2014년 졸업 연구 결과 보고서

통합관리시스템

팀명 J.net (just.net)

지도교수 이 병 천 교수님

팀 장 정 봉 준
팀 원 황 택 주
안 빈

2014. 05
중부대학교 정보보호학과

목 차

1. 서론

1-1. 연구 배경 및 목적

인터넷의 보급이 널리 확산되면서 다양한 정보가 축척되어 지금과 같은 정보화 사회가 형성되었다. 컴퓨터가 발전 할수록 관련된 직종도 늘어났으며, 관심을 갖는 사람들도 증가했다. 초기의 컴퓨터는 거대한 고철과 같았고, 계산기 정도의 능력밖에 안됐다. 차츰 소형화되면서 기능 또한 전기적 신호에서 텍스트로 발전하게 된다. 이때부터 컴퓨터는 다양한 운영체제가 생기게 되는데, 오늘날 보통 가정집이나 주변 PC방에 설치되어있는 익숙한 윈도우나 MAC이 그래픽 환경을 강점화화여 사용자의 편의를 대폭 활용한 운영체제이며, 이와 다르게 사용자의 편의보다는 대형 컴퓨터나 특정 사용분야에 강점화 된 운영체제가 있다. 대표적으로 리눅스와 유닉스가 있다. 여기서 우리는 유닉스에 대해 접근하기로 했다. 앞서 관심이 많다는 것은 그것을 배우고 활용하려는 사람이 많다는 뜻인데, 이를 활성화하기 위해서는 좋은 실습환경이 뒷받침되어야 된다고 생각했다. 과연 '좋은 실습환경은 무엇일까?' 우리는 고심 끝에 실습을 하며 겪었던 고충을 통해 두 가지 해답을 찾아냈다. 첫 번째로는 관리자가 실습환경에 주기적인 유지와 보수다. 두 번째로는 관리자의 편의성이다. 이유로는 관리가 수월해야 실습도중에 나타나는 문제를 최소화 할 수 있기 때문이다. 이 두 가지를 기반으로 유닉스시스템의 자원을 활용하여 통합관리 시스템을 구축하려 한다.

1-2. 유닉스에 대해서

1)유닉스의 역사

유닉스는 다른 컴퓨터의 시스템 프로그램과는 달리 일반인에게 공개함으로써 미국의 각 대학과 여러 기업체로 널리 퍼지게 되었다. 이렇게 유닉스가 널리 퍼지게 되면서 유닉스에 많은 기능들이 추가되었다. 그리하여 1978년에 UNIX V6(Version 6)라는 이름으로 일반인에게 공개되면서 IBM 370 컴퓨터에 이식하여 시험하였고 다른 많은 회사에서도 자사의 중대형 컴퓨터에 유닉스를 이식하기 시작하였다. 유닉스는 크게 2개의 계보로 발전하였는데, AT&T의 상업용 유닉스로서 SYSTEMV 계열과 학문적 연구 목적인 버클리 대학의 BSD(Berkeley Software Distribution) 계열로 나뉘어 발전하였다. 요즘은 두 가지 계열의 장점들을 결합하여 통합된 형태로 각 회사의 유닉스 운영체제로 사용하고 있다.

① 1960년대와 1970년대

1960년대 MIT (Massachusetts Institute of Technology), AT&T Bell 연구소 및 General Electric은 GE-645 메인프레임 컴퓨터 상에서 돌리기 위한 Multics (Multiplexed Information and Computing Service)라고 불리는 실험적인 운영 체제를 함께 연구하였다. 그 목적은 향상된 보안 등 여러가지 혁신적인 기능들을 가지는 쌍방향 운영 체제를 만드는 것이었다. 프로젝트 결과 제품 릴리즈를 개발하였지만, 이 릴리즈는 성능이 형편없다는 것이 드러났다. AT&T Bell 연구소는 이 프로젝트에서 손을 떼고, 연구원들을 다른 업무에 배치했다. Bell 연구소 팀의 개발자 중 한사람인 Ken Thompson은 GE-645 메인프레임에 대한 개발을 계속했고, 이 컴퓨터용으로 Space Travel이라는 게임을 만들었다. 하지만 이 게임은 GE 컴퓨터에서 매우 느리고, 운영에 비용이 많이 들었다.

Thompson은 Dennis Ritchie의 도움으로 이 게임을 DEC PDP-7 어셈블리 언어로 다시 만

들었다. 이 경험과 이전 Multics 프로젝트에 대한 경험으로 인해 Thompson은 DEC PDP-7을 위한 새로운 운영 체제를 연구하게 되었다. Thompson과 Ritchie는 Bell 연구소에서 Rudd Canaday를 포함한 개발팀을 이끌고, 새로운 멀티 태스킹 운영 체제 및 파일 시스템을 개발하게 되었다. 그들은 이 운영 체계에 명령행 인터프리터와 몇 가지 작은 유틸리티 프로그램도 집어넣었다. 이 프로젝트는 Uniplexed Information and Computing System의 약자로 Unics라고 불렸고, 두명이 동시 사용자를 지원할 수 있었다. 유닉스라는 이름은 Brian Kernighan이 Multics에 대한 반발로 지었다. 그 후, 거세된 Multics라는 Unics (내시(eunuchs)의 동음이의어)의 안좋은 말장난 때문에, 이름은 Unix로 바뀌었다,
Computer Science Research Group이 PDP-7보다 큰 컴퓨터에서 유닉스를 사용하기를 원했을 때까지도 Bell 연구소로부터의 재정적인 지원은 없었다. Thompson과 Ritchie는 PDP-11/20 용 유닉스에 텍스트 처리 기능을 추가하겠다는 약속을 했고, 이에 따라 Bell로 부터 약간의 재정적인 지원을 받게 되었다. 1970년, 유닉스 운영 체제 (Unix Operating System)라는 이름을 공식적으로 가지고 PDP-11/20에서 운영되었다. 여기에는 roff라는 텍스트 포맷 처리 프로그램과 텍스트 에디터가 추가되었다. 이들은 모두 PDP-11/20의 어셈블리 언어로 작성되었다. 이렇게 유닉스와 roff 및 에디터로 만들어진 초기의 "텍스트 처리 시스템"은 Bell 연구소에서 Bell의 특허 신청을 위한 텍스트 처리를 위해 운영되었다. 유닉스 프로그래머를 위한 매뉴얼은 1971년 11월 3일에 출판되었다.
1973년에 유닉스를 C 프로그래밍 언어로 재작성하기로 결정하였다. 이 변화로 인해 유닉스는 나중에 다른 플랫폼에서 돌아가도록 쉽게 수정이 가능해졌으며, 다른 개발자들이 독자적인 변형을 만들 수 있게 되었다. AT&T는 라이센스를 받고 미 정부, 대학, 기업들이 유닉스를 사용할 수 있도록 했다. 이 라이센스에는 PDP-11 어셈블리 코드로 만들어진 커널을 제외한 모든 소스 코드도 포함되어 있었다. 한편, 해적판 유닉스 커널이 1970년대 후반 널리 퍼져서 유닉스를 교육용 운용 체제로 폭넓게 채택되는데 일조를 했다.
개발은 계속 확대되어 Versions 4, 5, 6이 1975년 발표되었다. 이 버전들은 파이프 기능을 추가하여, 더욱 모듈화된 코드 기반 개발을 가능하게 함으로써 개발 속도를 더욱 높일 수 있었다. V5, 특히 V6에 의해 Bell 연구소 안팎으로 더욱 다양한 종류의 유닉스 버전들이 쏟아져 나오게 되었다. 예를 들면, PWB/UNIX, IS/1 (최초의 상용 유닉스) 등이 나왔으며, Wollongong 대학은 유닉스를 Interdata 7/32 용으로 포팅했다 (최초의 비-PDP 유닉스).
1978년, VAX 용 UNIX/32V가 발표되었다. 이 당시에, 600대 이상의 컴퓨터에서 유닉스가 운영되고 있었다. 널리 사용된 Research Unix의 마지막 버전인 Version 7 Unix는 1979년 발표되었다. Versions 8, 9, 10은 1980년대 개발되었지만 몇 몇 대학들에서만 사용되었다.

