EL/IX 단계 3을 적용한 실시간 운영체제 Qplus-P용 C 표준 라이브러리의 설계 및 구현

김 도 형[†] · 신 창 민[††] · 박 승 민[†††]

요 약

디지털 TV, 인터넷 셋탑박스, 인터넷 전화기 등과 같은 정보가전 제품이 속속 등장하면서 이들 제품의 기능을 제어하는데 필수적인 실시간 운영체제 시장이 크게 성장하고 있다. 한국전자통신연구원에서는 소형의 휴대 정보 단말에서부터 디지털 셋탑박스 및 홈 서버까지 다양한 종류의 정보가전 기기에 공통으로 사용될 수 있는 확장 가능한 표준 실시간 운영체제 Qplus-P를 개발하였다. 본 논문에서는 정보가전용 실시간 운영체제 Qplus-P에 탑재되는 C 표준 라이브러리의 설계 및 구현에 대해 기술한다. Qplus-P C 표준 라이브러리는 레드햇에서 실시간 운영체제 국제 표준으로 제안된 EL/IX 응용 프로그램 인터페이스 단계에 따라 설계되었다. 그리고, Qplus-P 응용 프로그램 수행에 필요한 Tiny-X, 카페 등을 지원하기 위해 추가적으로 필요한 함수들도 구현되었다. 구현된 C 표준 라이브러리는 EL/IX 응용 프로그램 인터페이스 단계를 적용하지 않은 C 표준 라이브러리보다 라이브러리 크기를 30% 정도 줄일 수 있었다.

The Design and Implementation of EL/IX level3 C Standard Library for RTOS Qplus-P

Do-Hyung Kim[†] · Chang-Min Shin[††] · Sung-Min Park[†††]

ABSTRACT

As the products of internet appliance, such as digital TV, internet set-top box, and internet phone, are continually produced, the market of real time operating system which controls those products is being highly increased. ETRI developed the extensible real time operating system, Qplus-P, which can be used from PDA to internet set-top box and home server. This paper describes the design and implementation of C standard library for real-time operating system Qplus-P. The Qplus-P C standard library follows EL/IX API level, which is proposed to the real-time operating system international standard by the RedHat. And, the C standard library functions, which are needed to support the Tiny-X and Kaffe, are also implemented. The implemented C standard library can reduce the size of library about 30% compare to the C library that does not follow EL/IX API level.

1. 서 론

21세기 정보통신 분야에서 네트워크 기술을 탑재한 정보가전들의 정보 서비스가 우리의 일상 생활에 직접적으로 활용될 것이라는 예견은 이미 실현되고 있다. 현재 가정에서 흔히 사용되고 있는 TV, 냉장고, 전화기, 게임기 등이 지능을 가진 정보 단말기 역할을 수행하여 가정 내 뿐만 아니라 집 밖에서의 생활 양식까지도 바꾸어 놓고 있다. 예를 들어, 인터넷을 통하여 원격의 시스템과 멀티미디어 데이터를 교환하고 각종 제어 정보를 처리할 수 있게 되었으며, 가정 내에 있는 각종 정보 가전기기들 간에도 유무선 통신을 통한 대용량 멀티미디어 전송이 가능해 졌다. 디지털 TV, 인터넷 셋탑박스, 인터넷 전화기 등 디지털 및 네트워크 기술을 바탕으로 한 정보가전 제품이 속속 등장하면서 이들 제품의 기능을 제어하는데 필수적인 정보가전용 실시간 운영체제 시장이 크게 성장하고 있다[2, 3]. 컴퓨터 운영체제 업체들과 기존의 실시간 운영체제 전문업체들은 그 동안 통신, 산업전자, 항공 및 우주산업 분야의 제어 시스템을 중심으로 형성되어온 실시간 운용체제 시장이 멀티미디어의 필요성에 따라 정보가전용으로 확대됨에 따라 이를 선점하고자 노력하고 있다[5-11].

