VxWorks 운영체제 환경에서 개발 효율성 향상을 위한 디버깅 방법에 관한 연구

이원정*, 최윤석*, 주정현**
*삼성탈레스 해양/시스템 연구소, **국방과학연구소
e-mail:wj153.lee@samsung.com

Study for Effective Debugging Methodology in VxWorks Operating System

Won-Jung Lee*, Yoonsuk Choi*, Junghyun Ju**
*Samsung Thales, **Agency for Defense Development

요 약

윈도우즈, 리눅스 등 시장 점유율이 높은 운영체제의 경우에는 다양한 디버깅 도구들로 인해 개발 효율성이 향상된다. 반면 VxWorks 운영체제의 경우에는 실시간성이 요구되는 임베디드 환경에서 제한적으로 사용되어 디버깅 도구 부족 등 개발 효율성 측면에서 여러 제약사항들이 존재한다. 다양한 소프트웨어 오류 가운데 임베디드 환경의 제약으로 인한 대표적인 하드웨어 리소스인 CPU, 메모리와 관련된 디버깅 방안에 관해 소개하고 디버깅 효율성을 개선하고자 한다.

1. 서론

시스템 운용과정에서 CPU와 메모리 자원을 얼마만큼 사용하고 있는가를 파악하는 것은 소프트웨어를 개발하는 것만큼 중요한 일이다. 만약 CPU 사용률이 항상 100%로 유지되어 동작하고 있다면, 이는 고객으로부터 새롭게 추가되는 다양한 요구사항에 대해서 처리할 여력이 CPU 자원에는 더 이상 없음을 의미한다. 이를 개선하기 위해서 개발자들은 불필요하거나 중복되는 제어로직을 제거하여 소프트웨어를 보다 최적화시키거나, 또는 더 성능이 좋은 CPU 자원을 사용해야 할 것이다. 이는 메모리 자원에도 동일하게 적용된다. 시스템 운용과정에서 메모리 사용량을 지속적으로 관리함으로써 적정 수준의 메모리로 유지될 수 있도록 시스템을 안정화시켜야 할 것이다. 더욱이 제한된 자원에서 시스템을 운용하는 임베디드 환경의 소프트웨어 개발에는 이러한 제약이 더욱 엄격한 기준으로 관리된다. 고객의 요구사항들 중 시스템 운용과정의 CPU 및 메모리 사용률에 대한 제약조건을 관리하는 것도 이와 같은 맥락이다 [1].

C#, 자바와 같은 고차원적인 프로그래밍 언어를 사용할 경우 메모리 관리와 가비지 컬렉션을 언어차원에서 제공하므로 프로그래머가 메모리 관리를 고려하고 구현할 필요가 없다. 하지만 임베디드 환경에서 주로 사용되는 C나 C++와 같은 프로그래밍 언어의 경우에는 효율을 높이기 위한 목적으로 메모리 관리를 직접 해야 하기 때문에 메모리에 대해 고려가 필요적으로 요구된다. 비록 요즘 나오는 CPU는 대부분 가상 메모리 기능을 제공하므로 문제가 발생하더라도 해당 프로세스에 피해가 국한되긴 하지만, 전반적인 시스템 안정성을 저해하고 시스템 자원이 부족해지는 상황을 막기 위해서는 반드시 메모리 디버깅에 신경을 써야 한다.

메모리 관리 오류는 응용 프로그램과 시스템 동작에 예상하지 못한, 심지어는 치명적인 영향을 미칠 수 있다. 사용가능한 메모리가 감소함에 따라 프로세스와 시스템 전체를 정지시키거나, 손상된 메모리로 인해 시스템 전체가 알 수 없는 이유로 멈추는 경우도 발생한다. 시스템이 멈추지는 않더라도, 버퍼 넘침 현상으로 인해 시스템 보안이 취약해질 수도 있다. 오늘날 컴퓨터의 메인 메모리의 용량은 기가바이트 이상으로 장착되는 것이 일반적이므로 프로그램의 잠재된 소프트웨어 오류로 인한 메모리 누수 현상을 포함하고 있다면 응용 프로그램과 시스템에 증상이 나타나기까지 상당한 시간이 걸릴 것이다. 메모리 관리 오류는 기능 오류에 비해 쉽게 드러나지 않아 수정이 매우 어려운 경우가 많다.

실시간성을 보장하는 VxWorks 운영체제의 경우에는 커널이 사용하는 시스템 메모리 영역과 사용자가 사용하는 메모리 영역을 서로 공유한다. 이러한 조건에서는 메모리 관리에 대한 필요성이 더욱 커지고 응용 프로그램의 오류로 인해서 그 영향도가 전체 시스템까지 미쳐 운용이 불가능한 상태가 될 수도 있다. 이러한 VxWorks 운영체제 환경에서 CPU와 메모리에 대한 다양한 디버깅 도구를 소개하고 디버깅 효율성을 개선하고자 한다.

