Systems 3

OS Design

Marcel Waldvogel

(Handout)

Department of Computer and Information Science
University of Konstanz

Winter 2019/2020

Who is ~~individual~~ process #1?

Photo by The Circus (CC BY 3.0)

Chapter Goals

-

Boot process

Basic overview of the Linux boot process.

Actor	Actions
BIOS/UEFI	test hardware, execute Master Boot Record
MBR	load and execute Grand Unified Bootloader
GRUB	load kernel and initrd
Kernel	mount root file system and start init
Init	determine run level and start
Runlevel	start various services

Daemons

A daemon is a process that

- runs in background (detached from user session)
- ends traditionally with d (e.g. `syslogd`, `sshd`)
- is often a child of the init process

Daemons in C

To create a daemon in C on a Linux system the following steps have to be executed:

1. Fork and exit parent to run in background
2. Create a new session to detach from controlling terminal (`Ctrl-C`)
3. Handle signals
4. Fork and exit session leader (never have a controlling terminal again)
5. Change file mode mask (optional)
6. Change working directory (optional)
7. Close all open file descriptors
8. If privileged, do privileged operations now and then drop privileges
9. Prepare logging (see `syslog(3)`)

```
pid_t pid;

// Fork off the parent process
pid = fork();

// An error occurred
if (pid < 0)
   exit(EXIT_FAILURE);

// Success: Let the parent terminate
if (pid > 0)
   exit(EXIT_SUCCESS);

// Success: The child process becomes
// the session leader
if (setsid() < 0)
   exit(EXIT_FAILURE);

// Catch, ignore and handle signals
signal(SIGCHLD, SIG_IGN);
signal(SIGHUP, SIG_IGN);
// Fork again
pid = fork();

// An error occurred
if (pid < 0)
   exit(EXIT_FAILURE);

// Success: Let the parent terminate
if (pid > 0)
   exit(EXIT_SUCCESS);

// Set new file permissions
umask(0);

// (Change the working directory)
chdir("/");

// Close all open file descriptors
for (int x = sysconf(_SC_OPEN_MAX);
      x >= 0; x--)
   close(x);
```

Example taken from https://github.com/pasce/daemon-skeleton-linux-c

Daemons in C

Of course you could also just use `daemon(3)` ;-)

Init Systems

Daemons are rarely started manually, but instead by an init system, e.g.

- System V
- SystemD
- Upstart
- OpenRC
- runit

History

1. `/etc/rc`
2. `/etc/rc?.d/[SK]*`
3. Dependency graphs

Controlling with systemd

Systemd can be easily controlled by systemctl(1). For example to get the status of a service just call:

```
$ systemctl status sshd
-> ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2019-12-09 11:19:22 CET; 3 days ago
Docs: man:sshd(8)
man:sshd_config(5)
Process: 1505 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 1542 (sshd)
Tasks: 1 (limit: 4915)
Memory: 4.1M
CGroup: /system.slice/ssh.service
  1542 /usr/sbin/sshd -D
```

