

EN 帶你入門系列

{作業系統篇}

Ian / 陳毅

關於我

- 任職於禾薪科技擔任軟體工程師
- 開源專案參與
 - free5GC
 - ONF Aether project
 - 多場研討會講者
- 著有兩本資訊書籍
 - EN 帶你寫個作業系統
 - EN 帶你入門 5G 核心網路



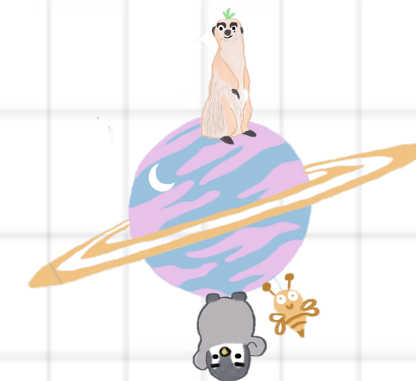
@ianchen0119

大綱

- 什麼是作業系統?
- 什麼是 RISC-V?
- 寫一個作業系統可以學到什麼?
- 會使用到的工具
- 基礎概念
- 延伸議題
- 問答時間



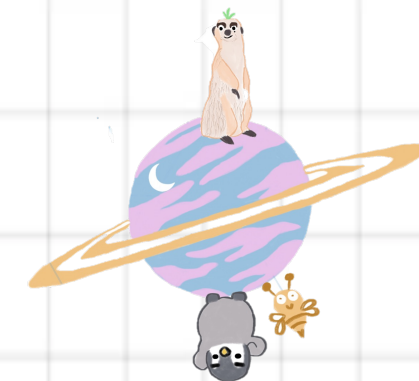
什麼是作業系統？



- 對我而言，作業系統是幫忙使用者控制硬體資源的軟體
 - 它為每個任務分配 CPU 資源與記憶體資源
 - 它幫你處理外接硬體傳送過來的訊號（如：鍵盤、網卡）
 - 它讓你能夠以指令或是操作圖像的方式，命令硬體完成工作



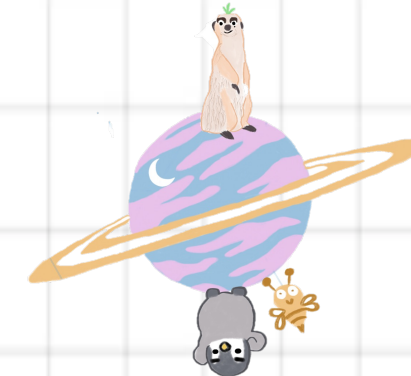
什麼是作業系統？



- 作業系統的「核心」做什麼？
 - 開機引導
 - 硬體設備管理
 - 檔案系統
 - 網路系統
 - 行程管理
 - 記憶體管理



什麼是 RISC-V?

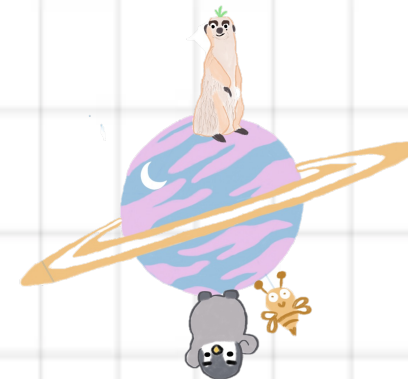


- ISA
- 開放硬體
- 由 UC Berkeley 發起
- Linux 與 AOSP 皆有 RISC-V 相關硬體的支援
- 多家公司採用，目前鎖定 AI 與車用市場