윈드리버(WindRiver) 사는 초창기에 자동화 시스템, 제어 시스템 등과 같은 경성 실시간 응용 분야에 치중하였으나, 최근에는 인터넷 정보 가전을 겨냥하여 그래픽 사용자 인터페이스를 강화하고 조립성을 가미한 토네이도(Tornado) II와

† 정 회 원 : 한국전자통신연구원 컴퓨터소프트웨어기술연구소 선임연구원
†† 정 회 원 : 한국전자통신연구원 정보가전연구부 연구원
††† 정 회 원 : 한국전자통신연구원 컴퓨터 소프트웨어기술연구소 책임연구원
논문접수 : 2002년 9월 27일, 심사완료 : 2002년 12월 11일

VxWorks 5.4를 발표하였다. QNX사는 x86칩에 최적화된 작고 빠른 성능의 QNX 4.25를 개발하였다. QNX 4.25는 개인 휴대 단말기나 웹 폰 등에서 VxWorks와 윈도우 CE에 비하여 월등한 성능을 제공하고 있다. 마이크로소프트 사는 인터넷 정보 가전을 위하여 윈도우 CE 및 개발 도구를 발표하였다. 윈도우 CE는 주로 셋탑박스나 디지털 TV와 같은 덩치 큰 내장형 시스템에 주로 적용되고 있다. 그리고, 레드햇(RedHat) 사는 시그너스(Cygnus) 사를 최근에 합병하고 실시간 운영체제 eCos 및 GNU 개발 도구를 보급하고 있으며, 임베디드 리눅스 개발 프로젝트인 EL/IX(Embedded Linux based on POSIX)를 주도하고 있다. 이와 같이 국외 실시간 운영체제 전문업체들은 다양한 제품으로 인터넷 정보가전 시장을 선점하려고 노력하고 있다. 하지만, 국내에서는 정보가전용 실시간 운영체제가 거의 개발되지 않고 있으며, 막대한 로열티를 지불하고 외국제품의 수입에 전적으로 의존하고 있는 실정이다.

삼성전자는 미국의 ISI사가 개발한pSOS를 구입하여 디지털 TV를 비롯한 정보가전 제품에 탑재하였고, 개인 정보 단말기류와 인터넷 전화기에는 윈도우 CE와 퍼스널자바 등을 사용하고 있다. LG전자는 엑셀레이티드 테크놀로지 사가 개발한 뉴클리어스의 소스 코드를 들여와 디지털TV용으로 사용하고 있다. 그리고, 대우전자는 미국의 테크네마사의 소스코드를 도입하여 이를 개량해 「알바트로스」란 자체 OS로 개발하고, 인터넷 셋탑박스에 탑재하였다.

한국전자통신연구원에서는 소형의 휴대 정보 단말에서부터 홈 서버까지 다양한 종류의 정보가전 기기에 공통으로 사용될 수 있는 확장 가능한 표준 실시간 운영체제 Qplus-P를 개발하였다. 실시간 운영체제 Qplus-P는 커널, 라이브러리, 통합 개발 도구 등으로 구성되고, 1차적으로 홈 서버에 탑재되는 것을 목표로 하였다. Qplus-P 라이브러리는 크게 C 표준 라이브러리, 그래픽/윈도우 라이브러리, 네트워크 라이브러리, 그리고 파일 시스템으로 구성된다. 본 논문에서는 Qplus-P에 탑재되는 C 표준 라이브러리의 설계 및 구현에 대해 다루고, 커널, 통합 개발도구, 다른 라이브러리에 대한 설명은 생략하도록 한다.

일반적으로 정보가전 기기에서는 사용할 수 있는 하드웨어 자원이 제한되어 있으므로, 가능한 C 표준 라이브러리를 정보가전 응용에 적합한 함수들로 구성하여, 라이브러리 크기를 줄이는 것이 유리하다. 하지만, 공개된 glibc 표준 라이브러리는 다양한 함수들이 포함되어 라이브러리 사이즈가 크고, 라이브러리에 포함된 함수들 중에서 정보가전 응용에 불필요한 함수들이 존재한다. 따라서, 특정 기준에 따라 C 라이브러리 함수들을 분류하여 정보가전 응용에 적합한 함수들만 구현하는 것이 필요하다. 본 논문에서는 레드햇 사에서 실시간 운영체제 표준으로 제시한 EL/IX 응용 프로그램 인터페이스(API) 단계(level)를 기준으로, 정보가전 응용에 적합한 Qplus-P C 표준 라이브러리를 구현하였다.

본 논문은 다음과 같이 구성되어 있다. 2절에서는 EL/IX 응용 프로그램 인터페이스 표준안에 대해서 간략히 기술하고, 3절에서는 C 표준 라이브러리 설계 및 구현 환경에 대해 다룬다. 4절에서는 C 표준 라이브러리의 구현과 특징에 대해 기술하고, 마지막으로 5절에서는 결론 및 향후 과제에 대해서 다룬다.

