CSMS의 UNIX Porting에 대한 Feasibility 研究

李美暎 · 李錫喆 / 網運営研究室

〈Abstract〉

The operations support systems such as CSMS, TLMOS, and SLMOS may be related each other, and these systems had better cooperate each other. At the present time, UNIX system is most popular operating system. TLMOS and SLMOS have been developed under the UNIX environment and using C language. On the other hand CSMS was built on HP RTE O.S. using Pascal.

When we constuct the operation and maintenance organization by networking several operations support system, the interface of intersystem must be simple and the integrity of whole system must be kept. It is somewhat valuable to investigate transfering CSMS from RTE O.S. to UNIX in order to integrate operations and maintenance systems.

Here, we give a basic material about porting CSMS in UNIX.

Ⅰ. 개 요

통신망 운용보전은 사용자 서어비스에 직결되어 있기 때문에 100%직영으로 수행되고 있는 반면 통신망 규모가 확대됨에 따라 점차 기업이윤에 미치는 비중이 증대되어 효율화가 요구되고 있다. 즉, 효율적인 집중운용보전 체제의 정립이 필요하다.

현재, 통신시설은 ESS교환시설, 가입자선로, 국간선로, 장거리회선 등 중요시설이 가입단위 구역으로 볼 수 있는 telephone area를 중심 구성으로 하여 설치, 운용되고 있다.[1]·

CSMS는 ESS교환시설의 보전업무를 효율적으로 직접 지원하는 ESS집중보전 시스팀이고, SLMOS는 가입자선로 보전센터의 구성 및 운용을 활성화하는 시스팀이며 TLMOS는 telephone area에서 착발신되는 장거리통신의 원활한 소통을 위하여 장거리회선을 감시제어 및 운용, 관리하는 시스팀이다.[2, 3, 4] 통신망 운용보전의 효율화를 위해서는 이와 같은 각각의 운용보전시스팀이 서로 연관, 상호보완 작용을 하

여야 한다.

CSMS는 HP1000 Minicomputer (O.S.는 RTE-IVB)를 이용한 시스템이고 SLMOS와 TLMOS는 VAX-11/750 (O.S.는 UNIX 4.1BSD)를 사용하여 개발된 시스템으로서 이 시스템들이 공동으로 통신망 운용보전 체제를 구성하기 위해서는 각 시스템의 portability가 높아야 하며 보전시스템간의 네트워크 형성시 인터페이스가 간단하여야 한다. 따라서 UNIX를 이용한 CSMS를 개발함으로써 전 보전시스템의 integrity를 높힐 수 있으며, 같은 기종의 컴퓨터를 사용함으로써 resource 사용도를 효과적으로 이룩할 수 있다.

UNIX는 현재 가장 널리 사용되는 오퍼레이팅 시스템으로서 여러가지 장점을 지니고 있다. UNIX의 개발 동기가 소프트웨어 시스템의 연구 개발을 좀 더 편리하고 효과적으로 할 수 있는 환경조성인 것처럼 UNIX는 간단하며 특정 응용부문에 치우침이 없이 일반적이다.

이와 같은 배경하에 CSMS를 UNIX에 porting하는 문제에 대한 타당성을 조사하였다. VAX-11/750이 CSMS에 적당한 호스트 컴퓨터인지, UNIX가 CSMS가 필요로 하는 오퍼레이팅 시스템으로서의 요건을 갖추고 있는지, 그리고 CSMS를 UNIX에 porting할때 사용할 언어에 대하여 조사하였다.

Ⅱ. Porting에 대한 Feasibility조사

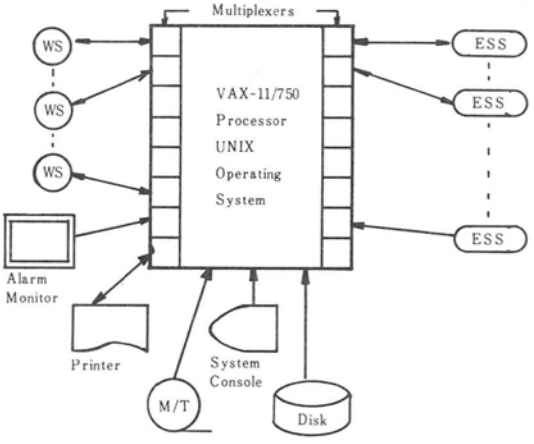
1. Hardware (VAX-11/750) [5]

