atobrute

Why Plaintext Isn't So Vanilla
Antony Scopazazz - 7/14/2011
An Apology

- I was sick...
Postulate 1

Data Size Matters
Postulate 2

String to Numeric Conversion is Expensive (atoi, atof, ...)
Postulate 3

*Everything is in a Binary Data Format (including 'plaintext')*
So....

Where does this get us?
So....

Where does this get us?
So....

Where does this get us?

Let's do a couple of thought experiments.
Why do computers use binary?

It is easy to use:

1. Anything simpler is trivial (0000000)
2. Anything more complicated requires additional hardware architecture (2010210)
3. Binary seems just right!
Hardware?!

Yes, believe it or not, all computers are fundamentally analogue devices...something we often lose sight of.
Hardware?!

Yes, believe it or not, all computers are fundamentally analogue devices...something we often lose sight of.

So this means that it is conceivable for future systems to implement transistors which are currently novel (i.e. crazy).
What does this mean exactly?

0

1

http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4819000
Thus...

frenz, romanze, countrymenz, lend mah yerz eerz

das rite... jussa liddle closer...

www.shmoop.com
Enter: bitarray

Written by Ilan Schnell, bitarray gives bit access to memory.

It is really fast.

You can get it either on the Cheeseshop or in EPD.

http://pypi.python.org/pypi/bitarray
bitarray API

>>> from bitarray import bitarray
>>> a = bitarray() # create empty bitarray
>>> a.append(True)
>>> a.extend([False, True, True])
>>> a
bitarray('1011')
Well...

If we want to fake futuristic transistor architecture, we need to have a way to map current binary systems to an arbitrary N-state machine.
Well...

If we want to fake futuristic transistor architecture, we need to have a way to map current binary systems to an arbitrary N-state machine. By analogy to bits, call indivisible units on an N-state machine 'nits.'
If we want to fake futuristic transistor architecture, we need to have a way to map current binary systems to an arbitrary N-state machine.

By analogy to bits, call indivisible units on an N-state machine 'nits.' Hence...
Enter: nitarray

nitarray is has a very similar interface to bitarray.
nitarray is has a very similar interface to bitarray.

In fact, nitarray uses encodings and decodings on top of bitarray to simulate N-state machines.
Enter: nitarray

nitarray is has a very similar interface to bitarray.

In fact, nitarray uses encodings and decodings on top of bitarray to simulate N-state machines.

Huzzah! We are no longer bound by our binary architecture.
nitarray API

In [1]: from nitarray import nit

In [2]: nit(10, 3)
Out[2]: [1, 0, 1]

In [3]: nit(1794864036260549376542, 42)

In [4]: nit(16, 2)
Out[4]: [1, 0, 0, 0, 0]

In [5]: nit(101, 256)
Out[5]: [101]
nitarray API

In [1]: from nitarray import nit_encoding

In [2]: nit_encoding(6)
Out[2]: {0: bitarray('000'),
      1: bitarray('001'),
      2: bitarray('010'),
      3: bitarray('011'),
      4: bitarray('100'),
      5: bitarray('101')}
nitarray API

In [2]: nitarray([1, 0, 2, 2, 0], 3)
Out[2]: nitarray('1,0,2,2,0', 3)

In [3]: nitarray('1,6,2,0,4', 7)
Out[3]: nitarray('1,6,2,0,4', 7)

In [4]: nitarray(37, 42)
Out[4]: nitarray
('0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0', 42)
nitarray API

In [5]: n = nitarray([1, 0, 2, 2, 0], 3)

In [6]: n.append(2)

In [7]: n
Out[7]: nitarray('1,0,2,2,0,2', 3)

In [8]: n.count(2)
Out[8]: 3

In [9]: n.remove(0)

In [10]: n
Out[10]: nitarray('1,2,2,0,2', 3)

In [11]: n.tolist()
Out[11]: [1, 2, 2, 0, 2]

In [12]: n.to01()
Out[12]: '1,2,2,0,2'

In [13]: int(n)
Out[13]: 155
nitarray API

In [5]: n = nitarray([1, 0, 2, 2, 0], 3)

In [6]: n.append(2)

In [7]: n
Out[7]: nitarray('1,0,2,2,0,2', 3)

In [8]: n.count(2)
Out[8]: 3

In [9]: n.remove(0)

In [10]: n
Out[10]: nitarray('1,2,2,0,2', 3)

In [11]: n.tolist()
Out[11]: [1, 2, 2, 0, 2]

In [12]: n.to01()
Out[12]: '1,2,2,0,2'

In [13]: int(n)
Out[13]: 155
int atoi(const char* string) {
    int value = 0;

    if (string) {
        while (*string && (*string <= '9' && *string >= '0')) {
            value = (value * 10) + (*string - '0');
            string++;
        }
    }

    return value;
}
Huh...

The expense of the atoi() call does not really change with N-state systems.
nitarray API

In [1]: from nitarray import nitarray

In [2]: n = nitarray([], 42)

In [3]: n.fromstring("No one expects the Spanish Inquisition!")

In [4]: n
Out[4]: nitarray
('1,36,2,27,0,32,2,27,2,26,2,17,0,32,2,17,2,36,2,28,2,17,'
'2,15,2,32,2,31,0,32,2,32,2,20,2,17,0,32,1,41,2,28,2,13,'
'2,26,2,21,2,31,2,20,0,32,1,31,2,26,2,29,2,33,2,21,2,31,'
'2,21,2,32,2,21,2,27,2,26,0,33', 42)

In [5]: n.tostring()
Out[5]: 'No one expects the Spanish Inquisition!'
Major Points

- Don't use atoi(), atol(), atof(), etc.
- ASCII / unicode are specialized binary formats.
- Standards are great; use HDF5 (PyTables, h5py)
- [http://scopatz.github.com/nitarray/](http://scopatz.github.com/nitarray/)
Questions?

YA RLY