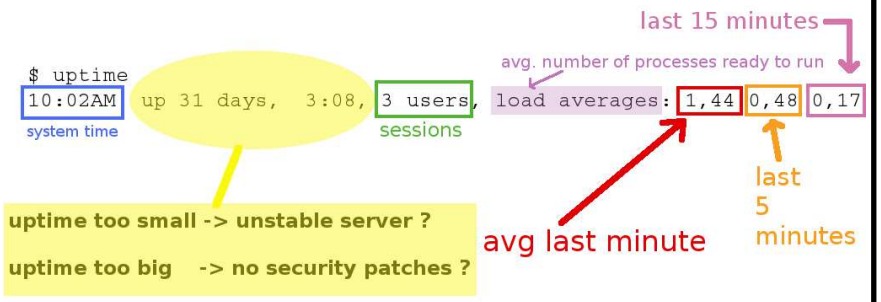
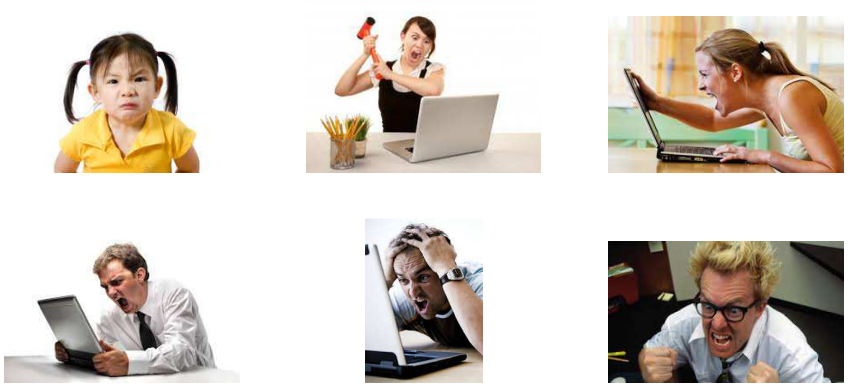
System Up and Running

Load Average: How Busy the System Is



We are now going to shut down the system



System Halt (1)

the command `shutdown` halts the system

this command is reserved to the super–user

- halt with `shutdown -h` (`-p` power off)
- reboot with `shutdown -r`
- shutdown requires a time (when to shutdown)
- shutdown notifies all users via the `wall` command

Examples:

- `shutdown -h 11:15`
- `shutdown -r +20`
- `shutdown -c` (Linux: cancel running shutdown)

System Halt, Respect Your Users

not immediately

not throwing out users

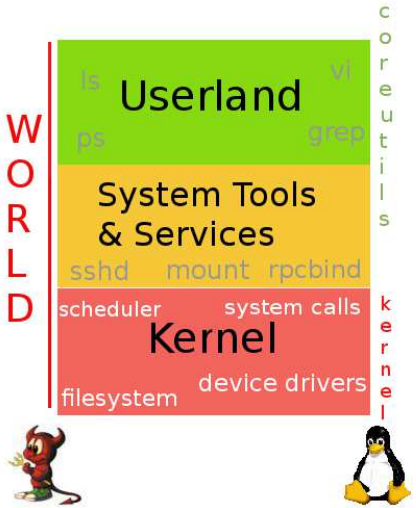
not, if load > 0

⇝make sure: no users, no processes, advance notice

System Halt (2)

- kills all processes
  - first per `TERM` signal
  - then per `KILL` signal
- writes all buffered data to disk (`sync`)

8. Kernel

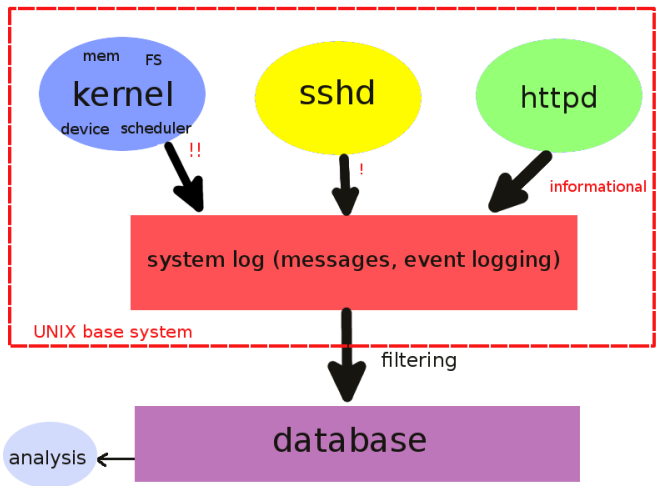


Installing a New Kernel

Usually **not necessary**, except you want

- install security patches
- faster boot-up
- less memory usage
- support for extra hardware components

9. System Logging



Nuclear World Example

International Atomic Energy Agency (IAEA)

