How to rewrite the OS using C by strong type

Metasepi Project / Kiwamu Okabe
Who am I?
☆ http://www.masterq.net/
☆ Organizer of Metasepi Project
☆ A developer of Ajhc Haskell compiler
☆ A Debian Maintainer
☆ 10 years' experience in developing OS using NetBSD
Agenda

☆ [1] Problems of OS using C
☆ [2] Type safety
☆ [3] Existing OS using strong type
☆ [4] Snatch-driven development
☆ [5] Demo
☆ [6] Case study of Snatch
☆ [7] Future work
[1] Problems of OS using C

☆ Most OS uses C language
☆ C is good for system programming
☆ But C occurs many problems
☆ Pointer to array doesn't know the length

char *array_p

char array[4]

'\0'

Length: 4
Page fault in kernel

☆ Page fault in user space => SEGV
☆ Page fault in kernel space => Halt!
Weak type

☆ Great use of (void *) type

☆ NetBSD kernel uses 45130 times!

```
$ pwd
/home/kiwamu/src/netbsd/sys
$ grep "void \*" `find . -name "*.c"` | wc -l
45130
```

☆ No choice but to use weak type for flexibility
[2] Type safety

☆ Get less runtime errors

- Weak typing language: 
  - Compile error
  - Runtime error

- Strong typing language: 
  - Compile error
  - Runtime error

You can see the area.

You can NOT see the area.
Avoid buffer overrun

Strong type avoids buffer overrun.

Know the length.

arrayP :: [Char]

Interface

Length: 4
Avoid page fault in kernel

Only touch the area constructed.
Flexibility without weak type

☆ Algebraic data type

```haskell
data Node = Leaf Integer | Branch Node Node
```

☆ Type class

```haskell
class Functor f where
  fmap :: (a -> b) -> f a -> f b
instance Functor [] where
  fmap f (x:xs) = f x : fmap f xs
  fmap f [] = []
instance Functor Maybe where
  fmap _ Nothing = Nothing
  fmap f (Just x) = Just (f x)
```

☆ Type inference
Kernel needs strong type

☆ IoT: Internet of Things
☆ Poor hardware, and Rich feature
☆ Many custom requests shower kernel
☆ Strong type is needed by kernel rather than application on user space
[3] Existing OS using strong type

Already we have.

☆ Funk
http://home.gna.org/funk/

☆ snowflake-os
https://code.google.com/p/snowflake-os/

☆ House
http://programatica.cs.pdx.edu/House/

Why isn't it for daily use?
Poor design and less functions

☆ Design from scratch
☆ Polling interrupt
☆ Not have bus driver
☆ Support less devices
☆ Only for x86
☆ Can't run Firefox
No compatible POSIX

- Own API
- Own compiler
- Own web browser
- Own API
- Own compiler
- Own web browser
- POSIX API
- Existing compiler
- Existing web browser

Need more man-hour

Need less man-hour
[4] Snatch-driven development

Rewrite kernel using C with strong type by little and little.
UNIX like OS needs reentrant
Strong type OS uses polling intr
Ajhc Haskell compiler

Context can run without lock.
The proof of the pudding is in the eating.
MCU app without OS #1

https://github.com/ajhc/demo-cortex-m3
MCU app without OS #2

Memory map

![Memory map diagram]

- **ROM**
  - TEXT
    - C code compiled
    - Haskell code compiled
    - Ajhc runtime
  - 11kB
    - 0x08000000
    - 0x080027d0

- **RAM**
  - DATA
  - 1kB
    - 0x20000000
    - 0x20000490
    - 0x20000500
  - BSS
    - Malloc heap (2kB)
    - Haskell heap (24kB)
  - 29kB
    - 0x200077ec
  - STACK
    - 2kB
    - 0x20008000

Note: Too large!
MCU app with OS

https://github.com/ajhc/demo-cortex-m3
[6] Case study of Snatch

We found some idioms that are useful to rewrite C language with Haskell.
Idiom 1: Call function

C and Haskell call with each other.
Idiom2: Read/Write memory

Haskell can read/write memory directly.

```
peek :: Ptr a -> IO a
poke :: Ptr a -> a -> IO ()
```
Idiom3: Read/Write struct

Read structs chained with pointer.

---

C language

```c
struct foo *fp;

struct foo f;
struct bar fb;
struct baz *fz
struct baz z;
int zi;
```

Haskell

```
fp :: Ptr Foo

instance Storable Foo

fb ::
instance Storable Bar

fz :: Ptr Baz

peek :: Ptr a -> IO a

instance Storable Baz
```
Idiom4: Foreign Primitives

Directly insert the text following const.
[7] Future work

☆ Benchmark
☆ Pointer combinator
☆ Share state between contexts
☆ Porting libraries running on GHC
☆ Debug method
☆ Fix many bugs
Try to use the other language

☆ ATS
http://www.ats-lang.org/
JATS-UG - Japan ATS User Group
http://jats-ug.metasepi.org/

☆ Rust
http://www.rust-lang.org/
Workshop at Nagoya

☆ Functional MCU programing workshop at Nagoya

☆ Meeting minutes

http://metasepi.org/posts/2014-01-05-mbed_fp_0.html