Open Computing Facility

Services and Processes

Lecture 4

aly, lemurseven

(slide credits kmo, rrchan, cooperc, keur)

Course Resources

- Your facilitators!
- Ed, Gradescope
- OCF Slack (ocf.io/slack) or Discord (ocf.io/discord) #decal-general
- All materials available at decal.ocf.io
- Ask questions / work on lab with us during lab sessions! (Tuesday 8-9pm in OCF Lab)

Outline

- Processes
- Services
- Intro to systemd

Disclaimer

- This is not cs162
- This topic gets very deeply technical and we will only scratch the surface here
- *Main goal:* become a more productive Linux user, not a kernel hacker

Processes

What is a process?

A process is a **single instance of a program.**

Processes are isolated from one another and have their own memory, threads, etc. (Additional isolation, such as filesystem or network isolation, is also possible.)

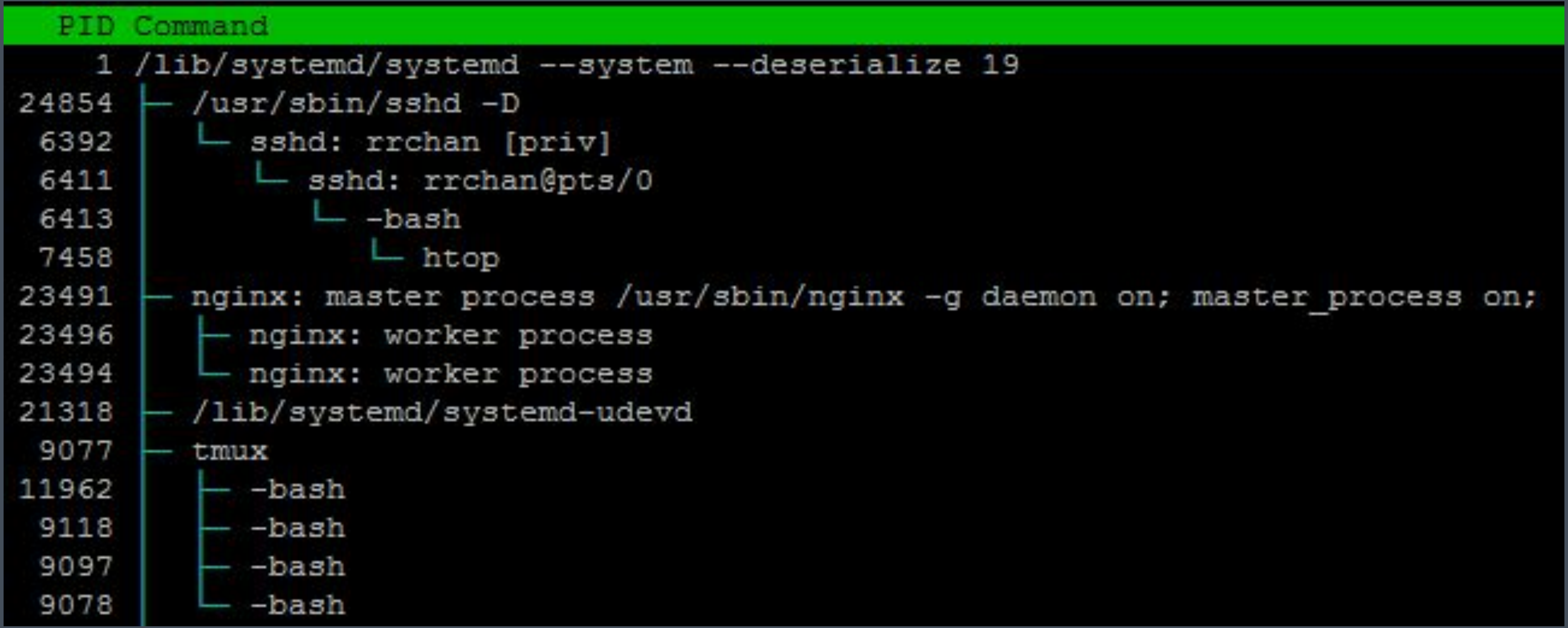
What is a process?

- PID: Process ID
- PPID: Parent’s PID
- UID: User running the process
- The program (executable) that the process is running
- The args (command line) of the process

(and more...)

Init

- First process started at boot, given PID 1
  - Manages all other services and processes
- Run `htop`, open tree view (f5). What is the root of the tree?



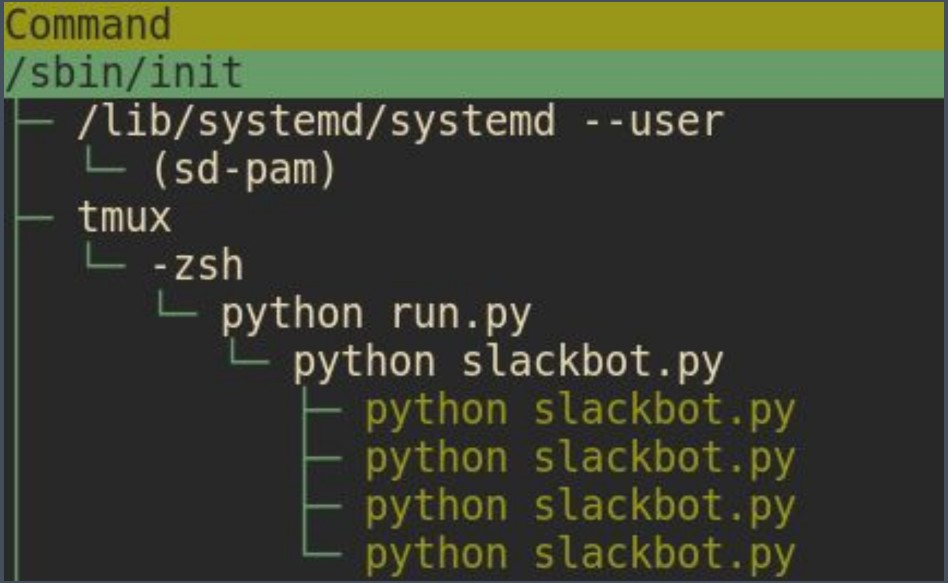
```
  PID Command
    1 /lib/systemd/systemd --system --deserialize 19
24854 ├─ /usr/sbin/sshd -D
 6392 │  └─ sshd: rrchan [priv]
 6411 │     └─ sshd: rrchan@pts/0
 6413 │        └─ -bash
 7458 │           └─ htop
23491 ├─ nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
23496 │  ├─ nginx: worker process
23494 │  └─ nginx: worker process
21318 ├─ /lib/systemd/systemd-udevd
 9077 ├─ tmux
11962 │  ├─ -bash
 9118 │  ├─ -bash
 9097 │  ├─ -bash
 9078 │  └─ -bash
```

Process Hierarchy

Each process is created by a parent

(Except PID 1)

Processes can have many children

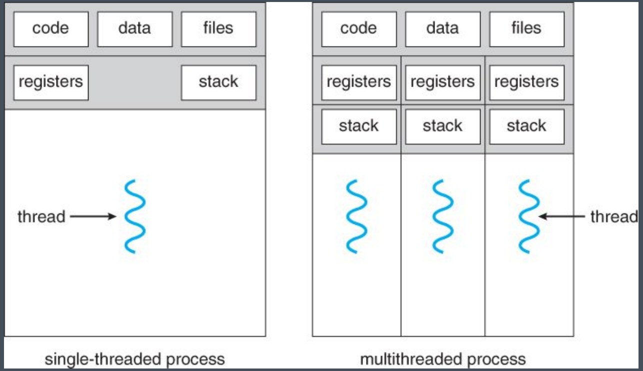


```
Command
/sbin/init
├─ /lib/systemd/systemd --user
│   └─ (sd-pam)
├─ tmux
│   └─ -zsh
│       └─ python run.py
│           └─ python slackbot.py
│               ├─ python slackbot.py
│               ├─ python slackbot.py
│               ├─ python slackbot.py
│               └─ python slackbot.py
```

