作業系統 HW2 報告

11127137 資工三甲 黃乙家

1. **開發環境**



一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

這次作業使用 Go 語言開發，系統為 Windows 11 Pro 23H2，CPU i7-12700H，記憶體 32G，而為了使 Linux 也能方便編譯，也提供了 Makefile，且這次作業沒有使用任何第三方套件，理應不需要 go mod download 或者 go mod tidy 來整理套件。

1. 專案架構

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

專案根目錄包含 main.go，而 cmd 目錄裡為六種演算法的實作程式碼，init 目錄為主程式執行前的必要初始化(在這次作業僅作為方便 Debug 使用)，lib 目錄包含各種用到的資料結構，test 目錄為資料結構與各種方法的 Unit Test。

1. 實作方法與運作流程

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

一開始，主程式會從 stdin 讀入檔案名稱，讀檔成功後判斷方法數，把必要的參數帶入，呼叫對應的方法。

一張含有 文字, 螢幕擷取畫面, 動畫, 卡通 的圖片

自動產生的描述

上圖為 FIFO 方法的程式，使用佇列資料結構實作，另外實作了 Contains 方法以線性搜尋是否包含元素，如果不包含表示發生 Fault，否則如果佇列滿了則刪除最先進入佇列的 page，然後新的再放進佇列，而若未滿則直接放進佇列即可。

一張含有 文字, 螢幕擷取畫面, 軟體, 作業系統 的圖片

自動產生的描述

LRU 演算法採用 list 實作，另外還使用 hashMap 進行搜尋加速，每次 pageReference 先查看是否有在 list 中，若有則需將該 page 移到 list 最前面或最後面，表示這是最近使用到的，如果沒有且 list 已滿則將最後面或最前面的 page 替換掉(即淘汰最久沒被使用的)，為了方便實作，程式碼總是會替換最前面的 page。

一張含有 文字, 螢幕擷取畫面, 動畫, 卡通 的圖片

自動產生的描述

而 LFU/MFU 則是改成每次 reference 需要紀錄次數，如果需要 replace 則將 list 中次數最低或最高的淘汰。

一張含有 文字, 螢幕擷取畫面, 多媒體軟體 的圖片

自動產生的描述

LFU\_LRU 則是將兩種方法結合在一起，如果 reference 需將 page 移到最前面或最後面，同時記錄次數，並且當發生 replace 時若次數相同則依 LRU 的規則淘汰對應的 page。

1. **分析**

使用測資二來分析後發現，FIFO 和 MFU 的 Fault 次數為 15 比其他方法多，而 Replacement 次數是 12 也是最多的，推測原因可能是 page reference 重複的次數較多，然而 FIFO 並沒有考慮頻率問題，MFU 則是會將頻率高的淘汰掉，使得錯誤率變高。

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

使用測資一分析時，發現如果 page Frame Number 提高，FIFO 與 MFU 的錯誤率理應下降卻反而上升了，也就是增加頁面幀數卻導致錯誤數增加，這種現象被稱為 Belady’s anomaly

一張含有 文字, 螢幕擷取畫面, 繪圖軟體, 多媒體軟體 的圖片

自動產生的描述