Rust support in seL4 userspace

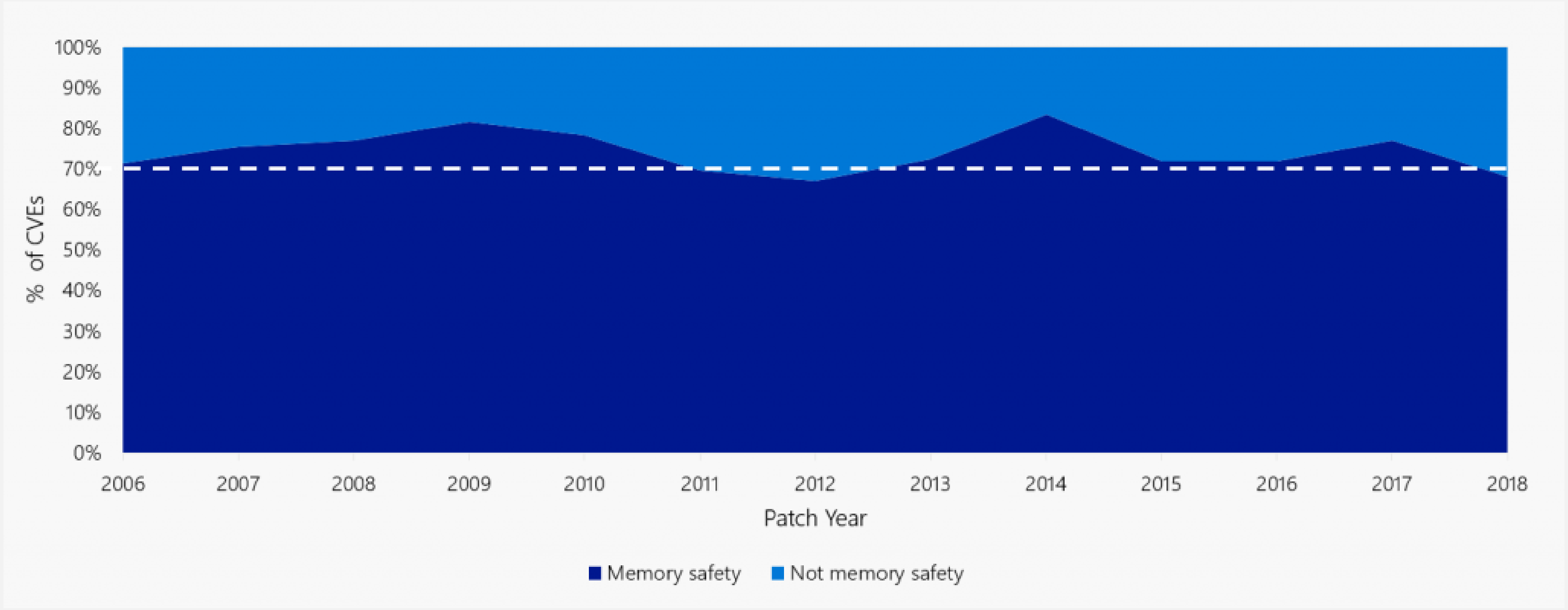
Present and future

Nick Spinale <nick@nickspinale.com>
seL4 Summit
October 11th, 2022

Colias Group

Memory safety

~70% of the vulnerabilities Microsoft assigns a CVE each year continue to be memory safety issues[1]

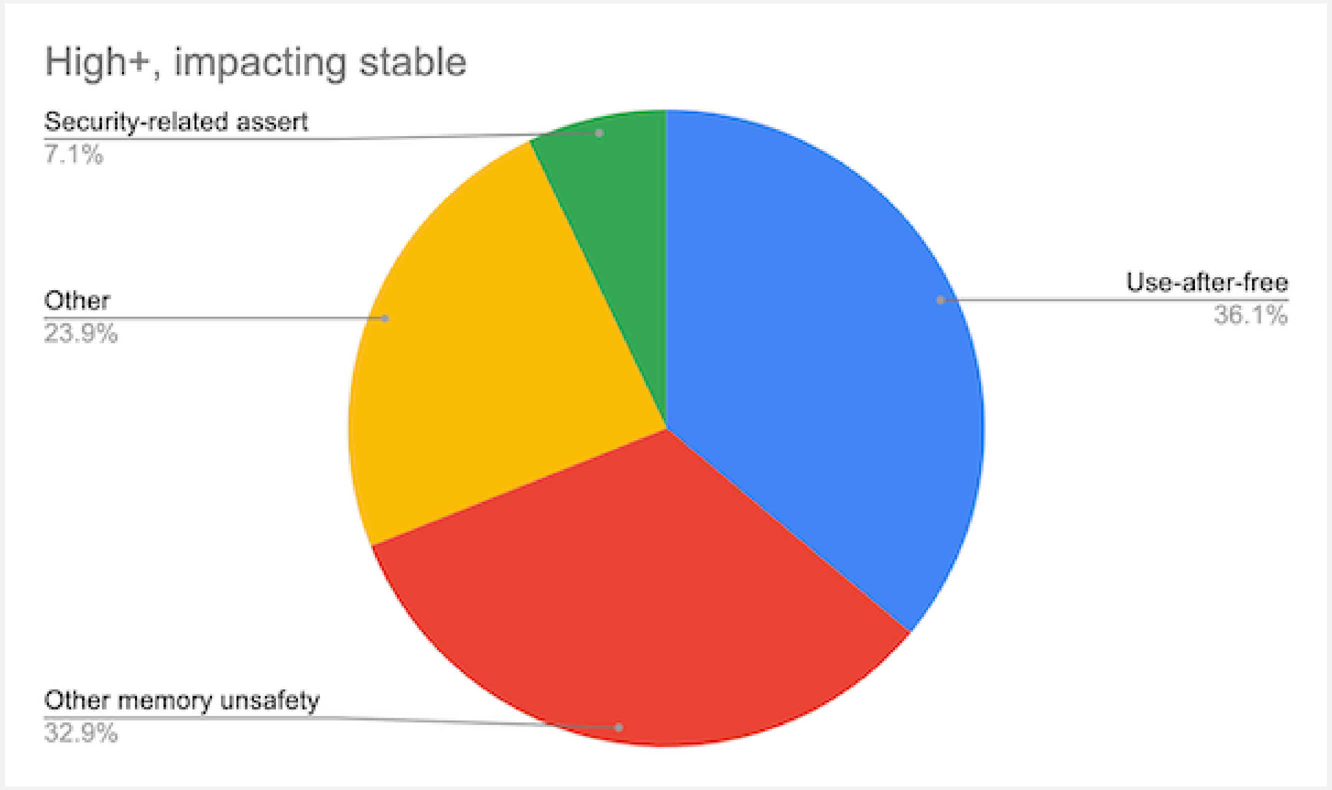


[1]https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/

Colias Group

Memory safety

The Chromium project finds that around 70% of [its] serious security bugs are memory safety problems[1]



[1]https://www.chromium.org/Home/chromium-security/memory-safety/

Colias Group

Rust

Enforces memory safety, without the overhead of a heavyweight language runtime, using compile-time analysis

Colias Group

Rust

Enforces memory safety, without the overhead of a heavyweight language runtime, using compile-time analysis

```rust
fn main() {
    let r;

    {
        let x = 5;
        r = &x;
    }

    println!("r: {}", r);
}
```

```
error[E0597]: `x` does not live long enough
 --> src/main.rs:6:13
  |
6 |         r = &x;
  |             ^^ borrowed value does not live long enough
7 |     }
  |     - `x` dropped here while still borrowed
8 |
9 |     println!("r: {}", r);
  |                       - borrow later used here
```

Rust

Enforces memory safety, without the overhead of a heavyweight language runtime, using compile-time analysis

Aims to provide “zero cost abstractions”

Rust

Enforces memory safety, without the overhead of a heavyweight language runtime, using compile-time analysis

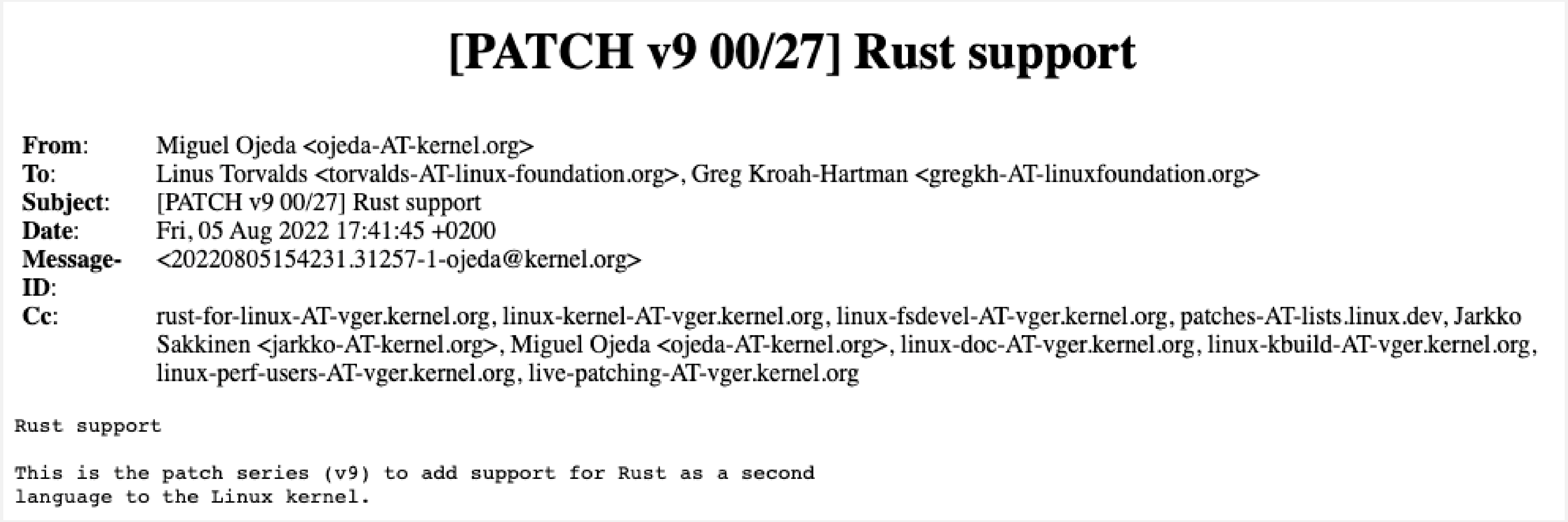
Aims to provide “zero cost abstractions”

