61 - TCP / IP Services

- **Service Daemons**
  They are normall started when the system boots-up or manually and provide services at one or more specific ports.
- **List of programs and ports they listen to in system.**
  `netstat --inet` Shows all current `tcp/ip` (TCP/UDP) connections.
  `netstat -ltunp` Shows all the open `tcp/udp` ports with Programs + PID
  `l`=listen, `t`=tcp, `u`=udp, `n`=numeric ip and ports, `p`=PID and listening program
  - **The superdaemons `inetd` and `xinetd`.**
    Certain services are made available via daemons running in the background. Certain other set of services are made available via a single daemon which watches the service ports and starts the appropriate program when a client of that service requests attention. This watching demaon is called 'superdaemon'. There are two versions of this type of superdaemons: `inetd` and `xinetd`(more recent).
  - `inetd` **Superdaemon**
    This daemon uses the settings in its configuration file `/etc/inetd.conf` to determine which service ports will be watched and which service programs are associated with them. When a service port receive a request from a client on a particular port , inetd can be configured to use a tcpwrapper which will check if the client host is allowed to use this service before the service program is started.
  - **The configuration file `inetd.conf`**
    Each port that need to be watched gets one configuration line. The parameters are separated with spaces or TABs. Here is the configuration line format:

service	socketType	protocol	wait	user	program	arguments

eg.

```
ftp       stream  tcp  nowait  root  /usr/sbin/tcpd wuftpd
telnet    stream  tcp  nowait  root  /usr/sbin/tcpd telnetd
```

service: name of the service referenced in the file `/etc/services`

socket: can be `stream`, `dgram`, `raw`, `rdm` or `seqpacket`
`stream`: TCP
`dgram`: UDP
`raw`: raw format
`rdm`: Reliable Delivered Message
`seqpacket`: Sequenced Packet Socket

wait: Can be `wait` or `nowait`
Tells `inetd` whether it should wait for the server to come back before accepting another client connection.
`nowait` is used for multi-threaded services(most services)
`wait` is used for single-threaded services(some UDP services)
eg: `comsat`, `biff`, `talkd` and `tftpd`

user: Which local user will be the owner of the service process.

program: Program to start the service (normally the `tcpd` tcpwrapper)

arguments: Either the service program as arguments for the `tcpd` tcp wrapper or the service program itself without tcpwrapper...NOT recommended.

- `xinetd` **Superdaemon**
  This more recent Superdaemon allows for more flexibility and security. It uses one main configuration file `/etc/xinetd.conf` which can be extended to multiple service definition files via the parameter `includedir`.
  eg. `includedir /etc/xinetd.d` . (All files in the `/etc/xinetd.d/` directory).
- **Advantages of xinetd over inetd:**
  - `xinetd` uses the control files(`hosts.allow` and `hosts.deny`) directly without the need to use the tcpwrapper `tcpd`.
  - Limits the connections either general, per client or per service
  - Certain clients can be given certain services vs. others
  - Protection against DenialOfService
  - Produces its own log files independantly from syslog
  - Possibility to redirect incoming requests to another server(eg. in a DMZ)
  - Full support of IPv6
  - Interaction with the client: Messages different for success vs. failure to connect.
- To convert the service definition parameters from `inetd` format to `xinetd` format, the tool `xconv.pl` can be used. It is delivered with the `xinetd` package.
  The `xinetd.conf` contains the default and per-service definitions.
  The default definitions are used for all of the services. In case of conflict with the per-service definition, the per-service prevail. For example for defaults and ftp service:

```
defaults
{
      instances   = 15
      log_type           = FILE /var/log/xinetd.log
      #log_type          = SYSLOG daemon info
      log_on_success     = HOST PID USERID DURATION EXIT
      log_on_failure     = HOST USERID RECORD
      only_from          = 192.0.0.0/8
      disabled           = shell login exec comsat
      disabled           += telnet ftp
      disabled           += name uucp tftp
      disabled           += finger systat netstat
}

service ftp
{
      socket_type        = stream
      wait               = no
      user               = root
      server             = /usr/sbin/in.ftpd
      server_args        = -l
      instances          = 4
      access_times       = 7:00-12:30 13:30-21:00
      nice               = 10
      only_from          = 192.168.1.0/24
      disabled           = yes
}
```

= sets the value,
+= adds the value (to default values) ,

-= deletes the value (from default values)

Deactivated parameters starts the line with a '#'. The parameters meanings are somewat similar to the `inetd.conf` but allows for more flexibility. The service definition block starts with the word `service` followed by the service name, then all of the parameters for this service are enclosed within curly brackets.'{....}'. The parameter `disable = yes` tells that the service is disabled. It must be set to `no` to enable it.

