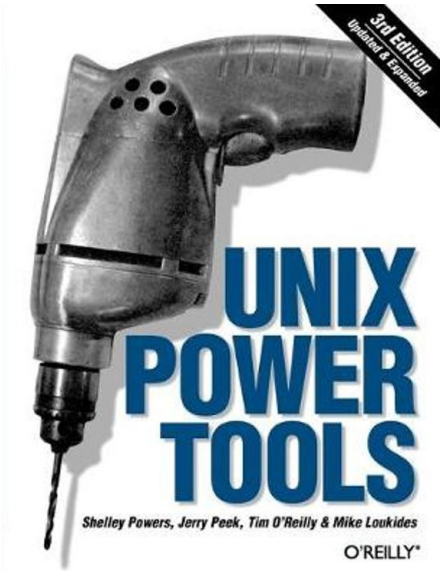
Kurs administrowania systemem Linux

Zajęcia nr 8: BSD Init i SystemD

Instytut Informatyki Uniwersytetu Wrocławskiego

26 kwietnia 2022

O'Reilly: Unix Power Tools

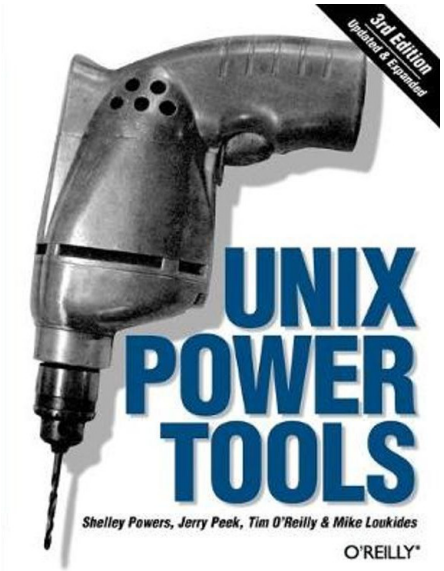


Unix Power Tools, 3rd Edition
by Jerry Peek, Shelley Powers, Tim O'Reilly, Mike Loukides
Released October 2002
Publisher(s): O'Reilly Media, Inc.
ISBN: 9780596003302

Jak zabezpieczyć ważny katalog przed przypadkowym skasowaniem całej zawartości?

```
$ rm *.bak
```

O'Reilly: Unix Power Tools

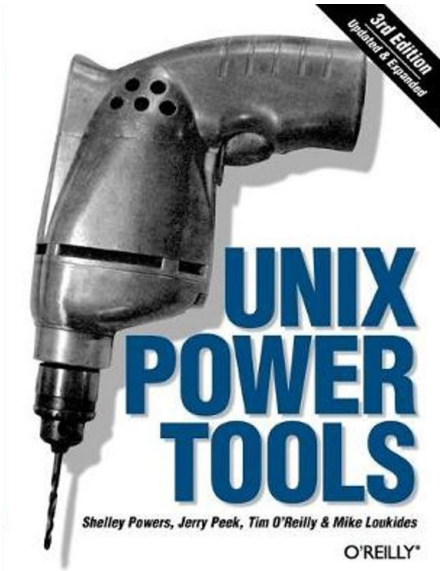


Unix Power Tools, 3rd Edition
by Jerry Peek, Shelley Powers, Tim O'Reilly, Mike Loukides
Released October 2002
Publisher(s): O'Reilly Media, Inc.
ISBN: 9780596003302

Jak zabezpieczyć ważny katalog przed przypadkowym skasowaniem całej zawartości?

```
$ rm * .bak
```

O'Reilly: Unix Power Tools



Unix Power Tools, 3rd Edition
by Jerry Peek, Shelley Powers, Tim O'Reilly, Mike Loukides
Released October 2002
Publisher(s): O'Reilly Media, Inc.
ISBN: 9780596003302

Jak zabezpieczyć ważny katalog przed przypadkowym skasowaniem całej zawartości?

```
$ rm * .bak
```

Należy w tym katalogu wykonać polecenie:

```
$ touch ./-i
```


I changed all my passwords to "incorrect".
So whenever I forget, it will tell me "Your password is incorrect."
memebingo.com

Init — proces o numerze 1

Cechy procesu init

- Najważniejszy demon w systemie.
- Jako jedyny proces jest uruchamiany bezpośrednio przez jądro.
- Jest pierwszym procesem uruchomionym w przestrzeni użytkownika.
- Jest przodkiem każdego procesu.
- Kończy działanie jako ostatni.

