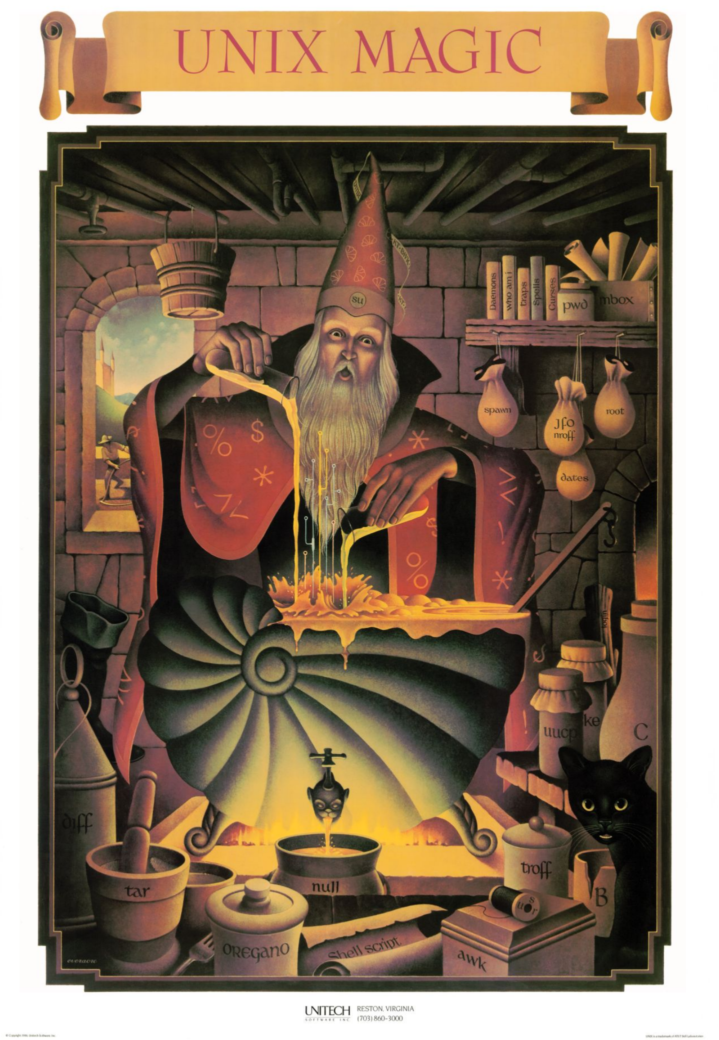
CS 447/647

COMPUTER SYSTEMS ADMINISTRATION



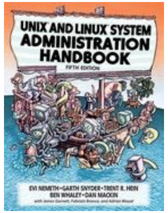
What do we manage?

- HPC clusters
- Standalone compute servers
- Storage servers
- Mail servers
    - cse.unr.edu
    - engr.unr.edu
- Jump Hosts
- VMs & containers
- Remote desktop servers
- Public computer labs
- Graduate research labs

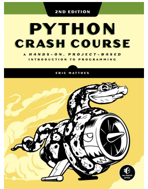
Syllabus

Book(s)

