man bpf

```
The bpf() system call performs a range of operations related to extended
Berkeley Packet Filters.  Extended BPF (or eBPF) is similar to the
original ("classic") BPF (cBPF) used to filter network packets.

For both cBPF and eBPF programs,
the kernel statically analyzes the programs before loading them, in
order to ensure that they cannot harm the running system.

eBPF extends cBPF in multiple ways, including the ability to call a
fixed set of in-kernel helper functions
and access shared data structures such as eBPF maps.
```

eBPF

“Superpowers have finally come to Linux”

“eBPF does to Linux what JavaScript does to HTML”

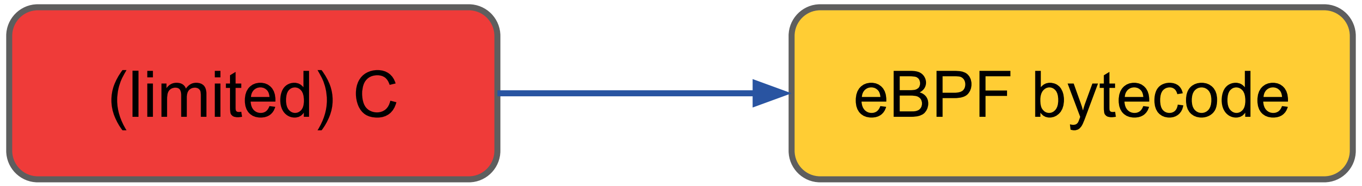
- Brendan Gregg, Netflix

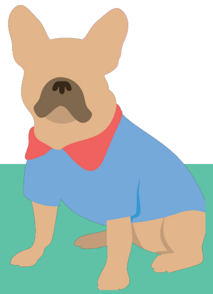
eBPF

Run code in the kernel
without having to write a kernel module

man bpf

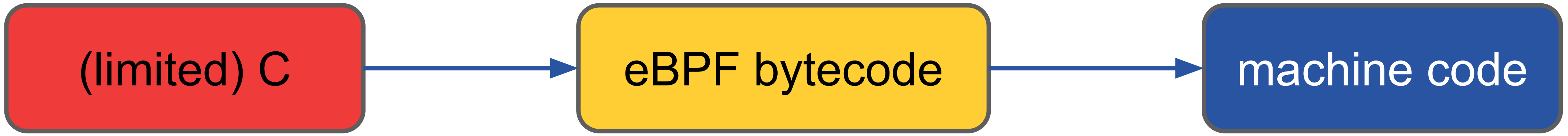
```
eBPF programs can be written in a restricted C that is compiled (using the
clang compiler) into eBPF bytecode.  Various features are omitted from this
restricted C, such as loops, global variables, variadic functions,
floating-point numbers, and passing structures as function arguments.
```





man bpf

The kernel contains a just-in-time (JIT) compiler that translates eBPF bytecode into native machine code for better performance.



