특 집

임베디드 운영체제 표준화 동향

김 선 자*, 김 흥 남**, 김 채 규**

● 목 차 ●

1. 서 론

임베디드 시스템의 규모가 커지고 복잡하고 다양한 기능을 수행하게 됨에 따라 그 소프트웨어도 간단한 제어용 프로그램만으로는 충분하지 않게 되었으며 운영체제가 필요하게 되었다. 임베디드 시스템의 특성상 초기에는 실시간 처리 요소를 갖는 다중처리(multi-tasking)가 가능한 상용 실시간 운영체제가 주로 사용되었다. pSOS[2], VxWorks[3], VRTX[4] 등 이러한 상용 운영체제는 실시간 처리 기능을 제공하는 데 목적을 두고 신뢰성에 중점을 두며, 범용성이 아닌 주로 한 가지 특수한 목적에 최적화된 시스템 운영체제이다. 그러나 최근 임베디드 시스템에 네트워크 접속, 멀티미디어 처리 등의 기능이 요구됨에 따라 운영체제도 점차 범용 운영체제의 기능에 가까워지고 있다[1]. 반면 하드웨어에 내장되어(embedded) 동작하는 특성상 자원의 한계 및 실행 환경의 특수성을 고려하여 기능을 확장하여야 한다.

또한 IDC가 2004년 인터넷 기기(Internet Appliances) 시장이 89백만달러에 이를 것으로 예측[2]하는 바와 같이 인터넷의 급속한 보급과 무선 통신의 활성화로 임베디드 시스템의 적용 분야가 확산되고 있다. 새로운 시스템의 개발은 새로운 응용 프로그램의 개발을 필요로 하고 있으며 점차 응용 프로그램의 개발이 임베디드 시스템의 성공 여부를 결정짓는 중요한 요소가 되고 있다. 따라서 응용 계층에 임베디드 운영체제의 호환성 지원을 위한 API 표준화 문제가 대두하게 되었다.

임베디드 운영체제 표준화의 또 다른 관점은 기능의 확장성이다. 즉 각 임베디드 시스템의 적용 분야에 따라 응용에 필요한 기능을 확장할 수 있는 기본 구조를 제시하는 방향이 요구되고 있다.

이러한 관점에서 본 고에서는 임베디드 운영체제의 표준화를 크게 두 가지 방향으로 살펴 보고자 한다. 첫째는 임베디드 시스템의 표준 API를 정의하는 것이고 두 번째는 표준 API들을 임베디드 시스템 규모별로 분류하는 것이다. 전자로는 POSIX Real Time API를 들 수 있으며 후자로는 POSIX의 Embedded System Profile 표준을 들 수 있다. EL/IX는 두 가지 방향을 모두 제시한다. 먼저 2절에서는 POSIX Embedded System Profile에 대해 기술하고 3절에서는 POSIX의 Real Time API, 4절에서는

* ETRI 컴퓨터 · 소프트웨어기술연구소 선임연구원
** ETRI 컴퓨터 · 소프트웨어기술연구소 책임연구원

EL/IX에 대해 설명한다. 마지막으로 5절에서 국내외 표준화 동향에 대해 간략히 기술하고 결론으로 맺는다.

2. POSIX 1003.13

1998년 IEEE에서는 응용에 대한 실시간 지원 환경(Realtime Envorinments Profile)에 대한 표준 규격 POSIX(Portable Operating System Interface for Computer Environments) 1003.13을 발표[6]하였다. 이 표준의 공식 명칭은 Standard for Information Technology - Standardized Application Environment Profile- POSIX Realtime Application Support이다. 약어로 POSIX/RT_AEP라 표시한다.

