The Computation and Representation of Address Ranges

With an Introduction to the Python `justbytes` Library

Anne Mulhern

Red Hat, Inc.

November 29, 2017

Audience

Programmers who write code that must compute with and represent address ranges.

- If you have never needed to discover free space on a device. . . then this talk is probably not for you.
- If you have never written code to display or calculate the size of a device. . . then the second part of the talk is not for you.

Expectations

What this talk will include.

- A little bit of personal experience.
- Some math.
- Some opinion.

Introduction

The Topic

What is this talk about?

The Domain

Allocating space on devices.

- Partitions
- Caches

Not the Domain

Where this is not an issue.

- Bandwidth
- Sales

Not the Domain

When it is just a question of the amount being moved.

The Domain

Allocating address regions is a *layout* problem.

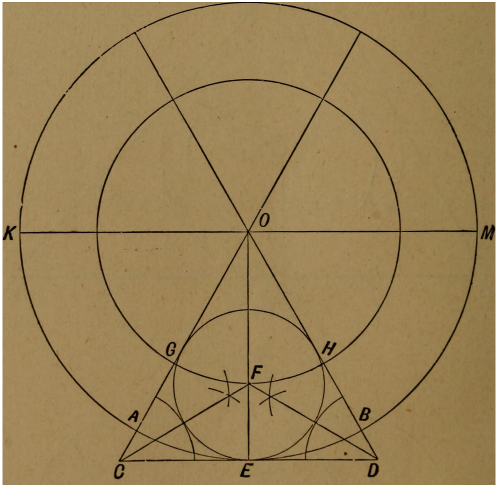


Fig. 87.

Conceptual View of Memory

Linear arrangement of slots.

Real Example in the Domain (I)

lsblk[1]

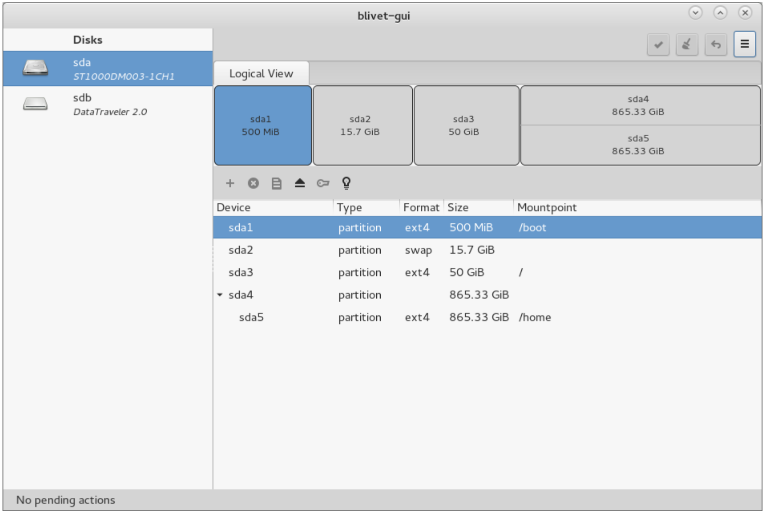
lsblk –ascii output

```
NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda         8:0    0 931.5G  0 disk
|-sda1      8:1    0   500M  0 part /boot
|-sda2      8:2    0  15.7G  0 part [SWAP]
|-sda3      8:3    0    50G  0 part /
|-sda4      8:4    0     1K  0 part
`-sda5      8:5    0 865.3G  0 part /home
sdb         8:16   1   7.3G  0 disk /run/media/mulhern/12CE-4D83
sr0        11:0    1  1024M  0 rom
```

[1]lsblk from util-linux 2.28

Real Example in the Domain (II)

blivet-gui [2]



Device	Type	Format	Size	Mountpoint
sda1	partition	ext4	500 MiB	/boot
sda2	partition	swap	15.7 GiB	
sda3	partition	ext4	50 GiB	/
sda4	partition		865.33 GiB	
sda5	partition	ext4	865.33 GiB	/home

No pending actions

[2] blivet-gui 1.0.7

Some Final Words (I)

Last thoughts on amounts.

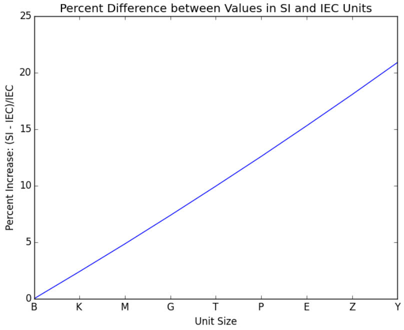
Dealing with amounts is hard [3].

[3] "Numerical Computing with IEEE Floating Point Arithmetic" by Michael L. Overton but also "The End of Error: Unum Computing" by John L. Gustafson.