寫一個作業系統可以學到什麼？

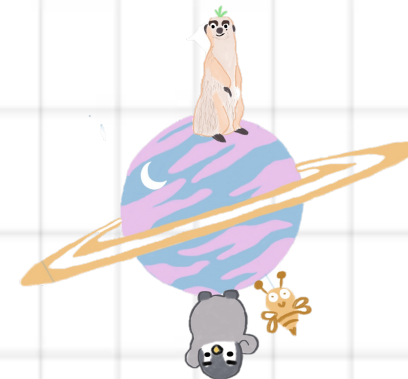
- 考試順利
- 有助於研究
- 對於資訊系統可以有更通透的理解
- 好找工作



@ianchen0119

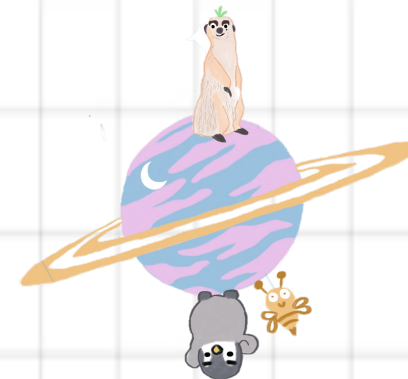
會使用到的工具

- QEMU
- Makefile
- Linker Script
- Assembly language
- C



基礎概念

- 系統引導
- 任務管理
- 中斷處理
- 記憶體管理



💡 補充：


本次直播介紹的設計以及使用的程式碼
皆來自於開源的 mini-riscv-os 專案



@ianchen0119

基礎概念：系統引導

- 作業系統要怎麼啟動？



```
.equ STACK_SIZE, 8192

.global _start

_start:
    # setup stacks per hart
    csrr t0, mhartid
    slli t0, t0, 10
    la    sp, stacks + STACK_SIZE

    add    sp, sp, t0
er

    # park harts with id != 0
    csrr a0, mhartid
    bnez a0, park

    j      os_main

park:
    wfi
    j park

stacks:
    .skip STACK_SIZE * 4
ks

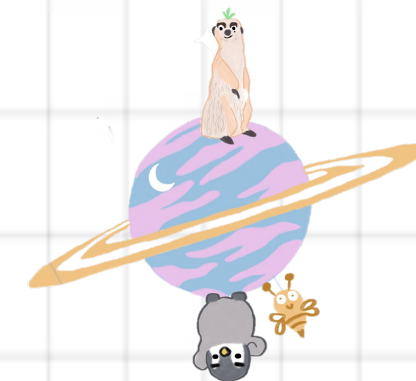
    # read current hart id
    # shift left the hart id by 1024
    # set the initial stack pointer
    # to the end of the stack space
    # move the current hart stack point
    # to its place in the stack space

    # read current hart id
    # if we're not on the hart 0
    # we park the hart

    # hart 0 jump to c
```



基礎概念：系統引導

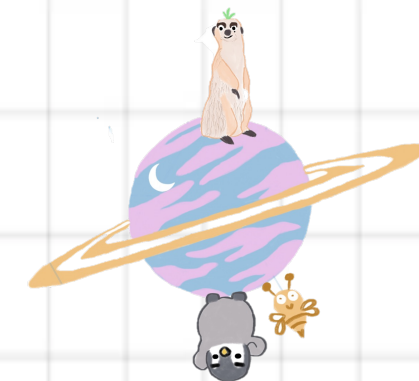


- 啟動後需要進行哪些初始化工作？

```
void os_start()
{
    uart_init();
    page_init();
    lib_puts("OS start!\n");
    user_init();
    trap_init();
    plic_init();
    virtio_disk_init();
    timer_init(); // start timer interrupt ...
}
```



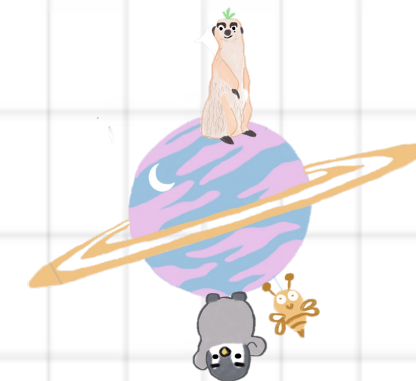
基礎概念：任務管理



- Process 的生命週期
- Process 的排程
 - Mini-riscv-os 採用最基本的 Round Robin 排程器
- Process 的資源管理
- 同步問題



基礎概念：中斷管理



- 什麼是中斷 (Interrupt) ?

