OSEK/VDX 운영체제 개발을 위한 커널 테스팅 기법

*임성락, *김재구, **유영창
*호서대학교 **(주)FALINUX
e-mail : srrim@hoseo.edu

A Scheme of Kernel Testing for the Development of OSEK/VDX OS

* Seong-Rak Rim, *Jae-Gu Kim, **Young-Chang Yu
*Dept. of Computer Engineering, Hoseo University
**Falinux co.,Ltd.

요 약

OSEK/VDX 운영체제를 개발하는 과정에서 커널 이미지를 생성하여 임베디드 보드에 업로드하고, 이를 실행시켜보는 일련의 반복적인 작업이 요구된다. 본 논문에서는 이러한 반복적인 작업들을 일괄적으로 처리할 수 있는 보다 편리한 커널 테스팅 기법을 제시한다. 제시한 기법의 타당성을 검토하기 위하여 ARM7TDM 가 탑재된 EZ-AT7 임베디드 보드에 OSEK/VDX 커널 이미지를 적용하여 실험하였다.

1. 서론

OSEK/VDX 운영체제는 자동차 전기 제어 장치(ECU)를 위하여 OSEK/VDX 에서 제안한 사양을 준수하는 실시간 운영체제이다[1]. OSEK/VDX 운영체제를 개발하는 과정에서 다음과 같은 일련의 작업 단계들이 요구된다.

[단계-1] 커널 소스 코드 작성
[단계-2] 커널 이미지 생성
[단계-3] 임베디드 보드로 커널 이미지 업로드
[단계-4] 테스트를 위한 커널 이미지 실행

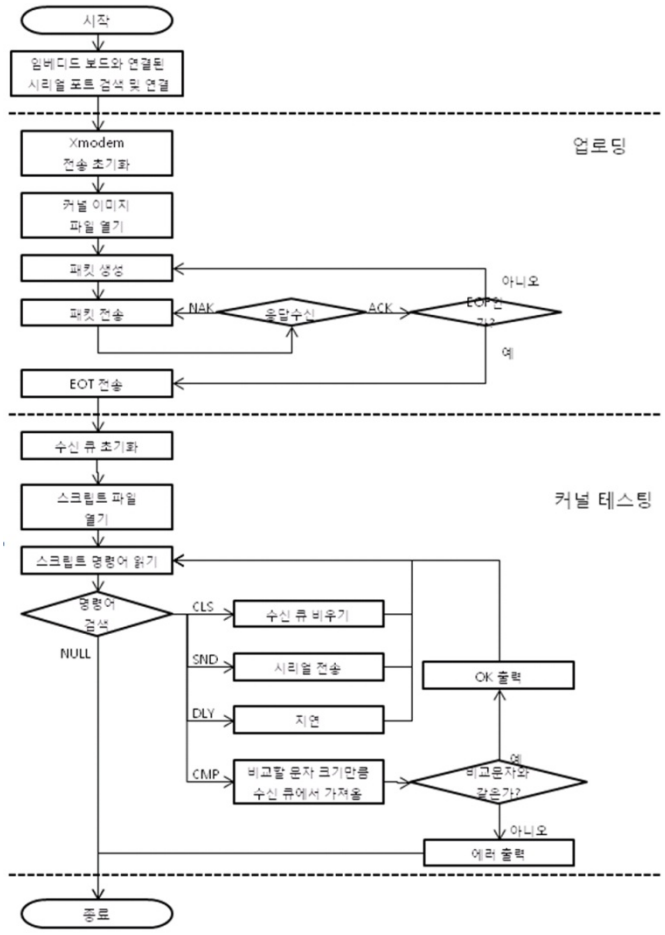
이러한 일련의 작업은 OSEK/VDX 운영체제를 개발하는 동안 반복적으로 수행되야 하며 테스트 결과를 확인하기 위해 시간이 많이 소요된다.

본 논문에서는 이러한 반복적인 작업들을 일괄적으로 처리할 수 있는 보다 편리한 커널 테스팅 기법을 제시한다. 제시한 기법의 기본 개념은 커널 테스팅을 위해 예상되는 커널 이미지 실행 결과 메시지를 스크립트 파일에 작성한 후 스크립트 파일의 내용과 비교하면서 커널 이미지를 실행시키는 것이다.

2. 설계

본 논문에서는 생성된 커널 이미지를 업로드 시키는 [단계-3]과 업로드된 커널 이미지를 실행시키는 [단계-4]의 작업과정을 일괄적으로 수행할 수 있도록 (그림 1)과 같이 커널 테스팅 기법을 설계한다.

(그림 1)에서 업로딩은 커널 이미지를 임베디드 보드에 업로드 한다. 이를 위해서는 임베디드 보드의 부트 로더와 이미지 파일 전송을 위한 파일 전송 프로토콜이 요구된다. 본 논문에서는 JBoot 부트로더[2]와 Xmodem 전송 프로토콜[3]을 이용한다.



(그림 1) 커널 테스팅 기법

(그림 1)에서 업로딩은 커널 파일 전송을 위해 임

베디드 보드에 전송 시작을 알리고, Xmodem 전송 초기화를 한다. 커널 이미지 파일을 열고 데이터를 읽어 패킷을 생성한 후 임베디드 보드로 전송한다. 그리고 임베디드 보드로부터 응답을 기다린다. 응답의 종류에는 다음과 같은 3 가지(ACK, NAK, CAN) 경우를 지원한다.