Writing hello world

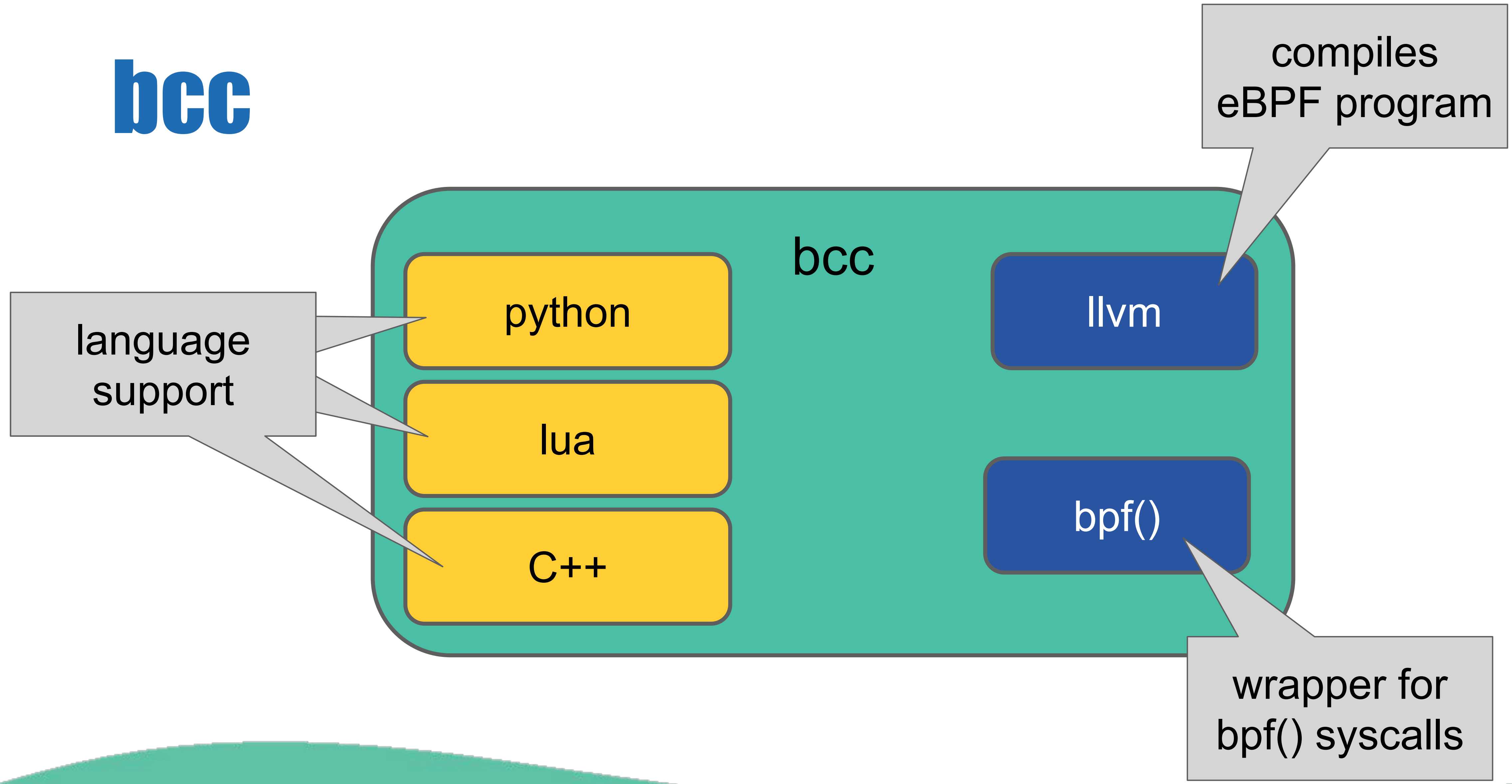
Avoiding tool chain hell

@lizrice

bcc

"BCC makes BPF programs easier to write, with kernel instrumentation in C (and includes a C wrapper around LLVM), and front-ends in Python and lua."

bcc


bcc
python
lua
C++
llvm
bpf()
language support
compiles eBPF program
wrapper for bpf() syscalls