- 中斷類型

- 軟體中斷
- 外部中斷
- 計時器中斷

- 流程簡介

- IRQ (Interrupt Request)
- CPU 暫停手邊任務
- CPU 呼叫對應的 ISR (Interrupt Service Routine) 處理 IRQ
- CPU 繼續處理原本的工作

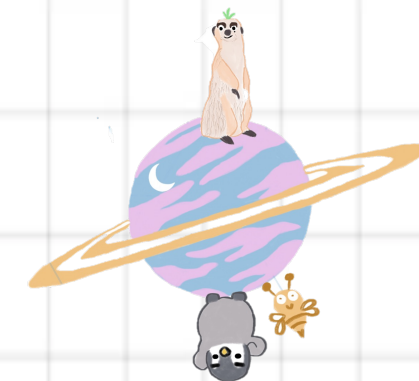
💡 延伸議題：

- 中斷太多是否會影響效能？如何改善？



@ianchen0119

基礎概念：中斷管理



- 作業系統幫我們做什麼？
 - 設定計時器中斷
 - 為每個 IRQ 定義 ISR
 - 當收到 IRQ，轉交給對應的 ISR
 - 過程中需考慮特權模式的切換



基礎概念：記憶體管理

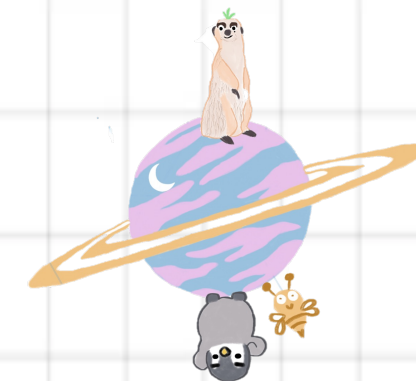
- Mini-riscv-os 允許 Process
 - 存取 stack 中的資料
 - 使用 heap 中的資料
- 如何做到呢？
 - 以 stack 為例：

```
// create a new task
int task_create(void (*task)(void))
{
    int i = taskTop++;
    if (i >= MAX_TASK)
    {
        return -1;
    }
    ctx_tasks[i].ra = (reg_t)task;
    // ctx_tasks[i].pc = (reg_t)task;
    ctx_tasks[i].sp = (reg_t)&task_stack[i][STACK_SIZE - 1];
    return i;
}

// switch to task[i]
void task_go(int i)
{
    ctx_now = &ctx_tasks[i];
    // switch_to(ctx_now);
    sys_switch(&ctx_os, &ctx_tasks[i]);
}
```



基礎概念：記憶體管理



- Mini-riscv-os 允許 Process
 - 存取 stack 中的資料
 - 使用 heap 中的資料
- 如何做到呢？以 heap 為例
 - 使用 linker script 提前預留一大段記憶體空間。
 - 使用組合語言 export heap 的起始與終點地址。
 - 作業系統開機時，將這塊記憶體以 PAGE_SIZE 計算出共有幾個 block 可被分配。
 - 同時使用額外的記憶體紀錄 Block 的分配狀況。



延伸議題

- 中斷的可擴展性 (scalability)
- 排程器的改良
- 考慮特權模式
- 改良記憶體管理機制
- 實作檔案系統



@ianchen0119



延伸議題 – 中斷的可擴展性

- 預設情況下，CPU 在處理 IRQ 時仍有可能再次收到 IRQ，較高優先權的 IRQ 會迫使原本的 ISR 讓出 CPU 資源。
- 這種一次處理多個中斷的情況，我們可以將它稱為 Nested Interrupt。
- 解法：
 - 處理 IRQ 時關閉 interrupt。
 - 縮短 IRQ 的處理時間。





