A9N

high portability, stability, and scalability.

伊組烈火 / @horizon2k38

About me

- horizon : @horizon2k38
- 多趣味

低レイヤーオタク

- 自宅の本棚
  - 見れば分かる通り, 低レイヤーの比率が高い
  - 低レイヤーたのしいよ＾＾
  - ソフトウェアアーキテクチャも好き
  - ネットワークも(少し)やっている

#Linuxを超えたい

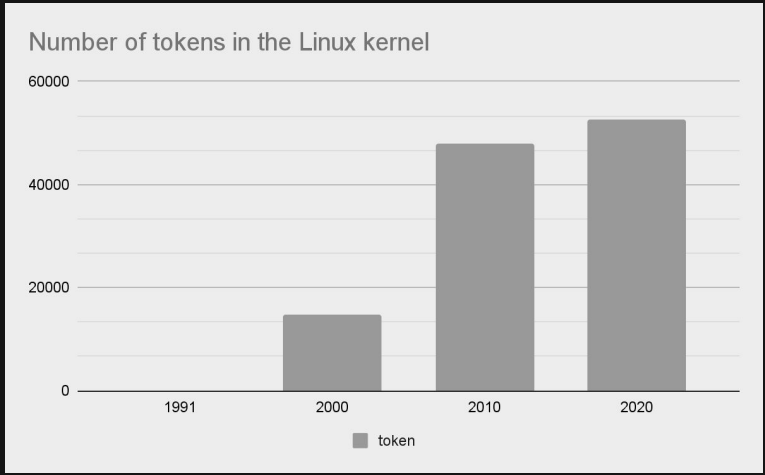
ただ, Linuxは今とても世界で使われている

- サーバー搭載OSシェアの**70%以上**がLinux
- スマートフォン搭載OSシェアの**70%以上**がAndroid
  - AndroidはLinuxカーネル

今すぐ置き換えるのは難しい……

- 大きな進歩は**小さな一歩**から
  - 一つだけLinuxに勝てるかもしれないものが作れるとしたらどこだろう?

Linuxでは学習がしづらい



実装を学ぶにはコードを読む必要が

あるが……

Linuxカーネルは2020年時点で**2000万行**

肥大と共にOSを学ぶハードルが高く
なってきている

cf. https://www.linuxfoundation.jp/wp-content/uploads/2020/09/JA2_2020-Linux-Kernel-History-Report.pdf
"Linux Foundation - 2020年度版 Linuxカーネルの歴史レポート" P. 6

教育用OS：MINIXによる学習

- **MINIX**は学習用OSとしてよく挙げられる
  - 長い間OSの教科書として使用されている
    “*Operating Systems: Design and Implementation*”
    という書籍とセット
    - Linuxはこれを読んだLinusにより作成された
  - カーネル自体は5000行程度
    - C とアセンブリで書かれている

MINIXのここがダメ

- 古い
  - 日本語の解説書籍が絶版
  - 例示されているデバイスドライバが古い
    - フロッピーディスク
  - (MINIX BOOKの時点では) 仮想記憶が存在しない
    - 現代において学ぶには不適切
- マクロの多用
  - EXTERN, PRIVATE, PUBLICが大量に存在する
    - 教育用としては癖のある書き方になってしまっている
    - そんなことをせずともC++でよいのでは
  - 多数のコンパイラに対応させた結果、分かりにくくなっている
- 変数名の省略が分かりにくい

