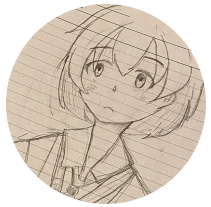
# ScarletというOSを作ってます

**鬼頭 太郎**
サイボウズ・ラボユース合宿LT

# 自己紹介

鬼頭 太郎 (きとう たろう)

- 立命館大学 M1 情報理工学研究科, M研究室
  - RISC-Vのハイパーバイザに関する研究
- 趣味
  - 情報系全般
    - 自作OS, しょうもないアプリ, ツール, etc...
    - 自宅鯖, ヤフオク漁り, おうちクラウド(k8s), etc...
  - 音楽
    - ベース, ギター(最近始めた), フルート, シンセ

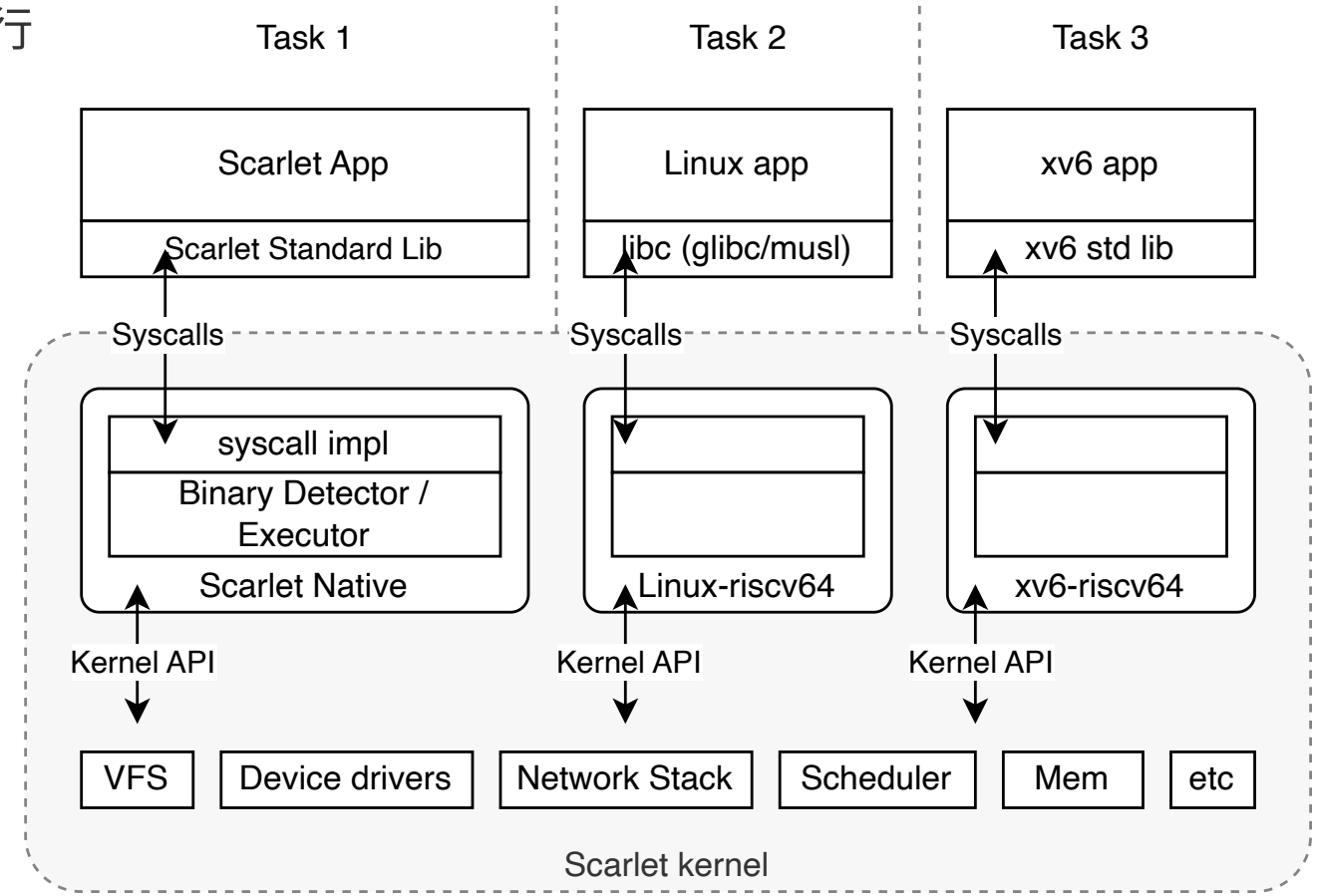


X: @petitstb
GitHub: petitstrawberry

# Scarlet[1]

A kernel in Rust designed to provide a universal, multi-ABI container runtime.と銘打ってます

