JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

기반 기술

J. Adv. Navig. Technol. 21(1): 78-83, Feb. 2017

부트 프로세스 변화에 따른 리눅스 서비스 관리 시스템 분석

Analysis of Linux Service Management System According to Boot Process Change

정 성 재[1] · 배 유 미[1] · 소 우 영[2*]
[1](주)엔버 기업부설연구소
[2]한남대학교 컴퓨터공학과

Sung-Jae Jung[1] · Yu-Mi Bae[1] · Wooyoung Soh[2*]
[1]Enber Co., Ltd., Seoul 05029, Korea
[2]Department of Computer Engineering, Hannam University, Daejeon 34430, Korea

[요 약]

유닉스의 영향을 받아 탄생한 리눅스는 커널을 제외하고는 대부분의 체계, 관련 프로그램 패키지, 명령어 등을 유닉스 계열에서 가져다가 사용하였다. 그러나 엔터프라이즈 리눅스의 최신 버전인 RHEL 7에서는 시스템 부팅 및 서비스 관리 프로그램으로 init 대신에 systemd를 채택하였다. systemd의 채택은 리눅스만의 독자적인 시스템 체제를 구축하는 것뿐만 아니라 특정 패키지 및 특정 명령어에 상당히 많은 기능을 부여했다는 점이 주목할 만하다. 유닉스의 아류로 인식되었던 리눅스가 클라우드 시대로 접어든 현 시점의 대세로 자리 잡게 되면서 독자적으로 다양한 변화를 시도하고 있는데, 이러한 변화의 시작이 systemd의 채택이라고 할 수 있다. 앞으로 리눅스가 유닉스의 굴레를 벗어나 어떠한 변화를 시도할지 주목할 필요가 있다.

[Abstract]

Linux was born under the influence of Unix except for the kernel, Linux was used to bring the Unix system and relevant program package, commands, etc. However, the latest version of Enterprise Linux, RHEL 7 was adopted systemd instead of init to boot program and management services. The adoption of systemd was building its own Linux system, the system only. Also noteworthy point is that given the considerable number of features specific to a particular package and instructions. As Linux has been recognized as a subtype of Unix itself as a catch all this time the turn of the tide in the cloud era and there on their own to try various changes, the start of the adoption of these changes can be called systemd. Linux is the future beyond the confines of Unix is noteworthy not attempt any changes.

Key word : Boot process, Linux, RHEL 7, Service management, Systemd.

https://doi.org/10.12673/jant.2017.21.1.78

Received 16 December 2016; **Revised** 2 February 2017
Accepted (Publication) 20 February 2017 (28 February 2017)

***Corresponding Author; Wooyoung Soh**

Tel: +82-42-629-7657
Email: wsoh@hnu.kr

Ⅰ. 서 론

유닉스(unix)의 영향을 받아 탄생한 리눅스는 커널(kernel)을 제외하고 대부분의 기본적인 체계, 관련 프로그램 패키지, 명령어 등을 유닉스 계열에서 가져다가 사용하였다[1]. 2000년대 초반에 레드햇(red hat)사가 유닉스 시장에 대항하기 위해 상용 버전의 레드햇 엔터프라이즈 리눅스(red hat enterprise linux, 이하 RHEL)를 출시하고, 지속적으로 업그레이드된 버전을 출시하였지만 유닉스 기반의 체제는 그대로 유지하였다[2]. 특히 시스템 부팅 체제는 Sys V 및 BSD 계열 유닉스에서 사용하던 init 프로세스를 그대로 적용해서 사용했다. 그러나 가장 최근에 출시된 RHEL 7 버전은 시스템 부팅 및 서비스 관리 프로그램으로 init 체제 대신에 systemd를 채택하면서 오랫동안 유지되어 왔던 부팅 체제에 변화의 바람이 불기 시작했다. systemd의 채택은 단순히 프로그램 버전 업그레이드에 따른 새로운 시스템 채택이 아닌 지금까지 리눅스 운영체제가 보여준 형태와는 많이 달라진 사례라고 볼 수 있다.

본 논문에서는 리눅스 부트 프로세스 체제인 init과 systemd에 대해 살펴보고 systemd 구조에 대해 알아본다. systemd 체제로 변화되면서 바뀐 리눅스 서비스 관리 체제에 대해서도 init 체제와 비교 분석하고 systemd만의 확장된 기능에 대해서도 살펴본다. 마지막으로 systemd의 채택에 따른 리눅스 운영체제에 대한 방향으로 결론을 맺는다.

