教學作業系統的設計

李 聰　　黃耀毅
國立中山大學電機工程學系
E-mail: tlee@mail.ee.nsysu.edu.tw, d003010014@student.nsysu.edu.tw

摘要

作業系統是一個電腦系統的軟硬體資源管理中心，作業系統的設計技術是資訊系統技術的核心技術。由於其所牽涉的軟硬體系統的多面性與其自身設計的複雜度，在作業系統教學時，規劃配合的學習實驗給初學者做學習與練習使用，要盡量讓同學能練習到課程教學的核心技術部分，在同學部分，有其學習的技術障礙。我們設計了一個教學型作業系統，目的為降低初學者的技術障礙，同時又能練習到教學上的核心技術部分，讓學習者能有整體的技術瞭解，能練習到關鍵部分，也僅需做各部分的低負擔設計，以降低其學習技術門檻，是此教學型作業系統設計的目的。

關鍵詞：作業系統、抽象化、機器模擬器、負載語言、實習規劃。

Abstract

An operating system is a center of resource management in a computer and information system. Design techniques of operating systems are core techniques of information system technology. Many aspects of hardware and software systems are related to operating systems. An operating system itself exhibits much design complexity. Hence, in teaching the operating system course, we need to plan laboratory practices of core techniques taught in the class for beginning learners. For learners, there are some technical barriers due to such complexity. We designed an instructional operating system to reduce such technical barriers and enable learners to practice required core techniques. Thus, we can achieve the goal of the instructional operating system that learners can gain system understanding and practice core parts of operating system design with low overhead and low technical threshold.

Keywords: Operating systems, abstraction, machine, load language, lab planning.

1. 前言

作業系統是一個電腦系統的軟硬體資源管理中心，作業系統的設計技術是資訊技術的核心技術，因此在各電機與資訊相關學系的教程中，均有作業系統的課程，列在資訊系統設計與進階程式設計的範疇。對於其所學技術的應用，可以使用於對電腦系統的管理與操作、有使用系統資源的應用程式設計、並行與平行程式設計、作業系統核心設計、作業系統之子系統設計與維護、與微處理機與系統晶片的嵌入式設計等。

為了做有效的作業系統教學[1][2]，除了在課程教學內容上，需要做設計體系及技術分析與設計的整理與準備。對於課程中所教的設計體系與各部分的設計技術，需要規劃施行在實務實現的程式接觸理解與練習。由於作業系統所牽涉的軟硬體系統的多面性與其自身設計的複雜度，在作業系統教學時，規劃配合的學習實驗給初學者做學習與練習使用，要盡量讓同學能練習到課程教學的核心技術部分。實務教學時，在同學部分，有其學習的技術障礙。如何提供初學者一個低技術障礙又有全貌性的理解，是其教學上所需要的課程重要輔助。

在作業系統的實務教學上，我們可以分為實務作業系統學習與教學型作業系統[3]實務練習兩個部分來考量與教學上的實施：實務作業系統學習與教學型作業系統實務練習。

對於實務作業系統的學習，我們需要有其原碼，可以安裝執行，也要有相關的技術資料敘述其核心設計的技術與規劃。目前較方便取得與研讀瞭解的實務作業系統列舉如下：Linux[4], BSD UNIX[5], 與 Minix[2]。其中 Linux 與 BSD UNIX 的系統是真實可以使用的系統，其原碼較為龐大，但可以根據技術資料的瞭解，分段去追蹤與使用。Minix 是一種教學型的作業系統，其雖簡單，但也是一個完整的作業系統，也可以安裝運作，其原碼較小，比較適合初學者追蹤瞭解，但其和下列教學型作業系統比較，缺少了規劃適合初學者做的系列練習。