Suitable for use cases from server to embedded, from OS kernel to application

Colias Group

Rust

Linux now supports Rust in the kernel (merged October 3rd, 2022)[1,2]



[PATCH v9 00/27] Rust support

From:	Miguel Ojeda <ojeda-AT-kernel.org>
To:	Linus Torvalds <torvalds-AT-linux-foundation.org>, Greg Kroah-Hartman <gregkh-AT-linuxfoundation.org>
Subject:	[PATCH v9 00/27] Rust support
Date:	Fri, 05 Aug 2022 17:41:45 +0200
Message-ID:	<20220805154231.31257-1-ojeda@kernel.org>
Cc:	rust-for-linux-AT-vger.kernel.org, linux-kernel-AT-vger.kernel.org, linux-fsdevel-AT-vger.kernel.org, patches-AT-lists.linux.dev, Jarkko Sakkinen <jarkko-AT-kernel.org>, Miguel Ojeda <ojeda-AT-kernel.org>, linux-doc-AT-vger.kernel.org, linux-kbuild-AT-vger.kernel.org, linux-perf-users-AT-vger.kernel.org, live-patching-AT-vger.kernel.org

```
Rust support

This is the patch series (v9) to add support for Rust as a second
language to the Linux kernel.
```

[1]https://lwn.net/ml/linux-kernel/20220805154231.31257-1-ojeda@kernel.org/
[2]https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=8aebac82933ff1a7c8eede18cab11e1115e2062b

Colias Group

Rust is a good fit for seL4 userspace

A high level language…

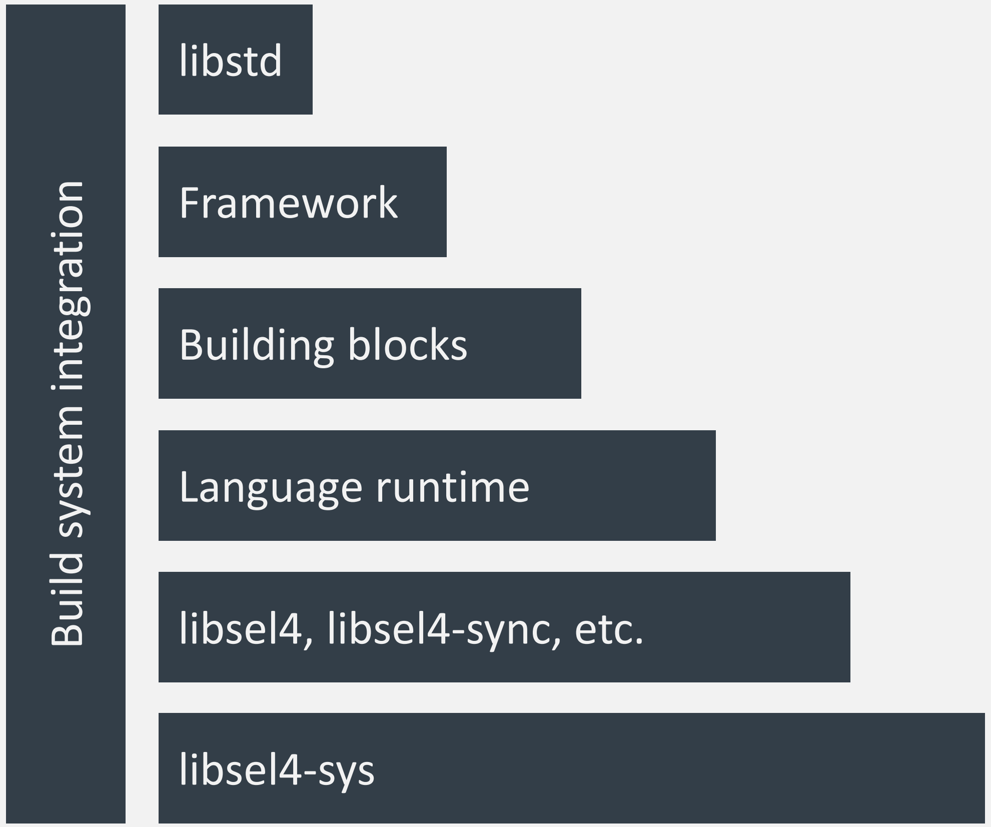
- Memory safety
- Abstraction
- Developer productivity (think drivers!)

…even for components without access to OS services

…even for resource-constrained systems

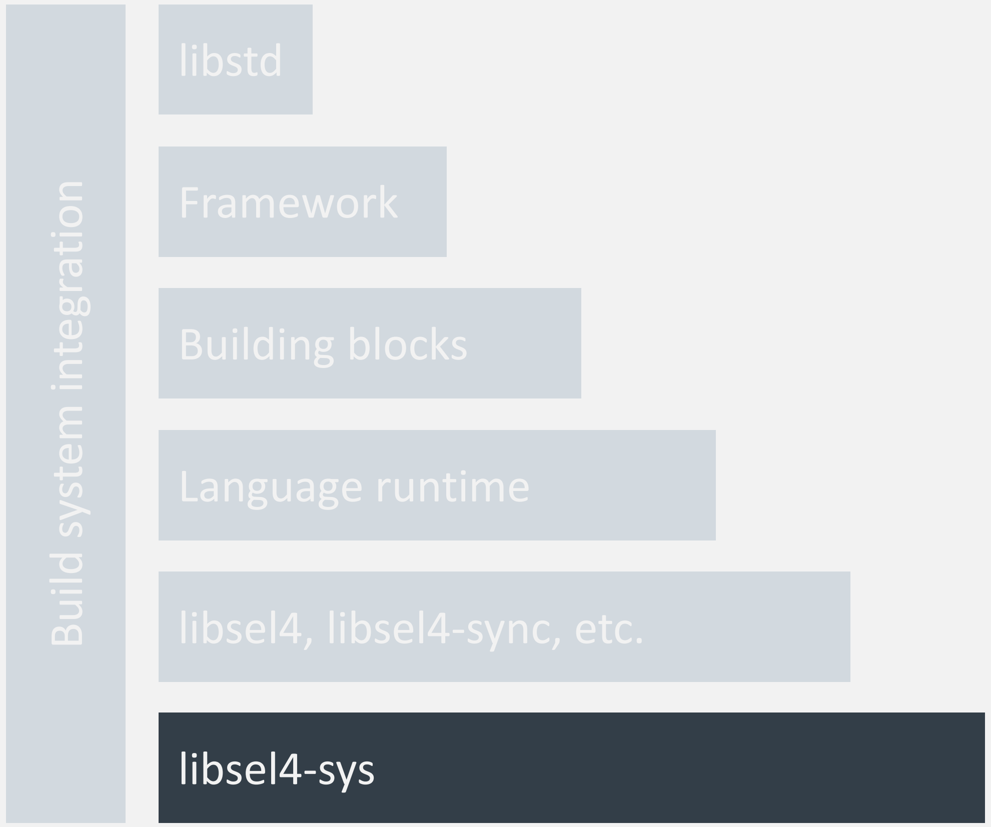
Low complexity and maintenance burden: no heavyweight language runtime to port

Levels of Rust support in seL4 userspace: a vision



Colias Group

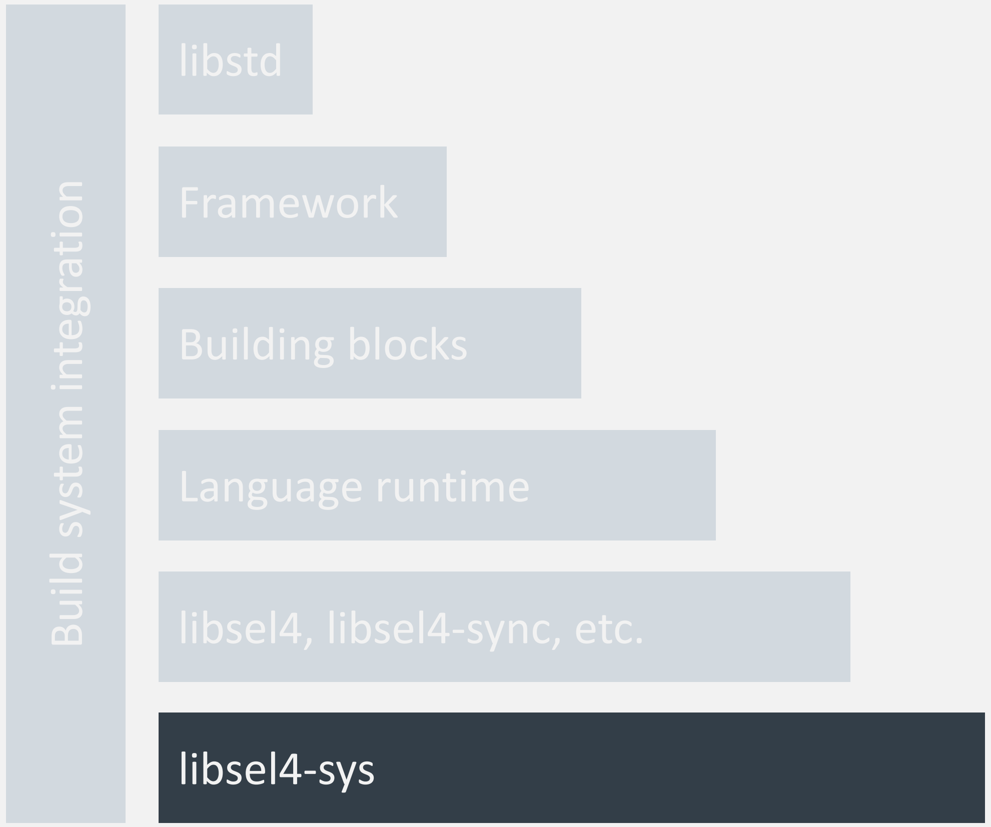
Levels of Rust support in seL4 userspace: a vision