Ⅱ. 리눅스 부트 프로세스의 변화

2-1 init 프로세스

리눅스는 초창기부터 리눅스는 초창기부터 SyS V 및 BSD 계열 유닉스에서 사용하던 init 프로세스를 사용해서 부팅을 해왔다. init 프로세스 체제는 부팅과 관련된 모든 작업을 init 프로세스에 위임하는 방식이다. 리눅스 시스템이 전원이 켜지고 BIOS(basic input output system) 점검을 마치면 커널이 로드되고, 커널은 우선 루트 파일 시스템(/)을 읽기 전용(read-only) 형태로 마운트하고 검사 후 이상이 없으면 쓰기 가능(read-write) 형태로 다시 마운트 한다. 이 후에 커널은 init 프로세스를 발생시키는데, 리눅스 부팅과 관련된 소프트웨어 구동은 init 프로세스에 위임된다[3,4]. init 프로세스는 리눅스시스템 내부의 최초의 프로세스로서 PID(process identity)가 1번이 할당되고, 그 이 후에 생성되는 프로세스는 모두 fork 방식으로 생성된다. 부팅 후에 생성되는 프로세스들은 전부 init 프로세스의 자식 프로세스 형태로 종속이 되면서 시스템의 종료 및 재부팅이 수월해진다. init 프로세스를 사용하는 CentOS 6 버전에서 pstree 명령으로 확인해보면 다음 그림 1과 같다. init 프로세스 체제는 실행 레벨(runlevel) 기반으로 여러 가지 부팅 모드로 나누어진다.

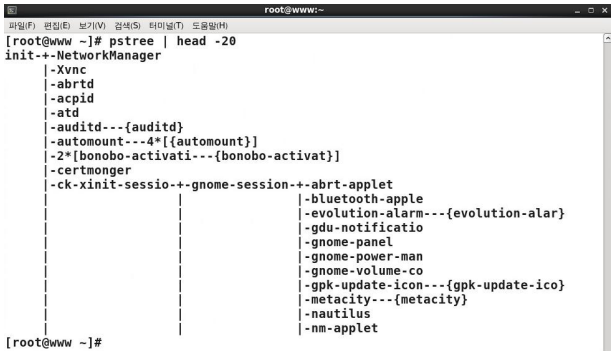

그림 1. CentOS 6의 init 프로세스
Fig. 1. The init process of CentOS 6.

실행 레벨이란 init 프로세스에 의해 수행되어질 내용이나 시스템 초기화 등을 정의해둔 여러 가지 모드를 뜻하는데, 리눅스는 총 8개의 레벨로 정의가 되어 있다. 일반적인 실행 레벨은 /etc/inittab에 정의되어 있는데, 0부터 6까지 총 7개의 레벨로 구성되어 있다. 마지막 8번째 레벨은 S 또는 s라는 문자로 표기되며, 기능적으로 보면 1번 레벨과 유사하지만 관련 프로세스나 데몬(daemon)을 최소화하여 유지 보수나 시스템 점검에 적합하다. 특히 시스템에 심각한 오류가 발생했을 때 주로 사용되어서 emergency 모드라 부르기도 한다. 일반적인 시스템의 부팅과 연관된 레벨은 3과 5가 있는데, 그래픽 환경으로 부팅하게 되면 실행레벨이 5이고 텍스트 환경으로 부팅하게 되면 실행레벨이 3이다. 각각의 실행 레벨별 주요 특징을 정리해보면 표 1과 같다.

표 1. 실행 레벨
Table 1. Run level.

Runlevel	Explanation
0	halt (Do NOT set initdefault to this)
1	Single user mode
2	Multiuser, without NFS (The same as 3, if you do not have networking)
3	Full mulituser mode
4	unused
5	X11
6	reboot (Do NOT set initdefault to this)

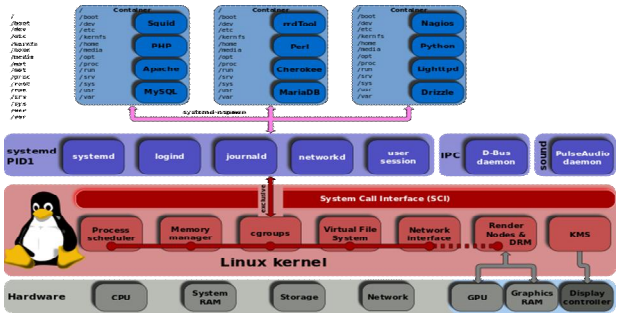

그림 2. 리눅스 커널과 systemd
Fig. 2. Linux kernel and systemd.