Modern Operating System

Practical, Educational

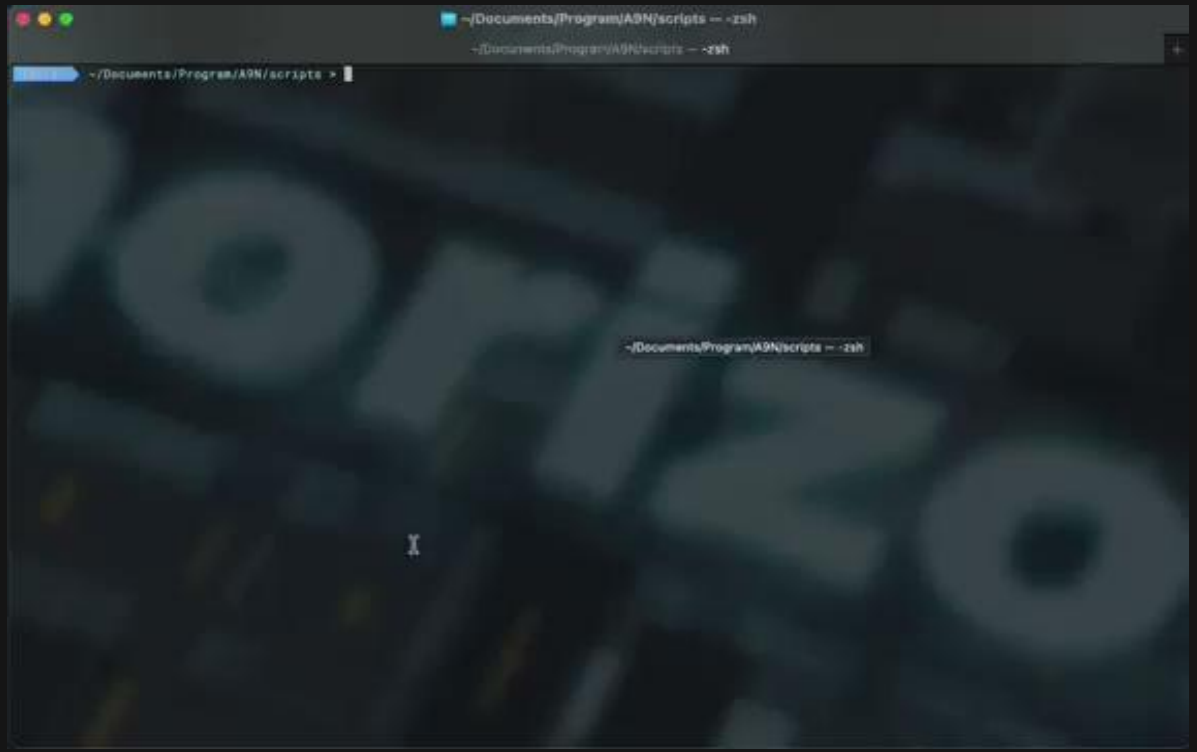
A9N

A9Nの目指す最初の一歩

今回の未踏ジュニアの内容

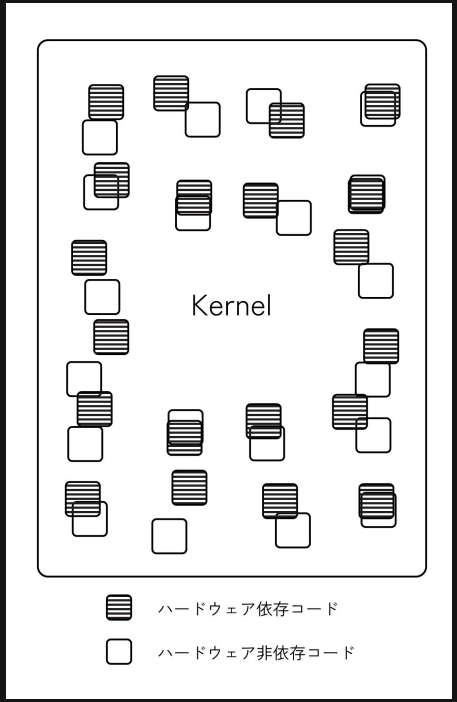
- **A9N (AbstractioN) カーネル**
  - HALを効果的に用いて移植容易性を実現した**マイクロカーネル**
  - **ASM/C/C++で書かれている**
  - ハードウェアとソフトウェアに**適切な境界を設定**し, 移植容易性と安定性を両立させる
- A9NDoc
  - A9Nの設計と実装を解説するドキュメント
  - OSの学習ハードルを下げ, より大きなものへとしていく
  - 可読性を重視する

デモ I: 録画済み


~/Documents/Program/A9N/scripts — -zsh
~/Documents/Program/A9N/scripts — -zsh
~/Documents/Program/A9N/scripts >
~/Documents/Program/A9N/scripts — -zsh