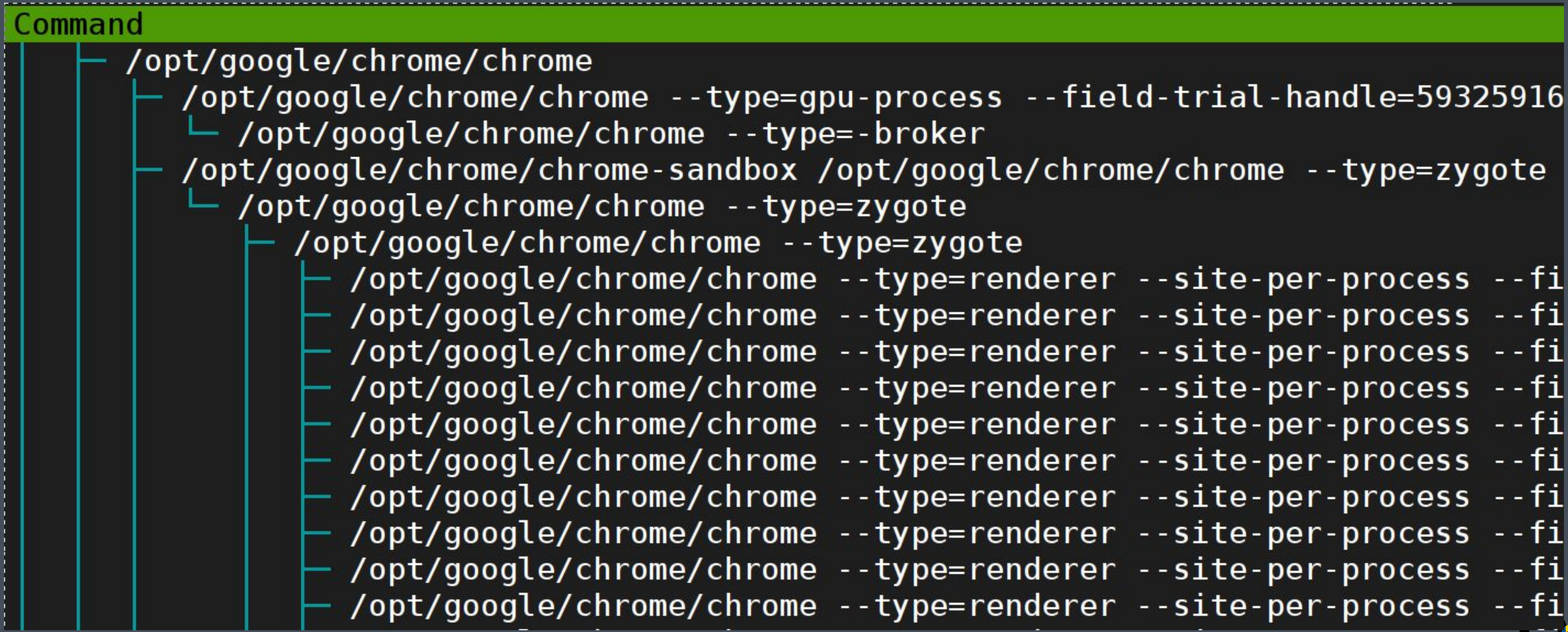
Processes vs Threads

A process has one or more threads.

- Processes have their own data and code, must use pipes, files, etc. to communicate with one another
- Threads share the same process but have different system states ("multithreaded process")



Aside: Why does Chrome spawn so many processes?



```
Command
 │     ├─ /opt/google/chrome/chrome
 │     │  ├─ /opt/google/chrome/chrome --type=gpu-process --field-trial-handle=59325916
 │     │  │  └─ /opt/google/chrome/chrome --type=-broker
 │     │  ├─ /opt/google/chrome/chrome-sandbox /opt/google/chrome/chrome --type=zygote
 │     │  │  └─ /opt/google/chrome/chrome --type=zygote
 │     │  │     ├─ /opt/google/chrome/chrome --type=zygote
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
 │     │  │     │  ├─ /opt/google/chrome/chrome --type=renderer --site-per-process --fi
```

How are processes created?

A process will **fork**(2) into a two new processes, which continue from the same place

The parent keeps the original PID and the child gets a new PID

Optionally, the new process (the child) **exec**(3) and begin running a new program

How many "henlo"s get printed?

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

How many “henlo”s get printed?

PID 1000

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

PID 1000

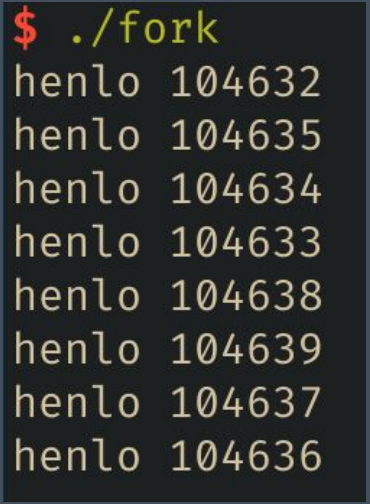
```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

PID 2000

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

How many "henlo"s get printed?

```
int main( void ) {
    fork();    2^1
    fork();    2^2
    fork();    2^3
    printf( "henlo: %d\n", getpid() );
}
```



```
$ ./fork
henlo 104632
henlo 104635
henlo 104634
henlo 104633
henlo 104638
henlo 104639
henlo 104637
henlo 104636
```

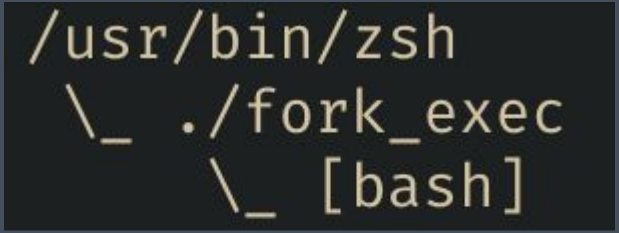
Have you seen this code before?

```
:(){ :|:& };:
```

```
bomb() {
    bomb | bomb &
} bomb
```

Making the child do something

```
int main( void ) {
    if ( fork() > 0 ) {
        /* parent process */
        wait( NULL );
    } else {
        /* child process */
        execv( “/bin/bash”, NULL );
    }
}
```



```
/usr/bin/zsh
 \_ ./fork_exec
     \_ [bash]
```

htop exploration

- htop(1): interactive terminal process manager
- https://peteris.rocks/blog/htop/

- PID/User/Command/CPU%/MEM%/TIME: self-explanatory
- PRI/NI: Priority and Niceness (take cs162, or google scheduling policy/SCHED_OTHER)
- VIRT/RES/SHR: virtual image, resident size, shared memory usage
- S: Status (S = sleep, R = running, T = terminated, Z = zombie)

What does this code do?