Zadania procesu init

- Rozruch systemu:
  - skonfigurowanie systemu po uruchomieniu (montowanie dysków, konfiguracja sieci itp.),
  - uruchomienie i nadzorowanie pracy wszystkich serwisów.
- Zamknięcie systemu:
  - dekonfiguracja systemu (np. odmontowanie dysków itp.),
  - zatrzymanie serwisów,
  - wyłączenie maszyny.
- Reparenting osieroconych procesów i sprzątanie po zakończonych procesach.

Klasyczny *init system*: BSD Init

Skrypty powłoki

- `/etc/rc` — uruchamiany przez `/sbin/init` podczas rozruchu systemu.
- `/etc/rc.shutdown` — uruchamiany przez `/bin/init` podczas zatrzymania systemu.
- `/etc/rc.d/`*`service`* — po jednym dla każdego serwisu.
- `/etc/defaults/rc.conf` — konfiguracja domyślna.
- `/etc/rc.conf` — konfiguracja lokalna systemu.

Skrypty /etc/rc i /etc/rc.shutdown

- Wczytują pliki `/etc/defaults/rc.conf` i `/etc/rc.conf`.
- Używają programu `rcorder(8)` do ustalenia kolejności uruchamiania/zatrzymywania serwisów (jedyny nie-skrypt w systemie!)
- W podanej kolejności wykonują skrypty `/etc/rc.d/`*`service`*.

Przykład: ssh

/etc/rc.d/ssh:

```
#!/bin/sh

# PROVIDE: sshd
# REQUIRE: LOGIN FILESYSTEMS
# KEYWORD: shutdown

. /etc/rc.subr

name="sshd"
desc="Secure Shell Daemon"
rcvar="sshd_enable"
command="/usr/sbin/${name}"
keygen_cmd="sshd_keygen"
start_precmd="sshd_precmd"
reload_precmd="sshd_configtest"
restart_precmd="sshd_configtest"
configtest_cmd="sshd_configtest"
pidfile="/var/run/${name}.pid"
extra_commands="configtest keygen reload"
...
load_rc_config $name
run_rc_command "$1"
```

/etc/defaults/rc.conf:

```
...
sshd_enable="NO"
sshd_program="/usr/sbin/sshd"
sshd_flags=""
...
```

/etc/rc.conf:

```
...
sshd_enable="YES"
...
```

Przykład: mój /etc/rc.conf (FreeBSD 13)

```
hostname="host"
keymap="pl.kbd"
ifconfig_re0="DHCP"
kld_list="i915kms.ko"
dumpdev="NO"
local_unbound_enable="NO"
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
sshd_enable="YES"
ntpdate_enable="YES"
powerd_enable="YES"
zfs_enable="YES"
sddm_enable="YES"
dbus_enable="YES"
hald_enable="YES"
clear_tmp_enable="YES"
geli_groups="storage"
geli_storage_flags="-k /etc/crypto.key -p"
geli_storage_devices=\
   "label/storage_disk1 label/storage_disk2 label/storage_disk3 gpt/zstorage_slog"
devfs_system_ruleset="localrules"
```

Cechy BSD Init

Zalety

- Demon `/sbin/init` uruchamia pojedynczy skrypt powłoki podczas uruchamiania/zatrzymania systemu i więcej się nie interesuje rozruchem/zatrzymaniem.
- Cały rozruch/zatrzymanie jest wykonywany *wyłącznie* przez zbiór skryptów powłoki.
- Konfigurowanie prostej instalacji ogranicza się do edycji pojedynczego pliku.

Wady

- Sekwencyjny (cały rozruch w pojedynczym wątku procesora).
- Powolny (wykonanie skryptów powłoki).
- Praktycznie brak nadzoru nad uruchomionymi serwisami.

Dygresja: metodologie programowania

Imperatywne

```
silnia = 1;
while (n > 0) {
   silnia *= n--;
}
```

Pułapki

- Niejasne działanie dla `n < 0`.
- Zmienia wartość `n`.

