UbiFOS™ 실시간 운영체제에서 POSIX지원을 위한 래퍼의 설계 및 구현

Design and Implementation of Wrapper to Support POSIX Standards on UbiFOS™ Real-Time Operating System

송예진, 조문행, 이철훈
충남대학교 컴퓨터공학과

Ye-Jin Song(syj3565@cnu.ac.kr), Moon-Haeng Cho(root4567@cnu.ac.kr),
Cheol-Hoon Lee(clee@cnu.ac.kr)

요약

최근의 내장형 시스템은 그 용도에 따라 특정 기능만 수행하는 단순한 응용프로그램을 탑재했던 과거와는 달리 멀티미디어 기능들이 하나로 통합된 디지털 컨버전스 기기로 진화하면서 응용프로그램의 복잡도가 현저히 증가하였다. 또한 응용프로그램은 그 시대의 요구에 따라 여러 응용프로그램들과 통합되고 진화해 간다. 이러한 응용프로그램을 개발하고 관리하기 위해서는 개발자와 관리자 간의 표준화된 인터페이스가 필요하다. 컴퓨팅 시스템에서 개방형 시스템 구조를 갖는 표준 중 운영체제의 인터페이스에 대한 표준으로 POSIX(Portable Operating System Interface)가 개발되었으며, 디지털 컨버전스 기기와 같이 실시간 운영체제 탑재를 요구하는 시스템을 위한 인터페이스 표준으로 POSIX.4계열이 있다. 본 논문에서는 개방형 실시간 운영체제 인터페이스 표준인 POSIX.4 지원을 위한 래퍼(wrapper)를 실시간 운영체제 UbiFOSTM에 설계 및 구현한 내용을 기술한다. 또한, POSIX.4 표준을 준수한 응용프로그램을 Linux와 UbiFOSTM에 각각 탑재하여 비교 실험하고 구현한 래퍼의 성능 오버헤드가 3~9㎲로 미미하다는 측정 결과를 제시한다.

■ 중심어 : | 실시간 운영체제 | 개방형 시스템 구조 | POSIX 표준 | 래퍼 |

Abstract

Recently, Embedded systems are different with the past as loading of a simple application program that executed specific functions according to the use and are evolved in the digital convergence integrated multimedia functions and then the complication of the application program is remarkably increased. This application program is combined and evolved with many application program in accordance with the demand of the age. For develope and manage this developing application is necessary standardized interface between developer and manager. POSIX was developed as the standard of the operating system in the standard interface which has the open system structure in computing system, and there is a posix.4 to standard for the system demands the loading of real-time operating system like a digital convergence devices. In this paper, we present the contents of designing and implementing the real-time operating system UbiFOSTM to wrapper for supporting the POSIX.4. Also, Experimental results show that implemented wrapper to application program standardizing POSIX.4 in linux and UbiFOSTM is slight only 3~9㎲.

■ keyword : | Real-Time Operating Systems | Open System Architecture | POSIX Standards | Wrapper |

* 본 연구는 정보통신부의 선도기반기술개발사업의 지원으로 수행되었습니다.

접수번호 : #070517-001
접수일자 : 2007년 05월 17일
심사완료일 : 2007년 07월 31일
교신저자 : 이철훈, e-mail : clee@cnu.ac.kr

I. 서 론

최근의 컴퓨팅 시스템은 산업 표준으로 제정된 개방형 시스템 구조를 갖는 소프트웨어를 사용하는 추세이다.

이러한 개방형 시스템 구조의 소프트웨어를 통한 응용프로그램 개발이 중요한 이유는 다음과 같다[1].

첫째, 과거 한 명의 개발자가 단순 기능의 소프트웨어를 설계부터 구현까지 도맡았던 시대와 달리, 최근의 소프트웨어는 통신, DMB (Digital Multimedia Broadcasting), 전자사전, 방송수신, 동영상, 음악 재생 그리고 카메라 기능과 같은 여러 기능들이 통합된 복잡한 응용프로그램으로 진화하면서 소프트웨어들 간의 통합과 많은 개발자들이 필요하며 그에 따른 표준화된 개발 인터페이스가 필요하다.

둘째, 소프트웨어 응용프로그램의 생명 주기가 길어지면서, 그에 따른 새로운 기능들의 추가와 수정이 요구되기 때문에 이를 효율적으로 유지 및 관리할 수 있어야 한다.

위와 같은 점을 고려하여 개방형 시스템 구조를 갖는 산업 표준은 많은 분야에서 개발, 제정, 사용되어 진다. 특히, 한정적인 하드웨어를 탑재하여 특정 목적을 수행하는 내장형 시스템의 응용프로그램을 개발하는 과정에서는 개방형 소프트웨어 사용은 필수 불가결한 부분이 되었고[11], 내장형 시스템의 핵심 소프트웨어인 실시간 운영체제(Real-Time Operating System)의 인터페이스 표준에 대한 연구도 오랫동안 진행되어 왔다[2][3].

