UNIT-4

1. Discuss how source code portability from IPV4 applications could be converted to use IPV6.

Ans:

- Most existing network applications are written assuming IPv4.
  - IPV4 Applications can be converted to IPV6 with minor modifications.
  - But we need to check if the receipient host has support for IPV6.
  - We may do this using #ifdefs in the code… But code will have too many of these **#ifdefs.**
  - Better way is to make the program protocol independent.
  - The first step is… remove all get_host_by_name and get_host_by_addr and use get_addr_info and get_name_info functions.
  - This helps deal with sockaddr as opaque objects referenced by a pointer an size which is what basic socket functions do.

2. Explain IPV6 addressing-testing Macros.

Ans:

The following 12 macros are defined to test an IPv6 address for certain properties.

The first seven macros test the basic type of IPv6 address.

The final five macros test the scope of an IPv6 multicast address.

```
#include <netinet/in.h>

int IN6_IS_ADDR_UNSPECIFIED(const struct in6_addr *aptr);

int IN6_IS_ADDR_LOOPBACK(const struct in6_addr *aptr);
```

```
#include <netinet/in.h>

int IN6_IS_ADDR_MULTICAST(const struct in6_addr *aptr);

int IN6_IS_ADDR_LINKLOCAL(const struct in6_addr *aptr);

int IN6_IS_ADDR_SITELOCAL(const struct in6_addr *aptr);

int IN6_IS_ADDR_V4MAPPED(const struct in6_addr *aptr);

int IN6_IS_ADDR_V4COMPAT(const struct in6_addr *aptr);

int IN6_IS_ADDR_MC_NODELOCAL(const struct in6_addr *aptr);

int IN6_IS_ADDR_MC_LINKLOCAL(const struct in6_addr *aptr);

int IN6_IS_ADDR_MC_SITELOCAL(const struct in6_addr *aptr);
```

```
int IN6_IS_ADDR_MC_ORGLOCAL(const struct in6_addr *aptr);
int IN6_IS_ADDR_MC_GLOBAL(const struct in6_addr *aptr);
```

3. What is daemon process? List out the numerous ways to start a daemon. Explain the significance of daemon_init function

Ans:

A **daemon process**, often simply called a daemon, is a background process that runs on a computer system, typically without direct user interaction. Daemons are usually started during the system boot process and continue to run in the background, performing various tasks such as managing system services, handling hardware events, or performing periodic maintenance.

There are several ways to start a daemon process:

1. Init Scripts: On Unix-like systems, init scripts are commonly used to start and stop daemons. These scripts are typically located in directories like `/etc/init.d/` or `/etc/rc.d/` and are executed during the system boot process.

2. Systemd: Modern Linux distributions often use systemd, which is a system and service manager. Systemd can start and manage daemons, and it uses configuration files (unit files) to define how services, including daemons, should be started.

3. Upstart: Some Linux distributions, like Ubuntu (before version 15.04), used Upstart as an init system. Upstart also provided a way to start and manage daemon processes.

4. cron: On Unix-like systems, cron can be used to schedule daemon-like tasks at specific intervals.

5. inetd/xinetd: These are super servers that can be used to start daemons on-demand in response to network requests.

```
#include "unp.h"
#include <syslog.h>
#define MAXFD 64
extern int daemon_proc;
int daemon_init(const char *pname, int facility){
    int i;
    pid_t pid;
    if ( (pid = Fork()) < 0)
        return (-1);
    else if (pid)
        _exit(0);
    if (setsid() < 0)
        return (-1);
```

```
    Signal(SIGHUP, SIG_IGN);
    if ( (pid = Fork()) < 0)
        return (-1);
    else if (pid)
        _exit(0);
    daemon_proc = 1;
    chdir("/");
    for (i = 0; i < MAXFD; i++)
        close(i);
    open("/dev/null", O_RDONLY);
    open("/dev/null", O_RDWR);
    open("/dev/null", O_RDWR);
    openlog(pname, LOG_PID, facility);
    return (0);
}
```

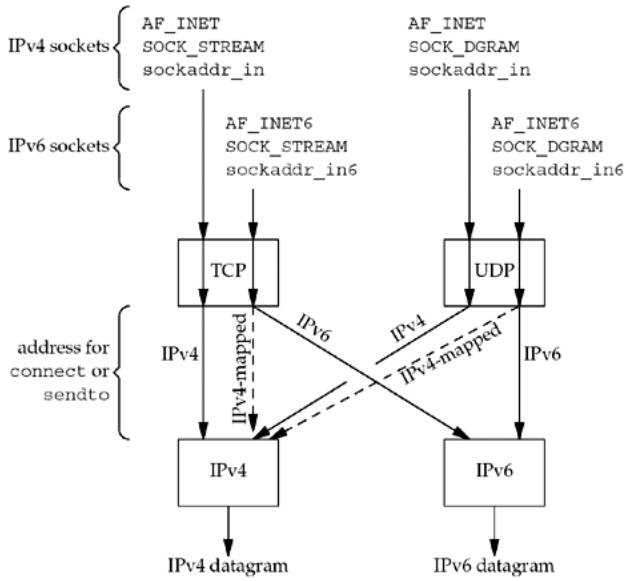
4. Explain with a neat block diagram Processing of client requests, depending on address type and socket type

