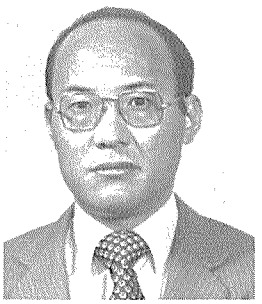
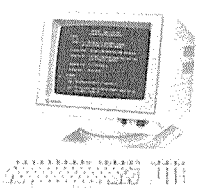


吳 吉 祿
韓國電子通信研究所
컴퓨터연구부장/工博

행정전산망 주전산기 운영체제



1. 개 요

목표 시스템은 다중처리기로 구성될 예정이므로 목표 시스템용 운영체제는 이러한 구조를 지원하기 위하여 기본적으로 다중처리기 운영체제를 개발하여야 한다. 다중처리기 운영체제는 UNIX System V Release 3.1과 완전 호환성을 유지하고 여러 처리기들이 공통의 기억장치와 주변장치들을 공유하도록 Tightly Coupled 시스템 방식을 채택하고 있다.

이러한 다중처리기 운영체제에 영문과 마찬가지로 한글을 시스템 수준에서나 또는 명령어, 사용자 프로그램에서 처리할 수 있도록 한글처리 기능을 추가할 예정이다. 이를 위하여 한글/영문으로 구성된 데이타들을 커널 수준에서 처리하고 입출력시키는 기능을 구현하여야 하며 명령어나 사용자 프로그램에서 한글 데이타를 처리할 수 있도록 인터페이스를 구현하여야 한다.

또한 근거리 네트 워크가 조성된 환경에서 목표 시스템들간의 자원을 편리한 방법으로 공유할 수 있도록 한글 처리 기능을 포함한 다중처리기 운영체제에 분산 운영체제 기능이 추가되어야 한다. 이를 구현하기 위해서 유닉스 커널 수준에서 기능들을 확장하여야 하며 그러한 기능의 사용을 위한 사용자 명령어 및 라이브러리들을 개발할 예정이다.

2. 다중처리기 운영체제

다중처리기 구조는 기억장치와 주변장치들을 공유하는 두 개 이상의 처리기들로 구성되어 있으며, 프로세스들이 여러 개의 처리기들에서 동시에 수행될 수 있으므로 시스템 성능을 보다 향상시킬 수 있다. 각각의 처리기는 독립적으로 수행하지만 모든 처리기들이 똑같은 커널을 수행한다.

다중처리기 운영체제에서는 두 개 이상의 프로세스가 서로다른 처리기들에서 커널 모드로 동시에 수행한다면

단일처리기 운영체제에서의 보호수단이 그대로 동작하더라도 Kernel Data Structure들이 파괴될 수 있다. 그러한 데이타 파괴를 방지하기 위하여 Locking Primitive들을 이용하여 커널 코드의 임계영역들에 대한 억세스를 순서화하여, 많아야 한 개의 처리기가 한번에 하나의 임계영역에 있는 코드들을 수행할 수 있게 커널을 여러 개의 임계영역들로 쪼개야 한다. 커널은 임계영역들로 쪼개기 위하여 Semaphore들을 사용하지만 Semaphore를 Implement하는 방법과 임계영역을 정의할 곳을 찾는 것이 매우 중요하다.

단일처리기 운영체제에서는 여러가지의 알고리즘들이 Sleep-Lock을 사용하여 임계영역을 순차적으로 억세스하도록 하지만 다중처리기 운영체제에서는 동작하지 않는다. 따라서 다중처리기 운영체제에서는 Locking Primitive가 Atomic이어야 한다. Lock의 Status를 Test하고 Lock을 Set하는 Action들이 한 프로세스만이 한번에 그 Lock을 조작할 수 있도록 Single, Indivisible Operation으로 처리해야 하므로 MC68000 Family의 Test-and-Set Instruction(TAS)을 사용한다.

Semaphore가 Dastructure를 Lock하려고 사용된다면, 프로세스는 Semaphore가 Lock되어 있을 경우 Sleep하고 커널이 다른 프로세스로 Context를 Switch하여 유용한 작업을 하도록 커널의 Semaphore Operation(P and V)을 더 복잡한 집합체로 구성해야 한다. Semaphore에 대한 억세스를 제어하기 위한 Lock, Field, Semaphore의 Value 그리고 그 Semaphore에서 Sleep하고 있는 프로세스들의 Queue로 구성되는 Structure가 되게 Semaphore를 정의한다. P and V Function은 단일처리기 운영체제의 Sleep and Wakeup Function과 유사하지만 Sleep and Wakeup에서 사용되는 Address가 Convenient Number인 반면 다중처리기 운영체제의 Semaphore는 Data Structure이다. 또한 Semaphore의 초기치가 0이면 Semaphore에서 P Operation을 하고 있을 때 프로세스는 항상 Sleep하므로 Operation은 단일처리기 운영체제의 Sleep Function으로 대체될 수 있으나, V Operation은 단일처리기 운영체제의 Wakeup Function이 그 Event에서 Sleep하고 있는 모든 프로세스들을 Wakeup하는 반면에 한 프로세스만 Wakeup한다.