운영체제의 인터페이스에 대한 산업 표준으로 IEEE와 ISO/IEC에서 제정한 POSIX(Portable Operating System Interface)가 있고, 이는 과거부터 최근까지 가장 오랫동안 사용되고 있는 UNIX 운영체제 기반의 개방형 소프트웨어를 표방하는 운영체제 인터페이스이다[4-10].

POSIX의 주요 목적은 이기종의 하드웨어와 여러 운영체제의 표준 인터페이스를 제정하고, 이를 통해 응용프로그램을 개발함으로써 운영체제 변경 시 응용프로그램의 수정 없이 이식하는 것이다.

본 논문에서는 실시간 운영체제 UbiFOS™에 개방형 운영체제 표준 인터페이스 중 실시간 지원과 관련된 표준인 POSIX.4(POSIX 1003.4)와 쓰레드에 대한 확장 표준인 POSIX.4a를 중점적으로, 이를 지원하기 위한 커널 API(Application Program Interface) 수정이 필요 없는 래퍼를 구현하여 이를 적용시켰다.

본 논문의 구성은 2장에서 관련연구로 POSIX에 대해 상세히 소개하고, 본 논문의 기반 실시간 운영체제인 UbiFOS™에 대해 기술한다. 3장에서는 실시간 운영체제 UbiFOS™에 POSIX 지원을 위한 래퍼의 설계 및 구현 내용을 기술하고, 4장에서는 구현물에 대한 검증을 위해 각각 Linux와 UbiFOS™에서 POSIX 응용프로그램을 시연한 실험 결과에 대해 기술한다. 마지막으로, 5장에서는 결론 및 향후 연구과제에 대해서 기술한다.

II. 관련연구

본 절에서는 개방형 운영체제 인터페이스 표준인 POSIX에 대해 상세히 기술하고, 개방형 운영체제 표준 중 실시간 지원을 위한 POSIX.4 계열을 탑재할 대상인 연성 실시간 운영체제 UbiFOS™을 소개한다.

1. 개방형 운영체제 인터페이스 표준 POSIX

POSIX는 운영체제의 다양한 특성에 따라 여러 표준 단체들이 각 특성에 맞는 표준을 제정한 표준들의 집합으로, 이미 승인되어 표준으로 제정된 것들과 개발이 진행 중인 것이 있다. 이런 POSIX는 다음 세 가지로 분류할 수 있다[4-10].

1.1 기본 표준

기본 표준은 운영체제의 특성에 따라 분류된 POSIX 표준들이다. [표 1]과 같은 각 표준을 통해 개발된 응용프로그램들은 그 해당 표준을 지원하는 어느 플랫폼에서도 구동된다.

표 1. POSIX 기본 표준

구 분	설 명
POSIX.1	System Interface (basic reference standard)[ab]
POSIX.2	Shell and Utilities[a]
POSIX.3	Methods for Testing Conformance to POSIX[a]
POSIX.4	Real-Time Extensions
POSIX.4a	Threads Extensions
POSIX.4b	Additional Real-Time Extensions
POSIX.6	Security Extensions
POSIX.7	System Administration
POSIX.8	Transparent File Access
POSIX.12	Protocol Independent Network Interfaces
POSIX.15	Batch Queuing Extensions
POSIX.17	Directory Service
a Approved IEEE standards b Appoved ISO/IEC standard	

1.2 언어 지원

[표 2]의 언어 지원에 따른 표준은 서로 다른 프로그래밍 언어에 대한 인터페이스를 제공한다.

표 2. POSIX 언어 지원

구 분	설 명
POSIX.5	Ada Bindings[a]
POSIX.9	Fortran 77 Bindings[a]
POSIX.16	C Language Bindings
POSIX.19	Fortran 90 Bindings
POSIX.20	Ada Bindings to Real-Time Extensions
a Approved IEEE standards	

1.3 개방형 시스템 환경

[표 3]의 개방형 시스템 환경에 대한 표준은 POSIX 환경과 응용프로그램을 위한 프로파일들에 대한 것이다.

표 3. POSIX 응용 개발 표준

구 분	설 명
POSIX.0	Guide to POSIX Open System Environment
POSIX.10	Supercomputing Application Environment Profile
POSIX.11	Transaction Processing Application Environment Profile
POSIX.13	Real-Time Application Environment Profile
POSIX.14	Multiprocessing Application Environment Profile
POSIX.18	POSIX Platform Application Environment Profile

POSIX 중 본 논문의 구현대상이 되는 POSIX.4 계열은 POSIX.1의 실시간성 지원과 관련하여 추가된 것으로 그 하위 이름에 따라 다음과 같은 특징을 갖는다[6-9].

