'91 韓國自動制御學術會議論文集 1991. 10.22~24

실시간 운영 체제의 구현

°박 병현 이 진수

포항공과대학

(Implementation of Real Time Operating System)

J.S. Lee °B.H. Park

Dept. of Eletronic and Eletrical Engineering

POSTECH

ABSTRACT

We propose a real time kernel chimera implemented under AT&T UNIX motorola versoin. Carnegie Mellon Univ.in U.S first developed chimera using SUN Worstation with Berkley UNIX based on VMEbus. The major differences between Canegie Mellon's and ours are downloading program and communication method between host and target. Original chimera used device driver but we used UNIX system call corresponding to shared memory when user downloads program and communicates. We modified kernel itself because the two different UNIX have different link editor.

1. 서 론

마이크로프로세서(microprocessor)의 속도가 증가하고 그 기능이 다양해짐에 따라 기존의 아날로그 컨트롤러(analog controller)가 디지탈(digital)화 하고 있다. 그리고 프로세스(processor)의 속도가 I/O를 처리하는 속도보다 훨씬 빨라짐에 따라 다중처리를 실시간으로 하는 것이 가능하게 되었다. 기존의 운영체제인 도스(DOS)는 다중처리의 기능이 없다. 그리고 다중처리가 가능한 유닉스(UNIX)나 OS/2 등은 실시간 처리를 하기에는 기본적으로 커널(kernel)의 양이 커서 유저 프로세스(user process)에 할당하는 시간이 작고 prority도 아주 낮다. 뿐만 아니라 실시간 프로그래밍(programming)를 할 수 있는 유틸리티(utility)등이 거의 없다. 그리고 일반적으로 제어에서 가장 많이 사용되어지는 방법으로 인터럽트(interrupt)에 의한 방법이 있는데 이것은 실시간처리에 있어서 가장 확실한 방법이기는 하나 다음과 같은 문제점이 있다. 즉 가장 간단한 예로 외부로부터 들어온 이벤트(event)를 처리하는 시간이 제어 루프(control loop)의 한 주기보다 길게 되면 제어에 심각한 영향을 미칠 수 있다는 것이다.그것은 프로그램이 이미 짜여진 순서에 의해서만 수행되기 때문이다. 그리고 데이터 버퍼(data buffer),큐(queue),플래그(flag)등을 여러 개의 인터럽터 핸들러(interrupr handler)가 사용할 경우 그것들에 대한 사용을 프로그래머가 대단한 주의를 기울여서 프로그래밍해야 한다. 이것은 힘든 일이고 뿐 아니라 많은 시간이 요구된다.그리고 신뢰성등 여러가지 문제점을 안고 있다.

실시간 운영 체제는 위와 같은 문제점을 극복할 수 있어야 한다.그러기 위해서 먼저 실시간 다중 운영 체제는 중앙연산 장치(CPU)의 효율을 극대화 하기 위해서 다중 처리를 기본으로 한다. 그리고 커널은 유닉스(UNIX)나 OS/2 등에 비해 다음과 같은 특징을 가지고 있어야한다. 첫째 커널의 운영 시간이 작아야 한다.핵심적으로 컨텍스트 스위칭 타임

(context switching time)이 짧아야 한다. 둘째는 각각의 타스크(task,process)간의 통신과, 호스트 컴퓨터(host computer)와 타겟 컴퓨터(target cpmputer)간의 통신이 가능해야 하고 타스크간의 동기화(syncronization)을 지원할 수 있어야 한다. 셋째로 시간의 제약성에 대하여 적절이 대응할 수 있는 타스크 스케쥬링(task scheduling)과 필요에 따라 사용자가 스케쥴링 방법을 바꿀 수 있는 유연성이 필요하다. 키메라(chimera)는 이러한 조건을 모두 만족한다.