Some Final Words (II)

Use of SI units in marketing.

Comparing apples with slightly larger apples is a marketing trick. You can say you have more apples if you have smaller apples, but does your customer really get more applesauce?



Confession

This is a hard problem?!

Until I started working on software supporting block devices I would not have thought that this was a difficult subject. But it is.

Nomenclature

For purposes of discussion...

Range	The word I use when I mean the size of an allocated partition, or of my computer's RAM, or the block size on a disk.

Topics

Problems to cover.

Computation	doing arithmetic with address range
Display	communicating address range values to a human reader

Display

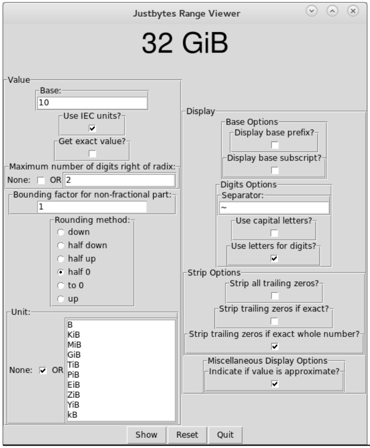
Showing values to a human reader.

This is not a discussion of machine readable output.

Display

Showing values to a human reader.

It is the business of the client to determine the *appearance* of the display; the library should return a *structure* representing a value to display to the client. How the structure is determined should be configurable[4].



Justbytes Range Viewer

32 GiB

Value

Base: 10

Use IEC units? ✓

Get exact value? ☐

Maximum number of digits right of radix: None: ☐ OR 2

Bounding factor for non-fractional part: 1

Rounding method: down / half down / half up / [✓ half 0] / to 0 / up

Unit: None: ✓ OR B, KiB, MiB, GiB, TiB, PiB, EiB, ZiB, YiB, kB

Display

Base Options
- Display base prefix? ☐
- Display base subscript? ☐

Digits Options
- Separator: ~
- Use capital letters? ☐
- Use letters for digits? ✓

Strip Options
- Strip all trailing zeros? ☐
- Strip trailing zeros if exact? ☐
- Strip trailing zeros if exact whole number? ✓

Miscellaneous Display Options
- Indicate if value is approximate? ✓

[4] `justbytes-gui` (`https://pypi.python.org/pypi/justbytes-gui/`) is a simple library for experimenting with the existing display options in the `justbytes` library.

Display Structure

Relevant information about the display value.

- Display Value (D)
  - Sign
  - Integer Part
  - Non-Repeating Fractional Part
  - Repeating Fractional Part
  - Base
- Relation of D to actual value (V) in units (U)
- Units (U)

Display Structure Examples

Default choices for the display value for 32 `GiB`.

Configuration	Value	Structure	Value
Base	10	Sign	$+$
IEC	True	Integer Part	[3, 2]
Exact Value	False	Non-Repeating Fractional Part	[]
Max Digits after Radix	2	Repeating Fractional Part	[]
Bounding Factor	1	Base	10
Rounding Method	half 0	Relation	$=$
Unit	None	Unit	GiB

Display Structure Examples

Consequences of choice of SI units for display of 32 `GiB`.

Configuration	Value	Structure	Value
Base	10	Sign	$+$
IEC	**False**	Integer Part	**[3, 4]**
Exact Value	False	Non-Repeating Fractional Part	**[3, 6]**
Max Digits after Radix	2	Repeating Fractional Part	[]
Bounding Factor	1	Base	10
Rounding Method	half 0	Relation	$>$
Unit	None	Unit	**GB**

Display Structure Examples

Consequences of some choices for display of 32 `GiB`.

Configuration	Value	Structure	Value
Base	**1000**	Sign	$+$
IEC	**False**	Integer Part	**[34, 359]**
Exact Value	**True**	Non-Repeating	**[738,368]**
Max Digits after Radix	2	Repeating	[]
Bounding Factor	1	Base	**1000**
Rounding Method	half 0	Relation	$=$
Unit	None	Unit	**MB**

Value Configuration

`justbytes`' defaults for the `getStringInfo()` method.

The computation of the display value is configurable, because it is not clear that there is just one best solution.

Value Configuration	Value
Base	10
IEC	True
Exact Value	False
Max Digits after Radix	2
Bounding Factor	1
Rounding Method	half 0
Unit	None

Given IEC units, I would prefer base 1024. $32 GiB + 512 B$ is "$>$ 32.00 GiB" with defaults; with base 1024 it is "32.0~0~512", i.e., precisely 32 GiB, 0 MiB, 0 KiB and 512 B.

Display Configuration

justbytes' defaults for the `__str__` method.

The display choices are configurable, because it is not clear that there is just one best solution.