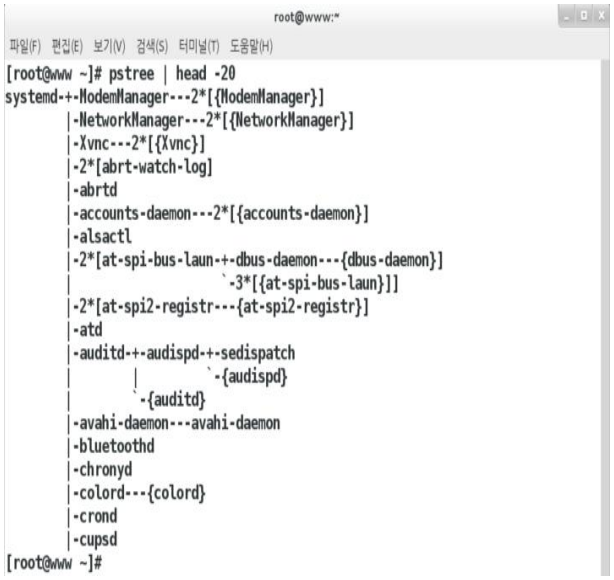

그림 3. CentOS 7의 systemd 프로세스
Fig. 3. The systemd process of CentOS 7.

2-2 systemd 프로세스

systemd는 System and Service Manager 약자인데, 시스템의 부팅 및 서비스 관리해주는 프로그램을 뜻하고 RHEL 7 버전부터 사용되고 있다[5,6]. RHEL 6 버전까지 사용되던 init 프로세스의 역할을 대행하면서 PID (process identity)도 동일하게 1번이 할당된다. systemd는 기존의 init의 역할 수행뿐만 아니라 병렬 처리를 통해 시스템 부팅 속도 증가, 데몬의 on-demand 시작 제공, 서비스의 의존성 자동 해결, 서비스를 시작을 위해 소켓 및 D-BUS 사용, Linux cgroups(control groups)를 사용한 프로세스 추적, 시스템 상태의 스냅샷(snapshot) 및 복원 기능 제공, 마운트 및 자동 마운트 지점 유지, 자체적인 로그 관리 등의 기능을 제공한다[7]. systemd를 사용하는 CentOS 7 버전에서 pstree 명령으로 확인해보면 그림 3과 같다.

Ⅲ. systemd의 구조

3-1 주요 구성 요소

systemd의 핵심은 유닛(unit)이라는 부르는 일종의 대상(object) 파일이고, 유닛은 service, target, socket, path 등과 같이 다양한 유형(type)을 가지고 있다. 유닛의 환경 설정 파일이 위치하는 디렉터리는 /etc/systemd/systemd이다. 이 디렉터리 내에는 '.wants'라는 디렉터리가 존재하는데, 이 디렉터리는 특정 유닛의 구동에 필요한 유닛들의 설정 파일을 심볼릭 링크(symbolic link)로 담고 있다.

1) 서비스(service)

시스템에서 제공하는 서비스, 즉 웹 서버이나 메일 서버 같은 데몬을 뜻한다. systemd 체제에서 서비스들은 init 체제에서 chkconfig, service 등 다양한 명령으로 제어된 것과 다르게 systemctl이라는 하나의 명령으로 제어된다. 서비스 관련 상태 정보는 'systemctl status 서비스명'으로 그림 4와 같이 확인이 가능하다.

2) 타겟(target)

타겟은 부팅 레벨, 특정 동기화 지점과 같이 유닛을 그룹화할 때 사용한다. 부팅과 관련된 주요 타겟을 정리해보면 다음의 표 2과 같다.