UNIX and Linux System Administration Handbook, 5th Edition

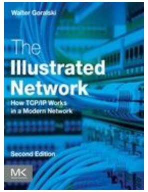


Python Crash Course, 2nd Edition



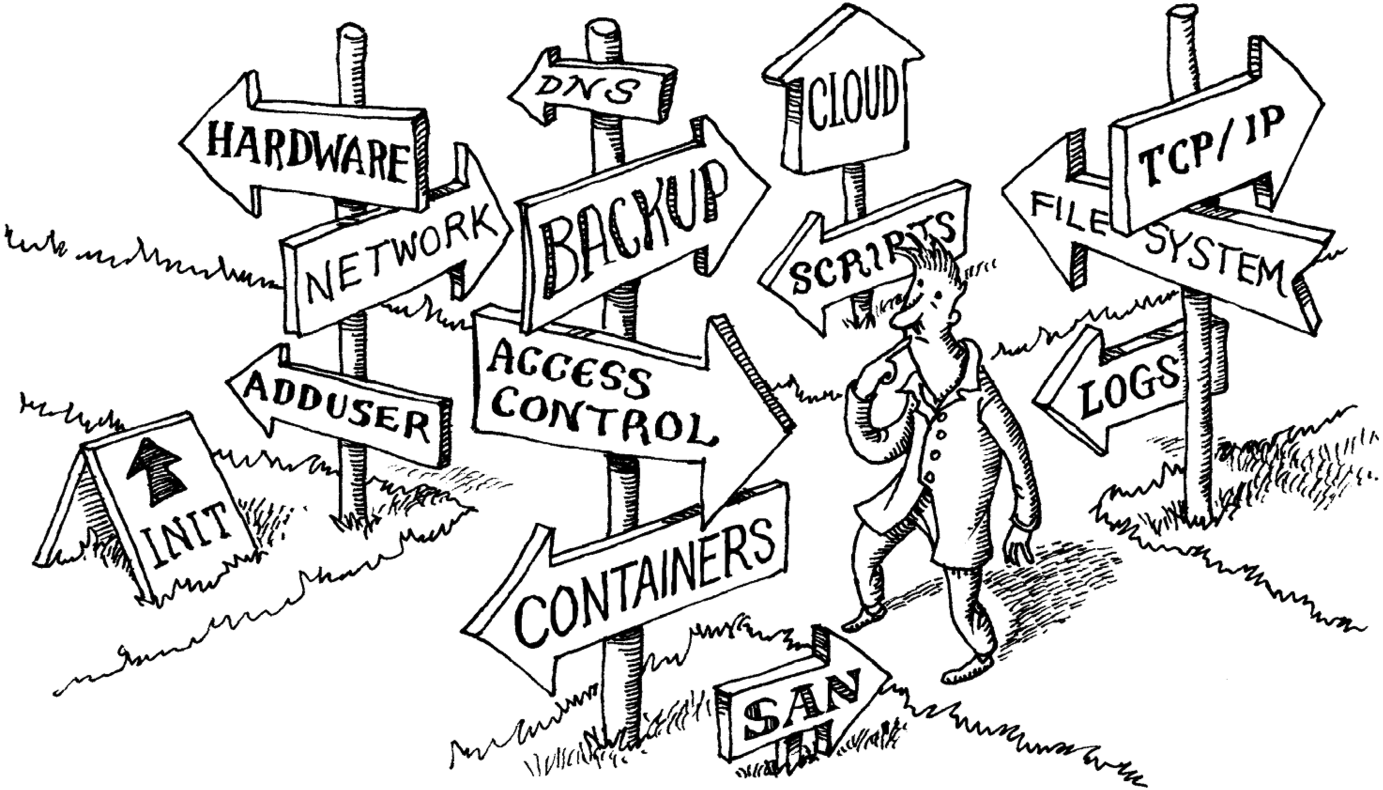
Computer Networks: A Systems Approach
https://book.systemsapproach.org/index.html

The Illustrated Network, 2nd Edition



https://guides.library.unr.edu/safari

What is Systems Administration?



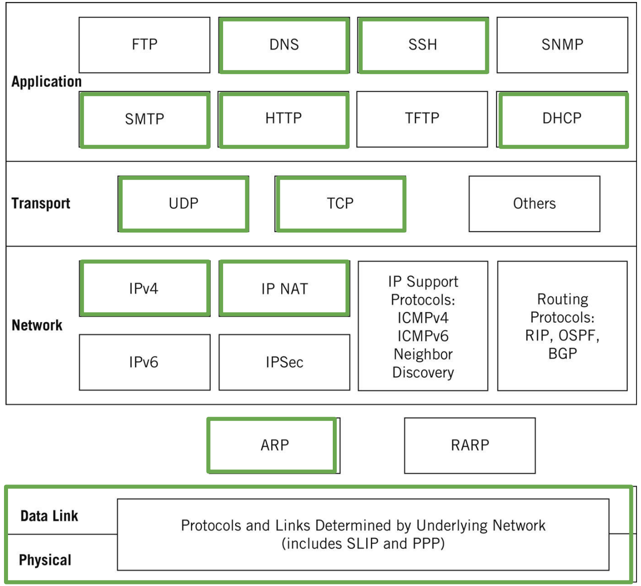
Systems Administration Tasks

- Controlling Access
- ***Adding Hardware***
- Automating Tasks
- Overseeing Backups
- Installing and Upgrading Software
- Managing Downtime
- Monitoring
- Troubleshooting
- Maintaining Documentation
- Performance Tuning/Optimization

- Developing Site Policies
- ***Working with vendors***
- Fire Fighting (On-Call)
- Writing Glue Code

What will you gain from this course?

