NAME

inetd - internet "super-server"

SYNOPSIS

inetd [**-dlWw**] [**-a** *address*] [**-C** *rate*] [**-c** *maximum*] [**-p** *filename*] [**-R** *rate*] [**-s** *maximum*]
[*configuration_file*]

DESCRIPTION

The **inetd** utility should be run at boot time by */etc/rc* (see rc(8)). It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. The server program is invoked with the service socket as its standard input, output and error descriptors. After the program is finished, **inetd** continues to listen on the socket (except in some cases which will be described below). Essentially, **inetd** allows running one daemon to invoke several others, reducing load on the system.

The following options are available:

-a *address*

Specify one specific IP address to bind to. Alternatively, a hostname can be specified, in which case the IPv4 or IPv6 address which corresponds to that hostname is used. Usually a hostname is specified when **inetd** is run inside a jail(8), in which case the hostname corresponds to that of the jail(8) environment.

When the hostname specification is used and both IPv4 and IPv6 bindings are desired, one entry with the appropriate *protocol* type for each binding is required for each service in */etc/inetd.conf*. For example, a TCP-based service would need two entries, one using "tcp4" for the *protocol* and the other using "tcp6". See the explanation of the */etc/inetd.conf protocol* field below.

-C *rate*

Specify the default maximum number of times a service can be invoked from a single IP address in one minute; the default is unlimited. May be overridden on a per-service basis with the "max-connections-per-ip-per-minute" parameter.

-c *maximum*

Specify the default maximum number of simultaneous invocations of each service; the default is unlimited. May be overridden on a per-service basis with the "max-child" parameter.

-d Turn on debugging.

-l Turn on logging of successful connections.

-p Specify an alternate file in which to store the process ID.

-R *rate*

Specify the maximum number of times a service can be invoked in one minute; the default is 256. A rate of 0 allows an unlimited number of invocations.

-s *maximum*

Specify the default maximum number of simultaneous invocations of each service from a single IP address; the default is unlimited. May be overridden on a per-service basis with the "max-child-per-ip" parameter.

-W Turn on TCP Wrapping for internal services which are built in to **inetd**.

-w Turn on TCP Wrapping for external services. See the *IMPLEMENTATION NOTES* section for more information on TCP Wrappers support.

Upon execution, **inetd** reads its configuration information from a configuration file which, by default, is */etc/inetd.conf*. There must be an entry for each field of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a "#" at the beginning of a line. There must be an entry for each field. The fields of the configuration file are as follows:

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
user[:group][/login-class]
server-program
server-program-arguments
```

To specify an ONC RPC-based service, the entry would contain these fields:

```
service-name/version
socket-type
rpc/protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
user[:group][/login-class]
server-program
server-program-arguments
```

There are two types of services that **inetd** can start: standard and TCPMUX. A standard service has a well-known port assigned to it; it may be a service that implements an official Internet standard or is a BSD-specific service. As described in RFC 1078, TCPMUX services are nonstandard services that do not have a well-known port assigned to them. They are invoked from **inetd** when a program connects to the "tcpmux" well-known port and specifies the service name. This feature is useful for adding locally-developed servers. TCPMUX requests are only accepted when the multiplexor service itself is enabled, above and beyond and specific TCPMUX-based servers; see the discussion of internal services below.

The *service-name* entry is the name of a valid service in the file */etc/services*, or the specification of a UNIX domain socket (see below). For "internal" services (discussed below), the service name should be the official name of the service (that is, the first entry in */etc/services*). When used to specify an ONC RPC-based service, this field is a valid RPC service name listed in the file */etc/rpc*. The part on the right of the "/" is the RPC version number. This can simply be a single numeric argument or a range of versions. A range is bounded by the low version to the high version - "rusers/1-3". For TCPMUX services, the value of the *service-name* field consists of the string "tcpmux" followed by a slash and the locally-chosen service name. The service names listed in */etc/services* and the name "help" are reserved. Try to choose unique names for your TCPMUX services by prefixing them with your organization’s name and suffixing them with a version number.

The *socket-type* should be one of "stream", "dgram", "raw", "rdm", or "seqpacket", depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket. TCPMUX services must use "stream".

The *protocol* must be a valid protocol or "unix". Examples are "tcp" or "udp", both of which imply IPv4 for backward compatibility. The names "tcp4" and "udp4" specify IPv4 only. The names "tcp6" and "udp6" specify IPv6 only. The names "tcp46" and "udp46" specify that the entry accepts both IPv4 and IPv6 connections via a wildcard AF_INET6 socket. Rpc based services are specified with the "rpc/tcp" or "rpc/udp" service type. One can use specify IPv4 and/or IPv6 with the 4, 6 or 46 suffix, for example "rpc/tcp6" or "rpc/udp46". TCPMUX services must use "tcp", "tcp4", "tcp6" or "tcp46".

The *wait/nowait* entry specifies whether the server that is invoked by **inetd** will take over the socket associated with the service access point, and thus whether **inetd** should wait for the server to exit before listening for new service requests. Datagram servers must use "wait", as they are always invoked with the original datagram socket bound to the specified service address. These servers must read at least one datagram from the socket before exiting. If a datagram server connects to its peer, freeing the socket so **inetd** can receive further messages on the socket, it is said to be a "multi-threaded" server; it should read one datagram from the socket and create a new socket connected to the peer. It should fork, and the parent should then exit to allow **inetd** to check for new service requests to spawn new servers. Datagram servers which process all incoming datagrams on a socket and eventually time out are said to be "single-threaded". The comsat(8) and talkd(8) utilities are examples of the latter type of datagram server. The tftpd(8) utility is an example of a multi-threaded datagram server.

Servers using stream sockets generally are multi-threaded and use the "nowait" entry. Connection requests for these services are accepted by **inetd**, and the server is given only the newly-accepted socket connected to a client of the service. Most stream-based services operate in this manner. Stream-based servers that use "wait" are started with the listening service socket, and must accept at least one connection request before exiting. Such a server would normally accept and process incoming connection requests until a timeout. TCPMUX services must use "nowait".

The maximum number of outstanding child processes (or "threads") for a "nowait" service may be explicitly specified by appending a "/" followed by the number to the "nowait" keyword. Normally (or if a value of zero is specified) there is no maximum. Otherwise, once the maximum is reached, further connection attempts will be queued up until an existing child process exits. This also works in the case of "wait" mode, although a value other than one (the default) might not make sense in some cases. You can also specify the maximum number of connections per minute for a given IP address by appending a "/" followed by the number to the maximum number of outstanding child processes. Once the maximum is reached, further connections from this IP address will be dropped until the end of the minute. In addition, you can specify the maximum number of simultaneous invocations of each service from a single IP address by appending a "/" followed by the number to the maximum number of outstanding child processes. Once the maximum is reached, further connections from this IP address will be dropped.

The *user* entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root. The optional *group* part separated by ":" allows a group name other than the default group for this user to be specified. The optional *login-class* part separated by "/" allows specification of a login class other than the default "daemon" login class.

The *server-program* entry should contain the pathname of the program which is to be executed by **inetd** when a request is found on its socket. If **inetd** provides this service internally, this entry should be "internal".

The *server-program-arguments* entry lists the arguments to be passed to the *server-program*, starting with argv[0], which usually is the name of the program. If the service is provided internally, the *service-name* of the service (and any arguments to it) or the word "internal" should take the place of this entry.

Currently, the only internal service to take arguments is "auth". Without options, the service will always return "ERROR : HIDDEN-USER". The available arguments to this service that alter its behavior are:

-d *fallback*

Provide a *fallback* username. If the real "auth" service is enabled (with the **-r** option discussed below), return this username instead of an error when lookups fail for either socket credentials or the username. If the real "auth" service is disabled, return this username for every request. This is primarily useful when running this service on a NAT machine.

-F Same as **-f** but without the restriction that the username in *.fakeid* must not match an existing user.

-f If the file *.fakeid* exists in the home directory of the identified user, report the username found in that file instead of the real username. If the username found in *.fakeid* is that of an existing user, then the real username is reported. If the **-i** flag is also given then the username in *.fakeid* is checked against existing user IDs instead.

-g Instead of returning the user's name to the ident requester, report a username made up of random alphanumeric characters, e.g., "c0c993". The **-g** flag overrides not only the user names, but also any fallback name, *.fakeid* or *.noident* files.

-i Return numeric user IDs instead of usernames.

-n If the file *.noident* exists in the home directory of the identified user, return "ERROR : HIDDEN-USER". This overrides any *fakeid* file which might exist.

-o *osname*
Use *osname* instead of the name of the system as reported by uname(3).

-r Offer a real "auth" service, as per RFC 1413. All the remaining flags apply only in this case.

-t *sec*[**.***usec*]
Specify a timeout for the service. The default timeout is 10.0 seconds.

The **inetd** utility also provides several other "trivial" services internally by use of routines within itself. These services are "echo", "discard", "chargen" (character generator), "daytime" (human readable time), and "time" (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). All of these services are available in both TCP and UDP versions; the UDP versions will refuse service if the request specifies a reply port corresponding to any internal service. (This is done as a defense against looping attacks; the remote IP address is logged.) For details of these services, consult the appropriate RFC document.

The TCPMUX-demultiplexing service is also implemented as an internal service. For any TCPMUX-based service to function, the following line must be included in *inetd.conf*:

