실시간 운영체제 Q+를 위한 C 표준 라이브러리 설계 및 구현

김 도 형† · 박 승 민†

요 약

본 논문에서는 정보가전용 실시간 운영체제로 개발되고 있는 Q+에 탑재될 C 표준 라이브러리의 설계 및 구현에 대해 기술한다. 실시간 운영체제에서의 C 표준 라이브러리는 표준 인터페이스에 따라 정의되어야 하고, 쓰레드들의 동시 수행이 가능하도록 하여야 한다. 구현된 C 표준 라이브러리는 POSIX.1에 정의된 표준 인터페이스를 따르고, 재진입가능 하도록 설계되었다. 그리고, POSIX.1에 정의된 표준 함수들 중에서 Q+ 응용 분야에 적합하고, 상용 실시간 운영체제들이 공통적으로 제공하는 함수들을 지원한다. C 표준 라이브러리 함수들은 함수간 호출관계를 분석해 결정된 구현 순서에 따라 Q+ 커널과 디지털 TV용 셋탑박스 상에서 구현되었다.

The Design and Implementation of C Standard Library for RTOS Q+

Do-Hyung Kim† · Sung-Min Park†

ABSTRACT

This paper describes the design and implementation of C standard library for real-time operating system Q+, that is being developed for the internet appliance. The C library in the real-time operating system should be defined according to the standard interface and support the concurrent execution of threads. The implemented C standard library is reentrant and follows POSIX.1 standard interface. And, the C standard library functions, which are adequate to the Q+ application and commonly provided by commercial real-time operating systems, are selected among POSIX.1 standard functions. The C standard library is implemented on the Q+ kernel and D-TV set-top box according to the implementation sequence, which is determined by analyzing the relation of function calls.

키워드 : POSIX.1, RTOS, Q+

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. 그리고, 디지털 TV, 인터넷 셋탑박스(set-top box), 인터넷 전화기 등 디지털 및 네트워크 기술을 바탕으로 한 정보가전 제품이 속속 등장하면서 이들 제품의 기능을 제어하는데 필수적인 실시간 운영체제(RTOS) 시장이 크게 성장하고 있다[1, 2].

컴퓨터 운영체제 업체들과 기존의 실시간 운영체제 전문 업체들은 그 동안 통신, 산업전자, 항공 및 우주산업 분야의 제어시스템을 중심으로 형성되어온 실시간 운용체계 시장이 멀티미디어의 필요성에 따라 정보가전용으로 확대됨에 따라 이를 선점하고자 노력하고 있다. 정보가전 시장을 겨냥해 윈도우CE를 출시한 마이크로소프트는 휴대형 PC, 휴대용 정보단말기(PDA) 등에 윈도우CE를 보급시킨 데 이어 인터넷 셋탑박스, 디지털TV 시장도 선점하려 하고 있다. 선 마이크로시스템 사도 실시간 운영체제 전문업체인 Chorus 사를 인수하고, 자체 개발한 퍼스널자바(pJAVA) 등을 내세워 정보가전용 실시간 운영체제 시장에 도전하고 있다. 그리고, 그 동안 실시간 운영체제 VRTX, VxWORKS, pSOS, QNX 등으로 실시간 운영체제 시장을 장악하고 있던 미국의 마이크로텍(Microtec), 윈드리버(WindRiver) 시스템, ISI사 및 캐나다의 QSSL사 등도 그 동안 축적된 노하우를 바탕으로 정보가전 시장을 위한 제품들을 지속적으로 개발하고 있다[6-10, 15-18]. 이러한 해외 업체들의 활발한 시장 선점 경쟁에도 불구하고, 국내에서는 정보 가전용 실시간 운영체제가 개발되지 않고 있으며 외국제품의 수입

† 정 회 원 : 한국전자통신연구원 컴퓨터소프트웨어기술연구소
논문접수 : 2000년 7월 8일, 심사완료 : 2001년 2월 22일

에 전적으로 의존하고 있는 실정이다[1, 2].

21세기 최대의 황금 시장이 될 정보가전 제품에 필수적인 정보가전용 실시간 운영체제의 개발 필요성에 의해 한국전자통신연구원(ETRI)에서는 삼성전자, 대우 전자, LG 전자 등과 공동으로 정보가전용 실시간 운영체제인 Q+(Q 플러스)를 개발하였다. 실시간 운영체제 Q+는 마이크로 커널, 라이브러리, 사용자 개발 도구, 그리고 응용 API 등으로 나누어지고, Q+ 라이브러리는 다시 C 표준 라이브러리, 그래픽/윈도우 라이브러리, 네트워크 라이브러리, 스마트카드 라이브러리, 그리고 파일 시스템으로 구성된다. 그래픽/윈도우 라이브러리는 Q+ 응용을 위한 그래픽 프리미티브(primitive) 라이브러리와 윈도우 매니저를 개발하고, 네트워크 라이브러리는 네트웍 공용 라이브러리, 텔넷(telnet) 클라이언트(client), tftp 클라이언트와 서버, DHCP 클라이언트, PPP 클라이언트 등을 구현한다. 스마트카드 라이브러리는 스마트카드 리더기 및 카드 관련 API를 개발하고, 파일 시스템은 플래시 파일 시스템과 디스크 파일 시스템을 구현한다. 본 논문에서는 Q+에 탑재될 라이브러리 중 C 표준 라이브러리의 설계 및 구현에 대해서 다루고, 커널, 개발도구, 그리고 다른 라이브러리들에 대한 자세한 내용은 생략하도록 한다. 실시간 운영체제 Q+는 1차적으로 인터넷 셋탑박스에 탑재되어 디지털 TV(D-TV)용 응용 분야를 목표로 1999년 1월부터 2000년 12월까지 개발되었다.