POSIX/RT_AEP 표준의 목적은 실시간 시스템 개발자와 실시간 응용 소프트웨어 개발자에게 POSIX 1003.1 즉 ISO/IEC 9945 표준 시리즈에 기반한 실시간 응용 환경 규격을 제공하기 위한 것이다. 즉 실시간 응용 프로그램이 이식성(portability)을 가지기 위해 필요한 응용 환경 규격(application environment profiles)을 제공한다. 기존의 소규모 실시간 커널 및 POSIX 호환의 실시간 커널을 참조하였으며 현존하는 실시간 시스템의 중요한 기능을 제공하기 위한 POSIX 기반의 인터페이스를 명시한다. 4가지 규격(profile)으로 구성되며 각 규격마다 요구되는 최소의 하드웨어 규격이 명시되었다. 이 규격에 의한 실시간 시스템 판매자는 특정 응용에 불필요한 부분은 구성에서 제외시키는 방법을 제공할 것을 권고한다.

이 표준 규격은 IEEE의 개방형 시스템 규격(Open System Environment, 이하 OSE) 분류에서 응용 환경 규격(Application Environment Profile, 이하 AEP)내 시스템 규격(System Profiles, 이하 PS) 내 일반 환경 규격(Generic Environment Profiles, 이하 PSE) 산하 실시간 환경 규격(Realtime Environments, 이하 PSE5)으로 정의된다. PSE5는 정해진 응답 시간을 요구하는 응용을 지원하기 위해 작성된 규격이며 PSE51, PSE52, PSE53 및 PSE54 의 4가지 규격으로 구성된다. (그림 1)은 PSE5 규격의 포함 관계를 나타낸다.

2.1 PSE51 : Minimal Realtime System Profile

메모리가 장착된 하나의 처리기(processor)가 있고 메모리 관리 유니트(Memory Management Unit, 이하 MMU)나 공통의 입출력 디바이스는 요구되지 않는 하드웨어 모델을 기반으로 한다. 입출력 디바이스가 장착될 수 있는 특정 목적의 전용 시스템(dedicated system)에 내장(embedded)되는 규격이다.

사용자와의 상호 작용(interaction)이나 파일 시스템은 필요로 하지 않는 시스템을 위한 규격이다. 프로그래밍 모델로는 하나의 POSIX 프로세스를 기반으로 하며 이 프로세스는 하나 또는 여러 개의 쓰레드를 가질 수 있다. 프로세스는 하나이나 쓰레드간 또는 또 다른 PSE5 내 4가지 규격을 만족하는 시스템과의 통신을 위해 메시지 전달 인터페이스(message passing interface)가 제공된다.

특별한 디바이스들이 메모리 맵 I/O나 기본 I/O 인터페이스에 의해 작동되고 제어되며 이러한 I/O 인터페이스는 본질적으로 비표준 I/O 하드웨어와 이식성이 없는 제어 코드에 대한 접근을 위한 단일 표준 방식을 제공한다.

2.2 PSE52 : Realtime Controller System Profile

PSE51에 파일 시스템 인터페이스와 비동기(asynchronous, non-blocking) 입출력 인터페이스를 추가한 규격이다. 단일 프로세스 및 메모리를 가지는 하드웨어 모델을 정의하며 여전히 MMU는 제외한다. 파일 시스템 인터페이스는 포함되어 있지만 대용량 디바이스는 포함되어 있지 않다. 이는 램디스크와 같이 메모리에 파일 시스템을 구현할 수도

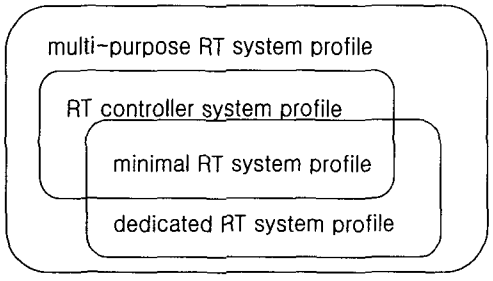
있기 때문이다.