2. EL/IX 응용 프로그램 인터페이스 표준안

EL/IX 응용 프로그램 인터페이스 표준안은 내장형 리눅스와 다른 실시간 운영체제에 적합한 단일 표준 응용 프로그램 인터페이스 제시를 목적으로 1999년 시그너스 사에 의해 제안되었다. EL/IX 표준안은 POSIX를 기반으로 응용 프로그램 인터페이스를 4단계로 나누어서 정의하였고, 각 단계에서도 사용자의 선택에 의해 기능을 추가 또는 삭제할 수 있는 옵션 기능을 제공한다. EL/IX 응용 프로그램 인터페이스 단계에 따라 라이브러리를 구성하게 되면, 동일한 응용 프로그램 인터페이스 규격을 만족하는 운영체제 사이에서는 응용 프로그램 간의 호환성을 제공하는 장점이 있다. EL/IX 표준안의 각 단계별 응용 프로그램 인터페이스 규격은 다음과 같다.

- 단계 1 : 대표적인 내장형 실시간 운영체제에서 모두 사용되는 기능을 제공
- 단계 2 : 단일 처리기 리눅스용 응용 프로그램 인터페이스 추가
- 단계 3 : 내장형 응용을 위한 다중 처리기 리눅스용 응용 프로그램 인터페이스 추가
- 단계 4 : POSIX 응용 프로그램 인터페이스 호환

2000년 8월에 EL/IX 규격을 지원하는 레드햇의 내장형 운영체제인 eCOS의 개발 버전이 발표되었고, 2000년 9월에 응용 프로그램 인터페이스 드래프트(draft) 규격 1.2가 릴리스 되었다. 현재 인텔, 도시바 등이 이 표준 안에 참여하고 있다. Qplus-P C 표준 라이브러리는 EL/IX 응용 프로그램 인터페이스 단계를 glibc에 최초로 적용하여 Qplus-P 응용에 적합한 함수들을 구현하였다.

3. C 표준 라이브러리의 설계

실시간 운영체제에서의 C 표준 라이브러리는 표준 인터페이스에 따라 정의되어야 한다. 응용 프로그램에서 사용되는 라이브러리 함수들이 표준 인터페이스에 따라 정의되면, 라이브러리 함수 호출로 인한 추가적인 코드 변경 없이 다른 운영체제 상에서 쉽게 수행될 수 있다. 그리고, 일반적으로 정보가전 기기에서는 사용할 수 있는 하드웨어 자원과 응용 분야가 제한되어 있기 때문에, 불필요한 C 라이브러리 함수들의 탑재는 가용 자원의 크기를 줄여 응용 프로그램들의 반

응 시간을 증가 시킬 수 있다. 이 절에서는 Qplus-P C 표준 라이브러리 개발시 참조된 코드와 라이브러리 크기를 줄이기 위한 함수 선택 기준에 대해서 다룬다.

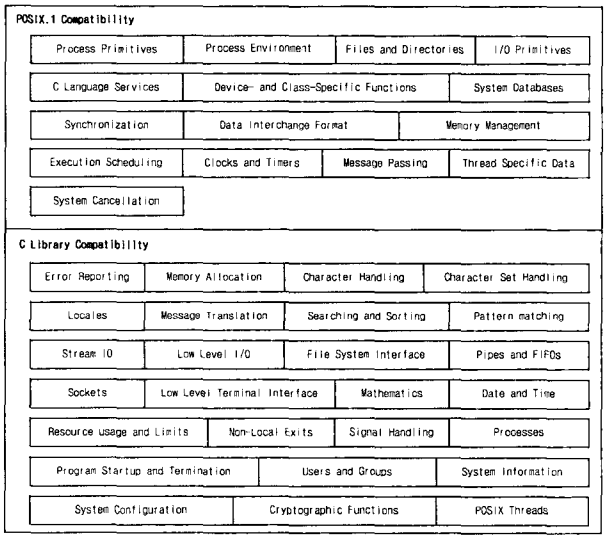
3.1 참조된 C 라이브러리 소스 코드

Qplus-P용 C 표준 라이브러리는 공개 소스인 GNU glibc-2.2.3을 이용하여 개발되었다. Glibc-2.2.3은 현재에도 계속적으로 발전되고 있는 코드이고, 다양한 종류의 함수들이 포함되어 있다[5, 6, 11, 12]. 따라서, C 표준 라이브러리가 Qplus-P응용 분야에 적합한 함수들로 구성되도록 glibc-2.2.3에 적절한 기준을 적용하여, Qplus-P 응용 분야에 적합한 함수들을 선택하는 것이 중요하다[13-16].