- POSIX.4 : POSIX.1에 기본적인 실시간성 지원
  - 동기화, 비동기화 I/O
  - 세마포(Semaphore)
  - 메모리 락킹(Memory Locking)
  - 공유 메모리(Shared Memory)
  - 실행 스케줄링 지원(Priority, Round-Robin)
  - 클럭과 타이머(Clocks and Timers)
  - 메시지 패싱(Message Passing)
- POSIX.4a : 강화된 실시간성 지원
  - 쓰레드 관리(Thread Management)
  - 시그널(Signals)
  - 프로세스 스케줄링(Process Scheduling)
  - 조건 변수(Condition variables)
  - 쓰레드 스케줄링(Thread Scheduling)
  - 쓰레드 안전 재진입 함수(Thread-safe reentrant functions)
- POSIX.4b : 보다 강화된 실시간성 지원
  - 프로세스 스폰(Process spawn)
  - 블록된 함수의 시간제한(Time-outs on blocking functions)
  - 실행시간 모니터링(Execution time monitoring)
  - 산발 서버 스케줄링(Sporadic Server Scheduling)
  - 인터럽트와 장치 관리(Interrupt and Device Control)

1.4 POSIX.4를 지원하는 실시간 운영체제

실시간 운영체제의 POSIX 지원은 표준 인터페이스를 통해 응용프로그래머에게 편의성을 제공하고 기존의 응용에 대한 호환성을 제공하기 위한 것이다. 대부분의 상용 실시간 운영체제에서는 POSIX를 지원하고 있다[16]. 상용 실시간 운영체제 중에서 VxWorks, QNX, Nucleus[17-19] 등은 POSIX 지원을 위한 래퍼를 두는 방식을 사용하고 있으며, NanoQplus[20]는

POSIX API 형태로 운영체제 기본 API를 구현하였다. 전자가 래퍼로 인한 성능상의 오버헤드가 존재하는 반면 운영체제의 수정이 필요 없다는 장점이 있다.

2. 실시간 운영체제 UbiFOS™

그림 1의 실시간 운영체제 UbiFOS™은 충남대학교 시스템소프트웨어 연구실에서 DVD player, 핸드폰, PDA, 프린터와 라우터 등의 내장형 시스템에 탑재하기 위해 개발된 우선순위 기반의 실시간 운영체제이다. 멀티태스킹 지원을 위해 0부터 255까지 256단계의 우선순위를 제공하는 선점형 스케줄러로 동일 우선순위에 대해 FIFO와 Round-Roubin 기법을 제공한다[13].

이 밖에도 태스크간의 통신을 위해 메시지 메일박스(Message Mailbox), 메시지 큐(Message Queue), 메시지 포트(Message Port), 태스크 포트(Task Port), 시그널을 제공하고 태스크간의 동기화를 위해 세마포와 이벤트 플래그(Event Flag)를 제공하고 있다.

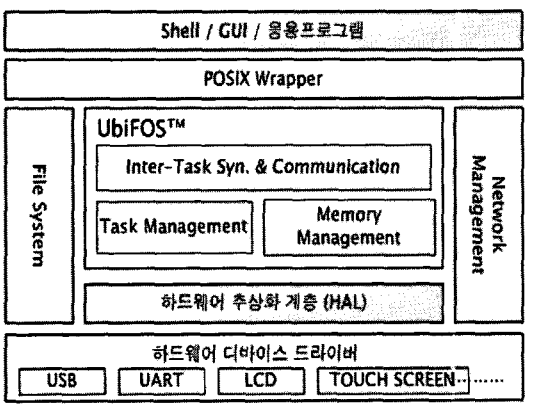

그림 1. UbiFOS™ 전체 구성도

2.1 태스크 관리

일반적으로 응용프로그램은 독립적인 여러 개의 프로그램으로 구성된다. 이러한 독립적인 프로그램 각각을 태스크라 한다. UbiFOS™은 실시간 시스템의 시간 제약사항을 해결하기 위해 시간 결정적인 스케줄링 기법[15]을 제공하며, 멀티태스크 모델이므로 시스템의 자원을 서로 공유하여 사용한다. 또한, 태스크는 자신만의 TCB(Task Control Block)를 갖고 있으며, 그 외에 스택, 우선순위, 실행함수 등을 포함하고 있다. 따라서 태스크는 스케줄링을 통해 스스로 실행할 수 있지만, 함수는 태스크에 의해 종속적이다.

2.2 세마포

세마포는 상호배제, 태스크 간 동기화라는 두가지 요구사항을 충족시키는 중요한 의미를 갖는다.

UbiFOS™은 동기화와 상호배제를 위한 바이너리 세마포와 다수의 공유 자원을 관리하기 위한 카운팅 세마포를 제공한다.

2.3 태스크간 통신 – Message Queue

UbiFOS™은 독립적인 태스크들 사이에 정보를 주고받기 위해서 태스크 사이의 통신기능을 제공한다. 메시지 큐는 이러한 태스크들 사이의 통신을 위한 기능 중에 하나로, 태스크 또는 ISR(Interrupt Service Routine)이 여러 개의 메시지를 다른 태스크로 전달하려 할 때 사용한다. UbiFOS™은 메시지 큐를 통해 전달되는 메시지의 크기의 가변성에 따라 가변길이 메시지 큐와 고정길이 메시지 큐를 제공하며 메시지는 포인터의 전달이 아닌 복사에 의한 전달 방법을 사용한다.