OR

Illustrate the steps that allow an IPV6 TCP client to communicate with IP dual stack

Ans:

1. An IPv4 server starts on an IPv4-only host and creates an IPv4 listening socket.
2. The IPv6 client starts and calls getaddrinfo asking for only IPv6 addresses (it requests the AF_INET6 address family and sets the AI_V4MAPPED flag in its *hints* structure). Since the IPv4-only server host has only A records, an IPV4-mapped IPv6 address is returned to the client.
3. The IPv6 client calls connect with the IPv4-mapped IPv6 address in the IPv6socket address structure. The kernel detects the mapped address and automatically sends an IPv4 SYN to the server.
4. The server responds with an IPv4 SYN/ACK, and the connection is established using IPv4 datagrams.



- If an IPv4 TCP client calls connect specifying an IPv4 address, or if an IPv4 UDP client calls sendto specifying an IPv4 address, nothing special is done.
- If an IPv6 TCP client calls connect specifying an IPv6 address, or if an IPv6 UDP client calls sendto specifying an IPv6 address, nothing special is done.
- If an IPv6 TCP client specifies an IPv4-mapped IPv6 address to connect or if an IPv6 UDP client specifies an IPv4-mapped IPv6 address to sendto, the kernel detects the mapped address and causes an IPv4 datagram to be sent instead of an IPv6 datagram.
- An IPv4 client cannot specify an IPv6 address to either connect or sendto because a 16-byte IPv6 address does not fit in the 4-byte in_addr structure within the IPv4 sockaddr_in structure.

5. Summarize the steps that allow an IPV4 TCP client to communicate with IPV6 server using dual stack

Ans:

Step 1: IPV6 Server starts, creates a listening IPV6 socket and it binds wildcard address to the socket.

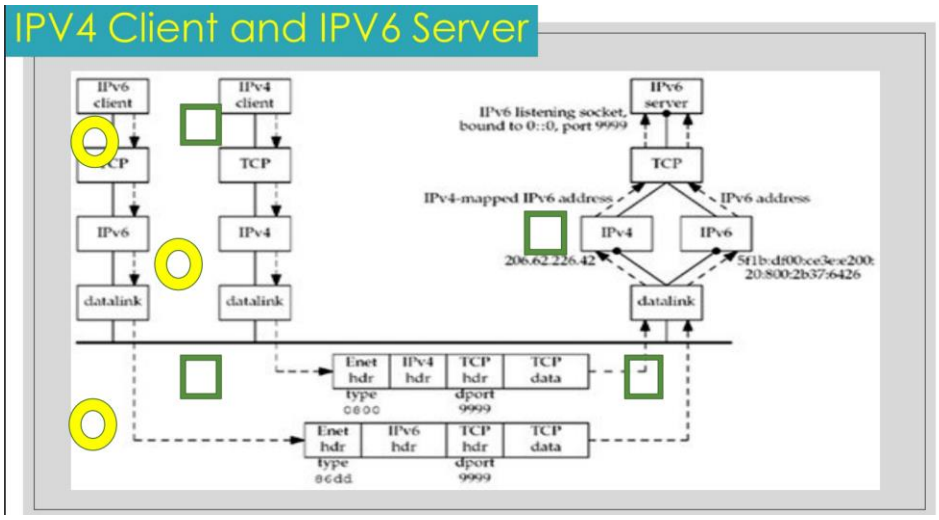
Step 2: IPV4 client calls get_host_by_name and finds an A record...

Steps 3: The Client calls 'connect' and client's host sends an IPV4 SYN to server.

Step 4: The Server host receives IPV4 SYN directed to IPV6 socket, sets a flag indicating this connection is using IPV4 mapped IPV6 address and responds with IPV4 SYN/ACK.

Step 5: When the server host sends IPV4-mapped-IPV6 address, IP Stack generates IPV4 datagram to IPV4 address.

Step 6: Dual Stack handles all the finer details and the Server is unaware that it is communicating with IPV4 client.



6. With a function prototype explain the syslog function.

Ans:

- Since a daemon does not have a controlling terminal, it cannot just fprintf to stderr.
- The common technique for logging messages from a daemon is to call the syslog function.
- Function prototype:

```
#include <syslog.h>

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

- Although this function was originally developed for BSD systems, it is provided by virtually all Unix vendors today.
- The description of syslog in the POSIX specification is consistent with what we describe here.
- RFC 3164 provides documentation of the BSD syslog protocol.