본 논문은 다음과 같이 구성되어 있다. 2절에서는 Q+에 탑재될 C 표준 라이브러리 함수들의 선택 기준 및 함수 구성 등에 대해서 기술하고, 3절에서는 구현 환경에 대해서 다룬다. 4절에서는 C 표준 라이브러리 구현에 대해 기술하고, 마지막으로 5절에서는 결론 및 향후 과제에 대해서 다룬다.

2. C 표준 라이브러리의 설계

이 절에서는 Q+용 C 표준 라이브러리 개발 시 참조된 코드와 함수 선택 기준 등에 대해서 기술한다.

2.1 참조된 C 라이브러리 소스 코드

실시간 운영체제에서의 C 표준 라이브러리는 표준 인터페이스에 따라 정의되어야 하고, 쓰레드들의 동시 수행이 가능하도록 하여야 한다. 만약 라이브러리 함수가 쓰레드들 사이에 공유될 수 없다면, 한번에 하나의 쓰레드만 수행되어 응용 프로그램의 반응 시간이 증가하게 된다. 그리고, 실시간 운영체제가 탑재되는 기기에서는 사용할 수 있는 하드웨어 자원과 응용 분야가 제한되어 있기 때문에, 불필요한 라이브러리 함수들의 탑재는 가용 자원의 크기를 줄여 응용 프로그램들의 반응 시간을 증가시킬 수 있다. 따라서, Q+에 탑재될 C 표준 라이브러리는 다음과 같은 요구조건을 만족시키도록 설계되었다.

- POSIX.1에 정의된 표준 인터페이스를 따르도록 설계함으로써 운영체제 간 호환성을 높일 수 있도록 한다.
- 가능한 경량(lightweight)으로 설계하여, 라이브러리 수행 시 시스템에 걸리는 부하를 줄이도록 한다.
- 개발 대상 시스템의 크기와 응용 분야에 적합하게 크기를 조정할 수 있도록 모듈화하게 설계한다.
- 재 진입가능(reentrant) 하도록 한다.

Q+에 탑재될 C 표준 라이브러리는 공개된 C 라이브러리 소스들을 이용하여 개발되었는데, GNU glibc-2.0.x와 시그너스(Cygnus)사의 newlib가 주로 참조되었다. GNU glibc-2.0.x는 현재에도 계속적으로 발전되고 있는 코드이고, newlib는 GNU glibc 코드를 기반으로 시그너스 사에서 새롭게 수정한 코드이다. 이 두 C 라이브러리 소스에서 제공하는 함수들을 조사하고, 구현 시 주로 참조할 소스를 결정하기 위해 두 소스를 분석하였다. 분석 결과, GNU glibc-2.0.x는 지원하는 함수들의 종류는 다양하지만, 분석이 어렵고 코드가 너무 크다는 문제점이 있었다. 반면, 시그너스 사의 newlib는 함수 종류는 다양하지 않지만, 코드가 작고 쉬우며, 재진입 가능하도록 설계되었다는 장점이 있었다. 따라서, Q+용 C 표준 라이브러리의 요구 조건을 만족시키고 개발 시의 부담을 줄이기 위해 시그너스 사의 newlib를 주로 참조하고, 부족한 부분에 대해서는 GNU glibc-2.0.x와 Nucleus C[12] 라이브러리 함수들을 참조하였다.

2.2 함수 선택 기준

C 라이브러리 표준에는 시스템 API가 정의되어 있는 POSIX.1과 실시간 확장 API가 정의되어 있는 POSIX.1b가 있다. Q+ 커널에서는 POSIX.1b에 정의된 실시간 확장 API 중에서 일부 기능들을 지원하는데, 여기에는 clock 처리 함수, 메시지 큐 처리 함수, 스케줄링 관련 함수, 세마포 관련 함수, mutex 관련 함수 등이 있다. Q+ C 표준 라이브러리는 커널에서 제공하는 실시간 확장 API들은 제외하고, POSIX.1에 정의된 함수들을 포함한다.

POSIX.1[3-5, 11-14]에는 많은 수의 표준 함수들이 정의되어 있다. 하지만, Q+가 탑재되는 셋탑박스나 정보가전에서는 가용 자원과 응용 분야가 제한되어 있기 때문에, 불필요한 라이브러리 함수들의 탑재는 가용 자원의 크기를 줄여 응용 프로그램들의 반응 시간을 증가시킬 수 있다. 따라서, 적절한 기준에 의해 POSIX.1에 정의된 표준 함수들 중에서 실제 구현할 함수들을 선택할 필요가 있다.

