Speculo

Shared memory made easy

Alberto Mardegan, Luxoft (and formerly Nokia, Canonical)

Motivation and goals

- D-Bus perceived as a bottleneck
- Alternative proposed: T-Bus
- Smaller scope: just a SHM library

Speculo: interlingua for "mirror"

- 2k LOC on top of shm_open and mmap
- Lockless
- Zerocopy
- POSIX

Design principles

- Data committed to the SHM area is immutable
- Only a few control bytes can change value
- Readers open the SHM area in read-only mode

Useful concepts

Memory barriers:

```
atomic_store(address, value)
```

Guarantee that once value becomes visible to other processes, also anything which was written to RAM before it, is visible.

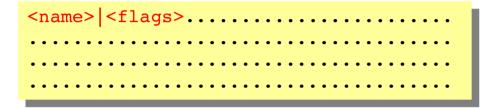
Step-by-step

We are going to see how data is written to the SHM area

(the reading part is left as a mental exercise to the audience)

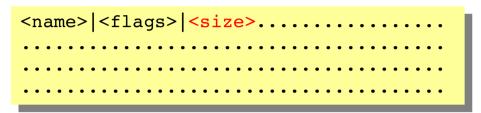
Structure on the heap

speculo_area_new(name, flags)



Structure on the heap

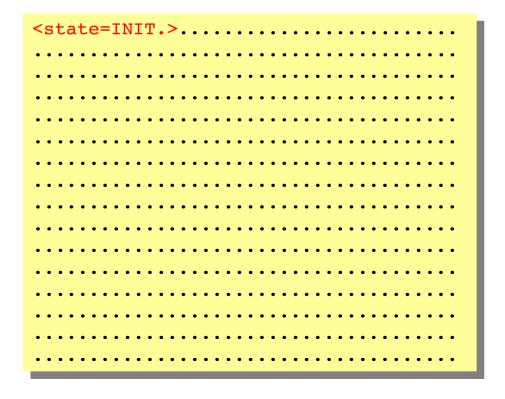
```
speculo_area_new(name, flags)
speculo_area_set_max_size(area, size)
```



```
speculo_area_new(name, flags)
speculo_area_set_max_size(area, size)
speculo_area_create(area)
```

Structure on the heap

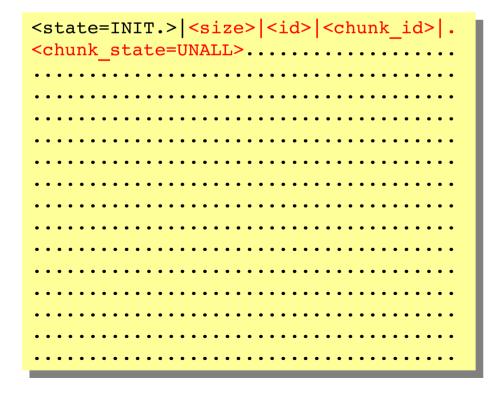
	<name> <flags> <size></size></flags></name>
	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •



```
speculo_area_new(name, flags)
speculo_area_set_max_size(area, size)
speculo_area_create(area)
```

Structure on the heap

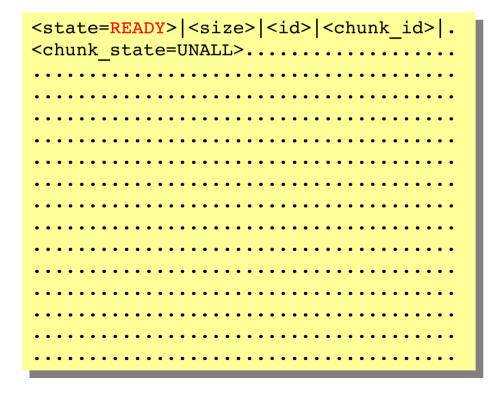
	<name> <flags> <size></size></flags></name>
	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •



```
speculo_area_new(name, flags)
speculo_area_set_max_size(area, size)
speculo area create(area)
```

Structure on the heap

<name> <flags> <size></size></flags></name>	
• • • • • • • • • • • • • • • • • • • •	• • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • •
•••••	



```
speculo_area_new(name, flags)
speculo_area_set_max_size(area, size)
speculo area create(area)
```

At this point, a manifest file gets written

- Tells the reader how to locate the SHM
- Always gets updated atomically

Structure on the heap

<name> <flags> <size></size></flags></name>	
• • • • • • • • • • • • • • • • • • • •	
• • • • • • • • • • • • • • • • • • • •	• • • • •
• • • • • • • • • • • • • • • • • • • •	• • • •