② 1980년대
1982년, AT&T는 Version 7에 기반한 UNIX System III를 개발하여 상용 버전으로써 직접 판매하였다. 한편 AT&T의 자회사인 Western Electric는 UNIX System (Versions 1에서 7)에 기반한 예전 유닉스 버전들을 계속해서 판매하고 있었다. 서로 다른 버전들에 대한 혼란을 종식시키기 위해, AT&T는 여러 대학들과 기업들에서 개발된 여러 버전들을 결합하여 UNIX System V Release 1을 만들었다. 이 버전에는 캘리포니아 대학교 버클리 (UCB)에서 개발된 vi 에디터, Curse 등의 기능이 추가되어 있었다. 이는 또한 DEC VAX 컴퓨터에 대한 지원도 포함하고 있었다.
새로운 상업적인 유닉스 릴리즈는 더 이상 소스 코드를 제공해주지 않았기 때문에, UCB는

UNIX System III 및 V에 대한 대안으로 계속해서 BSD Unix를 개발하였다. BSD의 개발 과정에서 가장 중요한 측면이라면 유닉스 커널에 TCP/IP 네트워크 코드를 추가한 점일 것이다. BSD 개발 노력으로 인해 네트워크 코드를 포함한 몇 가지 중요한 릴리즈들이 나왔다: 4.1cBSD, 4.2BSD, 4.3BSD, 4.3BSD-Tahoe ("Tahoe"는 BSD 커널의 첫 비-DEC 포트인 CCI Power 6/32 아키텍처에 대한 닉네임이 되었다), Net/1, 4.3BSD-Reno, Net/2, 4.4BSD, 4.4BSD-lite. 이 릴리즈 안에 들어있는 네트워크 코드는 오늘날 사용되는 거의 모든 TCP/IP 네트워크 코드의 선조격이다. 여기 들어있는 Berkeley Sockets API는 네트워크 API를 위한 사실상의 표준으로, 많은 플랫폼에서 복사되어 사용되었다.
다른 회사들도 자신의 컴퓨터에 사용하기 위해 UNIX System의 상용 버전을 제공하기 시작했다. 대부분의 이런 새 유닉스들은 AT&T의 라이센스 하에서 System V 기반으로 개발되었다. 일부는 BSD를 선택하였다. BSD의 개발자 중 한명인 Bill Joy는 1982년 Sun Microsystems를 공동창업하여, 자신들의 컴퓨터에서 사용하기 위해 SunOS (현재는 Solaris)를 만들었다.
1984년, 서로 호환이 되는 공개 표준 시스템을 만들기 위해 (즉 유닉스 시스템 표준화를 위해) X/Open이라는 산업 그룹이 만들어졌다.
AT&T는 파일 잠그기, 시스템 관리, 잡 컨트롤, 스트림, 원격 파일 시스템, TLI 등의 다양한 기능들을 UNIX System V에 추가했다. 1987년에서 1989년 사이 X/Open과는 별도로, AT&T는 Sun Microsystems와 협력하여 Xenix, BSD, SunOS, and System V를 통합한 System V Release 4 (SVR4)를 내놓았다. 이 새 릴리즈는 이전의 모든 기능들을 하나의 패키지로 통합하였으며, 라이센스 비용 또한 높아졌다.

③1990년대
1990년, Open Software Foundation은 그들의 표준 유닉스로 Mach와 BSD에 기반한 OSF/1을 내놓았다. 이 재단은 SVR4를 내놓은 AT&T와 Sun의 연합에 대항하여 여러 유닉스 관련 기업들이 1988년에 만들어졌다. 이에 대해, AT&T와 몇몇 기업들은 또다시 연합하여 "UNIX International" 그룹을 만들어 대항하였다.
1991년, 일단의 BSD 개발자들 (Donn Seeley, Mike Karels, Bill Jolitz, Trent Hein)이 캘리포니아 대학을 떠나서 Berkeley Software Design, Inc (BSDI)를 설립하였다. BSDI는 저렴한 인텔 플랫폼을 위해 완전한 기능을 갖춘 BSD 유닉스의 상업용 버전을 만들었고, 이로 인해 기업 환경에서 저렴한 하드웨어를 사용하는 방안에 대한 관심이 증가하였다. 곧 Bill Jolitz는 BSDI를 떠나서, FreeBSD, OpenBSD, NetBSD 등 무료 소프트웨어의 원조격인 386BSD의 확산을 위해 노력하게 되었다.
1993년까지 대부분의 상업적인 유닉스 업체들은 자신들의 상업용 유닉스 버전을 SVR4에 기반하도록 수정하였고, 많은 BSD 기능들을 이 위에 덧붙였다. 1994년, OSF는 OSF/1의 개발을 중지하였다,
UNIX System V Release 4가 출시된 직후, AT&T는 유닉스에 대한 모든 권리를 Novell에 매각했다. Novell은 자신들의 Netware를 UNIX System V Release 4와 결합하여 UnixWare라는 독자적인 유닉스를 출시했다. Novell은 이를 사용하여 Windows NT에 대항하려 했으나, 많은 어려움을 겪었다.
1994년, Novell은 유닉스 관련 자산을 분리하여 매각하기로 결정하였다. 유닉스 상표와 인증 권한은 X/Open 컨소시움에 매각되었다. 1996년, X/Open은 OSF와 합병되어 Open Group이 되었다. Open Group은 그 후 유닉스 운영 시스템에 대한 다양한 표준들을 정의하였다.
1995년, 기존 유닉스 라이센스의 관리와 지원 사업 + System V 코드 기반에 대한 추가적인 개발 권리가 Santa Cruz Operation으로 이관되었다.

④ 2000년대

2000년, Santa Cruz Operation은 유닉스 사업과 자산을 Caldera Systems에 매각하였다. Caldera Systems의 이름은 이후 SCO Group으로 바뀐다. SCO Group은 리눅스 사용자와 업체들에 대해서 대규모의 법적 행동에 들어갔다. SCO Group은 리눅스에 자사가 소유권을 가지고 있는 유닉스 코드가 들어있다는 주장 등, 여러 소송에서 다양한 법적 이론들을 내놓았다. SCO Group은 SCOsource라는 프로그램을 통해 UNIX System V Release 4 (및 자체 릴리즈인 UNIX System V, Release 5)에 기반한 코드로 된 운영 체제를 사용하고자 하는 모든 기업 및 개인들에게 라이센스를 제공하고 있다. 닷컴 붕괴로 유닉스 사업자들에게도 합병의 바람이 불게 되었다. 1980년대 태어난 많은 상업 유닉스 업체들 중에서 Hewlett-Packard의 HP-UX, IBM의 AIX, NeXT의 NEXTSTEP (나중에 OPENSTEP이 되었다가 이제 Mac OS X가 됨) 및 Sun의 Solaris 운영 체제들만이 아직도 시장에서 뛰고 있다; Digital Equipment Corporation, Data General, Santa Cruz Operation (현재는 Tarantella) 등은 다른 업체에 합병되거나 사업을 접었다. 리눅스와 오픈-소스 BSD의 등장도 기존의 상업 유닉스 영역을 침범하였다.

1976: UNIX Time-Sharing System 6th Edition(V6)
1978: V6 -> 1BSD 파생
1979: UNIX Time-Sharing System 7th Edition(V7)
1979: V7 -> UNIX 32V 파생
1980: UNIX 32V의 영향을 받은 3BSD 나옴
1983: UNIX System V 나옴 (V5 이래 분화되어 V6, V7 등의 영향을 받음)
1983: 4.2BSD 나옴
1985: 4.2BSD -> Mach 파생
1986: 4.3BSD 나옴

2)유닉스의 특징

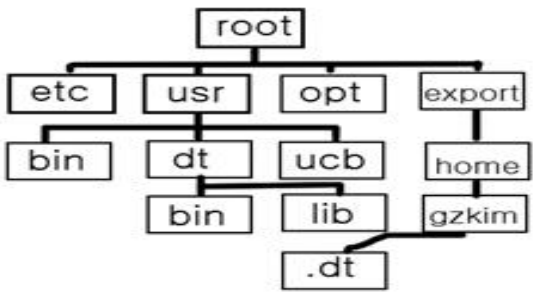
유닉스는 아주 일관된 운영체제이다. 시스템 자체가 원칙을 지키고 있어서 다른 운영체제보다 익히기가 쉽다. 윈도우 계열의 운영체제는 시스템 자체가 사람의 눈에 띄게 만들어서 접하기가 쉽지만, 일정수준 이상으로 시스템에 대해 알기엔 무척 어렵다. 하지만 솔라리스와 같은 유닉스 시스템은 처음엔 쓰기가 거북하고 어렵지만 어느 정도 익히게 되면 한쪽에서 익힌 기능을 다른 곳에서도 똑같이 사용할 수 있어 고급 사용자로 발전하기가 쉽다.

① 대화식 운영체제: 명령어를 입력받기 위해 쉘 프롬프트($)를 화면에 나타난다. 프롬프트가 나타난 상태에서 사용자가 명령을 입력하면 시스템은 그 명령을 수행하고 다시 새로운 명령을 받기 위해 대기하고 있다는 표시로 쉘 프롬프트($)를 나타낸다.
② 다중 작업 기능(multi-tasking): 한 번에 하나 이상의 작업을 수행하는 것을 말한다.
③ 다중 사용자 기능: 여러 대의 단말기(키보드와 모니터)가 하나의 컴퓨터에 연결되어 각 단말기에서 사용자들이 프로그래밍을 하거나 파일 편집을 동시에 수행할 수 있다. 즉, 여러 사람이 동시에 유닉스 시스템을 사용하여 개개의 작업을 수행할 수 있다.
④ 이식성(하드웨어 종류에 상관없이 운영되는 특성): 유닉스는 90% 이상이 C 언어로 구현되어 있고, 시스템 프로그램이 모듈화 되어 있어 다른 하드웨어 기종으로의 이식이 용이하다. 즉, 다른 기종으로 이식할 경우 하드웨어에 의존하는 부분인 어셈블리어로 작성된 부분을 새로운 환경으

로 변환시키고, C 언어로 구현된 나머지 부분을 재컴파일하여 실행하면 된다.
⑤ 계층적 트리 구조 파일 시스템: 유닉스는 계층적 트리 구조를 가짐으로써 파일 관리를 용이하게 한다.
⑥ 개발 도구 : 프로그래머가 여러가지 언어(Fortran, C, C++ 등)를 사용하여 프로그래밍할 수 있도록 많은 컴파일러(compiler)를 제공하고 있다.
* 컴파일러(compiler) : 명령어 번역 프로그램
⑦ 통신 : 유닉스 시스템은 서로 다른 컴퓨터와 통신 가능하도록 하기 위해 여러 가지의 통신 유틸리티(mail, ftp, telnet 등)를 제공한다.
⑧ 가상 메모리 : 한정된 메모리를 갖는 시스템에서 실제 메모리보다 더 큰 프로그램을 수행하기 위해 가상 메모리(virtual memory) 기법을 사용한다.