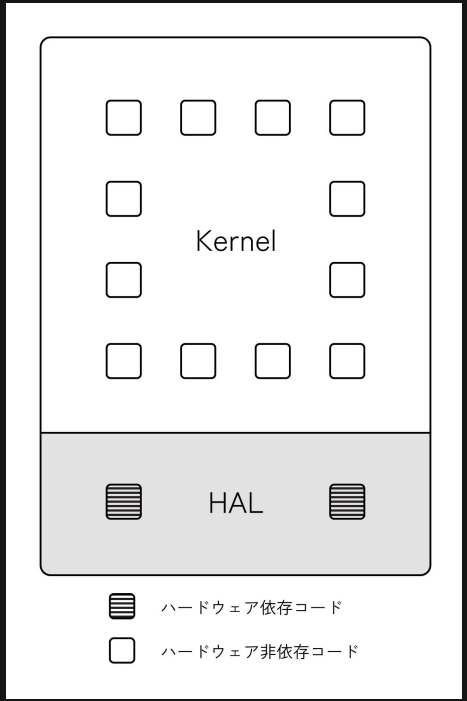
A9N Kernelを理解するための技術的知識

ハードウェアへの強い依存



- ハードウェア依存コードがカーネル全体に散らばったコードの場合
  - 保守性の低下
  - ハードウェアは腐りやすいので, ソフトウェアと齟齬が生じる
  - 他アーキテクチャへの移植が困難になる

HALのアイデア



- OSのハードウェア依存コードを分離する
  - ハードウェアを**抽象化**し, **統一されたインターフェース**を作成する
  - 移植時に書き換えるコード量が減る

A9NのHAL

- HALはインターフェースである
  - 実装すべき規約が明示されている
- カーネルの起動時にHALの実装が注入される
  - 各HALオブジェクトはAbstract Factoryで作成される
  - カーネル本体はハードウェアに依存しない
    - Testableなカーネルになる
    - 汎用OS上でもアプリケーションとして動作可能になる

HALの例：仮想記憶

- A9Nはページングによる仮想記憶機能を前提としている
- だが, ページテーブルの構造はアーキテクチャにより異なる
- HALは一方的に呼び出されるべきであり, HALからはカーネルの機能を呼び出したくない

-> どう抽象化するか？

A9Nにおける仮想記憶の抽象化

```
virtual bool is_table_exists
(
    kernel::physical_address top_page_table,
    kernel::virtual_address target_virtual_address
) = 0;

virtual void configure_page_table
(
    kernel::physical_address top_page_table_address,
    kernel::virtual_address target_virtual_address,
    kernel::physical_address page_table_address
) = 0;

virtual void map_virtual_memory
(
    kernel::physical_address top_page_table_address,
    kernel::virtual_address target_virtual_addresss,
    kernel::physical_address target_physical_address
) = 0;
```

基本的に, どのアーキテクチャでも

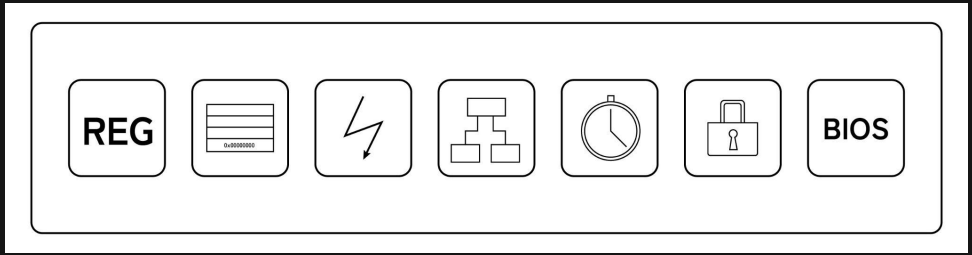
- プロセスごとに最上位ページテーブルを持つ
- PTEにはPresentビットが存在する

そのため,

1. ページテーブルが存在するかチェック
2. されていない場合, カーネルが物理メモリを割り当てる (メモリサーバーに移譲)
3. ページテーブルを設定する
4. ループ