2.3 PSE53 : Dedicated Realtime System Profile

다중 프로세스(multiple process) 모델을 가지며 디바이스 드라이버와 파일에 대한 공통의 인터페이스는 포함되나 계층적 구조의 파일 시스템(hierarchical file system)은 포함되지 않는다. 하나 또는 다수의 프로세서가 있고 MMU가 장착될 수도 있는 하드웨어 모델을 갖는다. MMU가 장착될 수도 있으므로 메모리 잠금(memory locking) 기능이 제공된다.

2.4 PSE54 : Multi-Purpose Realtime System Profile

앞서의 3가지 규격의 기능을 모두 포함하는 규격이며 포괄적인 기능을 제공한다. 서로 다른 실시간 태스크와 비실시간 태스크들이 혼합되어 수행될 수 있으며 프로그래밍 언어에 대한 옵션도 제공한다. 멀티 태스킹이 쓰레드 또는 프로세스, 혹은 쓰레드와 프로세스 모두에서 수행될 수 있기 때문에 멀티 쓰레드 프로세스(multi-threaded processes)의 지원이 필요하다.



(그림 1) PSE5의 포함 관계

3. POSIX Real Time API

범용 운영체제의 응용 소프트웨어 계층에 대한 인터페이스 표준으로 대표적인 것이 IEEE POSIX 1003.1이다. 이 표준은 또한 ISO/IEC 9945로도 불리운다. 수 년전부터 이 표준 API에 실시간 처리 기능을 위한 API를 추가하기 위한 노력이 있어 왔다. 이러한 노력은 개정안 1003.1b, 1003.1d 및 1003.1j로 나타나고 있다. 현재 1003.1b는 1003.1-1996년도판(ISO/IEC 9945-1:1996)에 흡수 통합되었다.

3.1 POSIX 1003.1b

POSIX 1003.1b는 1993년 발표된 최초의 실시간 기능을 추가하기 위한 API들을 정의한 규격[7]이다. 1996년도에 1003.1에 포함되었으며 다음과 같은 기능들을 정의하고 있다.

- real-time, queued signals
- process priority scheduling
- clocks and timers
- asynchronous I/O
- prioritized asynchronous I/O : async I/O queuing
- Guarantees file's data sync with disk
- fsync
- Files mapped as memory
- Lock all memory to avoid paing/swapping
- Lock memory range
- Set memory protection
- Message queues
- Counting Semaphores
- Shared Memory

3.2 POSIX 1003.1d

POSIX 1003.1d는 1999년 발표된 POSIX 1003.1의 개정안(Amendment d: Additional Realtime Extensions [C Language])[8]이다. 아직 1003.1에 통합되지 않았으며 다음과 같은 실시간 처리를 위한 기능들을 정의하고 있다.

- Efficient process creation/execution(spawn)
- New clock to measure execution time of thread

- New clock to measure execution time of process
- aperiodic scheduling(sporadic server)
- aperiodic scheduling(thread sporadic server)
- time out for blocking services : better error detection, recovery
- application can specify advisory info to system

3.3 POSIX 1003.1j

POSIX 1003.1j는 2000년 발표된 POSIX 1003.1의 개정안(Amendment j: Advanced Realtime Extensions [C Language])[9]이다. 1003.1b 및 1003.1d에서 다루지 못한 실시간 응용 프로그램 도메인을 지원하기 위한 기능들을 정의한다. 또한 실시간 응용이 시스템으로부터 예측 가능한 응답시간을 지원받을 수 있게 하기 위한 성능상의 기능들이 추가되었다. 이외에서 다중 처리기(multiprocessor) 시스템에서의 동기화나 이전 개정안의 정의들을 수정하였다. 다음과 같은 기능들을 정의하고 있다.