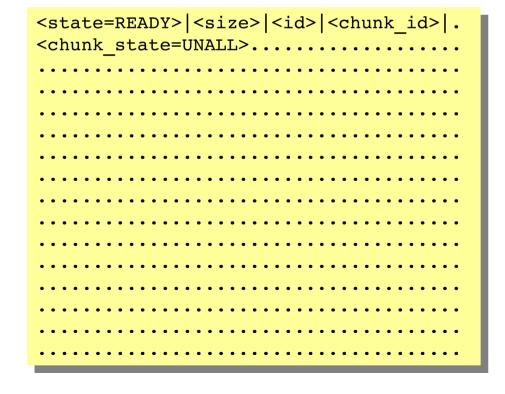
<pre><state=ready> <size> <id> <chunk_id> .</chunk_id></id></size></state=ready></pre>
<pre><chunk_state=unall></chunk_state=unall></pre>
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
•••••
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •

```
speculo_area_new(name, flags)
speculo_area_set_max_size(area, size)
speculo area create(area)
```

A reader would see that no chunks have been written

Structure on the heap

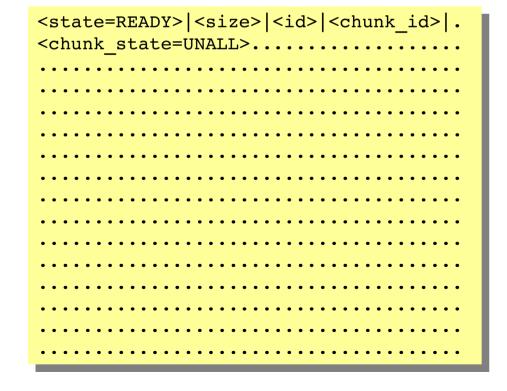
	<name> <flags> <size></size></flags></name>
	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •



Just initializes a struct on the stack

Structure on the heap

•••••	<name> <flags> <size></size></flags></name>
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
•••••	• • • • • • • • • • • • • • • • • • • •



A new chunk ID is assigned; the one stored in the area is incremented

Structure on the heap

<name> <flags> <size></size></flags></name>	
• • • • • • • • • • • • • • • • • • • •	• • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • •
••••••	• • • • • •

<pre><state=ready> <size> <id> <chunk_id> .</chunk_id></id></size></state=ready></pre>
<pre><chunk_state=unall> <chunk_id> </chunk_id></chunk_state=unall></pre>
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •

Chunk size, expiration timestamp and other few control bytes are written

A status marker is added after the end, for the next chunk

Structure on the heap

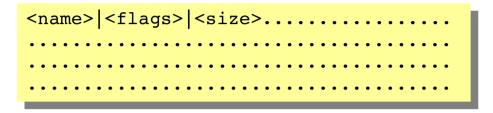
	<name> <flags> <size></size></flags></name>
	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •

```
<state=READY>|<size>|<id>|<chunk_id>|.
<chunk_state=UNALL>|<chunk_id>|<....>
<chunk_size>|
....
<chunk_state=UNALL>|
```

The state is updated

Readers are still ignoring this chunk

Structure on the heap



```
<state=READY>|<size>|<id>|<chunk_id>|.
<chunk_state=ALLOC>|<chunk_id>|<....>
<chunk_size>|.....
<chunk_state=UNALL>|
```

Structure on the heap

	<name> <flags> <size></size></flags></name>
	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •

<pre><chunk_state=alloc> <chunk_id> <> <chunk_size> Hello world!\0</chunk_size></chunk_id></chunk_state=alloc></pre> <pre><chunk_state=unall> </chunk_state=unall></pre>	<pre><state=ready> <size> <id> <chunk_id> .</chunk_id></id></size></state=ready></pre>
<pre>chunk_state=UNALL> </pre>	<pre><chunk_size> Hello world!\0</chunk_size></pre>
<ah.day.chunk_state=unall> </ah.day.chunk_state=unall>	
	<chunk_state=unall></chunk_state=unall>
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •
	• • • • • • • • • • • • • • • • • • • •

The chunk becomes visible to the readers

Structure on the heap

<name> <flags> <size></size></flags></name>	• • • •
• • • • • • • • • • • • • • • • • • • •	
• • • • • • • • • • • • • • • • • • • •	
• • • • • • • • • • • • • • • • • • • •	• • • •

<pre><state=ready> <size> <id> <chunk_id> .</chunk_id></id></size></state=ready></pre>
<pre><chunk_state=writn> <chunk_id> <></chunk_id></chunk_state=writn></pre>
<pre><chunk_size> Hello world! \0</chunk_size></pre>
<chunk_state=unall></chunk_state=unall>
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •

Writing one more chunk

Structure on the heap

<name></name>	<flags></flags>	<size></size>
• • • • • •	• • • • • • •	• • • • • • • • • • • • • • • • • • • •
• • • • • •	• • • • • • •	• • • • • • • • • • • • • • • • • • • •
• • • • • •	• • • • • • •	• • • • • • • • • • • • • • • • • • • •

```
<state=READY>|<size>|<id>|<chunk_id>|.
<chunk_state=WRITN>|<chunk_id>|<...>
<chunk_size>|Hello world!\0.....
.....</chunk_state=WRITN>|
<chunk_id>|<...><chunk_size>|Once upon
a time someone started to write a nice
story, but then something happened and
the need to justify the text perfectly
turned the story a bit meaningless, or
at least not as interesting as someone
hoped it would have been but who cares
<chunk_state=UNALL>|......
```

When there's not enough room for writing the next chunk, a new SHM area gets allocated.