3.2 함수 선택 기준

공개 소스인glibc-2.2.3에 포함된 함수들 중에서 정보가전 응용 분야에 적합한 함수들을 선택하기 위해서, 레드햇 사에서 실시간 운영체제 표준안으로 제시된 EL/IX 응용 프로그램 인터페이스 단계를 적용하였다[1]. EL/IX 응용 프로그램 인터페이스는 총 4 단계까지 정의되어 있는데, C 표준 라이브러리 구현에는 EL/IX 단계 3을 기준으로 하였다. EL/IX 단계 3은 임베디드 응용 분야를 위한 하나의 사용자와 다수의 프로세스들을 위한 기능들을 포함하고 있다.

EL/IX 표준안에서 정의된 응용 프로그램 인터페이스는 크게 POSIX.1 호환 함수와 C 호환 함수들로 구분된다. 이들 호환 함수 내에는 다수의 함수 그룹들이 존재하는데, (그림 1)은 이 함수 그룹들을 보여준다. 본 논문에서는 각 함수 그룹에 포함된 세부 함수들은 생략하도록 한다.



(그림 1) EL/IX 함수 그룹

(그림 1)에서 구분된 각 함수 그룹에는 다수의 함수들이 포함되어 있고, 이들 함수들에는 적절한 EL/IX 단계가 정의되어 있다. (그림 2)는 POSIX.1 호환 함수 그룹인 프로세스 환경(Process Environment)의 서브항목인 프로세스 검증(Identification)과 사용자 검증에 관련된 함수들을 보여준다. (그림 2)에서 보듯이 getuid, getgid 함수들은 EL/IX 단계 1로 정의되어 있고, getgroups, getlogin 함수들은 EL/IX 단계 4로 정의되어 있음을 알 수 있다.

Process Identification

Function	Level	Options	Notes
getpid()	1	c	1
getppid()	1	c	1

User Identification

Function	Level	Options	Notes
getuid()	1	c	1
geteuid()	1	c	1
getgid()	1	c	1
getegid()	1	c	1
setuid()	4		1
getgid()	4		
getgroups()	4		
getlogin()	4		
getlogin_r()	4		

(그림 2) EL/IX 단계 정의 예

구현된 C 표준 라이브러리는 (그림 2)와 같은 EL/IX 단계를 기준로 하고, 응용 프로그램 인터페이스 드래프트 규격 1.2에서 단계 3 이하(≤)로 정의된 함수들을 우선적으로 구현하였다.

3.3 구현 환경

실시간 운영체제 Qplus-P는 1차적으로 홈 서버에 탑재되는 것을 목표로 하였다. 홈 서버는 가정에서 인터넷을 사용할 수 있도록 지원하고, 가정내의 멀티미디어 기기를 연결하기 위한 다양한 인터페이스를 포함한다. 홈 서버의 하드웨어 사양은 다음과 같다.

- CPU : 펜티엄 3(1GHz)
- 메모리 : 256MBytes
- 디스크 : 40GBytes
- 플래시 : 32MBytes
- 그래픽카드 : nVIDIA Geforce2MX
- 주변 인터페이스 : 이더넷(Ethernet) 카드, IEEE 1394 카드, 무선 랜(LAN) 카드 등

Qplus-P 개발 시 홈 서버는 타겟(target)으로 사용되어 자체적으로 부팅할 수 없고, 호스트로부터 네트워크을 통해 부팅 및 실행에 필요한 정보를 다운로드 해야 한다. 이를 위해 호스트에는 홈 서버에 새 인터넷 주소를 할당하기 위한 DHCP(Dynamic Host Configuration Protocol) 데몬이 동작하고 있어야 한다. 그리고, 홈 서버가 부팅 이미지를 다운로드 할 수 있도록 해주는TFTP(Trivial File Transfer Protocol)데몬과 네트워크을 통해 홈 서버의 루트 파일 시스템 설정할 수 있도록 하는 NFS(Network File System) 데몬이 호스트에서 동작해야 한다. 개발에 사용된 호스트는

레드햇 7.1이 설치된 i686 리눅스 시스템이고, 타겟 홈 서버의 부팅 순서는 다음과 같다.