Basic bindings to seL4 API

Colias Group

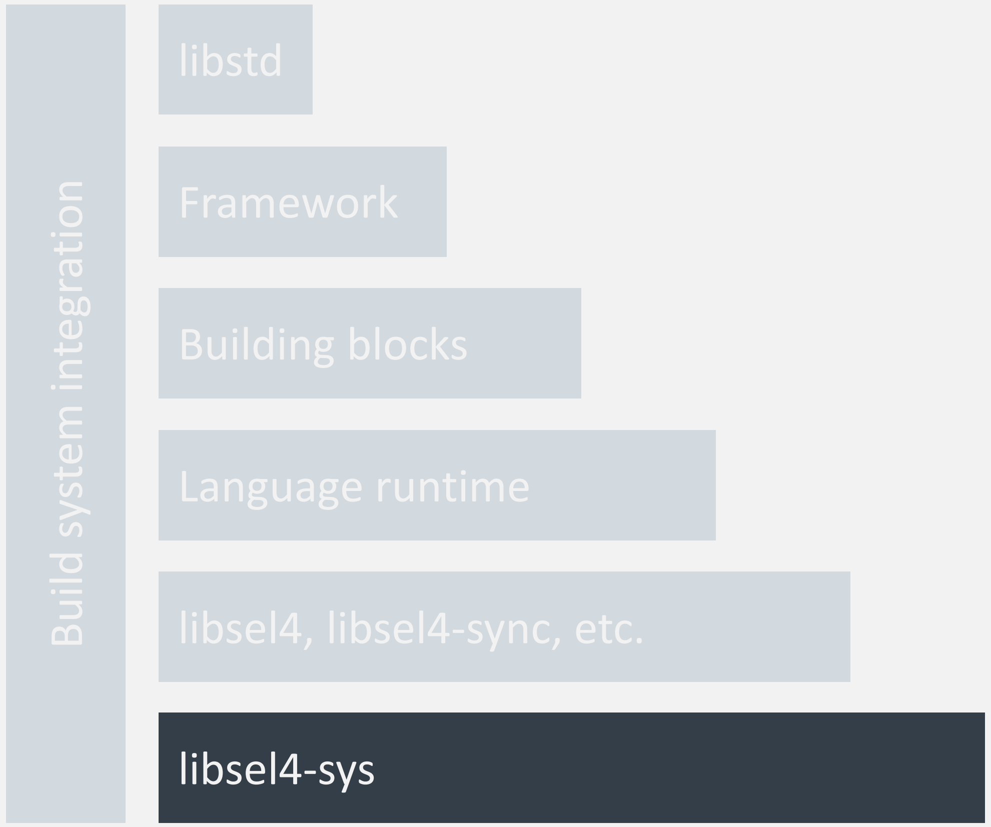
Levels of Rust support in seL4 userspace: a vision



```
type CPtr = ...;

#[repr(C)]
pub struct MessageInfo { ... }

#[repr(u32)]
enum Error { ... }

fn send(dest: CPtr, msg_info: MessageInfo) { ... }

fn tcb_suspend(service: TCB) -> Error { ... }

#[repr(C)]
pub struct IPCBuffer { ... }

#[thread_local]
pub static mut IPC_BUFFER: *mut IPCBuffer = ...;
```

Colias
Group

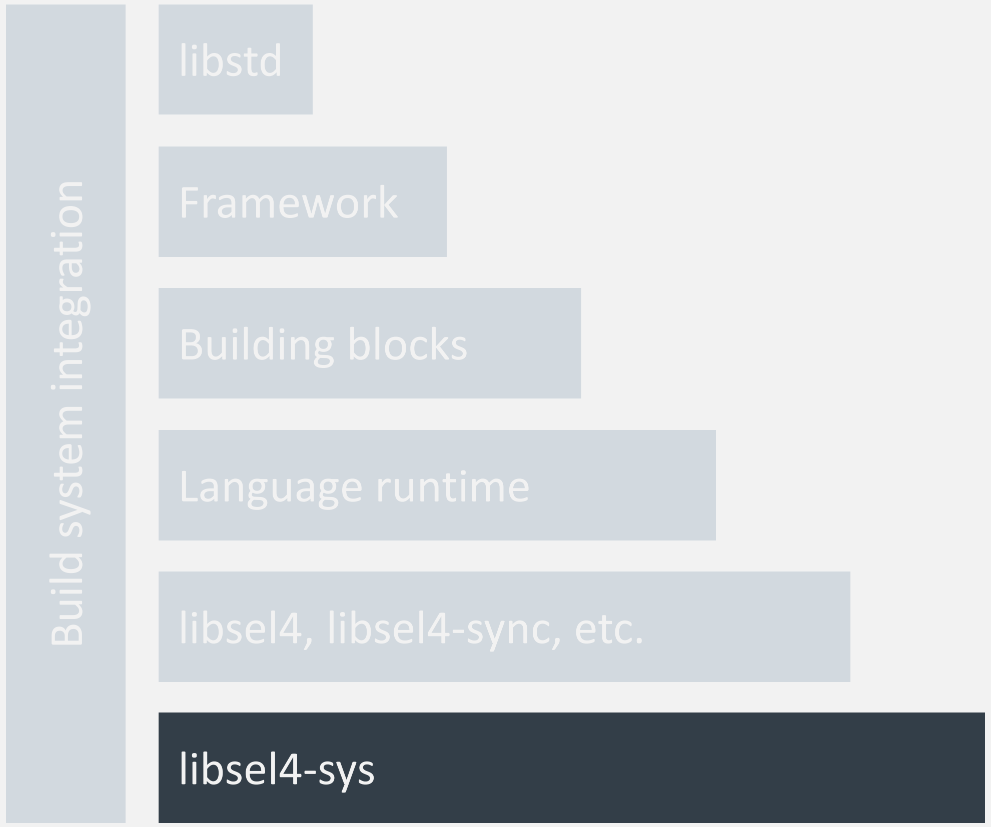
Levels of Rust support in seL4 userspace: a vision



Basic bindings to seL4 API

Colias Group

Levels of Rust support in seL4 userspace: a vision

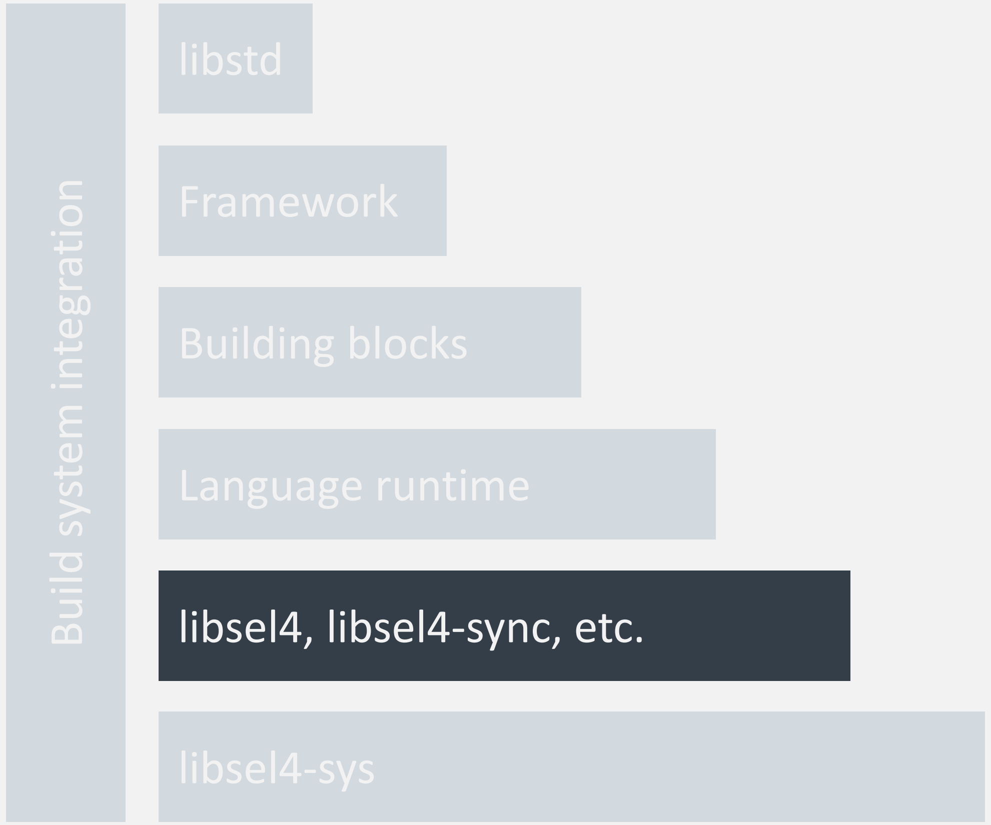


Basic bindings to seL4 API

Derived from C libsel4 or generated alongside it?

