
VERSI 1.0

MEI, 2023

# PRAKTIKUM SISTEM OPERASI

## Controlling Services and the Boot Process

TIM PENYUSUN:

MAHAR FAIQURAHMAN, S.KOM., M.T.

MUHAMMAD RIDHA AGAM

SYAHRUL PANGESTU

MADE WITH PRIDE BY: LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

# MODUL 5 SISTEM OPERASI – CONTROLLING SERVICES AND THE BOOT PROCESS

## CAPAIAN PEMBELAJARAN MATA KULIAH

- List system daemons and network services started by the systemd service and socket units.
- Control system daemons and network services, using systemctl.
- Describe the Red Hat Enterprise Linux boot process, set the default target used when booting, and boot a system to a non-default target.
- Log into a system and change the root password when the current root password has been lost.
- Manually repair file system configuration or corruption issues that stop the boot process.

## KEBUTUHAN HARDWARE & SOFTWARE

- Laptop/PC
- Virtual Machine (VMware, Virtual Box, VPC)
- Sistem Operasi CentOS, download image OVA (Wajib)

## MATERI PRAKTIKUM

- Identifying Automatically Started System Processes
- Controlling System Services
- Selecting the Boot Target

## MATERI PRAKTIKUM

### Identifying Automatically Started System Processes

Setelah menyelesaikan *section* ini, kita dapat menyebutkan *system daemon* dan *network services* yang dijalankan oleh *socket units* dan layanan **systemd**.

#### Introduction to systemd

*Daemon* **systemd** mengelola *startup* untuk Linux, termasuk layanan *startup* dan layanan *management* secara umum. *Startup* bertugas mengaktifkan sistem *resource*, *server daemons*, dan proses lain baik saat booting maupun saat sistem sedang berjalan.

*Daemon* merupakan proses yang sedang menunggu atau berjalan di *background* untuk menjalankan berbagai macam *task* atau tugas. Secara umum, *daemon* berjalan secara otomatis pada saat *boot time* dan terus berjalan hingga *shutdown* atau hingga di-*stop* secara manual. Umumnya nama pada kebanyakan program *daemon* diakhiri dengan huruf **d** contohnya **httpd**, **ftpd**, **dhcpd**.

Dalam pemahaman **systemd**, sebuah service sering kali mengacu pada satu atau lebih *daemon*. Terdapat istilah bernama **oneshoot**, **oneshot** adalah istilah yang digunakan dalam **systemd** untuk menggambarkan tipe layanan yang hanya dijalankan sekali pada saat dimulai atau dihentikan. **Oneshot** sendiri merujuk pada situasi ketika memulai atau menghentikan sebuah layanan yang hanya menghasilkan perubahan satu kali pada keadaan/state sistem. Artinya, tidak ada proses *daemon* yang tetap berjalan setelahnya. Dalam konteks **systemd**, **oneshot** mengacu pada operasi yang dilakukan sekali saja dan tidak berlanjut sebagai proses pada *background*.

Di dalam Red Hat Enterprise Linux, proses pertama yang berjalan dengan PID 1 adalah **systemd**. Beberapa fitur yang disediakan **systemd** yaitu:

- Kemampuan *parallelization* (memulai beberapa service secara bersamaan), yang mana dapat meningkatkan kecepatan *boot* dari sistem.
- Memulai *daemon* sesuai permintaan tanpa memerlukan service tambahan.
- Layanan *manajemen dependency* secara otomatis, yang mana dapat mengurangi *timeouts* yang panjang. Contohnya, sebuah layanan *network-dependent* tidak akan mulai berjalan hingga *network*/jaringan tersedia.
- Sebuah metode untuk melacak proses yang berkaitan secara bersama-sama dengan menggunakan Linux control groups.

**Describing Service Units**

**systemd** menggunakan *units* untuk mengelola tipe objek yang berbeda-beda. Berikut beberapa jenis *units* yang umum:

- *Units* service mempunyai sebuah ekstensi **.service** untuk mewakili *system services.* Tipe *units* ini digunakan untuk memulai akses daemon dengan intensitas yang sering, seperti contohnya *web server*.
- *Units socket* mempunyai sebuah ekstensi **.socket** untuk mewakili *inter-process communication* (IPC), *socket* yang harus diawasi **systemd.** Jika client menghubungkan ke *socket*, maka **systemd** akan memulai sebuah *daemon* dan meloloskan *connection* untuk client tersebut. Unit *socket* digunakan untuk menunda mulainya service pada *boot time* dan untuk mengurangi penggunaan service sesuai permintaan.
- *Units path* mempunyai sebuah ekstensi **.path** dan digunakan untuk menunda aktivasi service hingga terjadi perubahan pada sistem *file* tertentu.

*Command* **systemctl** digunakan untuk mengelola *units*. Contohnya, menampilkan tipe *unit* yang tersedia dengan dengan menggunakan *command* **systemctl -t help.**

**Listing Service Units**

Kita menggunakan *command* **systemctl** untuk melihat kondisi/*state* sistem saat ini. Contohnya *command* berikut menjabarkan semua *unit* service yang dimulai saat ini, untuk memberi *pagination* pada *output* gunakan **less.**

```
[root@localhost student]# systemctl list-units --type=service
  UNIT                      LOAD   ACTIVE SUB     DESCRIPTION
  abrt-ccpp.service         loaded active exited  Install ABRT coredump hook
  abrt-oops.service         loaded active running ABRT kernel log watcher
  abrt-xorg.service         loaded active running ABRT Xorg log watcher
  abrtd.service             loaded active running ABRT Automated Bug Reportin
  accounts-daemon.service   loaded active running Accounts Service
  alsa-state.service        loaded active running Manage Sound Card State (re
  atd.service               loaded active running Job spooling tools
  auditd.service            loaded active running Security Auditing Service
  avahi-daemon.service      loaded active running Avahi mDNS/DNS-SD Stack
  blk-availability.service  loaded active exited  Availability of block devic
```

