

Linuxカーネルクローンの 作り方

@RKX1209

自作OSもくもく会 LT 2016/9/11

自己紹介

@RKX1209

神戸大学工学部4年生 (インターンで関東に...)

OS, 仮想マシン(VMM), コンパイラ, セキュリティ

seccamp 14,15,16, 未踏16



Linuxカーネルクローン

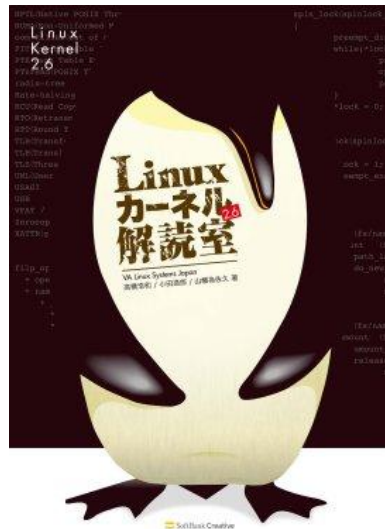
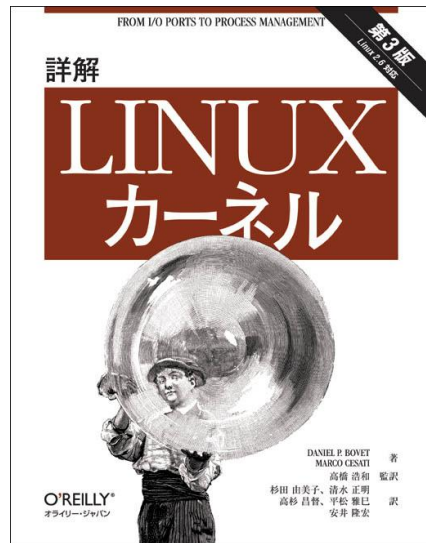
Linuxカーネル(2.6)を解読し、必要最低限の機能のみを自力で実装しなおす

あくまでカーネルのみの開発

シェルやウィンドウマネージャは未実装

「30日でできない 自作Linuxクローン開発」

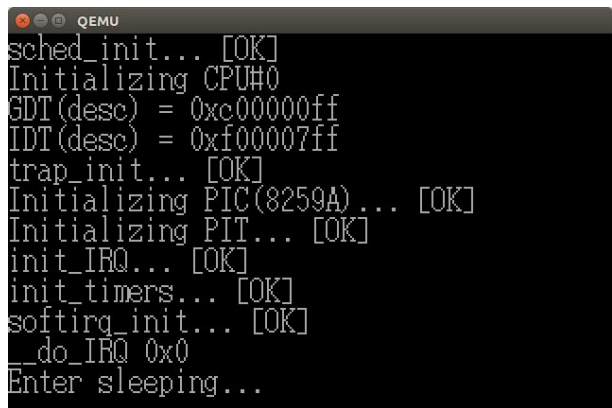
<http://rkx1209.hatenablog.com/entry/2014/12/25/225138>



Motivation

Linuxカーネルのクローンを開発する意義

- ❑ メジャーなOSのアーキテクチャに習うことでカーネル開発への理解を深める
- ❑ Linuxのコア部分のみ実装しているためLinux自体の解読にも役立つ

A screenshot of a QEMU terminal window with a black background and white text. The text shows the initial boot sequence of a Linux kernel, including scheduler initialization, CPU initialization, GDT and IDT setup, trap initialization, PIC and PIT initialization, and IRQ initialization. The process ends with the kernel entering a sleeping state.

```
QEMU
sched_init... [OK]
Initializing CPU#0
GDT(desc) = 0xc00000ff
IDT(desc) = 0xf00007ff
trap_init... [OK]
Initializing PIC(8259A)... [OK]
Initializing PIT... [OK]
init_IRQ... [OK]
init_timers... [OK]
softirq_init... [OK]
__do_IRQ 0x0
Enter sleeping...
```

<https://github.com/RKX1209/Abyon>

Linuxブートのおおまかな流れ

1. ブートローダー stage1 (MBR)
 - a. BIOSを使ってカーネル本体とstage2をメモリ上にロード
2. ブートローダー stage2
 - a. GDTの仮設定, カーネル本体へのジャンプ
3. startup_32 (arch/i386/kernel/head.S)
 - a. IDTの仮設定(割り込み割り込みは無視)
4. kernel_main (kernel/main.c)
 - a. 本格的な初期化の開始
5. rest_init(kernel/main.c)
 - a. カーネルスレッド起動

メモリマップ

<16-bit:real mode>

0x7c00 - 0x7e00 Boot Loader(loaderS1)

0x7000 - 0x???? Boot alloc bit map

0x8000 - 0x8200 Boot Loader[2](loaderS1) <= Copy

0x8200 - 0xA0000 Kernel Image(temporary)

<32-bit:protected mode>

0x00100000 - 0x???????? kernel body

0x???????? - 0x???????? init page table

~ init_pg_tables_end ~

0x00300000 - 0x???????? root disk(mounted on /)

4. kernel_main

- e820, early_bootメモリスシステムの初期化(**setup_arch**)
- プロセススケジューラの初期化(**sched_init**)
- トラップの初期化(**trap_init**)
- 割り込みの初期化(**init_IRQ**)
- タイマーの初期化(**init_timers**)
- ソフトウェア割り込みの初期化(**softirq_init**)
- スラブアロケータの初期化(**kmem_cache_init**)
- ファイルシステムの初期化(**vfs_caches_init**)

5. rest_init

- initカーネルスレッドの作成(kernel_thread)
- スケジュールの開始(schedule)
- idle状態に突入(cpu_idle)

カーネルは初期化が終わるとアイドル状態

後はイベント駆動(タイマー割り込み, ハードウェア割り込み)