3)유닉스의 구성
유닉스는 크게 시스템의 하드웨어 관리 작업을 비롯하여 시스템이 운영되기 위해서 필요한 기본적인 관리 작업을 하는 커널, 사용자 사이에서 다리 역할과 명령어 해석기 역할을 하는 셸, 하드디스크나 CD-ROM과 같은 저장 장치에 데이터를 효율적으로 관리하는 디렉터리 트리, 이 세 가지 요소로 구성된다.

① 커널은 유닉스에서 가장 핵심적인 부분으로 시스템의 하드웨어 관리와 관련된 작업을 주로 담당한다. 즉, 시스템 부팅 시에는 시스템 자체의 초기화를 위하여 현재 설치된 하드웨어의 초기화나 필요한 프로세스를 띄우는 역할을 담당하며, 그 이후에는 하드웨어, 스왑 공간, 프로세스, 메모리, 파일시스템 등의 관리를 담당한다. 인간의 뇌 중에서 간뇌에 해당하며 가장 기본적인 운영체제의 작업을 커널에서 하게 된다. 또한 셸에서 하드웨어에 데이터를 읽거나 기록하려고 할 때, 하드웨어와 이들 사이에서 다리 역할을 한다.
② 셸을 한마디로 표현하면 명령어 해석기이다. 사용자가 셸 프롬프트에서 내린 명령어를 해석하여 커널에게 전달하고, 커널이 실행한 것을 처리하여 사용자에게 전달하는 역할을 한다. 커널이 하드웨어와 셸 사이에서 다리 역할을 한다면, 셸은 커널과 사용자나 실행한 애플리케이션 사이에서 다리역할을 한다.



[그림1] 디렉토리 트리

③ 디렉터리트리는 유닉스에서 데이터를 하드디스크에 저장하기 위하여, 디렉터리와 파일로 이루어진 디렉터리 트리라는 구조를 사용한다. 이 구조는 파일을 조직적으로 보관하기 위하여 디렉터리를 쓰고 있고, 서로 연관된 모습이 나무를 거꾸로 뒤집은 놓은 듯 한 모습과 흡사하다하여 디렉터리 트리라한다.

1-3.솔라리스의 특징

현재 우리가 사용하는 유닉스 시스템은 수십가지에 이른다. 워크스테이션이나 서버 시장에서 최적화한 유닉스를 판매하는 회사 중에서 우리가 이름만 들어도 알 수 있는 회사들도 여러개가 있다. IBM의 AIX, HP의 HP_UX, SGI의 IRIX, 컴팩의 True64, 썬마이크로시스템즈 (이하 썬)의 솔라리스가 가장 대표적이며 인텔 호환 PC에서 작동하는 유닉스도 Minix, Linux, FreeBSD, NerBSD, OpenBSD, 솔라리스, BSDI, 넥스트스텝, Unixware, OpenServer 등을 비롯해 수많은 변종들이 존재한다. 이러한 수십 가지의 유닉스 중에서 시장 점유율이 가장 높은 것은 썬의 솔라리스이다. 그 이유가 무엇인지 알아보자. 솔라리스는 원래 썬의 하드웨어인 SPARC 계열로만 개발이 되었으나 인텔 기계 쪽으로도 이식되어 일반 사용자들도 솔라리스의 강력함을 느낄 수 있도록 하였다. 15년 이상 썬은 “The Network is The Computer"라는 구호 아래 유닉스 환경의 선도적인 역할을 하였기에 네트워크 기능이 매우 강력하고, 오랫동안 검증된 구조로 발전되어 확장성이 용이하고 안정성과 보안을 제공하면서도 관리상 오류와 문제 해결 방법을 단순화하여 관리비용을 줄일 수 있다.

많은 대학에 썬의 SPARC 기계와 솔라리스를 제공한 것이 썬이 유닉스 시장의 강자로 군림한 큰 이유가 된다. 컴퓨터를 사용하는 사람은 누구나 자신이 평상시에 많이 사용하던 익숙한 환경을 좋아한다. 이러한 경향은 일반 사용자보다 프로그래머에게 특히 강하게 나타나는 데 프로그래머에게 개발 환경이 바뀌면 엄청난 시간과 노력을 다시 투자해야 이전의 환경에서 자신의 능력과 비슷하게 된다. 대학에서 자주 썬 기계를 접하던 프로그래머들이 졸업한 후에 어떤 기계를 선호할까? 당연히 썬의 기계와 솔라리스라는 환경을 선호 할 것이다. 물론 썬 시스템 안정성과 사용의 편리함도 빼놓을 수 없는 장점이 될 것이다. 우수한 기술, 저렴한 가격, 뛰어난 안정성, 사용의 편리함에 덧붙여 썬의 오픈소스 정책을 통해 자신의 기술이나 개발에 관련된 수많은 기술을 외부와 공유하여 지금까지 이르렀다. 또한 썬은 솔라리스 8부터 솔라리스를 라이선스 비용을 받지 않고 무료로 배포한다는 놀라운 발표도 하였다. 솔라리스를 경험하기 위해서는 썬 홈페이지에서 아주 저렴한 가격에 구매해서 사용할 수 있다.

① 관리가 쉽다 : 시스템의 기능이 추가되고 발전함에 따라 상대적으로 시스템을 관리하는 것은 어려워진다. 그러나 솔라리스는 많은 관리 도구를 기본적으로 제공하기 때문에 오히려 관리가 용이해진 특징이 있다. 물론 저걸 다 배운다는 건 힘들다. 그래도 중요한 것들부터 배워나가면서 생각할 것이 “어차피 있는 기능들을 쓴다”는 생각으로 하게 되므로 복잡하게 되지 않는다.

② 신뢰성이 높다 : 산업체 표준인 케르베로스(Kerberos) V5, IP Sec 등의 암호화 프로토콜을 지원하며 스마트카드 리더를 통해서도 스마트카드 인증을 할 수 있다. 또한 RBAC(Role Based Access Control)를 사용하면 시스템 관리자가 암호를 노출시키지 않고도 다른 일반 사용자에게 일부 관리 기능을 제공할 수 있다.

③ 네트워크가 강력하다 : 솔라리스는 네트워크와 관련된 30년의 풍부한 경험을 이용하여 현존하는 대부분의 전통적인 네트워크 프로토콜과 IPv6, LDAP, XML, J2EE 등 최신 표준을 OS 차원에서 지원한다. Solaris Netlink를 사용하면 윈도우 NT 서버 시스템이 하는 역할을 대신 처리할 수 있어서 다른 윈도우 시스템과 인증, 파일 공유, 프린터 공유 등의 서비스를 아무런 제약 없이 이용할 수 있다.

④범용성이 뛰어나다 : 현재 대부분이 기업들은 하나의 시스템만 사용하는 환경이 아니다. 여러 시스템을 각기 용도에 따라 복합적으로 사용한다. 솔라리스에서는 PC, 워크스테이션, 메인 프레임 등을 손쉽게 연동하여 쓰는 기능을 제공한다. 또한 단일 제품 내에서 세계의 모든 언어를 지

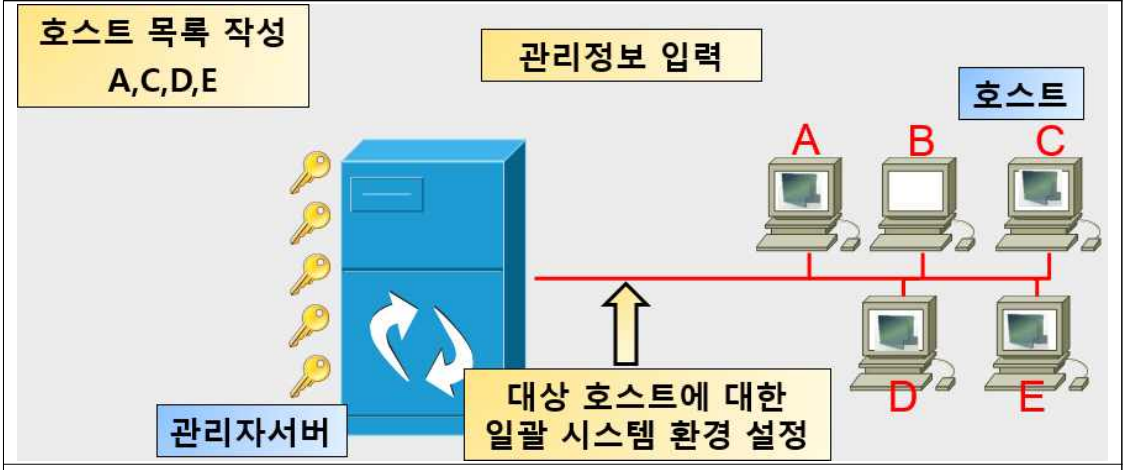
원하며, 어떠한 언어 환경이라도 솔라리스가 운영되는 도중에 추가하거나 제거할 수 있다
⑤확장성이 좋다 - 솔라리스는 멀티 프로세싱 환경을 고려하여 설계되어 확장성이 뛰어나다. 한 시스템에서 백만 개 이상의 프로그램을 동시에 실행할 수 있고 최대 128개의 CPU를 사용할 수 있다.