これにより, カーネルはmap_virtual_memoryという機構をユーザーに提供できる

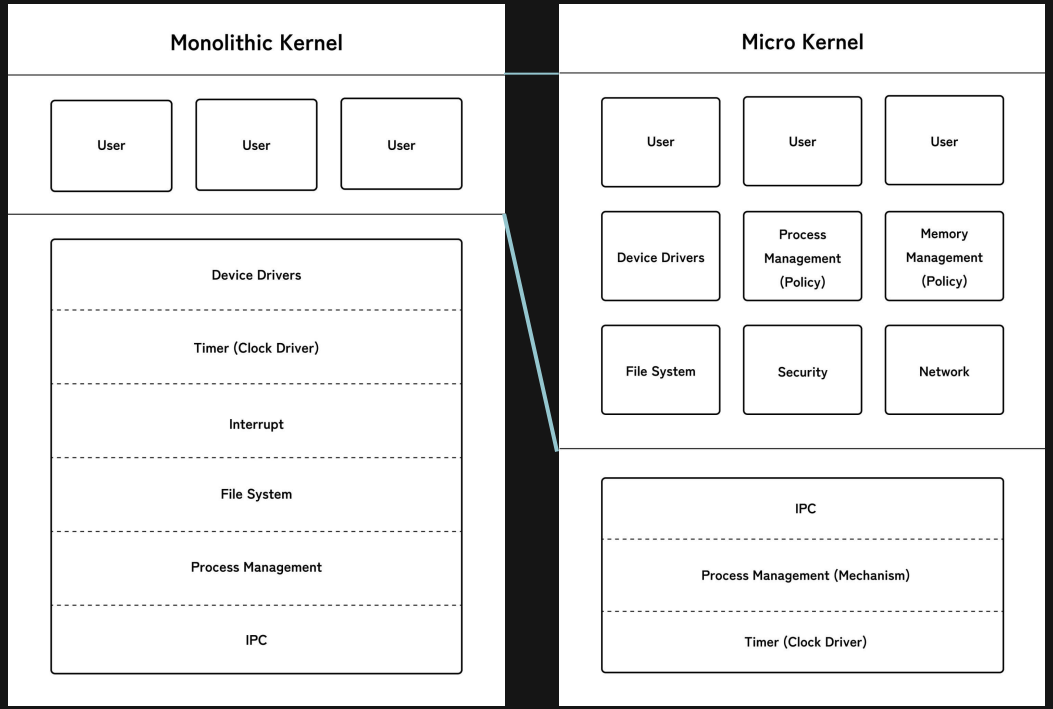
HALが抽象化する機能



- デバイスレジスタ
- デバイスアドレス
- **仮想記憶**
- 割り込み / システムコール
- DMA
- タイマ
- ロック
- ファームウェア

カーネルはこのレイヤーを通してハードウェアの機能にアクセスする

Monolithic / Microkernel