Display Configuration	Value
Show Base Prefix	False
Show Base Subscript	False
Digits Separator	~
Use Letters for Digits	True
Capitalize Digit Letters	False
Strip Trailing Zeros	False
Strip Trailing Zeros if Exact	False
Strip Trailing Zeros if Exact Whole Number	True
Show Relation of Display Value to Actual	True

There is a lot more information in "$>$ 1.00 TiB" than there is in "1.00 TiB".

Always Prefer IEC to SI Units

IEC units better reflect the structure of memory[5].

The structure of memory and the structure of addresses are based on powers of 2.

[5]Would 2^8have been better than 2^{10}? Probably.

Display

Summary[6].

- IEC units are *always* preferable to SI units.
- There is no single obvious best choice for certain display options.
- IEC units are most informative when coupled with base 1024, but that might be a hard sell.
- Showing relation of displayed value to actual value is a good idea.
- The client code decides the *appearance*, but `justbytes` decides the *value* of the display representation based on configuration options.

[6]For technical reasons I ended up writing a separate library, `justbases`, to handle the computation of the representation of the numeric value; it has a GUI as well: `https://pypi.python.org/pypi/justbases-gui/0.1.0`.

Computation

Computing with address ranges.

Computation

Rules for arithmetic operations.

1. Types are *always correct*.
2. Results are *always* exact.
3. Operands are *never* float or Decimal.
4. Results of computations never yield ranges to any power but 1.

Types are *Always* Correct

The usual rules of high-school arithmetic are *strictly* followed[8].

- Just as $1\,\mathrm{gallon} + \pi$ is meaningless, so is $32\,\mathrm{GiB} + 64$.[7]
- Just as $\frac{1\,\mathrm{gallon}}{2}$ is $\frac{1}{2}\,\mathrm{gallon}$, so $\frac{32\,\mathrm{GiB}}{2}$ is $16\,\mathrm{GiB}$.

Operations not allowed by these rules are called *nonsensical*.

[7]Some idiosyncracies of the Python libraries make it attractive to be able to add numbers to ranges, but this is not really a good idea. See `http://pythonhosted.org/justbytes/tutorial.html#using-the-additive-identity`.

[8]The formal type rules are stated in the `justbytes` comments for the various operations.

Results are *Always* Exact

There is no implicit rounding.

This means that $\frac{1}{3}32\,\mathrm{GiB} = 10\frac{2}{3}\,\mathrm{GiB}$ which is $11453246122\frac{2}{3}\,\mathrm{B}$. The sole motivation is so that the ordinary rules of arithmetic are preserved, i.e., associativity, distributivity, and so forth hold when computing with ranges. The sole drawback is that if a whole number of bytes is sought for the result, as is typical, the result must be explicitly rounded.

Operands are *Never* float or Decimal

Because most programmers do not have a good understanding of these types.

They tend to forget all sorts of things, like:

- Floating point numbers do not obey the usual laws of arithmetic.
- $0.3 \neq \frac{3}{10}$.[9]
- There are a very large number of pairs of floats such that $x \neq y$ but $\frac{x+y}{2} \in \{x, y\}$.
- Most of the problems with floats are true of Decimals at their precision limit[10].

[9] It's actually $\frac{5404319552844595}{18014398509481984}$.

[10] I think that the best use of Decimal is probably with decimal values with strict rules for rounding, like decimal currency.

Results of Computations *Never* Yield Ranges to any Power but 1.

No operation is allowed that would result in a Range with units that could not reasonably be used as a partition size.

Nonsensical:	n^R,$n+R$, etc.
Forbidden:	R^n, RR
Permitted:	$R+R$, nR, R/R, $R<R$, $R\%R$, etc.

Implementation

The internal representation of a Range is as a fractional quantity of bytes.

Given the restriction of Range result powers to 1, fractions are all that are necessary to achieve exact results, i.e., there is no requirement for symbolic computing.

Caveat

Something to watch out for.

A simple example.

The fraction $\frac{111}{11112121}$ has 13668 digits in the repeating part of its decimal representation. There is a noticeable delay before the value is correctly calculated.

Avoid requesting unbounded precision when displaying a range. This can result in a very time-consuming computation as the number of repeating digits is bounded by the magnitude of the denominator in the equivalent fraction.

Usage Example

pydevDAG with defaults

lsdev output

```
NAME                DEVTYPE     MAJOR              SIZE
/dev/sdb            disk        8            < 7.27 GiB
/dev/sr0            disk        11        < 1024.00 MiB
0xdae3640bc5005000  None        None               None
`-/dev/sda          disk        8          > 931.51 GiB
  |-/dev/sda1       partition   8               500 MiB
  |-/dev/sda2       partition   8           < 15.70 GiB
  |-/dev/sda3       partition   8                50 GiB
  |-/dev/sda4       partition   8                 1 KiB
  `-/dev/sda5       partition   8          < 865.33 GiB
```

Note how the relation of the displayed to the actual value is made clear by the $<$ and $>$ symbols.