```
int main( void ) {
    if ( fork() > 0 ) {
        /* parent process */
        sleep( 1 );
    } else {
        /* child process */
        exit( 1 );
    }
}
```

What does this code do?

Parent

Child

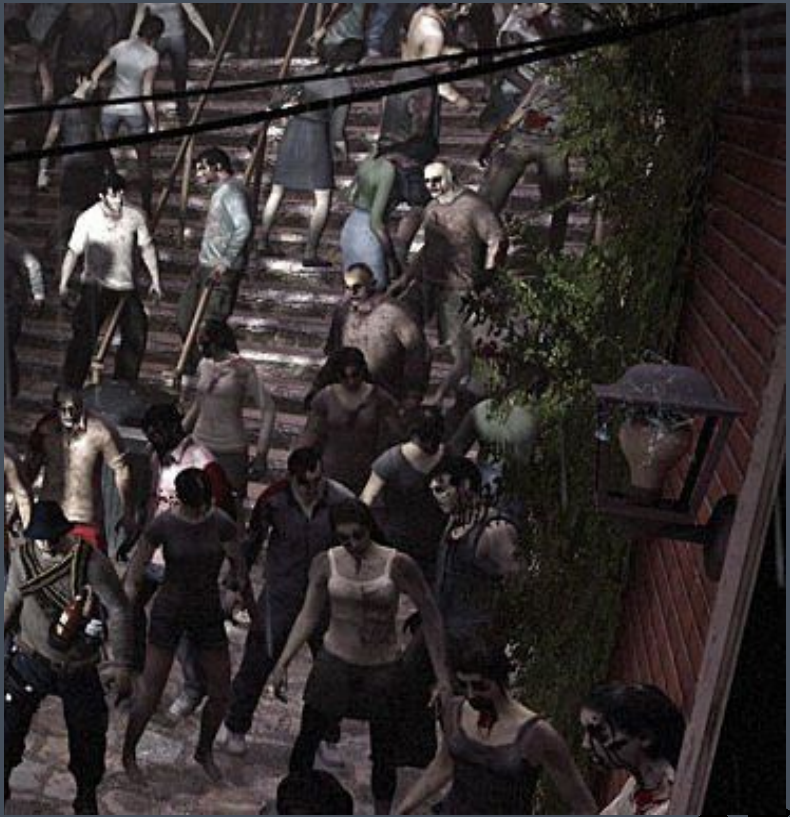
```
\_ /usr/bin/zsh
    \_ ./zombie-creator
        \_ [zombie-creator] <defunct>
```

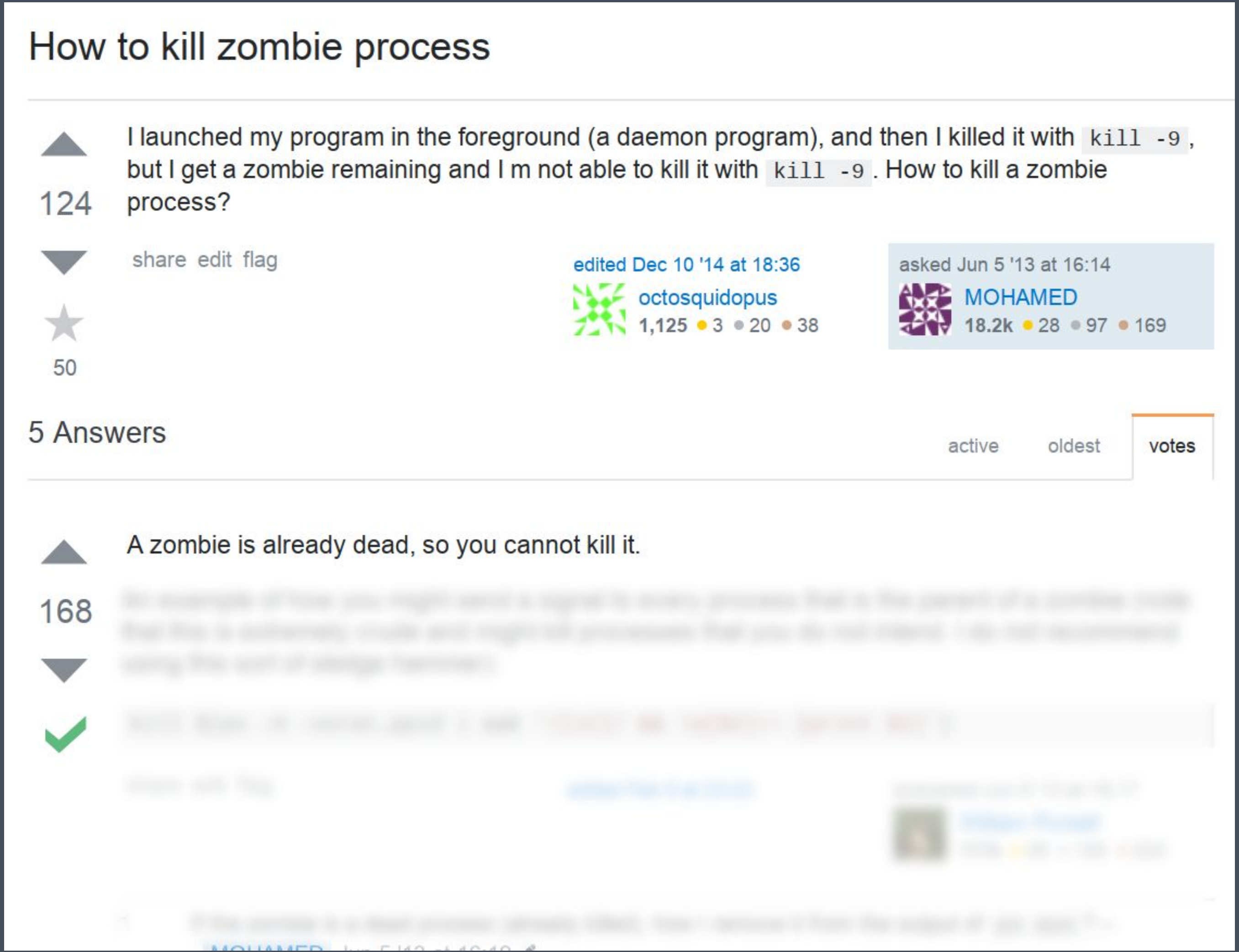
Zombie Process

When a child has died but has not been “reaped”

Child metadata stays in process table so parent can collect exit status

Totally normal, all children that exit are zombies!





How to kill zombie process

124

I launched my program in the foreground (a daemon program), and then I killed it with `kill -9`, but I get a zombie remaining and I m not able to kill it with `kill -9`. How to kill a zombie process?

50

share edit flag

edited Dec 10 '14 at 18:36
octosquidopus
1,125 ● 3 ● 20 ● 38

asked Jun 5 '13 at 16:14
MOHAMED
18.2k ● 28 ● 97 ● 169

5 Answers

active oldest votes

168

A zombie is already dead, so you cannot kill it.

[illegible]

What happens when a process dies

```
if ( fork() > 0 ) {
    /* parent process */
    sleep( 1 );
} else {
    /* child process */
    exit( 1 );
}
```

When a process exits, returns int (exit code 0-255)

0 is success and anything else indicates an error

What happens when a process dies

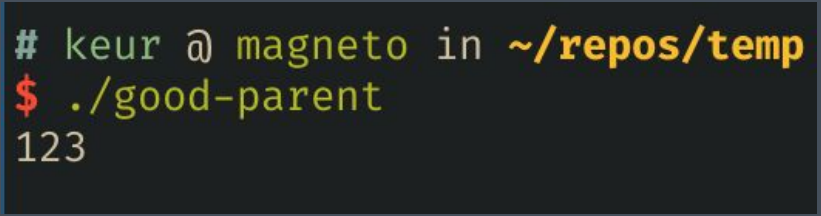
```
if ( fork() > 0 ) {
    /* parent process */
    sleep( 1 );
} else {
    /* child process */
    exit( 123 );
}
```

When a process exits, returns int (exit code 0-255)

0 is success and anything else indicates an error

Parents need to wait() on children

```
int main( void ) {
    if ( fork() > 0 ) {
        int status;
        sleep( 1 );
        wait( &status );
        printf( "%d\n", WEXITSTATUS ( status ) );
    } else {
        exit( 123 );
    }
```



```
# keur @ magneto in ~/repos/temp
$ ./good-parent
123
```

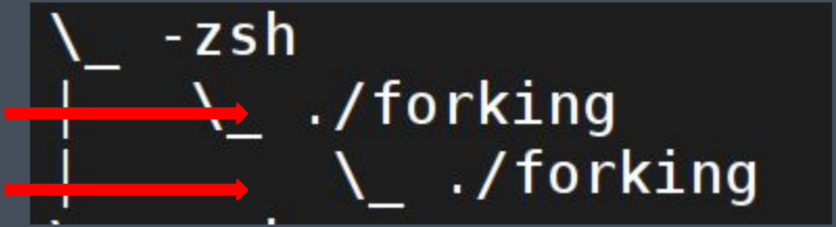
What happens if the parent exits first?

