실시간 운영체제에서 메모리 누수 방지를 위한 메모리 모니터링 기법 설계 및 구현*

조문행[0], 최인범, 정명조, 이철훈
충남대학교 컴퓨터공학과
mhcho[0], ibchoi, mjjung, chlee@ce.cnu.ac.kr

The Design and Implementation of Memory Monitoring Mechanism for Preventing A memory leakage on Real-Time Operating Systems

Moon-Haeng Cho[0], In-Bum Choi, Myoung-Jo Jung and Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National University

요 약

임베디드 시스템은 임베디드 시스템의 공간제약 특성과 고가의 메모리 가격으로 인하여 한정된 메모리 자원을 가질 수 밖에 없어 메모리 자원의 효율적인 사용 및 관리가 필요하다. 임베디드 시스템에 탑재되는 실시간 운영체제는 위와 같은 특성을 고려하여, CPU 와 함께 운영체제의 중요 자원인 메모리를 효율적으로 관리할 수 있어야 한다. 본 논문에서는 실시간 운영체제에서 메모리 누수 현상을 방지하고 메모리를 보다 효율적으로 관리할 수 있는 메모리 모니터링 기법을 설계 및 구현하였다.

1. 서 론

최근 유비쿼터스 컴퓨팅, 차세대 컴퓨터와 같은 신성장 동력의 모든 핵심은 임베디드 시스템 분야이다. 특히, IT 부문에서 임베디드 시스템의 한정된 자원을 효율적으로 관리하기 위한 실시간 운영체제(Real-Time Operating System)에 대한 관심과 연구는 최근에 폭발적으로 증가하였다.

임베디드 시스템의 한정된 자원을 관리하는 실시간 운영체제는 CPU 와 함께 운영체제의 성능에 큰 영향을 미치는 메모리 자원을 보다 효율적으로 관리할 수 있는 기능을 지원함으로써 시스템 성능을 향상시킬 수 있다.

본 논문의 구성은 2 장에서는 관련연구로서 실시간 운영체제인 /RTOS™에서 사용하고 있는 메모리 관리 기법과 힙 스토리지와 메모리 풀에서의 메모리 누수 방지기법에 대해 소개하고, 3 장에서는 실시간 운영체제인 /RTOS™의 메모리 모니터링을 통해 보다 효율적으로 메모리를 관리하고 메모리 누수 현상을 줄일 수 있도록 설계 및 구현한 메모리 모니터링 기법을 제시하며, 4 장에서는 테스트 환경 및 결과를 기술한다. 마지막으로, 5 장에서는 결론 및 향후 연구과제에 대해서 기술한다 [1][2].

2. 관련 연구

태스크에 메모리를 할당하는 방법은 전역변수로 선언하는 정적 메모리 할당(Static Memory Allocation)과 실행 시간에 메모리를 할당하는 동적 메모리 할당(Dynamic Memory Allocation)이 있다. 전역변수를 통해 정적으로 메모리를 할당할 경우 실시간 운영체제의 실행이미지의 크기에 영향을 미치며, 운영체제에서 동적으로 관리가 불가능해져 메모리 누수로 이어 질 수 있다. 그러므로 실행 시간에 동적으로 메모리를 할당 받아 사용하는 것이 보다 관리하기 용이하다.

/RTOS™에서는 동적 메모리 관리를 위해 가변 크기 메모리를 할당하는 힙 스토리지 매니저(Heap Storage Manager)와 힙의 누수 메모리를 관리하기 위한 기법이 존재한다. 또한, 고정 크기의 메모리를 할당하는 메모리 풀(Memory Pools)과 메모리 풀의 누수 메모리를 관리하기 위한 기법이 존재한다 [3][4].