Deklaratywne

```
silnia n
   | n > 0  = n * silnia (n-1)
   | n == 0 = 1
   | n < 0  = undefined
```

Zalety

- Opis tego, *co* chcemy osiągnąć, a nie *jak* to zrobić.

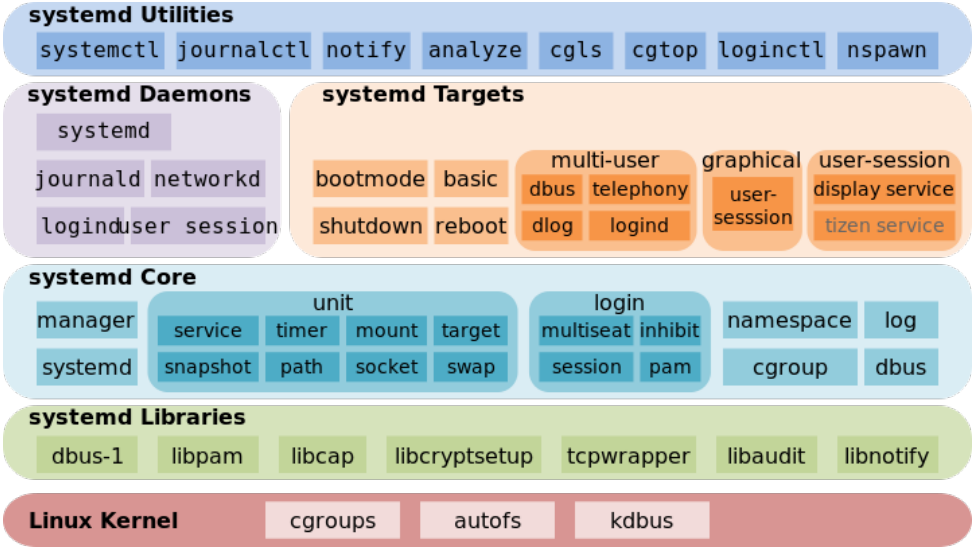
Programowania deklaratywne rozwinęło się wraz z językami funkcyjnymi (Lisp, Hope, SML, Miranda, Haskell). Sukcesy w głównym nurcie informatyki:

- Języki skryptowe: Python, Ruby.
- Języki zapytań w bazach danych: SQL.
- Systemy zarządzania konfiguracją: CFEngine, Puppet.
- Init systemy: OpenRC, SystemD.

SystemD

- Lennart Poettering, Kay Sievers, Harald Hoyer, Daniel Mack, Tom Gundersen, David Herrmann, 2010.
- Nowe podejście nawiązujące do idei *dependecy-based boot*.
- Składniki systemu są opisane za pomocą osobnych plików konfiguracyjnych.
- Brak skryptów rc: wszystkie czynności wykonuje `/sbin/init`.
- Jednostki SystemD opisują zasoby i zależności pomiędzy nimi.
- Katalogi zawierające jednostki: `/{etc,run,lib}/systemd/system/`, w podanej kolejności (np. plik w `etc` przesłania `lib`).
- Składnia jednostek: inspirowana przez pliki XDG `*.desktop` i pliki `*.ini` Microsoftu.

SystemD


systemd Utilities
systemctl
journalctl
notify
analyze
cgls
cgtop
loginctl
nspawn
systemd Daemons
systemd
journald
networkd
logind
user session
systemd Targets
bootmode
basic
shutdown
reboot
multi-user
dbus
telephony
dlog
logind
graphical
user-sesssion
user-session
display service
tizen service
systemd Core
manager
systemd
unit
service
timer
mount
target
snapshot
path
socket
swap
login
multiseat
inhibit
session
pam
namespace
log
cgroup
dbus
systemd Libraries
dbus-1
libpam
libcap
libcryptsetup
tcpwrapper
libaudit
libnotify
Linux Kernel
cgroups
autofs
kdbus