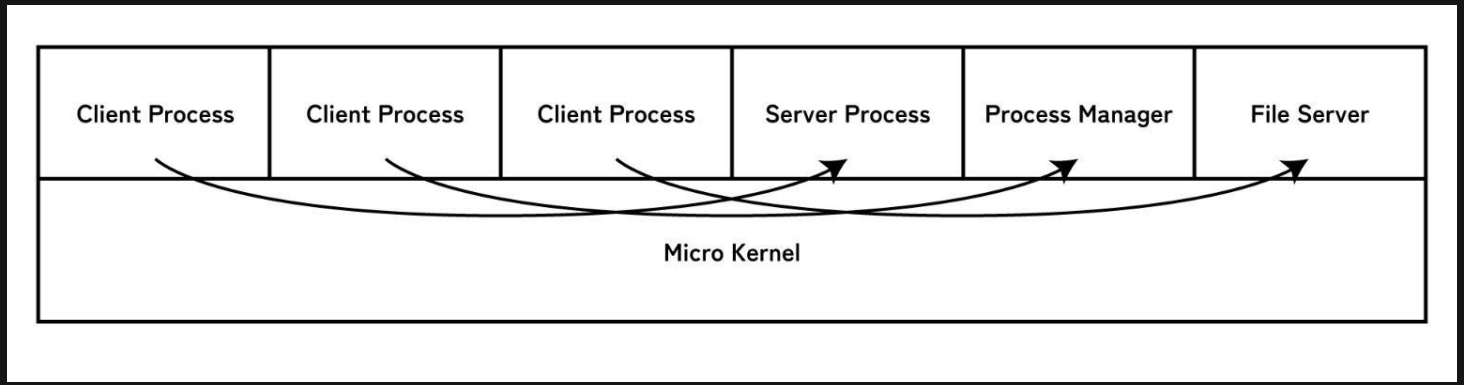


- カーネルは細分化し，小さくなる
- 不安定なドライバなどは分割する

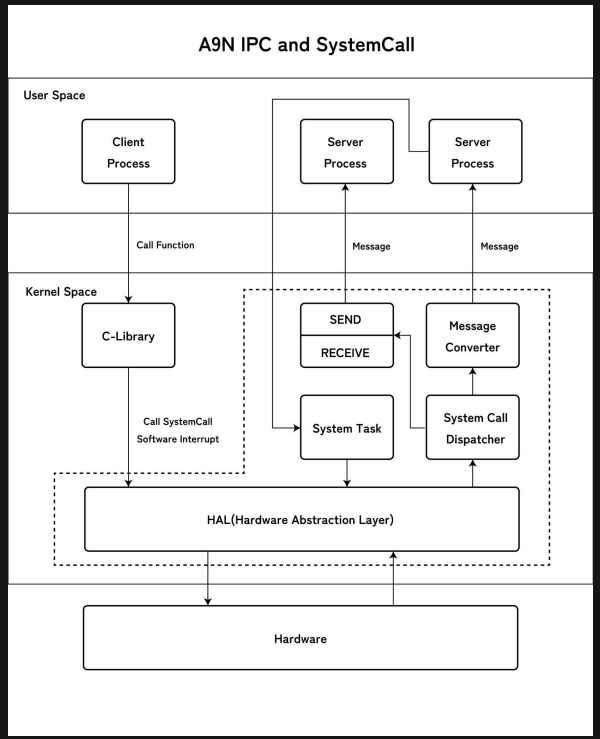
メッセージによるイベント駆動

クライアント・サーバモデル

- 機能を要求するクライアントは, サーバにメッセージを送信する
- カーネルはメッセージング基盤と最小限の機能のみ提供する
  - 機構はカーネルが提供し, 方針はユーザー空間上のプロセスが提供する