2.1 힙 스토리지 매니저와 누수 메모리 관리 기법

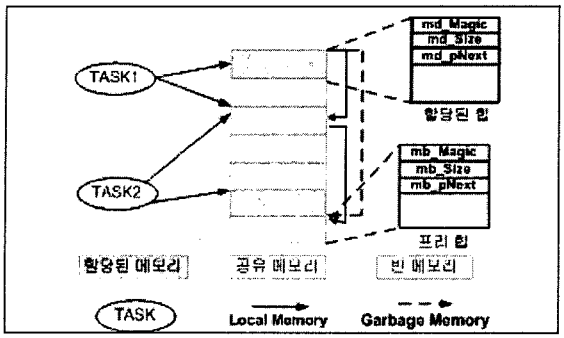
임의의 메모리 크기를 동적으로 할당하고 시스템 메모리의 힙(Heap)영역을 관리하기 위하여 힙 스토리지 매니저를 사용한다.

[그림 1]의 공유 메모리는 힙 영역에서 [그림 2]의 MK_GetMemory()를 통해 메모리를 할당하는데, 퍼스트 핏(First-Fit) 알고리즘에 따라 프리 블록 리스트로부터 적당한 메모리를 선택하여 할당하고, 공유 메모리가 아닌 메모리는 MK_GetLocalMem()를 통해 할당한다.

만약, 가용한 메모리가 존재하지 않으면, 메모리를 요청한 태스크는 가용한 메모리가 생길 때까지 대기(Pending)한다. 메모리 해제는 MK_FreeMemory()에 의

* 본 논문은 산업자원부의 지역전략산업 석,박사 연구인력 양성사업의 지원으로 수행된 것임

해 이루어진다.

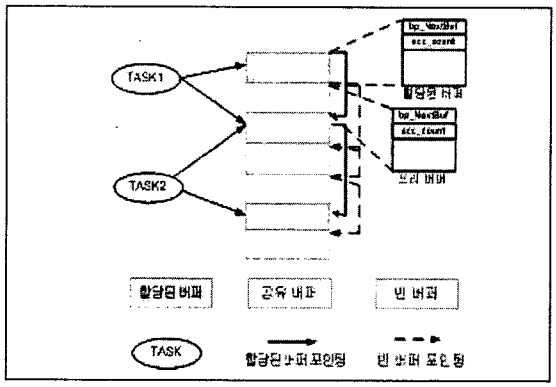


[그림 1] 힙 영역의 누수 메모리 관리

한편, 태스크가 종료(Termination)될 때 태스크가 직접 자신이 사용했던 메모리 영역을 스스로 제거할 수 있는 것이 아니고, 다른 태스크에 의해서 제거되어져야 하는데 그 제거 시점이 명확하지 못할 뿐만 아니라, 제거되지 않은 채 메모리에 남아 있는 경우가 발생한다. 이런 메모리 누수 현상을 방지하기 위하여, 공유메모리가 아닌 메모리를 MK_GetLocalMem()를 통해 할당하고, [그림 1]의 GarbageMemory 리스트를 생성한 후 MK_GarbageMemFree()를 통해 해제하여 메모리 누수 현상을 방지한다 [3].

2.2 메모리 풀(Memory Pool)

힙 스토리지 매니저를 통한 메모리 단편화(External Fragmentation) 문제와 가용 메모리 검색 시간으로 인한 시간 결정성 저해 문제를 해결하기 위해, 고정된 크기의 메모리 버퍼를 할당하는 메모리 풀을 사용한다.



[그림 2] 메모리 풀의 누수 메모리 관리

한편, 시스템 생성시 또는 특정 태스크의 필요에 의해 만들어진 메모리 풀은 메모리 풀의 해제 시점이 명확하지 않을 뿐만 아니라, 다른 태스크에 의해서 해제되어야 하는데 그 시점 또한 명확하지 못하여 해제되지 못하는 메모리 누수현상과 필요이상의 메모리 버퍼 생성을 통한 메모리 낭비 현상이 발생한다. 이런 문제를 방지하기 위하여 메모리 풀 생성시 할당 버퍼 비트맵과 접근 개수 비트맵, 버퍼의 접근 개수를 통해 메모리 누수 현상을 방지하고, 버퍼 합병을 통해 보다 효율적으로 메모리를 관리한다 [4].