또, 실시간 소프트웨어(software) 개발 환경으로 또스나 유닉스 같은 기존의 운영 체제이어야 한다. 본 논문에서 제안하는 실시간 다중 운영 체제는 유닉스를 그 개발 환경으로 하고 있다. 기존의 운영 체제가 그 개발 환경이 되어야 이미 개발된 많은 응용프로그램(application program)이나 라이브러리(library)등의 사용이 간편하기 때문이다. 그리고 유닉스를 사용하는 경우는 멀티유저(muliti-user)시스템을 구성할 수 있게 된다. Wind River System에서 만든 VxWorks나 Ready System에서 VRTX등도 이와 같은 형태를 갖추고 있다. 기존의 운영체제를 개발환경으로 채택하는 경우 호스트 컴퓨터(host cpmputer)에서 타겟 컴퓨터(target cpmputer)로 다운로드(download)하는 프로그램의 개발은 꼭 필요하다. 본 프로그램에서는 AT& T 유닉스 모토롤라 버전(sysV68)에서 제공하는 공유 메모리(shared memory) 방식을 이용해서 다운로드 프로그램을 개발했다.

하드웨어 개발환경으로는 모토롤라 델타 시스템(motorola delta system)을 이용한다. 델타 시스템은 VMEbus를 기본으로 하기때문에 새로운 기능을 하드웨어에 추가하기가 용이하다. 그리고 호스트와 타겟이 버스(bus)로 연결되어 있기 때문에 서로의 통신이 빠른 속도로 이루어질 수 있다. 호스트는 모토롤라사에서 나온 mvme147을 이용하고 타겟은 mvme143을 이용한다. 타겟은 모토롤라사의 68000계열의 CPU를 탑재한 보드이면 어떤 것이라도 사용이 가능하다. 혹은 그렇지 않은 경우는 크로스 컴파일러가 있으면 된다.

본 논문은 chimera의 전체적인 구성과 기능에 대한 설명,그리고 다운로드 프로그램에 대한 설명으로 구성되어 있다.

본 론

1. 하드웨어

하드웨어의 대체적인 그림은 아래와 같다.

DELTA SYSTEM H/W STRUCTURE

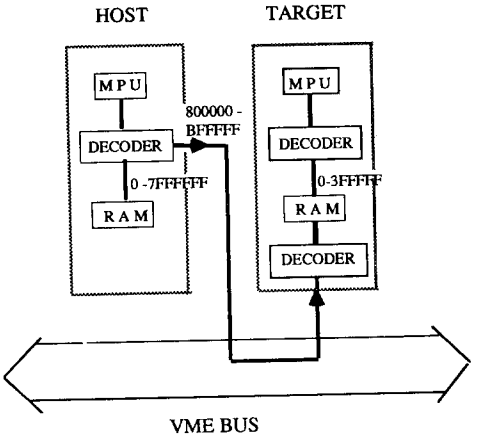

그림 2 - 1

버스상에 존재하는 호스트의 물리적인 주소는 0x0에서 0x399999이고 타겟인 mvme143은 버스상의 0x800000에서 0xb99999 번지를 할당받는다. mvme143은 버스상에 존재하는 자신의 주소를 143bug라는 모니터 프로그램을 통해서 소프트웨어적으로 정할 수 있다.

2. 다운로더에 대하여

버스를 통해서 화일을 다운로드하는 경우 가장 중요한 사항은 버스에 연결된 타겟의 물리적인 주소(physical address)를 유닉스의 논리적인 주소(logical address)를 통해서 액세스하는 것이다. 그리고 여기서는 공용 메모리를 이용해서 액세스한다.최근 국내에서 많이 쓰이는 VMEexec또 kernel을 다운로드하는데 이런 방식을 쓰고 있다. 서론에서 언급한 바대로 메모리를 곧장 액세스하기 때문에 빠른 속도로 다운로드하는 것이 가능하다. 공용 메모리에 의해 다운

로드를 하기 위해서는 sysV68이 제공하는 시스템 콜인 shmget, shmatt, shmdt, shmctl에 대해서 이해해야 한다. 각각의 기능을 간단히 설명하면 아래와 같다.