- 메모리 관리
  시스템에서 제공하는 물리 메모리(physical memory)의 종류별로 메모리 할당이나 접근을 수행하며 서로 다른 응용 프로그램들이 메모리 일부를 공유할 수 있도록 지원
- Clock과 Timer
  모노토닉 클럭(monotonic clock) 기능이 추가되었으며 응용이 지정하는 clock을 기반으로 절대(absolute) 또는 상대(relative) 시간 정지 기능이 제공됨
- 동기화
  다중 처리기를 갖는 시스템의 성능 향상을 위한 새로운 동기화 프리미티브. 이 기능은 실시간 처리와 직접적인 연관은 없음

4. EL/IX

시그너스(Cygnus Systems, 현재 RedHat에 통합되었음)가 1999년 Embedded Systems Conference에서 발표한 EL/IX(Embedded Linux based on POSIX)[10]는 임베디드 시스템에서 리눅스의 사용을 활성화하기 위한 개방형 표준이다. 임베디드 리눅스 뿐만 아니라 실시간 운영체제의 기능을 포함한 단일 API를 제시하였다. 즉 임베디드 응용의 요구사항에 따른 기능의 확장성(scalability)을 제공하고 동일한 레벨의 API 기능을 제공하는 운영체제들 사이에서는 응용의 이식성(portability)을 제공하는 것을 목표로 한다. 이러한 기능을 위한 API는 POSIX 1003.1 및 IOS C99 Standards의 서브셋을 정의하고 리눅스, BSD, SYSV의 기능들을 추가하여 임베디드 응용에 적합한 API를 구성하였다. 기본적으로 POSIX 1003.1(ISO/IEC 994501)을 만족하며 2000년 9월 Draft V1.2[11]가 발표되어 있다.

EL/IX에서는 API를 몇 단계의 레벨로 나누고 각 레벨에서 제공되는 기능은 그 상위 레벨에서는 모두 포함되도록 정의한다. 또한 시스템 구성 및 기능에 있어서 사용자가 선택할 수 있도록 옵션 문자(option letters)를 제공한다.

EL/IX의 각 레벨의 정의는 다음과 같다.

- 레벨 1 : 실시간 운영체제에 호환성을 갖는 레벨로 리눅스 및 eCOS, RTEMS, VxWorks, pSOS, VRTX32 등의 대표적인 임베디드 운영체제에서 모두 사용되는 기능들을 제공한다. 그러나 각 기능들이 그들과 동일한 것은 아니며 일부 축소되거나 수정되어 제공된다.
- 레벨 2 : 단일 처리(single process) 리눅스와 같은 프로그래밍 모델을 갖는다. 레벨 1의 기능에 실시간 운영체제에서는 쉽게 구현되지 않는 리눅스의 기능을 추가하였다. 또한 레벨 1에서 축소된 기능들은 완전한 구현을 제공한다.
- 레벨 3 : 다중 처리(multi-process) 리눅스와 같은 프로그래밍 모델을 갖는다. POSIX 1003.1을 기본으로 임베디드 응용에서는 쓰이지 않

는 기능(예:job 처리 기능)들은 제거된 API 집합으로 이루어진다.

- 레벨 4 : POSIX 1003.1 또는 리눅스 호환되는 계층이다. 임베디드 시스템에는 부적합한 기능들이 포함되어 있으며 실지로 EL/IX에는 해당되지 않는다.

EL/IX에서는 레벨과 함께 사용자에게 기능을 선택할 수 있는 옵션이 주어진다. 각 레벨에서 옵션을 통해 기능을 추가 또는 삭제할 수 있다. 중요한 옵션으로는 s(시그널 처리), f(화일 시스템 지원), t(터미널 지원), n(네트워크 지원), M(메모리 관리 지원), r(실시간 처리 지원), l(라이브러리만 지원) 및 c(호환성을 위해 지원) 등이다. 각 옵션은 다시 세부 기능별 옵션을 가진다.

EL/IX는 API를 임베디드 시스템의 규모별 부분집합으로 나누었다는 점에서는 POSIX 1003.13과 비슷하다. RedHat은 POSIX 1003.13과 EL/IX를 다음과 같이 자체 비교하였다.