여러 개의 Semaphore들을 Locking하고 있을 때 Locking 순서는 Deadlock을 방지하도록 균일 해야 한다. Deadlock은 Deadlock Detection Algotithm을 Implement함으로써 방지될 수 있으나 커널 코드가 복잡해지고, 프로세스가 동시에 여러 개의 Semaphore를 Lock해야 하는 경우가 커널에 그리 많지 않기때문에 Dealock이 발생하기 전에 Deadlock Condition을 피하도록 커널 알고리즘을 Implement 하는 것이 더 쉽다.

Interrupt Handler는 동시에 자원들을 사용하는 것으로부터 프로세스들을 보호하기 위하여 Semaphore를 Lock해야 하지만 Sleep으로 가지 않도록 Operation 대신 Spin Lock을 수행하여야 한다. 또한 Deadlock을 방지하기 위하여 커널은 Spin Lock을 수행하는 Interrupt를 막아야 한다.

다중처리기 운영체제에서 Wait 알고리즘은 Parent가 Zombie Child를 놓치지 않도록 해야한다. 따라서 각 Process Table Entry에 0으로 초기화된 Semaphore (zombie Semaphore)를 설정하여, Child가 Exit하지 않는 한 Wait에 있는 프로세스가 여기에서 Sleep하게 한다. 프로세스가 Exit할 때 Parent Semaphore에서 V Operation을 하여 Wait에서 Sleep하고 있는 Parent를 깨워주게 한다.

Device Driver Code에 Semaphore를 두지 않기 위하여 Driver Entry Point들에 P and V Operation 을 한다. Driver에 대한 모든 Entry Point들에 같은 Semaphore를 사용하며, 각각의 Driver에 다른 Semaphore를 사용함으로써 많아야 한 개의 프로세스가 한번에 그 Driver에 있는 임계영역의 코드를 수행할 수 있게 한다.

다중처리기 운영체제에서는 그 처리기에서 최근에 수행된 프로세스의 Context에서 커널이 Idle할 수 없다. 따라서 처리기당 하나씩의 Dummy Process를 만들어, 처리기가 할 일이 없을 때 커널은 Dummy Process 로 Context Switch를 하고 그 처리기의 Dummy Process의 Context에서 Idle하게 한다. Dummy Process는 커널 스택만으로 구성하며, Schedule되지 않게 함으로써 한 개의 처리기만이 자신의 Dummy Process에서 Idle할 수 있으므로 처리기들이 서로 다른 처리기에 영향을 주지 않는다.

3. 한글 UNIX 개발

가. 개 요

표준 UNIX는 7비트계의 ASCII를 내부 코드로 개발되어 영문을 사용하고 있는 반면에 한글은 컴퓨터에 한글을 표현하는 방법에 있어 1바이트 코드 또는 주로 8비트계 2바이트 코드를 사용하므로 한글을 위한 별도의 패키지가 없으면 표준 UNIX 시스템에서는 한글 자료를 인식하지 못하며 또한 처리하지도 못한다.

한글 UNIX는 영문과 마찬가지로 한글을 시스템에서나 또는 명령어, 사용자 프로그램에서 처리할 수 있게 하는 것인데 개발 목표는 다음과 같다.

1) 한글과 영문으로 구성된 Data를 UNIX 시스템 Kernel 수준에서 입출력하거나 처리하는 기능의 구현.
2) 명령어들이 한글을 처리할 수 있도록 하는 인터페이스의 구현 및 명령어들에 한글 처리 기능 구현
3) 사용자의 C 프로그램에 한글을 입출력하거나 처리할 수 있게 하는 인터페이스의 구현

한글 UNIX의 개발 방향은 각종 명령어들이 한글 처리를 위한 별도의 수정없이 영문과 같이 한글을 처리할 수 있도록 하는 한글 처리 환경의 구현을 지향한다. 이것은 본 한글 UNIX의 큰 장점이 될 것이며 기존의 다른 한글 UNIX들에 공통으로 존재하는 바 응용 프로그램들에게 한글 처리 환경을 제대로 제공하지 못한다는 치명적인 약점을 극복할 것이다. 또한, SVID(system Vinterface definition)에 맞게 개발하여 UNIX System V의 차후 Version과 호환성을 유지하도록 개발한다.