```c
int main( void ) {
    if ( fork() > 0 ) {
        /* parent process */
        sleep ( 1 );
        exit( 0 );
    } else {
        /* child process */
        sleep( 100 );
        exit( 123 );
    }
```

Process tree after start

Parent

Child

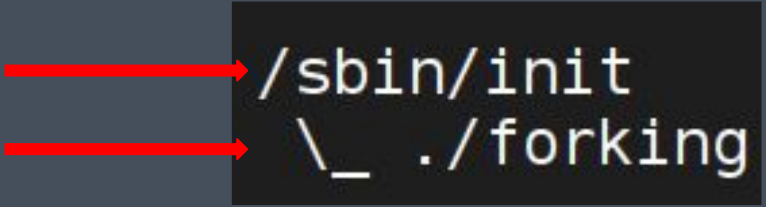


```
\_ -zsh
 |    \_ ./forking
 |        \_ ./forking
```

Process tree after 1 second

Parent

Child



```
/sbin/init
 \_ ./forking
```

What happens if the parent exits first?

If parent exits first, orphan processes are re-parented by the init process

init reaps all orphans that are zombies

When can zombies become a problem?

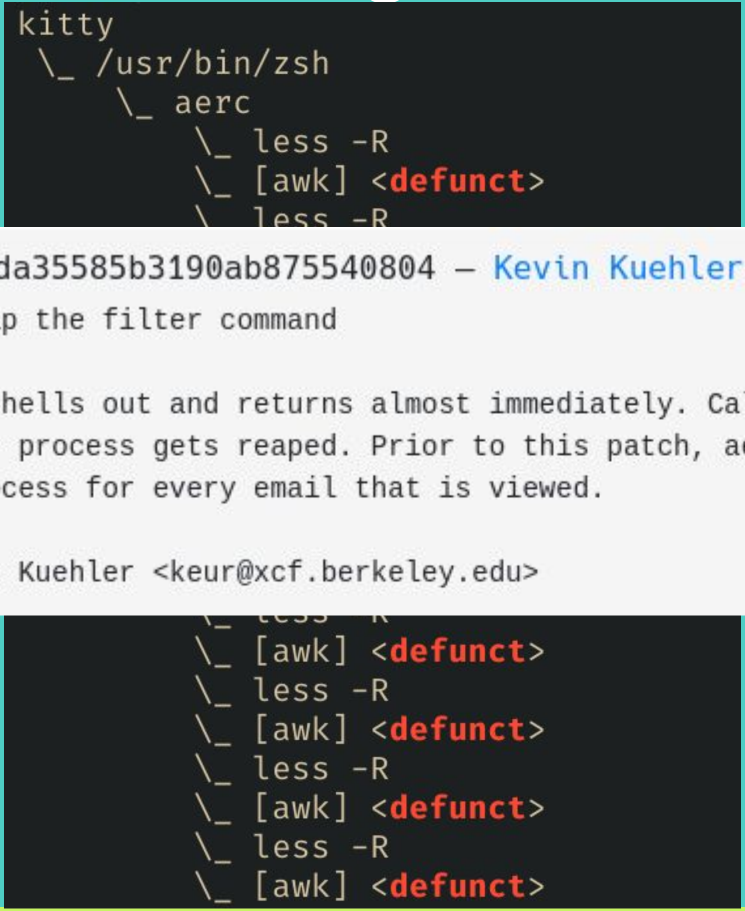
Parent doesn't wait() on children

Parent is long running process

Zombie child processes never become orphans

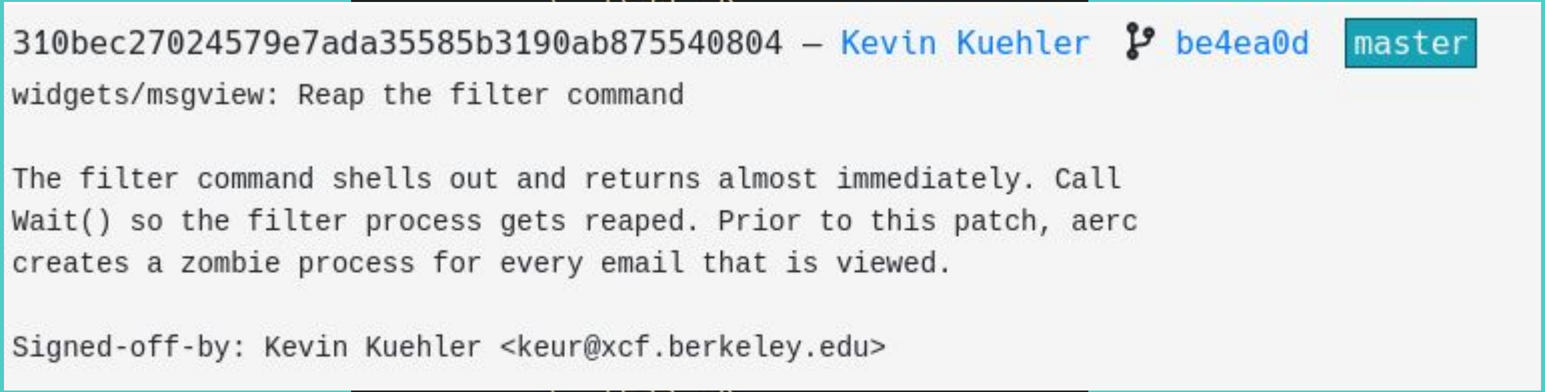
Resource leakage!

Zombie leakage in the wild



```
kitty
 \_ /usr/bin/zsh
     \_ aerc
         \_ less -R
         \_ [awk] <defunct>
         \_ less -R
```

```
         \_ [awk] <defunct>
         \_ less -R
         \_ [awk] <defunct>
         \_ less -R
         \_ [awk] <defunct>
         \_ less -R
         \_ [awk] <defunct>
```



```
310bec27024579e7ada35585b3190ab875540804 — Kevin Kuehler   be4ea0d  master
widgets/msgview: Reap the filter command

The filter command shells out and returns almost immediately. Call
Wait() so the filter process gets reaped. Prior to this patch, aerc
creates a zombie process for every email that is viewed.

