Taming of Pict

Matej Košík

Faculty of Informatics and Information Technology
Slovak University of Technology in Bratislava

January 21, 2008

Contents

Motivation

The object-capability paradigm

Pict specific details

Taming of Pict

Open Problems

Hypothesis: Advantages of the object-capability security paradigm

- POLA can be obeyed without discomfort
- expressivness
- elegance
- efficiency
- smaller TCB

Limitations

- source code of all parts of the system must be available

Contents

Motivation

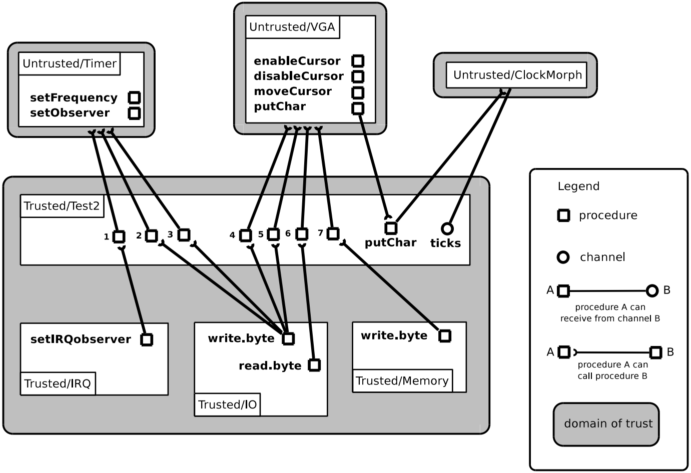
The object-capability paradigm

Pict specific details

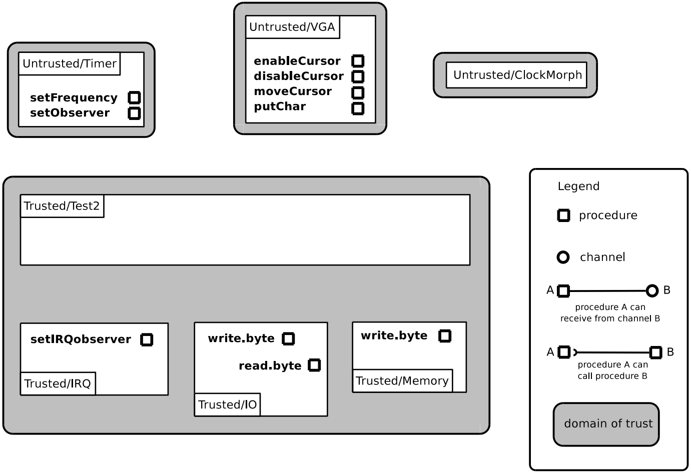
Taming of Pict

Open Problems

An example of “can call” relationship



An example of “can call” relationship



Untrusted/Timer

setFrequency ☐

setObserver ☐

Untrusted/VGA

enableCursor ☐

disableCursor ☐

moveCursor ☐

putChar ☐

Untrusted/ClockMorph

Trusted/Test2

setIRQobserver ☐

Trusted/IRQ