2-1 시스템 구축 환경

가격이 저렴하고 안정성, 여러 종류의 서버와 클라이언트를 통합하는 확장성, 일반 데스크탑에도 설치가 가능하다는 장점을 가진 Solaris를 사용하였고, vi 편집기와 ipmon 등을 도구로 사용하였다.

2-2 시스템 구성도

다음 그림과 같이 관리자서버와 호스트가 연결되어 있다. 호스트는 필요에 따라 다양한 방식으로 관리자 서버로부터 파일을 가져 올 수 있다. 자세한 설명은 2-3 참고



[그림 2] 시스템 구성도

2-3 시스템 내용 및 용어 설명

1) IPFilter

IPFilter는 솔라리스에서 무료로 사용할 수 있는 오픈 소스 방화벽(Firewall)이며, 솔라리스 10에는 기본으로 포함되어있다. 썬에서 개발한 방화벽 소프트웨어인 SunScreen을 대체할 예정이다. 상용 방화벽 장비보다는 못하지만 공개 소스치고는 성능이 매우 좋다.

방화벽은 주로 들어오는 패킷을 막는데 사용한다.
방화벽에는 룰이라는 것이 있는데 어떻게 설정하냐에 따라 어떤 패킷을 막을지 통과시킬지를 정할 수 있다.

```
block in all
block out all
```

[그림 3] 방화벽 룰 1

이 룰은 모든 패킷을 차단 하는 룰이다.

```
block in all
pass out all
```

[그림 4] 방화벽 룰 2

이 룰은 들어오는 패킷은 막고 나가는 패킷은 허용한다.

```
block in all
pass out on pcn0 proto tcp from any to any keep state
pass out on pcn0 proto udp from any to any keep state
pass in all
```

[그림 5] 방화벽 룰 3

위와 같은 룰이 있다면 처음엔 막으려고 생각했는데, 마지막줄에서 모두 통과시킨다. 결과적으로 인바운드 패킷은 통과한다. 물론 위 예는 극단적인 것이다. 따라서 더 이상 룰을 확인할 필요 없다는 것을 IPF에 명시해줘야 한다. 그 지시어가 quick이다.

```
block in quick all
pass out on pcn0 proto tcp from any to any keep state
pass out on pcn0 proto udp from any to any keep state
pass out on pcn0 proto icmp from any to any keep state
pass in all
```

[그림 6] 방화벽 룰 4

quick은 더 이상 아래의 룰을 확인할 필요 없다는 의미이다.

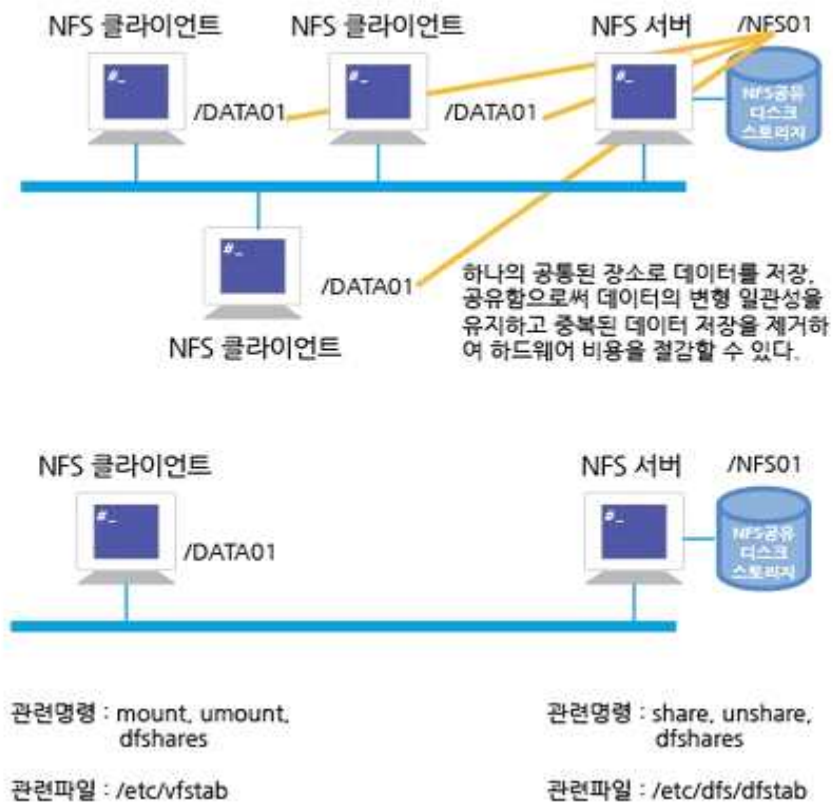
첫 번째 줄에 quick 지시어를 추가했다. 모든 인바운드 패킷은 첫 번째의 룰에서 블록/통과 여부가 블록으로 결정된다. quick지시어가 있기 때문에 이 룰을 만족한 모든 패킷은 아래의 룰을 확인하지 않는다. 마지막 줄에 pass in all이 있지만 첫 번째 룰만 보고 더 이상 확인하지 않기 때문에 결과적으로 패킷은 블록된다.

2)NFS

솔라리스를 비롯한 유닉스에서 네트워크로 하드디스크를 디렉터리 단위로 공유하는 방법과 윈도우에서 하드디스크를 디렉터리 단위로 공유할 때 사용하는 방법은 서로 다르지만, 기본 개념은 비슷하다. 네트워크로 하드디스크를 공유하려는 서버 측에서 자신의 하드디스크를 디렉터리 단위로 공유 할 수 있으며, 클라이언트 측에서는 서버에서 공유한 디렉터리를 자신의 특

정 디렉터리에 마운트 하여 사용 할 수 있다. 마운트 한 디렉터리는 자신의 하드디스크에 있는 디렉터리와 구별 없이 사용할 수 있다. 단 시스템이 서로 다르고, 그에 따라서 관리자도 다를 수 있으며, 네트워크로 공유하기 때문에 보안에 관련된 몇 가지 제한 사항이 있다. 솔라리는 윈도우와 같은 GUI 환경을 지원하지 않기 때문에 서버 측과 클라이언트 측에서 시스템 파일을 변경하거나 명령어를 사용해서 작업해야한다.
NFS는 네트워크 파일시스템(Network File system)을 나타내며 네트워크상에 존재하는 시스템간에 저장 공간을 공유하기 위해 개발된 서비스이다. 리눅스와 유닉스 시스템에 주로 사용이 되고 있는 기술이며 과거 디스크의 비용이 비쌌던 시절에 많은 시스템에 대해 디스크에 대한 비용을 줄이기 위해 만들어 졌다고 한다.

NFS 이해도



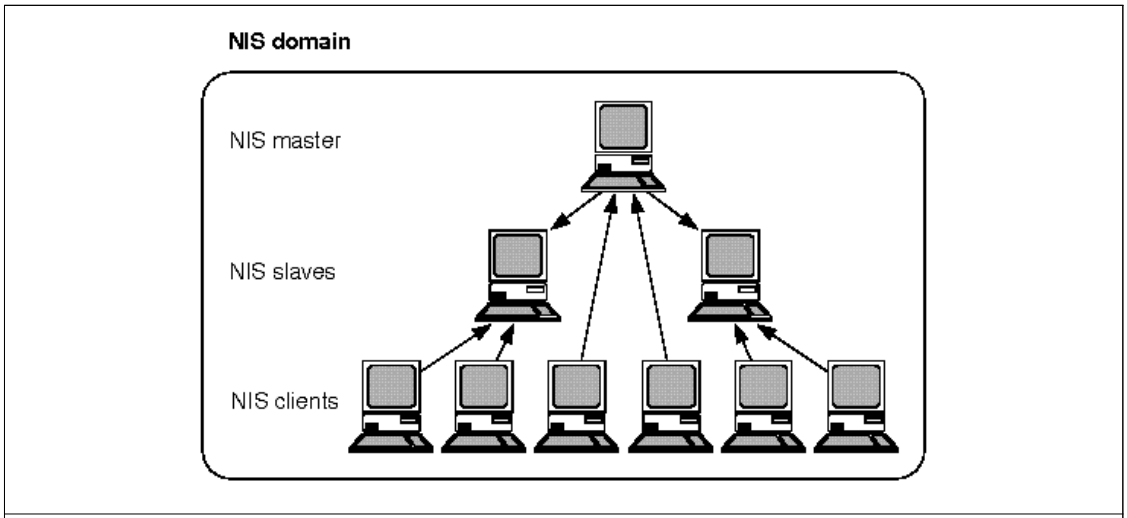
[그림 7] NFS 이해도

NFS로 공유한 디렉터리는 자신의 하드디스크에 있는 디렉터리와 사용함에 있어서 차이점을 느끼지 못하지만. 설정하는데 있어서는 많은 차이점이 있다. NFS는 수많은 프로토콜, 디몬 프로세스, 시스템파일, 로그 파일, 명령어로 이루어진 거대한 파일시스템이다. 서버측은 자신의

하드디스크 디렉터리를 공유하기 위해서 여러 시스템 파일이나 디몬 프로세스를 띄어야 하지만, 클라이언트 측에서는 자신의 하드디스크를 마운트해서 사용하는 것과 큰 차이점이 없이 설정이 가능하다. 클라이언트 측은 부가적인 디몬 프로세스가 필요하다는 것 이외에는 사용하는 명령어나 시스템 파일은 다르지 않다.

