M3K에서의 쓰레드 컴포넌트 구현

김 영호°, 고 영웅, 유 혁
고려대학교 컴퓨터학과

Implementation of Thread Component in M3K

Young-ho Kim°, Young-woong Ko, Chuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

마이크로 커널 구조는 필수 불가결한 커널 기능만을 가지게 되며, 운영체제의 기능성은 서버로 동작하게 된다. 따라서 모노리틱 운영체제에 비해서 커널 기능의 확장 및 개발이 용이하다는 장점을 가지게 된다. 본 연구에서는 기존에 제시된 마이크로 커널의 접근 방식에서 추가적으로 멀티미디어를 지원할 수 있는 멀티미디어 마이크로 커널(M3K)을 구현하고 있다. 특히 M3K는 멀티미디어의 실시간 특성 및 기능을 제공할 수 있는 구조로서 컴포넌트에 기반한 커널 프레임워크를 사용하고 있다. 본 논문은 M3K 마이크로 커널을 구현함에 있어서 커널 구조 자체를 컴포넌트화 시켜 필요한 기능만을 선택적으로 결합해서 사용할 수 있는 방법을 제시하고 있으며, 현재 동작중인 쓰레드 컴포넌트의 아키텍처 위주로 설명한다.

1. 서 론

마이크로 커널 구조[1][2][3][4][5][6][7]는 모노리틱 커널에 비해서 크기가 작고 커널의 기능성은 사용자 영역에서 수행되는 서버로 구현된다. 따라서 커널 내부에서 발생되는 지연이 작고 예측 가능하므로 실시간 시스템으로 활용되고 있다. 대표적인 예로서 QNX[7]와 같은 시스템을 언급할 수 있으며, 현재 개발중인 fiasco[4]와 같은 제 2세대 마이크로 커널 운영체제는 실시간 특성을 바탕으로 해서 멀티미디어 시스템으로 확장하고 있다. 일반적으로 마이크로 커널은 하드웨어 초기화 및 IPC, 쓰레드 관리, 주소 공간 관리 등과 같은 기능만을 제공하고, 메모리 관리 및 디바이스 드라이버, 파일 시스템, 네트워크 시스템 등은 사용자 영역에서 서버로 구현하고 있다. 기존의 마이크로 커널에 있어서 문제점은 커널을 구성하는 각 모듈이 밀접하게 결합되어 있어 추가적으로 기능확장을 하기에 제한적이며, 사용 가능한 정책이 한정적이다. 또한 성능을 향상시키기 위해서 커널을 수정하려면 어셈블리 언어와 같은 하위 레벨 언어를 사용해야 한다. 일례로 fiasco의 이전 버전인 L3/L4[3] 마이크로 커널은 대부분의 코드가 어셈블리 언어로 구현되어 있으며, 새로운 스케줄링 정책을 제공해주기 위해서 소스코드를 수정하는 것은 어려운 문제이다. 따라서 본 연구는 이러한 어려움을 해결하기 위해서 컴포넌트에 기반한 커널 프레임워크[9]의 개념으로 접근하고 있으며, 이것은 마이크로 커널을 구성하는 각 모듈을 하드웨어에 의존적인 부분과 그렇지 않은 부분의 두 단계로 구분하여, 각 단계의 내부 기능을 컴포넌트로 구현함으로써 새로운 기능의 추가 및 수정을 용이하게 하고 있다.

본 논문에서는 M3K라고 명명된 멀티미디어 마이크로 커널(MultiMedia Microkernel)의 아키텍처를 간략히 설명하고, 마이크로 커널을 이루는 컴포넌트의 일부로서 구현된 쓰레드 컴포넌트에 대해 설명하고자 한다.

이 연구는 한국과학재단의 1997년도 특정기초 연구과제 연구비 지원에 의한 결과임(과제번호 97-01-00-09-01-3)

2. M3K 커널 모델 및 구현

2.1. 구현 환경 및 커널 모델

M3K는 멀티미디어를 지원하기 위해 본 연구에서 구현 중

인 마이크로 커널이다. 이러한 M3K는 Linux상에서 어셈블리 언어와 객체지향언어인 C++를 이용하여 커널 이미지를 만들고, 직접 제작한 부트 로더를 사용하여 Intel 386[8] 플랫폼 상에서 수행시켰다. M3K는 하드웨어를 추상화(abstraction)하는 문맥(context), 주소(space), 예외(exception) 객체로 구성된 코어커널(CoreKernel)과 이들이 제공하는 인터페이스를 통해 쓰레드의 기능을 구현하는 쓰레드 컴포넌트로 이루어져 있다. 그림 1은 이러한 M3K의 코어커널과 쓰레드 컴포넌트의 관계를 나타내고 있다.

