Insider Tips: Server Management with inetd and xinetd

HERDING DAEMONS

If you have many daemons running on your server, it can be quite difficult to keep track of them all. inetd and xinetd manage these services centrally and also take care of exchanges with your clients, allowing programs without network code to operate as Internet servers. **BY MARC ANDRÉ SELIG**

Typical server applications handle network communication autonomously. They open a socket and listen at a TCP port, using the port to send and receive data. Interaction with a socket is not exactly rocket science, but coding a program for socket interaction can be troublesome and prone to error.

It can also be quite difficult to keep track of the various configurations. Central security settings involve a lot of effort if each server has its own set of configuration files. And what's more, it doesn't always make sense to re-invent the wheel by coding each program separately for network access.

Centralized Server Management

The so-called super servers, inetd and xinetd provide a simple alternative to independent daemons with their own network code. The super servers listen to network ports and pass on incoming connections to the appropriate processes. The processes use *stdin* and *stdout* to talk to their clients. As its configuration file, inetd uses a table of the TCP and UDP ports that it should listen to and the programs mapped to these ports. It opens sockets for the ports in the table and waits for incoming connections. Listing 1 shows a sample configuration.

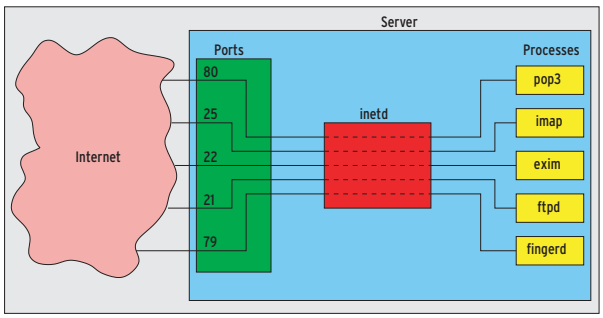

Figure 1: The Internet super server inetd offloads networking tasks from server processes. Processes can use *stdin* and *stdout* for client communication.

The fields in each line of the configuration file describe the service. The symbolic name, such as *smtp*, also specifies the required port, as inetd looks for the name in */etc/services*. Users are allowed to add their own ports to the *inetd* configuration file, however, if you add a port, make sure you choose a port number between 49152 and 65535, as these ports are reserved for private applications. IANA [1] has a complete list of registered numbers for UDP and TCP ports.

The second field in the configuration file has the socket type (this is always *stream* for TCP and *dgram* for UDP). The third field has the protocol (such as *tcp* or *udp*). The *nowait* option in the next column tells inetd not to wait for the running process to finish. In other words, the daemon will simply launch another process in case of another incoming connection.

wait tells inetd to wait for the running process to finish before accepting another connection to the port. *wait* is typically used with datagram sockets, such as UDP servers. UDP cannot assign datagrams to sessions at the transport layer. This means that inetd can only handle one process at any given time and will need to pass any datagrams for the port in question to the current process.

The last three arguments in the configuration line specify the user with whose privileges the server will run, the path to the binary, and any arguments. Depending on the implementation and

Listing 1: inetd Configuration File

```
01 # <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
02
03 smtp    stream tcp nowait mail /usr/sbin/exim exim -bs
04 printer stream tcp nowait lp   /usr/lib/cups/daemon/cups-lpd cups-lpd
05 ftp     stream tcp nowait root /usr/sbin/tcpd /usr/sbin/proftpd
```

version, the server may only support a small number of arguments (typically five or 20); the super server just ignores any other arguments.

Inetd does not provide access controls or security. This is the domain of programs such as the TCP wrapper program, *tcpd*, which parses */etc/hosts.allow* and */etc/hosts.deny* to apply granular controls to specific ports.

xinetd Does a Better Job

xinetd is a vastly improved version of the original inetd that supports modular configuration and mechanisms such as access control and protection against denial of service attacks. Listing 2 shows an example for the Exim mail server.

xinetd has two kinds of configuration files: */etc/xinetd.conf* contains global settings; individual services have their own description files. The file name typically includes the name of the service and is stored in the */etc/xinetd.d/* directory.

Besides standard options, you can configure a number of security and optimization variables, such as a list of hosts, or IP addresses, that are allowed to access a port. The ability to specify resource limits for the programs, or to log access in logfiles or the syslog, is another useful feature.

The configuration in Listing 3 allows a maximum of ten concurrent instances of each server. xinetd calls the servers with a low priority (*nice = 5*), does not start any more processes when the load reaches 2.5 (*max_load = 2.5*), and supports a maximum of 50 connections per second, blocking the service for 20 seconds if this limit is exceeded (*cps = 50 20*).

The *rlimit_as* variable restricts the percentage of memory that a process is allowed to hog. And *no_access = 24.0.0.0* blocks connections from the 24/8 subnet.

If the required server cannot launch because the required resources are exhausted, or the requesting computer has an IP address the administrator has blocked, this configuration directs xinetd to perform an *ident* lookup to discover the user name and logs this name along with the IP address.

Listing 2: /etc/xinetd.d/exim

```
01 service smtp
02 {
03         disable         = no
04         socket_type     = stream
05         wait            = no
06         user            = mail
07         server          = /usr/
   sbin/exim
08         server_args     = -bs
09         flags           = REUSE
10 }
```

Your Own Server

An inetd or xinetd server program can be very simple. It has to receive and send data using an existing TCP connection via standard input and output. Data from the client is accepted via *stdin*, and the server talks to the client via *stdout*. A trivial example of an inetd server only needs two lines of code:

```
#!/bin/sh
echo This is me and I ➥
am called `uname -n`.
```

To allow network access to this program, just add the following line to */etc/inetd.conf*:

Listing 3: /etc/xinetd.conf

```
01 defaults
02 {
03 instances      = 10
04 cps            = 50 20
05 nice           = 5
06 max_load       = 2.5
07 rlimit_as      = 32M
08 no_access      = 24.0.0.0
09 log_type       = SYSLOG daemon
10 log_on_failure = HOST USERID
11 log_on_success = HOST PID EXIT
   DURATION
12 }
```

```
sampleserver stream tcp ➥
nowait mas ➥
/usr/local/bin/sampleserver
```

sampleserver is the symbolic service name in this case. An entry for *sampleserver 50000/tcp* in */etc/services* tells inetd the port. After modifying the configuration file, you need to let the super server know about the changes. To do so, enter the following command as root: *killall -HUP inetd*.

The server should now be running as a TCP service. You can enter *telnet localhost 50000* to see the following output (the first three lines come from the telnet client):

```
Trying 127.0.0.1...
Connected to anmen.
Escape character is '^]'.
This is me and I am called
anmen.
Connection closed by foreign
host.
```

Pitfalls

Whether you write a program in C, Java, Perl, or shell script, make sure you pay a lot of attention to security. Don't forget to validate any input from clients.

Also, in the Unix workspace, a newline is a single \n, whereas the Internet newline is a combination of linefeed and newline (\r\n). inetd and xinetd do not convert character sets, so your server process will have to handle this itself.

Sense and Nonsense

It does not make sense to use the super server to launch every single daemon; each connection to one of these services causes overhead, as it involves relaunching the program. This is not a problem for small-footprint daemons that only run occasionally, but major services such as Apache or SSH take too long to launch, and this makes it impossible to use them effectively with inetd. On the upside, running as many services as possible via the super server keeps the process table clean and saves configuration work. ■

INFO

[1] IANA port list: *http://www.iana.org/assignments/port-numbers*

[2] xinetd: *http://www.xinetd.org*