Q+에 탑재될 C 표준 라이브러리 함수들의 선택은 크게 두 단계로 이루어 진다. 먼저, POSIX.1에 정의된 표준 함수들 중에서 Q+ 커널의 구현 방식과 디지털 TV 응용 분야에 적합하지 않는 함수들은 제외하였다. 예를 들어, Q+ 커널에서의 태스크 관리는 쓰레드(thread) 기반이므로, POSIX.1에 정의된 표준 함수들 중에서 프로세스에 관련된 함수들(예

를 들어, fork, exec, execl, kill)은 필요가 없다. 현재 Q+ 커널에서 쓰레드를 관리하기 위한 기능들(task_create, task_delete, et.all)을 별도로 지원하고 있다. Q+ 커널은 쓰레드 간의 기본적인 동기화 메커니즘을 지원하기 위해 이벤트(event)를 사용하고, 비동기화 메커니즘을 지원하는 시그널(signal)을 지원하지 않는다. 따라서, POSIX.1에 정의된 시그널 관련 함수들(sigaction, sigaddset, sigsend, et.all)도 C 표준 라이브러리에서 구현할 필요가 없다. 그리고, 디지털 TV 응용은 셋탑 박스를 통해 인터넷에 접속하고 다양한 미디어를 재생하기 위한 것으로, 기본 플랫 폼은 모두 그래픽 윈도우 매니저를 통해서 디지털 TV로 출력되고, 무선 리모콘이나 유·무선 키보드를 통해서 입력을 받게 된다. 따라서, POSIX.1에 정의된 다양한 터미널 설정 함수들(cfgetispeed, cfgetospeed, et.all)은 불필요하게 된다. 이러한 기준들을 적용하여 POSIX.1에서 불필요한 함수들을 제거하였는데, <표 1>은 그 결과를 보여준다.

〈표 1〉 첫번째 단계 함수 선택

표준 인터페이스 (POSIX.1)	구현 여부 (○/×)	이 유
입출력 함수	○	
터미널 제어 함수	×	디지털 TV를 통해 출력되므로, 다양한 터미널 제어 함수들은 불필요
오류처리 함수	○	
파일 시스템 관련 함수	○	
문자열 처리 함수	○	
메모리 관련 함수	○	
프로세스 관련 함수	×	Q+ 커널에서는 쓰레드 단위로 관리.
지역화 함수	○	
전역 점프 함수	○	
시그널 처리 함수	×	Q+ 커널에서는 시그널을 사용하지 않고, 이벤트를 사용.
시간 처리	○	
수학 함수	○	
기타 함수	○	

<표 1>에서 보듯이 Q+ 커널과 디지털 TV 응용 분야에 적합하지 않는 터미널 제어 함수, 프로세스 관련 함수, 시그널 처리함수 등이 1차적으로 제거되었다.

두 번째 단계에서는 현재 많이 사용되고 있는 실시간 운영체제들인 VxWorks[6, 7], pSOS[8, 9], VRTX[10], Chorus [11]에서 제공하는 C 라이브러리 함수들을 비교하여, 이들 운영체제에서 공통적으로 제공하는 함수들을 우선적으로 선정하였다. 예를 들어, fopen 함수는 VxWorks, pSOS, Chorus, VRTX에서 공통적으로 제공하기 때문에 Q+에서도 지원하지만, fnmatch 함수처럼 VxWorks, pSOS, Chorus, VRTX에서 제공하지 않는 함수들은 제거하였다. <표 2>는 두 번째 단계가 적용되는 예를 보여 준다.

첫번째와 두 번째 단계를 거쳐서 불필요한 함수들을 제거한 결과, POSIX.1에 정의된 표준 함수들 중에서 총 145개의 함수들이 선택되었다. 이들 함수들을 특성에 따라 입출력함수(파일 입출력 함수, 형식화 입출력 함수), 문자열처리 함수, 일반 유틸리티 함수(메모리 관련 함수, 기타 함수), 수학함수, 오류처리 함수, 시간 관련 함수 그룹들로 구분하였다. <표 3>은 각 함수 그룹에 포함된 C 표준 함수들을 보여준다.

〈표 2〉 두번째 단계 함수 선택

POSIX API	VxWorks	pSOS	Chorus	VRTX	Q+
close	○	close_f	○	○	○
fclose	○	○	○	○	○
fcntl	×	×	○	○	○
fdopen	○	×	○	×	×
fflush	○	○	○	○	○
fgetc	○	○	○	○	○
fnmatch	×	×	×	×	×

