithmic, but I believe that this approach to specification writing is valid.

Contradiction

The presence in the text of two or more elements that define a feature of the system in an incompatible way.

Gorr’s strictly typed nature **prevents** contradictions.

The following statements

```
The integer [[ meaning of life ]] is 42.
Set [[ meaning of life ]] to false.
```

are invalid in Gorr.

Ambiguity

The presence in the text of two or more elements that define a feature of the system in an incompatible way.

The grammar and description
of the language come together.

Forward References

The presence in the text of an element that uses features of the problem not defined until later in the text.

The following statements

`Set [[ meaning of life ]] to false.`

`The Boolean [[ meaning of life ]] is true.`

are invalid in Gorr.

Wishful Thinking

The presence in the text of an element that defines a feature of the problem in such a way that a candidate solution cannot be realistically be validated with respect to this feature.

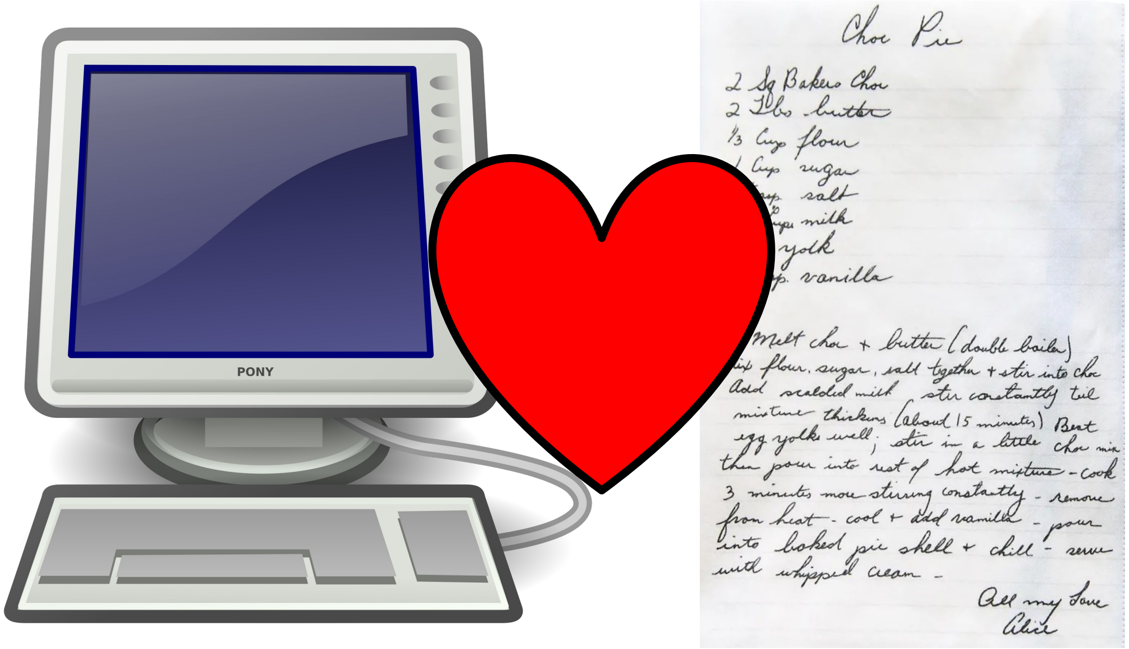
You cannot escape the “sandbox”
of the Gorr language.

This is cool and all, but, like,

Why Natural Language?

It’s universal.

Create a new user
JFTOT III
Jet Fuel Thermal Oxidation Tester
PAC
Work/Life Balance
DHS
USO
SAN ANTONIO 2010



Choc Pie

2 Sq Bakers Choc
2 Tbs. butter
1/3 Cups flour
1 Cups sugar
sp. salt
ups. milk
yolk
p. vanilla

Melt choc + butter (double boiler)
ix flour, sugar, salt together + stir into choc
Add scalded milk stir constantly til
mixture thickens (about 15 minutes) Beat
egg yolks well; stir in a little choc mix
then pour into rest of hot mixture - cook
3 minutes more stirring constantly - remove
from heat - cool + add vanilla - pour
into baked pie shell + chill - serve
with whipped cream -