- Repertoire of modular and composable tools
- Knowledge of Debian Linux distro internals
- Hands-on TCP/IP experience
- Experience with core Application Layer services
    - Internet Protocol Suite
- Knowledge of virtualization strategies



Application

FTP

DNS

SSH

SNMP

SMTP

HTTP

TFTP

DHCP

Transport

UDP

TCP

Others

Network

IPv4

IP NAT

IP Support Protocols: ICMPv4 ICMPv6 Neighbor Discovery

Routing Protocols: RIP, OSPF, BGP

IPv6

IPSec

ARP

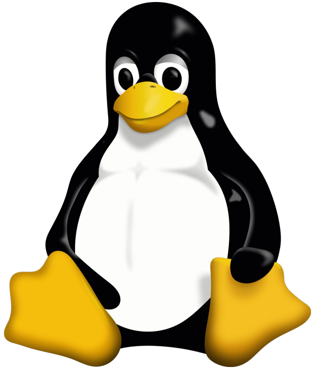
RARP

Data Link

Physical

Protocols and Links Determined by Underlying Network (includes SLIP and PPP)

Operating Systems



- Linux - Unix-like, Open-source Operating System

  Why? **Durability.**

- Windows - Microsoft's Graphical Operating System

  Why? **Any large organisation with a significant number of Windows users uses Windows Server/Technology.**

GNU\Linux Distributions

- Linux kernel + GNU software + other software = distribution
  - `ls, find, bash, parted`
- Allows easy installation and updating of the operating system.
- Variety = freedom to choose from hundreds. Each distribution has a niche.
- Major categories by package management system:
  - Redhat-derived: RPM packages RHEL, Fedora, CentOS, Oracle L., SUSE, etc.
  - **Debian**-derived: APT system, **deb** packages Debian, Ubuntu, Mint, etc.
  - Source based: portage, compiled with box-specific optimizations– Gentoo, Sabayon, etc.
- Timeline graphs: https://github.com/FabioLolix/LinuxTimeline

Unix Philosophy

“Unix has a culture; it has a distinctive art of programming; and it carries with it a powerful design philosophy. Understanding these traditions will help you build better software, even if you're developing for a non-Unix platform.”

Art of Unix Programming - Eric Raymond http://www.catb.org/~esr/writings/taoup/html/ch01s06.html

Unix Philosophy

- Douglas McIlroy: “**Write programs that do one thing and do it well**. Write programs to work together. Write programs to handle text streams, which is a universal interface.” Everything in UNIX is a file, preferably a text file.
- Eric Raymond: The “KISS” principle, from The Art of Unix Programming
  - Modularity – simple parts connected by clean interfaces
  - Clarity – better than cleverness
  - Simplicity – add complexity only when you must
  - Transparency – make debugging easier
  - Robustness – stems from transparency and simplicity
  - **Silence – when a program has nothing to say, it should keep silent**
  - Repair – when program must fail, fail loudly and as soon as possible
- "Those who don't understand Unix are condemned to reinvent it, poorly." – Henry Spencer

Unix Philosophy

i) **Make each program do one thing well.** To do a new job, build afresh rather than complicate old programs by adding new features.

(ii) **Expect the output of every program to become the input to another**, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.

(iii) **Design and build software, even operating systems, to be tried early,** ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.*

(iv) **Use tools in preference to unskilled help to lighten a programming task,** even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