VAX-11/750은 미니 컴퓨터로서 주기억 용량은 256K byte부터 최대 2M byte까지 가능하다. 메모리는 16K bit MOS RAM 칩으로 구성된 256K byte 어레이 카드를 실장함으로써 용량을 증설한다. (최대 8개 내장 가능)

I/O 시스템은 UNIBUS 서브 시스템과 MASSBUS 서브 시스템으로 구성되어 있다. UNIBUS 서브시스템은 asynchronous, bidirectional bus를 통해 터미널이나 멀티플렉서 같은 매개체 혹은 저속의 주변 장치를 연결한다. Device 연결은 n개까지 가능한데 일반적으로 시스템의 용도와 효율을 고려하여 적절하게 연결한다. MASSBUS 서브시스템은 디스크, 자기 테이프 등 고속기억 장치를 시스템에 연결하며, MASSBUS 어댑터 3개까지 연결 가능하다. MASSBUS 어댑터 하나에 device controller 8개까지 수용 가능하므로 시스템에 총 24개의 device controller를 연결할 수 있다.

CSMS에서 필요한 I/O port는 workstation용 7개, ESS용 16개와 그밖에 프린터, 경보감시기를 연결해야 하는데 이것은 UNIBUS 에 멀티 플렉서를 연결하여 사용하면 충분하다. 시스템 콘솔은 프로세서에 직접 연결한다. 메시지와 프로그램 저장용으로 사용하기 위해 용량 50 M byte의 디스크 드라이버가 필요하고 시스템 back-up과 데이터의 장기 보관을 위하여 자기 테이프 드라이버가 필요한데 이는 MASSBUS에 의해 연결되며, MASSBUS의 전송 속도는 2 Mb/s이므로 디스크 액세스 시간 제약은 없다.

시스템의 H/W 구성도는 다음 〈그림 1〉과 같다.



〈그림 1〉 Hardware 구성도

2. Operating System

UXIX시스템은 programing research를 좀 더 편리하고 효과적으로 수행할 수 있는 환경조성을 위해 간단하고 일반적이며 고수준의 오퍼레이팅 시스템으로 개발되었으며, micro version, time sharing version, real time version으로 구분할 수 있다.

Micro version은 마이크로 컴퓨터에 implement한 것으로 LSI-version이 있다. Time sharing

version은 multi-user를 위한 시스팀으로 Berkeley version(4.2 BSD)과 AT & T version (UNIX System V)으로 나뉘어 발전해 왔고 두 시스팀은 많은 면에서 비슷하지만 compatible하지는 않다. Real time version은 UNIX 시스팀의 실시간 처리 기능이 미비함을 보완한 것으로 MERT, DMERT가 있다. 일반적으로 UNIX라 하면 time sharing version을 말하며 시스팀 구성은 〈그림2〉와 같이 user, system, service, machine interface function으로 구분할 수 있다.[6]

이 절에서는 CSMS를 실현하기 위하여 오퍼레이팅 시스팀이 갖추어야 할 요건을 memory management, file system management, process management, interprocess communication, 그리고 기타 그밖의 문제에 대해 살펴보고 UNIX시분할 시스팀이 이러한 조건에 적합한지 여부를 논의한다.

가. Memory Management

Memory Management 문제에 있어 특별한 제약은 없으며 효과적으로 resource를 이용할 수 있으면 된다.

UNIX에서는 virtual memory 개념을 사용하여 프로세스가 점유하는 virtual address space에서 현재 수행해야 할 부분을 physical address space의 사용 가능한 부분으로 mapping 시켜 준다.

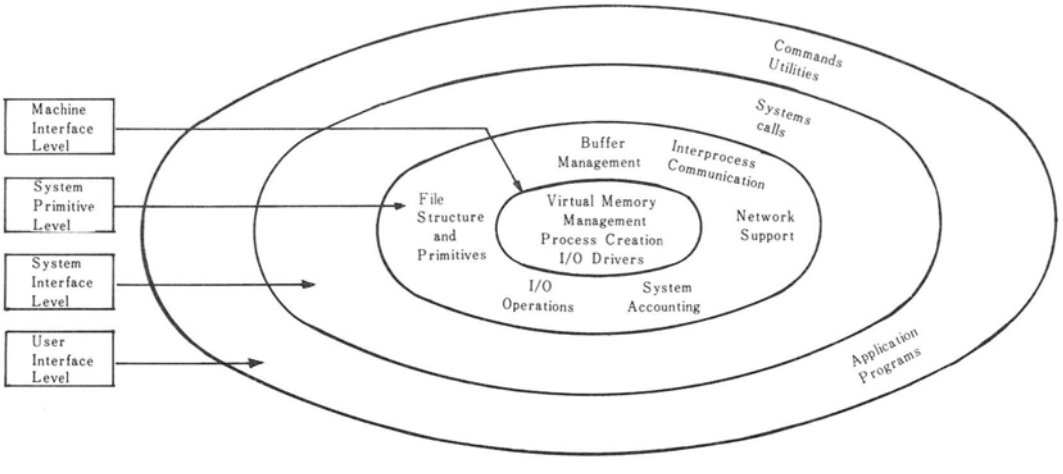
이때 main memory와 secondary memory의 할당은 first-fit 알고리즘을 이용한다.