Signed-off-by: Kevin Kuehler <keur@xcf.berkeley.edu>
```

Inter-process Communication

Various ways in which processes can communicate

- Exit codes
- Signals (e.g. SIGTERM, SIGKILL, SIGINT)
- Pipes (STDIN, STDOUT, STDERR)
- Sockets (UNIX socket, IP socket)
- Message Bus (e.g. dbus on Linux)
- ... and many many more...

Process Signals

- **SIGTERM**: tell a process to exit now
- **SIGKILL**: terminate process immediately
- **SIGINT**: interrupt, when you press Ctrl+C
- **SIGHUP**: the user closes the terminal window
- **SIGWINCH**: terminal window resized
- **SIGSTOP** / **SIGCONT**: stop/resume

SIGKILL and SIGSTOP cannot be handled

Services

What is a service?

- A service is a type of process known as a **Daemon**
  - A daemon is a noninteractive background process
  - Typically names end with a ‘d’, but not always the case
- Services are controlled by an **init** system
- Examples: sshd, httpd, rsyslogd, nginx, postfix, ...
- Not a strict definition

Services - What’s the point?

- Can run for long periods of time
  - Useful in many cases, such as web servers
- Can be publicly accessible or shared between multiple users
  - Can be networked.
- Ex. sshd allows for incoming ssh connections - very important
- Can write your own services!

Systemd

- **Systemd** is an init system that manages processes and services
  - Most commonly used init system on modern Linux systems
- Provides tools for users to manage services
  - `systemctl` - start, stop, check status, and more
  - `journalctl` - check systemd journal
- Some debate in the free software community about systemd
  - ~~systemd bad use runit~~

Systemd Unit Files

- Service behavior defined by systemd **unit files**
  - How should this service start up? How should it respond to various `systemctl` management commands?

A (simplified) Unit File: helloworld.service

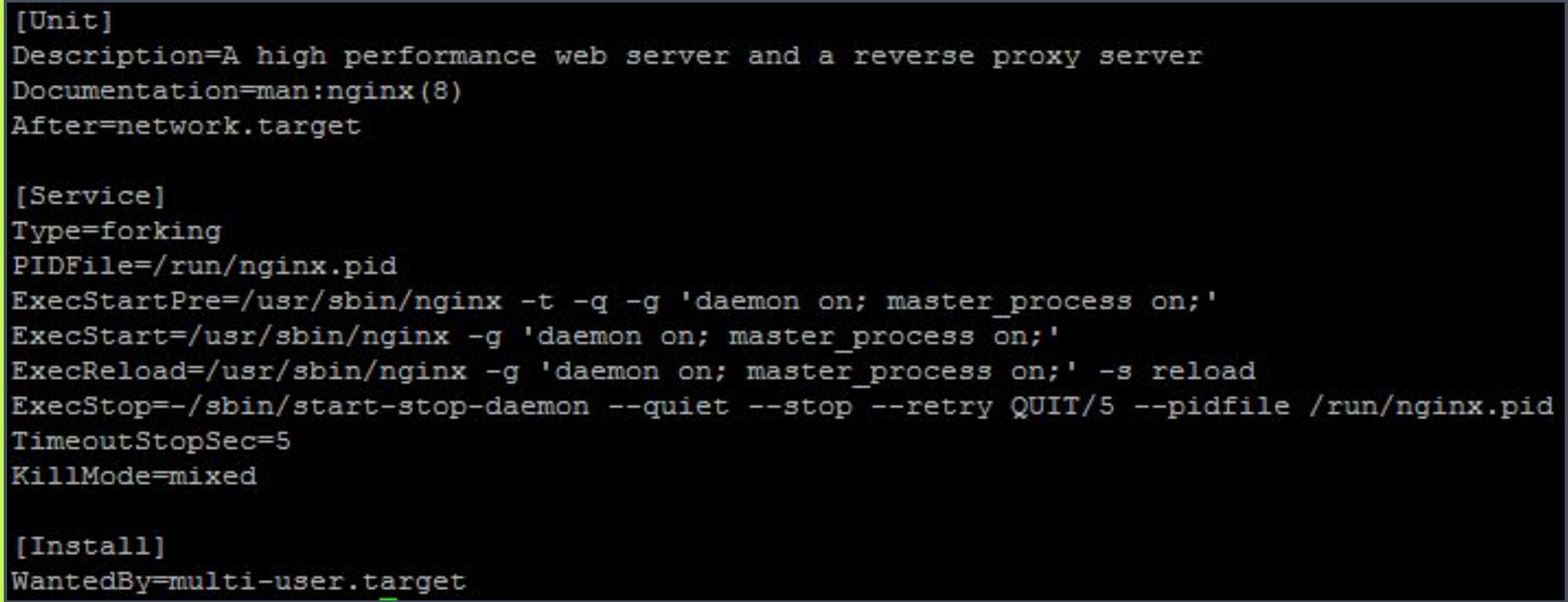
```
[Unit]
Description=A simple unit file
[Service]
ExecStart=/usr/bin/helloworld
User=ocfstaff
Restart=always
[Install]
WantedBy=multi-user.target
```

Description - what the service does

Commands

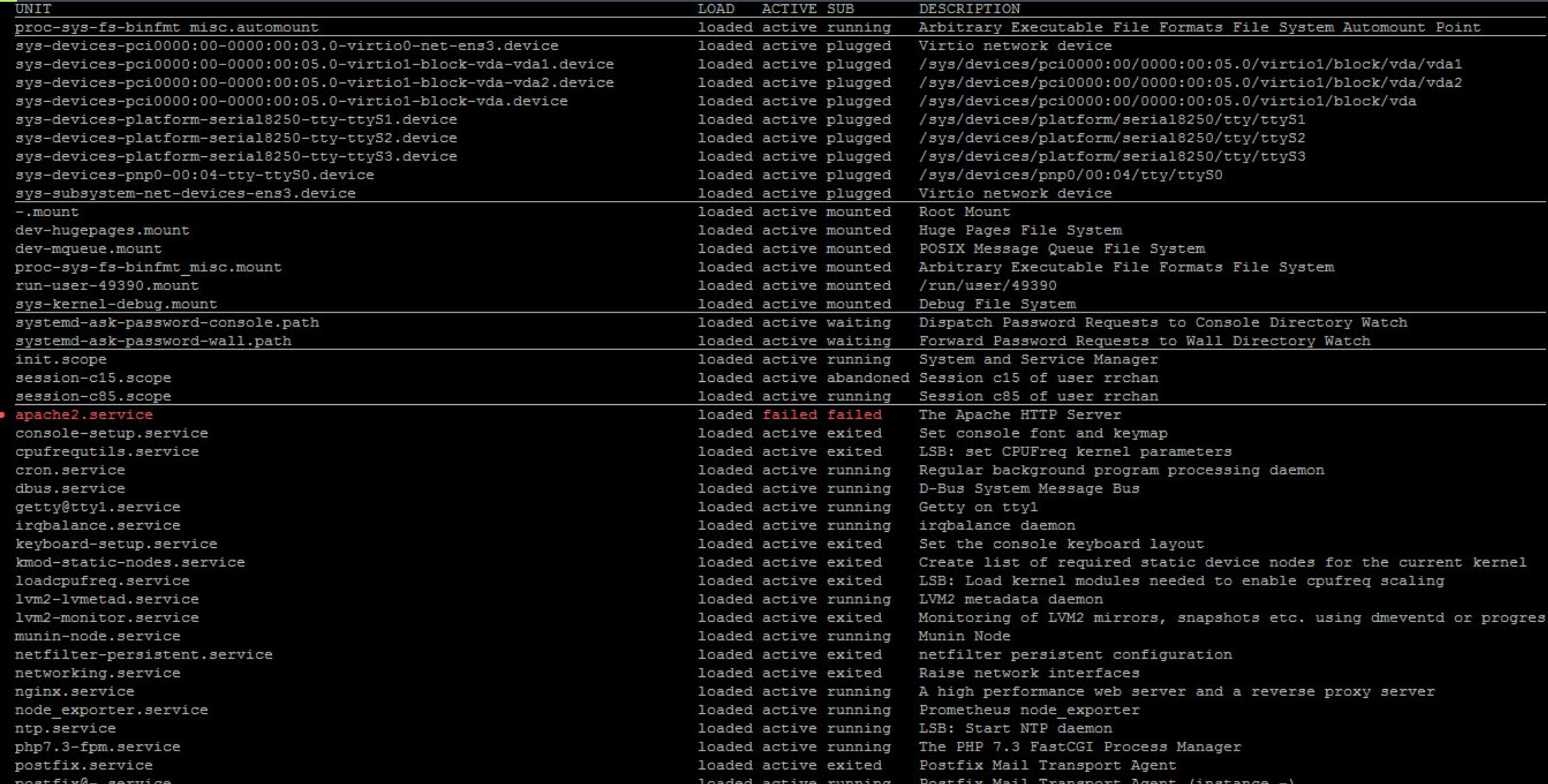
When this unit should get started

Example Unit file - Nginx



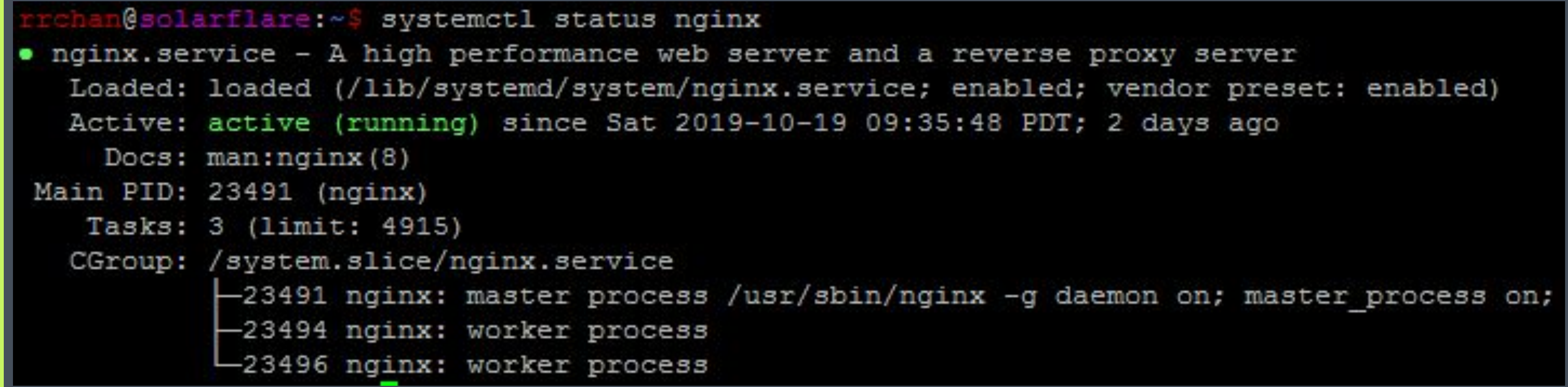
```
[Unit]
Description=A high performance web server and a reverse proxy server
Documentation=man:nginx(8)
After=network.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'
ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'
ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload
ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid
TimeoutStopSec=5
KillMode=mixed