Składnia jednostki — przykład

```
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Jednostki systemd w porównaniu z SysV Init

`service`	serwis
`socket`	gniazdo; usługa podobna do `xinetd`
`device`	konfiguracja urządzeń
`mount`	punkt montowania; odpowiada pojedynczemu wierszowi `/etc/fstab`
`automount`	punkty automontowania
`swap`	partycja lub plik wymiany
`target`	cel konfiguracji; odpowiada *runlevelowi*
`path`	ścieżka w systemie do katalogu lub pliku
`timer`	odpowiada usłudze `cron`
`snapshot`	zapisany stan SystemD
`slice`	hierarchiczna grupa jednostek
`scope`	zewnętrznie utworzony proces

Polecenia SystemD

- `systemctl`, `journalctl`, `loginctl`, `machinectl`, `busctl`, `timedatectl`, `localectl`, `hostnamectl`
- `systemd-cgls`, `systemd-cgtop`, `systemd-nspawn`, `systemd-analyze`, `systemd-cat`, `systemd-detect-virt`, `systemd-delta`, `systemd-run`, `systemd-path`
- `systemd`, `systemd-notify`, `systemd-tty-ask-password-agent`, `systemd-ask-password`, `systemd-machine-id-setup`, `systemd-tmpfiles`, `systemd-inhibit`, `systemd-escape`, `systemd-stdio-bridge`

Praca z SystemD

Sprawdzanie zależności *targetów*

- `systemctl list-*`
- `systemctl list-dependencies graphical.target`

Zmiana *targetu*

- `systemctl isolate ...`

Sprawdzanie konfiguracji i stanu usług

- `systemctl status|show|cat ...`

Zmiana konfiguracji i stanu usług

- `systemctl enable|disable|start|stop|reload|restart ...`

Konfiguracja SystemD

Stale działające demony `/sbin/init` (link do `/lib/systemd/systemd`):

- systemowy,
- użytkownika (opcja `--user`).

Kolejność wyszukiwania plików jednostek systemowych:

- `/etc/systemd/system/`
- `/run/systemd/system/`
- `/lib/systemd/system/`

Kolejność wyszukiwania plików jednostek użytkownika:

- `$XDG_CONFIG_HOME/systemd/user/`
- `$HOME/.config/systemd/user/`
- `/etc/systemd/user/`
- `/run/systemd/user/`
- `$XDG_DATA_HOME/systemd/user/`
- `$HOME/.local/share/systemd/user/`
- `/usr/lib/systemd/user/`

Targety

- Odpowiedniki *runleveli* z SysV Init.
- Tak jak w OpenRC mają swoje nazwy i jest ich wiele (kilkadziesiąt).
- Link symboliczny `/lib/systemd/system/default.target`
- Kompatybilność z SysV Init: `runlevel{0..6}.target` — linki do, odpowiednio, `poweroff.target`, `rescue.target`, `multi-user.target`, `graphical.target` i `reboot.target`.
- Zależności *targetu* *X*: linki symboliczne w katalogu `/etc/systemd/system/`*X*`.target.wants/`.
- Dodawanie usuwanie zależności (linków): `systemctl [enable|disable]` *unit*
- Nazwa targetu jest podana w treści *unit*-u (`WantedBy`).
- Konfiguracja niestandardowych targetów: ręcznie za pomocą `ln -s`.

Jednostki SystemD

- *X*.service opisuje serwis (przeważnie demona). **Zastępuje** /etc/rc.d/*X* (BSD) i /etc/init.d/*X* (SysV).
- *X*.socket opisuje gniazdo (lokalne lub sieciowe). Powinien mu towarzyszyć *X*.service. Serwis jest uruchamiany miarę potrzeb, w razie połączenia klienta z gniazdem. **Zastępuje** inetd(8), xinetd(8).
- *X*.timer opisuje periodyczne uruchamianie usługi. Powinien mu towarzyszyć *X*.service lub inna jednostka uruchamiana periodycznie. **Zastępuje** cron(8), anacron(8).
- *X*.device — opis konfiguracji urządzenia. **Zastępuje** konfigurację udevd w /etc/udev/*.
- *X*.link i *X*.network — konfiguracja interfejsów sieciowych i sieci. **Zastępuje** interfaces(5).
- *X*.netdev — konfiguracja interfejsów wirtualnych. **Zastępuje** brctl(8), vconf(8) i in.

Jednostki SystemD, cd.

- *X*.`path` — monitoruje ustaloną ścieżkę w systemie plików. Powinien mu towarzyszyć *X*.`service` lub inna jednostka uruchamiana w razie zmiany ścieżki. Wykorzystuje `inotify(7)`. **Zastępuje** `inotify_watch(1)`.
- *X*.`mount` — opis punktu montażowego. **Zastępuje** jeden wiersz `fstab(5)`.
- *X*.`swap` — konfiguracja partycji wymiany. **Zastępuje** wpis w `fstab(5)`.
- *X*.`automount` — wyzwalacz montowania w miarę potrzeb. Powinien mu towarzyszyć opis punktu montażowego *X*.`mount`. **Zastępuje** `inotify_wait(1)`.
- *X*.`scope`, *X*.`slice` — grupowanie procesów z wykorzystaniem cgroups. **Zastępują** `libcgroup(3)`.
- *X*.`snaphot` — zapis stanu systemu.

Przykład: fstrim

fstrim.timer