Colias Group

Levels of Rust support in seL4 userspace: a vision

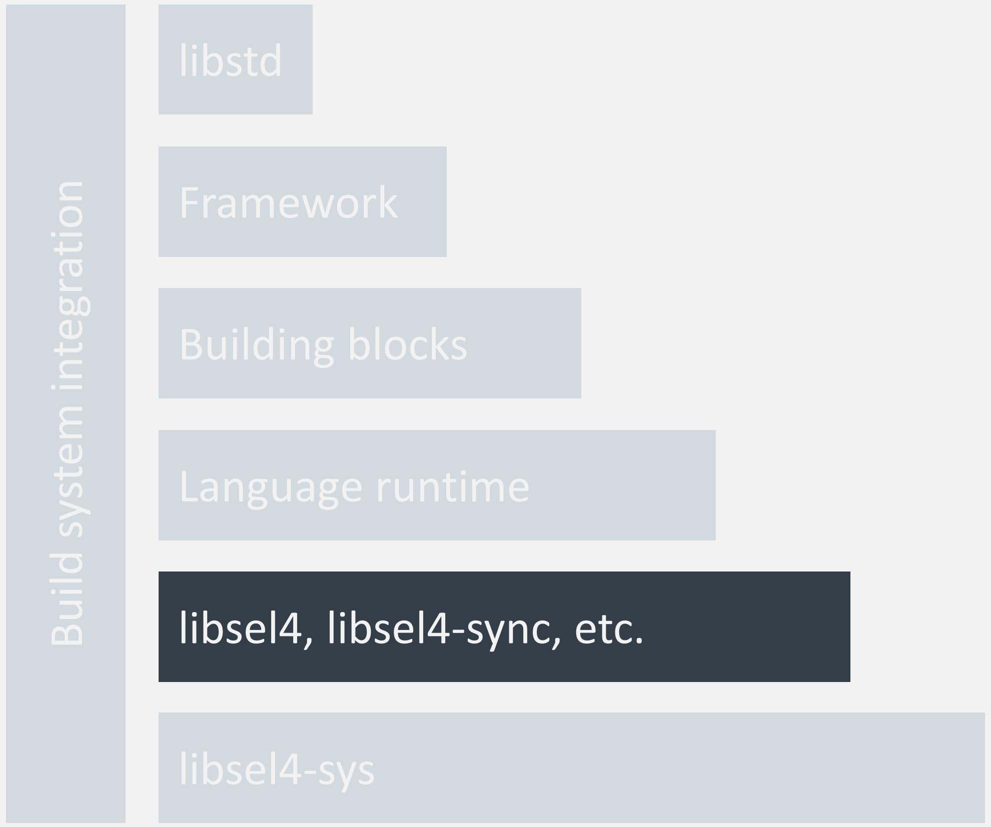


Idiomatic expression of the seL4 API and additional basic constructs

- Unopinionated
- Dependency-free
- Leverage Rust type system

Colias Group

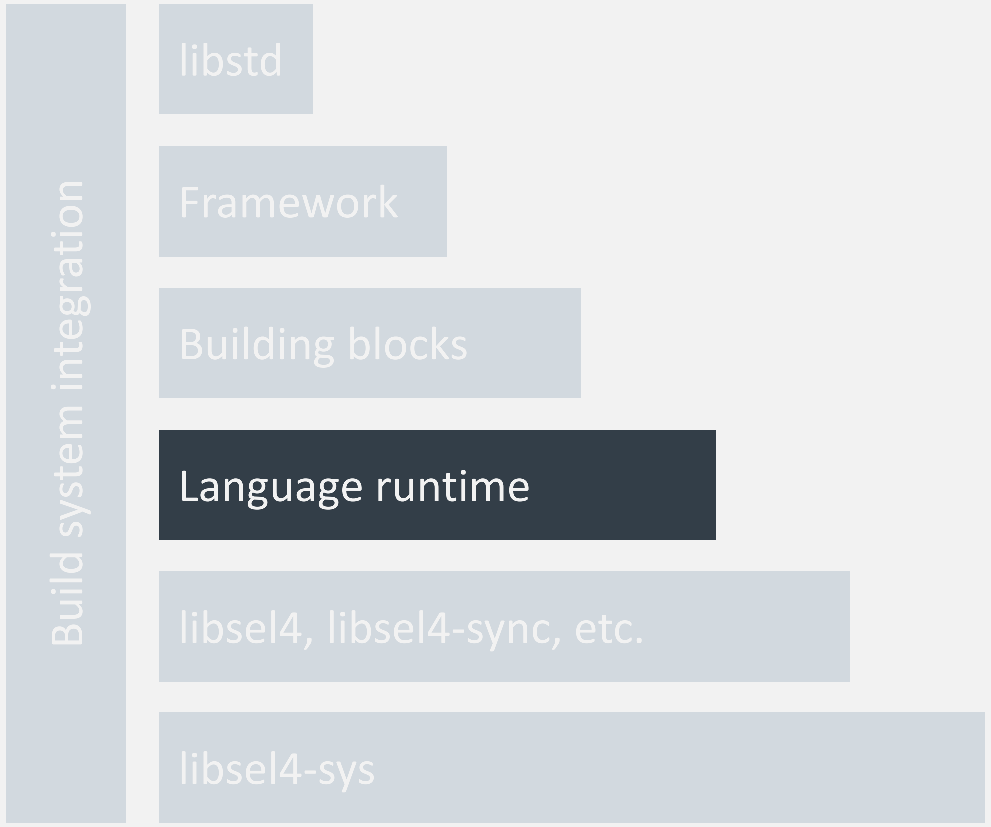
Levels of Rust support in seL4 userspace: a vision



```rust
impl TCB {
    pub fn suspend(&self) -> Result<()> { ... }
}

#[thread_local]
pub static IPC_BUFFER: RefCell<IPCBuffer> = ...;

pub struct Mutex { ... }

// example

let data = Mutex::new(nfn, 0);
{
    *data.lock().unwrap() += 1;
}
```

Colias Group

Levels of Rust support in seL4 userspace: a vision

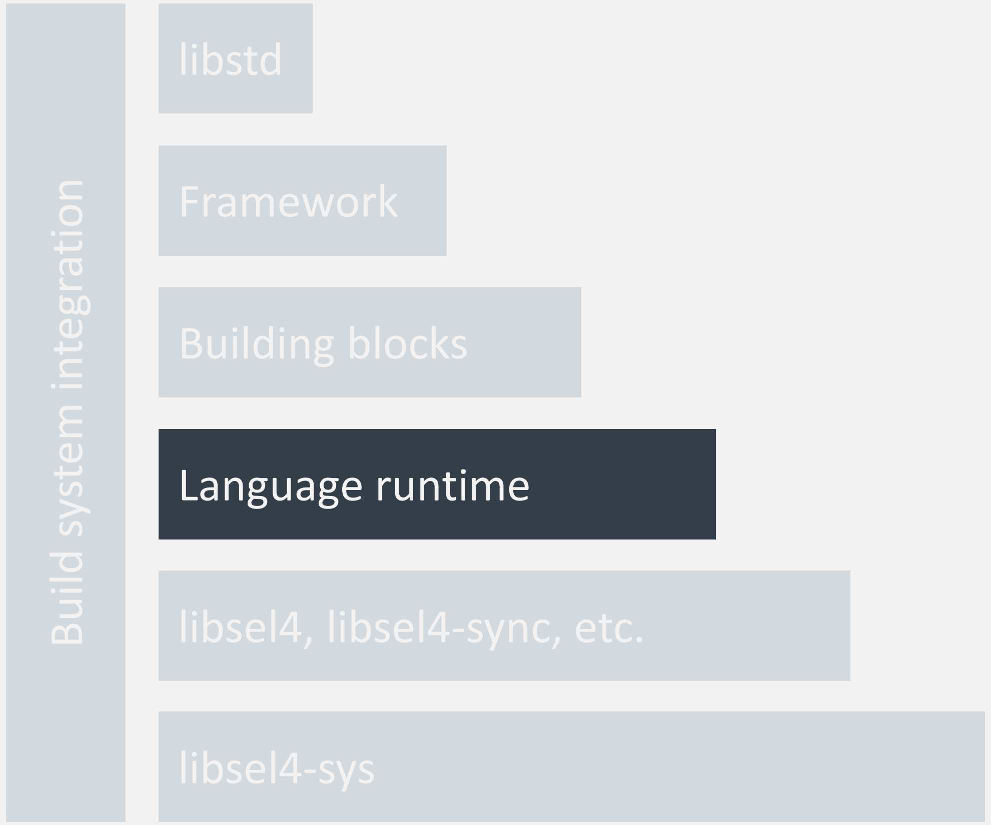


Configurable, minimal language runtime

- Entrypoint and process initialization
- Optional heap allocator
- Optional exception handling

Colias Group

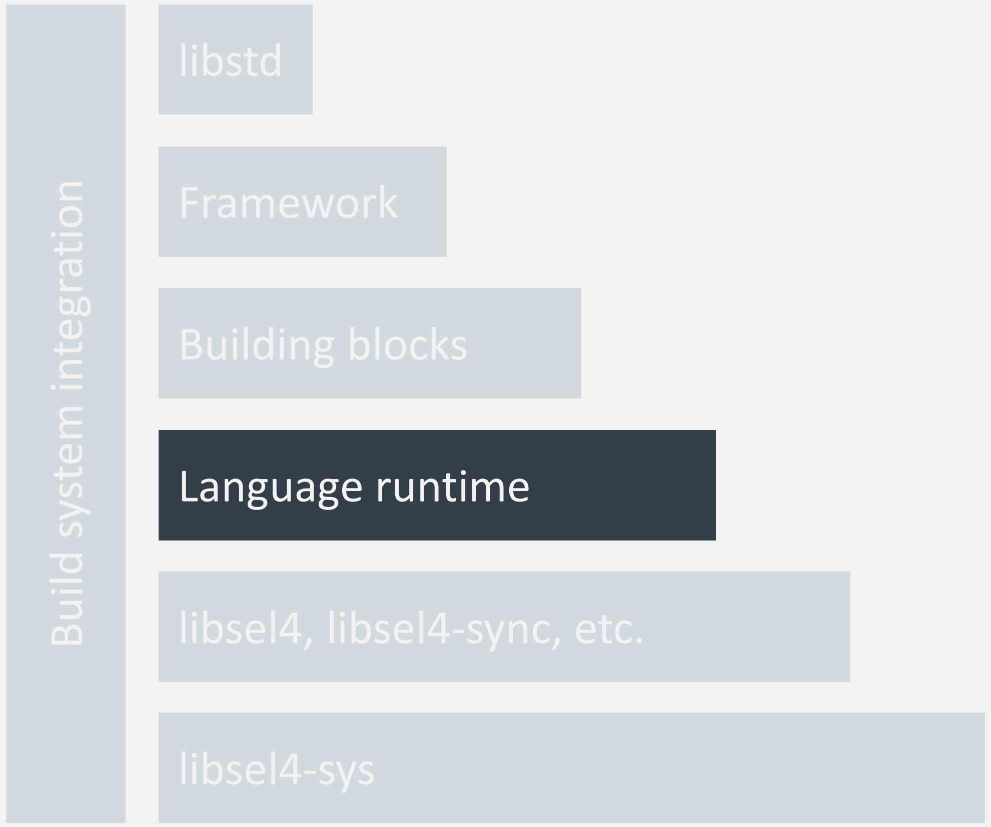
Levels of Rust support in seL4 userspace: a vision



```
#[root_task_main]
fn main(bootinfo: BootInfo) {
    debug_println!("{:#?}", bootinfo);

    let v = vec![0, 1, 2];
    debug_println!("{:?}", v);

    let result = catch_unwind(|| {
        panic!("uh oh");
    });
    assert!(result.is_err());
}
```