*Output* diatas membatasi tipe unit yang dijabarkan hanya tipe *unit service* saja dengan menggunakan opsi **--type=service**. *Output* mempunyai beberapa kolom sebagai berikut:

Kolom pada *output command* **systemctl list-units**:

**UNIT**
Nama *unit service.*
**LOAD**
Melihat apakah **systemd** berhasil mengurai konfigurasi *unit* dengan benar dan memuat/*load unit* ke dalam memori atau belum.
**ACTIVE**
Status aktivasi *high-level* dari *unit*. Informasi ini menunjukan apakah *unit* telah dijalankan dengan sukses atau tidak.
**SUB**
Status aktivasi *low-level* dari unit. Informasi ini menunjukan informasi yang lebih rinci mengenai *unit.* informasi bermacam-macam berdasarkan jenis *unit*, status, dan bagaimana *unit* dijalankan.
**DESCRIPTION**
Deskripsi singkat dari *unit.*

Secara *default*, *command* **systemctl list-units --type service** menjabarkan *unit service* dengan status aktivasi **active.** Opsi **--all** menjabarkan semua *unit service* tanpa memperhatikan status aktivasi. Gunakan opsi **--state=** untuk membatasi *output* berdasarkan nilai dari **LOAD, ACTIVE**, atau **SUB**.

```
[root@localhost student]# systemctl list-units --type=service --all
  UNIT                     LOAD      ACTIVE   SUB     DESCRIPTION
  abrt-ccpp.service        loaded    active   exited  Install ABRT coredump h
  abrt-oops.service        loaded    active   running ABRT kernel log watcher
  abrt-vmcore.service      loaded    inactive dead    Harvest vmcores for ABR
  abrt-xorg.service        loaded    active   running ABRT Xorg log watcher
  abrtd.service            loaded    active   running ABRT Automated Bug Repo
  accounts-daemon.service  loaded    active   running Accounts Service
  alsa-restore.service     loaded    inactive dead    Save/Restore Sound Card
  alsa-state.service       loaded    active   running Manage Sound Card State
```

*Command* **systemctl** tanpa argumen apapun menjabarkan semua unit yang ter-*load* dan aktif.

```
[root@localhost student]# systemctl
  UNIT                      LOAD   ACTIVE SUB       DESCRIPTION
  proc-sys-fs-binfmt_misc.automount loaded active waiting   Arbitrary Executable
  sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device loaded active plugged
  sys-devices-pci0000:00-0000:00:05.0-sound-card0.device loaded active plugged
  sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-s
  sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-s
  sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda.d
  sys-devices-platform-serial8250-tty-ttyS0.device loaded active plugged   /sys/
  sys-devices-platform-serial8250-tty-ttyS1.device loaded active plugged   /sys/
  sys-devices-platform-serial8250-tty-ttyS2.device loaded active plugged   /sys/
```

*Command* **systemctl list-units** menampilkan *unit* yang coba diurai dan di-*load* oleh layanan **systemd** ke dalam memori, perlu diketahui bahwa **list-units** bukan menampilkan *service* yang di-*install*, melainkan *service* yang tidak diaktifkan. untuk melihat semua *status file unit* yang di-*instal*, gunakan perintah **systemctl list-unit-files** seperti berikut:

```
[root@localhost student]# systemctl list-unit-files --type=service
UNIT FILE                                     STATE
abrt-ccpp.service                             enabled
abrt-oops.service                             enabled
abrt-pstoreoops.service                       disabled
abrt-vmcore.service                           enabled
abrt-xorg.service                             enabled
abrtd.service                                 enabled
accounts-daemon.service                       enabled
alsa-restore.service                          static
alsa-state.service                            static
anaconda-direct.service                       static
anaconda-nm-config.service                    static
anaconda-noshell.service                      static
```

Di dalam *output command* **systemctl list-units-files,** keterangan yang ada untuk bidang **STATE** adalah **enable**, **disable**, **static**, dan **masked.**

**Viewing Service States**

Melihat status spesifik *unit* dengan **systemctl status name.type**. Jika tipe *unit* tidak tersedia, **systemctl** akan menunjukan status *unit service* tidak ada.

```
[root@localhost student]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor prese
t: enabled)
   Active: active (running) since Sun 2023-04-02 08:52:13 EDT; 13h ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 984 (sshd)
    Tasks: 1
   CGroup: /system.slice/sshd.service
           └─984 /usr/sbin/sshd -D

Apr 02 08:52:13 localhost.localdomain systemd[1]: Starting OpenSSH server d...
Apr 02 08:52:13 localhost.localdomain sshd[984]: Server listening on 0.0.0....
Apr 02 08:52:13 localhost.localdomain sshd[984]: Server listening on :: por...
Apr 02 08:52:13 localhost.localdomain systemd[1]: Started OpenSSH server da...
Hint: Some lines were ellipsized, use -l to show in full.
```

*Command* ini menampilkan status *service* saat ini. Arti dari masing-masing *field* adalah:

**Service Unit Information**

| Field | Description |
|---|---|
| Loaded | Menunjukan apakah *unit service* ter-*load* (termuat) dalam memori. |
| Active | Menunjukan apakah *unit service* berjalan dan jika iya, berapa lama telah berjalan. |
| Main PID | Proses ID pertama dari *service,* termasuk nama *command*. |
| Status | Informasi tambahan mengenai *service.* |

Beberapa kata kunci yang menunjukan status dari *service* dapat ditemukan pada *output* status:

**Service States in the Output of systemctl**

| Keyword | Description |
|---|---|
| loaded | Konfigurasi file unit yang telah diproses. |
| active(running) | Berjalan dengan satu atau beberapa proses yang berkelanjutan. |
| active(exited) | Berhasil menyelesaikan sebuah *one-time configuration*. |
| active(waiting) | Berjalan namun menunggu sebuah *event.* |
| inactive | Tidak berjalan. |
| enabled | Telah dimulai saat *boot*. |
| disabled | Tidak disetel untuk mulai saat *boot*. |
| static | Tidak dapat diaktifkan, namun dapat dijalankan dengan mengaktifkan *unit* secara otomatis. |

**Verify the Status of a Service**

*Command* **systemctl** menyediakan metode untuk memverifikasi status secara spesifik dari sebuah *service*. Contohnya seperti *command* berikut untuk memverifikasi bahwa *unit service* saat ini berjalan (*active*):

```
[root@localhost student]# systemctl is-active sshd.service
active
```

*Command* mengembalikan status dari *unit service,* yang mana biasanya **active** atau **inactive.**

Jalankan *command* berikut ini untuk memverifikasi sebuah *unit service* mana yang teraktivasi untuk dimulai secara otomatis ketika sistem *booting*:

```
[root@localhost student]# systemctl is-enabled sshd.service
enabled
```

*Command* mengembalikan unit service mana yang sudah teraktivasi untuk berjalan pada booting, yang mana ini biasanya **enabled**, atau **disabled.**

Untuk Memverifikasi apakah unit gagal dalam *startup*, jalankan *command* berikut:

```
[root@localhost student]# systemctl is-failed sshd.service
active
```

Command mengembalikan nilai antara *active* jika berjalan dengan baik dan *failed* jika mengalami *error* ketika *startup*. Di dalam kasus ini, unit terhenti akan mengembalikan nilai **unknown** atau **inactive.**

Untuk menjabarkan *unit* yang gagal, jalankan *command* **systemctl --failed --type=service.**

```
[root@localhost student]# systemctl --failed --type=service
  UNIT          LOAD   ACTIVE SUB    DESCRIPTION
● kdump.service loaded failed failed Crash recovery kernel arming

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

1 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

## Controlling System Services

Setelah menyelesaikan bab ini, kita harus dapat mengontrol *system daemon* dan *network services* menggunakan **systemctl**.

### Starting and Stopping Services

*Services* butuh untuk di-*stop* atau di-*start* dengan beberapa macam alasan: seperti ketika *service* butuh di-*update,* konfigurasi *file* mungkin butuh diubah, atau mungkin *service* butuh di-*uninstall,* atau mungkin seorang *administrator* memulai *service* secara manual karena jarang menggunakan *service* tersebut.

Untuk memulai sebuah service, langkah pertama yang harus dilakukan adalah mengecek apakah *service* sudah berjalan atau belum dengan menggunakan **systemctl status**. Selanjutnya gunakan *command* **systemctl start** sebagai user **root** (menggunakan **sudo**). Berikut contoh cara memulai **sshd.service**:

```
[root@localhost student]# systemctl start sshd.service
```

Mengingat service **systemd** mencari file **.service** untuk mengelola *service* di dalam *command* tanpa menyebutkan tipe *service* dan nama *service*. Maka *command* diatas dapat dijalankan sebagai berikut:

```
[root@localhost student]# systemctl start sshd
```

Untuk menghentikan *service* yang saat ini berjalan, gunakan argumen **stop** dengan *command* **systemctl**. berikut adalah contoh untuk meng-*stop service* **sshd.service**:

```
[root@localhost student]# systemctl stop sshd.service
[root@localhost student]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor prese
t: enabled)
   Active: inactive (dead) since Sun 2023-04-02 22:50:29 EDT; 7s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 31567 ExecStart=/usr/sbin/sshd -D $OPTIONS (code=exited, status=0/S
UCCESS)
 Main PID: 31567 (code=exited, status=0/SUCCESS)

Apr 02 22:40:12 localhost.localdomain systemd[1]: Starting OpenSSH server d...
Apr 02 22:40:12 localhost.localdomain sshd[31567]: Server listening on 0.0....
Apr 02 22:40:12 localhost.localdomain sshd[31567]: Server listening on :: p...
Apr 02 22:40:12 localhost.localdomain systemd[1]: Started OpenSSH server da...
Apr 02 22:50:29 localhost.localdomain systemd[1]: Stopping OpenSSH server d...
Apr 02 22:50:29 localhost.localdomain systemd[1]: Stopped OpenSSH server da...
Hint: Some lines were ellipsized, use -l to show in full.
```

**Restarting and Reloading Services**

Ketika melakukan *restart* terhadap *service* yang sedang berjalan, *service* akan berhenti dan nantinya akan berjalan kembali. Pada saat *restart service*, *process* ID akan berubah dan mendapatkan sebuah proses id baru ketika *startup*. Untuk *restart service* yang sedang berjalan, gunakan argumen **restart** dengan *command* **systemctl**. Berikut adalah contoh untuk menunjukan bagaimana cara me-*restart* **sshd.service**:

```
[root@localhost student]# systemctl restart sshd.service
```

Beberapa *service* mempunyai kemampuan untuk me-*reload file* konfigurasinya tanpa perlu melakukan *restart*. Proses ini disebut *service reload*. Melakukan *reload* sebuah *service* tidak akan merubah proses ID yang tergabung dengan bebagaimacam proses *service* lainya. Untuk melakukan *reload* gunakan argumen **reload** dengan *command* **systemctl**. Berikut adalah contoh bagaimana cara *reload service* **sshd.service** setelah konfigurasi dirubah:

```
[root@localhost student]# systemctl reload sshd.service
```

Ketika kita tidak yakin apakah *service* mempunyai mempunyai fungsionalitas untuk me-*reload* perubahan *file* konfigurasi, gunakan argumen **reload-or-restart** dengan *command* **systemctl**. Karena kita harus berhusnudzon terhadap *service* ehe. *Command* tersebut me-*reload* perubahan pada konfigurasi jika fungsionalitas *reloading* tersedia. Jika tidak tersedia, maka *command* **restart** terhadap *service* tetap digunakan untuk mengimplementasi perubahan pada konfigurasi yang baru:

```
[root@localhost student]# systemctl reload-or-restart sshd.service
```

**Listing Unit Dependencies**

Beberapa *service* membutuhkan *service* lain berjalan terlebih dulu sebelum dijalankan, sehingga butuh membuat *dependencies* pada *service* yang lain. Karena mungkin saja *service* lain tidak berjalan ketika *booting*, tetapi berjalan ketika dibutuhkan saja. Pada kedua kasus tersebut, **systemd** dan **systemctl** akan menjalankan *service* sesuai kebutuhan, entah untuk *resolve dependencies* atau untuk menjalankan *service* yang jarang digunakan. Bingung kan? saya berikan contohnya, jika *service print* CUPS tidak aktif, tetapi ada file yang ditempatkan di *print spool directory*, maka sistem akan secara otomatis memulai *daemon* atau menjalankan perintah terkait CUPS untuk memproses permintaan *print* tersebut. Dengan demikian, sistem akan mengaktifkan layanan yang diperlukan untuk memenuhi permintaan print tersebut secara otomatis:

```
[root@localhost student]# systemctl stop cups.service
Warning: Stopping cups.service, but it can still be activated by:
  cups.path
  cups.socket
```

Jika ingin benar-benar memberhentikan *printing service* pada sebuah sistem, berhentikan ketiga unit tersebut. Menonaktifkan *service* dan menonaktifkan *dependencies*.

*Command* **systemctl list-dependencies UNIT** menampilkan hirarki dari *dependencies* untuk menjalankan *unit service*. Untuk membuat *reverse dependencies* (unit yang bergantung pada *unit* yang ditentukan), gunakan opsi **--reverse** dengan *command* **systemctl**.

```
[root@localhost student]# systemctl list-dependencies sshd.service
sshd.service
● ├─sshd-keygen.service
● ├─system.slice
● └─basic.target
●   ├─microcode.service
●   ├─rhel-dmesg.service
●   ├─selinux-policy-migrate-local-changes@targeted.service
●   ├─paths.target
●   ├─slices.target
●   │ ├─-.slice
●   │ └─system.slice
●   ├─sockets.target
●   │ ├─avahi-daemon.socket
```

**Masking and Unmasking Services**

Pada satu waktu, sebuah sistem mempunyai berbagai macam *service* berbeda yang terinstall dan *conflict* satu sama lain. Contohnya, terdapat beberapa method untuk mengelola *email server* (contohnya **postfix** dan **sendmail**). *Masking* sebuah *service* dapat mengurangi risiko kesalahan administrator menjalankan *service* yang dapat menyebabkan *conflict* satu sama lain secara tidak sengaja. *Masking* membuat sebuah *link* di dalam direktori konfigurasi pada **/dev/null** *file* yang mana dapat mencegah *service* untuk dimulai.

```
[root@localhost student]# systemctl mask sendmail.service
Created symlink from /etc/systemd/system/sendmail.service to /dev/null.
[root@localhost student]# systemctl list-unit-files --type=service
UNIT FILE                                   STATE
sendmail.service                            masked
```

Mencoba untuk menjalankan *unit service* yang di-*masking* akan gagal dengan *output* seperti berikut:

```
[root@localhost student]# systemctl start sendmail.service
Failed to start sendmail.service: Unit is masked.
```

Gunakan *command* **systemctl unmask** untuk unmask unit service:

```
[root@localhost student]# systemctl unmask sendmail
Removed symlink /etc/systemd/system/sendmail.service.
```

**Enabling Services to Start or Stop at Boot**

Menjalankan sebuah *service* pada sistem yang berjalan tidak menjamin *service* berjalan secara otomatis ketika sistem *reboot*. Mirip dengan menghentikan sebuah *service* pada sistem yang berjalan, tidak membuat *service* kembali berjalan kembali ketika sistem *reboot*. Membuat *link* pada direktori konfigurasi *systemd* membuat *service* dapat berjalan ketika *boot*. *Command* **systemctl** membuat dan menghapus *link* tersebut.

Untuk menjalankan *service* ketika *boot*, gunakan *command* **systemctl enable**.

```
[root@localhost student]# systemctl enable sshd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/sshd.service
to /usr/lib/systemd/system/sshd.service.
```

*Command* diatas membuat *link* simbolik dari *file unit service*, biasanya di dalam direktori **/usr/lib/systemd/system**, dimana lokasi *systemd* pada *disk* menyimpan *file*, yang mana di dalam direktori **/etc/systemd/system/TARGETNAME.target.wants**. Mengaktifkan sebuah *service* bukan berarti menjalankan *service* di dalam *session* saat ini. Untuk menjalankan *service* dan mengizinkannya untuk berjalan secara otomatis ketika *boot*, eksekusi kedua *command* **systemctl start** dan **systemctl enable**.

Untuk menonaktifkan *service* dari berjalan secara otomatis, gunakan *command* berikut, yang mana menghapus *link* simbolik yang dibuat ketika mengaktifkan *service*. Perlu dicatat bahwa menonaktifkan sebuah *service* tidak memberhentikan *service*-nya.