3. 메모리 누수 방지와 효율적 메모리 관리를 위한 메모리 모니터링 기법

3.1 메모리 누수 현상

시스템 생성시 또는 특정 태스크의 필요에 의해 만들어진 공유 메모리 공간이 태스크가 삭제되었을 때 해제되지 않아 방치된 경우 누수 메모리가 되며 시스템의 성능 저하를 야기 할 수 있다.

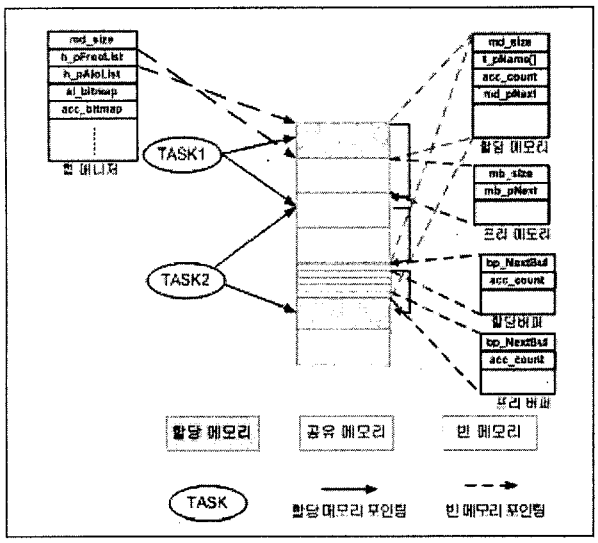
3.2 메모리 모니터링과 누수 방지 기법

메모리 모니터링이란 현재 시스템이 사용하고 있는 메모리 영역에 대한 모든 정보 즉, 시스템 전체에 할당된 메모리 크기와 특정 태스크가 메모리의 어느 지역부터 어떤 크기로 할당되어 있는지, 그리고 메모리가 공유메모리 인지에 대한 정보 등을 통해 시스템에 할당된 메모리를 관리할 수 있도록 하는 매커니즘이다.

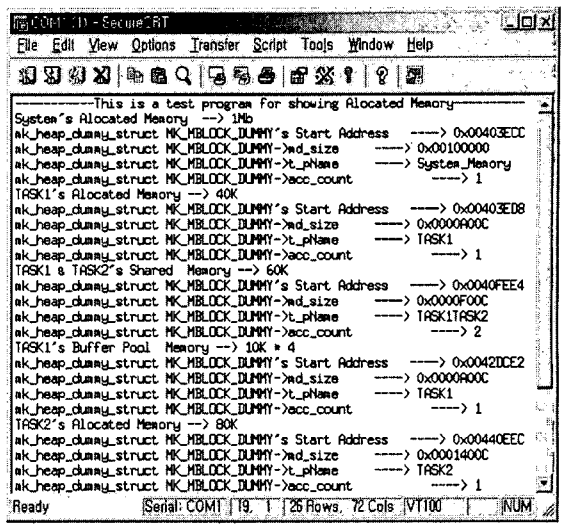
메모리 모니터링을 위해 전역 변수의 힙 스토리지 매너저를 두고 힙 스토리지 매니저에 할당된 메모리에 대한 포인터와 할당비트맵을 두어 메모리에 대한 할당 정보를 관리하고 공유 메모리인지를 판단하기 위해 접근 개수 비트맵을 둔다. 그리고, 할당된 메모리 영역을 관리하는 구조체에 태스크를 구분하기 위한 태스크 이름을 저장할 수 있는 필드와 할당된 메모리를 포인팅하기 위한 필드, 접근 개수를 계산하는 필드를 추가한다.

```
typedef struct mk_heap_struct {
    . . .
    Struct mk_heap_dummy_struct *h_pAlocMBlockList;
// 할당된 메모리의 포인터
    ULONG   al_bitmap;    // 할당 메모리 비트맵
    UINT    acc_bitmap;    // 할당 메모리 접근 개수
비트맵

} MK_POOL;
    // 힙 매니저 구조체에 멤버 추가

typedef struct mk_heap_dummy_struct {
    . . .
    CHAR   *t_pName[];     // 태스크들의 이름
    UINT    acc_count;     // 접근 개수
    struct mk_heap_dummy_struct   *md_pNext;
            // 할당된 다음 메모리의 포인터
} MK_MBLOCK_DUMMY;
   // 할당 메모리 관리 구조체에 멤버 추가
```

