MagniOS
Mini Uni Kernel OS

What is a Computer?

computer
kəmˈpjuːtə/
noun

1. an electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals.

- “google”

Components

- Memory
- Storage
- Input device
- Output device
- Processor

Operating System

Abstracts hardware to a standard and provides low-level services

Application

DIY

Microkernel

In computer science, a microkernel (also known as μ-kernel) is the near-minimum amount of software that can provide the mechanisms needed to implement an operating system (OS). These mechanisms include low-level address space management, thread management, and inter-process communication (IPC).

- wikipedia

Unikernel - “fixed purpose computer”

- Specialized
- Single address space (machine images)
- “Library” operating system
- Configuration/selection of libraries
- “Fixed-purpose computer”
- Run often on a hypervisor/virtual(ized) computer

Programming Language

“A **programming language** is a formal computer language or constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms.” - wikipedia

My take:

It’s a virtual computer!

And it’s (more) portable

http://urbit.org/ - an operating function

http://moronlab.blogspot.hk/2010/01/urbit-functional-programming-from.html

If an alien species had invented computing, what might it look like? I think Urbit is a credible answer to that question. I'm joking, but only partially: from the perspective of "throw out *everything* and start over from scratch to see what modern computing would be like", Urbit seems like it may be a sincere attempt. At one point in the past, the ideas appeared to be*deliberately* obfuscated, like performance art or like Brainfuck. They've subsequently made efforts to explain the ideas and build things up from basic concepts, and I think the authors are sincere, but don't hold me to that. In conclusion, I think it's technically neat, but I think it'd be a lot neater if it were documented and presented within a grounding of typical CS and software terminology/concepts. Or if they can't give up the custom terminology, I'd want to see ~10X more material (per unit concept) introducing and clarifying the ideas slowly from the ground up. If they're inventing their own alien computer science, then they need to write the alien "C Programming Language" (K&R) and alien "Structure and Interpretation of Computer Programs" [3].

Urbit is a clean-slate system software stack defined as a deterministic computer. An encrypted P2P network, %ames , runs on a functional operating system, Arvo, written in a strict, typed functional language, Hoon, which compiles itself to a combinator interpreter, Nock, whose spec gzips to 340 bytes.

Urbit glossary

Urbit defines the **Hoon** syntax that implements **Arvo** that you use for high-level programming.

The inner workings is built using the core **Hoon** that is based on mathematical **Nock** .

Nock is a *Turing-complete, non-lambda combinator interpreter*.

Hoon is a *strict, higher-order typed functional language* that compiles itself to Nock.

An Urbit `ship` is a *cryptographic identity, a human-memorable name*, and a packet routing address. Ships are classed by the number of bits in their address.

(galaxy, star, planet, moon, comet)

Urbit Hoon

Glyphs and characters

Hoon is a heavy punctuation user. To make ASCII great again, we've mapped each punctuation glyph to a syllable:

```
ace [1 space]   gal <              pal (
bar |           gap [>1 space, nl]  par )
bas \           gar >              sel [
buc $           hax #              sem ;
cab _           hep -              ser ]
cen %           kel {              sig ~
col :           ker }              soq '
com ,           ket ^              tar *
doq "           lus +              tec `
dot .           pam &              tis =
fas /           pat @              wut ?
zap !
```

```
|=  end/atom
  =/  count  1
  |-  ^-  (list tape)
  ?:  =(end count)  ~
  :_  $(count (add 1 count))
  ?:  =(0 (mod count 15))
    "FizzBuzz"
  ?:  =(0 (mod count 5))
    "Fizz"
  ?:  =(0 (mod count 3))
    "Buzz"
  (pave !>(count))
```

Ok… back to MagniOS

Jml - a small simple (to implement) powerful programming language

“ Input ⤔ process ⤔ output “

REST ⤔ program ⤔ HTML

Library-OS functions
- Simple storage API
- Simple IO (http, mqtt)

JML - Just a Macro Language

The simplest language you can implement...

```
echo "[* 2 [+ 3 4 [* 5 6]]]" | ./run -t

>>>[* 2 [+ 3 4 [* 5 6]]]<<<
>>>[* 2 [+ 3 4 30]]<<<
>>>[* 2 37]<<<
>>>74<<<
74
```

JML - Just a Macro Language

The simplest language you can implement...

```
>>>[map fac [iota 1 5]]<<<
>>>[map fac 1 2 3 4 5 ]<<<
>>>[fac 1] [fac 2] [fac 3] [fac 4] [fac 5] <<<
>>>[* [iota 1 1]] [* [iota 1 2]] [* [iota 1 3]] [* [iota 1 4]] [*
[iota 1 5]] <<<
>>>[* 1 ] [* 1 2 ] [* 1 2 3 ] [* 1 2 3 4 ] [* 1 2 3 4 5 ] <<<
>>>1 2 6 24 120 <<<
1 2 6 24 120
```

JML - Just a Macro Language

The simplest language you can implement...

```
[macro calculate $user $what $n]
    <h1>Calculate $what($n)</h1>
    <p>Hi $user!</p>
    <p>The result is: [$what $n]</p>
[/macro]

[calculate Peter fac 6]
[calculate Jonas uppercase foobar]
```

```
[macro fac $n]
  [* [iota 1 $n]]
[/macro]

[macro max $a $b]
  [if [< $a $b] $b $a]
[/macro]

[map fac 2 4 6]
-> 2 24 720 40320
```

JML - Just a Macro Language

The simplest language you can implement...

no-if-if no-switch-switch

```
[if [< 3 4] second first]

[   [if    [< 3 4]   second first]    3   4   ]
```

```
[macro switch $x] [case-$x ] [/macro]

[macro case-one] 1 [/macro]
[macro case-two] 2 [/macro]
[macro case-three] 3 [/macro]
[macro case-four] 4 [/macro]
```

Website

```
[macro /]
  <h1>Home</h1>

  <p> ... </p>

  <ul>[map li 1 2 3]</ul>
[/macro

[macro li @txt] <li>@txt</li> [/macro]
```

Ajax/REST

```
[macro /input]

  Your name: <form><input name='name' type='isindex'></form>

[/macro]


[macro /name @name] Hello @name! [/macro]
```

Persistent Storage

```
[data foo bar fie]
[data fum foobar]


[data-foo]          => bar fie

[datas]             => foo fum

[have-data foo]     => 1 or 0
```

Questions

http://github.com/yesco/jml