이전 버전의 init에서는 부팅과 관련된 런레벨을 0~6까지 숫자값으로 정의했는데, systemd 체제에서는 사용자의 편의를 도모하기 위해 runlevel***n***.target이라는 이름으로 심볼릭 링크를 제공하고 있다. 관련 파일은 /lib/systemd/system 디렉터리에서 그림 5와 같이 확인할 수 있다.

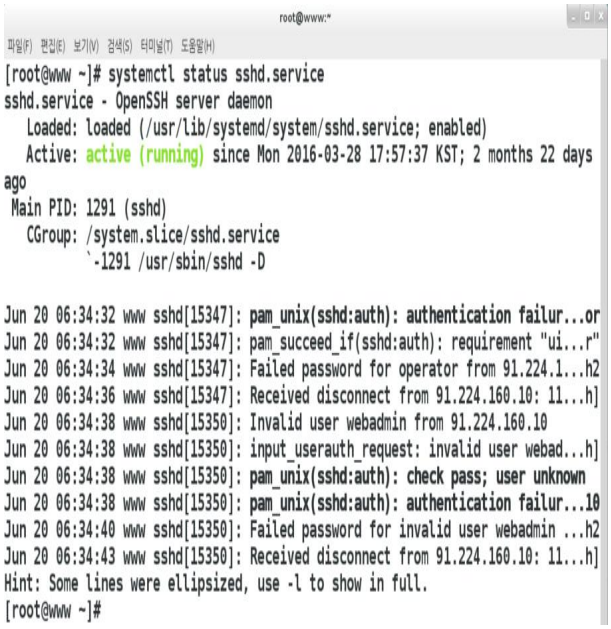

그림 4. 서비스의 상태 확인
Fig. 4. Confirm of Service status.

표 2. systemd의 주요 타겟
Table 2. The main systemd target.

Target	Purpose
graphical.target	System supports multiple users, graphical and text-based logins.
multi-user.target	System supports multiple users, text-based logins only.
rescue.target	sulogin prompt, basic system initialization completed.

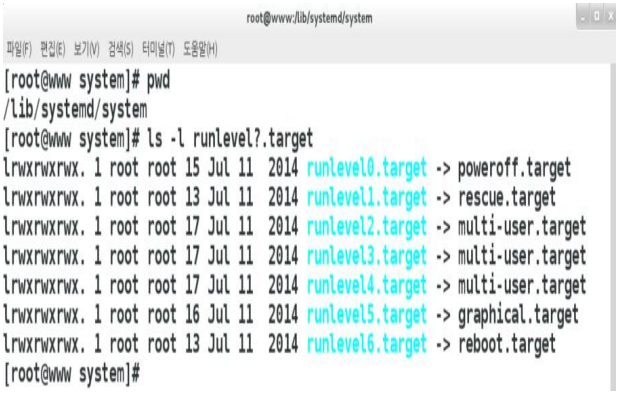

그림 5. 런레벨과 관련 있는 타겟
Fig. 5. Target files associated with Runlevel.

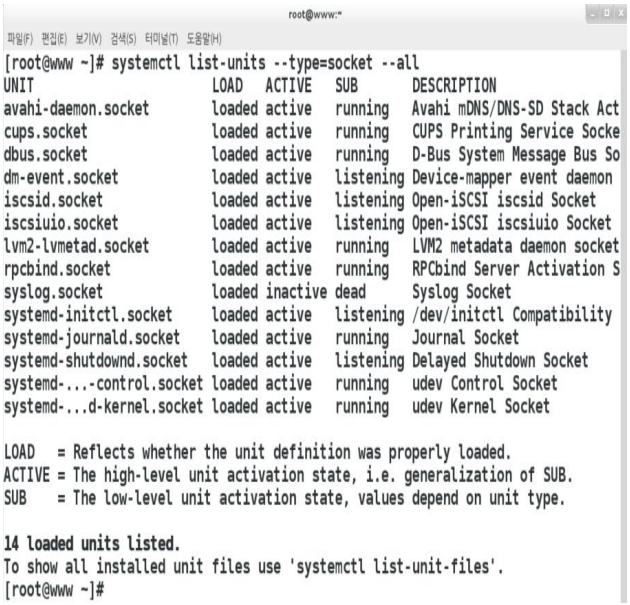

그림 6. 소켓 유닛의 목록
Fig. 6. List of active Socket Unit.