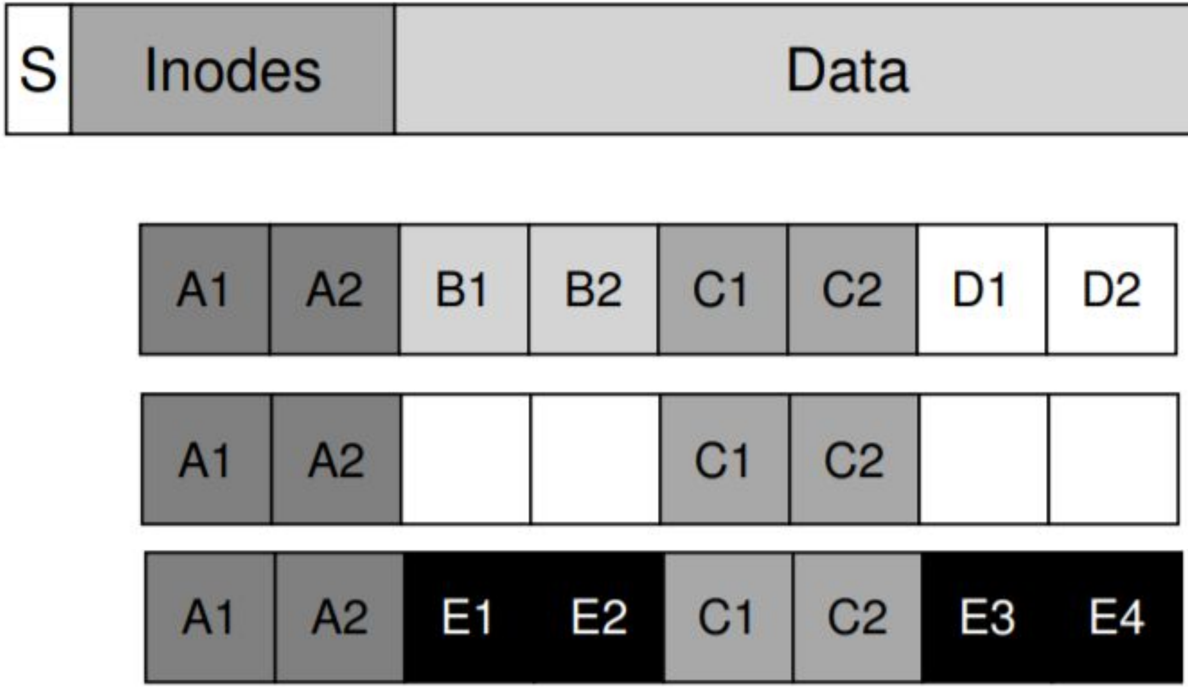
“I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)”

Linus Torvalds, age 21, 1991

“When in doubt, use brute force.”

— Ken Thompson

Brute Force Example - Unix Filesystem



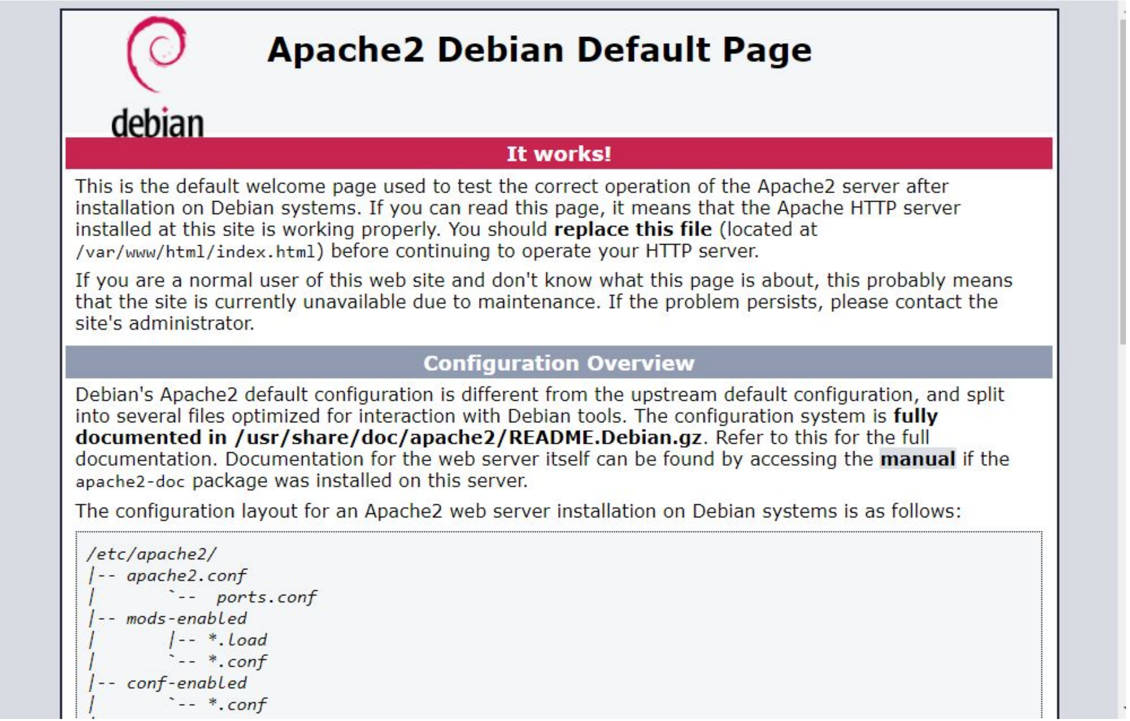
“Operating Systems: Three Easy Pieces” (Chapter: LOCALITY AND THE FAST FILE SYSTEM) by Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau

Rule of Modularity: Write simple parts connected by clean interfaces.