2.4 시그널

UbiFOS™에서 지원하는 시그널은 하나 이상의 태스크들에게 비동기적인 이벤트를 알리기 위해 사용된다. 시그널은 비동기적인 이벤트라는 점에서는 인터럽트와 비슷하지만, 인터럽트는 현재 수행중인(RUNNING) 태스크에 비동기적인 이벤트를 발생하지만, 시그널은 실행 대기(READY) 상태의 태스크뿐만 아니라 심지어 BLOCK상태와 같이 특정 이벤트를 기다리는 태스크들에게도 시그널 이벤트를 발생한다.

2.5 타이머

UbiFOS™의 타이머는 태스크에게 특정 시간이 지났음을 알리기 위한 모듈로 타이머에서 사용되는 시간의 단위는 틱(Tick)이다. 하드웨어 타이머에 의해 호출되는 주기적인 인터럽트 처리 루틴을 통해 틱 값을 감소시키고 주어진 틱 값이 다 소모되어 0이 되면 응용프로

그램 작성자가 정의한 특정 루틴(Expiration Routine)을 수행한다.

III. POSIX 지원을 위한 래퍼의 설계 및 구현

본 논문은 실시간 운영체제 UbiFOS™의 개방형 운영체제 인터페이스 지원을 위해 운영체제 인터페이스 표준인 POSIX 중 실시간 운영체제에서 지원해야 할 대상을 중심으로 래퍼를 설계 및 구현하였다.

POSIX 중 POSIX.4 계열이 실시간 운영체제와 관련된 운영체제 인터페이스 표준으로, 이를 실시간 운영체제 UbiFOS™에 적용하기 위해 다음의 특성을 고려하였다.

- 멀티태스킹 구조로 프로세스 관련 기능은 제공하지 않으며, 공유 메모리를 지원한다.
- MMU 탑재 유무에 영향을 받지 않는 메모리 관리 기법으로 동적 메모리 할당 기법을 제공하고 가상 메모리는 제공하지 않는다.
- 고정 우선순위 방식의 연성 실시간 스케줄러로 경성 실시간 스케줄러는 지원하지 않는다.

POSIX는 [그림 2]에서와 같이 하나의 Process 안에 여러 개의 쓰레드가 존재하는 다중 쓰레드 프로세스(Multithreaded Process) 구조로 되어 있고, 하나의 쓰레드마다 자신만의 문맥을 저장하는 메모리 공간인 레지스터 (register)와 스택 (stack)영역을 가지며, 하나의 Process 안에 존재하는 쓰레드들은 코드, 데이터, 파일 영역을 공유한다[12].

그리고 UbiFOS™은 멀티태스킹 구조이다. 멀티태스킹이란 하나의 CPU에서 여러 개의 태스크가 동시에 수행될 수 있도록 하는 방식이며, 각 태스크는 자신의 TCB(Task control block)에 PC(Program Counter), 레지스터, 스택, 실행함수 (function call), 우선순위 (priority), 시그널 핸들러(Signal Handler) 등을 저장한다.

본 논문에서는 POSIX 표준 중 UbiFOS™의 특성을 고려하여, 실시간 지원을 위한 쓰레드, 세마포, 뮤텍스 (Mutex), 메시지 큐, 시그널, 타이머와 관련된 표준 POSIX API와 UbiFOS™이 제공하는 API가 호환 가능하도록 래퍼를 설계 및 구현하였다.

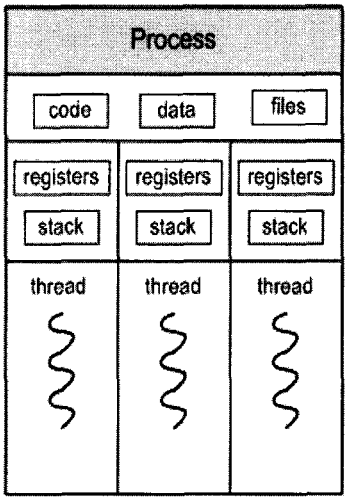

그림 2. Multithreaded Process 구조

1. 쓰레드

1.1 pthread

쓰레드는 세미 프로세스라고 불리며 POSIX의 쓰레드는 흔히 pthread라고 한다. 쓰레드는 커널에 의해 시간이 분할되어 실행되고, 같은 메모리 공간을 공유한다.

UbiFOS™에서 태스크의 구조체 제어블럭을 쓰레드의 구조체 제어블럭으로 제어하여, POSIX 쓰레드와 동일한 기능을 수행하는 래퍼를 설계 및 구현하였다.