- ABI Moduleを使った複数OS/ABIの透過的な実行
  - 色々なOS向けアプリケーションを**透過的に**実行可能
- 基本的な機能が実装済み
  - 仮想メモリ, スケジューラ,
  - ソフトウェアタイマ, IPC, tty
  - ファイルシステム (cpiofs, tmpfs)
  - グラフィック(framebufferのみ)
  - virtioデバイスのドライバ
    - virtio-blk, virtio-net, virtio-gpu
  - 今のところriscv64のみ対応




[1] Scarlet: https://github.com/petitstrawberry/Scarlet

# 色々な努力

OS/ABIごとに想定する環境はさまざま

- Filesystem
- 標準入出力
- 環境変数
- IPC
  - パイプ, シグナル的な仕組みなど
- etc...

これら

色々な努力があって...一部を紹介

# Kernel Object

カーネル空間とユーザ空間で共有されるリソースや, タスク・ABI module間で共有されうるリソースをKernel Objectとして抽象化 - それぞれのTaskがKernelObjectのHandle Tableを持つ - ユーザ空間/ABI moduleからはHandleを通じてアクセス (fdみたいなもの)

一部の例 (まあ、Capabiltyベースのやつ)

- FileObject
  - ファイルパスを持ったオブジェクト
  - 基本的にはStreamObjectなど他のKernelObjectを内包する
  - Filesystem実装がこれを生成・管理
- StreamObject
  - 順次読み出し, 書き込み可能なオブジェクト
  - read, writeなどのメソッドを持つ
- PipeObject
  - StreamObjectを実装している
  - WriterとReaderに分かれている

# コンテナ的な仕組み (fs)

VFSを各タスクで独立させて持たせることで, コンテナ的な動作や, OS/ABI独自のファイルシステムをマウントしたりやディレクトリ構造を実現可能

VFS間の連携も可能なので, タスク間(異種ABI moduleも含む)でルートは独立しながらHomeディレクトリを共有したりも可能

- Overlay, bind mountなどが実装ずみ

# Transparent Executor (1/2)

というのがあって... Scarletでバイナリ実行する上で最も重要な部分

決まりごと (execveのようなシステムコール実行時は, 常にTransparent Executorを通す)

**役割**

- バイナリの種別判定
  - バイナリの実行処理をどのABI moduleに任せるか決定
- ABI moduleの切り替え
- ABI module呼び出し
  - ABI moduleの初期化, 終了処理
  - FSの構成, 切り替え
  - スタックの構成
  - ファイルディスクリプタの変換
  - 標準入出力の変換
  - 環境変数の変換, etc...

# Transparent Executor (2/2)

FSの切り替えの話

まず, Scarletを起動した際(init時点)のディレクトリ構成

```
/ (root)
 ├──── data
 │     └──── config
 │           └──── linux-riscv64
 │                 └──── read-write data
 ├──── home (shared among all tasks)
 │     └──── user home directories
 └──── system
       ├──── linux-riscv64 (read-only) root of linux tasks
       │     └──── bin
       └──── scarlet (read-only) root of Scarlet native tasks
             └──── bin
```

うまい具合にOverlayFSで重ねると, linux-riscv64のバイナリが実行できる環境になる

# Transparent Executorの実装 (OverlayFSを利用したバイナリ実行環境の構築)

Transparent Executorが実行時に
OverlayFSを利用して, linux-riscv64のバイナリが実行できる環境を構築する

最終的にLinux Taskから見ると以下のようなディレクトリ構成になる

```
/ (linux task root) (Overlay: /system/linux-riscv64 + /data/config/linux-riscv64)
 ├──── home (shared among all tasks)
 ├──── bin
 │     └──── linux-riscv64 binaries
 └──── scarlet (Scarlet Native Global VFS)
       ├──── data
       └──── system
```

# 実際のところどれくらい動くのか

Linux-riscv64, xv6-riscv64, ScarletのABI moduleを実装中

# Linux ABI module (Linuxアプリがちょっと動く)

大体30個くらいのLinuxシステムコールが実装されてる
(今回のために実装したもの以外も含む)