延伸議題 – 中斷的可擴展性

- 解法[2]也正是 Linux Kernel 給出的答案
 - 當中斷發生，CPU 會先對它做「必要的」處理。
 - 以網卡為例，首要任務就是先將網卡 buffer 中的封包複製到作業系統中。
 - 以上被稱為 hardirq。
 - 接著依情況直接接續處理 softirq，或是喚醒 softirqd。



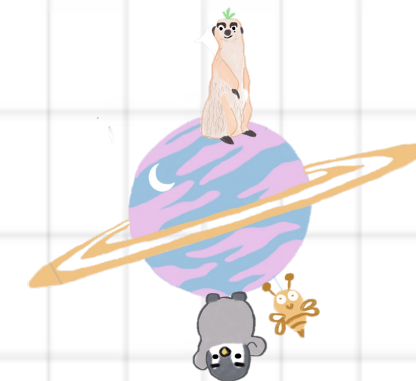
延伸議題 – 排程器的改良



- 可以實作更為複雜的排程器，甚至是讓 OS 支援多核心的處理。
- 可以實作 Mutex 以及 EDF 或 RM 這類常出現在 RTOS 的排程演算法，實作出一個小規模的 RTOS。



延伸議題 – 特權模式



- RISC-V 定義了三種模式給予系統基本的安全性：
 - Machine Mode
 - Supervisor Mode
 - User Mode
- Bare Metal: Machine Mode only
- RTOS: Machine + User Mode
- General: Machine + Supervisor + User Mode



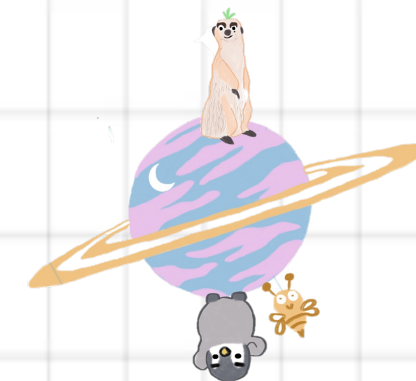


延伸議題 – 記憶體管理

- Mini-riscv-os 會在系統啟動時預留一個很大的空間，作為每個 Process 的 stack 空間。
- 每一個 Process 都直接存取到 physical address，而非 Virtual Address。
 - 實務上這樣非常危險，惡意的 Process 能夠接觸到不屬於自己的記憶體空間。
- 可參考 RISC-V 特權指定規格書，實作 page table。
 - Sv32, Sv39 以及 Sv48



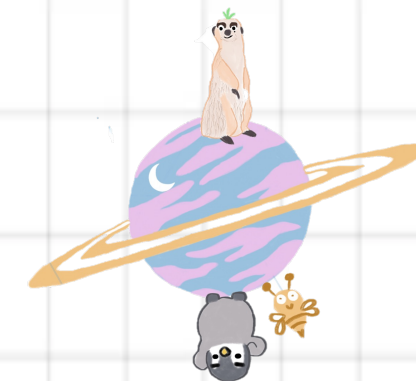
延伸議題 – 實作檔案系統



- Mini-riscv-os 已經實作了 Virtio 相關的驅動程式，能夠讀取虛擬磁碟的資料。
- 我們可以以此基礎實作出檔案系統，這部分可以參考 xv6 作業系統並加以改良。



問與答





EN 帶你寫個作業系統

RISC-V 開發輕旅行



「計算機結構X作業系統實務X開...」具...
一本全方位的作業系統開發入門指南

書籍特色

- 1. 第一本繁體中文的 RISC-V 相關書籍
- 2. 探討數個開放原始碼專案的設計細節！
- 3. 實務與理論兼具的技術書籍沒有碰過作業系統沒...



Ian Chen [立即驗證](#)

Book(s) Author / Software Developer / free5GC

台灣 Taiwan Hsinchu City · [聯絡資料](#)

Medium [↗](#)

752 人關注 · 500+ 位聯絡人

[願意接收](#) [新增區塊](#) [更多內容](#)

 free5GC

 國立交通大學



@ianchen0119