[Install]
WantedBy=multi-user.target
```

systemctl



```
  UNIT                                                                      LOAD   ACTIVE SUB       DESCRIPTION
  proc-sys-fs-binfmt_misc.automount                                         loaded active running   Arbitrary Executable File Formats File System Automount Point
  sys-devices-pci0000:00-0000:00:03.0-virtio0-net-ens3.device               loaded active plugged   Virtio network device
  sys-devices-pci0000:00-0000:00:05.0-virtio1-block-vda-vda1.device         loaded active plugged   /sys/devices/pci0000:00/0000:00:05.0/virtio1/block/vda/vda1
  sys-devices-pci0000:00-0000:00:05.0-virtio1-block-vda-vda2.device         loaded active plugged   /sys/devices/pci0000:00/0000:00:05.0/virtio1/block/vda/vda2
  sys-devices-pci0000:00-0000:00:05.0-virtio1-block-vda.device              loaded active plugged   /sys/devices/pci0000:00/0000:00:05.0/virtio1/block/vda
  sys-devices-platform-serial8250-tty-ttyS1.device                          loaded active plugged   /sys/devices/platform/serial8250/tty/ttyS1
  sys-devices-platform-serial8250-tty-ttyS2.device                          loaded active plugged   /sys/devices/platform/serial8250/tty/ttyS2
  sys-devices-platform-serial8250-tty-ttyS3.device                          loaded active plugged   /sys/devices/platform/serial8250/tty/ttyS3
  sys-devices-pnp0-00:04-tty-ttyS0.device                                   loaded active plugged   /sys/devices/pnp0/00:04/tty/ttyS0
  sys-subsystem-net-devices-ens3.device                                     loaded active plugged   Virtio network device
  -.mount                                                                   loaded active mounted   Root Mount
  dev-hugepages.mount                                                       loaded active mounted   Huge Pages File System
  dev-mqueue.mount                                                          loaded active mounted   POSIX Message Queue File System
  proc-sys-fs-binfmt_misc.mount                                             loaded active mounted   Arbitrary Executable File Formats File System
  run-user-49390.mount                                                      loaded active mounted   /run/user/49390
  sys-kernel-debug.mount                                                    loaded active mounted   Debug File System
  systemd-ask-password-console.path                                         loaded active waiting   Dispatch Password Requests to Console Directory Watch
  systemd-ask-password-wall.path                                            loaded active waiting   Forward Password Requests to Wall Directory Watch
  init.scope                                                                loaded active running   System and Service Manager
  session-c15.scope                                                         loaded active abandoned Session c15 of user rrchan
  session-c85.scope                                                         loaded active running   Session c85 of user rrchan
● apache2.service                                                           loaded failed failed    The Apache HTTP Server
  console-setup.service                                                     loaded active exited    Set console font and keymap
  cpufrequtils.service                                                      loaded active exited    LSB: set CPUFreq kernel parameters
  cron.service                                                              loaded active running   Regular background program processing daemon
  dbus.service                                                              loaded active running   D-Bus System Message Bus
  getty@tty1.service                                                        loaded active running   Getty on tty1
  irqbalance.service                                                        loaded active running   irqbalance daemon
  keyboard-setup.service                                                    loaded active exited    Set the console keyboard layout
  kmod-static-nodes.service                                                 loaded active exited    Create list of required static device nodes for the current kernel
  loadcpufreq.service                                                       loaded active exited    LSB: Load kernel modules needed to enable cpufreq scaling
  lvm2-lvmetad.service                                                      loaded active running   LVM2 metadata daemon
  lvm2-monitor.service                                                      loaded active exited    Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progres
  munin-node.service                                                        loaded active running   Munin Node
  netfilter-persistent.service                                              loaded active exited    netfilter persistent configuration
  networking.service                                                        loaded active exited    Raise network interfaces
  nginx.service                                                             loaded active running   A high performance web server and a reverse proxy server
  node_exporter.service                                                     loaded active running   Prometheus node_exporter
  ntp.service                                                               loaded active running   LSB: Start NTP daemon
  php7.3-fpm.service                                                        loaded active running   The PHP 7.3 FastCGI Process Manager
  postfix.service                                                           loaded active exited    Postfix Mail Transport Agent
  postfix@-.service                                                         loaded active running   Postfix Mail Transport Agent (instance -)
```

systemctl status [name]



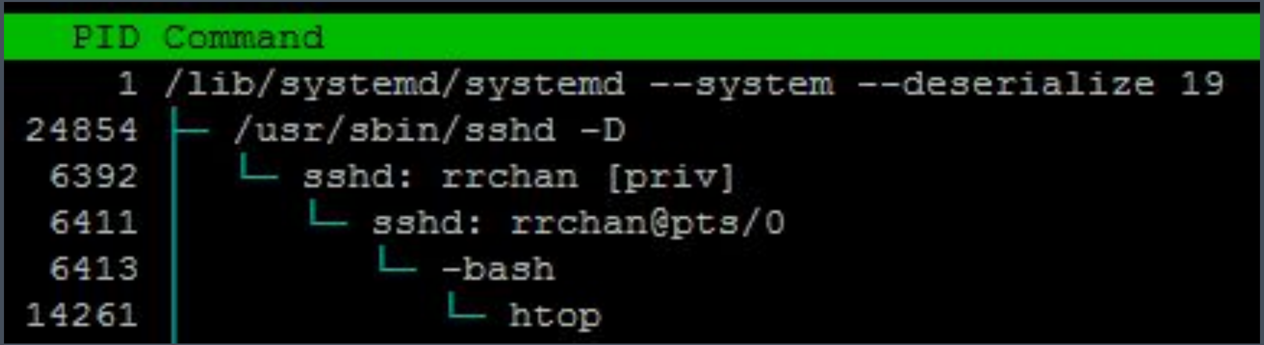
```
rrchan@solarflare:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-10-19 09:35:48 PDT; 2 days ago
     Docs: man:nginx(8)
 Main PID: 23491 (nginx)
    Tasks: 3 (limit: 4915)
   CGroup: /system.slice/nginx.service
           ├─23491 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─23494 nginx: worker process
           └─23496 nginx: worker process
```

More systemctl!

- `systemctl start [name]` - starts a service
- `systemctl stop [name]` - stops a service
- `systemctl restart [name]` - restarts a service
- `systemctl reload [name]` - reload a service’s configuration
- `systemctl enable [name]` - sets a service to start on boot
- `systemctl disable [name]` - opposite of *enable*

Example - sshd

- Handles ssh connections



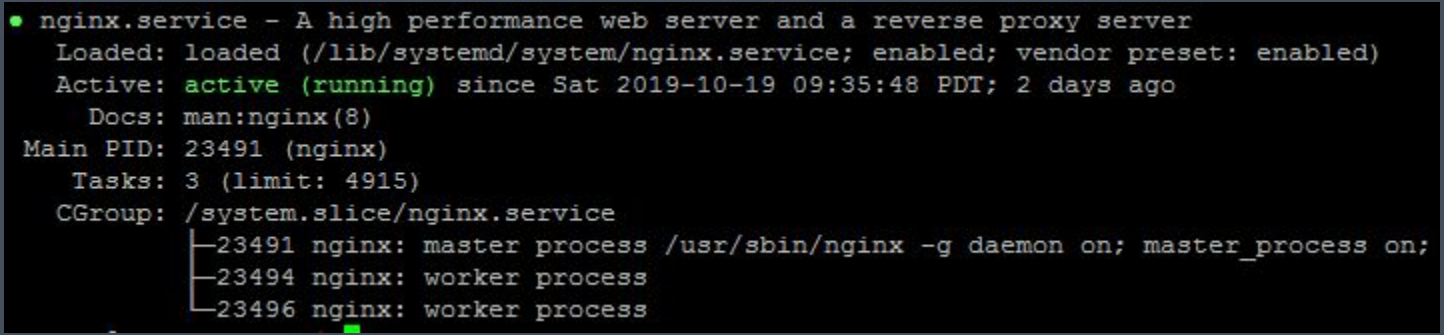
```
  PID Command
    1 /lib/systemd/systemd --system --deserialize 19
24854 ├─ /usr/sbin/sshd -D
 6392 │  └─ sshd: rrchan [priv]
 6411 │     └─ sshd: rrchan@pts/0
 6413 │        └─ -bash
14261 │           └─ htop
```

- Notice that *bash* and *htop* are children of sshd, since I am ssh’d into this machine.

Example - nginx and httpd

- nginx - http webserver daemon named ‘Nginx’
- httpd - http webserver daemon named ‘Apache’

Both examples of services which listen on port 80 and serve content.



