行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※※※※

※ 即時核心程式上的 Java 與通信子系統的研究與製作(Ⅱ) ※

※An Implementation of Java Personality and Communication ※

※ Facility on a Real-time Kernel(Ⅱ) ※

※※※※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：□個別型計畫 ¦整合型計畫

計畫編號：NSC **89-2213-E-009-021**

執行期間：88 年 8 月 1 日至 89 年 7 月 31 日

計畫主持人：張瑞川

執行單位：國立交通大學資訊科學學系

中 華 民 國 89 **年** 09 **月** 07 **日**

行政院國家科學委員會專題研究計畫成果報告

即時核心程式上的 Java 與通信子系統的研究與製作(Ⅱ)

An Implementation of Java Personality and Communication Facility on a Real-time Kernel(Ⅱ)

計畫編號：NSC 89-2213-E-009-021
執行期限：88 年 8 月 1 日至 89 年 7 月 31 日
主持人：張瑞川　　國立交通大學資訊科學學系

1. Chinese Abstract

本計劃為期二年，研究的目的在於探討新的即時核心程式架構，利用元件化技術來研發一個完全模組化的即時系統。由於 Java 的開放性技術儼然成為構建網路上應用程式的標準，Java **應用程式具有高度的可移植性優點，可以跨平台執行，節省應用程式的開發成本。因此，我們希望在我們的即時核心程式上提供** Java **的** personality **以及元件化的網路通訊模組，此網路通訊模組以支援** TCP/IP **通訊協定為主，因為** TCP/IP **是網際網路上最主要的通訊協定。 我們提供支援** Java **功能的平台。此** Java **平台包含一個** Java Virtual Machine**，且將成為我們研究** Java **技術的平台，進一步發展滿足即時應用與嵌入式系統的需求。**

在本報告中，我們將描述我們移植 Java Virtual Machine, TCP/IP **到即時核心程式上的成果。**

關鍵詞： Java; Java Virtual Machine; TCP/IP; **即時；嵌入式系統；元件**

Abstract

This project last for two years and the goal of this integrated project is to explore a new technique for a real-time kernel, which uses component mechanism to develop a completely modularized real-time system.

Because Java's open technology is becoming the de facto standard for building networked applications. Java applications have the advantage of high portability that can be executed on cross-platforms. This promotes application sharing and saves application development costs. Our goal is to provide the Java personality and componentized communication facility on the real-time kernel. We use TCP/IP Stacks as our communication protocol since TCP/IP is widely used as Internet communication standard. We will provide a Java platform, which will serve as a test-bed for our research of Java technology. We plan to develop real-time embedded Java based on our real-time kernel and explore the research issues and applications of Java Network Computers.

In this report we will describe our results on porting a Java Virtual Machine and TCP/IP to a real-time kernel.

Keywords: Java; Java Virtual Machine; TCP/IP; Real Time; Embedded Systems; Component

2. 計畫緣由與目的

我們探討新的即時核心程式架構，以達到一個完全模組化的即時系統

軟體元件庫。在這個新的元件式組合架構下，作業系統相關的程式均高度模組化，每個模組都是獨立的軟體元件，藉由新的組合機制，我們可以很迅速的增加、修改或置換不同的軟體元件，如此系統便有了不同的 personality，有助於我們在系統軟體技術方面的研究。此系統可以成為我們研究的平台，例如，藉由置換不同的軟體元件，核心程式可以很容易的研究不同的管理策略，諸如排程方法或記憶體管理策略等等。而發展環境也可以針對不同的需要應用（例如即時應用、嵌入式應用、行動式計算應用等等），經由發展及置換不同的軟體元件，即可展現不同的 personality，而不必對核心程式大肆修改，因此也可以很快地應用於不同的產品上，如 Set-top box、Media Server、Cellar Phone 等等，節省軟體開發的成本與時間。

我們的總體目標即是利用元件化技術來研發一個新的即時系統，而其中，研究與製作核心程式、通訊模組及儲存裝置（檔案系統）、與發展環境。此三項需要高度的整合，缺一不可。

我們將提供支援 Java 功能的平台(Java Platform)，因為開放式系統是所有軟體及硬體廠商所追求的目標，希望藉由開放式的標準能達到軟體的互通性，節省軟體的發展成本。然而長久以來，軟體廠商仍需要針對不同的作業系統、不同的硬體架構來發展不同版本的軟體，如 Macintosh 版、Windows 版、DOS 版、UNIX 版等等。即應用程式無法直接在不同的硬體或作業系統上執行，都必須經過某種程度的修改與移植。直至 1995 年 3 月，Sun Microsystems 公司首先推出 Java 技術，希望藉由 Java 開放式的平台，使任何執行於 Java Virtual Machine 上的 Java 應用程式都只要發展一次即能跨平台執行，即 "Write Once，Run Anywhere"！許多主要的電腦公司如 Netscape、IBM、Oracle 及 Microsoft 等等及學術界相繼投入研究 Java 技術。Java 開放式的技術儼然已成為 de facto 標準。因此，我們在發展自己的即時系統平台時，我們希望此系統具有支援 Java 功能的 personality，成為 Java Computer。

