計算機組織與 RISC-V CPU 設計 · 第 8 章

作業系統支援

OS 概觀 / 排程 / 記憶體管理 / 分頁與虛擬記憶體 / TLB

摘要

作業系統是「使用者/應用程式」與「硬體」之間的介面, 而許多硬體機制正是為了支援作業系統而設計的。本章從計算機結構的視角檢視 OS: 批次系統遺留的中斷/計時器/特權模式、多道程式設計與排程, 以及本章的重點——記憶體管理: 交換、分割、分頁、虛擬記憶體、**TLB** 與分段。這些概念與 RISC-V 特權結構 (第 21 章) 直接對應。本章對應 Stallings 第 8 章。

目錄

1 作業系統概觀

1.1 OS 的兩種角色

- 介面 (便利性):OS 把硬體包裝成服務: 程式建立與執行、I/O 存取、檔案系統、錯誤偵測與回報、記帳。應用程式透過系統呼叫使用這些服務 (RISC-V:`ecall` 指令)。
- 資源管理者 (效率):OS 管理處理器時間、記憶體、I/O 裝置的分配。特別之處在於它放棄控制權後, 得依賴硬體機制 (計時器中斷) 才能奪回。

1.2 從批次到多道程式設計

早期一次一個作業 (serial processing) 浪費昂貴的機器時間; 批次系統以監督程式 **(monitor)** 自動接續作業, 但 I/O 等待仍使處理器閒置。多道程式設計 **(multiprogramming)**: 記憶體中同時保有多個作業, 一個作業等待 I/O 時切換到另一個, 大幅提高使用率。分時 **(time sharing)** 把同一思想用於互動使用者: 計時器中斷驅動時間片輪轉。

支援 OS 的硬體機制

多道程式設計依賴四項硬體特性:(1) 中斷 (I/O 完成通知、計時器);(2) 特權模式 (使用者/監督模式的指令與資源隔離,RISC-V:U/S/M 三級模式);(3) 記憶體保護 (阻止作業互相干擾與碰觸 OS);(4) 計時器。這些都是「結構為 OS 服務」的例子。

2 排程

- 長程排程: 決定哪些作業進入系統 (進入「就緒」佇列); 控制多道程度。
- 中程排程: 交換 (swapping)——把整個行程搬出/搬入記憶體。
- 短程排程 **(dispatcher)**: 從就緒佇列挑選下一個獲得 CPU 的行程。

行程狀態的五態模型: 新建 → 就緒 ↔ 執行 → 結束, 執行中因等待 I/O 而入等待 **(blocked)**,I/O 完成回到就緒。OS 為每個行程維護行程控制區塊 **(PCB)**: 識別碼、處理器狀態 (暫存器、PC)、記憶體指標、I/O 狀態、記帳資訊——脈絡切換就是保存/載入 PCB 中的處理器狀態。

3 記憶體管理

3.1 交換與分割

交換: 記憶體不夠時把等待中的行程搬到磁碟。固定分割簡單但有內部碎裂; 可變分割按需分配, 但反覆進出後產生外部碎裂, 需要壓實 (compaction)。分割時代的定址問題以基底暫存器 + 界限暫存器解決: 邏輯位址加上基底成為實體位址, 超過界限則陷入例外——這也是最原始的記憶體保護。

3.2 分頁 (Paging)

把記憶體分成固定大小的頁框 **(frame)**, 程式分成同樣大小的頁 **(page)**(典型 4 KiB)。頁可放進任意頁框, 由每行程一張頁表 **(page table)** 記錄映射:

$$\text{虛擬位址} = (\text{頁號 } p,\ \text{頁內偏移 } d) \xrightarrow{\text{頁表}[p]} \text{實體位址} = (\text{頁框號 } f,\ d)$$

沒有外部碎裂; 內部碎裂平均僅半頁。

3.3 虛擬記憶體與需求分頁

需求分頁 **(demand paging)**: 頁只在被參考時才載入。行程的工作集 (working set) 通常遠小於其總大小 (區域性原理再次登場!), 因此: 記憶體能容納更多行程; 行程可以大於實體記憶體——程式設計師面對的是巨大的虛擬位址空間。

參考不在記憶體的頁時觸發頁錯失 **(page fault)**:OS 掛起行程、發動磁碟 I/O 載入該頁、完成後恢復。頁替換演算法 (LRU 近似、clock) 決定犧牲頁; 過度換頁導致顛簸 **(thrashing)**。

3.4 頁表結構與 TLB

頁表本身可能很大 (32 位元空間/4 KiB 頁 = 一百萬項), 因此使用多層頁表 (頁表自身也分頁) 或反向頁表 (以頁框為項, 雜湊查找)。

每次記憶體參考都要查頁表, 等於加倍記憶體存取。解法是 **TLB**(Translation Lookaside Buffer): 頁表項的專用快取 (全關聯或高關聯度、數十至數千項)。TLB 命中則直接得到頁框號; 未命中才走頁表 (硬體行走或軟體處理)。

RISC-V 的對應

RISC-V 特權結構定義 **Sv32**(RV32: 二層頁表、4 KiB 頁、32 位元虛擬位址) 與 Sv39/Sv48(RV64)。`satp` CSR 指向根頁表; 頁表項含 V(有效)、R/W/X(讀寫執行權限)、U(使用者可存取)、A/D(參考/髒) 位元。頁錯失以例外 (cause 12/13/15) 進入 S 模式處理常式——與本章的通用描述一一對應 (詳見第 21 章)。

3.5 分段

分段 **(segmentation)** 依程式的邏輯結構 (程式碼、資料、堆疊) 劃分, 大小可變、程式設計師可見; 利於保護與共享。可與分頁結合 (段內再分頁)。現代主流 (含 RISC-V) 實際上採純分頁, 以頁層級權限位元達成保護。

4 本章重點回顧

本章要點

- OS 既是介面也是資源管理者; 中斷、特權模式、記憶體保護、計時器是它依賴的四大硬體機制。
- 多道程式設計以「I/O 等待時切換行程」提高處理器使用率; 脈絡切換 = 保存/載入 PCB。
- 分頁消除外部碎裂; 虛擬記憶體讓行程大於實體記憶體, 倚仗區域性原理。
- TLB 是頁表的快取, 使位址轉換不再加倍記憶體存取。
- RISC-V:U/S/M 特權模式、ecall 系統呼叫、satp+Sv32 分頁——本章概念的具體實現。

5 複習問題

1. 說明多道程式設計如何提高處理器使用率; 它依賴哪些硬體機制?
2. 內部碎裂與外部碎裂的差異? 分頁各消除/保留哪個?
3. 虛擬位址 0x00403A8C, 頁大小 4 KiB: 頁號與偏移各是多少 (十六進位)?
4. 頁錯失的處理流程為何? 為什麼頁錯失處理必須由 OS(而非硬體) 完成?
5. TLB 解決什麼問題?TLB 未命中與頁錯失有何不同?
6. 記憶體存取 100 ns、TLB 命中率 98%、TLB 存取 1 ns、頁表走訪需 2 次記憶體存取: 求平均位址轉換開銷。
7. 分頁與分段的設計哲學差異? 為什麼現代 ISA 傾向純分頁?

參考資料

[1] W. Stallings, *Computer Organization and Architecture*, 6th ed., Chapter 8.

[2] R. Arpaci-Dusseau and A. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*. https://ostep.org

[3] RISC-V International, *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture*.