對於教學型作業系統實務練習，在教學型作業系統[3]上，除了上述 Minix 作業系統外，另有 Nachos[6][7]、MIT 作業系統工程課程[8]、Harvard OS/161[9]等等作業系統與教學練習規劃。在既往的使用上，我們發現這些教學型作業系統與練習資料上，在教學實施上有一些實際上的問題，包括學習的技術障礙比較高、練習的部分較為分散、整體性理解不足、作業系統的重點部分之練習規劃不足等。

基於此方面的考量，我們決定規劃與設計一個

教學型作業系統，以達到改進上述問題的目的，使得我們修課的同學不需要較高等級的資訊設計能力與付出功夫，也可以在課程過程中，在助教的帶領輔導下，逐步瞭解一個教學型作業系統的組成與系統整合，並逐步練習到各教學重點的作業系統實做策略與機制。此教學型作業系統的目的為降低初學者的技術障礙，同時又能練習到教學上的核心技術部分，讓學習者能有整體的技術瞭解，能練習到關鍵部分，也僅需做各部分的簡化設計，以降低其學習門檻，是此教學型作業系統設計的目的。

在此研究中，我們規劃了此教學型系統的設計技術，包括對此作業系統設計的抽象化[10]拆解，形成軟硬體各子系統的分解設計。另外，對於硬體系統的執行，我們規劃了對其進行簡化的模擬器[11]與負載語言設計，其對硬體系統及機器語言相關的理解與程式細節形成相當成度的簡化，因此可以大為降低初學者對此方面的學習負擔，此簡化部分並不會減少其在作業系統管理部分所需做的機制與策略的練習，因而可以集中心力在此課程重點部分做實務的理解與練習。

教學實務實施上，目前我們所開設的作業系統課程已使用此教學型作業系統與其實習規劃，我們實務的經驗上，初次學習的同學可以在助教的輔導上，適應此課程在程式實務上的差距，並且理解整體系統設計，進行所規劃課程重點上的作業系統機制與策略的程式實做實習。

在以下各節中，我們將先敘述此教學型作業系統的系統架構，再介紹機器模擬器與負載語言，接下來說明配合此教學型作業系統的實習規劃，最後做一結論。

2. 系統架構

在我們的教學作業系統研究設計中，我們設定了兩個主要的目標：(1)降低作業系統初學者學習作業系統的負擔與(2)支援作業系統中各部份元件的分解及整合設計模擬運作。

針對這兩個目標，我們分別規劃了一些實做的做法：

- 規劃簡化機器與機器負載語言，以減少實務上的細節操作設定的使用者負擔 。
- 透過模擬的方式來了解元件之間的運作 。
- 各元件形成一模型並提供相關的抽象介面(abstraction)，透過此介面來整合連結各個元件。
- 各個元件可依設計需要來替換內部的。機制，但提供的相關抽象介面仍然不變。

對於整體軟體環境的設計，如圖 1 所示，此教學型作業系統可分成四個部分的設計：

- 負載：包括若干個行程(process)，以為系統的工作負載。
- OS 核心：OS 組成包含一個處理機制以及一些服務常式。
- CPU 模擬器：組成包含模擬器、直譯器、以及分散事件行列。
- 硬體：作業系統周邊的硬體設備，常見的硬體設備有計時器、鍵盤、螢幕等。

