```
# 中期答辩.
~$ echo $title
基于参与者模型的异步框架及其 C++ 实现
~$ echo $author
卢星宇 / 软件工程
```

- 是什么?
- 怎么做? 有卵用?
- 做了啥? 能做完?

是什么？

- 参与者模型（*Actor*）
- 异步框架

Message-passing

Concurrent Constructs

- 收/发 件箱
- ID
- 只能 收/发 某 ID

解决兵兵问题

```erlang
% Erlang

ping(0) ->
    pong ! done;

ping(N) ->
    pong ! ball,
    receive
        ball -> ok
    end,
    ping(N - 1).
```

```
pong() ->
    receive
        done -> ok;
        ball ->
            ping ! ball,
            pong()
    end.
```

- 没有 *critical section*
- 没有 *deadlock*

- 无锁编程
- 为 RPC 提供天然的模型

怎么做？

三步走

- 用户级无阻塞 I/O：协程
- 消息传递：调度器
- OS 无阻塞 I/O：
  - ET Reactor

做了啥?

```
# C++ 协程 (coroutine) 库.
~$ echo $url
github.com/anqurvanillapy/acoro
```

读了 ZeroMQ `zactor`

```
# 凭个人理解重写与优化: hon (70%),
# (如信箱数据结构优化为无锁循环队列).
~$ echo $url
github.com/anqurvanillapy/hon
```

- Pony 源码/论文 (20%)
- 相关项目 (20%):
  - dasynq, napajs
- 设计参考 (20%):
  - Erlang/Elixir

 # 相关论文 (30%).

```
~$ cat good-papers
1. "Native Actors – A Scalable Software Platform for Distributed, Heterogeneous Environments"
2. "Revisiting Actor Programming in C++"
```

- 待参考方案:
  - CAF (C++)
  - Actix (Rust)

- 总进度：40~50%
  - 协程：√
  - 调度器：70%
  - ASIO：20%

```
~$ ^D
Thanks.
```