國內外已有許多 Java 技術應用的研究(詳細情形請見參考文獻)，目前 Java 在嵌入式系統 (Embedded Systems)與即時系統的應用引起多方面熱烈地討論。Java 應用在嵌入式系統的研究，有如 Sun Microsystems、Mitsubishi、NewMonics Inc.、Iowa State University、UCSC、Utah University、、、等等，而 Java 應用在即時系統的研究亦有 NewMonics Inc.、Iowa State University、CMU 等等，他們提出了他們的 Embedded Java、Real-time Java、Real-time Java Virtual Machine 等等，但 Java 是否適合即時應用仍無定論， Java 應如何的延伸以符合即時需求，仍需深入研究。因此，我們將在我們的 Java 平台上進一步研究 3 項重要課題:

- Java 是否適合應用於嵌入式系統以及如何達到嵌入式系統的需求，我們希望我們的 Java 平台能符合 Sun Microsystems 所提出的嵌入式 Java 標準(Java Embedded Standard)，
- Java 是否能夠達到即時系統的要求, Java Virtual Machine 應與核心程式如何配合以達到即時應用，
- Java Network Computer 的應用。

因此，我們研究在元件化即時核心程式上設計製作一個支援 Java 功能的研究平台，並針對即時應用及嵌入式系統的需來調整此 Java 平台，例如改進排程方法，改善記憶體管理與資源回收（Garbage Collection）方式，增加 Java 應用程式與 Java Virtual Machine 及核心程式之間的互動等等。網路通訊模組則以支援 TCP/IP 通訊協定為主，因為 TCP/IP 是網際網路上最主要的通訊協定，網際網路上的電腦都需經由 TCP/IP 來與其他電腦通訊，作業系統都有支援 TCP/IP，如 Linux、BSD 等等。我們也將針對即時應用及嵌入式系統的需來，對 TCP/IP Stacks 在資源使用上做最佳化的研究，如研究資源預留(Resource Reservation)策略及簡化資源需求等等，以滿足即時應

用及嵌入式應用。

3. 結果與討論

本計劃今年完成的研究成果如下:

(1) 通訊協定 TCP/IP 的移植。
(2) 建置元件化的視窗系統。
(3) 移植 POSIX threads 與 Java Virtual Machine (JVM)。

茲將其研究成果分述如下:

3.1 通訊協定 TCP/IP 的移植

TCP/IP 網路協定已經成為今日有線網路通訊的必備標準，為了讓我們的系統也能跟世界上大多數的機器溝通，因此我們決定在我們的系統上加入 TCP/IP 的網路功能。

實作上，我們是以一個通訊協定組 (Communication stacks) 的方式在我們的系統上加入 TCP/IP 的網路功能。在移植過程中，我們將 Linux 作業系統的 TCP/IP 程式組作了相當的修改。首先，我們將 Linux 的使用者和核心模式 (user/kernel mode) 整合成在單一定址空間 (single address space) 上執行，這樣可以讓所有網路程式跳過系統呼叫 (system call) 的過程而直接使用函式呼叫即可。此外，我們也將原本 Linux 的 TCP/IP 程式組跟檔案系統互相結合的特性刪除，為了做到這點，我們將原來程式碼裡所有跟檔案系統相關的資料結構一律予以刪去，而最重要的 socket 描述值則換成用相關資料結構的指標 (socket address) 來代替原來的檔案描述值 (file descriptor)，我們這樣做的目的主要是將系統裡的通訊協定能獨立成一個單一模組，盡少讓模組間產生互相依賴或緊密結合的情形，以符合我們元件化模組設計的原則。

3.2 建置元件化的視窗系統

為了要在 JVM 上能提供視窗及圖形支援 Abstract Window Toolkit (AWT)，以提供使用者圖形化介面，我們也在系統上實作了一套視窗系統，以提供 AWT 所需的函式庫。該視窗系統主要分兩層：下層是由驅動程式函式庫 SVGA lib 所提供的函式組合而成的圖形裝置介面 (GDI) 模組；上層則是根據 MVC 的架構，將視窗系統分成三個主要的模組，即： Model、View 及 Controller。在控制及使用硬體裝置時，是透過下層的圖形裝置介面 (GDI) 模組去存取。換言之，我們利用圖形裝置介面 (GDI) 將上層的 MVC 架構及底層與機器硬體有關的驅動程式隔開。這樣做對於我們將來要把視窗系統移植到不用的硬體平台時既方便又簡單，因為我們祇要修改跟機器架構有關的裝置驅動程式，然後提供驅動程式函式庫 SVGA lib 相同的函式即可。