[표 4]의 pthread_create()함수와 같이 쓰레드를 지원하기 위한 함수들을 UbiFOS™의 태스크 관련 함수들과 래핑하여, [그림 3]과 같이 POSIX 쓰레드 함수와 동일한 기능을 지원할 수 있도록 하는 래퍼를 설계 및 구현하였다. UbiFOS™에서는 태스크 생성과 실행을 구분해서 관리하지만 POSIX에서는 쓰레드 생성시 제어권이 바로 생성된 쓰레드로 천이 되므로, UbiFOS™의 태스크 생성 함수 에서도 동일한 기능을 지원하기 위해 생성과 실행을 통합한 MK_CreateTask()함수를 구현하였다.

표 4. 쓰레드 관련 함수의 래핑

POSIX 함수	UbiFOS™ 함수	설 명
pthread_create	MK_CreateTask	Thread를 생성하는 함수
pthread_join	MK_Join	해당 Thread가 종료할 때까지 기다리고 자원을 해제
pthread_detach	MK_Detach	Process에서 생성된 Thread를 분리시키는 함수
pthread_exit	MK_DeleteTask	Thread를 종료하는 함수
pthread_self	MK_Get CurrentTask	현재 수행중인 Thread의 식별자를 얻는 함수

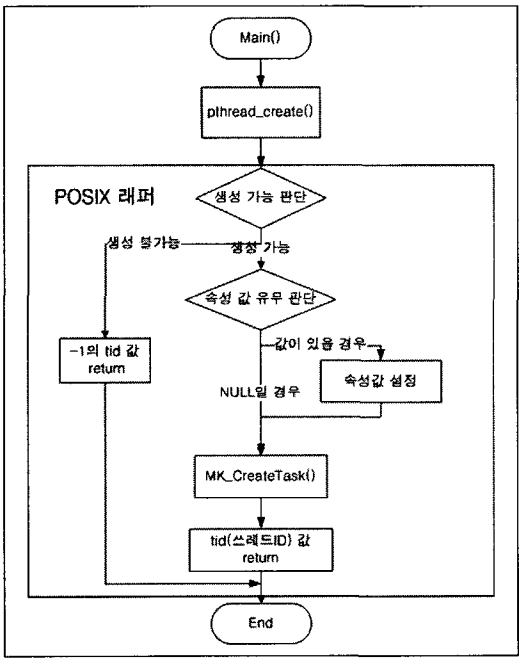

그림 3. POSIX 쓰레드 래퍼 구조

1.2 쓰레드의 속성(attribute)

속성객체 변수는 스케줄링 정책, 스택 주소, 스택 크기, 우선순위와 같은 스택 제어관련 필드를 포함한 자료구조로, NULL값을 지정하면 기본 값들이 설정되고 속성 제어함수를 통해 각 값들을 변경할 수 있다.

1.3 쓰레드의 동기화

POSIX 쓰레드의 동기화는 세마포와 뮤텍스를 통하여 수행되는데, 세마포는 한 공유 대상의 자원이 여러 개일 경우 관리하는 카운팅 세마포(Counting Semaphore)와 뮤텍스 처럼 한 공유 대상의 자원이 하나일 경우 이를 관리하는 바이너리 세마포(Binary Semaphore)가 있다.

뮤텍스는 여러 개의 쓰레드가 공유하는 코드 영역을 단지 한번에 하나의 쓰레드만 실행 가능하도록 하는 방법으로 상호배제를 지원한다.

UbiFOS™에서는 카운팅 세마포와 바이너리 세마포를 별도로 구분하지 않고, 세마포의 초기 값이 양수이면 카운팅 세마포로 1이면 바이너리 세마포로 간주하여 동기화와 상호배제를 제공한다.

또한, pthread_mutex_lock과 pthread_mutex_trylock 함수는 MK_SemaphorePend함수에 옵션을 사용하여 같은 기능을 수행할 수 있도록 래퍼를 구현하였다.

1.4 쓰레드의 취소

쓰레드의 취소는 하나의 프로세스 안에서 실행 중인 다른 쓰레드를 종료시키기 위한 목적으로 사용된다. 취소 요청을 받은 쓰레드는 설정에 따라서 곧 바로 종료할 수도 있고, 취소 지점 (cancellation point)을 벗어난 후 종료 할 수도 있다.

1.5 실시간성 지원을 위한 쓰레드의 스케줄링

멀티 프로세스 모델을 기본으로 하는 POSIX.4는 실시간 프로세스 스케줄링을 위하여 다음 3가지 정책을 설정할 수 있도록 한다.

- SCHED_FIFO : 고정 우선순위 기반의 선점형 스케줄링 기법으로, 동일한 우선순위의 프로세스들은 FIFO(First-In-First-Out)로 스케줄링 한다.
- SCHED_RR : 스케줄링은 FIFO와 유사하며, 동일 우선순위의 프로세스들은 시간분할(time-sliced) 기법으로 스케줄링 한다.
- SCHED_OTHER : 추가 구현 가능한 스케줄링 기법으로, EDF(Earliest Deadline First)와 RM(Rate Monotonic)이 있다.