3) 소켓(socket)

소켓 유닛은 systemd에 의해 제어되면서 소켓 기반으로 동작하는 파일 시스템 FIFO, 네트워크 소켓, IPC 등이 해당된다. 관련 정보는 'man systemd.socket'에서 확인할 수 있고, 시스템에 동작중인 소켓 유닛 정보를 확인해보면 그림 6과 같다.

3-2 시스템 관리 체제

systemd가 시스템 부팅을 비롯하여 서비스를 관리해주는 역할을 수행하면서 모든 역할을 제어할 수 있는 명령어를 만들어냈는데, 이 명령이 systemctl이다. systemctl은 systemd 기반의 시스템 및 서비스 관리를 제어하는 명령으로 이전 버전의 init 체계에서 사용하던 명령어인 service, chkconfig, init 등의 기능을 합쳐놓은 것과 같은 역할을 수행한다. 따라서 systemd 체제에서의 리눅스 시스템 및 서비스 관리는 systemctl 명령의 적절한 사용에 있다.

Ⅳ. 리눅스 서비스 관리 체제 비교 분석

4-1 init과 systemd의 관련 명령 비교

1) 서비스의 부팅 시 활성화 및 비활성화

init 체제에서는 부팅 시 특정 서비스의 활성화 및 비활성화는 chkconfig 명령을 사용하였으나 systemd는 systemctl 명령으로 제어한다. systemctl 명령은 추가로 is-enabled라는 인자값(argument)을 이용하여 부팅 시에 특정 서비스가 구동되지는 여부를 확인할 수 있는 기능을 제공하고, mask, unmask라는 인자값을 이용하여 특정 서비스의 활성화를 막거나 해제할 수 있는 기능을 추가로 제공하고 있다. init 기반과 systemd 기반에서 특정 서비스를 부팅 시 활성화 및 비활성화하는 방법을 비교해보면 표 3와 같다.

2) 동작 중인 서비스의 제어

동작 중인 서비스의 제어를 init 체제에서는 service 명령을 사용하였으나 systemd는 systemctl 명령으로 제어한다. systemctl 명령은 동작 중인 서비스 제어 이외에도 is-active라는 인자값을 이용하여 서비스가 활성화되어 있는지 여부를 조회할 수 있는 기능을 제공하고 있다. init 기반과 systemd 기반에서 동작 중인 특정 서비스를 활성화 및 비활성화하는 방법을 비교해보면 표 4과 같다.

4-2 systemctl의 추가적인 기능

1) 시스템의 종료와 재부팅

init 체제에서는 시스템의 종료와 재부팅을 런레벨(Runlevel) 변경이라고 하는데, 이 때 사용하는 명령이 init, shutdown, halt, reboot, poweroff 등이 있다. systemd 체제에서도 이러한 전통적인 명령어 사용이 가능하지만, systemctl 명령만으로 poweroff, rescue, reboot, halt, emergency 등의 인자값을 이용해서 시스템의 종료와 재부팅이 가능하다. systemd 기반에서 런레벨을 변경하는 방법은 표 5와 같다.

표 3. 부팅 시 서비스 활성화 및 비활성화 비교
Table 3. Configure a service to start/stop at boot time.

init
chkconfig sshd on # chkconfig sshd off
systemd
systemctl enable sshd.service # systemctl disable sshd.service

표 4. 동작중인 서비스의 제어
Table 4. Start/stop a service on a running system.

init
service sshd start # service sshd stop
systemd
systemctl start sshd.service # systemctl stop sshd.service

표 5. 런레벨 변경 예
Table 5. Examples of Runlevel change.

```
# systemctl rescue
# systemctl poweroff
# systemctl reboot
```

표 6. 부팅 시 적용되는 런레벨 변경 예
Table 6. Examples of selecting a boot target.

```
# systemctl get-default
# systemctl set-default multi-user.target
# systemctl isolate runlevel3.target
```

표 7. 소켓 및 의존성 확인 예
Table 7. Examples of unit information.

```
# systemctl list-units
# systemctl list-unit-files
# systemctl list-sockets
# systemctl list-dependencies sshd.service
```

2) 부팅 시 적용되는 런레벨의 변경

init 체제에서는 부팅 시의 런레벨 지정은 /etc/inittab라는 파일에 설정하였으나 systemd에서는 더 이상 이 파일을 사용하지 않고, systemctl이라는 명령을 사용한다. systemctl 명령에서 get-default, set-default, isolate 등의 인자값을 사용해서 런레벨의 확인 및 변경이 가능하고, /etc/systemd/system/default.target이라는 심볼릭 링크 파일이 생성되면서 부팅 시의 런레벨 값이 지정된다. systemd기반에서 부팅 시에 적용되는 런레별 값의 변경은 표 6과 같다.