나. File System Management

CSMS는 ESS에서 나오는 메시지를 실시간으로 수집하여 ESS국마다 별도의 파일로 만들어 디스크에 저장하며 일반 기능과 교환기에 따른 특수 기능은 이 파일에 들어 있는 데이터를 이용하여 교환기의 상태를 검색, 혹은 고장을 분석하고 통계 처리를 한다. 그러므로 디스크 액세스에 따른 overhead가 적도록 system buffering, physical disk address calculation, disk head position이 효율적으로 이루어 져야 한다. 그리고 CSMS에서는 대부분의 일반 기능이나 특수 기능이 디스크에 저장된 ESS 메시지 파일을 이용하므로 여러 workstation에서 동시에 보전 작업이 이루어지기 위해서는 여러 프로세서가 동시에 파일을 액세스하는 것이 가능해야 한다.

UNIX의 디스크 파일 시스팀은 512-byte 블럭의 단위로 이루어진 randomly addressable array로 간주된다.[8] 그러므로 파일은 디스크의 free storage pool로부터 할당된 여러 블럭으로 이루워져 실제 physical 상으로는 uncontinuous하게 storage가 할당된다. Logical file의 physical address mapping은 i-node를 통하여 이루워지며 direct address mapping은 10 블럭(5,120bytes)까지이고 그 다음은 indirect 하게 이루워져 address mapping에 따른 overhead가 발생한다.

그러므로 logical file의 physical address mapping이 간단하고 디스크 헤드의 이동을 최



〈그림2〉 UNIX Operating System 구성

소로 하기위해 자주 사용하는 파일은 continuous 하게 할당하는 것이 바람직하다.[7] 이와같은 일을 할 수 있는 logical file system을 구성하여 사용자 UNIX파일 시스팀과의 인터페이스 역할을 해 주는 것이 좋다.

다. Process Management

CSMS의 process scheduling 방법은 3가지로 나눌 수 있다. 첫째, 교환기로부터 출력되는 메시지를 수집, 저장하는 프로세스들(GTMSG, COLL)은 실시간으로 동작하여야 데이터의 손실을 방지할 수 있다.

둘째, 여러 workstation에서 동시에 보전 작업을 실시하고 있는 보전요원들을 위해서는 시분할 방식을 택하여 여러 사용자에게 균등하게 서어비스를 제공할 수 있어야 한다. (다양한 일반 기능과 특수 기능 사용)

셋째, 일일 단위, 주 단위 등 주기적으로 수행되는 전자 교환기의 보전 상태를 표시하는 메니저 리포트, 통화량 집계 리포트 등의 통계자료를 출력해주는 프로세스(특수기능)들은 background로 수행함으로써 CPU를 효율적으로 사용할 수 있다.

UNIX process scheduling 방식은 이와 같은 요구를 만족시킨다. UNIX시스팀은 우선도에 따라 프로세스를 체계화시키므로 실시간 프로세스에는 다른 시분할 프로세스를 보다 높은 base priority를 할당함으로써 먼저 수행 가능하다.[8] 또한 같은 priority process간에는 round robin 방식을 사용하여 정해진 시간 단위만큼 교대로 CPU를 점유하므로 시분할 프로세스간에 균등하게 CPU를 할당하여 multi-user를 조정한다. Background process에는 시분할 프로세스 보다 낮은 우선도를 할당함으로써 실시간 프로세스나 시분할 프로세스가 CPU를 사용하지 않는 유지시간에 수행하여 CPU 사용의 효율을 기한다.

라. Interprocess Communication

CSMS에서는 크게 두 종류의 프로세스 간 communication이 사용된다. (〈그림 3〉참조)[9] 첫째, ESS로부터 출력되는 메시지를 수집하는 프로세스(GTMSG)와 디스크 파일에 이를 저장하는 프로세스(COLL)사이에 메시지 전달을 위한 통신이 필요하고, 메시지 저장 프로세스에서는 실시간으로 메시지를 분석하는 메시지 다중발생 감시 프로세스(MMT)에 즉시 이 메시지를 전달한다.