〈표 3〉 최종적으로 선택된 함수들

C 표준 라이브러리 함수		함수 명
입출력함수	파일 입출력	fclose, fflush, fgetc, fgets, fgetpos, fopen, fprintf, fputc, fputs, fread, creat, freopen, fscanf, fsetpos, fwrite, fileno, fseek, ftell, rewind, tmpfile, tmpnam, open, close, read, write, fstat, stat, fsync, sync, lseek, mkdir, rmdir, opendir, readddir, closedir, rename, remove, unlink, fcntl
	형식화 입출력	getc, getchar, gets, printf, putc, putchar, puts, scanf, setbuf,setvbuf, sprintf, sscanf, ungetc, vfprintf, vprintf,vsprintf
문자열 처리함수		strcat, strchr, strcmp, strcoll, strcpy, strcspn, strlen, strncat, strncmp, strncpy, strpbrk, strrchr, strspn, strstr, strtod, strtok, strtol, strtoul, strxfrm
일반유틸리티 함수	메모리 관리	calloc, free, malloc, memchr, memcmp, memmove, memset, realloc
	문자값 조사	isalnum, isalpha, iscntrl, isdigit, isgraph, islower, isprint, isspace, isupper, isxdigit, tolower, toupper
	기 타	assert, atof, atoi, atol bsearch, qsort, mbtowc, rand, srand
수 학 함 수		acos, asin, atan, atan2, cos, cosh, sin, sinh, tan, tanh, div, exp, fabs, floor, fmod, frexp, labs, ldexp, ldiv, log, log10,modf, pow, sqrt, abs, ceil
오류처리 함수		perror, ferror, strerror, feof, clearerr
시간관련 함수		asctime, ctime, difftime, gmtime, localtime, mktime, strftime, time

Q+에 탑재될 C 표준 라이브러리의 개발은 일단 <표 3>에 정의된 함수들을 우선적으로 구현하고, 추가적으로 요구되는 함수들은 추후 구현하기로 하였다.

2.3 시스템 호출로 정의된 함수

<표 3>에서 정의된 C 표준 라이브러리 함수들 중에서 18개 함수들이 시스템 호출로 제공된다. 시스템 호출 함수의 선택은 유닉스 시스템 V에서 시스템 호출로 정의된 C 라이브러리 함수 목록과 소스 분석 결과 나타난 Q+ 내의 다른 모듈(예를 들어, 커널이나 파일 시스템)에 대한 함수의 의존성(dependence)을 기준으로 하였다. 여기서, 의존성이란 특정 함수가 다른 모듈에서 제공하는 기능을 사용하게 되면, 특정 함수는 그 모듈에 대해 의존성이 있다고 정의한다. 시스템 호출 함수들은 대부분 파일 시스템과 커널 모듈에서 구현되어 제공되어 진다. <표 4>는 시스템 호출

로 제공되는 함수들과 파일 시스템 및 커널 모듈에서 정의된 시스템 호출들을 보여 준다.

〈표 4〉 시스템 호출로 정의된 함수

C 표준 라이브러리	시스템 호출	구현 서브시스템
open	sys_open	파일 시스템
close	sys_close	파일 시스템
read	sys_read	파일 시스템
write	sys_write	파일 시스템
fstat	sys_fstat	파일 시스템
stat	sys_stat	파일 시스템
fsync	sys_fsync	파일 시스템
sync	sys_sync	파일 시스템
lseek	sys_lseek	파일 시스템
mkdir	sys_mkdir	파일 시스템
rmdir	sys_rmdir	파일 시스템
opendir	sys_opendir	파일 시스템
readdir	sys_readdir	파일 시스템
closedir	sys_closedir	파일 시스템
rename	sys_rename	파일 시스템
remove	sys_emove	파일 시스템
ioctl	sys_ioctl	커널
fcntl	sys_fcntl	파일 시스템

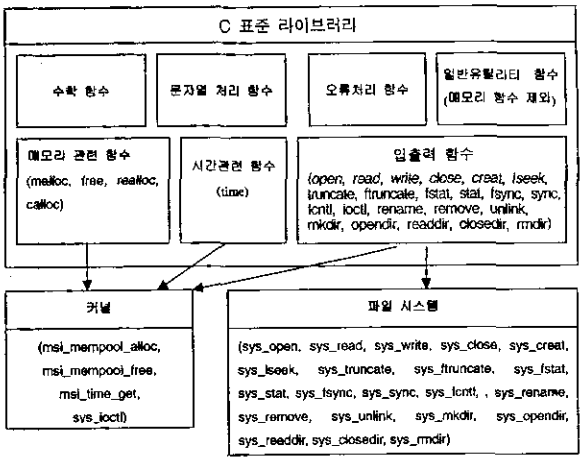
<표 4>에서 보듯이 커널이나 파일 시스템에서 제공되는 시스템 호출 함수들은 POSIX.1에 정의된 표준 인터페이스와 다르다. 이들 시스템 호출들의 인터페이스를 POSIX.1에 정의된 표준 인터페이스로 변환하기 위해, C 라이브러리에서는 각 시스템 호출 함수에 해당하는 템플릿(template) 함수를 정의하였다. (그림 1)은 open 템플릿 함수의 간단한 구조를 보여준다.

```
int open(…)
{
        int ret;
        ret = sys_open(..); /* call file systems function */
        if (ret < 0) { /* error occurred */
                errno = -ret;  /* set error number */
                return 1;
        } else
                return ret;
}
```

(그림 1) open 함수의 구조

2.4 타 모듈과의 인터페이스

Q+에 탑재될 C 표준 라이브러리는 필요한 시스템 호출 등을 커널과 파일 시스템 모듈에서 제공받아 응용 프로그램에게 필요한 기능들을 제공하게 된다. (그림 2)은 C 표준 라이브러리, 커널, 그리고 파일 시스템 사이의 인터페이스를 보여준다. (그림 2)와 같이, 입출력 함수들은 파일 시스템과 커널에서 제공하는 시스템 호출을 이용하여 필요한 기능을 수행하고, 메모리 관련 함수는 커널의 플랫(flat) 메모리 할당과 해제 루틴인 msi_mempool_alloc, msi_mempool_free를 호출하여 필요한 기능들을 제공한다. 그리고, 시간 관련 함수들은 시스템 클럭(clock)을 얻기 위해 커널에서 정의된 msi_time_get를 호출하게 된다. 하지만, 그 외의 다른 함수 그룹들인 수학 함수, 문자열 처리 함수, 오류처리 함수 등은 다른 모듈들과 별도의 인터페이스 없이 동작하게 된다.



