리눅스 시스템에서의 고장 처리 프로세스 구현

*최영환, **조상영

한국외국어대학교

*cyon96@naver.com, **sycho@hufs.ac.kr

Implementation of fault handling process on Linux systems

Younghwan Choi, Sang-Young Cho

Hankook University of Foreign Studies

요 약

IT 분야는 최근 고도의 신뢰성을 요구하는 국방, 의료, 건설, 자동차, 선박, 항공의 다양한 분야와 융복합화가 이루어지고 있다. 본 논문은 고장 감내 기법을 적용하여 신뢰성 있는 임베디드 시스템을 구축하기 위한 고장 처리 프로세스의 구현에 대해 기술한다. 임베디드 시스템은 자원의 제약으로 인하여 기존의 자원 중복에 의한 결함 허용 기법을 적용하기 어렵다. 제안하는 고장 처리 프로세스는 하드웨어 또는 소프트웨어 중복 없이 시스템의 최악의 상황에서 동작하며 다양한 결함 허용 기법을 적용할 수 있도록 기본 동작 환경을 제공한다.

1. 서론

여러 산업 분야와 IT 기술이 융합하면서 국방, 의료, 건설, 자동차, 선박, 항공과 같은 고도의 신뢰성 및 가용성을 요구하는 분야에서 고장 감내 기술이 적용된 임베디드 시스템을 요구하고 있다.

고장 감내 기술은 전통적으로 인공위성을 포함한 항공 관련 분야와 금융계에서 사용하는 메인프레임급 시스템에서 많은 연구가 되어 왔으며 대부분 하드웨어 및 소프트웨어 자원을 중복하여 사용하거나 원자적 동작을 위한 로그 데이터를 기반으로 동작하고 있다. 임베디드 시스템은 서버와 달리 자원의 제약을 많이 받기 때문에 기존의 고성능 시스템에서 연구되었던 자원 중복에 의한 고장 감내 기술은 적용하기 쉽지 않다. 따라서 자원을 최소화하면서 고장의 진단, 격리, 복구하는 기법이 개발되어야 한다. 본 논문에서는 신뢰성과 가용성이 필요한 임베디드 Linux 운영체제의 고장 감내 기술을 위한 커널 단계의 고장 진단 및 복구 프레임워크 기술에 대해 기술한다. 특히, 최소한의 자원을 사용하며 시스템의 치명적 오류가 발생했을 경우에 기존의 다양한 고장 감내 기법을 적용할 수 있는 고장 처리 프로세스를 구동하는 기법 구현에 대하여 다룬다.

2. 임베디드 운영체제 고장 감내 연구

최근에 연구되고 있는 임베디드 운영체제를 위한 고장 감내 기술은 오류의 원인으로 가장 크게 작용하고 있는 디바이스 드라이버의 보호를 통한 복구[2], 커널 내부의 데이터 구조를 검사하여 파일 시스템, 메모리 손실, 커널 스택 등의 오류를 판별하며 복구하는 방식, 시스템 호출에 대한 강화된 인수 제공 및 인수 검사를 통한 오류 제거, 쓰레드의 형태에 따라 복구 형태를 달리하는 방식 등이 연구되고 있다[1]. 근본적으로 마이크로커널[3]을 사용한 가상화 기법을 이용하여 고장을 격리시키고 복구하는 방식도 있으며 고장 발생 시에 재부팅 또는 빠른 부분 재부팅을 통하여 복구하는 방식 등이 있다. 시스템의 강인성을 검사하기 위하여 고장 주입 기업에 대한 연구와 오류 발생 시의 원인을 파악하기 위한 유틸리티 등에 대한 연구가 있다. 그러나 이러한 연구들이 대부분 자원을 충분히 사용하는 시스템을 가정하여 연구가 되고 있으며 임베디드 운영체제의 특정 레이어나 부분적인 모듈에 국한되어 연구되고 있으며 소스 코드를 수정하여 개발되기 때문에 운영체제의 버전 갱신에 대처하기가 어렵다.