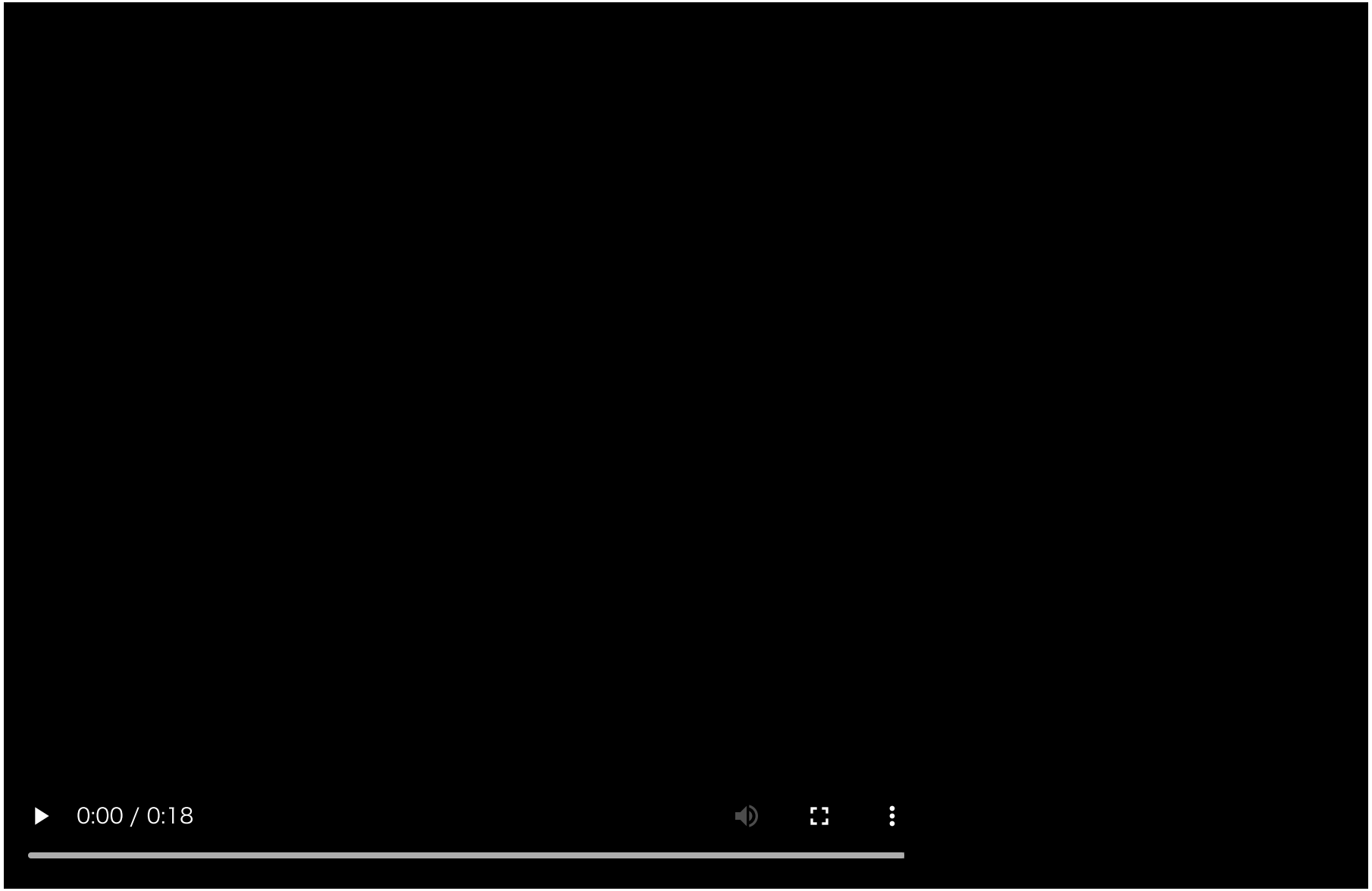
busyboxとかを動かしたり(次のページのバイナリを動かしたり...)

straceでトレースしたり, 実際にScarletで動かしながら必要そうなシステムコールを特定して実装していく

- fcntl, ioctl, mkdirat, openat, close, pipe2, getdents64, lseek, read, write, readv, writev, newfstatat, exit, exit_group, set_robust_list, nanosleep, clock_gettime, uname, getuid, brk, munmap, clone, mmap, mprotect
  - 必要最低限の機能だけを実装
- set_tid_address, rt_sigaction, rt_sigprocmask
  - Scarletは現状threadやsignalをサポートしていないので, 0を返すだけ

# Linux アプリがそのまま動く (DOOMとか)

ただし, 今のところstaticリンクのバイナリのみ (現在, 動的リンクも対応中)



# おわりに

複数OS/ABIの透過的な実行可能なScarletを紹介

複数OS/ABIにまたがるバイナリ実行の裏では, 大きな環境変更が行われるが,
ユーザは意識する必要がない

今後は, ネットワーク系の機能を実装したり, 動的リンク対応したり, キーボード入力をちゃんとしたりすれば, もっといろんなLinuxアプリが動くようになるかも

X11とかが動くところまで行けば, Linux ABI moduleに一区切りつけたい

- POSIXとかUNIX的でないものを動かしていきたい
- x86_64とかにも対応して, windows, mikanosとか, wasabiとかも動かしたい
- aarch64できれば...