3) NIS

NIS는 Network Information Service의 약자로 /etc 디렉토리의 파일을 네트워크로 공유하는 가장 대표적인 네이밍 서비스로서 유닉스 환경에서 시스템의 정보를 공유하는 표준 환경이다. 80년대 썬에서 만들었으며 현존하는 상용 유닉스와 공개형 유닉스 모두 NIS를 제공한다. NIS를 사용 할 때 가장 큰 장점은 네트워크를 비롯한 시스템의 중요한 설정 파일을 각각의 시스템이 독자적으로 가지고 있지 않고, 한 대의 시스템에서 가진 설정 파일을 NIS를 사용하는 모든 시스템이 공유한다는 점이다. 네트워크에 관련된 정보나 사용자에 대한 정보를 서버 시스템에서 변경하면 클라이언트에서는 따로 변경을 하지 않아도 자동으로 변경된 정보를 가져가고, 새로운 시스템이 추가되어도 NIS만 설정하면 네트워크에 관한 정보나 사용자에 관한 정보는 NIS로부터 자동으로 가져오므로 간단하게 설정할 수 있다.



[그림 8] NIS 구성도

NIS를 사용하는 시스템은 각 기능에 따라 크게 세 가지 종류(NIS 마스터 서버, NIS 슬레이브 서버, NIS 클라이언트)로 나뉠 수 있으며 NIS를 사용하는 시스템은 이 세 가지 종류 중에서 반드시 하나에 속한다.

① NIS 마스터 서버
NIS 도메인에는 반드시 하나가 존재하는 서버가 NIS 마스터 서버이다. NIS 마스터 서버는 /etc 디렉토리 안에 있는 시스템 파일을 직접 처리해서 만든 원본 NIS 맵 데이터를 저장한다. NIS 네임 스페이스의 여러 정보를 변경할 수 있는 시스템은 오로지 NIS 마스터 서버뿐이다.

NIS 마스터 서버에 있는 원본 NIS 맵 데이터를 사용하여 NIS 슬레이브 서버나 NIS 클라이언트가 요구하는 정보를 제공하며, 원본 맵 데이터가 변경되면 NIS 슬레이브 서버에 변경된 맵 데이터를 전달(push) 하는 기능도 갖추고 있다. 갱신된 NIS 맵 파일을 NIS 슬레이브 서버에 푸시하기 위해 NIS 마스터 서버에서는 yppush 명령어를 사용한다. NIS 마스터 서버에서 yppush 명령어를 사용하면 NIS 마스터 서버의 ypserv 디몬 프로세스가 자신에게 등록된 NIS 슬레이브 서버의 ypserv 프로세스에게 NIS 맵 파일이 갱신되었으니 최신의 NIS 맵 파일을 가져가라고 지시한다. 이 지시를 받은 NIS 슬레이브 서버의 ypserv 디몬 프로세스는 ypxfr 명령어를 실행하여 NIS 마스터 서버의 ypxfrd 디몬 프로세스를 통해 NIS 맵 파일을 전송받는다.

② NIS 슬레이브 서버
NIS 마스터 서버가 한 대 있다면 NIS 클라이언트의 수가 많아짐에 따라서 NIS 마스터 서버에 부하가 많이 걸린다. 또한 여러 이유에 의해서 NIS 마스터 서버에 접속할 수 없는 경우가 발생하면 NIS 클라이언트는 자신이 필요한 데이터를 찾지 못하여 시스템이 오작동을 할 우려도 있다. 이런 상황에 대비하여 사용되는 시스템을 NIS 슬레이브 서버라 부른다.

NIS 도메인에 존재하는 NIS 마스터 서버의 개수는 오로지 한 개로 제한되지만 NIS 슬레이브 서버의 개수는 제한이 없다. NIS 도메인에 NIS 슬레이브 서버가 전혀 존재하지 않아도 상관없으며, 수 십대 혹은 수 백대의 NIS 슬레이브 서버가 동작해도 전혀 상관없다. /etc 디렉토리의 시스템 파일로부터 직접 만든 원본 NIS 맵 파일을 가지고 있는 NIS 마스터 서버와 달리 NIS 슬레이브 서버는 NIS 마스터 서버에서 가져온 NIS 맵 파일을 가지고 있다. NIS 슬레이브는 다른 NIS 슬레이브 서버나 NIS 클라이언트가 요구하는 정보를 제공한다.

NIS 마스터 서버의 원본 맵 파일을 NIS 슬레이브 서버가 복사하는 방법에는 두 가지가 있다. 첫 번째는 NIS 마스터 서버에서 맵 파일이 변경될 경우에 해당 맵 파일만 NIS 서버에 전달하는 방법이고, 두 번째는 NIS 슬레이브 서버에서 주기적으로 NIS 마스터 서버에 접속하여 현재 맵 파일을 복사해서 가지고 오는 방법이다.
첫 번째 방법을 푸시라고 부르며, 두 번째 방법을 풀(pull)이라 부른다. NIS 슬레이브 서버에서는 맵 데이터를 최신 데이터로 유지하기 위해서 푸시와 풀 두 방법을 모두 사용할 수 있다. 단 풀은 NIS 슬레이브 서버가 직접 맵 파일을 가지고 오기 때문에 별다른 설정이 필요 없이 원활하게 동작하지만, 푸시는 NIS 마스터 서버가 NIS 슬레이브 서버에만 맵 파일을 전달한다. 새로 추가된 NIS 슬레이브 서버가 푸시를 사용하여 맵 파일을 전송하기 위해서는 반드시 NIS 마스터 서버에 등록하는 과정이 필요하다.

③ NIS 클라이언트
NIS 클라이언트는 자신이 필요한 정보를 NIS 마스터 서버나 NIS 슬레이브 서버에서 가져온다. NIS 슬레이브 서버는 자신이 서버의 역할을 하고 있기 때문에 모든 NIS 맵 파일을 복사하여 가지고 있지만, NIS 클라이언트는 필요한 정보가 있을 때마다 NIS 마스터 서버나 슬레이브 서버에 접속해서 자신에게 필요한 정보만 가져온다. NIS 클라이언트는 NIS 맵 파일을 전혀 가지고 있지 않는다. NIS 클라이언트는 필요한 정보가 있을 때만 NIS 마스터 서버나 NIS 슬레이브 서버에 접속하고, 현재 접속중인 서버가 정상적으로 동작하지 않으면 즉시 다른

서버로 재접속하여 필요한 정보를 가지고 온다.
NIS 클라이언트가 NIS 마스터 서버나 NIS 슬레이브 서버가 접속하는 것을 바인드(bind)라고 부른다. 모든 NIS 마스터 서버와 NIS 슬레이브 서버는 항상 NIS 클라이언트를 겸하고 있다. NIS 마스터 서버와 NIS 슬레이브 서버는 클라이언트가 요구하는 정보를 제공하기도 하지만, 자신이 운영되기 위해서 필요한 정보도 직접 제공한다.

2-4 시스템 구축 및 평가

1) IPFilter

```
pass in quick on e1000g0 from 192.168.101.151 to 192.168.101.150 port=23
block in log quick on e1000g0 from any to 192.168.101.150 port=23
```

[그림 9] 시스템 룰 설정

솔라리스에 있는 방화벽인 ipfilter의 룰 설정 내용이다.
ip가 192.168.101.150 이고 port번호가 23인 사용자를 제외하고 모두 차단한다.

```
# ftp 192.168.191.150
ftp: connect: 연결이 거부됨
ftp> ls
Not connected.
ftp>
```

[그림 10] ftp 차단

그래서 port번호가 21번인 ftp는 차단 당한다.

```
# telnet 192.168.101.150
Trying 192.168.101.150...
Connected to 192.168.101.150.
Escape character is '^]'.
login: test1
Password:
Last login: Fri Mar 21 12:47:12 from 192.
Oracle Corporation      SunOS 5.10      C
$
```

[그림 11] 차단 되지 않고 통과

반면 port번호가 23번, IP가 192.168.101.150인 이 사용자는 방화벽에 의해 차단되지 않는다.

```
# svcadm disable ipfilter
# svcadm enable ipfilter
# /usr/sbin/ipmon
21/03/2014 13:02:05.283732 e1000g0 @0:3 b 192.168.101.151,32936 -> 192.168.101.150,21 PR tcp len 20 52 -S IN
```

[그림 12] 도구 ipmon을 이용한 로그 기록

로그를 남기는 도구 ipmon을 이용하여 이렇게 로그를 남길 수도 있다.