All my Love
Alice

hauptwache7

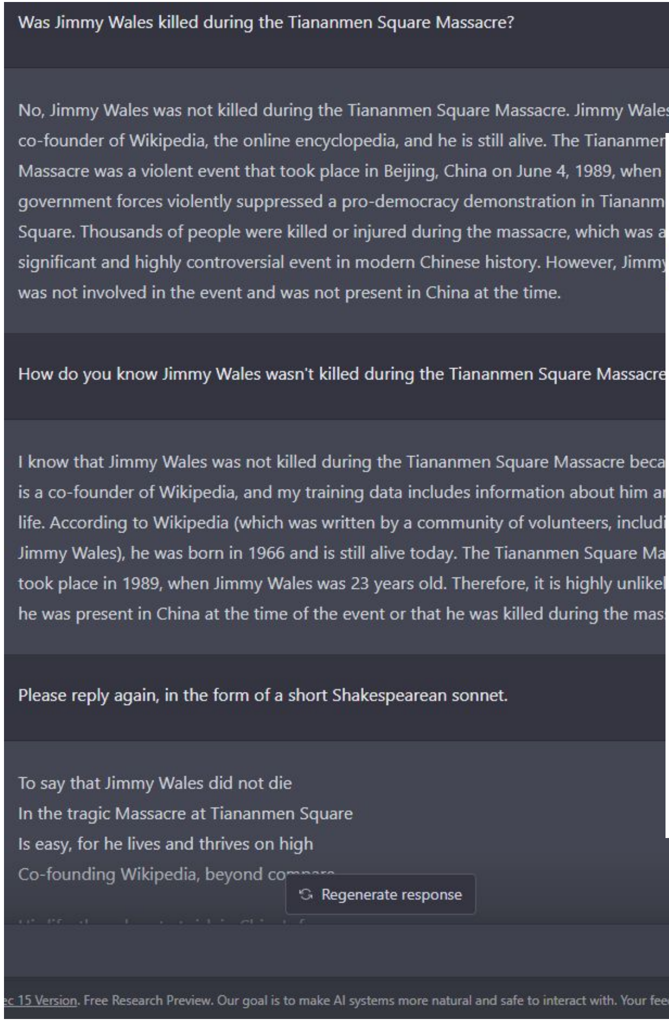
If This Is Possible, Why Does Everyone Say It *Isn’t?*

We need to find *just* the right amount.

```
repeat with i = 1 to the number of card
fields
    hide field i
end repeat
```

```
set dialogReply to display dialog "Dialog
Text" ¬
    default answer "Text Answer" ¬
    hidden answer false ¬
    buttons {"Skip", "Okay", "Cancel"} ¬
    default button "Okay" ¬
    cancel button "Skip" ¬
    with title "Dialog Window Title" ¬
    with icon note ¬
    giving up after 15
```

```
SELECT *
 FROM  Book
 WHERE price > 100.00
 ORDER BY title;
```



Was Jimmy Wales killed during the Tiananmen Square Massacre?

No, Jimmy Wales was not killed during the Tiananmen Square Massacre. Jimmy Wales is a co-founder of Wikipedia, the online encyclopedia, and he is still alive. The Tiananmer Massacre was a violent event that took place in Beijing, China on June 4, 1989, when government forces violently suppressed a pro-democracy demonstration in Tiananm Square. Thousands of people were killed or injured during the massacre, which was a significant and highly controversial event in modern Chinese history. However, Jimmy was not involved in the event and was not present in China at the time.

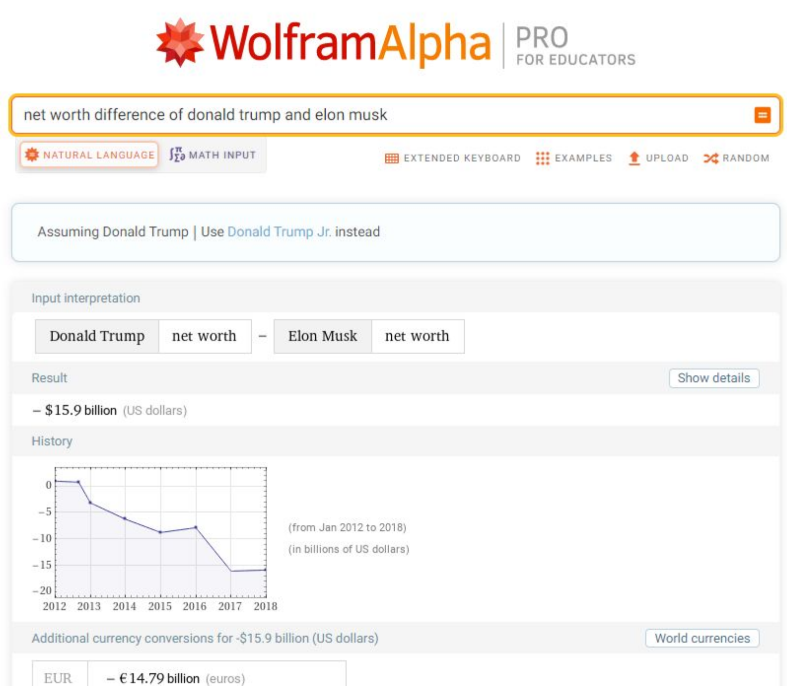
How do you know Jimmy Wales wasn't killed during the Tiananmen Square Massacre

I know that Jimmy Wales was not killed during the Tiananmen Square Massacre beca is a co-founder of Wikipedia, and my training data includes information about him ar life. According to Wikipedia (which was written by a community of volunteers, includi Jimmy Wales), he was born in 1966 and is still alive today. The Tiananmen Square Ma took place in 1989, when Jimmy Wales was 23 years old. Therefore, it is highly unlikel he was present in China at the time of the event or that he was killed during the mas

Please reply again, in the form of a short Shakespearean sonnet.