Colias Group

Levels of Rust support in seL4 userspace: a vision

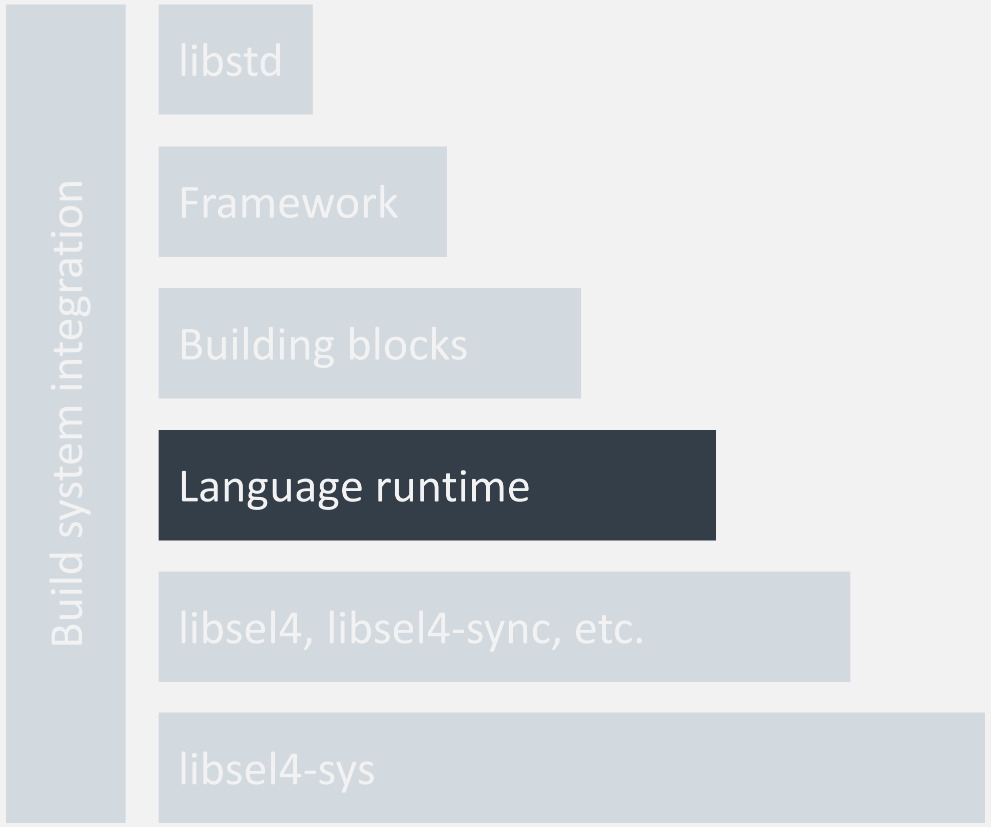


Configurable, minimal language runtime

- Entrypoint and process initialization
- Optional heap allocator
- Optional exception handling

Colias Group

Levels of Rust support in seL4 userspace: a vision



Configurable, minimal language runtime

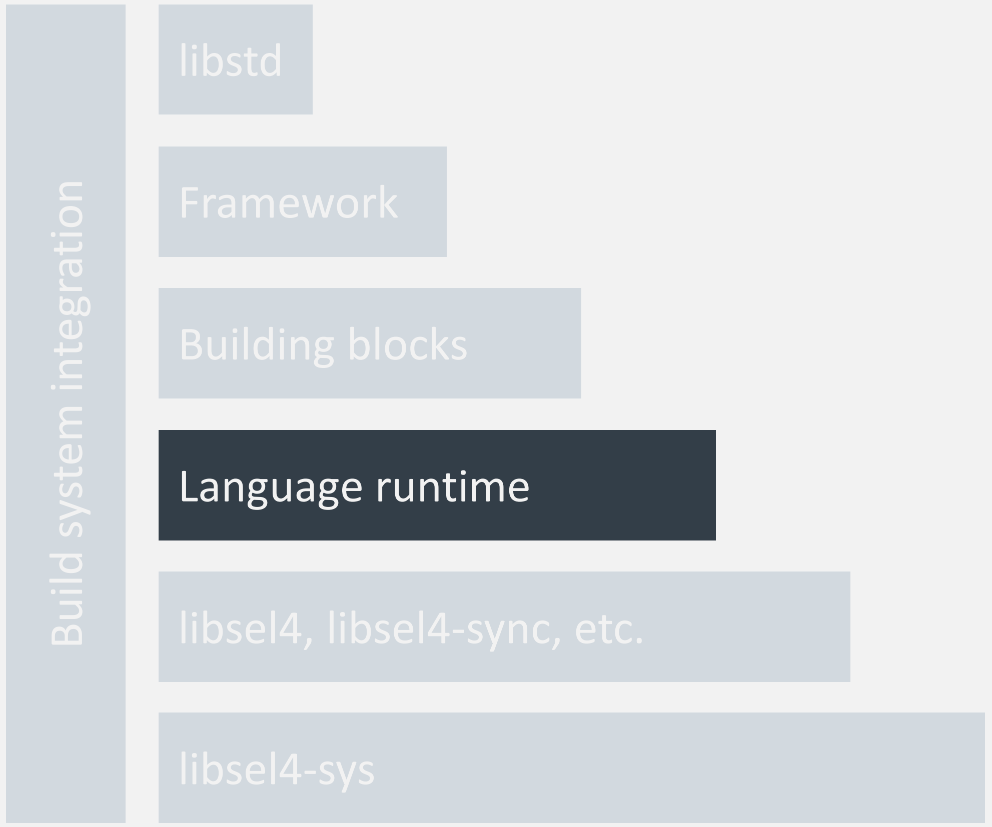
- Entrypoint and process initialization
- Optional heap allocator
- Optional exception handling

Configurability for…

- Require heap? Static or dynamic? Self-managed?
- Require exception handling?
- Root task or not? If not, what environment?
- What debugging facilities are available, and where?

Colias Group

Levels of Rust support in seL4 userspace: a vision



Configurable, minimal language runtime

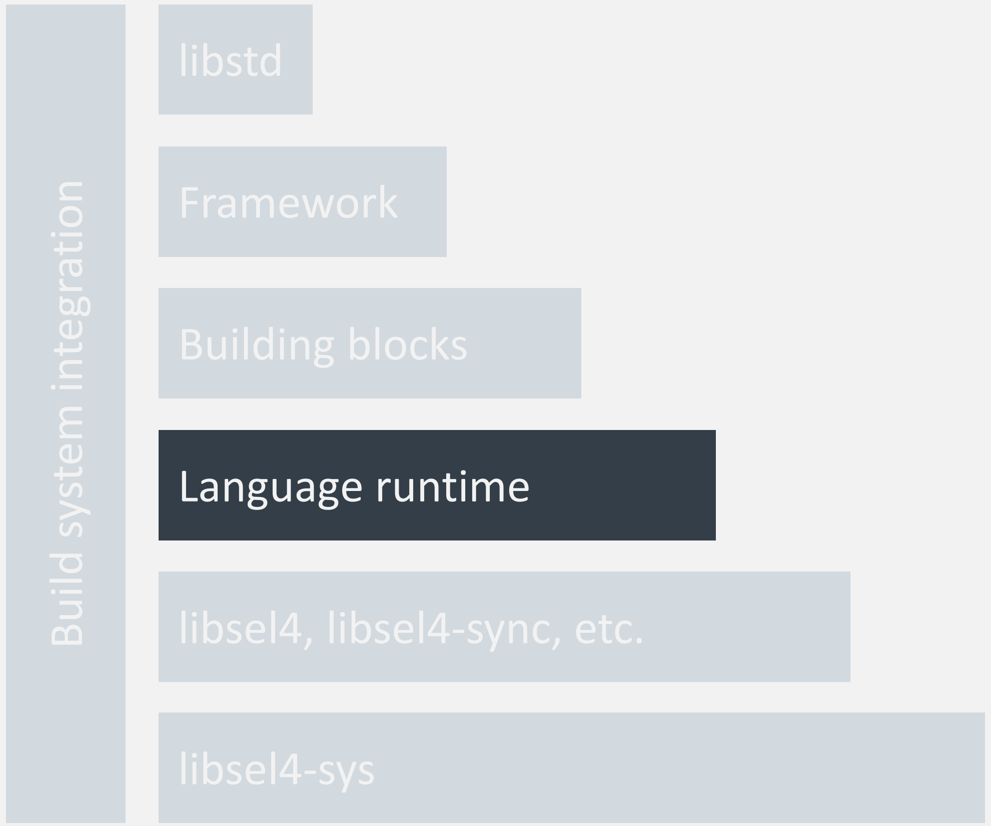
- Entrypoint and process initialization
- Optional heap allocator
- Optional exception handling

Configurability via…

- Build system-level abstraction
- Language-level abstraction
- Link-level abstraction