임베디드 시스템은 자원의 제약을 많이 받고 있기 때문에 자원의 사용을 최소화하면서 오류 검출, 오류 처리, 결함 처리, 시스템 복구 기법이 개발되어야 한다. 특히 임베디드 시스템의 특성상 하드웨어의 중복성을 사용하는 것이 쉽지 않다. 소프트웨어적인 접근 방법을 사용하며 가벼운 형태로 개발되어야 한다. 리눅스 커널은 다른 RTOS에 비하여 강력한 기능을 제공하면서 복잡한 구조를 가지고 있다. 근본적으로 모노리틱 커널의 형태를 취하고 있기 때문에 한 컴포넌트의 고장이 다른 컴포넌트에 전파되기가 쉽다. 따라서 정형적 방법을 사용하여 변경된 커널의 안정성을 증명할 수 없다. 리눅스 커널은 끊임없이 버전이 갱신되고 있으며 마이너 버전의 차이에도 소스의 구조가 많이 상이할 수 있다. 따라서 소스 전체에

영향을 줄 수 있는 고장 감내 프레임워크는 바람직하지 않다.

3. 고장 처리 프로세스 설계 및 구현

고장 처리 프로세스는 인터럽트 잠김과 같은 최악의 상황에서도 오류 검출 및 처리를 위하여 하드웨어를 장악하여 동작할 수 있어야 한다. 본 논문에서는 WDT(Watchdog Timer)의 기능을 이용하여 고장 처리 프로세스를 구현한다. 개발 환경은 MBA2440 보드를 사용하고 있으며 이 보드에서 사용하는 AP(Application Processor)는 삼성의 S3C2440이며 S3C2440은 ARM920T를 기본 코어로 사용한다. WDT는 프로세서의 동작이 잡음이나 시스템 에러에 의해 멈추었을 경우 또는 인터럽트가 마스크된 상태에서 무한 루프에 빠져 있을 경우에 리셋을 발생시켜서 다시 시스템을 초기화하고 동작 할 수 있게 한다. WDT 타이머는 보통 리셋의 발생 대신에 인터럽트를 발생시키는 것이 가능하다. 고장 처리 프로세스의 구현을 위하여 WDT로 하여금 시스템의 이상 상태에서 ARM 프로세서의 FIQ를 이용하여 FIQ 처리 루틴으로 분기하고 FIQ 분기 루틴을 고장 처리 프로세스로 사용한다. 이를 위하여 Linux 2.6.30.6을 MBA2440 보드에 이식하고 Linux 소스를 수정하였다. 시스템의 이상 유무를 판단하기 위해서는 WDT와 다른 타이머를 사용한다. 다른 타이머는 주기적으로 수행하며 WDT를 계속적으로 재초기화시키고 이상이 발생했을 경우에는 재초기화가 수행되지 않아 인터럽트를 발생시키도록 한다. 이를 위하여 소스의 수정은 다음과 같다.

리눅스에서 WDT의 구동은 장치 관리자 모듈로 만들어져 있다. 리눅스 커널은 초기화 과정 중에 machine 초기화를 하게 되고 이 과정에서 프로세서의 IRQ를 초기화하면서 추가적으로 WDT를 등록한다. 이후에 장치 관리자들을 초기화 하면서 WDT 장치 관리자 모듈을 초기화하고 WDT의 제어 레지스터 설정과 핸들러 루틴을 등록한다. 이를 위하여 linux/drivers/char/ 디렉토리의 watchdog/s3c2410_wdt.c와 linux/arch/arm/plat-s3c 디렉토리의 time.c가 수정되었다. 또한 S3C2440에서는 WDT가 AC97과 같은 인터럽트 번호를 쓰고 있기 때문에 AC97과 구별하기 위한 서브 인터럽트 번호를 설정해 준다. 갖고 있다.