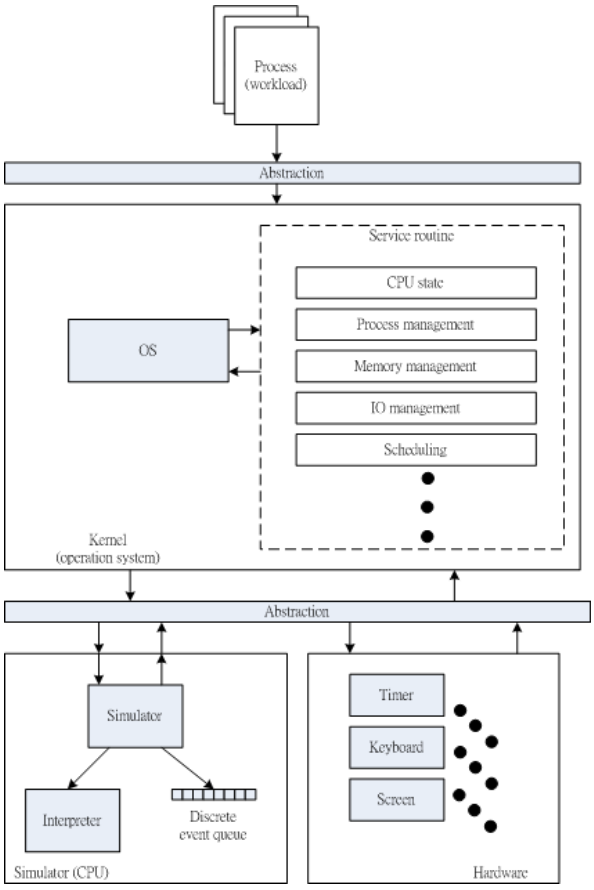

圖 1. 教學作業系統軟體環境

此四個部分的設計均使用軟體組成，其彼此之間透過抽象化的介面來溝通整合。

- 核心所提供的抽象介面：其包括針對外部使用者所提供的介面，可做初始化整個教學型作業系統，並載入 process、重設 OS 模擬器、開始進行模擬機器與 OS 的執行。也包括針對教學型作業系統內部(CPU, 硬體)的介面，可做處理傳入的負載與將負載傳給各作業系統的子系統執行。
- CPU 所提供的抽象介面：開始模擬 CPU 運作，處理行程負載與根據負載做對 event queue 的處理。
- 硬體所提供的抽象介面：提供硬體的查詢介面。

在此軟體環境中，我們可以設計作業系統的程式，並且將作業系統的設計，插入此軟體模擬的整合環境，也可以隨時調整作業系統的各部分機制與策略的設計，以觀察不同機制與策略，在系統管理上的不同效應。

3. 機器模擬器與行程負載語言

在行程的負載上，在目前的模擬器中，我們將 CPU 與行程做了不影響作業系統功能的簡化，這可以使得學習者可以不需要去讀懂與處理真實 CPU 的指令集與機器碼，也不需要做 CPU 硬體的初始化動作。

我們將此行程負載以一種程序性的簡化負載語言加以敘述，其內包括一般行程對機器與 OS 會產生的負載效應與引用，包括一般計算負載、trap 負載、exception 負載、interrupt 負載，及這些負載的整合流程敘述。

如圖 2 所示，行程負載可以有一些變數與參數的設定，並可以在行程中使用與修改，行程中可以有控制流，並有處理時間，也可以做系統呼叫。如此簡化的行程內不需要有真實的功能，亦可以有效的作業系統操作，包括行程排程、輸入輸出、檔案管理、與記憶管理。

```
variable_list           # 描述程序會用到的變數
{
        times  10       # 變數 times，內容爲數值 10
        data   "abc"    # 變數 data，內容爲字串 abc
}
scheduling_state        # 系統需要用到的參數
{
        priority 10     # 設定優先權
}
process test_process
  # 此爲一個程序名爲 test_process 的程序
{
  # 程序的內容如下
        # 將 times 變數的值載入暫存器 0 中
        load(0, times) ;
        loop times      # 迴圈做 times 次
        {
                load( 1, data) ;
                # 花費 5 個單位時間做處理
                time_advance(5) ;
        }
        # 做系統呼叫，將此程序睡眠 10 個單位時間
        sleep(10) ;
        # 花費 5 個單位時間做處理
        time_advance(5) ;
        exit( ) ;# 做系統呼叫，終止當前程式
}
```

圖 2. 行程負載範例

行程的記憶管理可以使用記憶足跡的記憶負載敘述，可以驅動作業系統內的虛擬記憶系統，做頁(page)的記憶管理。

使用此系統的學習者可以將所想要執行的行程負載用此負載語言先行撰寫為行程描述檔，再如圖 3 所示，將其透過對應此語言的語彙分析器與語法剖析器輸入到模擬環境內部的資料結構，成為可模擬的各個行程負載。