Colias Group

Levels of Rust support in seL4 userspace: a vision

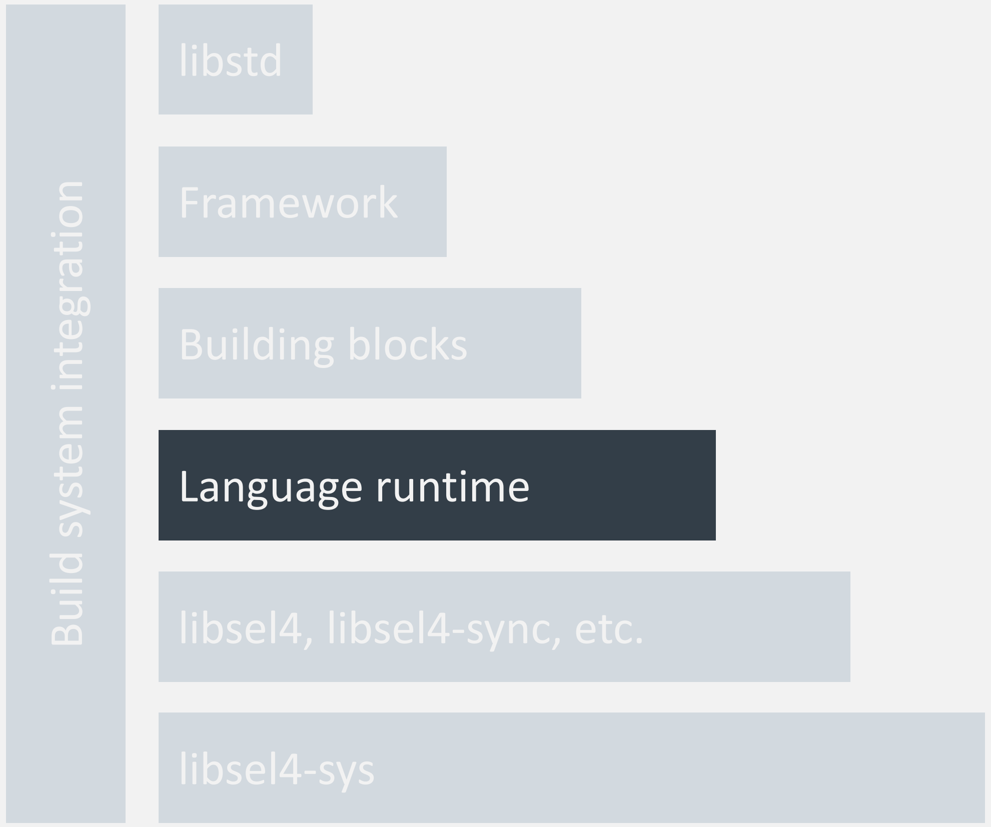


Configurable, minimal language runtime

- Entrypoint and process initialization
- Optional heap allocator
- Optional exception handling

Colias Group

Levels of Rust support in seL4 userspace: a vision

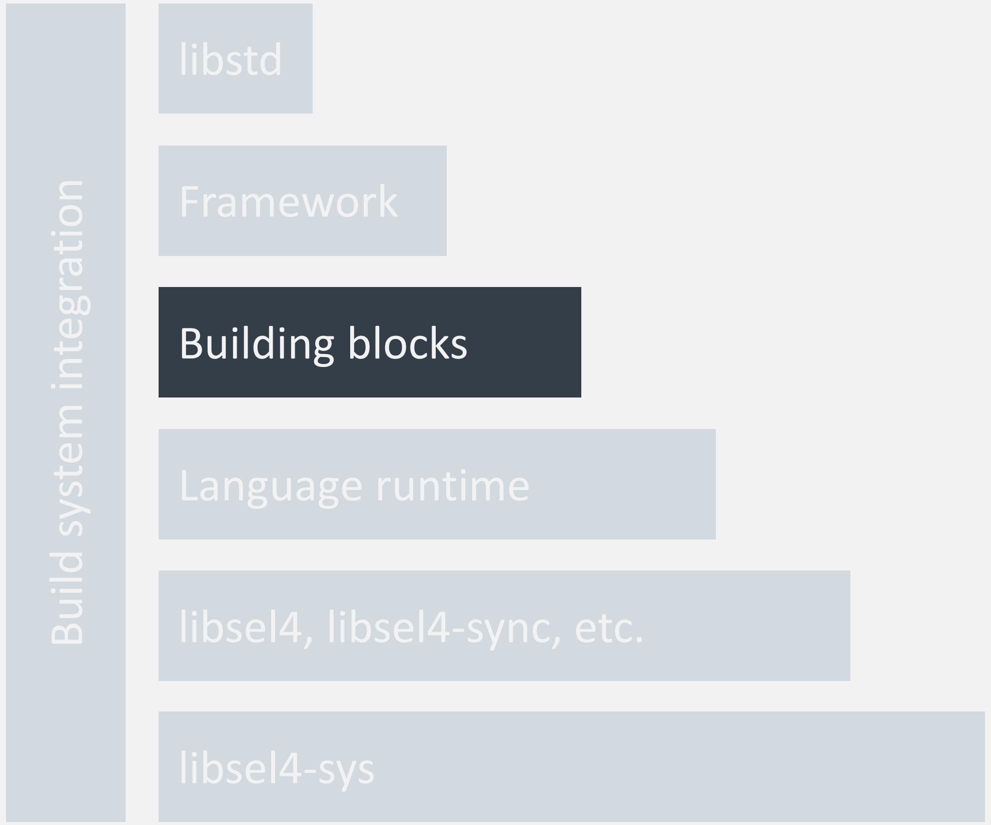


Configurable, minimal language runtime

- Entrypoint and process initialization
- Optional heap allocator
- Optional exception handling

Should this be compatible with libsel4runtime?