2) NFS

1. 파일 공유 시스템

```
# hostname
server
# vi /etc/hosts
"/etc/hosts" [읽기 전용] 8 행, 131 문자
#
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.168.148.4   server  loghost
192.168.148.7   client
192.168.148.6   user
```

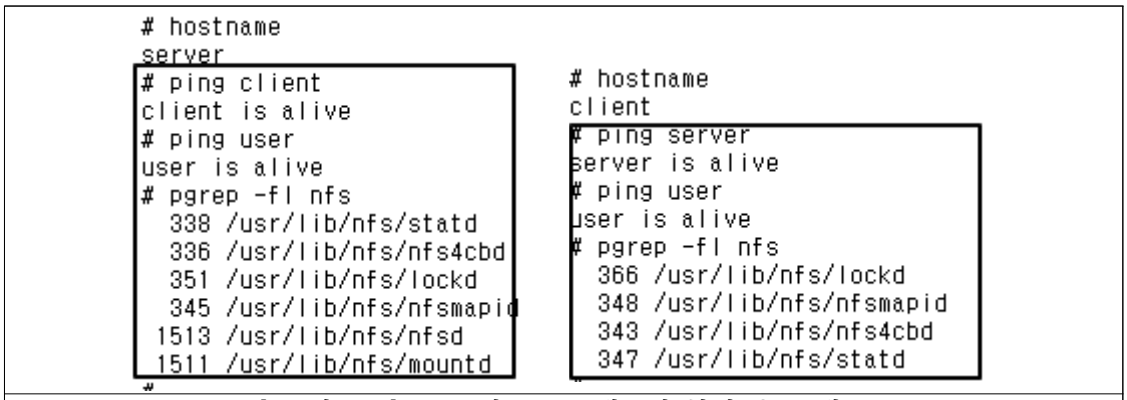
[그림 13] server의 /etc/hosts 내용

호스트의 이름은 SERVER이며, vi /etc/hosts를 이용하여 연결할 호스트의 ip와 호스트 명이 적혀있다.

```
# hostname
client
# vi /etc/hosts
"/etc/hosts" [읽기 전용] 8 행, 131 문자
#
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.168.148.7   client  loghost
192.168.148.6   user
192.168.148.4   server
~
```

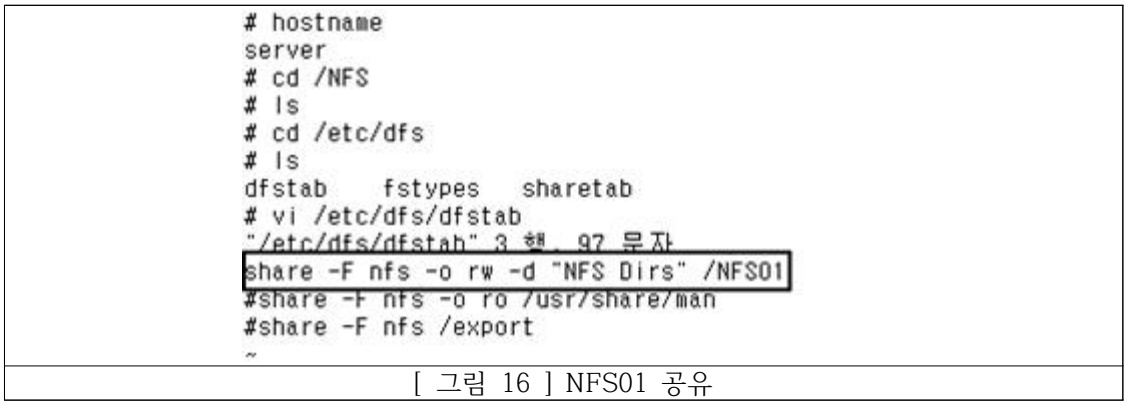
[그림 14] client의 /etc/hosts 내용

호스트의 이름은 CLIENT이다. 설정은 위와 동일하다.



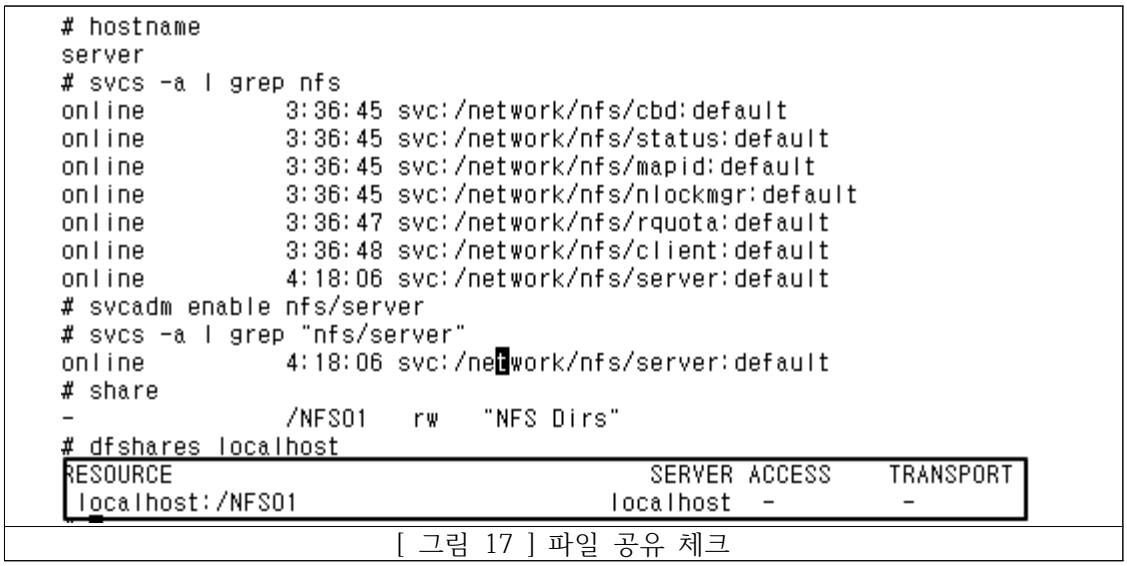
[그림 15] NFS 디몬 프로세스의 활성 유무 체크

PING을 이용하여 연결 상태를 확인하고, NFS의 디몬 프로세스의 활성 유무를 체크한다.
연결 상태가 양호 하다면, 다음으로는 서버 측에서 파일 공유를 하기위해 /etc/dfs/dfstab으로 들어간다.



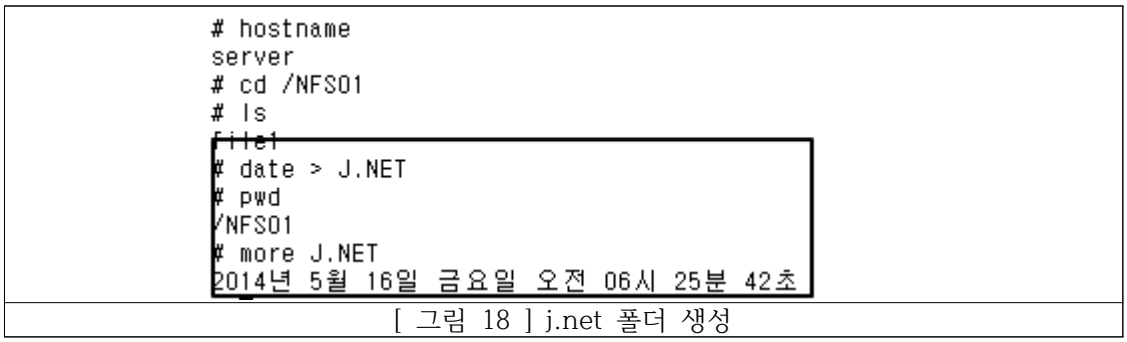
[그림 16] NFS01 공유

서버 측에서는 공유할 폴더를 미리 지정하고, 없다면 mkdir을 이용하여 폴더를 만든다. 여기서 임의로 NFS01을 공유하기로 설정했다.



[그림 17] 파일 공유 체크

svcadm 명령어를 이용하여 NFS서버임을 활성화 시키고, 파일 공유가 실제로 되고 있는지 다시 체크한다.



[그림 18] j.net 폴더 생성

공유를 설정한 폴더에 접근하여 J.NET이라는 파일을 만들어 본다. 여기서 주목할 점은 호스트 네임이 서버라는 점과 파일을 만든 시점이다. 공유 서비스 상태가 원활한지 알아보기 위해 파일을 받을 클라이언트에 접속하여 이를 확인해본다.

```
# hostname
client
# dfshares server
RESOURCE                                  SERVER ACCESS    TRANSPORT
    server:/NFS01                         server  -         -
# mkdir /DATA01
# mount -F nfs server:/NFS01 /DATA01
nfs mount: mount: /DATA01: 장치 사용 중
# df -hF nfs
파일시스템             크기   사용   가용   용량    설치지점
server:/NFS01          8.6G   3.9G   4.6G    46%    /DATA01
# cd /DATA01
# ls
J.NET  file1
# more J.NET
2014년 5월 16일 금요일 오전 06시 25분 42초
```

[그림 19] 클라이언트 측에서 정상적으로 마운트

클라이언트 측에서도 NFS01폴더를 DATA01폴더로 마운트하여 J.NET이라는 파일을 정상적으로 사용하는 것을 볼 수 있다.

3) NIS

```
# ypinit -m

In order for NIS to operate sucessfully, we have to construct a list of the
NIS servers.  Please continue to add the names for YP servers in order of
preference, one per line.  When you are done with the list, type a <control D>
or a return on a line by itself.
        next host to add:  server
        next host to add:
```

[그림 20] 마스터 서버 생성

NIS 마스터서버를 생성한다.

```
# /usr/lib/netsvc/yp/ypstart
starting NIS (YP server) services: ypserv ypbind ypxfrd rpc.yppasswdd rpc.ypupda
ted done.
```

[그림 21] 마스터 서버 실행

NIS 마스터서버를 실행한다.

```
# domainname
JNET.nis.domain
# ypinit -c

In order for NIS to operate sucessfully, we have to construct a list of the
NIS servers.  Please continue to add the names for YP servers in order of
preference, one per line.  When you are done with the list, type a <control D>
or a return on a line by itself.
        next host to add:  server
        next host to add:
```

[그림 22] 슬레이브 서버 생성

NIS 슬레이브 서버를 생성한다. server 라고 쓴 부분은 마스터서버의 호스트네임이다.

```
# /usr/lib/netsvc/yp/ypstart
starting NIS (YP server) services: ypbind done.
```

[그림 23] 슬레이브 서버 실행

NIS 슬레이브 서버를 실행한다.

```
# useradd -u 500 -d /work love
# passwd love
새 암호:
새 암호를 다시 입력하십시오:
passwd: 암호(love용)가 성공적으로 변경되었습니다.
```

[그림 24] 마스터 서버에서 사용자 계정 생성

NIS 마스터 서버에서 uid값을 500으로 갖는 love 라는 계정을 생성한다.
제대로 생성이 되었는지 확인하기 위해 사용자 전환을 시도한다.

```
# su love
$ whoami
love
```

[그림 25] 생성한 계정 로그인 확인

```
# cd /var/yp
# /usr/ccs/bin/make
updated passwd
pushed passwd
updated netid
pushed netid
updated ageing
updated user_attr
pushed user_attr
```

[그림 26] 맵 서버 업데이트

사용자 계정이 추가되어 /etc/passwd 설정이 변경되었으므로 마스터서버에서 맵 서버를 업데이트한다.

```
# su love
su: 알 수 없는 id: love
```

[그림 27] 슬레이브 서버에서 로그인 시도

마스터 서버에서 생성한 사용자 계정을 슬레이브 서버에서 사용해 로그인을 시도해보지만, 슬레이브 서버 /etc/passwd 파일 안에는 love 계정 내용이 없다. 그래서 결국 로그인이 되지 않는다.