3) 소켓 및 의존성 확인

systemctl 명령은 list-units, list-unit-files라는 인자값을 이용해서 systemd의 핵심이라고 할 수 있는 유닛 및 유닛 관련 정보를 제공하고 있다. 이외에도 list-sockets, list-dependencies라는 인자값을 사용해서 소켓과 관련된 유닛의 정보와 특정 타겟 또는 유닛의 의존성 관련 정보도 제공하고 있다. systemd 기반에서 소켓 및 관련 의존성을 확인하는 방법은 표 7과 같다.

4-3 systemd의 기능 확대

1) 시스템 부팅과 관련된 성능 분석 도구 제공

systemd 체제에서는 시스템 부팅과 관련된 성능을 분석해주는 도구인 systemd-analyze라는 명령을 제공하고 있다. 기본적으로 시스템 부팅에 소요된 시간을 출력해주고, blame, critical-chain과 같은 인자값을 이용하면 각 서비스별로 부팅에 소요된 시간과 각 유닛에 소요된 시간 정보를 트리 형태로 출력해준다. 또한 plot이라는 인자값을 사용하면 시스템 부팅에 소요된 시간을 SVG(scalable vector graphic) 이미지 파일로 생성해서 가시성을 높여준다. systemd 기반에서 부팅 관련 성능을 분석하는 방법은 표 8과 같다.

2) 자체적인 로그 관리

systemd 체제에서는 관련 로그를 systemd-journald가 자체적으로 생성하고 관리한다. systemd-journald는 로그인 관련 정보를 커널로부터 받고, 사용자 프로세스 관련 정보는 syslog로부터 받는다. 관련 정보는 메타 데이터로 /run/log/journal 디렉터리에 저장되는 데, 재부팅하면 관련 정보가 사라지므로 별도의 관리가 필요하다. 로그 관리는 journalctl이라는 명령을 사용해서 제어한다. systemd 관련 전체 로그, 특정 서비스 관련 로그, 특정 수준(priority)과 관련된 로그, 특정 장치에 대한 로그, 특정 기간에 대한 로그 등과 관련된 정보를 출력할 수 있다. journalctl 명령으로 로그를 관리하는 방법은 표 9와 같이 지정하면 된다.

3) 기타 관리 명령

systemd 체제에서는 systemctl 명령 이외에도 많은 명령어를 포함하고 있다. 사용자 세션(session)을 관리하는 loginctl, CGroup(control group)을 관리하는 systemd-cgls와 systemd-cgtop, 호스트명을 설정하는 hostnamectl, 시스템의 로케일(locale)을 설정하는 localectl, 시간대를 설정하는 timedatectl 등 시스템 관리를 위한 명령어를 해당 패키지에서 자체적으로 보유하고 있다.

표 8. systemd-analyze 사용 예
Table 8. Examples of systemd-analyze.

```
# systemd-analyze
# systemd-analyze blame
# systemd-analyze critical-chain
# systemd-analyze plot > systemd.svg
```

표 9. journalctl 사용 예
Table 9. Examples of journalctl.

```
# journalctl
# journalctl /sbin/sshd
# journalctl -p err
# journalctl /dev/sda
# journalctl --since=2016-5-13
# journalctl --since=2016-5-13 --until=2016-6-30
```