write.byte ☐

read.byte ☐

Trusted/IO

write.byte ☐

Trusted/Memory

Legend

☐ procedure

○ channel

A ☐—○ B

procedure A can receive from channel B

A ☐›—☐ B

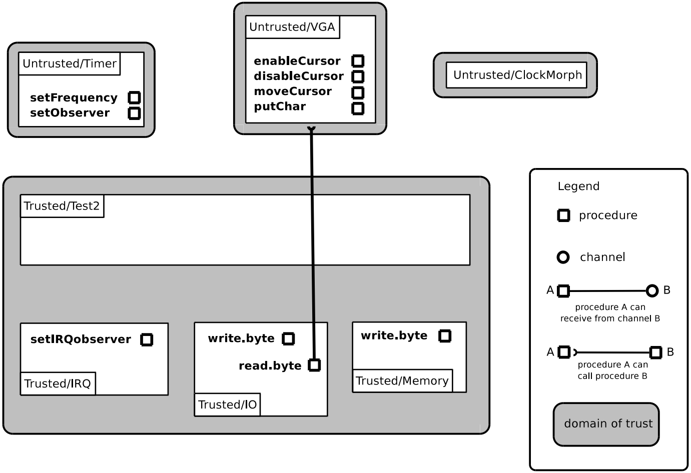
procedure A can call procedure B

domain of trust

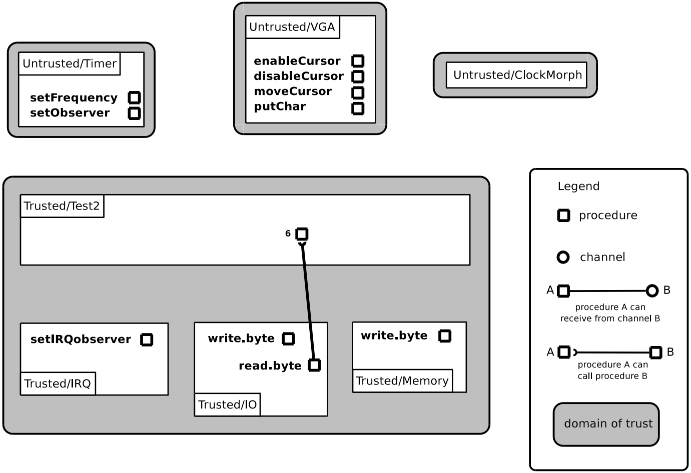
The Problem: Required authority of the untrusted VGA driver

- to read from the `0x3D5` I/O register
- to read from the `0x3D4` I/O register
- to write to the `0x3D4` I/O register
- to write to the frame buffer

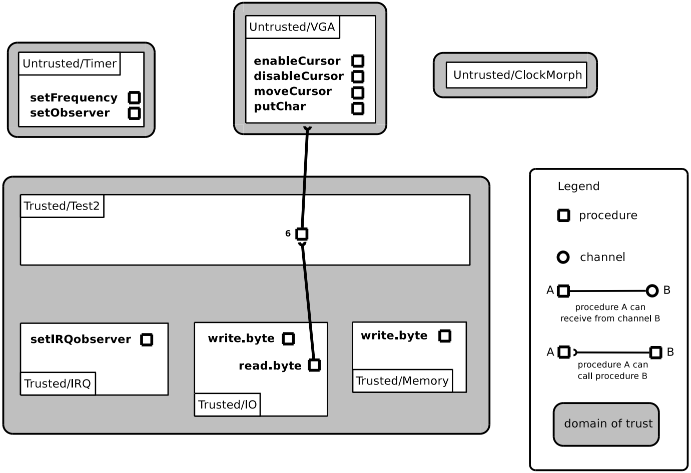
An example of "can call" relationship



An example of “can call” relationship



An example of “can call” relationship



An example of “can call” relationship

```
io.read.byte
```

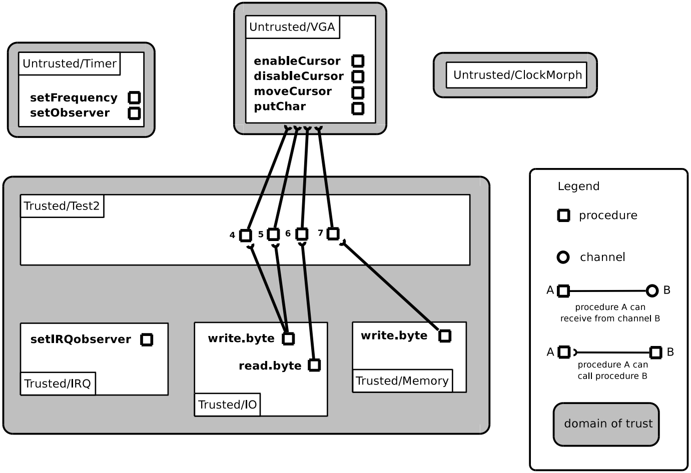
```
\() = (io.read.byte 981)
```

An example of “can call” relationship

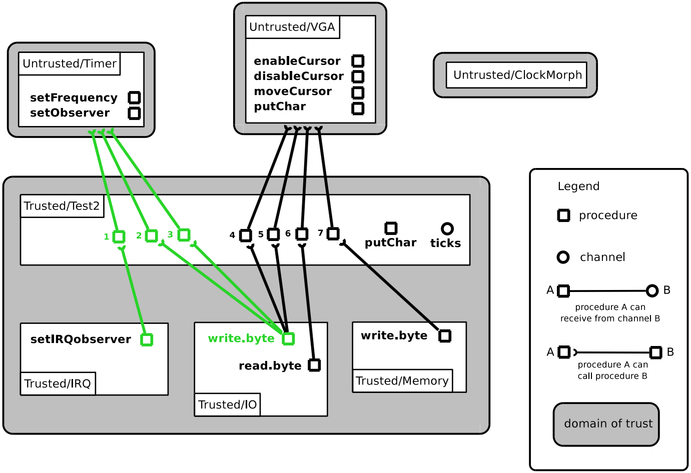
```
io.read.byte
```

```
\() = (io.read.byte 981)
```