メッセージング基盤の仕組み



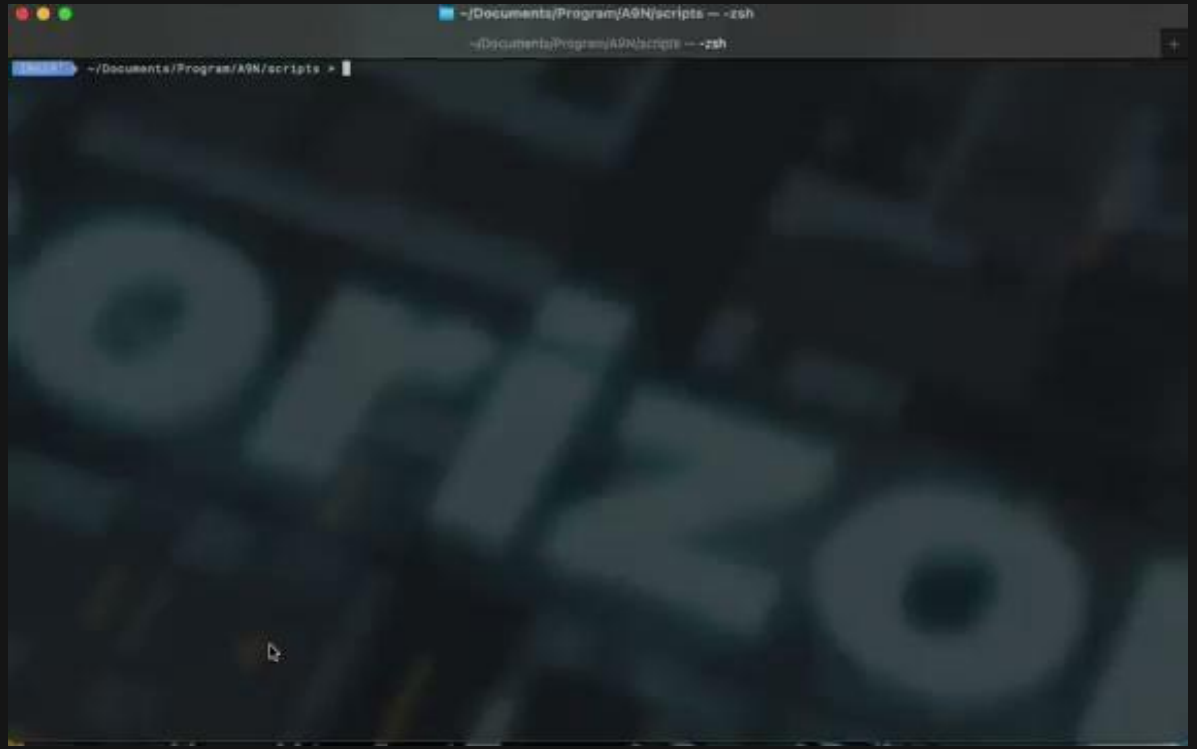
IPCはシステムコールとほぼ同じ仕組みで動作する

システムコールはメッセージに変換される

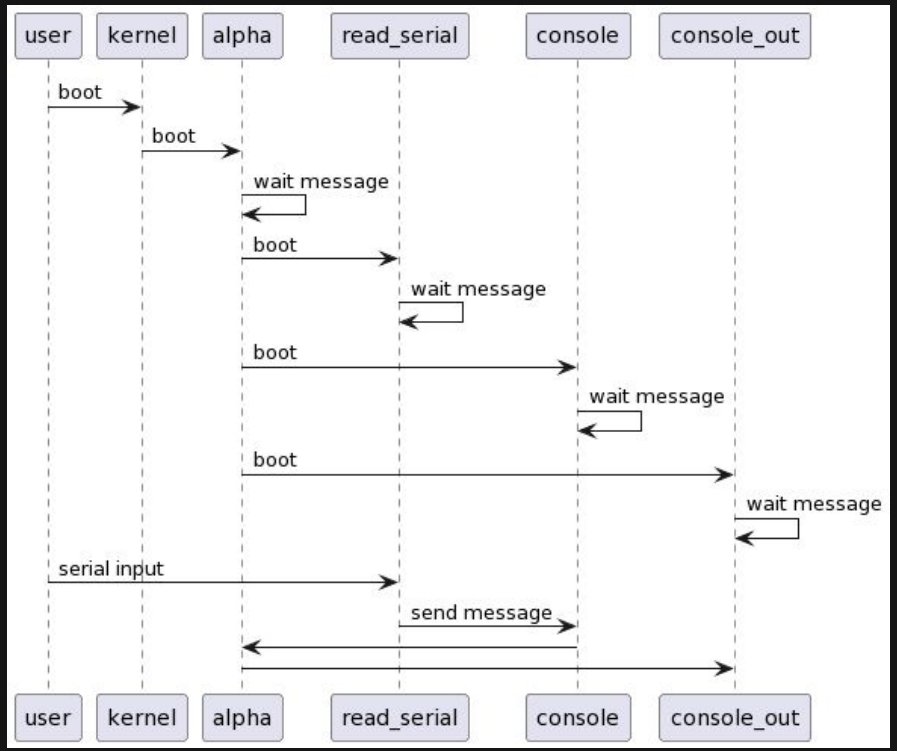
ユーザーモードで実行するには権限が足りない命令の場合, システムタスクへとメッセージが送信される

これらを踏まえてもう一度デモ

デモ II : 録画済み


~/Documents/Program/A9N/scripts — -zsh
~/Documents/Program/A9N/scripts — -zsh
~/Documents/Program/A9N/scripts >