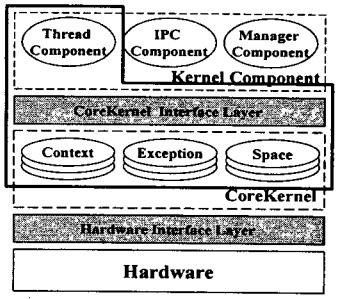

그림 1. M3K 커널 모델

그림 1에서 쓰레드 컴포넌트 이외에 프로세스간의 통신을 위한 IPC 컴포넌트 등이 커널 컴포넌트 내에서 구현 중에 있다. 매니저 컴포넌트는 코어커널 인터페이스를 사용해서 구현된 컴포넌트들을 관리하는 역할을 한다. 굵은 선으로 표시된 부분은 코어커널의 문맥, 주소, 예외 객체와 커널 컴포넌트내의 쓰레드 컴포넌트로, 현재 구현된 M3K를 나타내고 있다.

2.2. 부팅 및 하드웨어 초기화

그림 2처럼 커널 이미지는 코어커널의 문맥, 주소, 예외 객체로부터 각각 얻어진 오브젝트 파일과 쓰레드 컴포넌트의 오브젝트 파일을 이용하여 생성된다. 이러한 커널 이미지를 부팅시 메모리에 로딩시키기 위해서, 이미지 툴을 사용하여 부트로더를 커널 이미지 앞에 연결한다. 컴퓨터의 부팅이 시작되면 부트 로더는 커널 이미지를 메모리에 로드하고 제어를 커널의 초기화 부분에 넘겨준다. 제어권을 넘겨받은 커널은 코어커널의 문맥, 주소, 예외 객체를 생성하기 위해 하드웨어에 대한 검사와 초기화를 수행한다. 특히 이 때 커널 영역의 수행을 위한 주소공간을 정의하고, 전체 문맥 객체와 주소 객체를 생성하기 위해 필요한 자원을 예약하고, 예외 객체를 처리하기 위한 인터럽트 벡터 테이블을 초기화한다.

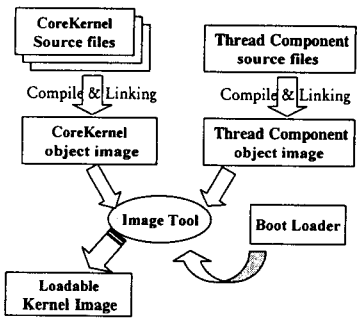

그림 2. 커널이미지 생성 단계

2.3. 코어커널 구현

코어커널은 하드웨어 플랫폼에 의존적인 부분에 대한 추상화를 제공해 주며, 문맥, 주소, 예외 객체로 이루어져있다. 문맥 객체는 프로세서의 상태 정보를 저장하기 위해 IP(Instruction Pointer), SP(Stack Pointer), 일반 레지스터들의 값, 페이지 디렉토리의 주소 등을 가지고 있다. 이러한 정보들은 하나의 문맥이 중간에 멈추고 다시 실행되더라도 마치 끊김이 없이 연속적으로 실행된 것처럼 보이게 한다. 문맥 객체에서는 이러한 정보 외에 하나의 문맥에서 다른 문맥으로 전환(Context Switch)하기 위한 인터페이스도 제공하고 있다. 이러한 문맥 객체의 정보와 인터페이스는 이후 쓰레드 컴포넌트에서 쓰레드 상태와 전이를 구현하는데 이용된다. 주소 객체는 프로세서가 실행되기 위한 주소 공간을 정의하는 것으로, 베이스(base)와 크기(size)를 갖는 세그먼트들로 표현된다. 특히 주소 객체에서는 수행될 수 있는 텍스트가 존재하는 코드 세그먼트와 이 때 필요한 데이터를 저장하는 데이터 세그먼트를 정의해서 주소 공간을 관리하고 있다. 또한 이러한 주소 객체에서는 주소 공간을 생성하고 삭제하는 인터페이스를 제공한다. 예외 객체는 시스템에서 발생하는 예외 상황에 대한 핸들러와 주변장치에서 발생하는 하드웨어 인터럽트 처리를 위한 핸들러를 지정할 수 있도록 인터럽트 벡터 테이블을 관리한다. 이러한 예외 객체에서는 예외 상황에 대한 핸들러를 등록시키는 인터페이스와 이미 등록된 핸들러를 해제하는 인터페이스를 제공한다.