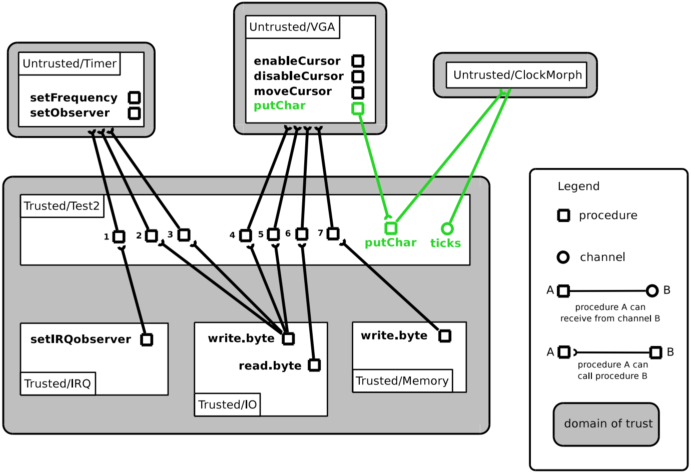
An example of “can call” relationship



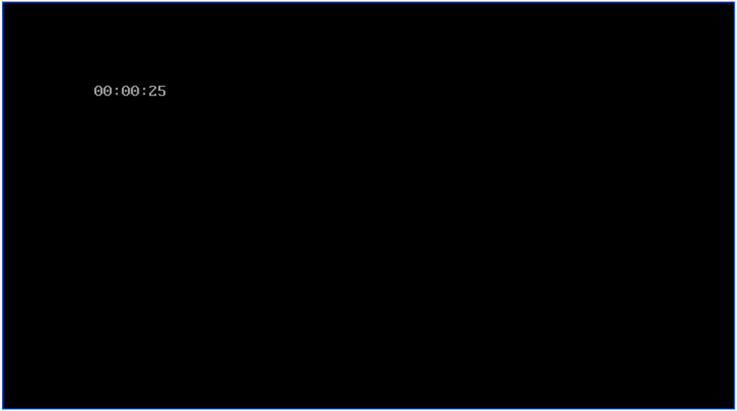
An example of “can call” relationship



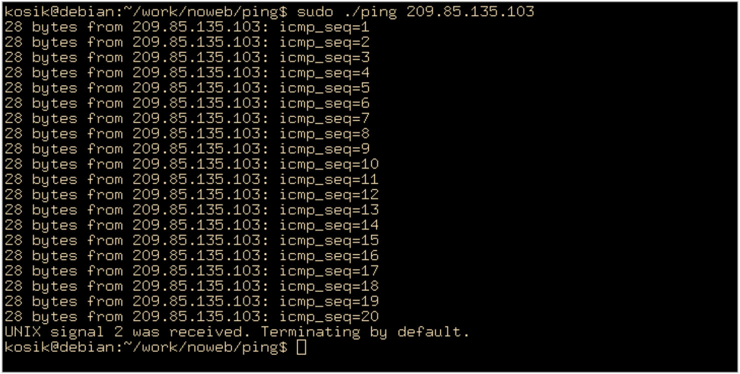
An example of “can call” relationship



Powerboxed `ClockMorph`



Powerboxed ping



```
kosik@debian:~/work/noweb/ping$ sudo ./ping 209.85.135.103
28 bytes from 209.85.135.103: icmp_seq=1
28 bytes from 209.85.135.103: icmp_seq=2
28 bytes from 209.85.135.103: icmp_seq=3
28 bytes from 209.85.135.103: icmp_seq=4
28 bytes from 209.85.135.103: icmp_seq=5
28 bytes from 209.85.135.103: icmp_seq=6
28 bytes from 209.85.135.103: icmp_seq=7
28 bytes from 209.85.135.103: icmp_seq=8
28 bytes from 209.85.135.103: icmp_seq=9
28 bytes from 209.85.135.103: icmp_seq=10
28 bytes from 209.85.135.103: icmp_seq=11
28 bytes from 209.85.135.103: icmp_seq=12
28 bytes from 209.85.135.103: icmp_seq=13
28 bytes from 209.85.135.103: icmp_seq=14
28 bytes from 209.85.135.103: icmp_seq=15
28 bytes from 209.85.135.103: icmp_seq=16
28 bytes from 209.85.135.103: icmp_seq=17
28 bytes from 209.85.135.103: icmp_seq=18
28 bytes from 209.85.135.103: icmp_seq=19
28 bytes from 209.85.135.103: icmp_seq=20
UNIX signal 2 was received. Terminating by default.
kosik@debian:~/work/noweb/ping$ 
```