Debugging dominates development time, and getting a working system out the door is usually less a result of brilliant design than it is of managing not to trip over your own feet too many times.

Rule of Modularity Example

Apache2 - HTTP Server (a patchy web server)



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

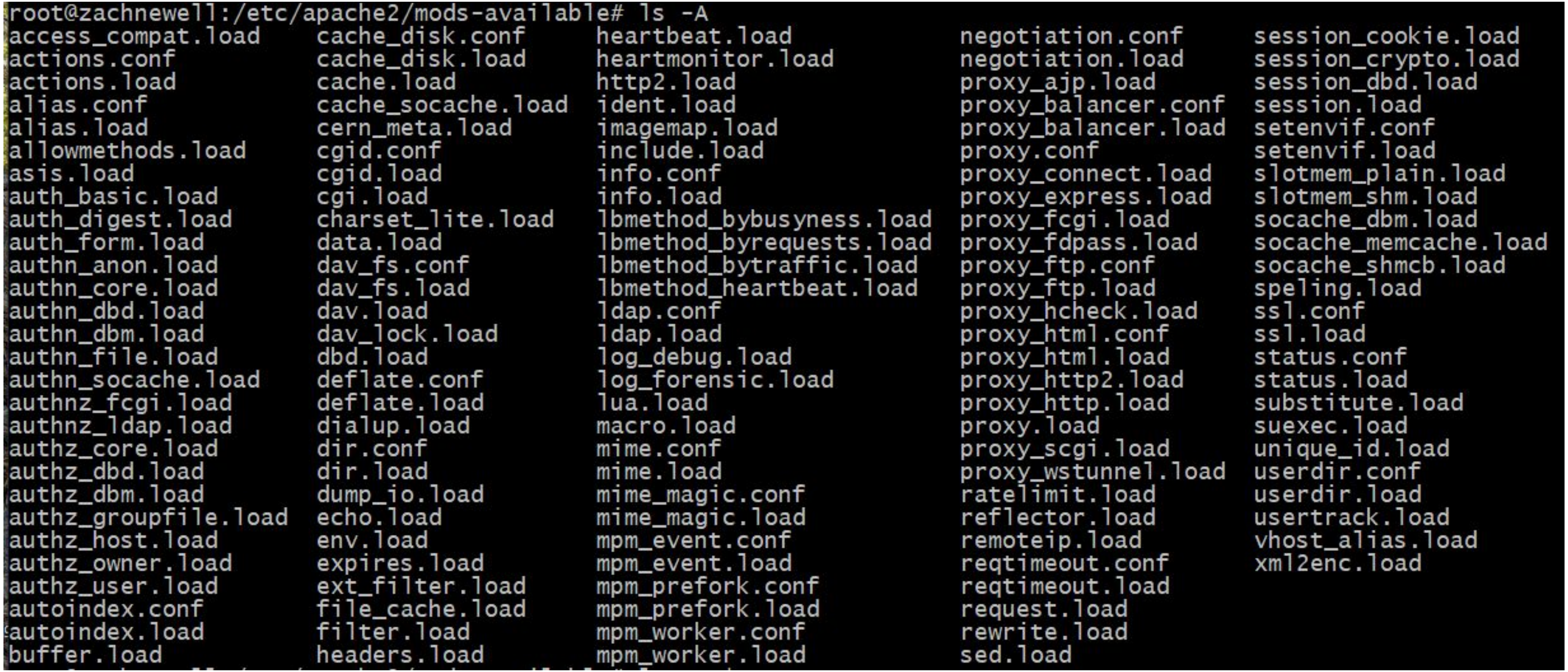
Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
```

Rule of Modularity