- Valid chunks get copied over
- Invalid data:
 - Expired chunks (by timestamp)
 - Obsoleted chunks (will explain later)

Structure on the heap

•••••	

<pre><state=ready> <size> <id> <chunk_id> .</chunk_id></id></size></state=ready></pre>
<pre><chunk state="WRITN"> <chunk id=""> <></chunk></chunk></pre>
<pre><chunk_size> Hello world!\0</chunk_size></pre>
•••••
<chunk state="WRITN"></chunk>
<pre><chunk_id> <><chunk_size> Once upon</chunk_size></chunk_id></pre>
a time someone started to write a nice
story, but then something happened and
the need to justify the text perfectly
turned the story a bit meaningless, or
at least not as interesting as someone
hoped it would have been but who cares
<pre><chunk_state=unall> </chunk_state=unall></pre>
•••••
• • • • • • • • • • • • • • • • • • • •
•••••
••••••

When there's not enough room for writing the next chunk, a new SHM area gets allocated.

- Valid chunks get copied over
- Invalid data:
 - Expired chunks (by timestamp)
 - Obsoleted chunks (will explain later)
- Manifest file is updated to point to the new SHM

Structure on the heap

<name> <flags> <size></size></flags></name>	
•••••	
•••••	
••••••	

<pre><state=ready> <size> <id> <chunk_id> .</chunk_id></id></size></state=ready></pre>
<pre><chunk state="WRITN"> <chunk id=""> <></chunk></chunk></pre>
<pre><chunk_size> Hello world!\0</chunk_size></pre>
• • • • • • • • • • • • • • • • • • • •

When there's not enough room for writing the next chunk, a new SHM area gets allocated.

- Valid chunks get copied over
- Invalid data:
 - Expired chunks (by timestamp)
 - Obsoleted chunks (will explain later)
- Manifest file is updated to point to the new SHM
- Old memory area is marked as obsolete

Structure on the heap

<name> <flags> <size></size></flags></name>	• • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •	• • • • • • • • • • • • • •

<pre><state=obsol> <size> <id> <chunk_id> .</chunk_id></id></size></state=obsol></pre>
<pre><chunk_state=writn> <chunk_id> <></chunk_id></chunk_state=writn></pre>
<pre><chunk_size> Hello world!\0</chunk_size></pre>
•••••
<chunk state="WRITN"></chunk>
<pre><chunk_id> <><chunk_size> Once upon</chunk_size></chunk_id></pre>
a time someone started to write a nice
story, but then something happened and
the need to justify the text perfectly
turned the story a bit meaningless, or
at least not as interesting as someone
hoped it would have been but who cares
<pre><chunk_state=unall> </chunk_state=unall></pre>
• • • • • • • • • • • • • • • • • • • •
•••••
•••••
•••••

When there's not enough room for writing the next chunk, a new SHM area gets allocated.

- Valid chunks get copied over
- Invalid data:
 - Expired chunks (by timestamp)
 - Obsoleted chunks (will explain later)
- Manifest file is updated to point to the new SHM
- Old memory area is marked as obsolete
- Old memory area is unlinked (current readers can still read it)

Structure on the heap

•••••	

<pre><state=obsol> <size> <id> <chunk_id> .</chunk_id></id></size></state=obsol></pre>
<pre><chunk state="WRITN"> <chunk id=""> <></chunk></chunk></pre>
<pre><chunk_size> Hello world!\0</chunk_size></pre>
• • • • • • • • • • • • • • • • • • • •
 chunk state=WRITN>
<pre><chunk_id> <><chunk_size> Once upon</chunk_size></chunk_id></pre>
a time someone started to write a nice
story, but then something happened and
the need to justify the text perfectly
turned the story a bit meaningless, or
at least not as interesting as someone
hoped it would have been but who cares
<pre><chunk_state=unall> </chunk_state=unall></pre>
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •
• • • • • • • • • • • • • • • • • • • •

Obsoleted data

- Chunks which have expired (by timestamp)
- Chunks which have been updated:

```
ptr = speculo_chunk_update(&chunk, size)
```

allocates a new chunk, but keeps the same ID: readers will only consider the new instance

Speculo use cases

- 1) Stream of messages
 - Set an expiration time on each message
 - Each message sender is a writer
- 2) Publishing a data block
 - Data can be updated

Question time

Code: http://gitlab.org/mardy/speculo

Docs: http://mardy.gitlab.io/speculo/

Me:

E-mail: mardy@users.sourceforge.net

Blog: http://blog.mardy.it (English, Interlingua, Italian)

http://mardy.livejournal.com (in Russian)

Twitter: mardy78