Reference Graph Dynamics [1, §9.2]

- connectivity by initial conditions
- connectivity by parenthood
- connectivity by introduction
- connectivity by endowment

Contents

Motivation

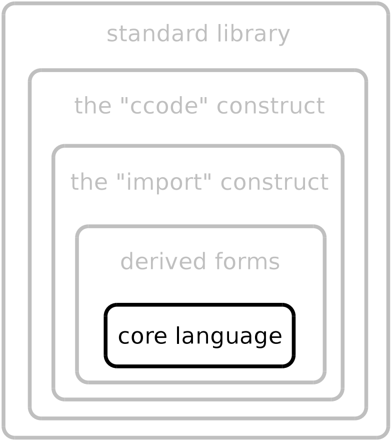
The object-capability paradigm

Pict specific details

Taming of Pict

Open Problems

Layers of the Pict programming language



standard library

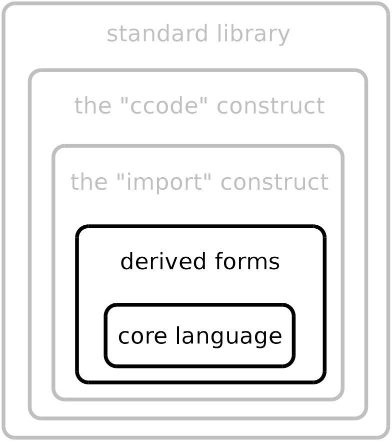
the "ccode" construct

the "import" construct

derived forms

core language

Layers of the Pict programming language



standard library

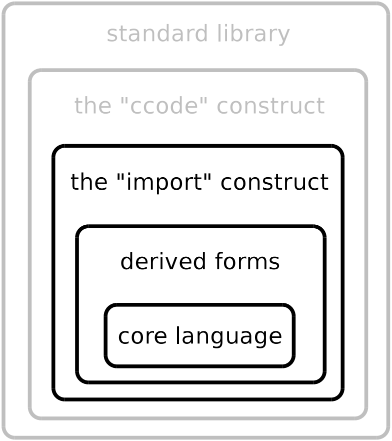
the "ccode" construct

the "import" construct

derived forms

core language

Layers of the Pict programming language



standard library

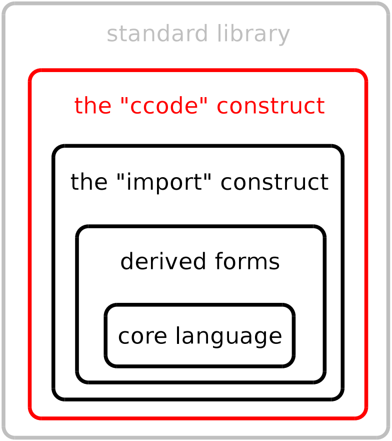
the "ccode" construct

the "import" construct

derived forms

core language

Layers of the Pict programming language


standard library

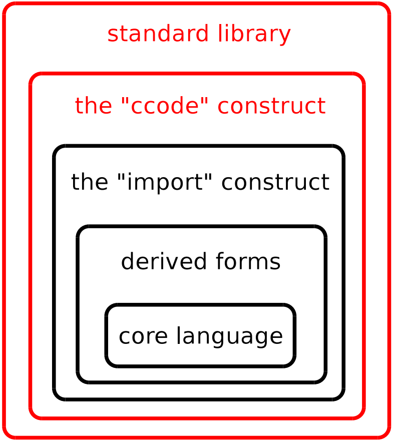
the "ccode" construct

the "import" construct

derived forms

core language

Layers of the Pict programming language



standard library

the "ccode" construct

the "import" construct

derived forms

core language

Contents

Motivation

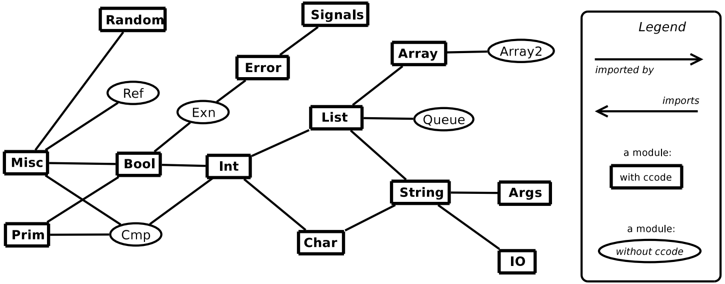
The object-capability paradigm

Pict specific details

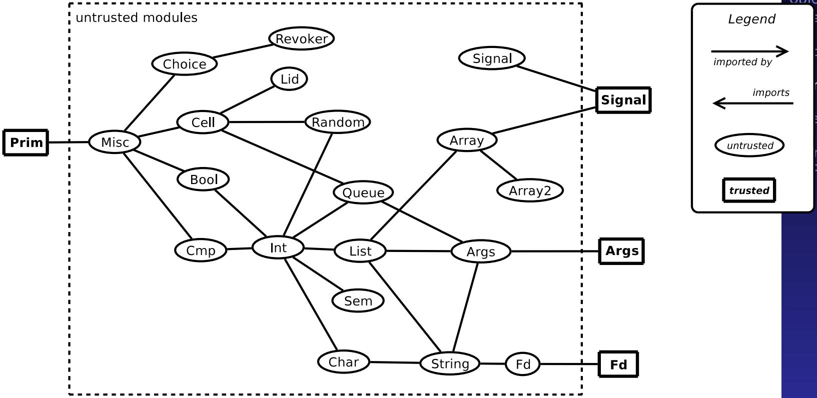
Taming of Pict

Open Problems

The original standard library



Refactored library



Contents

Motivation

The object-capability paradigm

Pict specific details

Taming of Pict

Open Problems

Open Problems

- any untrusted component can consume as much memory as it wishes
- any untrusted component can consume as much CPU ticks as it wishes
- formal methods can significantly refine assessments of upperbound of threat we face

References

Mark Samuel Miller.
Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control.
PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA, May 2006.