2.4. 쓰레드 컴포넌트 구현

코어커널에서 제공하는 인터페이스를 이용하여 한 주소 공간에 여러 개의 문맥이 동시에 존재하는 쓰레드 컴포넌트를 구현하였다. 이러한 쓰레드들은 일정한 시간 간격으로 동일한 주소 공간에서 문맥 전환을 하며 라운드 로빈(Round Robin) 방식으로 스케줄링 된다. 그림 3에서 TCB(Thread Control

Block)는 주소 공간이나 프로세서 상태와 같은 하드웨어 종속적인 정보를 가지고 있는 부분과 수행 시간이나 우선 순위(priority)와 같이 하드웨어 독립적인 정보를 지닌 부분으로 나뉘어진다. 전자는 코어커널에서 제공하는 인터페이스를 사용하여 정의되며, M3K에서 구현되는 모든 쓰레드 컴포넌트에서 필수적인 요소이다. 후자는 기존에 사용되는 쓰레드 컴포넌트에서 확장된 실시간 쓰레드 컴포넌트 및 멀티미디어 쓰레드 컴포넌트에서 추가적으로 필요한 정보를 지원할 수 있다.

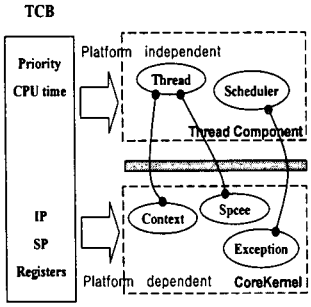

그림 3. M3K의 쓰레드 컴포넌트 구조

쓰레드 컴포넌트에서는 쓰레드를 실행하기 위한 문맥 객체를 생성하고 초기화하며, 주소 객체의 인터페이스를 이용하여 문맥 객체들이 실행될 주소 공간을 확보한다. 스케줄러는 생성된 여러 문맥 객체들을 관리하며, 정의된 스케줄링 정책에 따라서 스케줄링 시킨다. 본 연구에서는 라운드 로빈 방식의 스케줄러를 지원하기 위해서 일정한 시간 간격으로 인터럽트를 발생시키는 타이머 인터럽트를 사용하고 있다. 타이머에 의해 일정 시간마다 인터럽트가 발생하고, 이렇게 발생된 인터럽트는 타이머 인터럽트 핸들러를 통해 등록된 스케줄러를 수행시킨다.

실시간 스케줄링을 위해서는, 위에서 언급된 실시간 쓰레드 컴포넌트 또는 멀티미디어 쓰레드 컴포넌트에 실시간 스케줄링을 할 수 있는 스케줄러를 추가한 후 등록시키는 것으로 해결할 수 있다. 따라서 다양한 정책을 가질 수 있는 마이크로커널의 장점을 지닐 수 있는 것이다.

3. 결론

기존에 제시되었던 마이크로 커널은 하드웨어에 의존적이며, 추가적인 기능 및 정책을 추가하고자 할 때 어려움이 존재하였다. 이러한 문제에 대한 해결방안으로 본 연구에서는 컴포넌트에 기반한 커널 프레임워크를 제시하고 있으며, 실제 M3K 마이크로 커널에 쓰레드 컴포넌트를 구현해 봄으로써 유용성을 확인하고 있다.

향후 연구 과제로는 지금까지 구현된 쓰레드 컴포넌트 이외에 다중 주소 공간을 지원하는 메모리 관리 컴포넌트 등의 구현을 통해 실제 멀티미디어 응용을 실험할 수 있는 환경을 구축하는 것이다

참 고 문 헌

[1] D.R. Engler, M.F. Kaashoek, and J. OToole Jr, "Exokernel and operating system architecture for application-specific resource management" in Proceedings of the 15th ACM Symposium on Operating Systems Principles, pp. 251-266, December, 1995.

[2] M. Acceta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, M. Young, "MACH: A New Kernel Foundation for UNIX Development" in Proceedings of USENIX, summer 1986.

[3] Jochen Liedtke, "On microkernel construction" in Proceedings of the 15th ACM Symposium Operating System Principle(SOSP) (Copper Mountain Report, Colo., Dec. 1995). ACM Press, New York, 1995. pp 237-250.

[4] OS Group, TU Dresden, IBDR, The Dresden Real Time Operating System Project. Online at http://os.inf.tu-dresden.de/project/.

[5] Brian Bershad, Craig Chambers, Susan Eggers, and et. "SPIN an extensible microkernel for application-specific operating system services" Technical Report 94-03-03, Dept. of Comp. Sci. and Eng., University of Washington, Seattle, February, 1994.

[6] H. Tokuda, T. Nakajima, and P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System" in Proceedings of USENIX 1st Mach Workshop, October 1990.

[7] Dan Hildebrand, "An Architectural Overview of QNX" in 1st USENIX Workshop on Micro-kernels and Other Kernel Architectures, pp. 113-126, Seattle, WA, April 1992.

[8] James L. Turley, Advanced 80386 Programming Techniques, McGraw-Hill, 1998.

[9] 양순섭, 고영웅, 조유근, 신현식, 최진영, 유혁, "컴포넌트 기반 커널을 위한 프레임워크" 춘계 정보과학회 학술대회 논문집, 1999.