System Daemons

Marek Kozłowski

Faculty of Mathematics and Information Sciences
Warsaw University of Technology

Outline of the Lecture

Outline of the lecture

Daemons

- *Disk And Execution MONitors*, informally called *services* or *servers*: ssh, name, web, mail, database servers, etc.
- The suffix 'd': `sshd`, `named`, `httpd`, `smtpd`, `mysqld`, etc. may denote a daemon.
- Background processes (no controlling terminal).
- Typically event-driven or client-driven; most of the time they idle waiting for events (for example: ACPI) or client requests.
- Write control messages to their own logs or send them to a *syslog*.

Daemons (2)

- Only one instance should execute at a time.
- Usually (adopted) children of the `init` process.
- Run on behalf of system users (system accounts).
- For security reasons daemons that require limited file access often started in *chroot jail* environments.
- May rely on other daemons.
- For the reasons listed above it should not be started 'by hand'. Dedicated controlling scripts are provided for this purpose.

Controlling script

- Written in shell (usually `sh`) or some shell extension.
- Explicit, transparent, human readable and customizable.
- Its structure reflects system/distro design principles (short and simple for KISS systems, comprehensive for enterprise ones).
- If some variables should be set up those are not defined directly in a script but in an additional configuration file for that script.

Daemon-Related Directories

- `/usr/sbin/` – daemon programs.
- `/etc/` – service configuration files or directories.
- `/etc/init.d/` or `/etc/rc.d/` – controlling scripts.
- `/etc/conf.d/` or `/etc/sysconfig/` – configuration files for controlling scripts.
- `/var/run/*.pid` – files that store PIDs of running daemons; created by start-up scripts.
- Some slight differences between systems may exist.

Daemon-Related Files – Example

- Under Gentoo Linux the following files are related to the SSH service:
  - `/usr/sbin/sshd` – daemon program file,
  - `/etc/ssh/sshd_config` – SSH configuration,
  - `/etc/init.d/sshd` – controlling script,
  - `/etc/conf.d/sshd` – controlling script configuration file,
  - `/var/run/sshd.pid` – PID of the running sshd process.

Running Daemons

- Controlling scripts usually accept the following commands as parameters (none displays all supported commands):
  - `start` ,
  - `stop` ,
  - `restart` – on upgrade or configuration change,
  - `reload` – (optional) on configuration change,
  - `status` – (optional) display status.
- Other commands may be defined depending on the system and the daemon.
- Additional messages like: `[ok]` or `[fail]` are displayed on command success/failure in numerous systems.

Starting the SSH daemon in Gentoo

```
# /etc/init.d/sshd start
```

Outline of the lecture

Runlevels

- SysV distinguishes the following system operational modes:
  - single-user (maintenance and repair),
  - multi-user with no network,
  - multi-user (server default),
  - multi-user with GUI (workstation default),
  - system shutdown,
  - system reboot.
- For each of those sysv defines a separate so-called *runlevel*.

Runlevels (2)

- Most Unices define seven runleves identified by numbers (0-6).
- Mode to number assignments may differ between systems.
- Some runlevels may be not implemented (reserved for future use) or merged (undistinguishable).
- Runlevels are fully separable and independent of each other.
- Runlevels are not executed sequentially, for example: it is possible to switch from 3 to 0 or 6 directly without entering intermediate ones.
- For each runlevel a separate set of processes to be launched by the `init` process is defined.

Standard Runlevels

- Most SysV systems agree on the following runlevel assignment:
  - 0 – system shutdown,
  - 1 or S – single-user (in some systems those are two distinct runlevels that slightly differ),
  - 3 – multi-user, default,
  - 6 – system reboot.
- Numerous systems don't use other runlevels, i.e. the runlevels: 2, 3, 4 and 5 are undistinguishable.
- Many systems define:
  - 2 – multiuser with no network,
  - 5 – multiuser with GUI.

  The runlevel 4 is almost never implemented.
- A common practice is defining alias names for runlevels.

Changing Runlevels

- For changing current runlevel root should use the `init` command:

SysV shutdown

```
# init 0
```

SysV reboot

```
# init 6
```

Defining Runlevels – */etc/inittab*

- Available runlevels, the default runlevel and system behaviour for each runlevel are defined in the file `/etc/inittab` .