```
root@zachnewell:/etc/apache2/mods-available# ls -A
access_compat.load    cache_disk.conf     heartbeat.load            negotiation.conf     session_cookie.load
actions.conf          cache_disk.load     heartmonitor.load         negotiation.load     session_crypto.load
actions.load          cache.load          http2.load                proxy_ajp.load       session_dbd.load
alias.conf            cache_socache.load  ident.load                proxy_balancer.conf  session.load
alias.load            cern_meta.load      imagemap.load             proxy_balancer.load  setenvif.conf
allowmethods.load     cgid.conf           include.load              proxy.conf           setenvif.load
asis.load             cgid.load           info.conf                 proxy_connect.load   slotmem_plain.load
auth_basic.load       cgi.load            info.load                 proxy_express.load   slotmem_shm.load
auth_digest.load      charset_lite.load   lbmethod_bybusyness.load  proxy_fcgi.load      socache_dbm.load
auth_form.load        data.load           lbmethod_byrequests.load  proxy_fdpass.load    socache_memcache.load
authn_anon.load       dav_fs.conf         lbmethod_bytraffic.load   proxy_ftp.conf       socache_shmcb.load
authn_core.load       dav_fs.load         lbmethod_heartbeat.load   proxy_ftp.load       speling.load
authn_dbd.load        dav.load            ldap.conf                 proxy_hcheck.load    ssl.conf
authn_dbm.load        dav_lock.load       ldap.load                 proxy_html.conf      ssl.load
authn_file.load       dbd.load            log_debug.load            proxy_html.load      status.conf
authn_socache.load    deflate.conf        log_forensic.load         proxy_http2.load     status.load
authnz_fcgi.load      deflate.load        lua.load                  proxy_http.load      substitute.load
authnz_ldap.load      dialup.load         macro.load                proxy.load           suexec.load
authz_core.load       dir.conf            mime.conf                 proxy_scgi.load      unique_id.load
authz_dbd.load        dir.load            mime.load                 proxy_wstunnel.load  userdir.conf
authz_dbm.load        dump_io.load        mime_magic.conf           ratelimit.load       userdir.load
authz_groupfile.load  echo.load           mime_magic.load           reflector.load       usertrack.load
authz_host.load       env.load            mpm_event.conf            remoteip.load        vhost_alias.load
authz_owner.load      expires.load        mpm_event.load            reqtimeout.conf      xml2enc.load
authz_user.load       ext_filter.load     mpm_prefork.conf          reqtimeout.load
autoindex.conf        file_cache.load     mpm_prefork.load          request.load
autoindex.load        filter.load         mpm_worker.conf           rewrite.load
buffer.load           headers.load        mpm_worker.load           sed.load
```

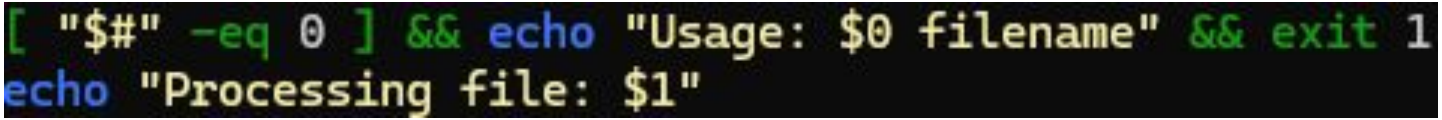
Rule of Clarity: Clarity is better than cleverness.

Because maintenance is so important and so expensive, write programs as if the most important communication they do is not to the computer that executes them but to the human beings who will read and maintain the source code in the future (including yourself).

Buying a small increase in performance with a large increase in the complexity and obscurity of your technique is a bad trade...

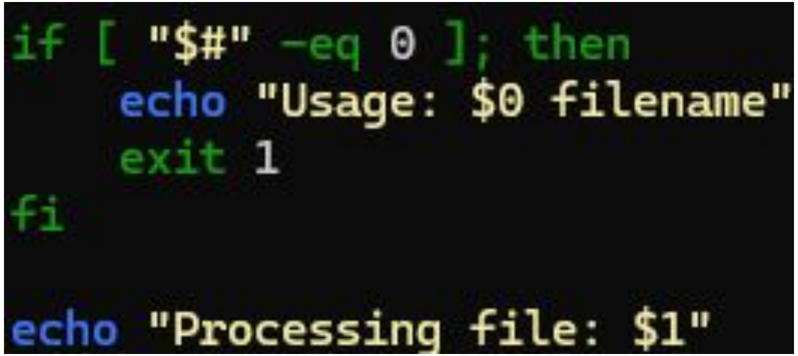
Rule of Clarity Example

Unclear



```
[ "$#" -eq 0 ] && echo "Usage: $0 filename" && exit 1
echo "Processing file: $1"
```

Clear