그렇다면 passwd 라는 맵 파일에는 love 계정에 대한 내용이 있을까

```
# ypmatch love passwd
love:HsEJ26SqnoGJs:500:1::/work:/bin/sh
```

[그림 28] passwd 맵 파일 확인

학인해보니 맵 파일에는 관련 내용이 업데이트가 되어 있다.

```
# ypinit -s server


Installing the YP database will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n]  n
OK, please remember to go back and redo manually whatever fails.  If you
don't, some part of the system (perhaps the yp itself) won't work.
The yp domain directory is /var/yp/JNET.nis.domain
There will be no further questions. The remainder of the procedure should take
a few minutes, to copy the data bases from server.
Transferring audit_user...
Transferring user_attr...
Transferring prof_attr...
Transferring exec_attr...
Transferring auth_attr...
Transferring ageing.byname...
Transferring auto.home...
```

[그림 28] 슬레이브 서버에서 맵 파일 다운로드

슬레이브 서버가 마스터 서버로부터 맵 파일을 받아 갱신하였다.

```
# su love
$ whoami
love
$
```

[그림 29] 슬레이브 서버에서 로그인 성공

그 결과 마스터 서버에서 만든 사용자 계정으로 슬레이브 서버에서도 로그인이 성공한다.
슬레이브 서버 내에 추가하지 않은 사용자 계정으로 로그인이 되는 이유는 마스터 서버에서 맵 서버에 설정 파일을 푸시했기 때문이고 슬레이브 서버는 마스터 서버와 연결이 되어 있기 때문에

3. 결론

관리자 한 명이 많은 pc를 관리하려면 번거롭고 시간과 노력이 많이 필요하기 때문에 좀 더 편리하고 효율적으로 관리할 방법을 찾기 위해 고안되었다.
보안상의 문제를 해결하기 위해 Solaris 내에서 ipfilter를 방화벽으로 사용하였고 효율적인 관리를 위해 NIS와 NFS 등의 네이밍 서비스를 이용하였다.
NIS는 네트워크를 비롯한 시스템의 중요한 설정 파일을 각각의 시스템이 독자적으로 가지고 있지 않고, 한 대의 시스템에서 가진 설정 파일을 NIS를 사용하는 모든 시스템이 공유를 하기 때문에 네트워크에 관련된 정보나 사용자에 대한 정보를 서버 시스템에서 변경하면 클라이언트에서는 따로 변경하지 않아도 자동으로 변경된 정보를 가져가고, 새로운 시스템이 추가되어도 NIS만 설정하면 네트워크에 관한 정보나 사용자에 관한 정보는 NIS로부터 자동으로 가져오므로 간단하게 설정할 수 있다는 장점이 있다.
NFS는 네트워크상에 존재하는 시스템 간에 저장 공간을 공유하기 위해 개발된 서비스로, 리눅스와 유닉스 시스템에 주로 사용이 되고 있는 기술이며 모든 클라이언트가 동일한 설정 파일을 사용할 수 있기 때문에 자료의 일관성과 신뢰성을 제공하고 다양한 운영체제에 분산된 파일 시스템을 접근하기 위한 싱글 포인트를 만들 수 있다는 장점이 있다.
이러한 여러 서비스의 장점을 잘 이용하여 많은 pc를 편리하고 효율적으로 관리 할 수 있다. 예컨대 50대의 pc가 있는 실습실 pc의 설정을 하나하나 바꾸려면 번거롭고 많은 시간이 들 것이다. 하지만 이와 같이 연구를 해봄으로서 관리자가 많은 pc를 관리할 때 생기는 번거로운 점을 해소할 수 있고 효율적으로 관리할 수 있다는 것을 알 수 있었다.
관리자의 입장이 되어 만들어본 통합관리시스템은 파일 공유 면에서는 만족하였으나, 이외에는 부가기능이 부족하다고 생각했다. 자주 공유하는 자원을 분석하여 필요에 따라 네트워크의 과부하를 줄이고 방화벽의 룰 설정을 보강하는 것으로 방향을 잡았다.

NIS와 NFS는 이미 구축이 되어 있는 시스템이다. 그리고 상위버전에는 보안상의 위험으로 삭제된 내용도 있다. 하지만 이런 설정들을 직접 해봄으로써 어떤 점이 문제인지 알 수 있었고 유닉스의 기초를 다지는 좋은 계기가 되었다.

4-1. 참고 문헌

한빛미디어 - 초보 유닉스 시스템 관리자를 위한 Solaris Bible

한빛미디어 - 유닉스 이론과 실습

통합 관리시스템 구축

2014. 5. 27

학과 : 정보보호학과
지도교수 : 이병천 교수님
5조 J.NET(just.net)

목 차

조원 소개 및 역할

조장

08 정봉준 : 서버 및 방화벽 관리

조원

07 황택주 : 방화벽 룰 설정

09 안　빈 : 서버구축 및 권한관리

주제 선정 및 구상도

주제 선정

◆ 교내 실습시스템 중에서 운용되는 호스트의 경우 관리자가 통일한 운영체제 및 설정상태의 실습환경을 구축 할 때

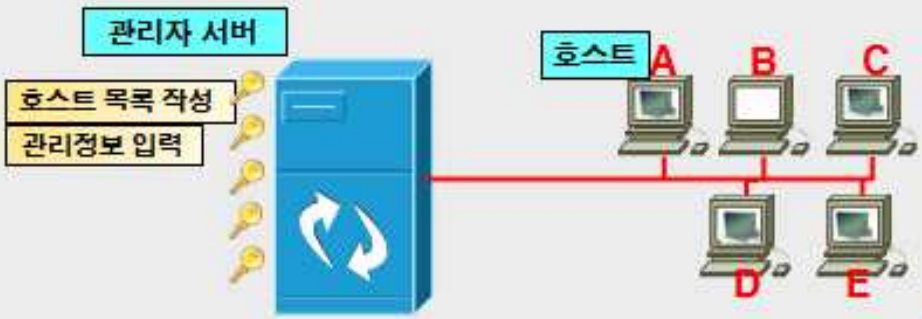
- 모든 실습자의 실습작업이 통일하지 않아 실습기간이 종료되면 호스트의 운영환경에 많은 변화가 수반되는 것이 불가피
- 또한 호스트의 수가 증가되거나 여러 기종의 호스트가 추가되는 상황도 예상됨에 따라 호스트의 실습환경 설정에 많은 변수와 비효율 요인이 발생

◆ 이러한 호스트 운영체제에서는 관리자 서버를 구축, 시스템의 운영환경 등을 주기적·체계적으로 설정·관리할 필요

주제 선정 및 구상도

관리자 서버

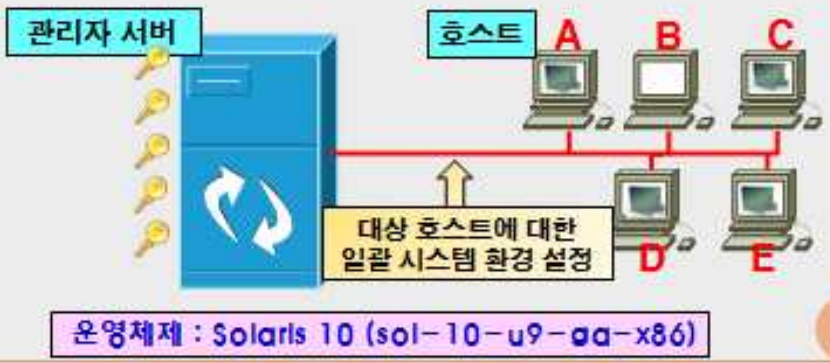
◆ 관리자 서버 : 호스트 목록 작성 및 관리용 데이터 입력
통을 통해 호스트의 운영환경 설정 및 보안관리



주제 선정 및 구상도

통합관리시스템

◆ 운영개념
- NFS/NIS를 통한 호스트의 운영환경 설정 및 관리
- 방화벽을 활용한 보안관리



운영체제 : Solaris 10 (sol-10-u9-ga-x86)

시스템 구현 설명

방화벽

◆ IPFilter : 솔라리스에서 무료로 사용할 수 있는 오픈 소스이며, 서버나 클라이언트를 외부로 보호할 때 설정

◆ 방화벽 룰 설정

```
pass in quick on e1000g0 from 192.168.101.151 to 192.168.101.150 port=23
block in log quick on e1000g0 from any to 192.168.101.150 port=23
```