/etc/inittab entry syntax

```
id:runlevel:action:process
```

- The fields are as follows:
  - *id* – entry identifier (1-4 characters),
  - *runlevel* – runlevel number,
  - *action* – pre-defined (see `man 5 inittab`) action, for example: `respawn` – start-up and keep running, `wait` – start and wait for its termination,
  - *process* – command to be executed.

getty

- getty or agetty (Linux) opens ttys, prompts for password and starts user's login shell.
- The following fragment of /etc/inittab starts agettys for tty1 in single user mode and for tty1–tty6 on multi-user runlevels:

```
c1:12345:respawn:/sbin/agetty 38400 tty1 linux
c2:2345:respawn:/sbin/agetty 38400 tty2 linux
c3:2345:respawn:/sbin/agetty 38400 tty3 linux
c4:2345:respawn:/sbin/agetty 38400 tty4 linux
c5:2345:respawn:/sbin/agetty 38400 tty5 linux
c6:2345:respawn:/sbin/agetty 38400 tty6 linux
```

Pure System V (Sysvinit)

- This solution is commonly implemented on enterprise-class systems.
- For each runlevel there is a directory: `/etc/rc0.d/`, `/etc/rc1.d/`, ..., `/etc/rc6.d/` .
- Each of those directories contains symlinks to all daemon control scripts.
- Symlink names start with `K` or `S` followed by some two-digit number and a script name, for example: `K36mysqld` or `S55sshd` .
- Symlinked scripts are run sequentially in lexicographical order. If a name starts with `K` the script is executed with `stop` command. For `S` – `start` .
- Additional tools are required for daemon management.

Runlevel Management in */etc/inittab* – Sysvinit

```
id:3:initdefault:

# Boot-time system configuration/initialization script.
si::sysinit:/etc/init.d/rcS

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
# Runlevel 0 is halt. Runlevel 6 is reboot.
# Runlevel 1 is single-user. Runlevels 2-5 are multi-user.
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# What to do at the "3 finger salute".
ca::ctrlaltdel:/sbin/shutdown -t1 -h now
```

Simplified SysV-based Subsystems

- Flexible general-purpose sysv-based Linux systems may use simplified solutions:
  - multi-user runlevels are merged,
  - only scripts that should be started are specified/symlinked,
  - scripts are started in order based on other factors,
  - utilities for daemon management are optional.

OpenRC

- OpenRC has been initially developed for Gentoo Linux:
  - symlinks in `/etc/runlevels/` subdirectories: `boot/`, `default/`, `single/` and `nonetwork/` are created for daemon startup scripts,
  - the startup order is random; dependencies may be encoded inside scripts,
  - the names of scripts and symlinks are the same.
- The symlinks can be managed by the commands: `rc-status` and `rc-update` .

The following two commands are equivalent:

Assigning SSH daemon to the default runlevel in Gentoo

```
# ln -s /etc/init.d/sshd /etc/runlevels/default/sshd
# rc-update add sshd default
```

Runlevel Management in */etc/inittab* – OpenRC

```
id:3:initdefault:

# System initialization, mount local filesystems, etc.
si::sysinit:/sbin/openrc sysinit

# Further system initialization, brings up the boot runlevel.
rc::bootwait:/sbin/openrc boot

l0:0:wait:/sbin/openrc shutdown
l1:1:wait:/sbin/openrc single
l2:2:wait:/sbin/openrc nonetwork
l3:3:wait:/sbin/openrc default
l4:4:wait:/sbin/openrc default
l5:5:wait:/sbin/openrc default
l6:6:wait:/sbin/openrc reboot

# What to do at the "3 finger salute".
ca::ctrlaltdel:/sbin/shutdown -t1 -h now
```

Outline of the lecture

/etc/rc.conf

- In BSD systems the idea of runlevels has been discarded.
- Most of the system start-up configuration is specified by the file `/etc/rc.conf` .
- A special variable or a set of variables control daemons to be started at boot-up.

Example – Starting Daemons in FreeBSD

‘/etc/rc.conf’ extract

```
...
sshd_enable=yes
moused_enable=yes
ntpd_enable=yes
powerd_enable=yes
...
```

BSD Unices and */usr/local*

- Contrary to Linux, BSD Unices strongly separate a system and software running on it.
- Software which is non-critical to the system itself is installed under the `/usr/local` :
  - `/usr/local/etc/` – configuration files,
  - `/usr/local/etc/rc.d/` – daemon controlling scripts,
  - `/usr/local/sbin/` – daemons,
  - etc.

Outline of the lecture

Systemd

- Windows-inspired (*svchost.exe*) sysv/bsd replacement for desktop-oriented Linux systems.
- Binary, complex, monolithic, tightly integrated solution which eliminates daemon controlling scripts.
- It tries to automagically do everything between a kernel and userland GUI applications.
- It automates managing some desktop-related services by some kind of *plug-and-play*.
- Limited documentation, complexity, frequent critical changes, binary files, lack of modularity, automation and *aggressive parallelization* make controlling/tracing/debugging systemd very questionable.

Systemd: a Case Study

Init-free Linux Distributions

- Main distributions not implementing systemd include:
  - Devuan,
  - Gentoo,
  - Slackware.
- Popular Arch Linux clones which are init-free:
  - Artix,
  - Hyperbola,
  - Parabola.
- A more comprehensive list and some arguments against using systemd can be found on: `https://nosystemd.org/` , `https://suckless.org/sucks/systemd/` and others.

