Daemons & inetd

Refs: Chapter 12

Daemons

- A daemon is a process that:
  - runs in the background
  - not associated with any terminal
- Unix systems typically have many daemon processes.

Common Daemons

- Web server (httpd)
- Mail server (sendmail)
- SuperServer (inetd)
- System logging (syslogd)
- Print server (lpd)
- router process (routed, gated)

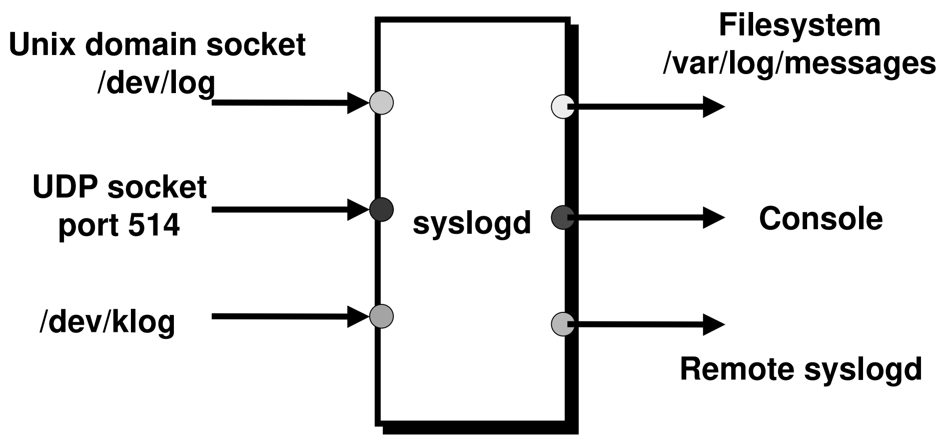
Daemon Output

- No terminal - must use something else:
  - file system
  - central logging facility
- Syslog is often use - provides central repository for system logging.

Syslog service

- **`syslogd`** daemon provides system logging services to clients.
- Simple API for clients (library provided by O.S.).
- Control of logging functions by sysadmin:
  - where messages should go
  - what kinds of messages are important

syslogd



Syslog messages

- Each message has:
  - a level indicating the importance (8 levels)
    - `LOG_EMERG` highest priority
    - `LOG_DEBUG` lowest priority
  - a facility that indicates the type of process that sent the message:
    - `LOG_MAIL, LOG_AUTH, LOG_USER, LOG_KERN, LOG_LPR, ...`
  - A text message.

/etc/syslog.conf

- Syslogd reads a configuration file that specifies how various messages should be handled (where they should go).
  - Sysadmin could set LOG_EMERG messages to be sent to the console
  - low priority messages from lpr could be thrown away.
  - Medium priority message from the mail server could be saved in a file.

Sending a message to syslogd

- Standard programming interface provided by **syslog**() function

```
#include <syslog.h>

void syslog( int priority,
             const char *message,
             . . . );
```


Syslog client/server

- Clients send messages to local syslogd through a unix domain (datagram) socket.
- All the details are handled by `syslog()`
- `syslogd` sends/receives messages to/from other hosts using UDP.

Back to daemons

- To force a process to run in the background just fork() and have the parent exit.
- There are a number of ways to disassociate a process from any controlling terminal.
- Daemons should close all unnecessary descriptors (often including stdin, stdout, stderr).

Too many daemons?

- There can be many servers running as daemons - most of them idle most of the time.
- Much of the startup code is the same for all the servers.
- Most of the servers are asleep, but use up space in the process table.

SuperServer

- Most Unix systems provide a “SuperServer” that solves the problem:
  - executes the startup code required by a bunch of servers.
  - Waits for incoming requests destined for the same bunch of servers.
  - When a request arrives - starts of the right server and gives it the request.

`inetd`

- The superserver is named inetd. This single daemon creates a number of sockets and waits for incoming requests.
- When a request arrives, inetd will fork and the child process handles the client.

`inetd` children

- The child process closes all unnecessary sockets.
- The child dup’s the client socket to descriptors 0,1 and 2 (stdin, stdout, stderr).
- The child exec’s the real server program, which handles the request and exits.

`inetd` based servers

- Servers that are started by inetd assume that the socket to the client is already established (descriptors 0,1 or 2).
- TCP servers started by `inetd` don’t call accept, so they must call `getpeername()` if they need to know the address of the client.

/etc/inetd.conf

- Inetd reads a configuration file that lists all the services it should handle. For each service inetd needs to know:
  - the port number and protocol
  - wait/nowait flag
  - login name the process should run as
  - pathname of real server program
  - command line arguments to server program

wait/nowait

- Specifying WAIT means that inetd should not look for new clients for the service until the child (the real server) has terminated.
- TCP servers usually specify nowait - this means inetd can start multiple copies of the TCP server program - providing concurrency.

UDP & wait/nowait

- Most UDP services run with inetd told to wait until the child server has died.
- What would happen if inetd did not wait for a UDP server to die before looking for new requests AND inetd get a time slice before the real server reads the request datagram?
- Some UDP servers hang out for a while, handling multiple clients.

Super inetd

- Some versions of inetd have server code to handle simple services such as echo server, daytime server, chargen, ...

Servers

- Servers that are expected to deal with frequent requests are typically not run from inetd: mail, web, NFS.

- Many servers are written so that a command line option can be used to run the server from inetd.