Creating a systemd service

To create your own service, just add a file like `webserver.service` to `/etc/systemd/system/` with the following content:

```
[Unit]
Description=Awesome webserver
After=network.target
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=always
RestartSec=1
User=www
ExecStart=/opt/webserver/server

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

Question: Why is the file located in `/etc`?

Filesystem Hierarchy Standard (FHS)[1]

The Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux distributions.

/bin	binaries for all users
/boot	boot loader files
/dev	device files
/etc	System-wide configuration files
/lib	Libraries for binaries in /bin and /sbin
/media	Mount point for removable media
/srv	Data served by the system
/tmp	Temporary files
/usr	User System Resources
/var	Variable data

[1] http://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

mount(8) and umount(8)

The mount command serves to attach the filesystem found on some device to the big file tree.

1. Find your drive
   ```
   lsblk
   ```
2. Create your mount point
   ```
   sudo mkdir /media/my-usb-drive
   ```
3. Mount your device
   ```
   sudo mount /dev/sdb1 /media/my-usb-drive
   ```
4. Eject your device
   ```
   sudo umount /media/my-usb-drive
   ```
5. Delete your mount point
   ```
   sudo rmdir /media/my-usb-drive
   ```

Process information pseudo-filesystem

The proc(5) filesystem is a pseudo-filesystem which provides an interface to kernel data structures.

Some examples are:

/proc/uptime seconds since kernel was booted

/proc/meminfo summary of memory usage

/proc/cpuinfo information about the CPU

/proc/cmdline kernel boot options

/proc/PID**/cmdline** command that originally started the process

/proc/PID**/fd** containing all file descriptors

Monitoring

For simple open source tools for Linux system monitoring:

- top
- atop[2]
- htop[3]
- glances[4]

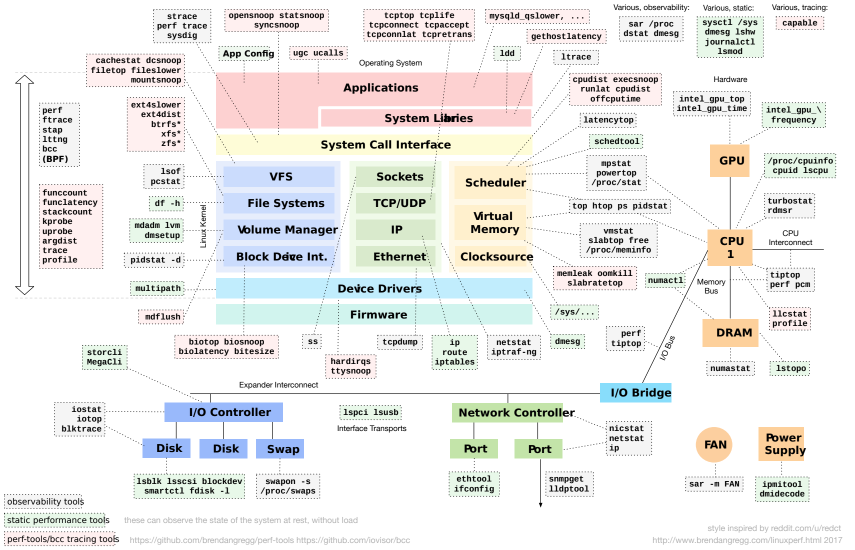
When managing multiple systems, use tools such as Nagios.

[2] https://www.atoptool.nl

[3] https://hisham.hm/htop/

[4] https://nicolargo.github.io/glances/

Linux Performance Tools



Various, observability: sar /proc dstat dmesg

Various, static: sysctl /sys dmesg lshw journalctl lsmod

Various, tracing: capable

Operating System

Hardware

Applications

System Libraries

System Call Interface

Linux Kernel

VFS

File Systems

Volume Manager

Block Device Int.

Sockets

TCP/UDP

IP

Ethernet

Scheduler

Virtual Memory

Clocksource

Device Drivers

Firmware

GPU

CPU 1

CPU Interconnect

Memory Bus

DRAM

I/O Bus

I/O Bridge

Expander Interconnect

I/O Controller

Disk

Disk

Swap

Interface Transports

Network Controller

Port

Port

FAN

Power Supply

strace perf trace sysdig

opensnoop statsnoop syncsnoop

tcptop tcplife tcpconnect tcpaccept tcpconnlat tcpretrans

mysqld_qslower, ...

gethostlatency

App Config

ugc ucalls

ldd

ltrace

cachestat dcsnoop filetop fileslower mountsnoop

ext4slower ext4dist btrfs* xfs* zfs*

perf ftrace stap lttng bcc (BPF)

funccount funclatency stackcount kprobe uprobe argdist trace profile

lsof pcstat

df -h

mdadm lvm dmsetup

pidstat -d

multipath

mdflush

cpudist execsnoop runlat cpudist offcputime

latencytop

schedtool

mpstat powertop /proc/stat

top htop ps pidstat

vmstat slabtop free /proc/meminfo

memleak oomkill slabratetop

/sys/...

dmesg

numactl

intel_gpu_top intel_gpu_time

intel_gpu_\ frequency

/proc/cpuinfo cpuid lscpu

turbostat rdmsr

tiptop perf pcm

llcstat profile

perf tiptop

numastat

lstopo

biotop biosnoop biolatency bitesize

ss

hardirqs ttysnoop

tcpdump

ip route iptables

netstat iptraf-ng

storcli MegaCli

iostat iotop blktrace

lspci lsusb

lsblk lsscsi blockdev smartctl fdisk -l

swapon -s /proc/swaps

nicstat netstat ip

ethtool ifconfig

snmpget lldptool

sar -m FAN

ipmitool dmidecode

observability tools

static performance tools

perf-tools/bcc tracing tools

these can observe the state of the system at rest, without load

https://github.com/brendangregg/perf-tools https://github.com/iovisor/bcc

style inspired by reddit.com/u/redct

http://www.brendangregg.com/linuxperf.html 2017

Brendan Greg (CC BY-SA 4.0)