shmget 사용자가 원하는 영역에 공용 메모리 영역을 선언한다. 그러면 공용메모리에관한 ID를 리턴한다. 이때 공용 메모리 영역은 논리적인 영역이거나 물리적인 영역일 수도 있다. 단 물리적인 영역이려면 슈퍼 유저(super user)이어야한다.

shmatt shmget에서 선언한 공용 메모리 영역에 대한 논리적인 주소의 시작 번지를 리턴한다.

shmdt 공용 메모리 영역에 대한 액세스가 끝났음을 나타낸다.

shmctl 이미 선언된 공용 메모리의 인수를 바꾸거나 ID를 없애는데 쓰인다.

위에서 shmget을 이용하면 타겟에 있는 버스상의 물리적인 주소에다 공용 메모리 영역을 선언할 수 있다. 이때 공용 메모리에 관한 ID를 리턴한다. 이 ID를 shmat에 이용하면 타겟에 있는 공용 메모리 영역의 물리적인 시작 주소에 대응하는 논리적인 시작주소를 리턴한다. 리턴된 논리적인 주소를 통해서 타겟의 물리적인 주소를 액세스한다. 이렇게 하면 호스트(유닉스)에 있는 화일을 타겟의 임의의 번지에 다운로드 할 수 있게 된다.

sysV68 환경하에서 화일을 다운로드할 때 타겟의 메모리 영역에 맞게 화일의 메모리 맵을 손질할 필요가 있다. sysV68에서 제공하는 컴파일러인 cc를 이용해서 만든 실행화일인 a.out 은 호스트의 메모리 맵에 맞게 만들어 진다. 그래서 타겟의 메모리 맵에는 맞지 않는 경우가 생긴다. 이때는 sysV68에서 제공하는 링크 에디터(link editor)를 사용한다.

3. 커널에 대하여

1)커널이 제공하는 기능

a. 가장 기본적인 기능으로 다중 처리가 가능하다. 간단한 예제를 보이면 아래와 같다.

```
int variable1,variable2

hello( )
{
  print("hello world \n") ;
  print(" variable1 = %d \n",variable1) ;
}
bye( )
{
  printf("Goodbye  \n") ;
  printf("variable2 = %d\n",variable2) ;
}
main( ) {
{
   printf("Main Begins\n") ;
   variable1 = 10 ;
   variable2 = 20 ;
   spawnd(hello)  ;
   spawnd(bye)    ;
   printf("Main Ends \n") ;
 }
```

위의 프로그램을 실행시키면 다음과 같은 출력을 얻을 수 있다.

```
Main  Begins
Main  Ends
Hellow World
```

```
Goodbye
variable = 10
variable = 20
```

위의 프로그램에서 가장 중요한 것은 spawnd라는 함수 호출에 의해서 hello와 bye라는 새로운 타스크가 생겨났다는 것이다. 그리고 그것들이 각각 독립적으로 수행되었다는 것이다. 그리고 그것을 출력을 통해서 확인할 수 있다.

b.호스트에서 실행되고 있는 서버를 통해서 유닉스의 화일 시스템을 액세스할 수 있게 한다. .

c.C 언어로 짜여져 있기 때문에 글로벌(global) 변수를 이용하면 타스크간의 통신을
빠른 속도로 할 수 있다. 그리고 하드웨어가 버스에 연결되어 있으므로 다른 보드끼리의 통신도 버스를 통해서 빠른 속도로 할 수 있다.

d.세마포어(semaphore)기능을 제공하므로 타스크간의 동기화가 가능하다.

e.타스크의 스케쥴링 알고리즘에 대한 선택이 가능하므로 상황에 따라 적절한 방법을 선택할 수 있다.

f.타스크의 우선순위를 프로그래머가 임의로 할당할 수 있다.
g.메모리를 다이나믹(dynamic)하게 할당(allocation)수 있다.

h.유닉스에서 쓰이는 대부분의 함수들을 지원한다.