(그림 2) C 라이브러리 인터페이스

3. 구현 환경

이 절에서는 Q+용 C 표준 라이브러리의 구현 환경에 대해서 기술하도록 한다.

C 표준 라이브러리의 구현 환경은 Q+ 커널과 셋탑박스의 개발 진도에 따라 크게 두 부분으로 나누어진다. 먼저 1999년 1월부터 1999년 10월까지는 아직 Q+ 커널과 셋탑박스가 개발되지 않았기 때문에, 상용화된 실시간 운영체제인 VxWorks 상에서 C 표준 라이브러리의 1차적인 구현 작업과 테스트 작업이 진행되었다. 1차 구현에 사용된 하드웨어와 개발 도구는 다음과 같다.

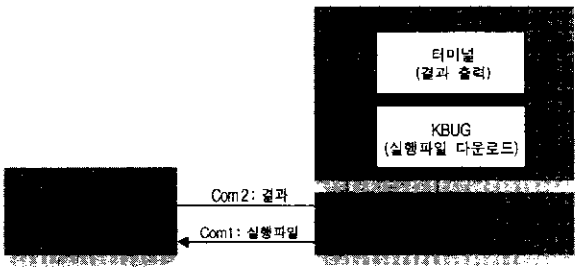
- 보 드 : ebsa-285 보드(StrongArm SA-110)
- 운영체제 : VxWorks
- 개발 환경 : Tornado 1.0
- 호 스 트 : PC-Pentium II
- 컴파일러 : Tornado Strong Arm Cross Compiler

개발 방식은 먼저 호스트 PC에 설치된 Tornado를 이용하여 소스를 개발한 다음, Strong Arm용 크로스 컴파일러를 이용하여 실행 파일을 생성한다. 그런 다음, ebsa-285 보드에서 동작하는 Tornado 타겟(target) 쉘에서 랜(LAN)을 통해 호스트 PC로부터 실행 파일을 다운로드하여 수행하게 된다. 수행 결과는 ebsa-285 보드와 시리얼로 연결된 호스트 PC 상의 터미널로 출력되게 된다. 1차 개발 단계에서는 Q+에 탑재될 C 표준 라이브러리 함수와 VxWorks에 탑재된 C 표준 라이브러리 함수에 동일한 입력 값을 주었을 때, 결과가 서로 동일한지를 비교함으로써 C 라이브러리 함수가 제대로 동작하는지 시험하였다.

2차 구현 작업은 1999년 11월부터 시작되었는데, 이때부터는 Q+ 커널 0.5와 셋탑박스 상에서 C 표준 라이브러리 개발이 이루어졌다. 2차 구현 환경에 사용된 하드웨어와 개발 도구는 다음과 같다.

- 하드웨어 : 셋탑박스(Strong Arm : SA-110, 8M Flash Memory, 32M RAM)
- 운영체제 : Q+ 커널 0.5
- 개발환경 : KBUG(Kernel Debug)
- 호 스 트 : PC-Pentium II
- 컴파일러 : GNU Strong Arm Cross Compiler

(그림 3)은 셋탑박스와 호스트 PC를 연결한 2차 구현 환경을 보여준다.



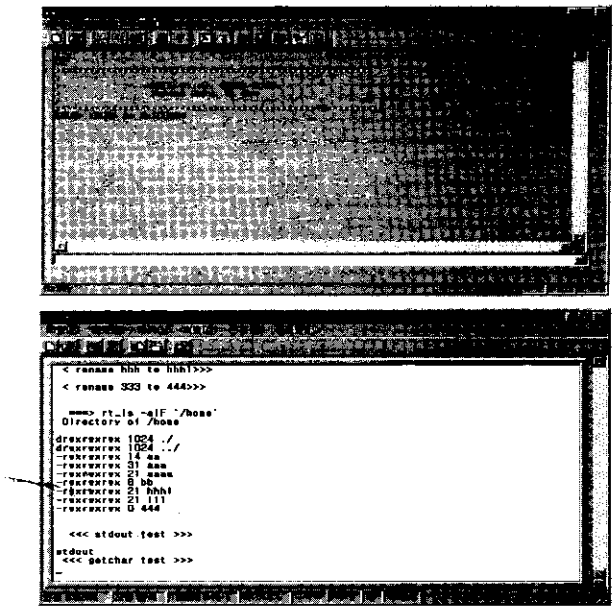
(그림 3) 2차 구현 환경

(그림 3)에서 보듯이 KBUG는 셋탑박스와 호스트 PC를 시리얼을 통해 서로 연결하여 실행 파일을 호스트 PC에서 셋탑박스로 다운로드할 수 있도록 한다. 실행 결과는 셋탑박스와 시리얼로 연결된 호스트 PC 상의 터미널로 출력된다.