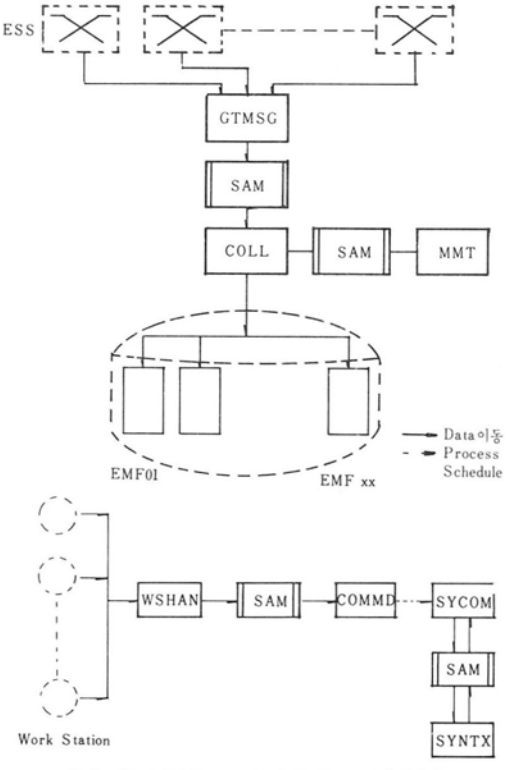
둘째, workstation에서 입력되는 명령을 처리하는 일련의 프로세스(WSHAN, COMMD, SYCOM, SYNTX)간의 통신이 필요하다. 즉, interprocess communication기법은 프로세스간 메시지 전달이 가능하여야 하고, 메시지의 도착을 감지하는 event-driven interrupt mechanism이 필요하다.

UNIX의 interprocess message facility와 signal facility는 이와 같은 interprocess communication을 가능하게 한다.

마. 기 타

ESS로부터 메시지를 수집하기 위해서는 driver routine에 약간의 수정이 필요한데 UNIX는 전체가 하나의 모듈로 구성되어 있어 수정할 때마다 새로 전체를 실장해야하는 단점이 있다.

UNIX에 shell이라는 command interpreter가 있으므로 CSMS command의 실장은 shell을 확장시켜 사용할 수 있다.



※ 현 시스팀은 SAM(System Available Memory)을 통하여 interprocess communication을 한다.

〈그림 3〉 Interprocess Communication

3. Language

현 CSMS는 HP 1000에서 사용 가능한 언어를 이용하여 개발되었다. 주 사용 언어는 Pascal/1000이고, 일부는 메모리 용량과 수행 시간의 제약 때문에 Fortran과 Assembley Language를 이용하였다.

UNIX시스템에서의 주요 언어는 C-Language이고 Pascal, Fortran 등도 사용 가능하다. 그러므로, CSMS를 UNIX에 porting하기 위해서는 주 언어로 Pascal를 이용하는 방법과 C를 이용하는 방법, 그리고 두 언어를 혼합해서 사용하는 방법이 있다.

가. C-Language 사용

C-language를 이용한다면 시스템의 모든 부분을 재설계하여 새로 개발하거나, 현재 있는 시스템의 Pascal statement 하나 하나를 C로 1 : 1 맵핑시켜야 한다.[10]

Pascal을 C로 맵핑시키는 것은 Pascal에 있는 기능을 C로 바로 표현 가능한가에 대한 문제와 오퍼레이팅 시스템(RTE-IVB)의 기능을 직접 사용하는 system call부분이 바로 convert 가능한지에 대하여 고려하여야 한다. 이런 문제를 고려할 때 단순한 convert 작업에도 많은 man-month가 소요되며, 시스템의 효율성을 고려할 경우에도 바람직하지 않으리라 생각한다. 따라서 C-Language를 이용할 시는 UNIX시스템의 효능을 기대할 수 있는데 이 때에는 상당히 많은 man-month가 소요되리라 생각한다.

나. Pascal 사용

Pascal language를 사용한다면 현 시스템의 많은 부분이 그대로 이용가능 할 것이고 hardware dependent한 부분과 오퍼레이팅 시스템의 기능을 직접 사용한 부분은 재설계 하여야 하며, 나머지는 Pascal/1000 을 Berkeley Pascal 로 컨버트하여야 한다.