① ACK : 정상적으로 전송된 경우로써 EOF 인지 확인하고 아니라면 패킷 생성과정으로 이동하고 EOF 라면 임베디드 보드로 EOT 문자를 전송한 후 프로그램을 종료한다.

② NAK : 비정상적으로 전송된 경우로써 패킷 전송 과정으로 이동하여 이전 패킷을 다시 전송한다.

③ CAN : 더 이상 전송하지 않고 프로그램을 종료한다.

(그림 1)에서 커널 테스팅은 임베디드 보드로부터 커널 실행 결과를 수신 큐에 저장하고, 스크립트 파일에 저장된 예상 실행 결과와 비교하여 정상적으로 동작하는지 확인한다. 스크립터 파일로부터 명령어를 한 줄 읽기를 시도하고 명령어가 있으면 어떤 명령어인지 판단하고 더 이상 읽을 명령어가 없으면 처리를 중단한다. 스크립트 파일은 커널 테스팅 결과를 임베디드 보드로부터 수신하여 결과가 정상인지를 비교하기 위하여 명령어들을 순차적으로 작성한 것으로 본 논문에서는 명령어를 다음 같이 정의한다.

① CMP(Compare) : 수신한 문자열과 데이터 비교

② SND(Send) : 임베디드 보드로 데이터 전송

③ DLY(Delay) : 데이터 만큼 지연

④ CLS(Clear String) : 수신 큐를 비움

명령어	탭 문자	데이터

(그림 2) 스크립트 구조

3. 구현 및 실험

본 논문에서 제시한 기법을 실험하기 위해 ARM7 기반의 EZ-AT7[4] 개발 보드를 사용하며, PC 와의 통신을 위해 RS232C 시리얼 통신을 사용한다. 커널 테스팅은 JAVA 를 이용하여 구현했고, RS232C 통신을 이용하기 위해 javacomm2.0[5] API 를 JAVA 에 추가하여 다음과 같은 과정으로 실험한다.

[과정-1] (그림 3)과 같은 스크립트 파일을 작성한다.

```
CMP     os₩r₩n₩r₩n        ①
DLY     300
CMP     0 start₩r₩n
DLY     300               ②
CMP     2 start₩r₩n
DLY     300
CMP     OK₩r₩n
DLY     300
CMP     2 end₩r₩n
DLY     300
```

(그림 3) 스크립트 파일

(그림 3)은 스크립트 파일의 일부분이다. ①은 커널 실행결과와 비교하기 위해 커널의 예상되는 출력 결과를 적은 것이다. ②는 임베디드 보드로부터 모든 데이터가 전송되기에 충분한 시간만큼 대기하도록 한다.

[과정-2] (그림 4)와 같은 Makefile 을 작성한다.

```
all: bootstrap sg $(OBJS)
    arm-elf-gcc $(LD_CFG) -o $(TARGET).elf $(OBJS) $(LIBS)
    arm-elf-objcopy $(OC_CFG) $(TARGET).elf $(TARGET).bin_

    dd if=$(TARGET).bin_ of=$(TARGET).bin bs=1k conv=sync
    dd if=$(TARGET).bin_ of=$(TARGET).bin bs=1k
    java -jar $(TOPDIR)/tester/tester.jar -a $(TOPDIR)/$(TARGET).bin $(TESTER_DIR)/task_test_03.txt
```

(그림 4) Makefile

(그림 4)는 Makefile 의 일부분으로 ①은 컴파일러를 이용한 커널 이미지 생성부분이고, ②는 커널 이미지 업로드와 커널 테스트를 위한 프로그램 실행 문이다.

[과정-3] Makefile 실행 및 결과 확인

Makefile 실행하여 (그림 5)와 같은 이미지 생성, 커널 이미지 업로딩 및 커널 테스팅 결과를 확인한다.

(그림 5)는 실행 결과의 일부분으로 ①은 커널 이미지 생성과정으로 커널 소스를 빌드하여 바이너리 파일로 생성되는 부분이고, ②는 생성한 커널 이미지를 업로드 하는 과정으로 전송 진행 상황을 표현하기 위해 하나의 패킷이 전송 될 때마다 '.' 문자를 출력한다. ③은 업로드한 커널 이미지를 실행하여 테스트하는 과정으로 테스트 중에는 실행된 결과와 스크립트 파일을 비교하여 처리된 결과를 확인한다

(그림 5) 실행 결과

4. 결론

본 논문에서는 OSEK/VDX 운영체제를 개발하기 위해 커널 이미지를 생성하고 커널 이미지를 업로딩하여 테스팅하는 일련의 반복적인 작업을 일괄적으로 처리할 수 있는 보다 편리한 커널 테스팅 기법을 제시하고 그 타당성을 실험하였다.

참고문헌

[1] OSEK/VDX Operating System Specification 2.2.3, 2005. 2. (http://www.osek-vdx.org)
[2] http://www.falinux.com
[3] http://www.terms.co.kr/Xmodem.htm
[4] http://www.falinux.com
[5] Communication API(http://kr.sun.com)