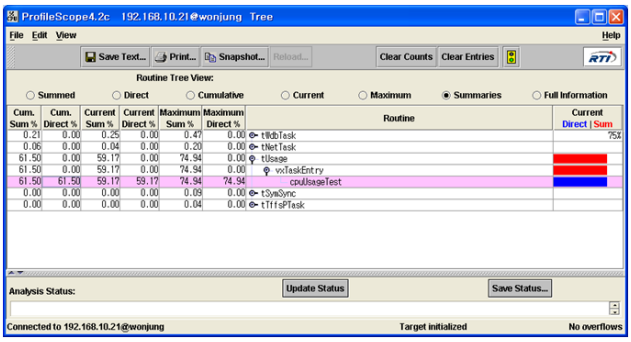
2. VxWorks 디버깅 방법

시스템 운용과정에서 VxWorks 운영체제의 CPU와 메모리 사용률을 확인하기 위한 디버깅 방법을 소개하고 GUI 기반의 디버깅 도구를 이용하여 개발자가 전체 응용 프로그램을 대상으로한 디버깅을 지원한다. 이러한 디버깅 도구는 네트워크 기반으로 타겟 시스템과 연결되어 상태 정보를 전달받아 표시한다.

가. CPU 사용률 디버깅

1) ProfileScope

ProfileScope는 Task, 프로세스, 함수 단위 통계치를 GUI 형태로 도식화하여 CPU 사용률을 보여준다. ProfileScope를 이용하여 이미 개발한 어떤 시스템의 성능이 미흡하여 소프트웨어적인 최적화가 필요한 경우, 모든 소프트웨어를 일일이 다 확인할 필요 없이 ProfileScope를 이용하여 특정 Task의 특정 함수가 성능 저하의 주요 요인인지를 확인하여 해당 부분만 집중적으로 최적화하는데 사용할 수 있다. 상세 정보는 현재값(Current), 누적값(Cumulative), 최대값(Maximum)을 확인할 수 있고, 누적 비율을 설정하여 디버깅이 가능하다. (그림 1)은 ProfileScope 실행화면을 나타낸다.



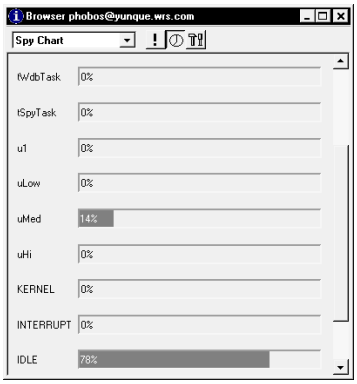
(그림 1) ProfileScope 실행화면

ProfileScope는 다양한 형태의 CPU 사용률 정보를 제공하며 다음은 각 정보에 대한 정의를 나타낸다[2][3][5].

- Current : 현 시점의 CPU 사용률 (%)
- Cumulative : 시작부터 누적된 CPU 사용률 (%)
- Maximum : 시작부터 가장 높은 CPU 사용률 (%)
- Direct : 서버루틴을 포함하지 않는 CPU 사용률 (%)
- Sum : 서버루틴을 포함하는 CPU 사용률 (%)

2) Spy 브라우저

Spy 브라우저는 VxWorks 구성 컨포넌트의 Spy 라이브러리를 이용하여 CPU 사용량을 측정한다. 이를 이용하기 위해서는 반드시 VxWorks 운영체제 프로젝트를 생성할 때 Spy 컨포넌트를 추가를 해서 컴파일을 수행해야 한다. Task 단위의 현재 CPU 사용률을 측정가능하고 누적값이나 최대값에 대한 정보는 확인할 수 없다. 따라서 전체적인 추이를 판단하기 위해 사용하는 경우 용이하게 사용가능하다. 추가기능으로는 현재 상태를 갱신하는 기능과 주기적으로 업데이트하는 기능 을 제공한다[4].

