Daemons

- init
- cron and atd
- inetd and xinetd
- Kernel daemons
- File service daemons
- Internet daemons
- Time synchronization daemons
- Booting and configuration daemons
- FTP and WWW proxy servers
init

- First process to run after booting
- PID of 1
- Either goes to single user mode or starts scripts to go to multi-user mode
- Runs some version of getty for console and serial logins
cron and atd

- crond runs commands at preset times
- so does atd
  - but can limit when jobs are run (based on load)
inetd and xinetd

- inetd is a daemon that manages other daemons
  - Starts client daemons only when there is work for them
  - Lets them die when their work is complete
- Only works with daemons that provide network services
  - Attaches itself to the network ports used by clients
  - When connection occurs, inetd starts the daemon, and connects standard I/O to the network port
- xinetd is an improved alternative
inetd uses /etc/inetd.conf to determine which ports and daemons to use (along with /etc/services)

```bash
# Sample portions of an /etc/inetd.conf from Solaris
#
ftp    stream tcp6 nowait root /usr/sbin/tcpd  in.ftpd
# telnet stream tcp6 nowait root /usr/sbin/tcpd  in.telnetd
# shell stream tcp  nowait root /usr/sbin/tcpd  in.rshd
shell stream tcp6 nowait root /usr/sbin/tcpd  in.rshd
login stream tcp6 nowait root /usr/sbin/tcpd  in.rlogind
exec stream tcp  nowait root /usr/sbin/tcpd  in.rexecd
exec stream tcp6 nowait root /usr/sbin/tcpd  in.rexecd
talk dgram udp  wait   root /usr/sbin/tcpd  in.talkd
time stream tcp6 nowait root internal
time dgram udp6 wait  root internal
amanda dgram udp  wait backup /opt/amanda/libexec/amandad amandad
```
xinetd

- /etc/xinetd.conf, and can also use a directory with entries like:

```
# default: off
# description: An xinetd internal
# service which echo's characters
# back to clients.
# This is the tcp version.

service echo  
{
  type = INTERNAL
  id = echo-stream
  socket_type = stream
  protocol = tcp
  user = root
  wait = no
  disable = yes
}

# default: off
# description: The talk server
# accepts talk requests for
# chatting with users on other
# systems.

service talk  
{
  disable = yes
  socket_type = dgram
  wait = yes
  user = nobody
  group = tty
  server = /usr/sbin/in.talkd
}
```
# /etc/services file

<table>
<thead>
<tr>
<th>service-name</th>
<th>port/protocol</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcpmux</td>
<td>1/tcp</td>
<td># TCP port service multiplexer</td>
</tr>
<tr>
<td>rje</td>
<td>5/tcp</td>
<td># Remote Job Entry</td>
</tr>
<tr>
<td>rje</td>
<td>5/udp</td>
<td># Remote Job Entry</td>
</tr>
<tr>
<td>echo</td>
<td>7/tcp</td>
<td></td>
</tr>
<tr>
<td>echo</td>
<td>7/udp</td>
<td></td>
</tr>
<tr>
<td>systat</td>
<td>11/tcp</td>
<td>users</td>
</tr>
<tr>
<td>systat</td>
<td>11/udp</td>
<td>users</td>
</tr>
<tr>
<td>daytime</td>
<td>13/tcp</td>
<td></td>
</tr>
<tr>
<td>daytime</td>
<td>13/udp</td>
<td></td>
</tr>
<tr>
<td>qotd</td>
<td>17/tcp</td>
<td>quote</td>
</tr>
<tr>
<td>qotd</td>
<td>17/udp</td>
<td>quote</td>
</tr>
<tr>
<td>ftp-data</td>
<td>20/tcp</td>
<td></td>
</tr>
<tr>
<td>ftp</td>
<td>21/tcp</td>
<td></td>
</tr>
<tr>
<td>ssh</td>
<td>22/tcp</td>
<td># SSH Remote Login Protocol</td>
</tr>
<tr>
<td>telnet</td>
<td>23/tcp</td>
<td></td>
</tr>
<tr>
<td>smtp</td>
<td>25/tcp</td>
<td>mail</td>
</tr>
<tr>
<td>smtp</td>
<td>25/udp</td>
<td>mail</td>
</tr>
</tbody>
</table>
Kernel daemons

- A few parts of the kernel are managed as if they were user processes
  - low PID processes, usually beginning with k
  - keventd, kupdated, klogd, kjournald
- Generally deal with memory management, synchronization of disk caches, and message logging
File service daemons

- `rpc.nfsd`: kernel daemon that serves NFS requests
- `rpc.mountd`: accepts filesystem mount requests
- `amd` and `automount`: mount on demand
- `rpc.lockd` and `rpc.statd`: NFS locking and NFS status
- `rpciod`: caches NFS blocks
- `rpc.rquotad`: serve remote quotas (NFS)
- `smbd`: Windows-compatible file and print services
- `nmbd`: Windows-compatible NetBIOS name service requests
Administrative database daemons

- ypbind: locate NIS servers
- ypserv: NIS server
- rpc.ypxfrd: transfer NIS database
- nscd: name service cache daemon
Internet daemons (1/2)

- talkd: network chat
- sendmail: MTA
- snmpd: remote network management
- rwhod: remote user lists
- vsftpd: very secure ftp daemon
- popper: basic mailbox access
- imapd: more functional mailbox access
- in.rlogind: remote logins
- in.telnetd: uses telnet protocol
Internet daemons (2/2)

- sshd: secure remote logins
- in.rshd: remote command execution
- rsyncd: synchronize files
- routed, gated: maintain routing tables
- named: DNS server
- syslogd: logging server
- in.fingerd: look up users
- httpd: WWW server
- lpd: print spooler
Booting & Configuration Daemons

- dhcpd: dynamic address assignment
- in.tftpd: trivial file transfer server
- rpc.bootparamd: provide info to diskless clients
Time synchronization daemons

- **timed**: synchronize clocks
  - (multiple implementations with same name)
- **ntpd, xntpd**: better implementation
  - more accurate, within a few milliseconds

*We enabled ntpd when we installed CentOS*
FTP servers

- File Transfer Protocol – predated the Web
- Anonymous FTP becoming less common
  - Non-anonymous FTP is a security concern (same as telnet – usernames and passwords in cleartext)
- vsftpd can be run standalone or via inetd
- To limit the security concerns, vsftpd can have authenticated users access their own chrooted space
- Do not make any ftp directories world writable!
  - Your machine becomes a free file server
Web proxies

- A proxy: someone who does something on your behalf

- Uses for web proxies:
  - Access management / filtering / logging
  - Bandwidth and latency reduction through caching
  - Load-spreading mechanism for busy web servers
Web caches (proxy server)

**Goal:** satisfy client request without involving origin server

- User sets browser to access Web via cache
- Browser sends all HTTP requests to cache
  - If object in cache: cache returns object
  - Else cache requests object from origin server, then returns object to client
More about Web caching

- Cache acts as both client and server
- Cache can do up-to-date check using If-modified-since HTTP header
  - Issue: should cache take risk and deliver cached object without checking?
    - Heuristics are used.
- Typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?
- Reduce response time for client request.
- Reduce traffic on an institution’s access link.
- Internet dense with caches enables “poor” content providers to effectively deliver content (that is, it reduces the load on Web servers).