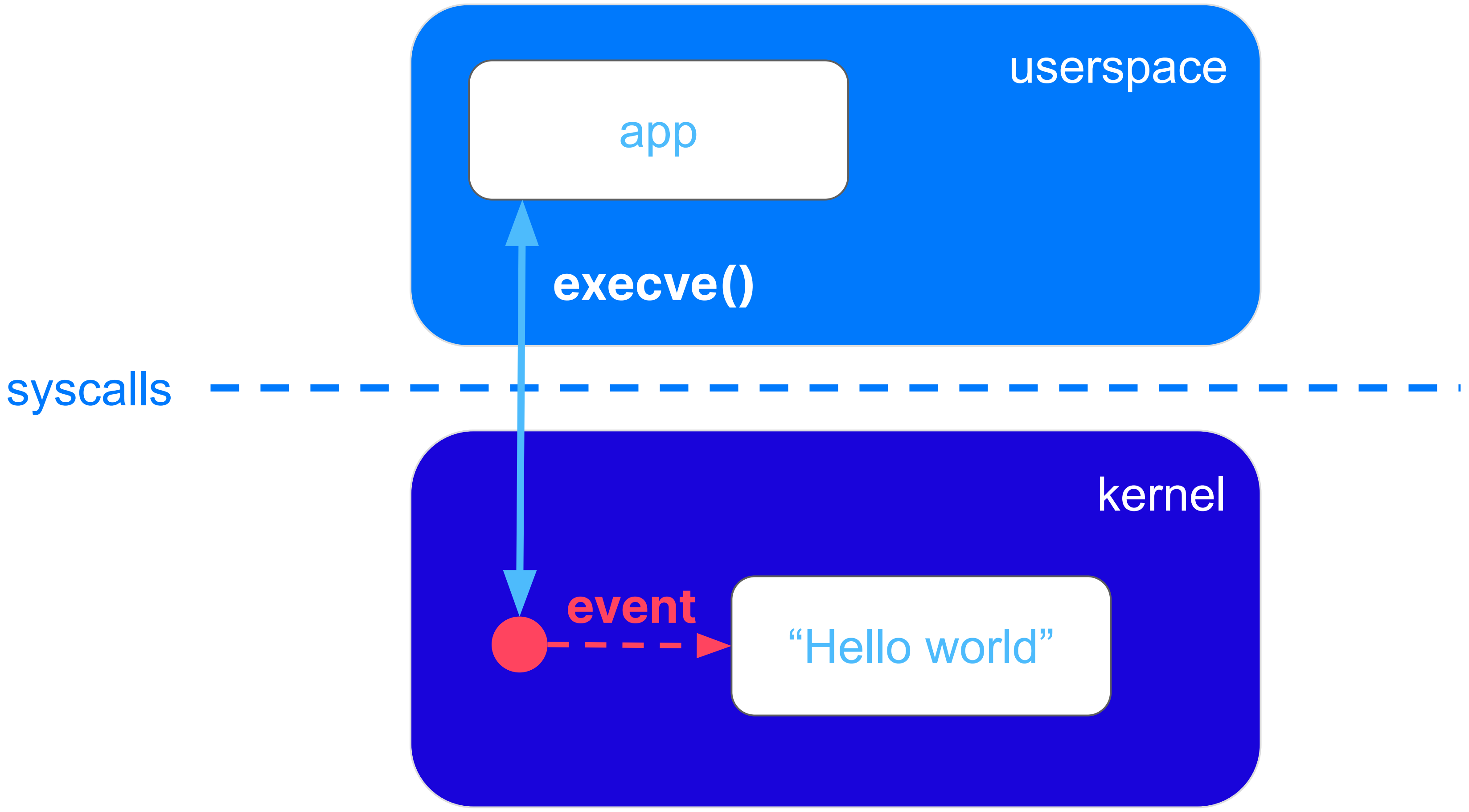
```
#!/usr/bin/python
from bcc import BPF

prog = """
int my_prog(void *ctx) {
   bpf_trace_printk("Hello world\\n");
   return 0;
}
"""

b = BPF(text=prog)
b.attach_kprobe(event="sys_clone", fn_name="my_prog")
b.trace_print()
```

Use strace to see the system calls

eBPF Hello World



Attaching BPF programs to events

@lizrice

Triggering eBPF programs

`eBPF programs can be attached to different events.`

- Kprobes
- Uprobes
- Tracepoints
- Network packets
- Perf events
- etc...

bpf_trace_printk()

Writing to

```
/sys/kernel/debug/tracing/trace_pipe
```

eBPF helper functions

These helpers are used by eBPF programs to interact with the system, or with the context in which they work. For instance, they can be used to print debugging messages, to get the time since the system was booted, to interact with eBPF maps, or to manipulate network packets.

```
bpf_trace_printk()
bpf_map_*_elem()
bpf_get_current_pid_tgid()
...
```