- **Tcpwrappers**
  The tcpwrappers are programs that uses configuration files to check if the client host is allowed to use the requested service. One commonly used tcpwrapper is `tcpd`. It uses the `/etc/hosts.allow` and `/etc/hosts.deny` files for this purpose. They contain a listing of the hosts concerned for each requested service. Here is the logic: If none of the two files exists, then all of the hosts are allowed to use all watched services.

  The access control software consults two files. The search stops at the first match:
  - Access will be granted when a (daemon,client) pair matches an entry in the `/etc/hosts.allow` file.
  - Otherwise, access will be denied when a (daemon,client) pair matches an entry in the `/etc/hosts.deny` file.
  - Otherwise, access will be granted.

  Format of `hosts.allow` and `hosts.deny`:
  Syntax:
  daemon: [*client1*].... [EXCEPT *client2* [*client3*] ....]
  eg.

```
ALL:           LOCAL @some_netgroup
ALL:           .foobar.edu EXCEPT terminalserver.foobar.edu
in.fingerd:    .mydomain.com EXCEPT hacker.mydomain.com
vsftpd:        .mylocal.domains
```

  Wildcards

  The access control language supports explicit wildcards:

`ALL`	The universal wildcard, always matches.
`LOCAL`	Matches any host whose name does not contain a dot character.
`UNKNOWN`	Matches any user whose name is unknown, and matches any host whose name or address are unknown. This pattern should be used with care: host names may be unavailable due to temporary name server problems. A network address will be unavailable when the software cannot figure out what type of network it is talking to.
`KNOWN`	Matches any user whose name is known, and matches any host whose name and address are known. This pattern should also be used with care for the same reasons as for `UNKNOWN`.
`PARANOID`	Matches any host whose name does not match its address. When `tcpd` is built with `-DPARANOID` (default), it drops requests from such clients even before looking at the access control tables. Build without `-DPARANOID` when you want more control over such

requests.

See `man hosts_access` for more information.

Examples:

To activate the service `vsftpd` (`ftp` Server):

- Edit this above file.
- Make sure that the follwoing line is as follows:
  `disable = no`
- Save the file and reload the `xinetd` daemon:
  `/etc/init.d/xinetd reload`
- Test the service:
  `telnet localhost 21` or
  `ftp localhost`

- **Description of some Network services**
  - File listing all available services under linux `/etc/services`
  (commented-out lines provides incoming requests for service only)

- **TELNET (Port 23**)
  - Description and purpose
  - Telnet Client vs Server(`telnetd`)
  - Port assignment feature
  - Login and logout procedure
  - Terminal specifications (VT100-VT102)
  - Telnet Client programs:
    - `telnet` - regular
    - `ktelnet` - KDE Telnet interface fount in ‚kpa' on CD
    - `TRMPTEL.EXE` - Windows progrqam.
      Small but good telnet prgm that runs
    - `puttytel.exe` - Telnet for windows from putty package
    - and others

- **FINGER (Port 79)**
  - Description and use(`finger` and `finger-server` packages)
    - `finger` (for local host users list)
    - `finger @`*`hostname`*
    - `finger @`*`hostIPAddr`*`.`
    - `finger root@idefix.michel.home`
    - Exercise with Finger

- **TALK (Port 517)**
  - Description and use talk root@idefix.michel.home
  - Exercise with talk
  - Install: `talk, talk-server, xhtalk, xtalk, ytalk`

- **Secure Shell - `ssh` (Port 22)**
  - Like telnet but uses SSL Dual keys encryption method (most secure)

- **Remote Login - rsh (kshell - Port 544)**
  - Differences between Telnet and rsh (remote login)
    - Telnet allows for use of different port number as 23, rsh uses
    - `rsh` can issue remote commands directly as one of its parameters
    - Telnet is (VT100-VT102) compatible

- **WRITE not a Service ....but just a program**
  - Description and use of write
    (Sending one-way messages to users logged on the same machine)

- **FTP (Port 21)**
  - Description and purpose
  - Login and Logout procedure
  - User Access limitation through:
    - `/etc/ftpaccess` Rules of `ftp` access (for `wu.ftpd` only)
    - `/etc/ftpusers` List of users that are NOT allowed to ftp
  - FTP Client(ftp) vs Server(ftpd)
  - FTP Command set (see the following ftp man pages)
  - Existing Confortable FTP Programs
    - `mc` - Midnight Commander
    - `gftp` - Gnome FTP from series gnm on CD
    - `xftp` - Double win. Type ftp prg. Series,xap on CD
    - `xmftp`
    - `IglooFTP`