나. 추진 연구 내용

1) 개 요

본 행정 전산망 주전산기용 한글 UNIX는 UNIX System V R3.1상에서 개발할 것이며 각국어 지원을 위한 국제 기능들(international capabilities) 을 이용한다. 또한, 주 전산기용 한글 UNIX에 쓰일 기본 한글 내부 코드는 KS C 5601-1987의 2 바이트 완성형이다.

2) 개발 항목

가) 한글 처리 기반 Layer

(1) 한글 처리 화일 서브 시스템

한글 처리 화일 서브 시스템에서는 화일이나 디렉토리들의 이름을 한글로 쓸 수 있게 한다.

(2) 한글 처리 TTY 서브 시스템

한글 처리 TTY 서브 시스템에서는 시스템 커널 수준에서 영문과 마찬가지로 한글을 입력하고 처리하며 그 결과를 출력할 수 있게 한다. 또한 단수 바이트형 한글 터미날이나 복수 바이트형 한글 터미날 등 여러가지의 한글 터미날을 시스템에 붙여 사용할 수 있게 한다.

나) 한글 인터페이스 Layer

한글 인터페이스 Layer는 명령어들이 한글을 처리할 수 있도록 하는 인터페이스를 구현하는데 복수 바이트 한글 처리를 위한 기능들을 제공한다. 영문이 단수 바이트인데 반해 한글은 복수 바이트이므로 한글과 영문으로 된 자료를 처리하는데 발생하는 제반 문제들, 예를 들면 패턴 매칭을 위한 정규 표현(regular expression)의 처리, Right Margin의 처리 등을 해결한다.

한글 인터페이스 Layer는 UNIX System V Release 3.1에 부터 제공되는 국제적 기능들(international capabilities)을 이용한다. 따라서, 국제 기능에서 제공하지 않는 기능들은 이 Layer에 구현할 예정이며 이것이 개발 방향에 위배될 때는 다음에 나오는 한글 유틸리티 Layer에서 구현할 방침이다.

다) 한글 유틸리티 Layer

한글 유틸리티 Layer는 사용자의 C 프로그램에서 복수 바이트 한글을 입출력 또는 처리할 수 있게 하며 Shell을 비롯하여 편집기들 그리고 각종 명령어들에 이르기까지 한글을 처리할 수 있게 한다. 여기서 Shell은 Bourne Shell을 의미하며 명령어는 표준 UNIX Syetem V에서 제공하는 것을 대상으로 한다.

4. 분산 UNIX 시스템

분산 UNIX 시스템을 구성하면서 고려하여야 할 사항은 우선 UNIX와의 호환성을 유지하며 사용자에게 사용상의 불편함을 주지 않기 위한 투명도(transparency)를 제공하고 사용상의 효율이 좋아야 한다는 것이다. 여기서는 분산 UNIX 시스템의 개발을 분산 화일 시스템의 개발과 분산 처리 기능의 개발로 나누어 설명한다.

가. 분산 화일 시스템

분산 화일 시스템은 UNIX 운영 체제를 기본으로 구현하여 사용자에게 자원의 공유를 위한 환경을 제공하기 위하여 AT & T의 UNIX System V Rel. 3.1 에서 제공하고 있는 RFS(Remote File Sharing)를 기본으로

구성하고 RFS에서 제공하고 있지 못하는 기능들을 추가로 구현한다. 기본적으로 분산 화일 시스템은 원격 마운트 기법을 통하여 사용자에게 사용상의 투명함을 제공하고, 분산된 시스템 내에서 Buffer Consistency를 유지하며, 화일과 레코드 수준에서의 잠금(locking) 기능과 안전성을 제공한다. 이 밖에 Remote Caching 기능, 시스템 관리 화일에 관한 중복 허용, 확장된 UNIX IPC의 기능을 제공하고 원격 프로시듀어 호출(remote procedure call)의 기능을 구현하여 여러가지 서비스를 담당할 수 있도록 한다.

구현되는 분산 화일 시스템은 UNIX의 Semantic을 그대로 유지하여 호환성을 가지도록 하고 분산 자원들에 대한 Name 서비스를 행하여 분산되어 있는 자원들을 Name 서버를 통하여 찾을 수 있다. 한편 사용자들의 데이타 뿐만 아니라 원격 디바이스들도 로컬 UNIX 시스템내에 존재하는 디바이스들과 동일한 방법으로 사용할 수 있게 하고 원거리 신호 처리 기능을 구현하며 외부 데이타 표현(eXternal Data Representation:XDR) 기능을 통한 이 기종 시스템 간의 자원 공유 기능을 제공할 수 있어야 한다.