개발 방식은 먼저 StrongArm용 크로스 컴파일러가 설치된 유닉스 기계에서 필요한 코드를 개발한 다음, .bin 형태의 실행 파일을 생성한다. 구현 작업에 사용된 유닉스 기계는 SunOS 5.5.1을 사용하는 Ultra-I 워크스테이션이다. 실행 파일을 생성한 다음, KBUG가 설치된 호스트 PC로 실행 파일을 옮기고, 다시 KBUG를 이용하여 실행파일을 셋탑박스로 다운로드 한다. 셋탑박스로 실행 파일을 다운로드한 다음, KBUG에서 시작 메모리 주소를 입력하게 되면 파일이 실행되고, 출력은 셋탑박스와 시리얼로 연결된 터미널을 통해 호스트 PC로 출력된다. 이때, 실행파일을 셋탑박스로 다운로드하는 주소(0x100000)와 실행 주소(0x100064)는 항상 일정하고, 하나의 실행파일을 수행한 다음에는 항상 셋탑박스를 리셋(reset)해야 한다. (그림 4)는 KBUG 수행 화면과 터미널을 통해 출력된 수행 결과를 보여준다.

4. 구 현

이 절에서는 Q+용 C 표준 라이브러리의 실제 구현 순서 및 추가로 정의된 함수들, 커널과의 통합 시 고려해야 할 사항, 그리고 C 표준 라이브러리의 특징 등에 대해서 기술한다.



(그림 4) KBUG 수행 화면과 출력 결과

4.1 구현 순서

C 표준 라이브러리 함수들 간에는 상호 함수를 호출하는 경우가 많기 때문에, 구현 시 다소 복잡한 면이 있다. 예를 들어, 함수 A를 개발한 다음, 함수 A가 제대로 동작하는지 조사하기 위해서는 함수 A가 호출한 함수 B가 제대로 동작하는 지 검증되어야 한다. 따라서, 함수들 간의 호출 관계를 이용하여 C 표준 라이브러리 함수들의 구현 순서를 결정하는 것이 필요하다. 함수간 호출 관계에 따른 구현 순서를 결정하기 위해서 newlib 소스를 분석하였다. <표 5>는 입출력 함수의 함수간 호출 관계를 간략히 보여 준다.

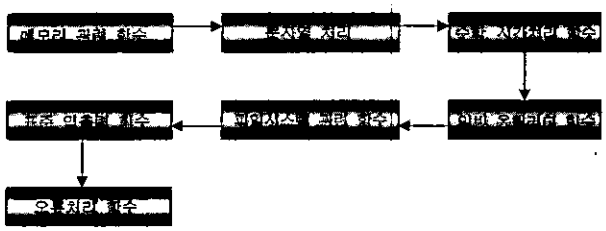
〈표 5〉 함수간 호출 관계

함수 명	라이브러리 함수	시스템 호출
fclose	fflush, free	_close
fflush		
fgets	memchr, memcpy	
fgetpos	ftell	
fopen	emset, malloc, fclosem	_open
fprintf	vfprintf	
fputs	fflush, memchr	_write
fread	memcpy	_read

<표 5>에서 보듯이 각 함수들 간의 호출 관계는 상당히 복잡하다는 것을 알 수 있다. 소스 분석을 통해 조사된 함수간 호출 관계를 기준으로 Q+에 탑재될 C 표준 라이브러리 함수 그룹들의 구현 순서를 (그림 5)와 같이 결정하였다. 물론, 각 함수 그룹 내에서도 함수간 호출 관계에 의해 구현 순서를 결정하였는데, 본 논문에서는 생략하도록 한다.

(그림 5)의 구현 순서에 따라 2000년 5월까지 1차로 C 표준 라이브러리의 개발을 완료하였고, 한달 간의 시험과 디버깅을 거쳐 2000년 6월에 알파버전을 릴리스(release) 하

였다. <표 6>은 1999년 11월부터 2000년 5월까지의 C 표준 라이브러리 개발 일정을 보여준다.



(그림 5) C표준 라이브러리 구현 순서

〈표 6〉 C 표준 라이브러리 개발 일정

일 정	함 수
1999.11~2000.1	메모리 관련 함수, 수학 함수, 문자열 처리 함수, 시간처리 함수
2000. 2~2000.3	일반 유틸리티 함수(메모리관련 함수 제외), 파일 시스템 관련 함수
2000. 4~2000.5	표준 입출력 함수, 오류처리 함수

4.2 추가로 정의된 함수들

C 표준 라이브러리에는 <표 3>에 정의된 표준 함수들 이외에 응용 API 모듈 등에서 필요한 기능들을 지원하기 위해, 추가적인 함수들이 다수 정의되어 있다. 이들 함수들은 크게 유니코드 문자열 처리 함수와 기타 함수들로 구분할 수 있다. Q+에 탑재될 그래픽 윈도우 라이브러리에서는 다국어를 지원하기 위해서 유니코드를 사용하는데, 응용 API 모듈에서 유니코드 문자열을 처리할 필요가 있어 유니코드 문자열처리 함수들이 필요로 하게 되었다. 유니코드 문자열처리 함수는 <표 3>에서 정의된 문자열 처리함수와 인터페이스 및 기능이 거의 동일하다. 기타 함수에는 특정 메모리 주소를 0로 초기화하거나, 복사하는 함수들과 입력 값에서 비트 값이 최초로 1인 위치를 찾아주는 함수 등으로 구성되어 있다. 유니 코드 문자열 함수와 기타 함수는 2000년 6월에 구현되었고, 이들 함수들은 <표 7>과 같다.