WDT를 FIQ로 설정 하지만 실제 FIQ가 발생하였을 때 FIQ를 처리하는데 두가지 문제점이 있다. 첫째는 기본적으로 리눅스에서는 FIQ를 사용하지 않기 때문에 FIQ 핸들러 부분을 구현하고 있지 않다. 둘째는 FIQ 모드에서 메모리에 접근하려고 할 경우 data abort 예외가 발생하게 된다. 위 두가지 문제점의 해결책은 다음과 같다. 첫 번째 문제는 FIQ는 IRQ와 거의 유사하게 작동하므로 핸들러 부분도 IRQ 핸들러와 유사하게 새로 작성하여 추가하여 FIQ 핸들러 동작을 구현함으로써 해결할 수 있다. 두번재 문제는 원 소스에서 FIQ를 사용하지 않기 때문에 FIQ 모드에 대해 스택 영역을 지정하지 않아 FIQ 핸들러 내에서 스택을 사용할 경우에 잘못된 주소값을 참조하여 발생하게 된다. 따라서, FIQ용 스택 영역을 정의하고 이를 FIQ 스택 포인터에 설정함으로써 해결할 수 있다. 이를 위하여 linux/arch/arm/kernel/ 디렉토리의 entry-armv.S와 linux/arch/arm/kernel/ 디렉토리의 setup.c를 수정하였다.

4. 고장 처리 프로세스의 응용

구현된 고장 처리 프로세스는 시스템의 하드웨어 잠김과 같은 심각한 오류에서 FIQ를 이용하여 동작한다. 이를 이용하여 다양한 고장 감내 기법을 적용하여 시스템의 신뢰성을 높일 수 있다. 본 논문에서는 고장 처리 프로세스 응용의 하나로 부분 부팅을 통하여 빠르게 전체 시스템을 복구시키는 방법을 설계하였다. 빠른 재부팅을 위한 부분부팅 의 시작점은 여러 곳이 가능하다. 커널의 시작부터 init 프로세서가 동작하는 부분 사이의 특정 위치를 선택해야 한다. 시작에 가까울수록 재부팅 시간이 많이 걸리지만 코드의 수정이 없이 가능할 것이고 init 프로세스의 시작에 가까울수록 시스템의 동작에 의하여 수정된 부분을 다시 설정하여야 하기 때문에 부팅 시간은 단축할 수 있지만 기존 코드의 변경이 많아진다. 현재는 start_kernel() 함수 진입까지의 과정을 부팅없이 수행하도록 되어 있다.

5. 결론

본 논문에서는 신뢰성과 가용성이 필요한 임베디드 Linux 운영체제의 고장 감내 기술을 위한 커널 단계의 고장 진단 및 복구 프레임워크 기술에 대하여 기술하였다. 특히, WDT와 ARM 코어의 FIQ를 이용한 고장 감내 처리를 위한 고장 처리 프로세스를 설계하고 구현하였다. 고장 처리 프로세스는 기존 Linux 코드의 동작을 분석하여 수정하였으며 인터럽트 락킹에 의한 고장 시에 동작을 하며 다양한 고장 복구 기법을 적용할 수 있다. 고장 발생 시의 빠른 재부팅을 위한 코드 분석을 통하여 전체 부팅 과정에서 부팅 가능 시점을 정리하였으며 인터럽트 락킹 시에 FIQ 핸들러를 통한 재부팅이 가능하도록 하였다. 고장 처리 프로세스는 커널 단계의 고장 진단, 고장 격리, 고장 복구에 필요한 토대를 제공하며 추가적인 기술의 적용이 용이하도록 되었다.

참고문헌

[1] M. M. Swift, M. Annamalai, B. N. Bershad, and H. M. Levy, “Recovering Device Drivers,” ACM Trans. on Computer Systems, Vol. 24, No. 4, pp. 333-360, Nov. 2006.

[2] G. Heiser, "The Role of Virtualization in Embedded Systems", Proc. of Fisrt Workshop on Isolation and Integration in Embedded Systems, pp.11-16, Apr. 2008.

[3] F. M. David, R. H. Campbell, “Building a Self-Healing Operating System”, 3rd IEEE Int. Sym. On Dependable, Autonomic and Secure Computing, pp. 3-10, 2007.

본 연구는 지식경제부 정보통신산업진흥원의 ITRC 사업에 지원을 받았음. (NIPA-2010-C1090-1031-0004)