우선순위를 갖는 멀티쓰레드 모델 기반의 실시간 운영체제인 UbiFOS™에서는 실시간 성 지원을 위해 기본적으로 쓰레드 간 선점을 허용하고 있으며, 동일 우

선순위의 쓰레드 간에는 SCHED_FIFO, SCHED_RR을 제공하고 있다. 본 논문에서 구현한 래퍼는 쓰레드 속성 값을 설정하는 함수를 통해 스케줄링 정책을 설정/변경하여 실시간 성을 지원한다. 그러나 연성 실시간 운영체제가 갖는 고정우선순위 스케줄링 기법만 제공하고 경성 실시간 운영체제 스케줄링 기법인 EDF나 RM은 설정할 수 없다.

2. 프로세스간 통신 - Message Queue

메시지 큐는 프로세스간 혹은 쓰레드간 통신을 위한 기능 중 하나로, 쓰레드에서 다른 쓰레드로 여러 개의 메시지를 전달하려 할 때 사용한다. 멀티쓰레드 모델인 UbiFOS™에서는 실시간 성 지원을 위해 쓰레드 간 메시지 송수신에 대한 대기 시간(Timeout) 값을 설정하여 특정 시간만 메시지 자원을 대기하도록 한다. 이에 본 구현 래퍼에서는 이 기능을 이용하여 기본 대기 시간 값을 설정하여 실시간 성을 지원하도록 하였다. 메시지 큐 생성은 POSIX 의 경우 파일 제어형태의 API를 사용하는 방식인데 반해 UbiFOS™에서는 메시지 큐 관리 블록을 통해 이루어지는 방식으로 10개의 메시지 큐 관리 블록을 생성할 수 있도록 제한하였으며, 메시지 큐 관리 블록에 대한 구분은 mqd_t타입 값을 관리 블록 배열의 인덱스로 사용하였다.

[표 5]는 메시지 큐와 관련된 래퍼 함수들이다.

표 5. 메시지 큐 관련 함수의 래핑

POSIX 함수	UbiFOS™ 함수	설 명
mq_open	MK_Create MsgQueue	Message Queue를 생성하는 함수
mq_close	MK_Delete MsgQueue	Message Queue를 제거하는 함수
mq_receive	MK_MsgQueuePost	Message Queue로부터 Message를 받는 함수
mq_send	MK_MsgQueuePend	Message Queue에 Message를 전달하는 함수

3. 실시간 시그널(Real-Time Signals)

시그널은 하나 이상의 쓰레드에게 이벤트 발생을 알리기 위해서 사용되어 지는데, POSIX.4에서는 아래와 같은 특징을 갖는다.

- 실시간 시그널은 큐잉(Queuing)이 가능해야 한다.
- 실시간 시그널 사이에는 우선순위가 존재하며, 실시간 성 지원을 위해 그 우선순위에 따라 처리 가능해야 한다.
- 실시간 시그널은 시그널 발생 함수와 시그널 처리 함수의 자료 교환이 가능해야 한다.

UbiFOS™에서 지원하는 시그널은 큐잉이 가능하며, 큐잉된 시그널 사이에는 우선순위가 존재한다. UbiFOS™의 시그널 기능을 실시간 시그널 지원을 위해 직접 사용이 가능하다.

다음은 시그널과 관련된 래퍼 함수들이다.

[표 6]과 같이 pthread_sigmask 와 pthread_kill 함수는 MK_SignalSend 함수에 옵션을 사용하여 해당 기능에 따른 정의 값을 두고 그 값에 따라 수행하도록 구현하였다.

표 6. Signal 관련 함수의 래핑

POSIX 함수	UbiFOS™ 함수	설 명
pthread _sigmmask	MK_SetEvent	특정 Thread만 Signal을 받도록 하는 함수
pthread_kill	MK_SetEvent	Thread로 해당 번호의 Signal을 전달하는 함수
sigwait	MK_EventPend	Signal의 전달을 동기적으로 기다리는 함수

4. 클럭(Clocks)고 시간제한(Timeouts)

UbiFOS™에서는 실시간 성 지원을 위해 모든 자원에 대한 요청 시 시간제한 값을 설정할 수 있도록 구성되어 있으며, 클럭 인터럽트를 처리하는 인터럽트 처리 루틴에서 이를 처리한다.

클럭은 하드웨어의 타이머 인터럽트를 설정하여 특정 주기로 발생하는 인터럽트로 실시간 운영체제의 심장 역할을 수행한다. 시간제한은 실시간 성 지원을 위해 자원에 대한 대기 시간을 한정하기 위한 기능이다. 시간제한 정책에서 설정된 값의 경과를 이 클럭을 처리하는 클럭 인터럽트 처리 함수를 통해 알 수 있으며, 실시간 성 지원을 위해 상위 우선순위의 태스크가 실행

준비 상태인지 파악하는 것도 클럭 인터럽트를 처리하는 인터럽트 처리 함수(Interrupt Service Routine)에서 수행한다.

POSIX.4b에서는 자원에 대한 시간제한 값을 설정하여 자원을 기다리는 쓰레드가 무한정 기다리는 것을 방지하며, 이에 대한 처리는 클럭 인터럽트를 통해 이루어진다.