Robin Milner.
Communicating and Mobile Systems: The π-calculus.
Cambridge University Press, 1999.

The original π-calculus (the syntax) [2]

$$\begin{array}{rcl}
\pi & ::= & \overline{x}y \mid x(z) \mid \tau \mid [x = y]\pi \\
P & ::= & (P \mid P) \mid \nu z P \mid !P \mid M \\
M & ::= & 0 \mid \pi.P \mid M + M
\end{array}$$

Matching action was omitted

$$\begin{aligned}
\pi &::= \bar{x}y \mid x(z) \mid \tau \\
P &::= (P \mid P) \mid \nu z P \mid !P \mid M \\
M &::= 0 \mid \pi.P \mid M + M
\end{aligned}$$

Silent action was omitted

$$
\begin{aligned}
\pi &::= \overline{x}y \mid x(z) \\
P &::= (P \mid P) \mid \nu z P \mid\ !P \mid M \\
M &::= 0 \mid \pi.P \mid M + M
\end{aligned}
$$

The Choice operator was omitted

$$\pi \quad ::= \quad \overline{x}y \;\mid\; x(z)$$

$$P \quad ::= \quad (P \mid P) \;\mid\; \nu z P \;\mid\; !P \;\mid\; M$$

$$M \quad ::= \quad 0 \;\mid\; \pi.P$$

Only asynchronous sends are allowed

$$P \;::=\; (P \mid P) \;\mid\; \nu z P \;\mid\; !P \;\mid\; M$$

$$M \;::=\; 0 \;\mid\; \overline{x}y.0 \;\mid\; x(z).P$$

It does not make sense to replicate $\mathbf{\overline{x}y.0}$ particles

$$\mathsf{P} \quad ::= \quad (\mathsf{P} \mid \mathsf{P}) \;\mid\; \nu \mathsf{z} \mathsf{P} \;\mid\; \nu \mathsf{x}\,(!\mathsf{x}(\mathsf{z}).\mathsf{P} \mid \mathsf{P}) \;\mid\; \mathsf{M}$$

$$\mathsf{M} \quad ::= \quad 0 \;\mid\; \overline{\mathsf{x}}\mathsf{y}.0 \;\mid\; \mathsf{x}(\mathsf{z}).\mathsf{P}$$

Cleaned up grammar of this subcalculus

$$
\begin{array}{lll}
\mathsf{P} & ::= & \mathsf{0} \\
 & \mid & \overline{\mathsf{x}}\mathsf{y.0} \\
 & \mid & \mathsf{x(z).P} \\
 & \mid & \nu\mathsf{zP} \\
 & \mid & (\mathsf{P} \mid \mathsf{P}) \\
 & \mid & \nu\mathsf{x}\,(\,!\mathsf{x(z).P} \;\mid\; \mathsf{P}\,)
\end{array}
$$

. . . in ASCII

```
P  ::=  ()
     |  xy
     |  x?z = Proc
     |  (new z:^Type Proc)
     |  (Proc | Proc)
     |  (def x = Proc Proc)
```