Pascal/1000은 standard Pascal의 superset으로 더 많은 기능을 제공하며 호스트 컴퓨터의 기종에 따라 Pascal implementation에 사소한 차이가 있으므로 convert 작업시에는 다음과 같은 문제점이 발생한다.[11]

—RTE-IVB에서는 사용자가 사용 가능한 logical memory space가 32K 이므로 프로그램의 크기가 클 경우 Pascal/1000에서 제공하는 segmentation기법을 이용하여 처리하였다. UNIX에서는 virtual memory 개념을 사용하므로 프로그램의 크기에 대한 제약이 없고, 또한 Berkeley Pascal에서는 segmentation 이 허용되지 않으므로 convert시 프로그램을 종합 혹은 재구성하여야 한다.

—Pascal 1000에서는 external routine call 이 가능하므로 프로그램의 많은 부분에서 RTE-IVB를 직접 이용하는 system call 'EXEC' 과 file management system call을 이용하였다.

그러므로 이에 해당하는 부분을 UNIX systen call로 대치되어야 한다.

—Pascal/1000에서는 upper case와 lower case 문자에 대한 구별을 두지 않으나 Berkeley Pascal에서는 reserved word는 모두 lower case 문자이어야 한다.

—그밖에 Pascal/1000 extension을 이용한 부분에 대해 약간의 convert 작업이 이루워져야 한다.

이와 같이 현 시스템을 UNIX에 porting하기 위해 Berkeley Pascal로 convert하는 것이 소요되는 man-month를 볼 때는 가장 경제적이지만 시스템의 효능면에서 볼때는 상당히 떨어지리라 생각된다.

다. C와 Pascal 혼용

C Language는 매우 고수준의 언어는 아니지만 범용 프로그래밍 언어이므로 여러 부문에 편리하고 효과적인 언어이다.

CSMS는 크게 기본기능, 일반기능, 특수기능으로 구분되며, 이 중 기본기능은 CSMS 의 바탕이 되는 것으로 많은 면에서 H/W와 RTE-IVE에 의존하고 있으며, 일반기능과 특수기능은 이 기본기능을 바탕으로 하여 그 위에서 동작하게 된다.

그러므로 기본 기능에 대해서는 새롭게 UNIX와 C-Language을 이용하여 재설계하는 것이 타당하고 일반기능과 특수기능은 현 시스템에서 사용한 Pascal/1000을 Berkeley Pascal 로 convert하여 사용하는 것이 시스템의 효능과 소요되는 man-month를 고려할 때 가장 적절하다고 생각한다.

Ⅲ. 결 론

CSMS는 실시간 멀티프로그래밍 오퍼레이팅 시스템을 필요로 하므로 UNIX시분할 시스템은 적합하지 않은 것처럼 보이나, UNIX 시스템은 일반적이며 융통성이 있는 시스템이므로 CSM-S를 UNIX에 porting하는 것은 가능하리라 생각된다. 그러나 기존 응용시스템을 새로운 오퍼레이팅 시스템하에 implement한다는 것은 상당히 많은 문제를 내포하고 있다.

Porting함으로써 얻을 수 있는 효과가 투입된 man-month와 비교할 때 그만한 가치가있는지 여부와 porting한 후 그 시스템이 정책상으로도 받아들여질 수 있는지 등 충분한 검토가 실시된 후 결정되어야 한다.

<參 考 文 獻>

1. 운용보전 시스템 개발사업, 1984
2. CSMS 사용자 지침서, 1983
3. TLMOS 사용자 지침서, 1985
4. SLMOS 사용자 지침서, 1983
5. VAX Hardware Handbook, 1980-1981
6. Bott, Ross A., "Dual Port Solves Compatibility Problem," Computer Design, pp. 205-214, Aug. 1984
7. Cohen, H., and J.C. Kaufeld, "UNIX Time-Sharing System : The Network Operations Center System," B.S.T.J., pp. 2289-2304, Vol. 57, No. 6, July-Aug, 1978
8. Thompson, K.,"UNIX Time-Sharing System : UNIX Implementation," B.S.T.J., pp. 1931-1946, Vol. 57, No. 6, July-August 1978
9. CSMS 시스템 설명서 1983
10. Ritchie, D.M, S.C. Johnson, M.E. Lesk, and B.W. Kernighan, "UNIX Time-Sharing System : The C Programming Language," B.S.T.J., pp. 1991-2019, Vol. 57, No. 6, July-August 1978
11. Joy, William, N. Susan, L. Graham, and Charles B. Haley, Berkely Pascal User's Manual Version 2.0, Oct. 1980