```
[root@localhost student]# systemctl disable sshd.service
Removed symlink /etc/systemd/system/multi-user.target.wants/sshd.service.
```

Untuk memastikan apakah *service* sudah aktif atau tidak, gunakan *command* **systemctl is-enabled**.

**Summary of systemctl Commands**

*Service* dapat dijalankan dan diberhentikan pada sistem yang berjalan dan diaktifkan atau dinonaktifkan untuk berjalan secara otomatis ketika *boot time*.

**Useful Service Management Commands**

| **Task** | **Command** |
|---|---|
| Melihat informasi secara detail mengenai status *unit*. | **systemctl status UNIT** |
| Menghentikan *service* pada sistem yang berjalan. | **systemctl stop UNIT** |
| Menjalankan *service* pada sistem yang berjalan. | **systemctl start UNIT** |
| Merestart sebuah *service* pada sistem yang berjalan. | **systemctl restart UNIT** |
| Me-*reload file* konfigurasi pada *service* yang berjalan. | **systemctl reload UNIT** |
| Menonaktifkan sepenuhnya *service* agar tidak dijalankan, baik secara manual maupun saat *boot*. | **systemctl mask UNIT** |
| Membuat sebuah *masked service* tersedia. | **systemctl unmask UNIT** |
| Mengkonfigurasi sebuah *service* agar berjalan ketika *boot time.* | **systemctl enable UNIT** |
| Menonaktifkan *service* berjalan saat *boot time.* | **systemctl disable UNIT** |
| Menjabarkan *unit* yang dibutuhkan dan diinginkan dengan *unit* yang spesifik | **systemctl list-dependencies UNIT** |

## Selecting the Boot Target

Setelah menyelesaikan bab ini, kita harus dapat mendeskripsikan *Red Hat Enterprise Linux boot process*, menyetel *default target* yang digunakan ketika *booting*, *boot* sebuah sistem menjadi sebuah *non-default target.*

**Describing the Red Hat Enterprise Linux 8 Boot Process**

Sistem komputer modern terdiri dari kombinasi kompleks perangkat keras dan perangkat lunak. Untuk menjalankan sistem dari kondisi mati hingga menjadi sistem yang berjalan dengan tampilan *login*, diperlukan kerjasama antara banyak komponen perangkat keras dan perangkat lunak. Berikut ini adalah gambaran secara umum mengenai tugas-tugas yang terlibat dalam proses booting sistem fisik **x86_64** dengan *Red Hat Enterprise Linux 8*. Daftar **x86_64** ini kurang lebih sama dengan virtual machine, namun pada virtual machine memiliki *hypervisor* untuk menangani beberapa perangkat keras yang spesifik melalui perangkat lunak.

- Ketika mesin hidup. *System firmware*, entah modern UEFI atau BIOS lama, menjalankan *Power On Self Test* (POST) dan mulai menginisialisasi beberapa *hardware*. Pengaturan konfigurasi sistem dilakukan melalui layar konfigurasi BIOS atau UEFI. Untuk mengakses layar tersebut, kita perlu menekan kombinasi tombol khusus, seperti **F2**, pada awal proses *booting*. Perlu diingat bahwa UEFI berbeda dengan UEFI *Champions League* ehee.

- *System firmware* mencari perangkat *bootable*, baik yang dikonfigurasi di *firmware boot* UEFI atau dengan mencari *Master Boot Record* (MBR) di semua *disk*, dalam urutan yang dikonfigurasi pada BIOS.

- *System firmware* membaca *boot loader* dari *disk* dan kemudian meneruskan kontrol sistem ke *boot loader*. Pada sistem *Red Hat Enterprise Linux 8*, *bootloadernya* adalah *GRand Unified Bootloader version 2* (GRUB2).
  Dikonfigurasi menggunakan *command* **grub2-install**, yang mana menginstall GRUB2 sebagai *boot loader* pada *disk.*

- GRUB2 memuat konfigurasinya dari *file* **/boot/grub2/grub.cgf** dan menampilkan sebuah menu dimana kita dapat memilih kernel mana untuk *boot*.
  Dikonfigurasi pada direktori **/etc/grub.d/**, *file* **/etc/default/grub**, dan *command* **grub2-mkconfig** untuk men-*generate* file **/boot/grub2/grub.cgf**.

- Setelah kita memilih kernel, atau timeout sudah berakhir, *boot loader* memuat kernel dan **initramfs** dari *disk* dan menempatkannya di dalam memori. **initramfs** adalah arsip yang berisi modul kernel untuk semua perangkat keras yang diperlukan saat *boot, initialization scripts*, dan lainya. *Pada Red Hat Enterprise Linux 8*, **initramfs** berisi seluruh sistem yang dapat digunakan dengan sendirinya.
  Dikonfigurasi dengan menggunakan direktori **/etc/dracut.conf.d/**, *Command* **dracut**, dan **lsinitrd** untuk menginspeksi *file* **initramfs**.

- *Boot loader* memberikan kendali kepada kernel, meneruskan opsi apa pun yang ditentukan pada *command line* kernel di *boot loader*, dan lokasi **initramfs** di memori.

Dikonfigurasi menggunakan direktori **/etc/grub.d/**, *file* **/etc/default/grub**, dan perintah **grub2-mkconfig** untuk menghasilkan *file* **/boot/grub2/grub.cfg**.

- Kernel menginisialisasi semua perangkat keras yang dapat ditemukan drivernya di **initramfs**, kemudian mengeksekusi **/sbin/init** dari **initramfs** sebagai PID 1. Pada *Red Hat Enterprise Linux 8*, **/sbin/init** adalah *link* ke **systemd**.
  Dikonfigurasi menggunakan parameter *command-line* kernel **init=**.

- *Instance* **systemd** dari **initramfs** mengeksekusi semua *unit* untuk target **initrd.target**. Termasuk *mounting root file system* pada *disk* di *directory* **/sysroot**.
  Dikonfigurasi menggunakan **/etc/fstab**.

- Kernel mengalihkan (*pivot*) *sistem file* **root** dari **initramfs** ke file sistem di **/sysroot**. **systemd** kemudian mengeksekusi ulang sendiri menggunakan salinan **systemd** yang diinstal pada *disk*.

- **Systemd** mencari target *default*, baik yang diteruskan dari *command-line* kernel atau dikonfigurasi pada sistem, lalu memulai (dan menghentikan) *unit* untuk mengikuti target konfigurasi tersebut, *solving dependencies* antara *unit* secara otomatis. intinya, target **systemd** adalah sekumpulan *unit* yang harus diaktifkan oleh sistem untuk meraih *state* yang diinginkan. Target ini biasanya memulai *login* berbasis teks atau tampilan *login*.
  Dikonfigurasi dengan menggunakan **/etc/systemd/system/default.target** dan **/etc/systemd/system/**.

**Rebooting dan Shutting Down**
Untuk mematikan atau *reboot* sebuah sistem yang berjalan dari *command-line*, kita dapat menggunakan *command* **systemctl**.

**systemctl poweroff** menghentikan semua *service* yang berjalan, *unmount* semua *file system* (atau me-*remount read-only* ketika tidak dapat di-*unmount*), dan selanjutnya mati.

**systemctl reboot** menghentikan semua *service* yang berjalan, *unmount* semua *file system*, dan me-*reboot* sistem.

Kita juga dapat menggunakan versi *command* yang lebih pendek seperti, **poweroff** dan **reboot**, yang mana sama dengan *link* simbolik *systemctl*-nya.

**Selecting a Systemd Target**
Target **systemd** adalah sekumpulan *unit* **systemd** yang harus dijalankan oleh sistem untuk mencapai *state* yang diinginkan. Tabel berikut menjabarkan target yang paling penting:

**Common Used Targets**

| Target | Purpose |
|---|---|

| | |
|---|---|
| **graphical.target** | Sistem support banyak *user*, *graphical* dan *text based login*. |
| **multi-user.target** | System *support* banyak *user*, hanya *text-based login*. |
| **rescue.target** | **sulogin** *prompt*, inisialisasi dasar sistem telah selesai. |
| **emergency.target** | **sulogin** *prompt*, **initramfs** *pivot* selesai, dan *system root ter-mount* pada **/** (**root**) sebagai *read only* |

Sebuah target dapat dipisahkan ke beberapa target lain. Contohnya, **graphical.target** mengandung **multiuser.target**, yang mana membuat bergantung pada **basic.target** dan lainya. kita dapat melihat *dependencies* dengan command berikut.