홈 서버는 먼저 부트 플로피 디스켓을 이용해서 기본적인 하드웨어를 초기화(예를 들어, 네트웍 초기화) 한 후에, 자신에게 인터넷 주소를 할당할 DHCP 서버(호스트)를 네트워크 상에서 찾아 서버에서 지정된 인터넷 주소를 할당받는다. 그런 다음, TFTP를 이용하여 서버로부터 부팅 이미지를 다운로드 하여 부팅한 후, 루트 파일 시스템 디렉터리 정보를 사용자가 입력하기를 기다린다. 사용자가 홈 서버의 루트 파일 시스템이 존재하는 호스트 내 디렉터리 경로를 입력하면, NFS를 이용해서 그 디렉터리를 자신의 루트 디렉터리로 설정하여 부팅하게 된다.

홈 서버는 위와 같이 부팅시 호스트의 특정 디렉터리를 자신의 루트 파일 시스템으로 설정하기 때문에, 홈 서버에서 동작하는 C 표준 라이브러리의 개발은 비교적 단순하다. 즉, 호스트에 설치된 i386 크로스 컴파일러를 이용하여, glibc-2.2.3을 컴파일 한 다음, 생성된 라이브러리를 홈 서버의 루트 파일 시스템으로 복사하면 된다. 홈 서버에는 컴파일러가 없기 때문에, 테스트 파일 수행도 동일한 방식으로 이루어 진다. 즉, 소스 파일을 먼저 호스트에서 크로스 컴파일러를 사용해서 컴파일 한 다음, 홈 서버의 루트 파일 시스템으로 실행 파일을 복사한다. 그러면, 홈 서버가 부팅한 후, 복사된 실행 파일을 홈 서버의 터미널 상에서 수행할 수 있게 된다.

4. 구 현

이 절에서는 C 표준 라이브러리의 구현 단계와 Qplus-P C 라이브러리의 특징에 대해서 기술한다.

4.1 구현 단계

C 표준 라이브러리는 공개 소스인 glibc-2.2.3과 레드햇 EL/IX 단계3을 이용하여 구현되었다. 구현 단계는 먼저 EL/IX 단계3을 glibc-2.2.3 소스에 적용하는 1단계와 1단계에서 생성된 동적 라이브러리에 Qplus-P 응용 프로그램 수행 시 필요한 함수들을 추가하는 2단계로 이루어 진다.

먼저 1단계에서는 glibc-2.2.3 소스 디렉터리 내의 Makefile에서 EL/IX 단계4인 함수들을 제거하기 위해, 소스 내의 Makefile routines을 크게 3부분으로 구분하였다. 즉, Makefile routines에 포함된 함수들 중에서 EL/IX 단계 3이하의 함수들은 기본적으로 포함하고, EL/IX 단계4인 함수들은 ifeq ($(elix-level4), yes)을 사용해서 제거하였다. 그리고, EL/IX 단계에 정의되지 않은 함수들은 ifeq ($(posix-install), yes)를 이용하여 분리하였다. 이렇게 하면, 소스 컴파일 시에 EL/IX 단계3이하로 정의된 함수들만 컴파일 된다. 하지만, glibc-2.2.3에서 EL/IX 단계 3까지의 함수들 만으로는 라이브러리가 제대로 만들어지지 않고, 컴파일 시 오류가 발생한다. 그것은 glibc-2.2.3 내의 함수들이 상호 함수 호출을 빈번하게 사용하여, 단계3으로 정의된 함수라도 단계4에 포함된 함수가 있어야 동작하기 때문이다. 즉, glibc-2.2.3에 레드햇 EL/IX 단계3만을 적용한 Qplus-P용 C 라이브러리를 생성할 수 없고, 단계4에 포함된 일부 함수들도 구현되어야 한다. 예를 들어, (그림 2)에서 보면, getgid, getgroups, getlogin_r 함수들은 EL/IX 표준안에 단계4로 정의되어 있지만, 이 함수들은 C 라이브러리 생성시 포함되어야 한다. 위와 같이 Makefile 에서 EL/IX 단계 3에는 포함되지 않지만, 라이브러리 생성시 필요한 함수들을 Makefile의 기본 routines에 포함시키는 작업을 수행하였다. 그리고, glibc-2.2.3 소스 내의 Makefile에는 다수의 바이너리(binary) 파일들이 생성되도록 설정되어 있는데, EL/IX 단계3을 적용하기 위해서는 이 부분을 일부 제거해야 한다. 왜냐하면, EL/IX 단계3 이하인 함수들만으로는 이들 바이너리를 생성하는 것이 불가능하기 때문이다. 1단계 작업이 완료된 다음, EL/IX 단계3으로 정의된 함수와 추가 함수들을 하나의 쉘 스크립트 파일(reduced-libc_pic.sh)로 만들어 라이브러리 구현에 사용하고, 소스 내의 Makefile들을 원래의 수정되지 않은 Makefile 로 복구하였다. (그림 3)은 쉘 스크립트 파일 일부를 보여준다. C 라이브러리 구현 시 필요한 함수들을 쉘 스크립트로 정의해서 사용하면 기존의 glibc-2.2.3 소스 내의 Makefile들을 수정할 필요가 없으므로, 구현이 간단해진다는 장점이 있다.