POSIX.4b에서 시간제한 값을 설정하는 경우는 다음과 같다.

- 언블록된 세마포 자원에 대한 대기 시
- 메시지 큐에서 메시지를 요청할 경우
- 메시지 큐에 메시지를 전송할 경우
- 언블록된 뮤텍스 자원에 대한 대기 시

Ⅳ. 실험 환경 및 결과

본 논문에서 설계 및 구현한 개방형 운영체제 표준을 지원하기 위한 POSIX 래퍼의 실험은 ARM920T 기반의 S3C2440 MCU가 탑재된 MBA2440 보드[13]에서 수행하였다. 개발도구로는 ARM Developer Suit v1.2, 디버거로는 OPENice-A1000 Emulator를 사용하였다.

실험에 사용된 예제는 공개된 POSIX 응용프로그램[14]으로 본 논문의 타겟보드와 Linux 버전에서 수행하여 같은 결과가 나오는지를 비교 실험하였다. 또한, 본 논문에서 구현한 래퍼로 인해 시간 결정성에 미치는 영향을 분석하기 위해 타켓보드의 LED 하드웨어와 오실러스코프를 이용하여 성능 오버헤드를 μs 단위까지 측정하였다.

[그림 4]에서 왼쪽은 Linux에서 실험한 결과이고, 오른쪽은 본 논문에서 구현한 UbiFOS™의 POSIX래퍼가 탑재된 MBA2440보드에서 UART를 통해 수행한 결과를 나타낸 것이다. 두 개의 쓰레드(쓰레드 tid 1과 2인)를 생성하여 각 쓰레드가 특정함수를 수행하는 것으로, 쓰레드를 생성하는 pthread_create()로 두 개의 쓰레드를 생성하고 연속적인 함수 실행을 위해 해당 쓰레드를 조인시킨 pthread_join() 함수로 구성되어 있다. 쓰레드 tid1과 tid2는 1씩 증가되는 특정 값을 출력하고 1tick 동안 sleep하는 함수를 수행한다. 만약, 함수 수행 후 조인하지 않으면 함수 실행이 종료되지만, 두 함수 모두 조인시켜 1tick 후에 쓰레드가 깨어나 함수를 재실행하도록 하였으며. 출력 값이 10이 되는 경우 리눅스에서는 Ctrl+C를 통해 수행을 종료하여 UbiFOS™에서 실행되는 값과 비교하였다.

그림 4. POSIX 쓰레드 생성 실험 결과

이를 통해 Linux와 UbiFOS™에서 같은 결과를 얻는 실험으로 기능 시험을 수행하였다. 래퍼에 대한 기능 시험에 사용된 다른 예제는 [14]에서 제시한 POSIX 응용프로그램 예제를 [그림 4]에서와 같이 Linux와 UbiFOS™에서 실험하였고, 본 논문에서는 쓰레드 생성과 조인에 대한 실험결과만 포함하였다.

본 논문에서 구현한 래퍼에 대한 실행 오버헤드의 측정은 래퍼의 코드 구간에 LED On과 LED Off를 설정하고, 텍트로닉스(Tektronix)의 TDS3054B 오실러스코프를 이용하여 LED On 시점의 시간과 Off 시점의 시간의 변이를 측정하는 방법으로 수행하였다.

[표 7]은 UbiFOS™에서 POSIX 지원을 위해 구현한 래퍼의 성능 상 오버헤드를 측정한 값으로, 쓰레드 생성시는 [그림 3]에서와 같이 생성 가능여부와 속성 값 설정여부, 설정 값에 따른 태스크 생성 오버헤드로 다른 모듈보다 대략 5μs의 추가 시간을 요구한다. 본 논문

에서 구현한 래퍼의 성능 오버헤드는 각 모듈별 3~9㎲로 미미하다 할 수 있다. 하지만 이런 성능상의 오버헤드는 본 UbiFOS™의 시간 결정성을 지원하는 스케줄링 방식[15]을 통해 보완이 가능하다.

표 7. UbiFOS™의 래퍼로 인한 모듈별 성능 오버헤드

Description	POSIX Function used	UbiFOS™ (ARM9 400MHz)
Thread	pthread_create()	8.74
Semaphore	sem_init()	3.15
MessageQueue	mq_open()	2.94
Signals	pthread_sigmask()	3.08
Timers	timer_create()	3.27

V. 결 론

최근의 내장형 시스템은 그 용도에 따라 특정 기능만 수행하는 단순한 응용프로그램을 탑재했던 과거와 달리 멀티미디어 기능들이 하나로 통합된 디지털 컨버전스 기기로 진화하면서 특정 목적에 따라서 다양한 형태로 개발되고 발전되어 가고 있다. 이에 따라 내장형 시스템에 탑재되는 응용프로그램도 장치에 맞게 다양화되고 있으며, 복잡도가 현저히 증가하였다. 이런 응용프로그램은 한 번의 개발로 끝나는 것이 아니고 그 시대의 요구에 따라 다양하게 통합되고 진화해 간다. 따라서 재사용성이 뛰어난 응용프로그램이 필요하고, 이를 위해 개발 시 표준 API를 사용하는 것이 바람직하며 플랫폼 구동의 기반이 되는 운영체제는 표준 API를 제공해야 한다.

