Porting IPv6 -- examples

Consider the following IPv4 code examples:

IPv4 client code

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
     ...
main(argc, argv) /* client side */
     int argc;
     char *argv[];
{
     struct sockaddr_in server;
     struct servent *sp;
     struct hostent *hp;
     int s;
     ...
     sp = getservbyname("login", "tcp");
     if (sp == NULL) {
             fprintf(stderr, "rlogin: tcp/login: unknown service\n");
             exit(1);
     }
     hp = gethostbyname(argv[1]);
     if (hp == NULL) {
             fprintf(stderr, "rlogin: %s: unknown host\n", argv[1]);
             exit(2);
```

```
        }
        memset((char *)&server, 0, sizeof(server));
        memcpy((char *)&server.sin_addr, hp->h_addr, hp->h_length);
        server.sin_len = sizeof(server);
        server.sin_family = hp->h_addrtype;
        server.sin_port = sp->s_port;
        s = socket(AF_INET, SOCK_STREAM, 0);
        if (s < 0) {
                perror("rlogin: socket");
                exit(3);
        }
        ...
        /* Connect does the bind for us */
        if (connect(s, (struct sockaddr *)&server, sizeof(server)) < 0) {
                perror("rlogin: connect");
                exit(5);
        }
        ...
        exit(0);
}
```

IPv4 server code

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
        ...
main(argc, argv) /* server side */
        int argc;
        char *argv[];
{
        int f;
        struct sockaddr_in from;
        struct sockaddr_in sin;
```

```
	struct servent *sp;



	sp = getservbyname("login", "tcp");
	if (sp == NULL) {
		fprintf(stderr,
			"rlogind: tcp/login: unknown service\n");
		exit(1);
	}
	...
#ifndef DEBUG
	/* Disassociate server from controlling terminal. */
	...
#endif



	memset((char *)&sin, 0, sizeof(sin));
	sin.sin_len = sizeof(sockaddr_in);
	sin.sin_port = sp->s_port;	/* Restricted port */
	sin.sin_addr.s_addr = INADDR_ANY;
	...
	f = socket(AF_INET, SOCK_STREAM, 0);
	...
	if (bind(f, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
		...
	}
	...
	listen(f, 5);
	for (;;) {
		int g, len = sizeof(from);



		g = accept(f, (struct sockaddr *) &from, &len);
		if (g < 0) {
			if (errno != EINTR)
```

```
                                syslog(LOG_ERR, "rlogind: accept: %m");
                        continue;
                }
                if (fork() == 0) {
                        close(f);
                        doit(g, &from);
                }
                close(g);
        }
        exit(0);
}
```

This code can be ported to IPv6 with only a small number of changes. These changes are highlighted in the examples below by comments in the code.

IPv4 client code ported to IPv6

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
     ...
main(argc, argv) /* client side */
     int argc;
     char *argv[];
{
     /*                                                    */
     /* OLD code: struct sockaddr_in server;               */
     /*                                                    */
     /* Change structure to sockaddr_in6 from sockaddr_in. */
     /*                                                    */
     struct sockaddr_in6 server;
     struct servent *sp;
     struct hostent *hp;
     int s;
```

```
...
sp = getservbyname("login", "tcp");
if (sp == NULL) {
        fprintf(stderr, "rlogin: tcp/login: unknown service\n");
        exit(1);
}



/*                                                     */
/* OLD code: hp = gethostbyname(argv[1]);              */
/*                                                     */
/* Use gethostbyname2 instead of gethostbyname.       */
/*                                                     */
hp = gethostbyname2(argv[1], AF_INET6);
if (hp == NULL) {
        fprintf(stderr, "rlogin: %s: unknown host\n", argv[1]);
        exit(2);
}
memset((char *)&server, 0, sizeof(server));



/*                                                     */
/* OLD code: Not applicable.                           */
/*                                                     */
/* If the len member was not in the original IPv4 code*/
/* add it now and make sure it is sin6_len for IPv6.  */
/*                                                     */
server.sin6_len = sizeof(server);



/*                                                     */
/* OLD code: memcpy((char *)&server.sin_addr, ...     */
/* OLD code: server.sin_family = hp->h_addrtype;      */
/* OLD code: server.sin_port = sp->s_port;            */
/*                                                     */
```

```
        /* Make sure you are using sockaddr_in6 members.      */
        /*                                                    */
        memcpy((char *)&server.sin6_addr, hp->h_addr, hp->h_length);
        server.sin6_family = hp->h_addrtype;
        server.sin6_port = sp->s_port;



        /*                                                    */
        /* OLD code: s = socket(AF_INET, SOCK_STREAM, 0);     */
        /*                                                    */
        /* Use the correct address family for IPv6.           */
        /*                                                    */
        s = socket(AF_INET6, SOCK_STREAM, 0);
        if (s < 0) {
                perror("rlogin: socket");
                exit(3);
        }
        ...
        /* Connect does the bind for us */
        if (connect(s, (struct sockaddr *)&server, sizeof(server)) < 0) {
                perror("rlogin: connect");
                exit(5);
        }
        ...
        exit(0);
    }
```

NOTE: In the assignments to `server.sin6_addr` and `server.sin6_family` `hp->h_length` will always be equal to **sizeof(struct in6addr)** and `hp->h_addrtype` will always be equal to **AF_INET6**.

IPv4 server code ported to IPv6

