

## Daemons & `inetd`

Refs: Chapter 12

Netprog: daemons and inetd

1

---

---

---

---

---

---

---

---

## Daemons



A daemon is a process that:

- runs in the background
- not associated with any terminal
  - output doesn't end up in another session.
  - terminal generated signals (^C) aren't received.

Netprog: daemons and inetd

2

---

---

---

---

---

---

---

---

## Unix and Daemons

- Unix systems typically have many daemon processes.
- Most servers run as a daemon process.

Netprog: daemons and inetd

3

---

---

---

---

---

---

---

---

## Common Daemons

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

Netprog: daemons and inetd

4

---

---

---

---

---

---

---

---

## Daemon Output

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

Netprog: daemons and inetd

5

---

---

---

---

---

---

---

---

## Syslog service

- **syslogd** daemon provides system logging services to "clients".
- Simple API for "clients"
  - A library provided by O.S.

Netprog: daemons and inetd

6

---

---

---

---

---

---

---

---

## Centralized Administration

- A system administrator can control logging functions by specifying:
  - where messages should go
  - what kinds of messages are important
  - what can be ignored

---

---

---

---

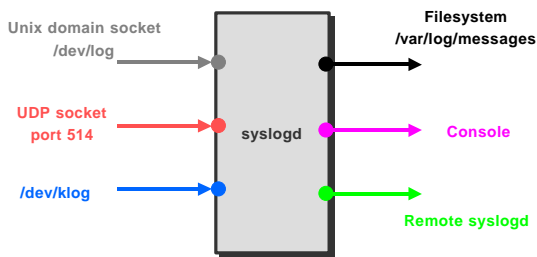
---

---

---

---

## syslogd



---

---

---

---

---

---

---

---

## Syslog messages

- Think of syslog as a server that accepts messages.
- Each message includes a number of fields, including:
  - a *level* indicating the importance (8 levels)
    - LOG\_EMERG highest priority
    - LOG\_DEBUG lowest priority

---

---

---

---

---

---

---

---

## Syslog message fields (cont.)

- a *facility* that indicates the type of process that sent the message:
  - LOG\_MAIL, LOG\_AUTH, LOG\_USER, LOG\_KERN, LOG\_LPR, ...
- A text *string*.

Message: *(level, facility, string)*

Netprog: daemons and inetd

10

---

---

---

---

---

---

---

---

## `/etc/syslog.conf`

- Syslogd reads a configuration file that specifies how various messages should be handled (where they should go).
- The sysadmin controls all logged messages by editing this file.

Netprog: daemons and inetd

11

---

---

---

---

---

---

---

---

## Examples

- 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.

Netprog: daemons and inetd

12

---

---

---

---

---

---

---

---

## Sending a message to syslogd

- Standard programming interface provided by `syslog()` function:

```
#include <syslog.h>
void syslog( int priority,
             const char *message,
             . . . );
```

- Works like `printf()`

Netprog: daemons and inetd

13

---

---

---

---

---

---

---

---

## 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.

Netprog: daemons and inetd

14

---

---

---

---

---

---

---

---

## 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.
  - Call `setsid()` and then `fork()` again.

Netprog: daemons and inetd

15

---

---

---

---

---

---

---

---

## Daemon initialization

- Daemons should close all unnecessary descriptors
  - often including `stdin`, `stdout`, `stderr`.
- Get set up for using syslog
  - Call `openlog()`
- Often change working directory.

Netprog: daemons and inetd 16

---

---

---


---

---

---

---

---



## Too many daemons?

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

Netprog: daemons and inetd 17

---

---

---


---

---

---

---

---



- 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 up the right server and *gives it the request*.

Netprog: daemons and inetd 18

---

---

---

---

---

---

---

---

## inetd

- The SuperServer is named **inetd**. This single daemon creates multiple sockets and waits for (multiple) incoming requests.
- **inetd** typically uses **select** to watch multiple sockets for input.
- When a request arrives, **inetd** will fork and the child process handles the client.

Netprog: daemons and inetd

19

---

---

---

---

---

---

---

---

## 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.

Netprog: daemons and inetd

20

---

---

---

---

---

---

---

---

## inetd based servers

- Servers that are started by **inetd** assume that the socket holding the request 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.

Netprog: daemons and inetd

21

---

---

---

---

---

---

---

---

## `/etc/inetd.conf`

- `inetd` reads a configuration file that lists all the services it should handle.
- `inetd` creates a socket for each listed service, and adds the socket to a `fd_set` given to `select()`.

Netprog: daemons and inetd

22

---

---

---

---

---

---

---

---

## `inetd` service specification

- For each service, `inetd` needs to know:
  - the port number and transport protocol
  - wait/howait flag.
  - login name the process should run as.
  - pathname of real server program.
  - command line arguments to server program.

Netprog: daemons and inetd

23

---

---

---

---

---

---

---

---

## example `/etc/inetd.conf`

```
# comments start with #
echo    stream  tcp  nowait  root    internal
echo    dgram  udp  wait   root    internal
chargen stream  tcp  nowait  root    internal
chargen dgram  udp  wait   root    internal
ftp     stream  tcp  nowait  root    /usr/sbin/ftpd ftpd -l
telnet  stream  tcp  nowait  root    /usr/sbin/telnetd telnetd
finger  stream  tcp  nowait  root    /usr/sbin/fingerd fingerd
# Authentication
auth    stream  tcp  nowait  nobody /usr/sbin/in.identd
        in.identd -l -e -o
# TFTP
tftp    dgram  udp  wait   root    /usr/sbin/tftpd tftpd -s
        /tftpboot
```

Netprog: daemons and inetd

24

---

---

---

---

---

---

---

---

## 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!

Netprog: daemons and inetd

25

---

---

---

---

---

---

---

---

## TCP and wait/nowait

TCP servers usually specify `nowait`.

This means `inetd` can start multiple copies of the TCP server program - providing concurrency!

Netprog: daemons and inetd

26

---

---

---

---

---

---

---

---

## 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.
  - `inetd` gets a time slice before the real server reads the request datagram?

Netprog: daemons and inetd

27

---

---

---

---

---

---

---

---

## UDP Servers that wait/nowait

- Some UDP servers hang out for a while, handling multiple clients before exiting.
- `inetd` was told to wait – so it ignores the socket until the UDP server exits.

Netprog: daemons and inetd

28

---

---

---

---

---

---

---

---

## Super inetd

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

Netprog: daemons and inetd

29

---

---

---

---

---

---

---

---

## 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`.

Netprog: daemons and inetd

30

---

---

---

---

---

---

---

---

## **xinetd**

- Some versions of Unix provide a service very similar to **inetd** called **xinetd**.
  - configuration scheme is different
  - basic idea (functionality) is the same...

---

---

---

---

---

---

---

---