```
if [ "$#" -eq 0 ]; then
    echo "Usage: $0 filename"
    exit 1
fi

echo "Processing file: $1"
```

Rule of Composition: Design programs to be connected with other programs.

Unix tradition strongly encourages writing programs that read and write simple, textual, stream-oriented, device-independent formats. Under classic Unix, as many programs as possible are written as simple filters, which take a simple text stream on input and process it into another simple text stream on output.

Unix culture values code which is useful to other programmers, while Windows culture values code which is useful to non-programmers.

- Joel Spolsky

*nix and Windows

- Unix programmer will create a command-line or text-driven core and occasionally, as an afterthought, build a GUI which drives that core.

- Windows programmer will tend to start with a GUI, and occasionally, as an afterthought, add a scripting language which can automate the operation of the GUI interface.

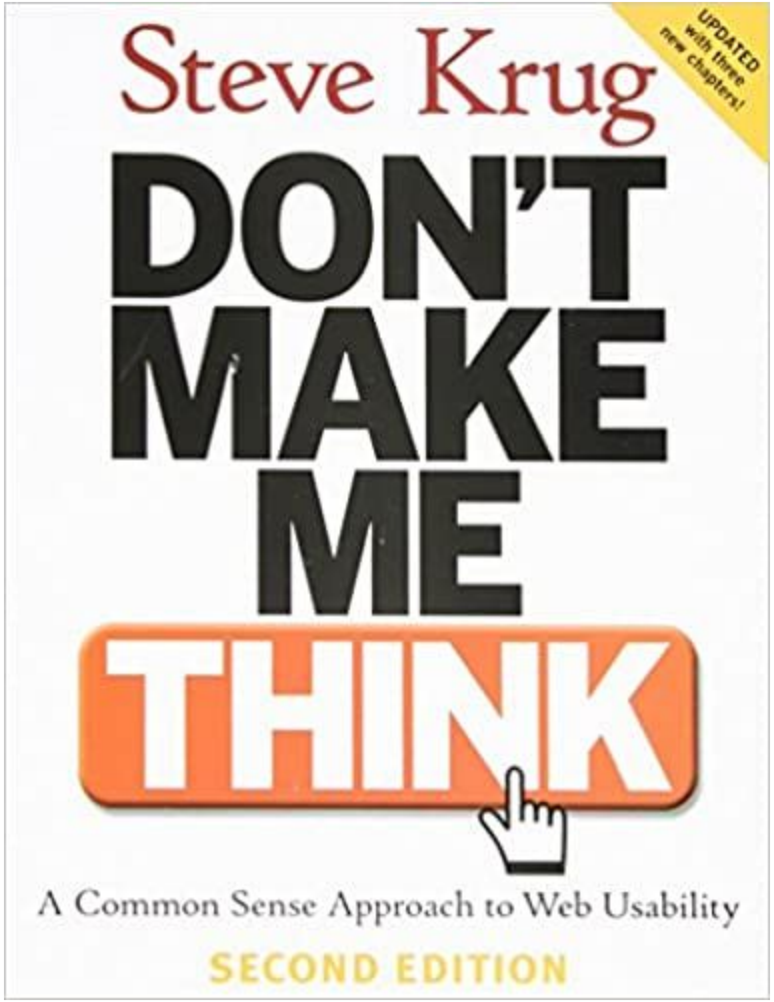
What Is the Operating System's Unifying Idea?

Everything is a file

Contrast to Windows

Don't make me think

Sweet mother of God-
so that's how you
buckle a seat belt!



UPDATED with three new chapters!