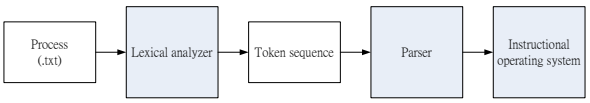

圖 3. 輸入行程負載到教學作業系統軟體環境

4. 實習規劃

由於考量修課同學課程負擔，目前實際用於課程的實習由三個專案所組成，從軟體環境的介紹，作業系統整合、行程負載的介紹、到作業系統內各個子系統內的程式介紹，到子系統內關鍵軟體元件的設計，其涵蓋印證作業系統整合設計與子系統設計的各個層面與關鍵設計實習，以達到支援作業系統課程學習的目標。

專案一為入門使用與學習，其分為四個基本的練習：

- 練習一: 熟悉使用教學型作業系統。
- 練習二: 自己做一個行程的範例，將此行程送到教學作業系統執行。
- 練習三: 追蹤教學型作業系統的主要運作方式。
- 練習四: 追蹤作業系統處理負載的方式。

專案二程序排程器使用與設計練習，其分為兩個基本的練習：

- 練習一: 撰寫一個 FIFO policy 的排程器。
- 練習二: 追蹤採用 multi-level feedback queue policy 的排程器的做法。

專案三為虛擬記憶的設計與使用練習，其分為兩個練習：

- 練習一: virtual memory 的實作。
- 練習二: memory footprint 的使用。

5. 結論

在本研究中，我們設計了一個教學作業系統，透過抽象分解出的子系統，我們整合出系統各部分硬體、作業系統各個組成與整合、與行程工作負載的軟體模擬環境。學習者可以敘述行程工作負載，以之操作作業系統，也可以學習作業系統各部分的

功能與整合機制，並撰寫關鍵軟體元件，觀察與測試其策略對系統執行與管理的效應。我們實際使用此教學作業系統於作業系統的課程，驗證此教學作業系統可達到我們降低學習技術障礙與負擔、整體系統瞭解與學習、與關鍵軟體技術實習的設計目標。

參考文獻

[1] A. Silberschatz, P. Galvin, G. Gagne, Operating System Concepts, 8th ed., Wiley, 2008

[2] A. S. Tanenbaum and A. S. Woodhull, Operating Systems Design and Implementation, 3rd ed., Prentice Hall, 2006.

[3] C.L. Anderson and M. Nguyen, "A survey of contemporary instructional operating systems for use in undergraduate courses,"*Journal of Computing Sciences in Colleges*, Vol. 21, No. 1, Oct. 2005.

[4] D.P. Bovet and M.Cesati, U*nderstanding the Linux Kernel*, 3rd ed., O'Reilly, Nov. 2005.

[5] M.K. McKusick and G.V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley, 2005.

[6] J.E. Gary, "Using Nachos is an upper division operating systems course," *Journal of Computing Sciences in Colleges*, Vol. 18, No. 2, Oct. 2002.

[7] W. A. Christopher, et al., *The Nachos Instructional Operating System*, University of California, Berkeley, EECS Dept., Technical Report No. UCB/CSD-93-739, Nov. 1992.

[8] F. Kaashoek, *Operating System Engineering*, MIT Open Courseware,http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-828-operating-system-engineering-fall-2006/, 2006.

[9] D.A. Holland, "A new instructional operating system," *Proc. of the 33rd SIGCSE Technical Symp. on Computer Science Education*, Feb. 2002.

[10] D. B urner, et al., "Modular design through component abstraction," *Proc. of the 2004 Int'l Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, Sep. 2004.

[11] S. Salmon and H. ElAarag, "Simulation based experiments using EDNAS: the Event-Driven Network Architecture Simulator," *Proc. of the Winter Simulation Conference*, Dec. 2011.