Levels of Rust support in seL4 userspace: a vision

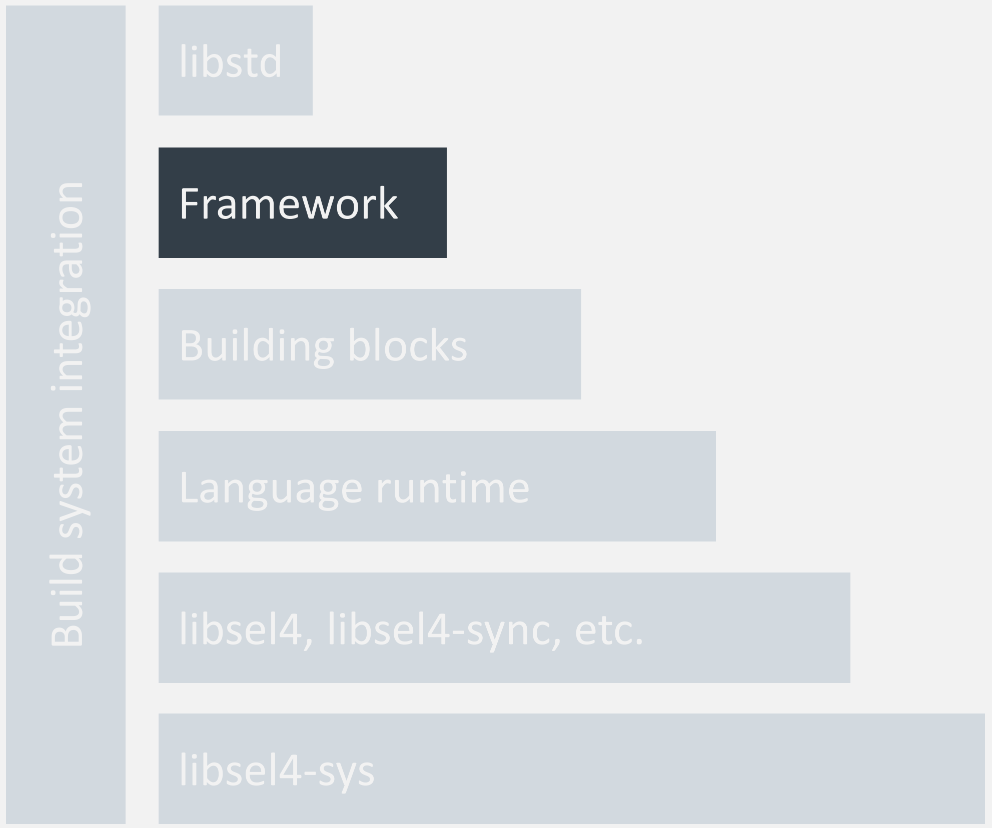


Reusable system building blocks

- Resource management libraries
- Drivers
- VMM

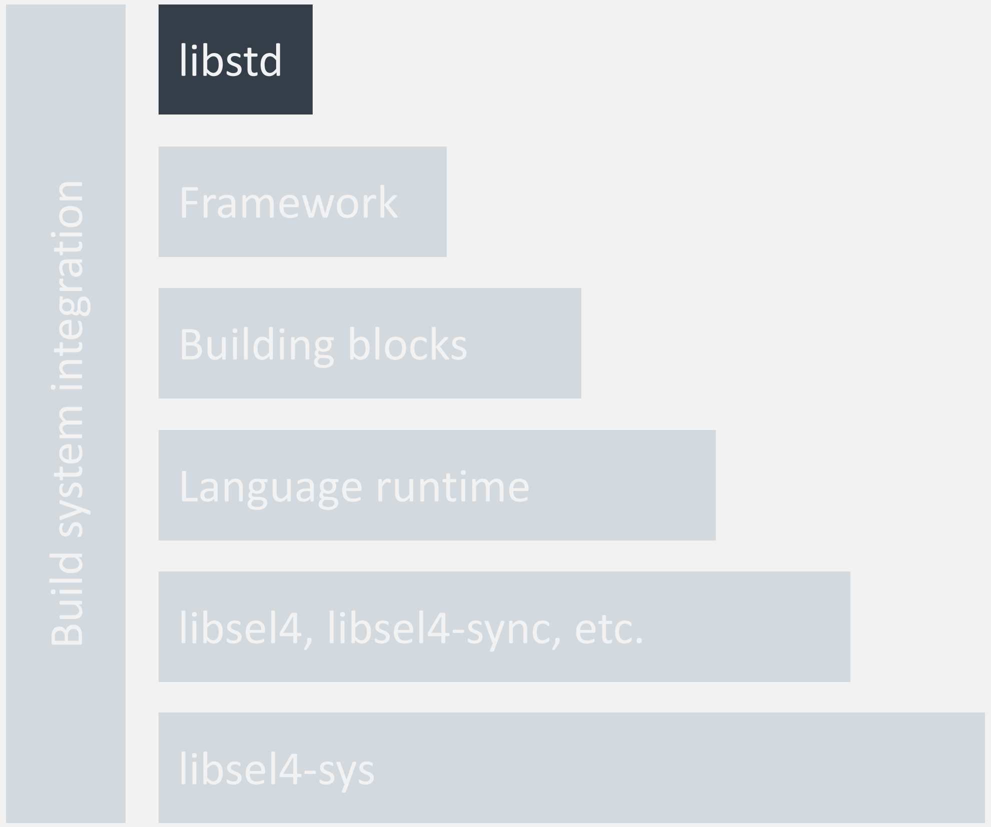
Colias Group

Levels of Rust support in seL4 userspace: a vision



Framework for composing a complete system

Levels of Rust support in seL4 userspace: a vision



libstd: System-dependent portion of the Rust standard library

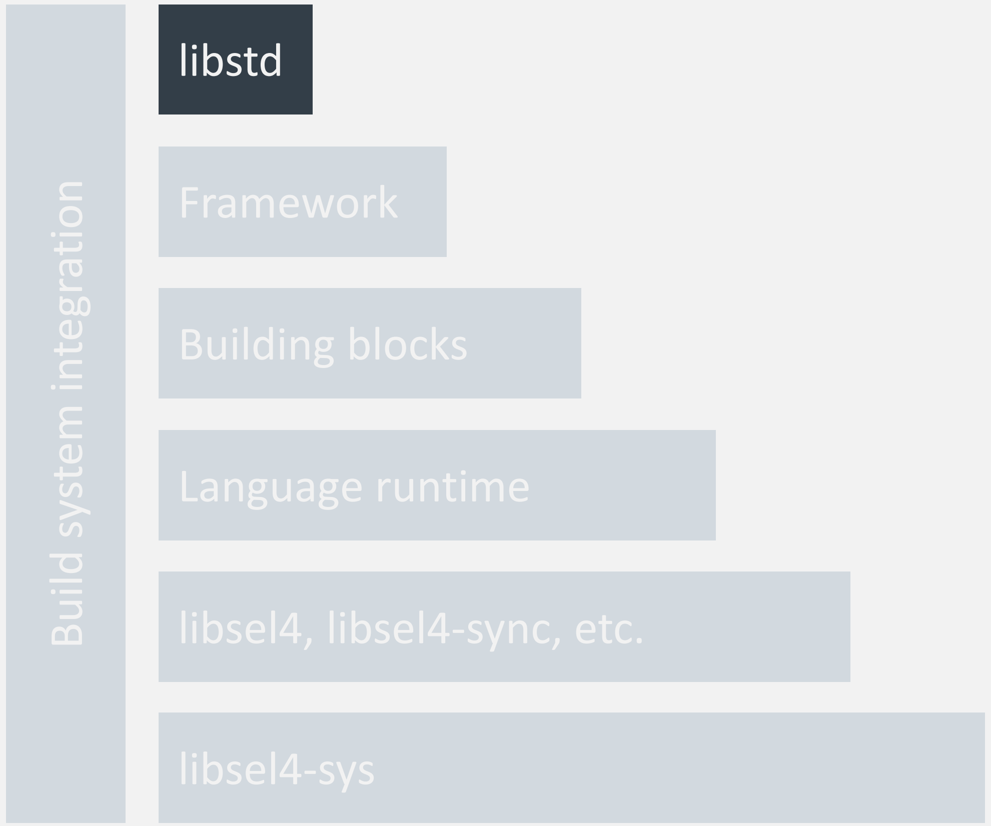
Enables use of crates which depend on libstd

- Example: Wasmtime in Veracruz[1]

[1]https://github.com/veracruz-project/veracruz

Colias Group

Levels of Rust support in seL4 userspace: a vision



libstd: System-dependent portion of the Rust standard library

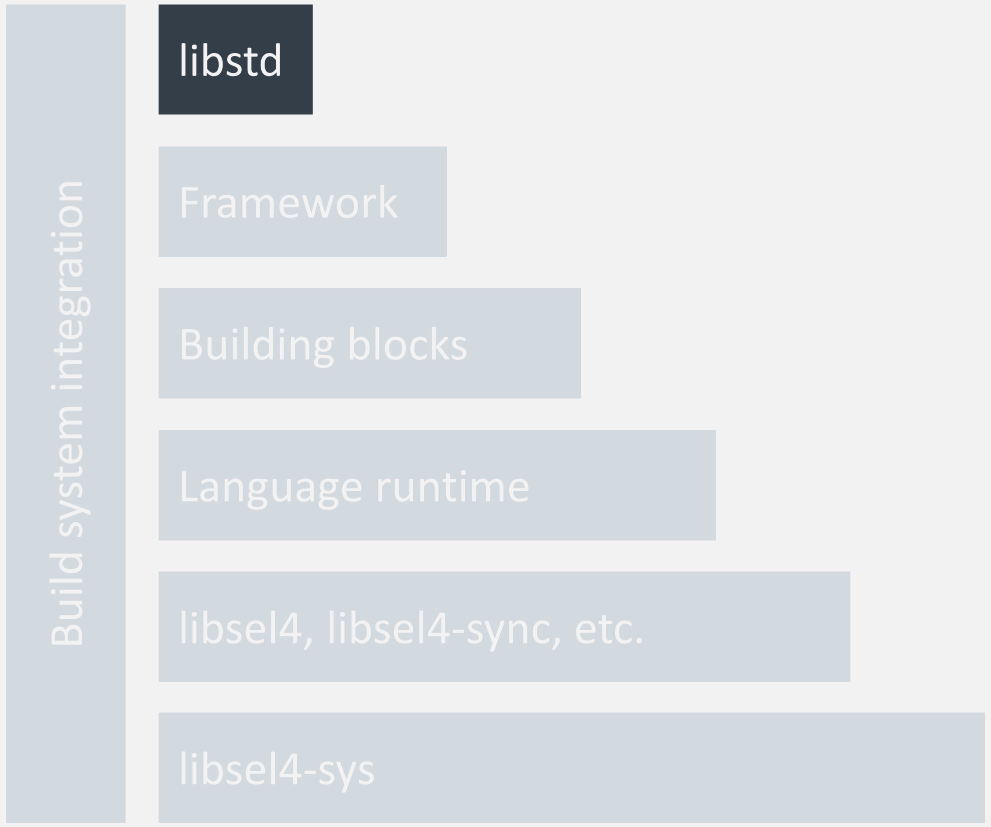
Enables use of crates which depend on libstd
- Example: Wasmtime in Veracruz[1]

Generic port with hooks, or per-framework ports?

[1]https://github.com/veracruz-project/veracruz

Colias Group

Levels of Rust support in seL4 userspace: a vision



libstd: System-dependent portion of the Rust standard library

Enables use of crates which depend on libstd

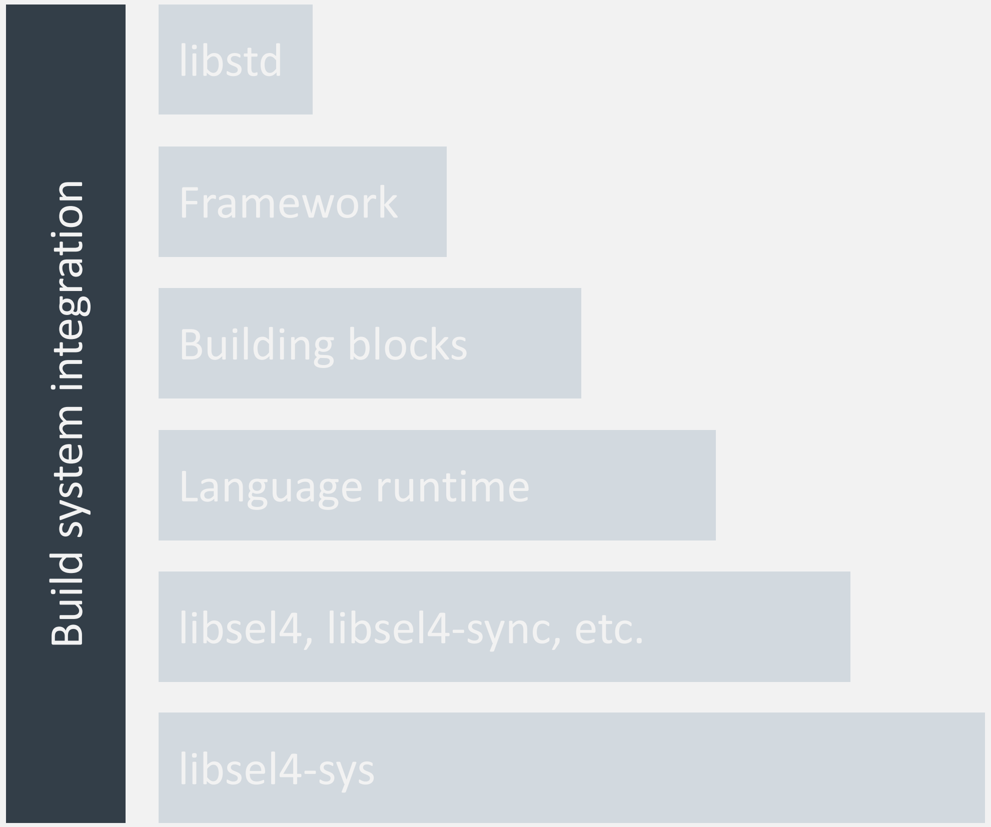
- Example: Wasmtime in Veracruz[1]

Upstream to Rust?