Ⅴ. 결 론

엔터프라이즈 리눅스인 RHEL 7에서 시스템 부트 체제를 init 대신에 systemd 체제로 바꾼 것은 크게 두 가지 관점에서 기존의 버전 업그레이드와 다르다고 볼 수 있다. 첫 번째는 유닉스 기반의 시스템 체제의 탈피하고 리눅스만의 독자적인 시스템 운영체제를 구축한 점이다. 리눅스는 대부분 유닉스 계열의 시스템 구조, 패키지, 명령어들을 가져다가 사용하는 추세이었다. 물론 SELinux와 같이 리눅스만의 고유한 보안 모듈도 있지만, 시스템에 적용하지 않고 사용이 가능하므로 리눅스에 아주 큰 변화를 주었다고는 볼 수 없다. 그러나 systemd는 시스템 구조의 핵심이고 유닉스에서 리눅스로 넘어온 것이 아닌 처음부터 리눅스를 기반으로 개발해서 적용했다는 점이다. 두 번째는 특정 패키지 및 특정 명령어에 상당히 많은 기능을 부여했다는 점이다. 공개 소프트웨어 대명사격인 리눅스 운영체제는 일반적으로 하나의 명령이나 패키지에 집중하기보다는 좋은 기능을 하는 다수의 명령을 채택하고 조합하여 사용하는 경향이 강하게 나타났다. 이러한 영향으로 인해 공개소프트웨어 개발자들도 특정한 기능이나 명령에 집중하고, 다른 기능들은 다른 프로그램을 사용하는 경향을 보여 왔다. systemd 패키지는 시스템 부팅과 서비스 관리 이외에도 자체적인 로그 관리, cgroups 사용 등 시스템 운영과 관련하여 상당히 많은 부분까지 관여하고 있고, 이러한 영향으로 핵심 명령어인 systemctl도 init 명령 체제에서 많은 명령어들이 하는 기능을 하나의 명령어로 처리할 수 있도록 만들어져있다.

유닉스의 아류로 인식되었던 리눅스가 클라우드 시대로 접어든 현 시점의 대세로 자리 잡게 되면서 리눅스 독자적으로 다양한 변화를 시도하고 있고, 이러한 변화의 시발점이 systemd라고 볼 수 있다. 앞으로 리눅스가 systemd 체제 이외에도 유닉스의 굴레를 벗어나 독자적으로 또 어떠한 새로운 변화를 시도할지 주목해야할 것으로 사료된다.

감사의 글

이 논문은 2106년도 한남대학교 학술연구조성비 지원에 의하여 연구되었음.

참고 문헌

[1] Y. M. Bae, and S. J. Jung, *To Conquer Linux Master Second Class*, Seoul, Korea: Booksholic publishing, 2014.

[2] Red Hat, Red Hat Enterprise Linux [Internet]. Available: https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux/.

[3] S. J. Jung, and Y. M. Bae, *To Conquer Linux Master First Class*, Seoul, Korea: Booksholic publishing, 2015.

[4] Canonical Ltd, Upstart [Internet]. Available: http://upstart.ubuntu.com/.

[5] Freedeskotop.org, systemd Systemd and Service Manager [Internet]. Available: https://www.freedesktop.org/wiki/Software/systemd/.

[6] Red Hat, Red Hat Enterprise Linux 7 [Internet]. Available: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/7.0_Release_Notes/.

[7] Wikipedia, systemd [Internet]. Available: https://en.wikipedia.org/wiki/Systemd/.

정 성 재 (Sung-Jae Jung)
1998년 2월 : 한남대학교 컴퓨터공학과 (공학사)
2003년 8월 : 한남대학교 컴퓨터공학과 (공학석사)
2011년 2월 : 한남대학교 컴퓨터공학과 (공학박사)
2005년 3월 ~ 2010년 2월: 한남대학교 국제IT교육센터 전임강사
2015년 11월 ~ 현재 : ㈜엔버 기업부설연구소 소장
※ 관심분야 : 리눅스, 정보보안, 시스템보안, 클라우드 컴퓨팅, 서버 가상화

배 유 미 (Yu-Mi Bae)
2005년 2월 : 한남대학교 컴퓨터멀티미디어과 (공학사)
2007년 8월 : 한남대학교 정보기술과 (공학석사)
2013년 2월 : 한남대학교 컴퓨터공학과 (공학박사)
2015년 11월 ~ 현재 : ㈜엔버 기업부설연구소 선임연구원
※ 관심분야 : 리눅스, 정보보안, 멀티미디어, 클라우드 컴퓨팅

소 우 영 (Wooyoung Soh)
1979년 2월 : 중앙대학교 전자계산학과 (이학사)
1981년 2월 : 서울대학교 전자계산학과 (이학석사)
1991년 2월 : 메릴랜드대학교 전자계산학과 (이학박사)
1991년 9월 ~ 현재 : 한남대학교 컴퓨터공학과 교수
※ 관심분야 : 정보보호, 신경회로망