```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-10-19 09:35:48 PDT; 2 days ago
     Docs: man:nginx(8)
 Main PID: 23491 (nginx)
    Tasks: 3 (limit: 4915)
   CGroup: /system.slice/nginx.service
           ├─23491 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─23494 nginx: worker process
           └─23496 nginx: worker process
```

Bonus: Runit

- A better init system that systemd (bc I said so)
- Minimal
- Understand your services
- Average Runit User:

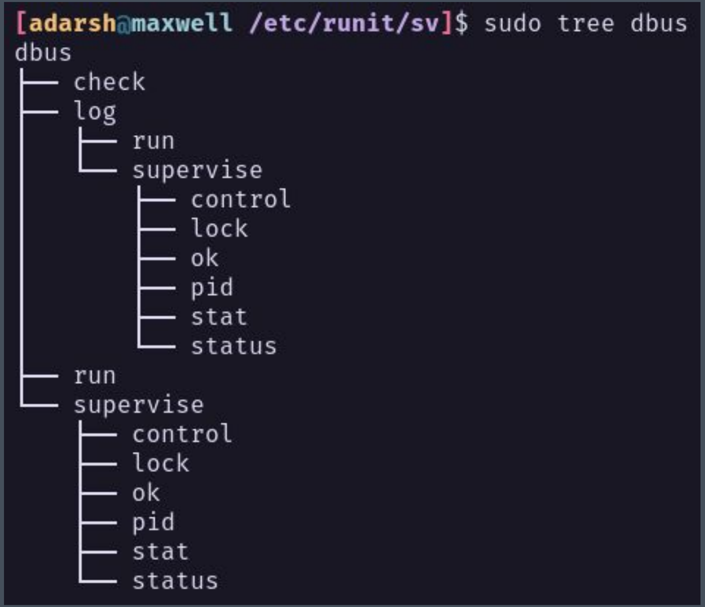
About Runit

- Created for the Void Linux distribution
  - Artix, Parabola
- Uses file, symbolic link, and folder management to start, stop, enable, disable, etc
- Everything is executable!

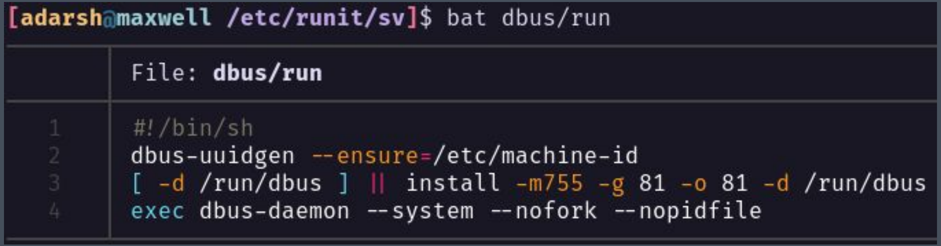
Hierarchy

- runsvdir: /etc/runit/runsvdir/current OR /run/runit/service #symlink
- Service Directory: /etc/runit/sv/myservice
  - ../myservice/run #executable
  - ../myservice/log/run #executable
  - And more!

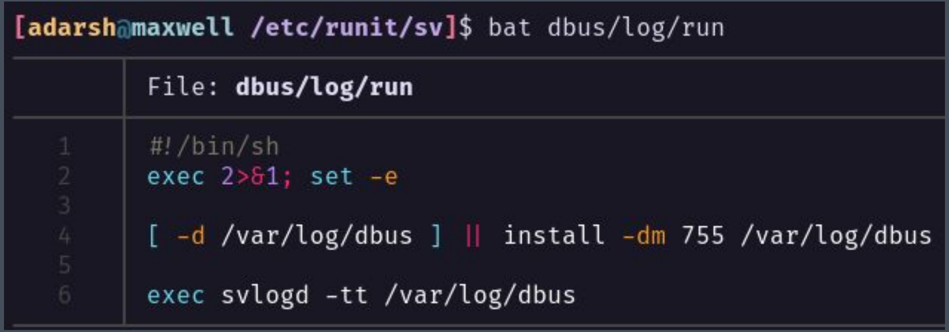
Example: dbus