```
[student@localhost bin]$ systemctl list-dependencies graphical.target | grep t
arget
graphical.target
● └─multi-user.target
●   ├─basic.target
●   │ ├─selinux-policy-migrate-local-changes@targeted.service
●   │ ├─paths.target
●   │ ├─slices.target
●   │ ├─sockets.target
●   │ ├─sysinit.target
●   │ │ ├─cryptsetup.target
●   │ │ ├─local-fs.target
●   │ │ └─swap.target
●   │ └─timers.target
●   ├─getty.target
●   ├─nfs-client.target
●   │ └─remote-fs-pre.target
●   └─remote-fs.target
●     └─nfs-client.target
●       └─remote-fs-pre.target
```

Untuk menjabarkan target yang tersedia, gunakan *command* berikut.

```
[student@localhost bin]$ systemctl list-units --type=target --all
  UNIT                        LOAD      ACTIVE   SUB    DESCRIPTION
  basic.target                loaded    active   active Basic System
  cryptsetup.target           loaded    active   active Local Encrypted Volumes
  emergency.target            loaded    inactive dead   Emergency Mode
  final.target                loaded    inactive dead   Final Step
  getty-pre.target            loaded    active   active Login Prompts (Pre)
  getty.target                loaded    active   active Login Prompts
  graphical.target            loaded    active   active Graphical Interface
  initrd-fs.target            loaded    inactive dead   Initrd File Systems
  initrd-root-fs.target       loaded    inactive dead   Initrd Root File System
  initrd-switch-root.target   loaded    inactive dead   Switch Root
  initrd.target               loaded    inactive dead   Initrd Default Target
  local-fs-pre.target         loaded    active   active Local File Systems (Pre)
  local-fs.target             loaded    active   active Local File Systems
  multi-user.target           loaded    active   active Multi-User System
  network-online.target       loaded    active   active Network is Online
  network-pre.target          loaded    active   active Network (Pre)
```

**Selecting a Target at Runtime**

Pada sistem yang berjalan, *administrator* dapat merubah target yang berbeda dengan menggunakan *command* **isolate**.

```
[root@localhost student]# systemctl isolate multi-user.target
```

*Isolating target* memberhentikan semua *service* yang tidak dibutuhkan oleh targetnya (beserta dependensinya), dan memulai *service* yang dibutuhkan.

Tidak semua target dapat di-*isolate*. Kita hanya dapat mengisolasi target yang mempunyai **AllowIsolate=yes** yang diatur di dalam *unit* filenya. Contohnya kita dapat *isolate graphical target*, tetapi tidak *cryptsetup* targetnya.

```
[root@localhost student]# systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
```

**Setting a Default Target**

Ketika sistem berjalan, **systemd** mengaktifkan target **default.target**. Normalnya *default target* berada di **/etc/systemd/system/** yang merupakan *link* simbolik baik antara **graphical.target** atau **multi-user.target**. Daripada mengedit *link* simbolik secara manual, *command* **systemctl** menyediakan dua *subcommand* untuk mengelola *link*: *get-default* dan *set-default*.