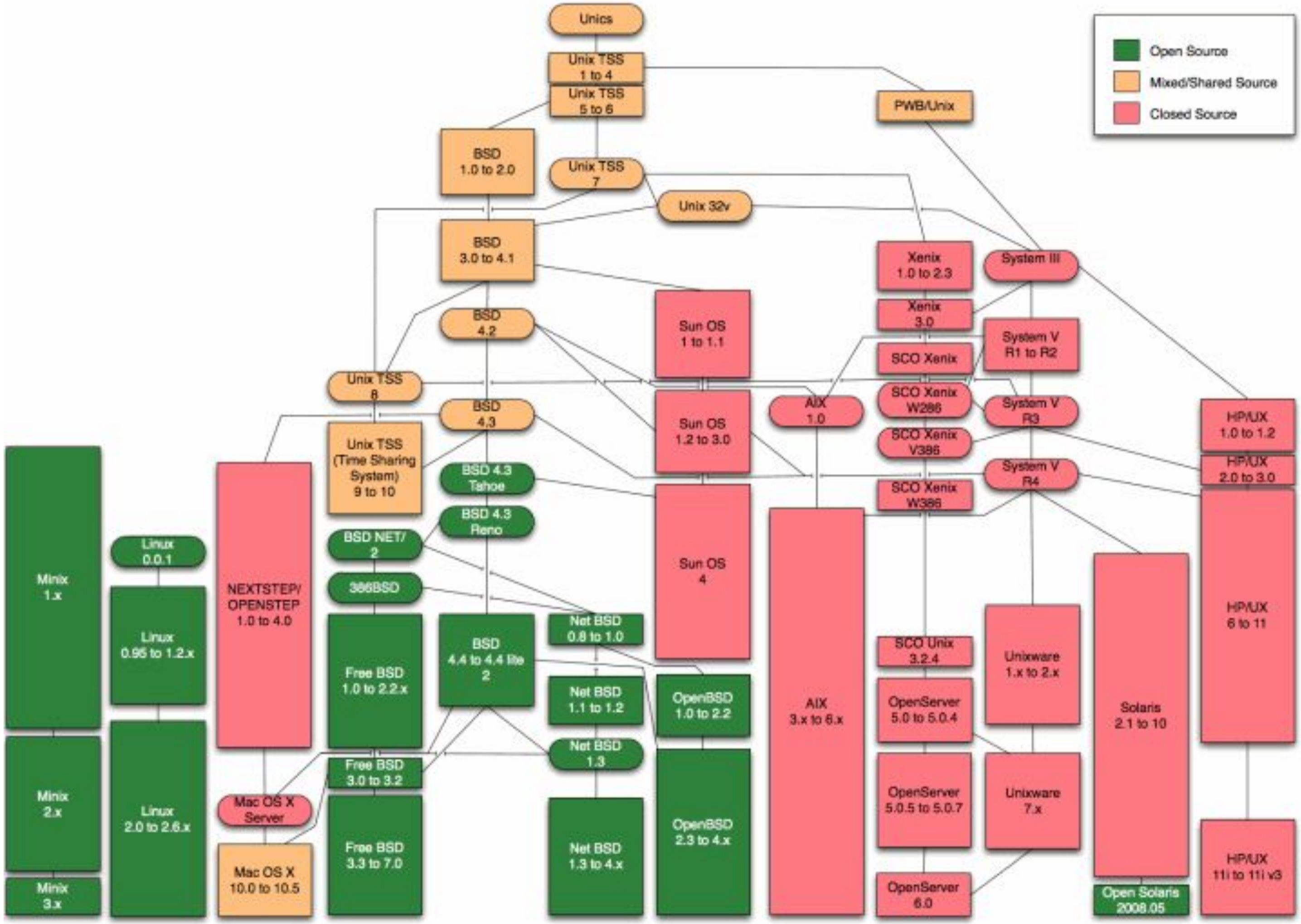
Steve Krug

DON'T MAKE ME THINK

A Common Sense Approach to Web Usability

SECOND EDITION

History



🟩 Open Source

🟧 Mixed/Shared Source

🟥 Closed Source

Unics

Unix TSS 1 to 4

Unix TSS 5 to 6

PWB/Unix

BSD 1.0 to 2.0

Unix TSS 7

Unix 32v

BSD 3.0 to 4.1

Xenix 1.0 to 2.3

System III

BSD 4.2

Sun OS 1 to 1.1

Xenix 3.0

System V R1 to R2

SCO Xenix

Unix TSS 8

BSD 4.3

Sun OS 1.2 to 3.0

AIX 1.0

SCO Xenix W286

System V R3

SCO Xenix V386

HP/UX 1.0 to 1.2

HP/UX 2.0 to 3.0

Unix TSS (Time Sharing System) 9 to 10

BSD 4.3 Tahoe

System V R4

SCO Xenix W386

BSD 4.3 Reno

BSD NET/2

Minix 1.x

Linux 0.0.1

NEXTSTEP/ OPENSTEP 1.0 to 4.0

Sun OS 4

386BSD

Linux 0.95 to 1.2.x

Free BSD 1.0 to 2.2.x

BSD 4.4 to 4.4 lite 2

Net BSD 0.8 to 1.0

AIX 3.x to 6.x

SCO Unix 3.2.4

Unixware 1.x to 2.x

Solaris 2.1 to 10

HP/UX 6 to 11

Net BSD 1.1 to 1.2

OpenBSD 1.0 to 2.2

OpenServer 5.0 to 5.0.4

Minix 2.x

Linux 2.0 to 2.6.x

Mac OS X Server

Free BSD 3.0 to 3.2

Net BSD 1.3

OpenServer 5.0.5 to 5.0.7

Unixware 7.x

Mac OS X 10.0 to 10.5

Free BSD 3.3 to 7.0

Net BSD 1.3 to 4.x

OpenBSD 2.3 to 4.x

HP/UX 11i to 11i v3

Minix 3.x

OpenServer 6.0

Open Solaris 2008.05