홈 서버에는 최종적으로 동적 라이브러리들만 탑재하므로, 작성된 쉘 스크립트 파일을 동적 C 라이브러리(libc-2.2.3.so) 생성에 적용하였다. 동적 C 라이브러리는 컴파일 과정 중 libc_pic.os를 기본으로 만들어 지는데, 이 때 쉘 스크립트 파일이 적용되도록 glibc-2.2.3 내의 Makerules파일을 (그림 4)와 같이 변경하였다. 1 단계에서 EL/IX 단계 3을 이용하여 동적 라이브러리를 생성한 다음, 홈 서버의 루트 파일 시스템의 특정 디렉터리로 복사하였다.

```
#! /bin/sh
# dir = $( mktemp -d ${ TMPDIR : -. } /depout.XXXXXXX )
file = $( mktemp ${ TMPDIR : -. } /depout.XXXXXXX )
rm $file
dir = $file
mkdir $dir
cd $dir

#extract original elix-level 3 objects
ar xv $OLDPWD/libc_pic.a C-address.os C-collate.os C-ctype.os
C-identification.os C-measurement.os C-messages.os C-monetary.os
C-name.os C-numeric.os C-paper.os C-telephone.os C-time.os
C_name.os SYS_libc.os __longjmp.os _exit.os
.....
wmemcpy.os wmemmove.os wmempcpy.os wordexp.os write.os
writev.os xdr.os xdr_array.os xdr_mem.os xdr_rec.os xmknod.os
xstat.os xstat64.os

rm - f $OLDPWD/reduced-libc_pic.a
ar rcs $OLDPWD/reduced-libc_pic.a *
cd -
rm - fr $dir
```

(그림 3) 쉘 스크립트 파일

```
$(common-objpfx)libc_pic.os : $(common-objpfx)libc_pic.a
        (cd $(common_objpfx); sh $(srcdir)/reduced libc_pic.sh)
        $(LINK.o) nostdlib nostartfiles r o $@ \
        $(LDFLAGS c_pic.os) -Wl, d \
        -Wl,- whole-archive $(common-objpfx)reduced libc_pic.a
```

(그림 4) 수정된 Makerules 파일

1단계 작업이 끝난 후, 생성된 동적 라이브러리를 이용하여 Qplus-P 환경 설정과 응용 프로그램 수행에 필요한 BusyBox, Tiny X, 카페(Kaffe), 그리고 디버거(Debugger) 등을 홈 서버에서 동작하도록 하는 2단계 작업이 진행되었다. 2단계 작업 동안 다수의 함수들이 새롭게 쉘 스크립트 파일에 추가되었으며, (그림 5)는 쉘 스크립트 파일에 추가된 함수들을 보여준다.

```
# extract extra objects needed for proper operation
ar xv $OLDPWD/libc_pic.a basename.os cxa_finalize.os obstack.os
old_atexit.os dl-addr.os dl-sym.os fgetgrent_r.os fgetpwent_r.os
sgetspent_r.os fgetspent_r.os ftime.os getgrent.os getgrent_r.os
getgrnam.os getgrgid.os getgrnam_r.os getgrgid_r.os getlogin.os
getpgrp.os getpt.os getpwent_r.os getspent_r.os syslog.os
.....
umount2.os memstream.os dprintf.os wstrops.os semop.os
semget.os semctl.os swapon.os swapoff.os pm_getmaps.os
pmap_clnt.os pm_getport.os pmap_prot.os pmap_prot2.os
pmap_rmt.os xdr.os xdr_ref.os rpc_cmsg.os peekc.os

# extract extra objects for gdbserver
ar xv $OLDPWD/libc_pic.a ptrace.os dl_sym.os mtrace.os
set-freeres.os
```

(그림 5) 쉘 스크립트 파일에 추가된 함수들

(그림 5)에서 보듯이 Qplus-P C 표준 라이브러리는 EL/IX 단계 3함수들과 Qplus-P 응용 프로그램 수행 환경을 지원하기 위한 함수들로 구현되었다.