```
[adarsh@maxwell /etc/runit/sv]$ sudo tree dbus
dbus
├── check
├── log
│   ├── run
│   └── supervise
│       ├── control
│       ├── lock
│       ├── ok
│       ├── pid
│       ├── stat
│       └── status
├── run
└── supervise
    ├── control
    ├── lock
    ├── ok
    ├── pid
    ├── stat
    └── status
```

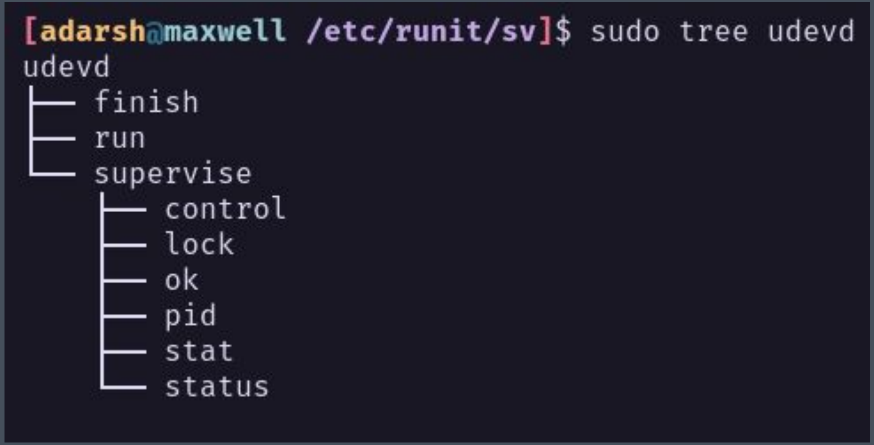


```
[adarsh@maxwell /etc/runit/sv]$ bat dbus/run
File: dbus/run
#!/bin/sh
dbus-uuidgen --ensure=/etc/machine-id
[ -d /run/dbus ] || install -m755 -g 81 -o 81 -d /run/dbus
exec dbus-daemon --system --nofork --nopidfile
```

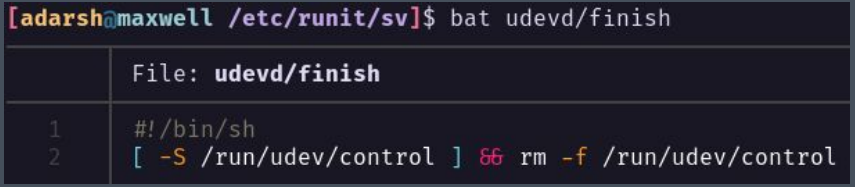


```
[adarsh@maxwell /etc/runit/sv]$ bat dbus/log/run
File: dbus/log/run
#!/bin/sh
exec 2>&1; set -e

[ -d /var/log/dbus ] || install -dm 755 /var/log/dbus

exec svlogd -tt /var/log/dbus
```

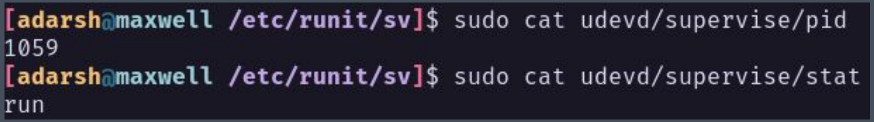
Example: udevd



```
[adarsh@maxwell /etc/runit/sv]$ sudo tree udevd
udevd
├── finish
├── run
└── supervise
    ├── control
    ├── lock
    ├── ok
    ├── pid
    ├── stat
    └── status
```



```
[adarsh@maxwell /etc/runit/sv]$ bat udevd/finish
       File: udevd/finish
       #!/bin/sh
       [ -S /run/udev/control ] && rm -f /run/udev/control
```



```
[adarsh@maxwell /etc/runit/sv]$ sudo cat udevd/supervise/pid
1059
[adarsh@maxwell /etc/runit/sv]$ sudo cat udevd/supervise/stat
run
```

Service Management

- Enabling:
```
ln -s /etc/runit/sv/dbus /run/runit/service/
```
- Disabling:
```
rm /run/runit/service
```
- Stop on boot:
```
touch /etc/runit/sv/dbus/down
```
- Start on boot:
```
rm /etc/runit/sv/dbus/down
```

Service Management

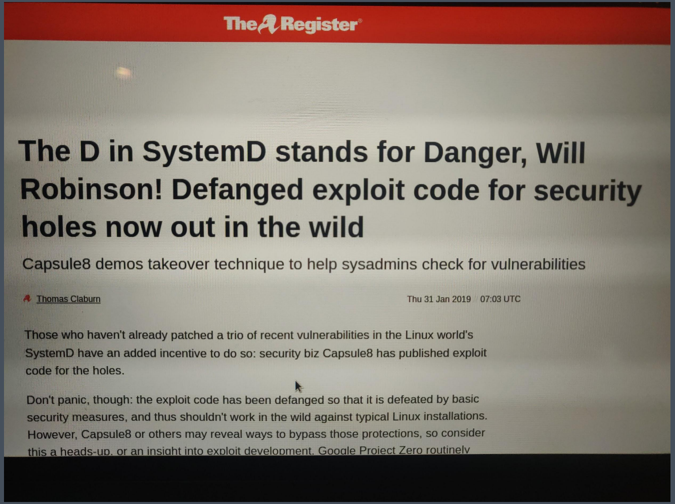
- Start: `sv up dbus`
- Stop: `sv down dbus`
- Run Once: `sv once dbus`
- Restart: `sv status dbus`
- Status: `sv status dbus`
- Etc...!

Minimal Philosophy

- Always prefer minimal, simple software
  - Readable
  - Bug Free
  - Simple
  - Unix Philosophy:
    Good at one thing and one thing only
- Also systemd is bad
- Like really bad

Like Really Bad (not OCF endorsed msg)

- https://artixlinux.org/faq.php
- https://chiefio.wordpress.com/2016/05/18/systemd-it-keeps-getting-worse/
- https://www.theregister.com/2019/01/31/systemd_exploit/



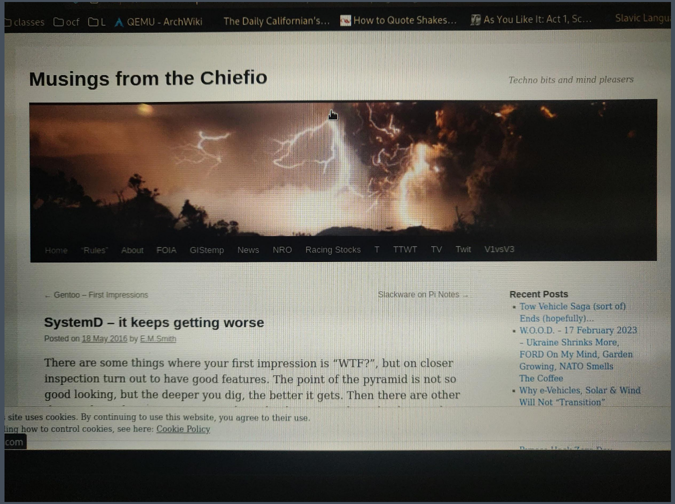
The D in SystemD stands for Danger, Will Robinson! Defanged exploit code for security holes now out in the wild

Capsule8 demos takeover technique to help sysadmins check for vulnerabilities

Thomas Claburn   Thu 31 Jan 2019 07:03 UTC

Those who haven't already patched a trio of recent vulnerabilities in the Linux world's SystemD have an added incentive to do so: security biz Capsule8 has published exploit code for the holes.

Don't panic, though: the exploit code has been defanged so that it is defeated by basic security measures, and thus shouldn't work in the wild against typical Linux installations. However, Capsule8 or others may reveal ways to bypass those protections, so consider this a heads-up, or an insight into exploit development. Google Project Zero routinely



Musings from the Chiefio

Techno bits and mind pleasers

← Gentoo – First Impressions   Slackware on Pi Notes →

SystemD – it keeps getting worse

Posted on 18 May 2016 by E.M.Smith

There are some things where your first impression is "WTF?", but on closer inspection turn out to have good features. The point of the pyramid is not so good looking, but the deeper you dig, the better it gets. Then there are other

Recent Posts
- Tow Vehicle Saga (sort of) Ends (hopefully)...
- W.O.O.D. - 17 February 2023 - Ukraine Shrinks More, FORD On My Mind, Garden Growing, NATO Smells The Coffee
- Why e-Vehicles, Solar & Wind Will Not "Transition"

site uses cookies. By continuing to use this website, you agree to their use.
ling how to control cookies, see here: Cookie Policy

com