〈표 7〉 추가로 정의된 함수들

구 분	함 수 명
유니코드 문자열 처리 함수	unistrcasecmp, unistrcat, unistrchr, unistrcmp, unistrcoll, unistrcpy, unistrcspn, unistrdup, unistrlen, unistrncasecmp, unistrncat, unistrncmp, unistrncpy, unistrpbrk, unistrrchr, unistrspn, unistrstr, unistrtok
기타 함수	bfill, bzero, bcmp, bcopy, ffs, index, lower, rindex, toint, upper

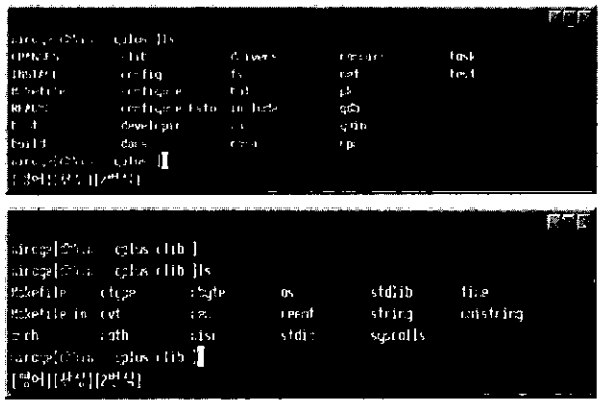
4.3 커널과의 통합

C 표준 라이브러리는 최종적으로 Q+에 탑재된 형태로 릴리스 되어야 한다. 개발된 C 표준 라이브러리를 Q+에 탑재할 때 고려해야 할 사항이 있는데, 그것은 C 표준 라이브러리에서 정의된 헤더 파일들과 데이터 타입들이 커널 및 다른 모듈들에서 정의된 헤더 파일들과 중복된다는 것이다. 예를 들어, Q+ 내의 다른 모듈에서도 stdlib.h, errno.h, types.h 등을 정의해서 필요한 함수나, 에러 번호, 그리고 데이터 타입 값을 정의해서 필요한 기능을 구현하게 된다. 따라서, Q+에 C 표준 라이브러리를 탑재하기 전에 먼저 이런 중복된 헤더 파일들과 데이터 타입들을 정리해야 할 필요가 있다. <표 8>은 다른 모듈과 중복되는 헤더 파일들을 간략히 보여준다.

〈표 8〉 중복되는 헤더파일

디렉토리	파 일
include	assert.h, ctype.h, errno.h, fcntl.h, math.h, stdio.h, stdlib.h, string.h, time.h, types.h unistd.h
include/sys	types.h

헤더파일과 데이터 타입의 정리가 Q+ 커널과 다른 모듈의 수행에 영향을 미치지 않도록 하기 위해, <표 8>의 헤더파일을 하나씩 정리할 때 마다 커널을 다시 컴파일 해보고, 커널 테스트 프로그램이나 다른 모듈의 테스트 프로그램들이 제대로 동작하는 지를 조사하였다. 위와 같은 작업을 통해 중복되는 헤더 파일과 데이터 타입을 정리한 다음, C 표준 라이브러리 알파 버전을 Q+ 소스 디렉토리에 2000년 6월에 첨가하고, Q+ 컴파일 시 C 표준 라이브러리를 같이 컴파일 하도록 Makefile을 수정하였다. 최종적으로, C 표준 라이브러리는 컴파일 후에 동적 라이브러리 형태(clib.a)로 생성된다. (그림 6)은 Q+에 합쳐진 C 표준 라이브러리의 소스 디렉토리를 보여준다.



(그림 6) Q+에 첨가된 C 표준 라이브러리 소스

4.4 C 표준 라이브러리의 확장 및 통합 시험

2000년 6월에 C 표준 라이브러리의 알파 버전이 릴리스된 다음, 2000년 11월까지 C 표준 라이브러리의 기능 확장 및 통합 테스트가 진행되었다. C 표준 라이브러리 알파버전은 플래시 파일 시스템 상에서 동작한다. 하지만, 플래시 파일 시스템은 멀티미디어 데이터를 저장하기에는 용량이 부족하여, 파일 시스템 모듈에서 디스크 파일 시스템을 새롭게 구현하여 2000년 7월에 릴리스하였다. 디스크 파일 시

스템 상에서 C 표준 라이브러리를 동작하도록 수정하는 작업이 2000년 8월까지 진행되었고, 2000년 11월까지 다른 모듈과의 통합 시험을 진행하였다. 마지막으로, 불필요한 코드의 삭제 및 함수 최적화 작업을 수행하고, 2000년 12월에 C 표준 라이브러리의 최종 버전을 릴리스하였다.