또한 사용자들에게 자원의 사용에 대한 신뢰성을 제공하기 위하여 데이타의 Consistency 유지가 필요하며 분산 자원들에 대한 보호가 요구된다. 또한 네트워크 등의 고장이 있을 때 이를 복구할 수 있는 기능이 있어야 한다. 그리고 원격 화일에 대한 버퍼 저장 (client caching) 기능을 제공하여 원격 자원 공유시 네트워크 메시지의 전송 수를 줄임으로써 네트워크의 오버헤드를 줄이고 시스템 성능의 향상을 기한다. 또한 시스템 관리자가 분산 UNIX 시스템을 효율적으로 관리하기 위한 기능들을 제공한다. 분산 시스템의 관리는 로컬 시스템 관리, 영역 관리 및 시스템 복구(recovery)와 시스템 효율을 결정하는 변수 조정 작업 등으로 나눌 수 있다.

분산 시스템에서는 사용자에게 높은 유용도를 제공하기 위하여 화일의 중복 저장을 구현하고 있지만 그 구현 정도가 상당히 어렵고 복잡하다. 몇몇 시스템에서는 화일 시스템의 중복도 고려하고 있지만 그 실현 정도는 매우 낮은 편이다. 일반적으로 시스템 관리를 위한 데이타 정도만을 중복 저장하고 사용자의 개인 데이타는 필요에 의하여 중복 저장하고 있다. 우리의 연구 범위에서도 분산 시스템의 구성 및 유지 관리를 위한 시스템 관련 정보만을 중복 저장한다.

나. 분산 처리 기능

분산 처리 기능의 개발을 위하여는 효율이 좋은 프로세스간 통신 방식이 분산되어 있는 환경에서도 로컬 시스템 내에서와 같이 동작될 수 있도록 네트워크를 통하여 확장되어야 한다. 우리의 환경에서는 Pipe, Message 등의 UNIX 프로세스간 통신 방식을 확장하여 사용하고 원격 프로시듀어 호출을 구현하여 사용한다. 원격 프로시듀어 호출 방식은 동기화 등을 쉽게할 수 있고 Stub Routine을 통하여 여러가지 다양한 서비스를 구현할 수 있다. 한편, UNIX System V Rel.3 이상에서는 입출력을 위한 디바이스 취급에 Stream 방식을 도입하여 사용하고 있으므로 기본적으로 Stream 방식을 사용하고 원격 프로시듀어 호출 방식 및 통신 프로토콜들을 Stream을 이용하여 구현한다.

프로세스의 원격 처리는 전체 시스템의 자원을 충분히 활용하기 위하여 필요하며 분산 시스템 내의 각 시스템 상태를 파악할 수 있어야 한다. 분산 시스템은 네트워크로 연결되어 동작하므로 원격 시스템의 전체적인 상태를 항상 간직하여 자원의 유용도를 높일 수 있도록 하는 것이 중요하다. 이를 위하여 네트워크의 상태나 원격 시스템의 사용 상태를 조사하는 기능을 수행할 수 있는 도구가 필요하다. 또한 분산 프로세스의 Debugging도 필요하다. 이것은 분산되어 동작하는 프로세스들의 상태를 추적함으로써 고장을 발견하는 데 사용된다. 다음으로는 분산 UNIX 시스템에서 필수적인 네트워크 서비스를 효율적이고 편리하게 수행토록 한다. 네트워크 서비스 기능은 사용자 입장에서 사용하기 편리한 많은 서비스를 제공하면서 이들 서비스가 통신 프로토콜 및 사용 매체에 무관하도록 한다. 이를 위해 Stream 기능과 UNIX에서 제공되는 OSI 통신 프로토콜 모델의 트랜스포트층 사용을 위한 인터페이스 라이브러리를 사용한다. 그리고 분산 환경에서의 효율을 측정할 수 있는 효율 측정 도구가 구현되어야 한다.

이러한 분산 UNIX 시스템의 기능들은 다중 운영 체제 및 한글 처리 기능과 결합하여 운영 체제 내에서 동작된다. 이러한 기능들은 구현되어 있는 것도 있고 새로 구현하여야 하는 것도 있다. 우리는 항상 발전해가는 연구의 동향을 추적하여 기능을 확장하는 데 반영할 것이다.