Chernobyl, Fukushima etc

Priorities: how important is the message

Facility: which subsystem

Configuration: where should the message be delivered

the log file is the newspaper

Nuclear World: INES scale

```
Level 7: Major accident
         Chernobyl, 26 April 1986
         Fukushima, 11 March 2011
Level 6: Serious accident
         Kyshtym, Mayak, Soviet Union, 29 September 1957
Level 5: Accident with wider consequences
         Harrisburg, Pennsylvania, 28 March 1979
         Goiânia, Brazil, 13 September 1987
Level 4: Accident with local consequences
         Tokaimura, Japan, 1999
Level 3: Serious incident
Level 2: Incident
         Cattenom, 18. Januar 2012
Level 1: Anomaly
Level 0: Deviation
```


Auditing

Auditing = the ability to tell **when who** *did* **what** *to* **what**

Example:

- on **Monday, June 1st, 2014**,
- user `video-wizard`
- did **overflow**
- the `/home` filesystem

Resource: system wide log file

(extension: network wide log file)

How to Write to the System Log (1)

`open()`, `fopen()` ? No! (Do not even think about it.)

Assume two processes writing simultaneously.

Serialization needed!

Assume you want to store the logs somewhere else.

Configurability needed!

How to Write to the System Log (2)

Solution: a special process, called `syslogd` *(syslog daemon)*

- serializes write requests
- can be configured in various ways
- may be reached over a network
- is supported by the C library (`syslog(3)`)

System Messages: Facility

which **subsystem** causes the message

- Kernel
- Mail System
- System Daemons
- Printer System

⋮

-

Keywords:

auth, authpriv, console, cron, daemon, ftp, kern, lpr, mail, mark, news, ntp, security, syslog, user, uucp, local0 through local7

System Messages: Priority

how **important** is the message

value	constant	name	description
0	`LOG_EMERG`	emergency	system is unusable
1	`LOG_ALERT`	alert	action must be taken immediately
2	`LOG_CRIT`	critical	critical conditions (probably hardware)
3	`LOG_ERR`	error	error conditions
4	`LOG_WARN`	warning	warning conditions
5	`LOG_NOTICE`	notice	normal but significant condition
6	`LOG_INFO`	info	informational message
7	`LOG_DEBUG`	debug	debug-level message

Keywords:

emerg, alert, crit, err, warning, notice, info, debug

10. Network

Network Configuration

subtle differences between UNIX systems

1. Network Interface Card (NIC)
   - must be recognized by the kernel

   ⇝ kernel configuration

   - is then available under a name like
     - *fxp0, em0, vr0, ...* depends on driver (BSD)
     - *eth0, eth1, ...* (Linux)
2. IP address (broadcast, netmask)
   - must be configured via `ifconfig`
   - example (Linux/Solaris/BSD)

```
ifconfig eth0 134.96.216.97 netmask 255.255.255.0 \
         broadcast 134.96.216.255
```


3. Routing
   - adding a default gateway
   - example `route add default gw 134.96.216.1` (Linux)
   - example `route add default 134.96.216.1` (BSD)
4. DNS
   - add entry `nameserver` in `/etc/resolv.conf`
   - add entry `search` in `/etc/resolv.conf`
   - use DNS diagnosis tools `dig` and `host`
   - do **not** use `nslookup`

DNS Records

there are different *types* of addresses

- A records: request *host*, reply *IP*

```
$ dig +short isl-s-01.htw-saarland.de
134.96.216.91
```

- MX records: request *mail-domain*, reply *mail server* (with prio)

```
$ dig +short htw-saarland.de MX
80 m-relay2.rz.uni-saarland.de.
90 m-relay3.rz.uni-saarland.de.
20 m-relay.htw-saarland.de.
80 m-relay.rz.uni-saarland.de.
```

- SOA records: request *domain*, reply *administrative parameters*

```
$ dig +short htw-saarland.de SOA
ns.rz.uni-saarland.de. Margit\.Meyer.htw-saarland.de. ...
```

- NS records: request *domain*, reply *name-server*

```
$ dig +short htw-saarland.de NS
ns.rz.uni-saarland.de.
ns1.htw-saarland.de.
ns.htw-saarland.de.
ws-ber1.win-ip.dfn.de.
```

- PTR records: reverse DNS lookup

```
$ dig +short 81.216.96.134.in-addr.arpa ptr
isl-c-01.htw-saarland.de.
```

- CNAME records: alias names

```
$ dig +short www.htw-saarland.de cname
www-portal.htw-saarland.de.
```