デモで発生したIPCの流れ



- カーネルは最初にalphaサーバーを起動する
- alphaサーバーはread_serial, console, console_outを起動する
- read_serialはユーザ入力を受け取り, consoleへメッセージを送る
- consoleは受信メッセージをバッファに入れる
- Enterキーメッセージにより, バッファがalphaへ出力される
- alphaは物理メモリ情報を持っているため, console_outへ出力する

未踏ジュニア期間中(約半年)の開発

- A9NLoader
  - A9N向けブートローダー (UEFI + EDK2)の実装.
- A9N Kernel
  - 基本的な動作は実装できた
    - とはいえ, まだ全く満足していない
  - 2024/1/1にpublic repositoryにします
- A9N HAL
  - x86_64向けの実装を作成した

つらかったポイント

- どう抽象化するか
  - 具象を出来るだけ漏らず, 適切に分離する技法
- x86_64アーキテクチャ
  - 後方互換性を保つためのデザインがされているため, 開発が非常に苦しい
  - 割り込みハンドラを共通化する機能が備わっていない
    - NASMでマクロを使用して解決
  - 負の遺産をどう用いるか
    - CSだけか->結局GSとFSを使う羽目に
    - Long Modeに付き纏う
  - Higher-Halfへカーネルをリマップ
- 低レイヤー
  - デバッグが難しい

課題点

- Microkernelに徹せていない
  - より多くのカーネル機能をユーザランドへ委譲する
- 危険なC＋＋
  - Rustで書き直す
- 64bitが前提である
  - 32bitマイコンのようなものにも移植できるように変更する

展望：レベル1

- マルチプロセッサ
- Connect to the Internet
  - TCP/IP Serverの実装
- WASM
  - Wasm Runtimeの実装
- コンテナ
  - Linuxバイナリを動作させる
- GUI
  - Protocol-Based Window Systemの実装
- Rust化

展望：レベル2

- A9NDocを完成させる
  - 学習用にA9Nを使えるものにする
- 実用と教育を両立させる
  - MINIXを置き換える存在となる
- A9NDocで育てた人材をA9Nプロジェクトに集める
  - より大きなものへとしていく

A9Nと未来

- A9Nが普及したら
  - 人類はマイクロカーネルによりセキュリティと安定性を手に入れる
  - 拡張しやすいカーネルは新たな発見を生む

つまり、A9Nが普及すると、

技術革新が促進され, 人類は新たなステージに到達する

ご清聴ありがとうございました

参考文献 I

Andrew S. Tanenbaum (2007) “*モダン オペレーティング システム 原著第二版*”,

(水野忠則・太田剛・最所 圭三・福田晃・吉沢康文 訳), ピアソン・エデュケーション・ジャパン

Andrew S. Tanenbaum (2007) “*オペレーティングシステム 第三版*”,

(吉澤康文・木村信二・永見明久・峯博史 訳), ピアソン・エデュケーション・ジャパン

Robert C. Martin (2017) “*Clean Architecture*”, (角 征典・高木 正弘 訳), KADOKAWA

坂井 弘亮 (2015), “*ハロー"Hello,World" OSと標準ライブラリのシゴトとしくみ*”, 秀和システム

Jorrit N. Herder (2008), “*DEMYSTIFYING MICROKERNELS AND MINIX 3*”,http://www.minix3.org/docs/jorrit-herder/disi08-talk.pdf

Daniel P. Bovet・Marco Cesati (2007) “*詳解LINUXカーネル*”,

(杉田由美子・清水正明・高杉昌督・平松雅巳・安井隆宏 訳), O’Reilly Japan

Marshall Kirk McKusic・George V. Neville-Neil (2005) “*BSDカーネルの設計と実装*,

(砂原秀樹・歌代和正 訳), ASCII

怒田晟也 (2023) “*自作OSで学ぶマイクロカーネルの設計と実装*”, 秀和システム

坂井 弘亮 (2010), “*リンカ・ローダ実践開発テクニック*”, CQ出版