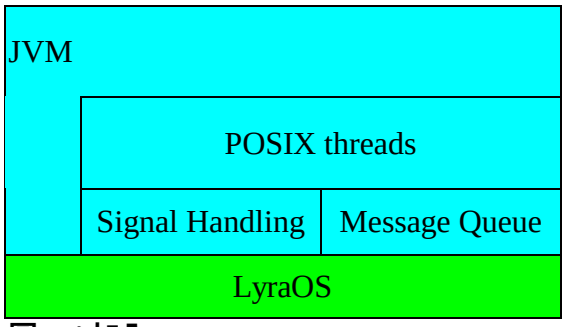
視窗系統模組化的設計除了享有高度移植性的好處以外，在整個視窗系統的擴充方面也很容易。我們隨時可以根據原來的一些設計原則輕易的新增我們所要的 widget，如：新的硬體裝置控制器，讓硬體裝置具有驅動程式後，我們便可在視窗系統上直接設計我們想要的使用者控制介面。

3.3 移植 POSIX threads 與 Java Virtual Machine (JVM)

因為 JVM 支授 multithreading 而且是以 POSIX threads 為基礎的關係，讓我們在設計系統時有兩個考量：即（一）將我們既有的 threads 一律換成符合 POSIX threads 的標準；或（二）在我們既有的系統上再嵌入一層 POSIX threads 的元件。為了擴充性的考量，我們最終選擇了後者。

而又因為 multithreading 必須提供一些基本的 Inter Processes Communication (IPC) 機制，於是我們將 Signaling 和 Message Queue 分別以元件式的模組加入到我們的系統上，使得我們的系統更趨完善和功能多元化。

完成上述的工作後，我們的系統架構就會變成如圖一所示：



圖一：加入 POSIX threads、Signal Handling **和** Message Queue **後的系統架構**

在圖一中，我們已完成並通過測試的部分有：即時核心程式 (**即** LyraOS)、Signal Handling **、** Message Queue **以及** POSIX threads **。目前則正進行** JVM **的測試。**

4. 計畫成果自評

本計劃今年的執行目標在移植 TCP/IP **方面已經完成，我們在研究中運用了軟體元件化技術, 完成了釐定** TCP/IP stack **與核心程式記憶體管理及驅動程式之間的界面，並成功將之整合成通訊協定模組成為系統中之軟體元件。**

此外，我們也順利地在我們的系統上實作了一個視窗系統，使得我們的系統能提供視窗及圖形介面的支援。我們將視窗系統的設計元件化，使得我們的視窗系統在移植性和擴充性上都有不錯的效果。

在建構 Java Virtual Machine (JVM) **方面，我們目前已完成記憶體配置與網路通訊協定上以及我們系統間的介面釐定。我們所面對的問題主要發生在** Multithreading **上：為了移植的需求，我們的系統必須能提供標準執行緒** (POSIX threads) **的介面，使得我們的系統能順利與** JVM **上之** Java threads **結合與溝通。到目前為止，大部份的** POSIX threads **介面我們都已經完成，尚有一小部份的功能尚待驗證，而我們也將一切努力以期能在最短時間內完成所有的移植工作並作更進深入的** Real-time Java **研究。**

5. 參考文獻

5.1 參考文獻 – Garbage Collection

[1] A. W. Appel, J. R. Ellis, and K. Li. “Real-time concurrent garbage collection on stock multiprocessors,” In Proceedings of SIGPLAN PLDI’88, pages 11-20, June 1988.

[2] A. W. Appel and K. Li, “Virtual memory primitives for user programs,” In Proceedings of the ASPLOS IV, pages 96-107, April 1991.

[3] D. L. Andre, “Paging in Lisp programs.,” Master's thesis, University of Maryland, 1986.

[4] A. W. Appel, “Garbage collection can be faster than stack allocation,” Information Processing Letters, 25(4):275-279, June 1987.

5.2 參考文獻 – JAVA Technology

[1] K. Asha, “Java-Linux 1.1.1 HOWTO,” http://www.blackdown.org/java-linux/JDK-1.1.1/Howto-1.1.1/Java-Linux-1.1.1-Howto.html, May 18, 1997.

[2] A. Baird-Smith, “Jigsaw – An Object-Oriented Web Server in Java,” http://www.w3.org/pub/WWW/Jigsaw/.

[3] B. Ford, K. V. Maren, J. Lepreau, S. Clawson, B. R. J. Turner, “The Flux OS Toolkit: Reusable Components for OS Implementation,” Proc. of the Sixth Workshop on Hot Topics in Operating Systems, May 1997.

[4] B. Ford, G. Back, G.Benson (U.C. Davis), J. Lepreau, A. Lin (MIT), and O. Shivers (MIT),“The Flux OSKit: A Substrate for OS and Language Research,” Proceedings of the 16th ACM Symposium on Operating Systems Principles, Saint-Malo, France, Oct. 1997.

5.3 參考文獻 – TCP/IP Stack

[1] D. E. Comer and D. L. Stevens, “Internetworking with TCP/IP Vol I, II, III,” Prentice-Hall International, Inc., 1993.

[2] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, “The Design and Implementation of the 4.4 BSD Operating System,” Addison-Wesley Publishing Company, 1996.

[3] W. R. Stevens, “UNIX Network Programming,” Prentice-Hall International, Inc., 1991.

[4] S. A. Thomas, “IPng and the TCP/IP Protocols – Implementing the Next Generation Internet,” John Wiley & Sons, Inc., 1996.