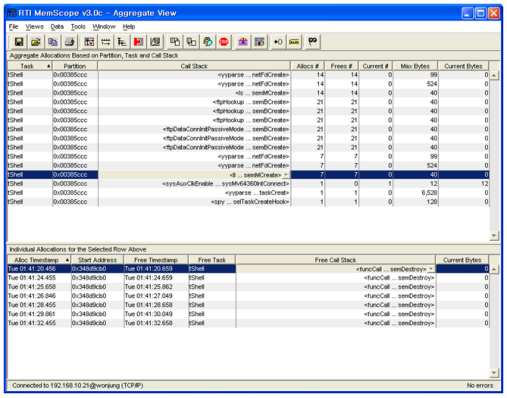


(그림 2) Browser의 Memory Usage 실행화면

나. 메모리 사용률 디버깅

1) MemScope

MemScope는 Task, 프로세스, 함수 단위 통계치를 GUI 형태로 도식화하여 메모리 사용률을 보여준다. 일반적으로 런타임 시에 시스템 메모리의 할당/해제를 반복하게 되는데 이때 구현상의 오류로 인해 정상적으로 메모리를 반환하지 않고 계속 할당만 하는 경우에는 시스템 메모리는 지속적으로 고갈되어 비정상적인 동작을 하게 될 것이다. 하지만 시스템 메모리의 용량이 크고 메모리 누수(Leakage)가 느린 속도로 발생하는 경우에는 메모리 고갈 상태가 상당한 시간이 경과한 이후에 발생하게 될 것이며, 디버깅을 위해 상황 재현에도 많은 시간이 소요되게 된다. 또한 계속적인 메모리 할당/해제에 의해 발생하는 메모리 단편화(Fragmentation) 상황을 확인할 수도 있다. MemScope는 메모리의 할당/해제 과정과 메모리 누수를 실시간으로 감시함으로써 시스템 전체 메모리의 동작상태를 확인할 수 있다. 상세 정보는 현재값(Current), 누적값(Cumulative), 최대값(Maximum)을 확인할 수 있고, 누적 비율을 설정하여 디버깅이 가능하다. (그림 2)은 MemScope 실행화면을 나타낸다.



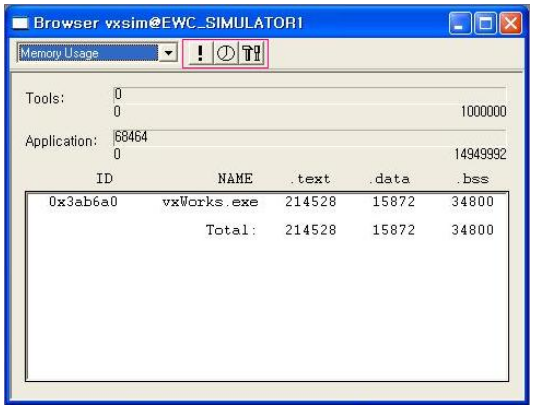
(그림 3) MemScope 실행화면

MemScope는 다양한 형태의 메모리 사용률 정보를 제공하며 다음은 각 정보에 대한 정의를 나타낸다[2][3][5].

- Current : 현 시점의 CPU 사용률 (%)
- Cumulative : 시작부터 누적된 CPU 사용률 (%)
- Maximum : 시작부터 가장 높은 CPU 사용률 (%)
- Direct : 서버루틴을 포함하지 않는 CPU 사용률 (%)
- Sum : 서버루틴을 포함하는 CPU 사용률 (%)

2) Memory Usage 브라우저

Memory Usage 브라우저는 시스템 전체의 메모리 사용률을 측정가능하고 누적값이나 최대값에 대한 정보는 확인할 수 없다. 따라서 전체적인 추이를 판단하기 위해 사용하는 경우 용이하게 이용가능하다. 추가기능으로는 현재 상태를 갱신하는 기능과 주기적으로 업데이트하는 기능을 제공한다[4].



(그림 4) Browser의 Memory Usage 실행화면

3. 결론

실시간성이 요구되는 임베디드 환경에서 사용되는 VxWorks 운영체제의 경우에는 윈도우즈, 리눅스 등과 같이 보편적으로 사용되는 운영체제에 비해 디버깅을 위한 도구가 부족하고 개발 효율성 측면에서 여러 제약사항이 존재한다. VxWorks 운영체제가 동작하는 임베디드 환경은 제한적인 하드웨어 자원을 이용하여 운용되며 대표적인 하드웨어 자원은 CPU와 메모리 자원이다. 소프트웨어 오류로 인해 CPU 혹은 메모리 사용률이 증가하는 경우에는 디버깅을 위한 상당한 시간이 요구된다. 이러한 CPU와 메모리 사용률을 시스템 운용과정에서 디버깅하기 위해 GUI 기반 도구인 ProfileScope와 MemScope를 통해 Task, 함수 단위로 분석함으로써 소프트웨어 오류 및 최적화가 요구되는 부분을 확인하여 개발 효율성을 개선한다.

참고문헌

[1] 박재호, "리눅스 개발자를 위한 디버깅 기법," 임베디드 월드, 03, 2006.

[2] Wind River, "Real-Time Visualization User's Manual 1.0".

[3] Wind River, "VxWorks API Reference".

[4] Wind River, "Tornado User's Guide".

[5] 백운철, "임베디드 시스템 개발을 위한 통합개발환경," 임베디드 월드, 09, 2005.

[6] 박재호, "리눅스 문제 분석과 해결," 에이콘 출판사, 09, 2006

[7] Yun Xia, Zhu Miaoliang, Yuan Shuhong, "THE IMPLEMENTATION OF FAULTS DEBUGGING AND DETECTION BASED ON VXWORKS," Computer Application and Software, 01, 2007.