```
[root@localhost student]# systemctl get-default
multi-user.target
[root@localhost student]# systemctl set-default graphical.target
Removed symlink /etc/systemd/system/default.target.
Created symlink from /etc/systemd/system/default.target to /usr/lib/systemd/sy
stem/graphical.target.
[root@localhost student]# systemctl get-default
graphical.target
```

**Selecting a Different Target at Boot Time**

Untuk memilih target berbeda pada *boot time*, tambahkan opsi **systemd.unit=target.target** untuk *command-line* kernel dari *boot loader*.

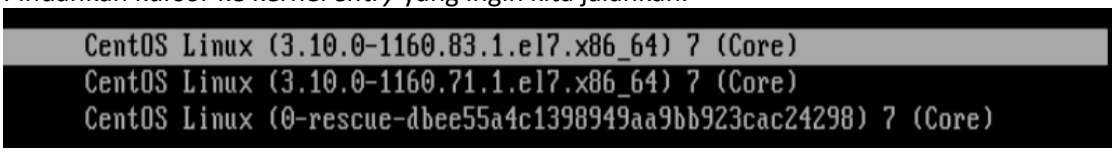
Contohnya, untuk mem-*boot system* ke dalam *rescue shell*, kita dapat mengubah *configuration system* dengan hampir tidak ada layanan yang berjalan. Tambahkan opsi berikut ke *command line* kernel dari *boot loader*.