Usage Example

pydevDAG with base 1024

lsdev output

```
NAME                DEVTYPE     MAJOR                SIZE
/dev/sdb            disk        8          7.273~704  GiB
/dev/sr0            disk        11      1023.1023~512 MiB
0xdae3640bc5005000  None        None                 None
`-/dev/sda          disk        8        931.525~728  GiB
  |-/dev/sda1       partition   8                500  MiB
  |-/dev/sda2       partition   8           15.712~0  GiB
  |-/dev/sda3       partition   8                 50  GiB
  |-/dev/sda4       partition   8                  1  KiB
  `-/dev/sda5       partition   8          865.335~0  GiB
```

Note that all values are exact.

Try It Out

`Packages available.`

- `justbytes` is available in Fedora as python{2,3}-justbytes and from Pypi via pip.
- `justbytes` tutorial available at `http://pythonhosted.org/justbytes/tutorial.html#`.
- `justbytes-gui` is available only on Pypi and expected to remain so.

What I Haven't Talked About

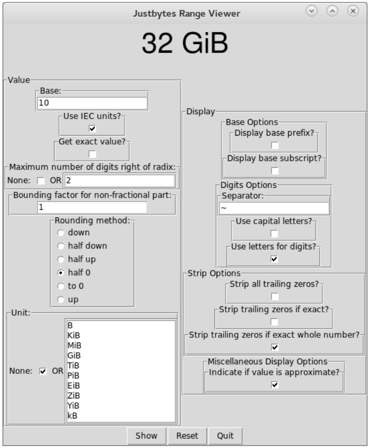
Major things I haven't solved.

- Input of address ranges.
- Computing with address ranges and other units.

Input of Address Ranges (I)

Getting an address range non-programatically.

I haven't really solved this problem to my satisfaction; this is why `justbytes-gui` is just a library and not also an application.



Justbytes Range Viewer

32 GiB

Value
- Base: 10
- Use IEC units? ✓
- Get exact value? ☐
- Maximum number of digits right of radix: None: ☐ OR 2
- Bounding factor for non-fractional part: 1
- Rounding method:
  - ○ down
  - ○ half down
  - ○ half up
  - ● half 0
  - ○ to 0
  - ○ up
- Unit: None: ✓ OR
  - B
  - KiB
  - MiB
  - GiB
  - TiB
  - PiB
  - EiB
  - ZiB
  - YiB
  - kB

Display
- Base Options
  - Display base prefix? ☐
  - Display base subscript? ☐
- Digits Options
  - Separator: ~
  - Use capital letters? ☐
  - Use letters for digits? ✓
- Strip Options
  - Strip all trailing zeros? ☐
  - Strip trailing zeros if exact? ☐
  - Strip trailing zeros if exact whole number? ✓
- Miscellaneous Display Options
  - Indicate if value is approximate? ✓

Input of Address Ranges (II)

Some thoughts.

- Best to avoid it altogether, wherever possible.
- In many practical situations, it is better to express a range as a fraction of some related quantity, i.e., not as an absolute value.
- In a GUI, it is best to limit the amount of free-form text the user can enter. Certainly, choice of units should be handled by a widget.
- In a CLI or configuration file, there might be benefit to separating the numeric part from the units.
- Textual representations specific to floating point, e.g., 1e32, should be disallowed[11].
- Numbers with decimal points, e.g., 0.2, should be given their literal meaning, i.e., $\frac{1}{5}$ in this case, rather than their floating point interpretation, i.e., $\frac{3602879701896397}{18014398509481984}$.

[11] That they are derived from scientific notation is not a justification for allowing them.

Computing with Address Ranges and Other Units (I)

The only types of arguments to Range operations are numerical types or Range objects.

There are perfectly good Python libraries[12] for computing with units and verifying dimensionality, but they are all for physical quantities, i.e. not the domain. But it is generally the case that computations with address ranges are not actually unitless. With a more complete library, dimensionality can be checked.

Example.

The sysfs size attribute for block devices is the number of sectors in the device. Converting this value to a range requires multiplying the size by 512, the number of bytes in the sector and converting the result. Better to use the actual units, like $s\,\mathrm{blocks}\frac{512\,\mathrm{bytes}}{\mathrm{block}} = 512s\,\mathrm{bytes}$.

[12] Pint: `https://pypi.python.org/pypi/Pint`, etc.

Computing with Address Ranges and Other Units (II)

Designing a library to satisfy this problem is a large undertaking with limited reward.

- The design of the library is not obvious.
- Such a library would help to find bugs, but it would require considerable investment to use properly.

Thanks

Any questions?