2).커널의 구현 방법에 대하여

커널은 타스크 스케쥴러에 관한 부분과 컨텍스트 스위칭에 관한 부분 그리고 메모리 할당에 관한 부분, 그리고 호스트와 통신하는 부분들로 이루어져있다.

a.스케쥴러

스케쥴러는 타스크에 관한 모든 자료를 관리한다. 그리고 그 자료에 의해서 CPU의 서비스를 받을 타스크를 정하는데 그 방법으로서는 세가지가 있다. 라운드로빈(round-robin)에 의한 것,우선 순위에 의한 것 ,가장 시급한 것을 먼저 수행하는(minimun-laxity-first)것인데 이중 실시간 처리에 가장 적합한 가장 시급한 것을 먼저 수행하는 것(minimun-laxity-first)를 설명한다. 이 방법은 사용자가 시간의 한계를 정해놓는다. 그러면 스케쥴러는 한계를 지난 타스크가 있으면 그 타스크를 가장 우선적으로 수행한다. 그리고 시간의 한계를 지난 타스크가 없으면 우선 순위에 의한 스케쥴링을 한다. 그리고 우선 순위가 같으면 라운드 로빈에 의한 스케쥴링을 한다.

b.컨텍스트 스위칭

이 부분은 CPU에 따라 달라지는 부분으로 가능한 한 짧아야 한다. 컨텍스트 스위칭에 관한 데이터는 각각의 타스크를 관리하는 자료구조(data struct)안에 스택으로 저장되어 있다. 타이머에서 인터럽터가 걸리면 다른 인터럽터는 마스킹(masking)되고 스케쥴러 루틴을 부른다. 스케쥴러로부터는 다음에 수행될 타스크에 대한 정보가 리턴된다.

c.메모리 할당 부분

메모리 할당은 커널이 다운로드된 뒷 부분부터를 이용한다. 메모리 할당을 위한 자료구조는 링크드리스트(linked-list)를 이용한다. 타스크 관리를 위한 자료 구조의 메모리도 메모리 할당을 통해서 확보한다.

d.호스트와의 통신

호스트와의 통신은 공용 메모리 방식을 이용한다. 호스트의 화일에 액세스하는 것도공용 메모리 방식을 이용한다.

3. 결 론

여러 가지 제어에서 컴퓨터가 담당하는 부분은 더욱 커지고 있다. 그리고 그 역할도 복잡해지고 있다. 따라서 제어에서 여러가지 일을 실시간으로 처리하기 위해서는 신뢰 성있고 다양한 기능을 제공하는 실시간 운영 체제의 개발이 필수적이라고할 수 있다. 본 논문에서 재안하는 실시간 운영체제는 기본적인 기능을 갖추고 있지만 보완해야할 부분이 많다. 특히 사용자 프로그램을 수시로, 커널과는 따로 다운로드 할 수 있는 기능과 다른 컴퓨터와 네트워크(network)기능을 갖추는 것이 보완해야할 가장 중요한 부분이라고 생각한다.

참 고 문 헌

[1] Mauriche ,J. Bach , "The design of the UNIX Operating system",Prentice hall,inc.,Englewood Cliffs,Newe Jesry

[2] David B. Stewart, Donald E.Schmitz, Pradeep K.Khosla, "CHIMERA II ,Program documentation",Advanced Manipulators Laboratory, The Robotics Institute and Department of Electrical and Computer Engineering, Carnegie Mellon Univ., PIttsburgh,Pennsyvania.

[3] Motorola sysV68 user's guide.

[4] Motorola sysV68 user's reference manual.

[5] Motorola sysV68 programmer's guide.

[6] Motorola sysV68 programmer's reference manual.

[6] Motorola Mvme143 MPU VMEmodule User's manual.

[7] Motorola Mvme143bug debugging Package User's Manual.

[8] Sun Microsystem Programmer guide,Writing device driver.

[9] Motorola pSOS+ RTEID-Compiliant Real-Time Kernel User's Manual.