[1]https://github.com/veracruz-project/veracruz

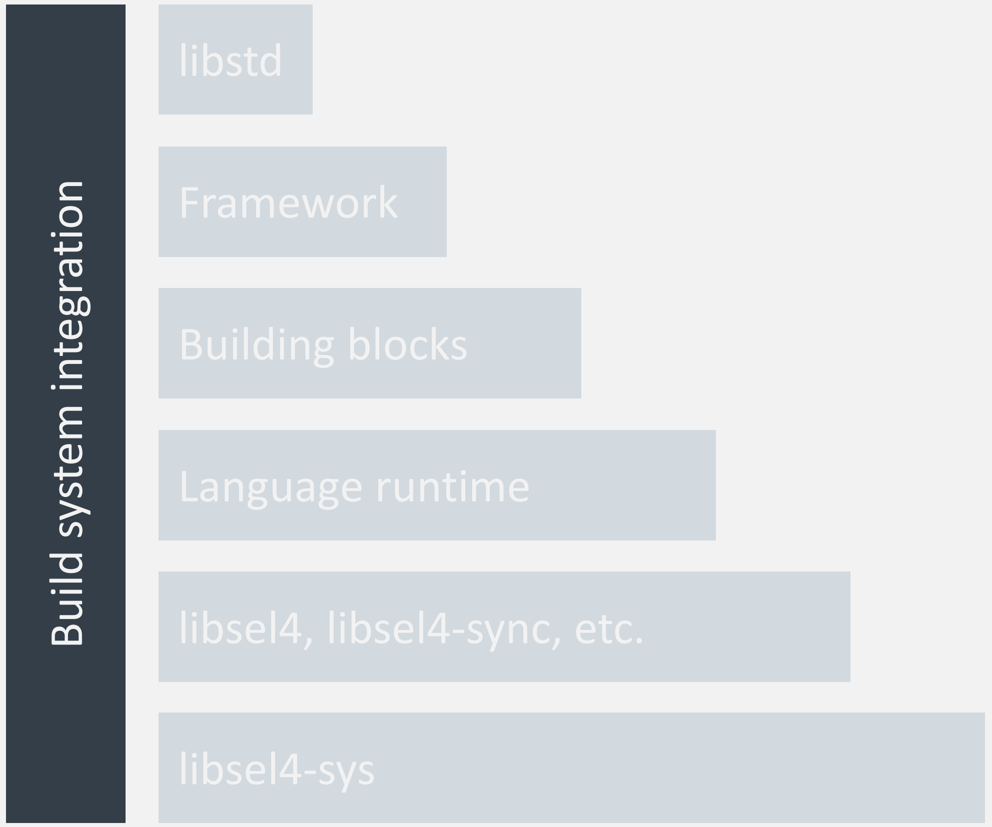
Colias Group

Levels of Rust support in seL4 userspace: a vision



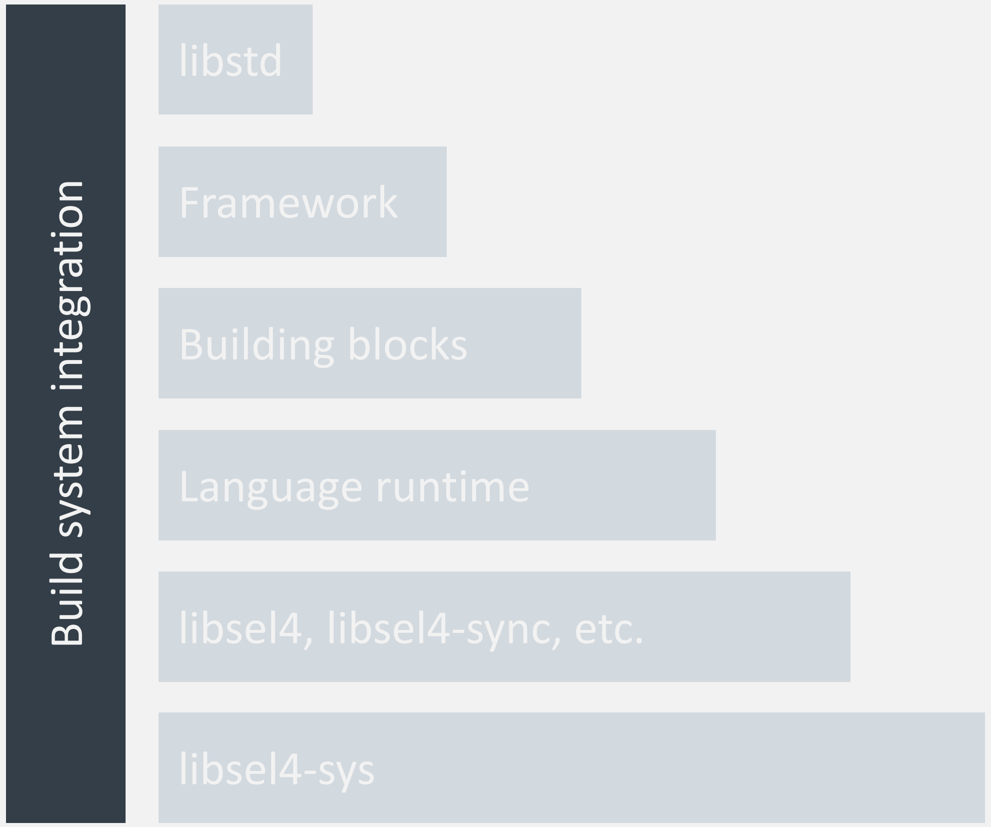
Should not be overlooked; impacts developer experience

Levels of Rust support in seL4 userspace: a vision



Should not be overlooked; is an element of rigorous engineering

Levels of Rust support in seL4 userspace: a vision



Should not be overlooked

Idea: support decoupling kernel and userspace build systems

Efforts throughout the ecosystem

- Robigalia
- feL4
- Ferros
- IceCap
- KataOS

Efforts throughout the ecosystem: Robigalia

https://rbg.systems/ (coordination point: #robigalia:http://matrix.org)

[To] create a highly reliable persistent capability OS, continuing the heritage of EROS and Coyotos[1]

Dynamism

Pure Rust libsel4-sys

[1]https://rbg.systems/

Efforts throughout the ecosystem: feL4

https://github.com/maindotrs/cargo-fel4

Encapsulates the development flow of a seL4-based system with a root task written in Rust

feL4 [aims] to automate and simplify the development process [of Rust in seL4 userspace]

Efforts throughout the ecosystem: FerrOS

https://github.com/auxoncorp/ferros

Later today: “FerrOS: Rust-y unikernels on seL4 w/ compile-time assurances”

Provides smart type-safe wrappers around seL4 features with an emphasis on compile-time resource tracking[1]

[1]https://github.com/auxoncorp/ferros

Colias Group

Efforts throughout the ecosystem: IceCap

https://gitlab.com/icecap-project/icecap

Arm Research project exploring virtualization-based confidential computing

Hypervisor decoupled from generic framework

Status: in a transitional period

My testing ground for the ideas discussed in this presentation:
https://gitlab.com/coliasgroup/icecap/icecap

Colias
Group

Efforts throughout the ecosystem: KataOS

https://github.com/AmbiML/sparrow-kata (part of Google's AmbiML project)

First open-source release in August 2022, with more to come

Builds on Robigalia's libsel4-sys, including upstreaming `syscall_stub_gen_rs.py`

Designed with resource-constrained systems in mind

Leverages CAmkES with a CapDL loader written in Rust

Towards convergence upstream

In order to pool ideas and resources

Join the discussion at https://sel4.discourse.group/

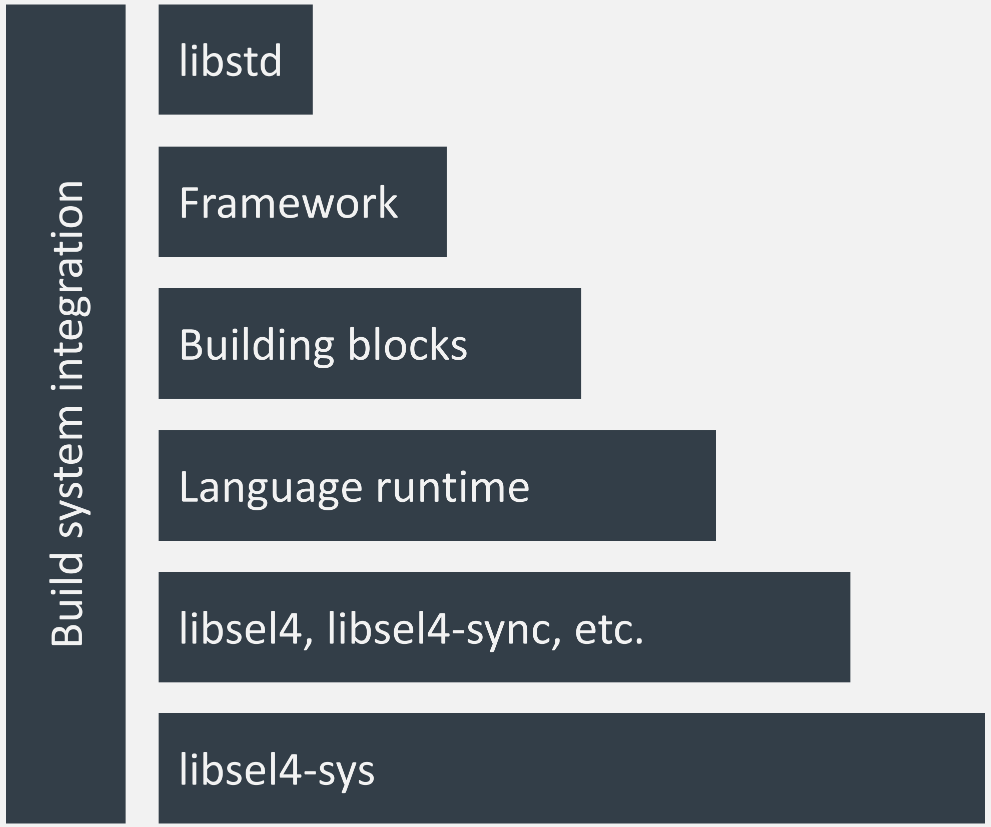
At which levels are there general enough solutions?

Colias Group

Discussion

Colias Group

Summary



Efforts throughout the ecosystem:

- Robigalia
- feL4
- FerrOS
- IceCap
- KataOS

Towards convergence upstream