IP 192.168.101.150, port 23을 제외하고 전부 차단

시스템 구현 설명

NFS

◆ NFS(Network file system) : 네트워크 파일시스템으로 클라이언트에서 다른 서버(시스템)의 자원을 자신의 자원처럼 사용이 가능
-> 검색,저장,수정

◆ 따라서 사용빈도가 높은 일반파일이나, 용량이 큰 파일을 공유하여 각 사용자들의 권한에 따라 파일들의 액세스가 가능

◆ 원활한 통신을 위해 서버와 클라이언트간 접속경로를 명시하고, 데몬 프로세스를 활성화하여 잠금 파일의 정보, 포트번호,사용자 및 시스템의 이름을 인식

시스템 구현 설명

NFS

데몬 프로세스 활성화

```
"/etc/dfs/dfstab" 12 행, 415 문자
# /etc/init.d/nfs.server start
# pgrep -fl nfs
  343 /usr/lib/nfs/statd
  347 /usr/lib/nfs/nfsmapid
  345 /usr/lib/nfs/nfs4cbd
  353 /usr/lib/nfs/lockd
 1098 /usr/lib/nfs/mountd
 1100 /usr/lib/nfs/nfsd
# share
-               /export/home    rw   ""
# dfshares
RESOURCE                                SERVER ACCESS    TRANSPORT
    server:/export/home                 server  -        -
```

```
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.168.148.7   client  loghost
192.168.148.6   user
192.168.148.4   server
```

접속경로

시스템 구현 설명

NIS

◆ NIS(Network Information Service) : /etc 디렉터리의 파일을 네트워크로 공유하는 가장 대표적인 네이밍 서비스

◆ 시스템의 중요한 설정 파일을 각각의 시스템이 독자적으로 가지고 있지 않고, 한 대의 시스템에서 가진 설정파일을 새로운 시스템이 추가되어도 간단하게 설정할 수 있음

❖ NFS와 같아 보이나 NIS는 시스템파일 위주라는 점과 공유 방식이 다름

시스템 구현 설명

NIS

◆ NIS(Network Information Service) : /etc 디렉터리의 파일을 네트워크로 공유하는 가장 대표적인 네이밍 서비스

◆ 시스템의 중요한 설정 파일을 각각의 시스템이 독자적으로 가지고 있지 않고, 한 대의 시스템에서 가진 설정파일을 새로운 시스템이 추가되어도 간단하게 설정할 수 있음

❖ NFS와 같아 보이나 NIS는 시스템파일 위주라는 점과 공유 방식이 다름

시스템 구축

NIS

- 마스터 서버(관리자서버)에서 시스템 환경설정을 바꾼 경우 이와 같이 업데이트가 된다.

```
# /usr/ccs/bin/make
updated hosts
pushed hosts
updated ipnodes
pushed ipnodes
updated netid
pushed netid
```

- 슬레이브 서버(호스트 A,C,D,E)의 경우 업데이트 된 내용이 자동으로 슬레이브 서버에 적용이 된다.
- 클라이언트(B)는 직접 마스터 서버에 접속하여 업데이트를 한다.

시스템 구축

방화벽 구축

포트번호 21번 FTP

```
# ftp 192.168.191.150
ftp: connect: 연결이 거부됨
ftp> ls
Not connected.
ftp>
```

포트번호 23번 텔넷

```
# telnet 192.168.101.150
Trying 192.168.101.150...
Connected to 192.168.101.150.
Escape character is '^]'.
login: test1
Password:
Last login: Fri Mar 21 12:47:12 from 192.
Oracle Corporation      SunOS 5.10      C
$
```

```
# svcadm disable ipfilter
# svcadm enable ipfi
# /usr/sbin/ipmon
21/03/2014 13:02:05.283732 e1000g0 @0:3 b 192.168.101.151,32936 -> 192.168.101.150,21 PR tcp len 20 52 -S IN
```

↓또한 무엇이 차단 되었는지 로그 기록이 남는다

시스템 구축

NFS를 활용한 공유폴더 설정

◆서버측 공유폴더 설정

폴더공유

```
# hostname
server
# cd /NFS
# ls
# cd /etc/dfs
# ls
dfstab    fstypes   sharetab
# vi /etc/dfs/dfstab
"/etc/dfs/dfstab" 3 행, 97 문자
share -F nfs -o rw -d "NFS Dirs" /NFS01
#share -F nfs -o ro /usr/share/man
#share -F nfs /export
```

```
server
# svcs -a | grep nfs
online          3:36:45 svc:/network/nfs/cbd:default
online          3:36:45 svc:/network/nfs/status:default
online          3:36:45 svc:/network/n
online          3:36:45 svc:/network/n
online          3:36:47 svc:/network/n
online          3:36:48 svc:/network/n
online          4:18:06 svc:/network/n
# svcadm enable nfs/server
# svcs -a | grep "nfs/server"
online          4:18:06 svc:/network/n
# share
-               /NFS01    rw    "NFS Dir
# dfshares localhost
RESOURCE                                  SERVER ACCESS    TRANSPORT
localhost:/NFS01                          localhost  -         -
```

```
# hostname
server
# cd /NFS01
# ls
# date > J.NET
# pwd
/NFS01
# more J.NET
2014년 5월 16일 금요일 오전 06시 25분 42초
```

파일 시간대 확인

시스템 구축

NFS를 활용한 공유폴더 설정

◆ 클라이언트측 파일 받기

클라이언트

```
client
# vi /etc/hosts
"/etc/hosts" [읽기 전용] 8 행, 131 문자
#
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.168.148.7   client  loghost
192.168.148.6   user
192.168.148.4   server
```

```
# hostname
client
# dfshares server
RESOURCE                                SERVER ACCESS    TRANSPORT
     server:/NFS01                      server  -        -
# mkdir /DATA01
# mount -F nfs server:/NFS01 /DATA01
nfs mount: mount: /DATA01: 장치 사용 중
# df -hF nfs
파일시스템             크기   사용   가용   용량   설치지점
server:/NFS01          8.6G   3.9G   4.6G    46%   /DATA01
# cd /DATA01
# ls
J.NET  file1
# more J.NET
2014년 5월 16일 금요일 오전 09시 25분 42초
```

동일한 시간대의 파일이 공유되고 있다.
클라이언트의 대수에 상관없이 동일한 방식으로 적용할 수 있다.

시스템 구축

NIS를 활용한 시스템 파일 공유

◆ 마스터 서버에서 계정 생성

```
# useradd -u 500 -d /work love
# passwd love
새 암호:
새 암호를 다시 입력하십시오:
passwd: 암호(love용)가 성공적으로 변경되었습니다.
```

NIS 마스터서버에 uid값을 500으로 갖는 사용자 계정 생성

```
# cd /var/yp
# /usr/ccs/bin/make
updated passwd
pushed passwd
updated netid
pushed netid
updated ageing
updated user_attr
pushed user_attr
```

변경 된 설정이 맵 파일에 저장됨

시스템 구축

NIS를 활용한 시스템 파일 공유

◆ 슬레이브 서버에서 로그인 시도

```
# su love
su: 알 수 없는 id: love
```

마스터 서버에서 생성한 계정으로 로그인을 시도하였으나 로그인이 되지 않는다.

```
# ypmatch love passwd
love:HsEJ26SqnoGJs:500:1::/work:/bin/sh
```

NIS passwd 맵 파일에는 love 계정 정보가 있음

시스템 구축

NIS를 활용한 시스템 파일 공유

◆ 마스터 서버에서 NIS 맵 파일을 가져옴

```
# ypinit -s server

installing the YP database will requ
Questions will all be asked at the b

Do you want this procedure to quit on non-fatal errors? [y/n: n]  n
OK, please remember to go back and redo manually whatever fails.  If you
don't, some part of the system (perhaps the yp itself) won't work.
The yp domain directory is /var/yp/JNET.nis.domain
There will be no further questions. The remainder of the procedure should take
a few minutes, to copy the data bases from server.
Transferring audit_user...
Transferring user_attr...
Transferring prof_attr...
Transferring exec_attr...
Transferring auth_attr...
Transferring ypserv.byname...
Transferring auto.home...
```

슬레이브 서버가 마스터 서버로부터 NIS 맵 파일을 받아 갱신한다.

```
# cat /etc/passwd | grep love
#
```

```
# su love
$ whoami
love
$
```

passwd 파일에 계정 정보가 없음에도 불구하고 로그인이 된다.

결론 및 의견

◆ 결론 및 해결방안

– 관리자 입장되어 만들어본 통합관리시스템은 파일 공유 면에서는 만족하였으나 이외에는 부가기능이 부족하다고 생각했다. 그래서 r 명령어를 통한 간편한 로그인 등의 기능과 방화벽 IPfilter를 통한 방화벽 기능을 추가했다. 시스템을 계속 보완해서 에러를 줄이는 방향으로 갈 것이다.

◆ 졸작을 수행하면서

– 관리자 한 명이 많은 pc를 관리하는 것이 번거롭고 시간이 많이 필요하기 때문에, 보다 더 효율적으로 관리할 방법을 찾기 위해 이 주제로 정하게 되었다.

또한 NIS와 NFS는 이미 구축이 되어 있는 시스템이다. 상위버전에는 보안상의 문제로 삭제된 내용도 있다. 하지만 이런 설정들을 직접 해봄으로써 어떤 점이 문제이고, 어떤 점이 효율적인지 알 수 있었고 유닉스의 기초를 다지는 좋은 계기가 되었다.