4.2 Qplus-P C 표준 라이브러리의 특징

Qplus-P C 표준 라이브러리는 레드햇 사에서 제안된 EL/IX 표준안을 glibc-2.2.3에 최초로 적용하여 구현되었다. 그리고, C 표준 라이브러리는 Qplus-P 응용 프로그램 수행에 필요한 기본적인 Tiny X, 카페, 디버거 등이 동작할 수 있도록 다수의 함수들이 새롭게 추가되었다. 따라서, Qplus P용 C 표준 라이브러리는 레드햇 사의 EL/IX 단계3과는 다른 Qplus-P 응용 분야에 적합한 함수들을 포함하고 있다 (EL/IX 단계3 < Qplus P C 라이브러리 < Glibc-2.2.3).

일반적으로 glibc-2.2.3을 설치하게 되면, 많은 수의 함수들이 포함되기 때문에 동적 라이브러리의 크기가 커진다. 하지만, 구현된 C 표준 라이브러리는 동적 라이브러리의 크기를 상당히 줄이면서도 Qplus-P응용 프로그램 수행에 필요한 함수들을 지원한다. <표 1>은 glibc-2.2.3을 설치했을 때와 Qplus-P용 C 라이브러리를 설치했을 때, 동적 C 라이브러리 glibc-2.2.3.so의 크기를 비교한 것이다.

<표 1> 동적 라이브러리 비교

동적 라이브러리 이름	크기
libc 2.2.3.so(표준 버전)	5.4M (심볼 포함 시)
	1.2M (심볼 제거 시)
libc 2.2.3.so(Qplus P 버전)	3.9M (심볼 포함 시)
	890K (심볼 제거 시)

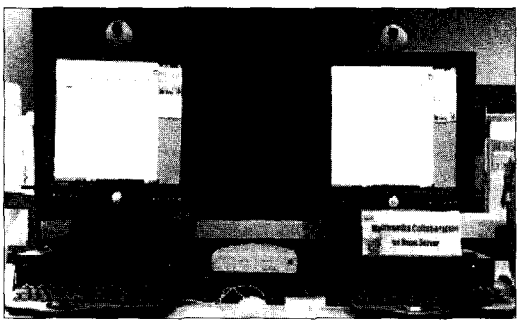
<표 1>에서 보듯이 Qplus-P용 C 라이브러리는 표준 버전의 동적 C 라이브러리보다 심볼 포함시는 약 1.5MByte, 심볼 제거시는 약 330Kbyte 작다는 것을 알 수 있다. 즉, Qplus-P용 C 표준 라이브러리는 EL/IX를 적용하지 않은 C 라이브러리보다 라이브러리 크기를 약 30%정도 줄일 수 있었다. 이는 하드웨어 자원이 제한되어 있는 소형 정보가전 기기에서 보다 작은 라이브러리로 Qplus-P 응용들을 지원할 수 있다는 것을 의미한다.

4.3 C 라이브러리의 수행

C 표준 라이브러리는 일반적으로 다른 응용 프로그램에 포함되어 수행되므로, 이 절에서는 Qplus-P 응용 프로그램들의 수행 예제를 통해 C 라이브러리의 수행을 간접적으로 보여준다.

(그림 6)은 Qplus P를 이용한 DVD 플레이어의 수행 화면을 보여준다. (그림 6)에서 보듯이 Qplus-P는 홈 서버에 탑재되고, DVD 플레이어는 홈 서버의 한 응용으로 수행된다. (그림 6)에서 모니터 앞에 위치한 것이 Qplus P가 탑재된 홈 서버이다.

(그림 6) DVD 플레이어 수행 화면



(그림 7) 화상 통화 수행 화면

(그림 7)은 Qplus-P가 설치된 홈 서버를 이용한 화상 통화의 수행 화면을 보여준다. 두 개의 홈 서버에 설치된 카메라를 이용하여 통화를 하게 되면, 홈 서버의 웹 브라우저를 통해 화상이 표시되게 된다.