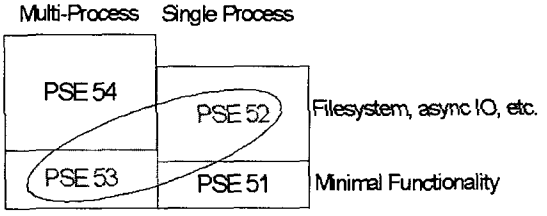
PSE51 : 레벨 1에 시그널과 메모리 관리 기능 옵션을 일부 추가

PSE52 : 레벨 2에 시그널, 메모리 관리, 파일 시스템 옵션을 추가

PSE53 : 레벨 3에 시그널 및 메모리 관리 기능의 일부 옵션을 추가

PSE54 : 레벨 4에서 GNU, BSD, SYSV 호환 기능들을 제거

(그림 2)는 EL/IX 레벨 3이 PSE52과 PSE53의 수퍼셋임을 나타낸다.



(그림 2) PSE5와 EL/IX 레벨 3

5. 국내외 표준화 동향

위에서 우리는 임베디드 시스템의 운영체제 API에 대한 표준 규격들을 살펴 보았다. 임베디드 시스템에 요구되고 있는 다기능화는 임베디드 운영체제와는 관계 없이 보였던 POSIX와도 연관성을 가지게 하였다. 현재 POSIX에서는 1003.1을 기반으로 실시간 운영체제 기능의 추가 및 인증 시험, 유틸리티, 시스템 관리, GUI 등에 대한 표준의 확장이 고려되고 있다. 이러한 표준화 활동은 지속적으로 수행될 것이며 VxWorks와 같은 상용 실시간 운영체제에서도 POSIX API를 지원하기 위한 노력이 계속되고 있다[3]. GNU 리눅스에서도 POSIX 1003.1b의 대부분 기능을 지원하고 있다. 그러나 아직 POSIX 1003.13에 따라 구현한 사례는 나타나지 않고 있다. 그리고 POSIX를 기반으로 하면서 임베디드 리눅스를 포함하는 API 표준인 EL/IX는 레벨 1을 만족하는 운영체제로써 eCOS가 지원되고 있다.

최근 API 표준 이외에 플랫폼 표준을 위한 작업이 진행되고 있다. 레드햇, 몬타비스타, 리네오, 리눅스웍스 등의 유수의 임베디드 리눅스 업체들이 모여 ELC(Embedded Linux Consortium)를 결성하고 2001년도부터 임베디드 리눅스 플랫폼에 대한 표준화 활동을 시작하였다[15]. 이 단체에서의 표준화 활동은 임베디드 시스템에서 운영체제로 임베디드 리눅스를 채택하였을 경우 플랫폼별로 각 계층의 최적의 솔루션을 제안한다는 의미의 표준이다. 따라서 운영체제 위의 각 계층별로 제시된 솔루션이 모여 플랫폼 표준을 구성한다. 최근 IBM이 참여하여 힘을 얻고 있으며 현재 표준화 추진 프로세스를 정립하고 초기 작업반을 구성하는 단계이며 레드햇은 EL/IX가 ELC 플랫폼 표준에 있어서 운영체제 표준으로 채택되도록 노력하고 있다.

한편 국내에서도 인터넷 정보가전을 위한 인터넷 정보가전 표준 포럼이 결성되어 정보가전용 임

베디드 운영체제 표준화 작업이 수행되고 있다. 우선 다중처리 임베디드 시스템에 적합한 API 표준으로 PSE52와 PSE53을 합한 규모에 해당하는 EL/IX 레벨 3의 API를 기반으로 표준 초안이 작성되었다. 또한 ELC 국내 멤버들을 중심으로 국내 임베디드 리눅스 업체의 플랫폼 표준을 제정하고 국제 ELC 표준화에 반영하고자 하는 표준화 활동을 시작하였다.