```
tcpmux    stream    tcp       nowait    root      internal
```

When given the **-l** option **inetd** will log an entry to syslog each time a connection is accepted, noting the service selected and the IP-number of the remote requester if available. Unless otherwise specified in the configuration file, and in the absence of the **-W** and **-w** options, **inetd** will log to the "daemon" facility.

The **inetd** utility rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread. Except when started in debugging mode, or configured otherwise with the **-p** option, **inetd** records its process ID in the file */var/run/inetd.pid* to assist in reconfiguration.

IMPLEMENTATION NOTES

TCP Wrappers

When given the **-w** option, **inetd** will wrap all services specified as "stream nowait" or "dgram" except for "internal" services. If the **-W** option is given, such "internal" services will be wrapped. If both options are given, wrapping for both internal and external services will be enabled. Either wrapping option will cause failed connections to be logged to the "auth" syslog facility. Adding the **-l** flag to the wrapping options will include successful connections in the logging to the "auth" facility.

Note that **inetd** only wraps requests for a "wait" service while no servers are available to service requests. Once a connection to such a service has been allowed, **inetd** has no control over subsequent connections to the service until no more servers are left listening for connection requests.

When wrapping is enabled, the *tcpd* daemon is not required, as that functionality is builtin. For more information on TCP Wrappers, see the relevant documentation (hosts_access(5)). When reading that document, keep in mind that "internal" services have no associated daemon name. Therefore, the service name as specified in *inetd.conf* should be used as the daemon name for "internal" services.

TCPMUX

RFC 1078 describes the TCPMUX protocol: ‘‘A TCP client connects to a foreign host on TCP port 1. It sends the service name followed by a carriage-return line-feed <CRLF>. The service name is never case sensitive. The server replies with a single character indicating positive (+) or negative (-) acknowledgment, immediately followed by an optional message of explanation, terminated with a <CRLF>. If the reply was positive, the selected protocol begins; otherwise the connection is closed.’’ The program is passed the TCP connection as file descriptors 0 and 1.

If the TCPMUX service name begins with a "+", **inetd** returns the positive reply for the program. This allows you to invoke programs that use stdin/stdout without putting any special server code in them.

The special service name "help" causes **inetd** to list the TCPMUX services which are enabled in *inetd.conf*.

IPsec

The implementation includes a tiny hack to support IPsec policy settings for each socket. A special form of comment line, starting with "#@", is interpreted as a policy specifier. Everything after the "#@" will be used as an IPsec policy string, as described in ipsec_set_policy(3). Each policy specifier is applied to all the following lines in *inetd.conf* until the next policy specifier. An empty policy specifier resets the IPsec policy.

If an invalid IPsec policy specifier appears in *inetd.conf*, **inetd** will provide an error message via the syslog(3) interface and abort execution.

UNIX Domain Sockets

In addition to running services on IP sockets, **inetd** can also manage UNIX domain sockets. To do this you specify a *protocol* of "unix" and specify the UNIX domain socket as the *service-name*. The *service-type* may be "stream" or "dgram". The specification of the socket must be an absolute path name, optionally prefixed by an owner and mode of the form *:user:group:mode:*. The specification:

```
      :news:daemon:220:/var/run/sock
```

creates a socket owned by user "news" in group "daemon" with permissions allowing only that user and group to connect. The default owner is the user that **inetd** is running as. The default mode only allows the socket's owner to connect.

WARNING: while creating a UNIX domain socket, **inetd** must change the ownership and permissions on the socket. This can only be done securely if the directory in which the socket is created is writable only by root. Do *NOT* use **inetd** to create sockets in world writable directories such as */tmp*; use */var/run* or a similar directory instead.

Internal services may be run on UNIX domain sockets, in the usual way. In this case the name of the internal service is determined using the last component of the socket's pathname. For example, specifying a socket named */var/run/chargen* would invoke the "chargen" service when a connection is received on that socket.

FILES

/etc/inetd.conf	configuration file
/etc/netconfig	network configuration data base
/etc/rpc	translation of service names to RPC program numbers
/etc/services	translation of service names to port numbers
/var/run/inetd.pid	the pid of the currently running **inetd**

EXAMPLES

Examples for a variety of services are available in */etc/inetd.conf*.

It includes examples for **bootpd**, **comsat**, **cvs**, **date**, **fingerd**, **ftpd**, **imapd**, **nc**, **nmbd**, **nntpd**, **rlogind**, **rpc.rquotad**, **rpc.rusersd**, **rpc.rwalld**, **rpc.statd**, **rpc.sprayd**, **rshd**, **prometheus_sysctl_exporter**, **smtpd**, **smbd**, **swat talkd**, **telnetd**, **tftpd**, **uucpd**.

The internal services provided by **inetd** for daytime, time, echo, discard and chargen are also included, as well as chargen for **ipsec** Authentication Headers

Examples for handling auth requests via **identd**, are similarly included.

ERROR MESSAGES

The **inetd** server logs error messages using syslog(3). Important error messages and their explanations are:

service/*protocol* server failing (looping), service terminated.
The number of requests for the specified service in the past minute exceeded the limit. The limit exists to prevent a broken program or a malicious user from swamping the system. This message may occur for several reasons:

1. There are many hosts requesting the service within a short time period.

2. A broken client program is requesting the service too frequently.

3. A malicious user is running a program to invoke the service in a denial-of-service attack.

4. The invoked service program has an error that causes clients to retry quickly.

Use the **-R** *rate* option, as described above, to change the rate limit. Once the limit is reached, the service will be reenabled automatically in 10 minutes.

service/*protocol*: No such user *user*, service ignored
service/*protocol*: getpwnam: *user*: No such user
No entry for *user* exists in the passwd(5) database. The first message occurs when **inetd** (re)reads the configuration file. The second message occurs when the service is invoked.

service: can’t set uid *uid*
service: can’t set gid *gid*
The user or group ID for the entry’s *user* field is invalid.

setsockopt(SO_PRIVSTATE): Operation not supported
The **inetd** utility attempted to renounce the privileged state associated with a socket but was unable to.

unknown *rpc/udp* or *rpc/tcp*
No entry was found for either *udp* or *tcp* in the netconfig(5) database.

unknown *rpc/udp6* or *rpc/tcp6*
No entry was found for either *udp6* or *tcp6* in the netconfig(5) database.

SEE ALSO

cvs(1) (*ports/devel/opencvs*), date(1), nc(1), ipsec_set_policy(3), ipsec(4), hosts_access(5), hosts_options(5), login.conf(5), netconfig(5), passwd(5), rpc(5), services(5), bootpd(8), comsat(8), fingerd(8), ftpd(8), imapd(8) (*ports/mail/courier-imap*), nmbd(8) (*ports/net/samba412*), rlogind(8), rpc.rquotad(8), rpc.rusersd(8), rpc.rwalld(8), rpc.statd(8), rshd(8), prometheus_sysctl_exporter(8), smbd(8) (*ports/net/samba412*), talkd(8), telnetd(8) (*ports/net/freebsd-telnetd*), tftpd(8), uucpd(8) (*ports/net/freebsd-uucp*)

Michael C. St. Johns, *Identification Protocol*, RFC1413.

HISTORY

The **inetd** utility appeared in 4.3BSD. TCPMUX is based on code and documentation by Mark Lottor. Support for ONC RPC-based services is modeled after that provided by SunOS 4.1. The IPsec hack was contributed by the KAME project in 1999. The FreeBSD TCP Wrappers support first appeared in FreeBSD 3.2.