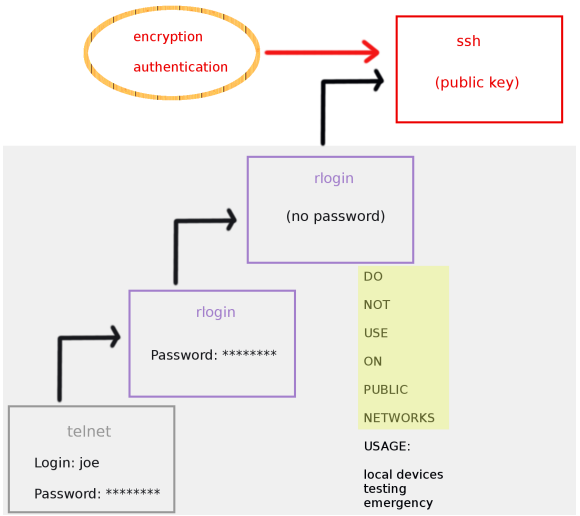

Root Servers

```
$ dig +short . NS | sort
a.root-servers.net.
b.root-servers.net.
c.root-servers.net.
d.root-servers.net.
e.root-servers.net.
f.root-servers.net.
g.root-servers.net.
h.root-servers.net.
i.root-servers.net.
j.root-servers.net.
k.root-servers.net.
l.root-servers.net.
m.root-servers.net.
```



```
$ for x in ‘dig +short . NS | sort‘; do
echo $x" "‘dig +short $x‘ ;
  done
a.root-servers.net. 198.41.0.4
b.root-servers.net. 192.228.79.201
c.root-servers.net. 192.33.4.12
d.root-servers.net. 199.7.91.13
e.root-servers.net. 192.203.230.10
f.root-servers.net. 192.5.5.241
g.root-servers.net. 192.112.36.4
h.root-servers.net. 128.63.2.53
i.root-servers.net. 192.36.148.17
j.root-servers.net. 192.58.128.30
k.root-servers.net. 193.0.14.129
l.root-servers.net. 199.7.83.42
m.root-servers.net. 202.12.27.33
```

Login via Network



Login via Network

- `telnet` : a virtual terminal, first form of remote login
- `rlogin` : simplified remote login
- `ssh` : encrypted, authenticated remote login

Problems of rlogin/telnet

the network connection is

- unauthenticated
  - is the target host genuine?
  - is the connecting host genuine?
- unencrypted
  everybody sniffing on the wire
  (for example promiscious mode NIC)
  can read passwords, transmitted data, ...

Public Key Cryptography (1)

solves both problems

every user U has

- a public key P_U
- a secret (private) key S_U

Example: To send a message m to Alice, Bob must compute

$$m' = E(P_{Alice}, m)$$

Alice decrypts m' by computing

$$D(S_{Alice}, m')$$

Public Key Cryptography (2)

The encryption function $E()$

and

the decryption function $D()$

are public.

$\rightsquigarrow$ it must be impossible to compute S_U from P_U

Public Key Cryptography (3)

There are three algorithms which are more or less used in PKC:

- RSA (based on factoring, 1978)
- DSA (based on discrete logs in Galois fields, 1985)
- ECDSA (based on discrete logs on elliptic curves, 1989)

World records for breaking these schemes:

- factoring 728 bits (220 decimal digits) in 2016 (Inria, FR)
- factoring 768 bits special number in 2010 (Uni Bonn)
- factoring 663 bits (200 decimal digits) in 2005 (Uni Bonn)
- discrete log in $GF(p)$, p with 768 bits in June, 2016 (U Leipzig)
- discrete log in $GF(p)$, p with 596 bits in 2014 (Loria, FR)
- DL on EC over $GF(p)$, p with 113 bits in 2015

Recommended key sizes for these schemes

- RSA 2048 bits
- DSA 2048 bits
- ECDSA 160 bits

ssh

Client side: `ssh`
Server side: `sshd`

Implementation: OpenSSH and others

Properties

- authenticated
  - connecting host must prove its identity (public key)
  - accepting host must prove its identity (public key)
  - user must prove his identity (public key, password)
- encrypted connection (especially no plain text passwords)

Public Key authentication:

```
$ ssh isl-l-01
Enter passphrase for key '/home/dweber/.ssh/id_dsa':
Last login: Mon Jul 16 15:46:13 2012 from stl-s-studwork.htw-saarland.de
FreeBSD 9.0-STABLE (ISL-S-01) #0: Wed Jun 13 01:32:10 CEST 2012
```


12. Firewalling

Keep the bad guys out,

and let the good guys in.

Firewalls have rules to *refuse* IP packets,

and *accept* them.

First–match–logic: `iptables`, `ipfw`

- rule after rule
- until a rule (positively or negatively) matches

Last–match–logic: `pf`, `ipfilter`

- rule after rule
- the last rule that matches determines target

⇝firewalling is part of TCP/IP code, therefore part of kernel

ipfw: the FreeBSD Way of Firewalling

enable firewalling in `/etc/rc.conf`

```
firewall_enable="YES"
firewall_type="client"
```

add rules to `/etc/rc.firewall` for the chosen firewall type

- open – no rules
- client – no servers on this machine
- simple – basic server configuration (DNS, HTTP, NTP)
- closed – all IP services disabled, except loopback

Good Luck while Defending Against Hackers

`http://www.claybennett.com/pages/info_superhighway.html`