```
[Unit]
Description=Discard unused blocks once a week
Documentation=man:fstrim

[Timer]
OnCalendar=weekly
AccuracySec=1h
Persistent=true

[Install]
WantedBy=timers.target
```

Przykład: fstrim, cd.

fstrim.service

```
[Unit]
Description=Discard unused blocks

[Service]
Type=oneshot
ExecStart=/sbin/fstrim -av
```

Przykład: eth0

eth0.link

```
[Match]
MACAddress=00:36:e4:aa:33:76

[Link]
MACAddressPolicy=random
Name=eth0
```

99-default.link

```
[Link]
NamePolicy=kernel database onboard slot path mac
MACAddressPolicy=persistent
```

Przykład: /tmp w ramdysku

tmp.service

```
[Unit]
Description=Temporary Directory
Documentation=man:hier(7)
Documentation=http://www.freedesktop.org/wiki/Software/...
ConditionPathIsSymbolicLink=!/tmp
DefaultDependencies=no
Conflicts=umount.target
Before=local-fs.target umount.target
After=swap.target

[Mount]
What=tmpfs
Where=/tmp
Type=tmpfs
Options=mode=1777,strictatime,nosuid,nodev

[Install]
WantedBy=local-fs.target
```

Przykład: swap w skompresowanym ramdysku

zramswap.service

```
[Unit]
Description=Configure zram device and use it for swap
After=systemd-modules-load.service

[Service]
RemainAfterExit=yes
ExecStartPre=/bin/sh -c '/sbin/zramctl -f --size 3072M > /run/zramswap'
ExecStart=/bin/sh -c 'test -b $(cat /run/zramswap) || exit 1; \
   /sbin/mkswap $(cat /run/zramswap); \
   /sbin/swapon -o discard $(cat /run/zramswap)'
ExecStop=/bin/sh -c 'test -b $(cat /run/zramswap) || exit 1; \
   /sbin/swapoff $(cat /run/zramswap)'
ExecStopPost=/bin/sh -c 'test -b $(cat /run/zramswap) || exit 1; \
   /sbin/zramctl -r $(cat /run/zramswap); cat /dev/null > /run/zramswap'