[그림 3] 메모리 모니터링을 위한 구조체와 함수



[그림 4] 메모리 모니터링을 통한 메모리 관리



[그림 5] 테스트 결과

힙 매니저에는 시스템 전체에 할당된 메모리 크기인 md_size 와 프리 메모리 영역과 할당된 메모리 영역을 관리할 수 있는 필드들을 통해 메모리 모니터링을 할 수 있다.

[그림 4] 에서와 같이 Task1, Task2 가 힙 매니저에서 가변메모리로 할당된 영역과 버퍼 풀로 할당된 메모리 영역이 있을 경우, 힙 매니저에서는 al_bitmap 을 통해 할당된 메모리 영역에 대한 위치와 h_pAloList 포인터를 통해 할당된 메모리 영역에 순차적으로 접근하여 어떤 태스크에 어떤 크기로 할당되었는지에 대한 정보를 알 수 있으며, acc_count 가 2 이상이 되면 set 되는 acc_bitmap 을 통해 공유 메모리인지에 대한 정보를 파악할 수 있다. 그리고, 파악된 정보를 바탕으로 힙 매니저의 acc_bitmap 의 필드가 0으로 set 되어 진 곳이 메모리 해제가 이루어 지지 않았을 경우, 메모리 누수로 파악하여 특정 시점에 누수된 메모리를 해제할 수 있다.

4. 테스트 환경 및 결과

본 논문에서 구현한 실시간 운영체제의 메모리 모니터링기법은 삼성에서 제작한 ARM940T 기반의 S5H5002 Evaluation Board 에 *RTOS™*를 탑재하여 구현하였으며, 컴파일러는 Metrowwerks 사의 Code Warrior 를 사용하였다.

[그림 5]의 테스트 결과를 통해 어느 메모리 위치가 어떤 크기로 어떤 태스크가 사용하고 있는지에 대한 명확한 정보를 파악할 수 있으며, 그 정보를 바탕으로 메모리 누수현상도 방지할 수 있다.

5. 결론 및 향후 연구 과제

본 논문에서는 실시간 운영체제의 메모리 관리에 있어서 메모리 관리를 보다 명확하게 하는 메모리 모니터링 기법과 이를 통해 시스템에 할당된 메모리에 대한 정보 파악과 이 정보를 바탕으로 메모리 누수현상을 방지할 수 있는 기법에 대해 설계하고 구현한 내용을 기술하였다.

향후 연구과제로는 실시간 운영체제에서 시간 결정성을 보장하면서 보다 효율적으로 메모리를 관리할 수 있는 기법과 메모리 모니터링 기법을 통해 파악된 정보를 바탕으로 특정 메모리 뱅크를 On/Off 하여 메모리에서 사용되는 전력을 줄이는 DPM(Dynamic Power Management)기법에 대한 연구를 할 것이다.

참고문헌

[1] Embedded System & RTOS, http://www.inestech.com.

[2] David Lafreniere, "An Efficient Dynamic Storage Allocator", Embedded Systems Programming, Sept. 1998.

[3] 조문행, 양희권, 이철훈, "Implementation for Preventing A memory leakage on Real-Time Operating Systems", 한국정보과학회, 제 31권 제 1호, p.163-165, 2005.04.

[4] 조문행, 정명조, 이철훈, "The Design and Implementation for Preventing A memory leakage of Memory Pool on Memory Management of Real-Time Operating Systems", 한국정보과학회, 제 31권 제 1호, p.163-165, 2005.04.