4.5 Q+용 C 표준 라이브러리의 특징

Q+에 탑재될 C 표준 라이브러리의 특징은 쓰레드들이 라이브러리 함수를 공유할 수 있도록 재 진입가능(reentrant) 하도록 설계되었고, Q+ 커널과 D-TV 응용 분야에 적합한 함수들을 포함하고 불필요한 함수들을 제거하였다는 것이다.

Q+가 탑재될 D-TV 셋탑박스의 기본 응용 분야는 가정에서 셋탑박스를 이용하여 인터넷에 접속하여 쇼핑을 하거나, 음악이나 동영상을 재생하고, 메일을 보내거나 받는 것이다. 여기서 셋탑박스의 기본 출력은 디지털 TV 이고, 입력은 무선 리모콘이나 유·무선 키보드가 된다. 이러한 응용 분야를 지원하기 위해 POSIX.1에 정의된 많은 수의 함수들을 지원할 필요가 없다. Q+에 탑재될 C 표준 라이브러리는 구현 시 POSIX.1에 정의된 함수들 중에서 Q+ 커널과 D-TV 응용 분야에 적합한 함수들과 다른 실시간 운영체제들이 공통적으로 제공하는 함수들을 우선적으로 구현하였다. 따라서, Q+용 C 표준 라이브러리는 다른 실시간 운영체제들 보다 함수 개수를 줄이면서도, Q+ 응용에 필요한 기능들을 제공할 수 있었다. 예를 들어, POSIX.1에 정의된 총 325개의 표준 함수들 중에서 VxWorks는 172개, pSOS는 147개, Chorus는 172개, VRTX는 175개를 구현하였지만, Q+는 145개의 함수들만 지원하게 된다. 이는 실제 생성된 라이브러리의 크기를 줄일 수 있어, 응용 프로그램의 크기를 줄일 수 있는 장점이 된다.

5. 결론 및 향후 과제

본 논문에서는 정보가전용 실시간 운영체제 Q+에 탑재될 C 표준 라이브러리의 설계 및 구현에 대해 기술하였다. Q+는 1999년 1월부터 2000년 12월까지 1차적으로 디지털 TV용 인터넷 셋탑박스에 탑재되는 것을 목표로 개발되었으며, 크게 커널, 라이브러리, 개발 도구, 응용 API로 구성된다. Q+용 C 표준 라이브러리는 POSIX.1에 정의된 표준 인터페이스를 따르고, 재진입가능 하도록 설계되었다. C 표준 라이브러리 개발에는 공개된 C 라이브러리 소스들이 참조되었는데, 주로 참조된 코드는 시그너스 사의 newlib와 GNU glibc-2.0.x이다. POSIX.1에는 많은 수의 함수들이 정의되어 있는데, 실제로 필요한 함수들을 선택하기 위해서 Q+ 커널과 디지털 TV 응용 분야에 적합한 함수들을 1차적으로 선정하고, 기존의 실시간 운영체제들이 공통적으로 제공하는 함수들을 위주로 2차로 선택하였다. 이렇게 선정된 C 표준 함수들은 Q+ 커널과 셋탑박스, 그리고 개발 도구인 KBUG 등을 이용하여 구현되었다.

앞으로 Q+는 디지털 TV용 셋탑박스 뿐만 아니라, 정보가전에 탑재되는 표준 실시간 OS로 확장될 예정이다. 따라서, C 표준 라이브러리는 정보가전 응용 분야가 요구하는 추가적인 함수들을 지원하도록 확장되어야 하고, 추가적으로 표준 C++ 라이브러리도 Q+에 탑재할 예정이다.

참 고 문 헌

[1] 한국전자통신연구원, "정보가전용 실시간 OS 컨퍼런스", RTOS'99 자료집, 1999.

[2] 한국전자통신연구원, "정보가전용 실시간 OS 컨퍼런스", RTOS 2000 자료집, 2000

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

[4] 'The GNU C Library Reference Manual', 4 Oct. 1996.

[5] 채신부, '개방형 운영체제 인터페이스(POSIX.1) 표준', 1993.

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

[7] ' VxWorks Training Workshop', Wind River Systems, 1996.

[8] 'pSOSystem System Calls', Integrated Systems, 1997.

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

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

[11] 'Standard C Library Function', Chorus OS man page, 1994.

[12] 'Nucleus C-Library Reference Manual', Accelerated Technology, 1993.

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

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

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

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

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

[18] 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

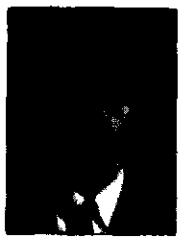
e-mail : dhkim@chinmi.etri.re.kr

1993년 경북대학교 컴퓨터공학과(공학사)

1995년 포항공과대학 전자계산학과 (이학석사)

1995년~현재 한국전자통신연구원 컴퓨터·소프트웨어기술연구소 선임연구원

관심분야 : 결함포용, 실시간시스템, 성능감시, 성능평가



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 컴퓨터소프트웨어기술 연구소 책임연구원

관심분야 : 실시간 운영체제, 네트워크 컴퓨팅