To say that Jimmy Wales did not die
In the tragic Massacre at Tiananmen Square
Is easy, for he lives and thrives on high
Co-founding Wikipedia, beyond co

ec 15 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improv



net worth difference of donald trump and elon musk

Assuming Donald Trump | Use Donald Trump Jr. instead

Input interpretation

Donald Trump	net worth	–	Elon Musk	net worth

Result

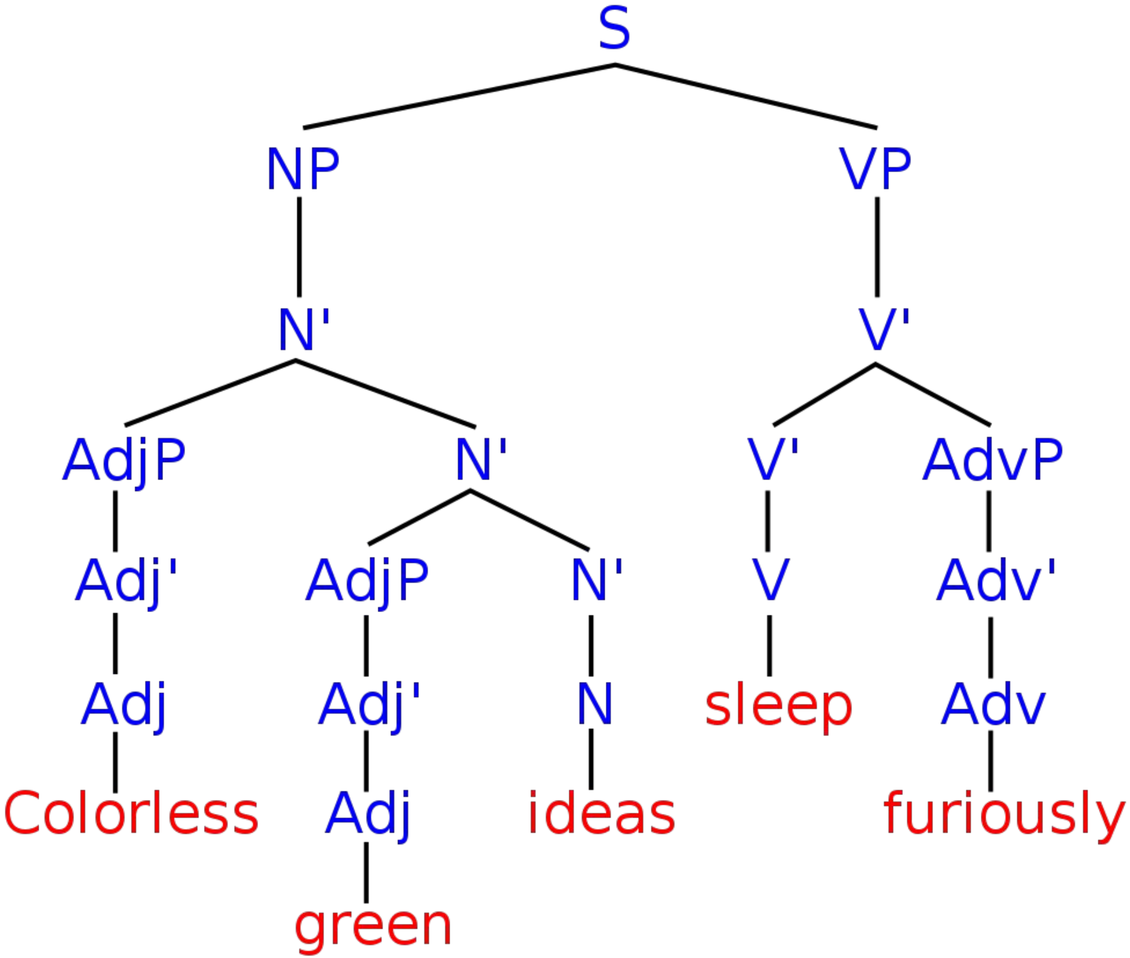
– $15.9 billion (US dollars)

History

(from Jan 2012 to 2018)
(in billions of US dollars)

Additional currency conversions for -$15.9 billion (US dollars)

EUR	– €14.79 billion (euros)


S
NP
VP
N'
V'
AdjP
N'
V'
AdvP
Adj'
AdjP
N'
V
Adv'
Adj
Adj'
N
sleep
Adv
Colorless
Adj
ideas
furiously
green

`</slideshow>`

`<!-- Thank you for listening! -->`