[Install]
WantedBy=sysinit.target
```

Krytyka SystemD

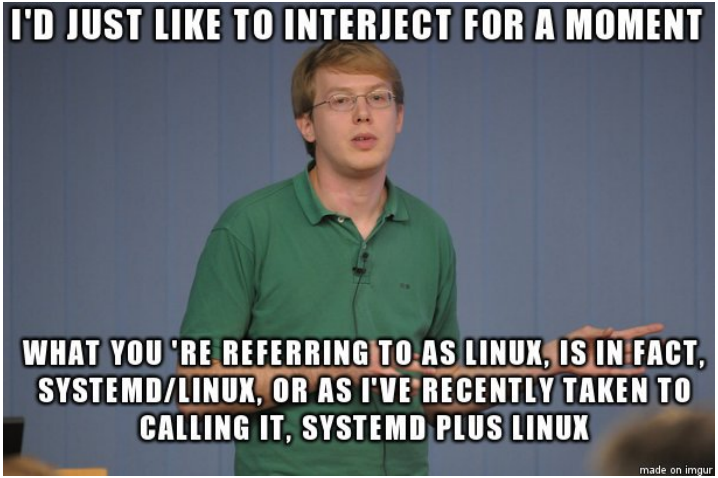
- Zbyt skomplikowany i rozbudowany.
- Zamienia Linuksa (filozofia klocków Lego) w monolit złożony z jednego wielkiego programu — nie pozostawia miejsca na wybór komponentów dystrybucji.
- Sprzeczny z filozofią Uniksa.
- Zamiast wolności wyboru fragmentów oprogramowania — dyktatura SystemD.
- Przerost funkcji kosztem modularności i jakości kodu.
- Po wprowadzeniu SystemD jako domyślnego systemu w Debianie powstał fork Devuan.
- W Debianie są spakietowane także stary SysV Init i OpenRC. Jednak nie ma wymogu, aby pakiety zawierały konfigurację dla tych systemów (dla SystemD też nie, ale on ma mechanizmy kompatybilności z SysV Init).

Lennart Poettering

- 15/10/1980, Guatemala
- obywatelstwo niemieckie
- Autor:
  - PulseAudio (2004)
  - Avahi (2005)
  - SystemD (2010)
- Krytykuje filozofię Uniksa.
- Krytykuje przenośność oprogramowania (*portability vs. innovation*), postuluje zerwanie kompatybilności Linuksa z POSIX i SUS.
- Krytykuje linuksowy sposób dystrybucji oprogramowania.

The classic Linux distribution scheme is frequently not what end users want, either. Many users are used to app markets like Android, Windows or iOS/Mac have. Markets are a platform that doesn't package, build or maintain software like distributions do, but simply allows users to quickly find and download the software they need, with the app vendor responsible for keeping the app updated, secured, and all that on the vendor's release cycle.

SystemD


I'D JUST LIKE TO INTERJECT FOR A MOMENT
WHAT YOU 'RE REFERRING TO AS LINUX, IS IN FACT, SYSTEMD/LINUX, OR AS I'VE RECENTLY TAKEN TO CALLING IT, SYSTEMD PLUS LINUX
made on imgur


I don't need bloatware! I'd like to follow the Unix way!
systemd

Dygresja: czego oprogramowanie nie powinno robić

Nieoczekiwane połączenia z Internetem

- `systemd-timesyncd.service` i serwery NTP. Trzeba wiedzieć, żeby wyłączyć: `timedatectl set-ntp false`
- Amarok i CDDB
- vim i słownik ortograficzny języka polskiego
- *Aplikacje nieprzeznaczone do łączenia się z Internetem nie powinny tego robić!*

Nieoczekiwane modyfikacje plików

- ebook-viewer modyfikuje przeglądane pliki EPUB
- *Aplikacje nieprzeznaczone do modyfikowania plików nie powinny tego robić!*

Nie inwestuję w instrumenty finansowe, których nie rozumiem.

— *Warren Buffet*

Nie używam programów, których nie rozumiem.

— *Ja*

System V Init

- Główny plik konfiguracyjny: `/etc/inittab`.
- Koncepcja *runlevels*.
- Biblioteka skryptów uruchamiających w `/etc/init.d/`.
- Katalog `/etc/rcS.d/` oraz dla każdego runlevelu katalog `/etc/rc`*i*`.d/`.
- Program `telinit(8)` do zmiany bieżącego runlevelu.

Katalog /etc/init.d/

- Zawiera skrypty konfigurujące usługi, po jednym dla każdej usługi.
- Każdy skrypt wymaga jednego parametru:
  `start`, `stop`, `reload`, `force-reload`, `restart`, `status`.
- Skrypt można uruchamiać samodzielnie, np.:
  `/etc/init.d/`*`usługa`* `stop`.
- Specjalne polecenie `service(8)`:

```
service script command
service --status-all
```

Runlevels

- Opisują stan konfiguracji systemu (które skrypty z `/etc/init.d` należy wykonać z parametrem `start` lub `stop`).
- Ponumerowane liczbami, zwykle 0–6.
- W Debianie było: 0 — *shutdown*, 1 — *single user mode*, 2 — *multiuser terminal*, 3, 4 — *user defined*, 5 — *multiuser graphical*, 6 — *reboot*.
- Dodatkowo: S — *single user mode*.
- Z każdym *runlevelem* i jest związany katalog `/etc/rc`i`.d/`. Dodatkowo katalog `/etc/rcS.d/`.
- W każdym katalogu dowiązania symboliczne do skryptów z `/etc/init.d`.

Runlevels, cd.

- Nazwa dowiązania `Snnusługa` odpowiada uruchomieniu skryptu `/etc/init.d/usługa` z parametrem `start`, a `Knnusługa` — stop.
- Liczby dwucyfrowe *nn* zapewniają odpowiednią kolejność.
- Skrypt `/etc/init.d/rc` wywołany z parametrem *i* wywołuje skrypt z `/etc/init.d` z parametrem `start` jeśli w nowym *runlevelu* jego dowiązanie zaczyna się na `S`, a w poprzednim *runlevelu* nie było `S` itd.
- Skrypty z `/etc/rcS.d/` wykonywane przed (1–5) lub po (0, 6) skryptach `/etc/rci.d/`.

Główny plik konfiguracyjny `/etc/inittab`

Zawiera m. in.:

- Numer domyślnego runlevelu, np. `id:5:initdefault:`.
- Konfigurację terminali, np. `1:2345:respawn:/sbin/agetty tty1`
- Uruchomienie skryptów runlevelu, np. `l5:wait:/etc/init.d/rc 5`

Zarządzanie dowiązaniami w katalogach /etc/rc*i*.d/

- Dawniej można było ręcznie użyć `ln -s`.
- Problem: dobranie właściwej kolejności skryptów.
- Rozwiązanie: *Dependency-based boot*.
- Nagłówki skryptów z `/etc/init.d`:

```
### BEGIN INIT INFO
# Provides:          boot_facility_1 [ boot_facility_2 ...]
# Required-Start:    boot_facility_1 [ boot_facility_2 ...]
# Required-Stop:     boot_facility_1 [ boot_facility_2 ...]
# Should-Start:      boot_facility_1 [ boot_facility_2 ...]
# Should-Stop:       boot_facility_1 [ boot_facility_2 ...]
# X-Start-Before:    boot_facility_1 [ boot_facility_2 ...]
# X-Stop-After:      boot_facility_1 [ boot_facility_2 ...]
# Default-Start:     run_level_1 [ run_level_2 ...]
# Default-Stop:      run_level_1 [ run_level_2 ...]
# X-Interactive:     true
# Short-Description: single_line_description
# Description:       multiline_description
### END INIT INFO
```

Linux Standard Base script dependency information

- Niskopoziomowe polecenie `insserv(8)` wyliczające właściwą kolejność skryptów na podstawie nagłówków.
- Wylicza pliki `/etc/init.d/.depend.{boot,start,stop}` w składni podobnej do `Makefile`.
- Konfiguracja: `/etc/insserv.conf` i `/etc/insserv.conf.d/*`.
- Konfiguracja katalogów `/etc/rc`*`i`*`.d/` — polecenie `update-rc.d(8)`:

```
update-rc.d [-n] [-f] usługa remove
update-rc.d [-n] usługa defaults
update-rc.d [-n] usługa disable|enable [ S|2|3|4|5 ]
update-rc.d [-n] usługa start|stop
```

- Ostatnia wersja odpowiada poleceniu `service`.

Krytyka SysV Init

Wady

- Jest powolny.
- Bardzo trudno zrównoleglić (procesory mają obecnie wiele rdzeni!).
- *Runlevels* są zbyt ograniczone (jest ich mało, nie są hierarchiczne).
- *LSB dependency boot headers* są zbyt ograniczone.
- Poza `respawn` w `inittab` brak możliwości nadzorowania i wznawiania upadłych serwisów.
- Brak jednolitego logowania zdarzeń (`rsyslog` nie wystarcza).
- Brak jednolitego zarządzania zasobami (w tym wykorzystania *cgroups* i *namespaces*).

Lepsze rozwiązania

- SystemD — uniwersalny, rozbudowany.
- OpenRC – prosty, ale z ograniczeniami.

Deklaratywne podejście do rozruchu systemu

Runlevel powinien opisywać konfigurację docelową, a nie sposób jej osiągnięcia.

Najstarsze implementacje

- SysV Init *LSB script dependency information* — oryginalnie Suse Linux, 2000. Zaadaptowane przez większość dystrybucji, w tym Debiana.
- *Gentoo modular init scripts* (Baselayout), 2001. Niezależny projekt w 2007 pod nazwą OpenRC (główny program przepisany w C). Od 2011 z powrotem jako część projektu Gentoo.

Linux Standard Base script dependency headers

- Usługi: serwisy oraz usługi systemowe, np. `local_fs`, `syslog` (zob. `/etc/insserv.conf`).
- Usługi (*facilities*) są wierzchołkami grafu zależności.
- Zależności opisane za pomocą nagłówków w skryptach:
  - `Provides`
  - `Required-Start`, `Required-Stop` (hard dependency)
  - `Should-Start`, `Should-Stop` (soft dependency)
  - `Default-Start`, `Default-Stop` (runlevel reverse dependency)
  - `X-Start-Before`, `X-Stop-After` (reverse dependency)
  - `X-Interactive` (needs user interaction, don't parallelize)
  - `Short-Description`, `Description`
- `insserv` *statycznie* wyznacza porządek uruchamiania/zatrzymywania (*Makefile-like* `/etc/init.d/.depend.{boot,start,stop}`).
- `startpar(8)` uruchamiany przez `init(8)` uruchamia równolegle skrypty z `/etc/init.d/` zgodnie z porządkiem zapisanym w plikach `.depend.*`.

OpenRC

- Opis serwisu podzielony na konfigurację `/etc/conf.d/`*`service`* i skrypt inicjalizujący `/etc/init.d/`*`service`*.
- Oba pliki: POSIX sh, interpretowane przez `/sbin/openrc-run`.
- Biblioteka funkcji pomocniczych: `libeinfo`.
- Na wzór Debiana polecenie `start-stop-daemon(8)`.
- Skrypt serwisu zawiera funkcję `depend` opisującą zależności.
- Argumenty skryptu: `start`, `stop`, `status`.
- Program `/sbin/openrc` uruchamiany przez `init(8)`.
- `openrc(8)` ma za zadanie osiągnąć podany runlevel, po czym zapisuje stan systemu i kończy działanie.
- Skompilowana konfiguracja jest zapisana w *cache'u*. `openrc(8)` używa `mtime` do śledzenia zmian w plikach źródłowych.
- Stan systemu można sprawdzać (`rc-status(8)`) i modyfikować (`rc-service(8)`).
- Wywołanie `openrc(8)` bez parametru przywraca konfigurację bieżącego runlevela.

Konfiguracja OpenRC

Zależności w funkcji depend() skryptu serwisu

- `need` (hard dependency)
- `want` (medium dependency)
- `use` (soft dependency)
- `before`, `after` (soft reverse dependency)
- `provide`
- `keyword` (platform-specific override)

Runlevels

- Runlevele mają nazwy, a nie numery. Może ich być dowolnie wiele. Są plikami w katalogu `/etc/runlevels/`.
- Runlevele zależą od serwisów.
- Zależności runleveli można zmieniać za pomocą `rc-update(8)`.
- *Stacked runlevels* — zależności runleveli od innych runleveli.
- Współbieżne uruchamianie jako opcja.

Monitorowanie serwisów w OpenRC

- `openrc(8)` kończy pracę po osiągnięciu podanego *runlevela*.
- Monitorowanie serwisów możliwe poprzez zastąpienie polecenia `start-stop-daemon(8)` wywołaniem demona nadzorującego: `runit` (reimplementacja `daemontools`) albo `s6`.
- Własna implementacja: `supervise-daemon(8)`. Uruchamiany program nie może się *fork*-ować.
- Można korzystać z *cgroups*, np.
  `rc-service` *`service`* `cgroup-cleanup`

OpenRC a SysV Init z *dependency boot headers*

- OpenRC zachowuje większość terminologii i zwyczajów SysV Init.
- Funkcja `depend` skryptu serwisu odpowiada *dependency boot headers*.
- Wygodniejszy opis runleveli, możliwość ich hierarchizacji.
- Dynamiczne wyznaczanie grafu zależności (w SysV Init statyczne za pomocą `insserv(8)`).
- Możliwość korzystania z *cgroups* w celu np. zatrzymywania grup zależnych serwisów.