Systemd Services

- Under systemd daemons are called *services*.
- Services are defined by `/usr/lib/systemd/system/*.service` files similar to Windows **.ini* files.
- There are also `*.mount`, `*.socket` and a few other file types for analogous purposes. All those are called *units*.
- Default configuration can be overwritten by placing unit files in `/etc/systemd/system/`. Major changes to default configuration are generally not recommended.
- Units can be run automagically if systemd finds it desirable.

Systemd Targets

- Target is a set of services, mounts, sockets, etc. So-called *wants* are used to include them.
- Target may include other targets by *wants*.
- Typically there are approx. 50-100 targets.
- The default target (*multi-user*) directly and indirectly includes all units to be executed at boot time.

Systemd *systemctl*

- `systemctl` is a general-purpose management utility for systemd.
- With this command users are able to check systemd status, ask systemd for starting/stopping services or enabling/disabling them from the default target.
- Administrators should be aware that systemd may start a service automatically even if instructed not to do it.
- Increasing number of `*ctl` utilities which act as interfaces to systemd are provided with newer versions.

Systemd and Unix Daemons

- Most Unix daemons can be started as systemd services.
- For some daemons systemd introduces alternative, tightly integrated (and quite often: poorly designed) services that are obligatory. An example is `journald` – a systemd binary replacement for standard Unix `syslog` daemon.
- Some standard daemons can still be installed and double such services.

Outline of the lecture

Cron – Key Concepts

- Cron is a standard Unix daemon for executing commands periodically on certain times and dates.
- Once a minute cron checks *crontab* files and executes all commands that are scheduled to run at this time.
- A system crontab as well as users' crontabs can exist. The system crontab name is usually one of the followings:
  - `/etc/crontab` ,
  - `/var/spool/cron/root` ,

  while users' crontabs are located in `/var/spool/cron/` and named with user names, for example: `/var/spool/cron/smithj` .
- Cron doesn't need to be restarted nor reloaded on crontab changes.

Cron Implementations

- Popular cron implementations include:
  - *vixie-cron*,
  - *dcron*,
  - *cronie*,
  - *anacron*.
- The main differences include:
  - crontab file location and slight syntax differences,
  - ways of specifying which users can access the cron,
  - some extensions.

System Crontab File – */etc/crontab*

Crontab file syntax

```
min hour dmonth month dweek   user   command
```

- The fields are as follows:
  - *min* – minutes (0-59),
  - *hour* – hour (0-23),
  - *dmonth* – day of month (1-31),
  - *month* – month (1-12)
  - *dweek* – day of week (0 and 7 denote Sunday),
  - *user* – this field exists only in the main crontab file of certain cron implementations,
  - *command* – command to execute; specifying full paths is highly recommended.
- Numerous special characters can be used for the first 5 fields:
  - `*` matches any value,
  - `,` allows specifying lists,
  - `-` allows specifying ranges,
  - `/` specifies increment value, for example: `*/2` – every even value.

Crontab – Examples

Power off the computer every day at 11:15 p.m.

```
15 23 * * *   root   init 0
```

Make a backup copy every business day at 10 p.m.

```
0 22 * * 1-5   root   /root/make_backup
```

Prepare on Friday 13^{th}

```
1 0 13 * 5   root   /root/enhanced_fault_tolerance
```

(Optional) Split Crontab

- Crontab entries are preceded by some small set of variables (`HOME`, `PATH`, `SHELL`).
- Crontab entries my require different variable settings.
- File management is easier than file content management.
- For those purposes some implementations allow splitting a crontab into several separate files placed in the `/etc/cron.d` directory.

(Optional) Crontab Directories – */etc/cron.*/*

- Some cron implementations check the following directories:
  - `/etc/cron.hourly/`
  - `/etc/cron.daily/`
  - `/etc/cron.weekly/`
  - `/etc/cron.monthly/`

  and execute all programs (usually scripts) placed in them with respective frequency.

Users' Crontabs

- Users are able to define their own crontabs. Those are stored in `/var/spool/cron/*` files named the same as their owners (for example: `/var/spool/cron/smithj`).
- Usually the `crontab` command is intended for crontab management.
- Some systems allow all users to use cron, while other may grant this access:
  - only to members of the `cron` group,
  - only to users listed in `/etc/cron.allow` ,
  - based on other factors.

Outline of the lecture

Event Logging – Key Concepts

- Kernel and daemons send messages to a *syslog* daemon.
- Each syslog message contains date and time, sending host, sending facility or program (service, kernel or daemon name), level (warning, error etc) and detailed information.
- Syslog messages are then redirected to a terminal (for example: `tty12`), log files (`/var/log/*`) or to a remote syslog instance.
- Some daemons don't use a syslog but manage their own log file hierarchy.
- It is possible to interact with syslog via the command line utility: `logger` .