5. 결론 및 향후 연구

본 논문에서는 프로세스 기반의 정보가전용 실시간 운영체제Qplus-P에 탑재되는 C 표준 라이브러리의 설계 및 구현에 대해 기술하였다. 실시간 운영체제Qplus-P는 크게 커널, 라이브러리, 통합 개발 도구로 구성된다. C 표준 라이브러리는 공개 소스인 glibc-2.2.3에 레드햇 EL/IX 표준안을 최초로 적용하여 구현되었고, Qplus-P 응용 프로그램 개발에 필요한 함수들도 추가로 구현되었다. 구현된 C 표준 라이브러리는 EL/IX를 적용하지 않은C 라이브러리보다 라이브러리 크기를 30% 정도 줄일 수 있었다.

앞으로 실시간 운영체제 Qplus-P는 홈 서버 뿐만 아니라, 웹 패드, 개인 휴대 단말기 등의 다양한 정보가전에 탑재될 예정이다. 따라서, C 표준 라이브러리는 정보가전 응용 분야가 요구하는 추가적인 함수들을 지원하도록 확장되어야 하고, 개인 휴대 단말기 등과 같은 작은 크기의 정보가전과 홈 서버에서 제공되어야 하는 함수들을 선택적으로 구성할 수 있도록 하여야 한다.

참 고 문 헌

[1] Nick Garnett, "EL/IX Base API Specification DRAFT-V1.2," Sep., 2000.

[2] 한국전자통신연구원, "정보가전용 실시간 운영체제 컨퍼런스", RTOS'99 자료집, 1999.

[3] 한국전자통신연구원, "정보가전용 실시간 운영체제 컨퍼런스", RTOS 2000 자료집, 2000.

[4] 한국전자통신연구원, "인터넷정보가전기술개발 워크샵 자료집", 2001.

[5] ISO/IEC 9945-1, "C 언어를 위한 시스템 응용 프로그래밍 인터페이스(API) 표준", 1993.

[6] 체신부, "개방형 운영체제 인터페이스(POSIX.1) 표준", 1993.

[7] "VxWorks 5.3.1 Programmer's Guide Edition 1," Wind River Systems, 1997.

[8] "VxWorks Training Workshop," Wind River Systems, 1996.

[9] "pSOSystem Programmer's Reference," Integrated Systems, 1997.

[10] "VTRX Reference Guide," Mentor Graphics Corporation, 1997.

[11] Brian W. Kernighan, Dennis M. Ritchie, "The C Programming Language," Prentice Hall, 1988.

[12] W. Richard Stevens, "Advanced Programming in the UNIX Environment," Addison-Wesley Publishing Company, 1992.

[13] Narayanan AK, "Design of a safe string library for C," Software-Practice & Experience, Vol.24, No.6, pp.565-578, June, 1994.

[14] Plauger PJ, "Embedded C++," Embedded Systems Programming, Vol.9, No.12, pp.125-126, Nov., 1996.

[15] Woehr JJ, "A C++ library for IBM MQSeries," Dr. Dobb's Journal, Vol.25, No.7, pp.52-55, July, 2000.

[16] Lee Jin S., Hayati Samad, Hayward Vincent, and Lloyd John E, "IMPLEMENTATION OF RCCL, A ROBOT CONTROL C LIBRARY ON A MICROVAX II," Proceedings of SPIE - The International Society for Optical Engineering, Vol.726, pp.472-480, 1987.

김 도 형

e-mail : kimdh@etri.re.kr

1993년 경북대학교 컴퓨터공학과(공학사)

1995년 포항공과대학 전자계산학과 (이학석사)

1995년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 선임연구원

관심분야 : 결함포용, 실시간시스템, 성능감시, 성능평가

신 창 민

e-mail : cmshin@etri.re.kr

1996년 고려대학교 컴퓨터학과(이학사)

1998년 고려대학교 컴퓨터학과(이학석사)

1998년-2001년 (주)대우통신

2001년-현재 한국전자통신연구원 정보가전연구부 연구원

관심분야 : 운영체제, 모바일컴퓨팅, 편재형 컴퓨팅, 무선통신, 3D 그래픽스

박 승 민

e-mail : minpark@etri.re.kr

1981년 울산대학교 전자공학과(학사)

1983년 홍익대학교 전자공학과(석사)

1983년~1984년 LG 전자 연구원

1984년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 책임연구원

관심분야 : 실시간 운영체제, 네트워크 컴퓨팅