```
systemd.unit=rescue.target
```

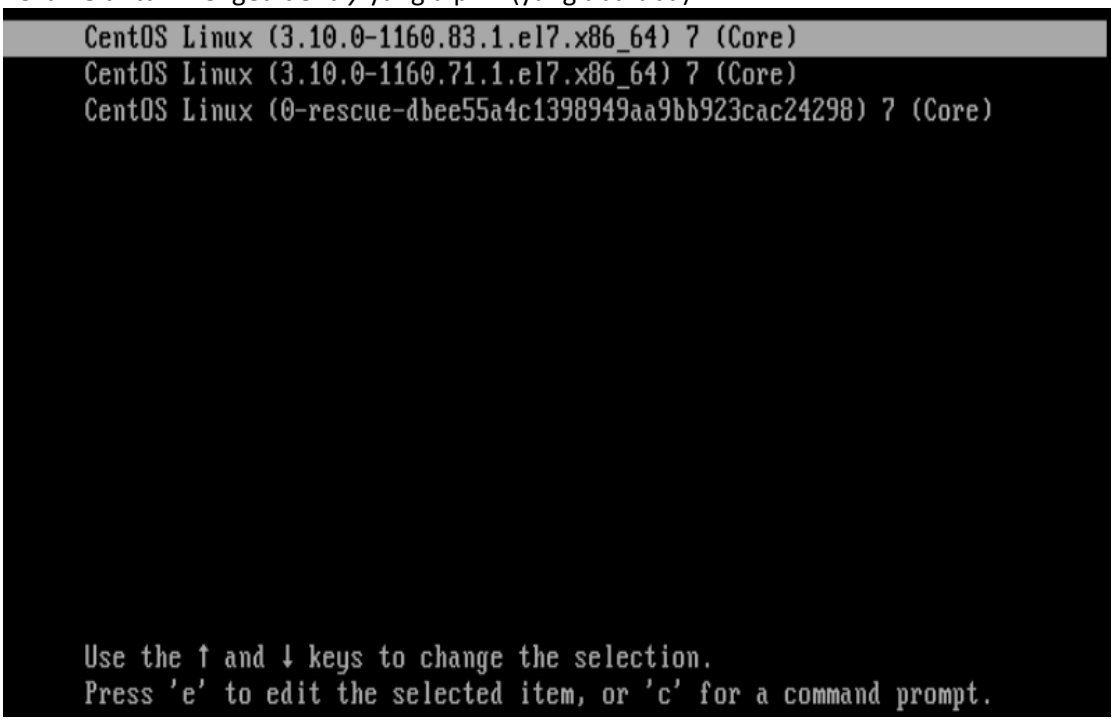
Konfigurasi yang dirubah hanya mempengaruhi sebuah *single boot*, membuat *tool* yang berguna untuk *troubleshooting* proses *boot*.

Untuk menggunakan metode memilih target berbeda, gunakan prosedur berikut:

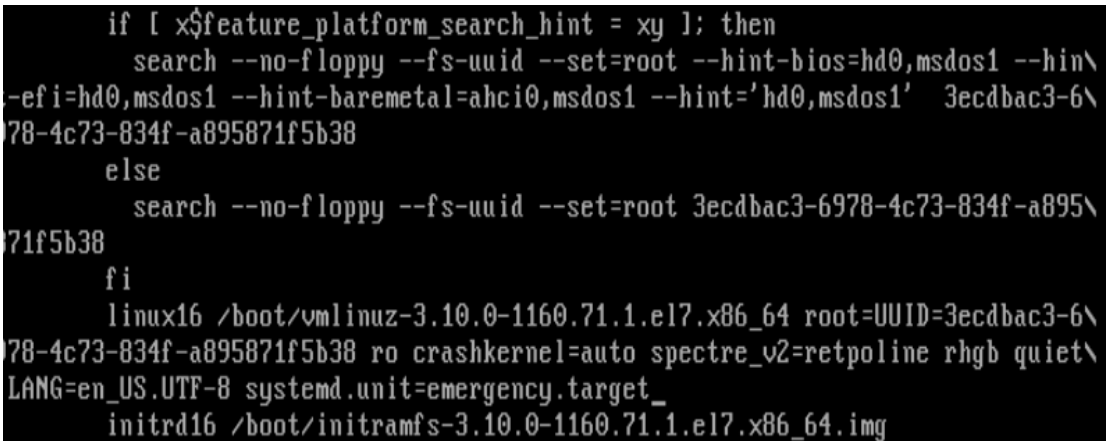
1. *Boot* atau *reboot system*
2. Interupsi *boot loader menu* dengan menekan key tertentu (kecuali **enter** yang mengindikasikan normal *boot*)
3. Pindahkan kursor ke kernel *entry* yang ingin kita jalankan.



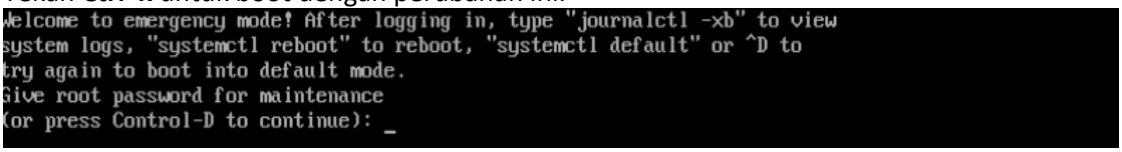

4. Tekan **e** untuk mengedit *entry* yang dipilih (yang abu-abu).




5. Pindah kursor ke baris bawah dan cari baris dengan awalan **linux**. Ini lah yang dimaksud kernel *command-line*.



6. Tambahkan **systemd.unit=target.target**. Contoh: **systemd.unit=emergency.target**.
7. Tekan **Ctrl+x** untuk boot dengan perubahan ini.



## LEMBAR KERJA MUDAH DAN BERFAEDAH

### KEGIATAN 1

**Identifying Automatically Started System Processes**

1. Gunakan CentOS seperti biasanya, masuk sebagai **student** atau **root** sesuai kebutuhan.
2. Tampilkan/jabarkan semua *unit service* yang terinstall
3. Tampilkan/jabarkan semua *unit socket*, yang aktif dan inaktif
4. Periksa status *service* **chronyd**. *Service* ini digunakan untuk **Network Time Protocol** (**NTP**)
    - Tampilkan status *service* **chronyd**. Sebutkan berapa proses ID *daemon* yang aktif.
    - Pastikan bahwa *daemon* yang ditampilkan/dijabarkan berjalan. Gunakan proses ID tersebut untuk melihat. (ex. ps -p PID)
5. Periksa status *service* **sshd**. *Service* ini digunakan untuk mengamankan komunikasi yang terenkripsi antar sistem.
    - Periksa apakah *service* **ssid** sudah diizinkan untuk memulai saat *system boot*.
    - Periksa apakah *service* **sshd** sudah aktif. (hanya menampilkan hasil aktif atau tidak saja).
    - Tampilkan status *service* **sshd.**
6. Tampilkan/jabarkan **enabled** atau **disabled** *states* dari semua *unit service*.

### KEGIATAN 2

**Controlling System Services**

1. Gunakan CentOS seperti biasanya, masuk sebagai **student** atau **root** sesuai kebutuhan.
2. Eksekusi *command* **systemctl restart** dan **systemctl reload** pada *service* **sshd**. Amati setiap step yang dilakukan.
    - Tampilkan status *service* **sshd**. Catat PID **sshd** *daemon*.
    - Restart *service* **sshd** dan lihat statusnya. PID *daemon* apakah berubah?
    - Reload *service* **sshd** dan lihat statusnya. PID *daemon* apakah berubah?
3. Cek status **chronyd** apakah berjalan?
4. Berhentikan *service* **chronyd** dan lihat statusnya
5. Pastikan *service* **chronyd** sudah ter-*enabled* untuk memulai *boot system*.
6. Reboot **workstation/centos**, selanjutnya lihat status *service* **chronyd**.
7. *Disable service* **chronyd** sehingga tidak dapat berjalan pada *system boot*, selanjutnya lihat status *service*nya.
8. *Reboot* **workstation/centos**, dan lihat status *service* **chronyd** nya.

### KEGIATAN 3

1. Masuk **centos**, buka terminal dan cek *default* target saat ini apa.
2. Ganti ke *multi-user* target secara manual tanpa *rebooting.*
3. Akses *text-base console*. Gunakan key **ctrl+alt+f1** dengan menggunakan tombol atau *entry menu*. *Login* sebagai **root**.
4. Konfigurasi **workstation/centos** yang dipakai menjadi otomatis *boot multi-user target*. dan selanjutnya *reboot* **workstation/centos** yang dipakai untuk memastikan. Ketika selesai, ubah *default* **systemd** *target* kembali menjadi **graphical** *target*.
    - Gunakan *command* **systemctl set-default** untuk mengatur *default target.*
    - *Reboot.*

- *Login* sebagai **root.**
- Atur target *default* **sistemd** kembali menjadi **graphical target**.

5. Pada step ini, kalian akan mempraktekan penggunaan *rescue mode* untuk *recover* sistem. Akses *boot loader* dengan melakukan *reboot* **workstation/centos** kembali. Dari menu *boot loader*, *boot* kedalam *rescue target*.
6. Pastikan sudah dengan *rescue mode*, dan **root** *file system* sudah pada *mode read/write*.
7. Tekan **ctrl+d** untuk melanjutkan proses *boot*.

## RUBRIK PENILAIAN

| Aspek Penilaian | Bobot Penilaian |
|---|---|
| Ketepatan menjawab semua kegiatan | 65% |
| Pemahaman setiap aspek materi yang dibahas | 20% |
| Quiz pertemuan materi minggu pertama | 15% |
| **Total** | **100%** |