Syslog and Syslog-ng

- Syslog is a protocol (IETF RFC 5424) while syslog-ng is an implementation.
- Syslog-ng extends the original syslogd model with rich content-based filtering capabilities, flexible configuration options and adds important features like remote logging (over TCP/IP).
- Syslong-ng is installed as a default syslog for Linux, however it's been ported to FreeBSD, AIX, HP-UX, Solaris, Tru64 etc.
- For most systems the syslog-ng is a requisite dependency.

Syslog-ng Configuration

- The syslog-ng configuration is typically stored in the file `/etc/syslog-ng.conf` .
- It defines:
  - `sources` – it's safe to leave it untouched,
  - `destinations` – where to redirect messages,
  - `filters` – filtering criteria,
  - `logs` – bind sources and destinations based on filters.
- Many administrators log messages from main daemons (DNS, mail, ssh) to separate files. In addition all messages are displayed on the `tty12` .
- For security reasons it is safe to additionally redirect all syslog messages to a remote host.

Destinations and Filters – Examples

- program is a daemon name while facility is a type of service – any daemon that provides this kind of service.

Sample filters

```
filter f_mail { facility(mail); };
filter f_named { program("sshd"); };
filter f_err { level(err); };
```

- Note that TCPconnections to the port 524 are used for remote logging.

Sample destinations

```
destination d_mail { file("/var/log/mail.log"); };
destination d_sshd { file("/var/log/sshd.log"); };
destination d_remote_host { tcp("194.29.178.3" port(524)); };
destination d_messages { file("/var/log/messages"); };
destination d_console_all { file("/dev/tty12"); };
```

Logs – Examples

- `flags(final)` indicates final destination, i.e. it prevents storing the same entry in several log files.

Sample logs

```
log { source(src); destination(d_merckx); };
log { source(src); destination(d_console_all); };
log { source(src); filter(f_mail); destination(d_mail); flags(final); };
log { source(src); filter(f_sshd); destination(d_sshd); flags(final); };
log { source(src); destination(d_messages); };
```

Log Analyzing

- `less` is a recommended command for manual log reading cause it doesn't cache the whole file. Never open files of the size several gigabytes with editors!
- Pipeline filters, especially `grep` are very useful for manual log reading if all messages are stored in a single file.
- Most systems offer tools for automated log analyzing. They are usually perl rexexp parsers run once a day (by cron) and send daily reports to an administrator's e-mail address.

Log Administration – Logrotate

- Logrotate is another useful tool for log management which is initiated once a day by cron.
- Logrotate periodically (for example: once a week) cuts off log files and compresses them.
- Very old log archives (for example: older than a month) are automatically deleted.
- Logrotate can operate on syslog files as well as other log files stored in `/var/log/*`.

Log-rotated log files

```
# ls /var/log/ | grep messages
messages
messages-20110331.gz
messages-20110407.gz
messages-20110414.gz
messages-20110421.gz
```

Outline of the lecture

Standard Commands – *dmesg*

- `dmesg` is used to examine or control the kernel ring buffer. The program helps users to print out their bootup messages.
- Always use `dmesg` in a pipeline with filters (typically `grep` or `more`).
- Syslog daemon is usually started when entering the multi-user runlevel. `dmesg` logs all messages since loading a kernel so it is extremely useful for identifying hardware detection problems and improper driver initialization.

Modular Kernels (Linux)

- Linux kernels can be monolithic, modular or mixed.
- Some Linux distros (for example: Gentoo) allow building a custom kernel.
- Most distributions with pre-build kernels use modular ones.
- Kernel modules typically are placed in the `/lib/modules/` subdirectory (named based on the kernel version).
- Module management utilities include:
  - `lsmod` – list loaded modules,
  - `modinfo` – get module information,
  - `modprobe` – load a module (options can be specified),
  - `rmmod` – unload a module.
- Note that some modules are automatically loaded as dependencies and cannot be unloaded before their dependants.

Modular vs Monolithic Kernels (Linux)

- Monolithic kernels are smaller and faster.
- Modular kernels ore more flexible:
  - modules can be unloaded then loaded with different parameter sets,
  - can handle runtime hardware changes.
- Desktop oriented distros prefer modular kernels.

Module Autoloading (Linux)

- Most modules are loaded at boot time based on the hardware auto-detection.
- Some are loaded automagically shall the need arise.
- Other modules have to be explicit specified. Unfortunately there is no standard solution for Linux systems. Consult your system's manual for details.
- Module loading options are usually specified in the file `/etc/modprobe.conf` or the `/etc/modules-load.d` directory.