6. 결 론

임베디드 운영체제의 표준화는 앞으로 다양하고 새로운 임베디드 시스템용 응용 소프트웨어가 개발되어야 함을 고려할 때 시급히 이루어져야 할 분야이다. 이러한 표준화는 POSIX를 기반으로 했을 때 시스템 규모별 제공되는 기능의 API를 정의하는 것도 중요하지만 각 규모별 API를 확장할 수 있는 방법 및 기능의 단위를 나누는 방법에 대한 논의가 활발히 이루어 져야 할 것으로 생각된다. 또한 임베디드 시스템의 특성상 상위 응용 프로그램에 대한 호환성 제공 뿐만 아니라 하위 디바이스 드라이버에 대한 통일된 인터페이스 규격을 제공한다면 임베디드 시스템의 활용 가능성을 한층 더 높이게 될 것이다.

참고문헌

[1] 김선자 외, "인터넷 정보가전용 RTOS 기술 현황", 정보과학회지, 제 19권 제 4호, 2001, 한국정보과학회

[2] Integrated Systems, "pSOS System Concepts", 1996, http://www.isi.com

[3] WindRiver, VxWorks 5.4, http://www.windrivers.com

[4] http://www.mentor.com/embedded/vrtxos

[5] IDC, "Embedded and Handheld Operating Environment Market Forecast and Analysis, 2000-2004", May 2000.

[6] IEEE, IEEE-SA Standards Board, "IEEE Standard for Information Technology-Standardized Application Environment Profile-POSIX Realtime Application Support(AEP)", March. 1998.

[7] Bill O. Gallmeisster, "POSIX.4: Programming for the Real World", 1995, O'Reilly & Associates, Inc.

[8] IEEE, IEEE-SA Standards Board, "IEEE Standard for Information Technology-Portable Operating System Interface(POSIX)-Part 1: System Application Program Interface(API) - Amendment d: Additional Realtime Extensions[C Language]", Sept. 1999.

[9] IEEE, IEEE-SA Standards Board, "IEEE Standard for Information Technology - Portable Operating System Interface(POSIX)-Part 1: System Application Program Interface(API) - Amendment j: Advanced Realtime Extensions[C Language]", Jan. 2000.

[10] http://www.elix.redhat.com

[11] Nick Garnett, "EL/IX Base API Specification - Draft V1.2", Sept. 2000, http://sources.redhat.com/elix

[12] http://www.redhat.com

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

[14] http://www.lineo.com

[15] http://www.embedded-linux.org

저자약력

김 선 자

1985년 숙명여자대학교 수학과(이학사)
1995년 충남대학교 대학원 컴퓨터 공학과(공학석사)
1987년~현재 ETRI 컴퓨터·소프트웨어기술연구소 정보가전연구부 선임연구원
관심분야 : 운영체제, DSM, 인터넷 서버
e-mail : sunjakim@etri.re.kr

김 채 규

1978년 고려대학교 수학과(이학사)
1993년 호주 시드니 공과대 전산과학(석사)
1994년 호주 Wollongong대 전산과학(박사)
1997년-현재 ETRI 컴퓨터·소프트웨어기술연구소 책임연구원(정보가전연구부장)
관심분야 : 인터넷 정보가전, 실시간 운영체제, 멀티미디어, 데이터베이스, 전자상거래 등
e-mail : kyu@etri.re.kr

김 흥 남

1980년 서울대학교 전자공학과 학사
1989년 미국 Ball State University 전산학 석사
1996년 미국 Pennsylvania State University 전산학 박사
1983년-현재 ETRI 컴퓨터·소프트웨어기술연구소 정보가전연구부 책임연구원(내장형 S/W 연구팀장)
관심분야 : 실시간 운영체제, 비디오 압축 알고리즘, 분산 멀티미디어 시스템
e-mail : hnkim@etri.re.kr