컴퓨팅 시스템에서 개방형 시스템 구조를 갖는 인터페이스 표준 중 운영체제에 대한 인터페이스 표준으로 POSIX가 개발되었으며, 디지털 컨버전스 기기와 같이 실시간 운영체제 탑재를 요구하는 시스템을 위한 표준으로 POSIX.4계열이 있다.

본 논문에서는 개방형 실시간 운영체제 표준인 POSIX.4 지원을 위한 래퍼를 실시간 운영체제 UbiFOS™에 설계 및 구현하였다. 본 논문에서 구현한 래퍼에 대한 기능 실험을 통해 성능 검증을 하였으며, 성능상의 오버헤드 측정 결과 래퍼로 인한 성능 오버헤드는 3~9㎲로 미미하였다. 또한, UbiFOS™의 POSIX 표준 지원을 통해 기존 POSIX API로 개발된 응용프로그램은 추가적인 작업없이 탑재가 가능하게 되었다.

향후 연구과제로는 본 논문에서 설계 및 구현한 POSIX.4 계열의 표준 외의 추가적인 표준을 본 논문의 기반이 되는 실시간 운영체제 UbiFOS™에 적용하는 것이다.

참 고 문 헌

[1] http:\\www.embedded.com

[2] http:\\www.partow.net

[3] http:\\ecos.sourceware.org

[4] ISO/IEC Standard 9945-1:1990 and IEEE Standard 1003.1-1990, "Information Technology - Portable Operating System Interface (POSIX) - Part 1:System Application Program Interface (API) [C Language]," The Institute of Electrical and Electronics Engineers, 1990.

[5] B. O. Gallmeister and C. Lanier, "Early Experience with POSIX 1003.4 and POSIX 1003.4a," Proceedings of IEEE Real-Time Systems Symposium, pp.190-198, 1991.

[6] IEEE Standards Project P1003.4, "Draft Standard for Information Technology - Portable Operating System Interface(POSIX) - Part 1: System Application Program Interface (API) - Amendment 1: Realtime Extension [C Language]," Draft13, The Institute of Electrical and Electronics Engineers, 1992.

[7] IEEE Standards Project P1003.4a, "Threads Extension for Portable Operating Systems," Draft6, The Institute of Electrical and Electronics Engineers, 1992.

[8] IEEE Standards Project P1003.4b, "Draft Standard for Information Technology -

Portable Operating System Interface(POSIX) - Part 1: Realtime System API Extension," Draft6, The Institute of Electrical and Electronics Engineers, 1993.

[9] IEEE Standards Project P1003.13, "Draft Standard for Information Technology - Standardized Application Environment Profile - POSIX Realtime Application Support(AEP)," Draft5, The Institute of Electrical and Electronics Engineers, 1992.

[10] http:\\standards.ieee.org/catalog/posix.html

[11] D. E. Simon, *An Embedded Software Primer*, Addson-Wesley, 1994.

[12] S. Galvin, *Operating Systems Concepts*, Addson-Wesley, 1994.

[13] http://www.aijisystem.com.

[14] http://www.joinc.co.kr/modules/moniwiki/wiki.php/article/Pthread_API_Reference.

[15] S. J. Oh, et al., "Deterministic Task Scheduling for Embedded Real-Time Operating Systems," IEICE Trans. Inf.&Syst., Vol.E87-D, No.2, pp.123-126, 2004.

[16] http://www.realtime-info.be

[17] http://www.windriver.com

[18] http://www.qnx.com

[19] http://www.atinucleus.com

[20] http://www.qplus.or.kr

저 자 소 개

송 예 진(Ye-Jin Song) 준회원

- 2006년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2006년 3월 ~ 현재 : 충남대학교 컴퓨터공학과(공학석사)

조 문 행(Moon-Haeng Cho) 정회원

- 2004년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2006년 2월 : 충남대학교 컴퓨터공학과(공학석사)
- 2006년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 박사과정 재학

<관심분야> : 실시간 컴퓨팅, 실시간 운영체제, 초소형 초절전 실시간 운영체제

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 2월 : 서울대학교 전자공학과(공학사)
- 1988년 2월 : 한국과학기술원 전기및전자공학과(공학석사)
- 1992년 2월 : 한국과학기술원 전기및전자공학과 (공학박사)
- 1983년 3월 ~ 1986년 2월 : 삼성전자 컴퓨터사업부 연구원
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터사업부 선임연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원연구원
- 1995년 2월 ~ 현재 : 충남대학교 컴퓨터공학과 교수
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 초빙연구원

<관심분야> : 실시간시스템, 운영체제, 고장허용 컴퓨팅