```
    #include <sys/types.h>
```

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
     ...
main(argc, argv) /* server side */
     int argc;
     char *argv[];
{
     int f;



     /*                                                    */
     /* OLD code: struct sockaddr_in from;                 */
     /* OLD code: struct sockaddr_in sin;                  */
     /*                                                    */
     /* Change structure to sockaddr_in6 from sockaddr_in. */
     /*                                                    */
     struct sockaddr_in6 from;
     struct sockaddr_in6 sin;
     struct servent *sp;



     sp = getservbyname("login", "tcp");
     if (sp == NULL) {
             fprintf(stderr,
                     "rlogind: tcp/login: unknown service\n");
             exit(1);
     }
     ...
#ifndef DEBUG
     /* Disassociate server from controlling terminal. */
     ...
#endif
```

```
memset((char *)&sin, 0, sizeof(sin));



/*                                                   */
/* OLD code: Not applicable.                         */
/*                                                   */
/* If the len member was not in the original IPv4 code*/
/* add it now and make sure it is sin6_len for IPv6.  */
/*                                                   */
sin.sin6_len = sizeof(sin);



/*                                                   */
/* OLD code: sin.sin_port = sp->s_port;              */
/*                                                   */
/* Make sure you are using sockaddr_in6 members.     */
/*                                                   */
sin.sin6_port = sp->s_port;     /* Restricted port */



/*                                                   */
/* OLD code: sin.sin_addr.s_addr = INADDR_ANY;       */
/*                                                   */
/* Make the modifications for assigning in6addr_any to*/
/* sin6_addr.                                        */
/*                                                   */
sin.sin6_addr = in6addr_any;
...



/*                                                   */
/* OLD code: f = socket(AF_INET, SOCK_STREAM, 0);    */
/*                                                   */
```

```
        /* Use the correct address family for IPv6.           */
        /*                                                     */
        f = socket(AF_INET6, SOCK_STREAM, 0);
        ...
        if (bind(f, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
                ...
        }
        ...
        listen(f, 5);
        for (;;) {
                int g, len = sizeof(from);



                g = accept(f, (struct sockaddr *) &from, &len);
                if (g < 0) {
                        if (errno != EINTR)
                                syslog(LOG_ERR, "rlogind: accept: %m");
                        continue;
                }
                if (fork() == 0) {
                        close(f);
                        doit(g, &from);
                }
                close(g);
        }
        exit(0);
    }
```

As can be seen in the two IPv6 ported examples, there are only a few changes required to port IPv4 applications to IPv6. You may want to go one step further and use the new **getaddrinfo**(3N) and **getnameinfo**(3N) functions to make your IPv6 application more portable. The following examples show how you could modify the client and server examples to use **getaddrinfo**(3N).

IPv6 client code using getaddrinfo

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>



main(argc, argv) /* client side */
     int argc;
     char *argv[];
{
     struct addrinfo req, *ans;
     int code, s;



     req.ai_flags = 0;



     req.ai_family = PF_INET6;              /* Same as AF_INET6.            */
     req.ai_socktype = SOCK_STREAM;



     /*                                                   */
     /* Use default protocol (in this case tcp)           */
     /*                                                   */
     req.ai_protocol = 0;
     if ((code = getaddrinfo(argv[1], "login", &req, &ans)) != 0) {
             fprintf(stderr, "rlogin: getaddrinfo failed code %d\n",
                     code);
             exit(1);
     }
```

```
/*                                                      */
/* ans must contain at least one addrinfo, use          */
/* the first.                                           */
/*                                                      */
s = socket(ans->ai_family, ans->ai_socktype, ans->ai_protocol);
if (s < 0) {
        perror("rlogin: socket");
        exit(3);
}



...



/* Connect does the bind for us */
if (connect (s, ans->ai_addr, ans->ai_addrlen) < 0) {
        perror("rlogin: connect");
        exit(5);
}



...

/*                                                      */
/* Free answers after use                               */
/*                                                      */
freeaddrinfo(ans);



/* ... */



exit(0);
```

```
}
```

IPv6 server code using getaddrinfo

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>



main(argc, argv) /* server side */
     int argc;
     char *argv[];
{
     struct sockaddr_in6 from;
     struct addrinfo req, *ans;
     int code, f, len;



     /*                                                      */
     /* Set ai_flags to AI_PASSIVE to indicate that return */
     /* address is suitable for bind()                      */
     /*                                                      */
     req.ai_flags = AI_PASSIVE;



     req.ai_family = PF_INET6;             /* Same as AF_INET6.    */



     req.ai_socktype = SOCK_STREAM;
     req.ai_protocol = 0;
     if ((code = getaddrinfo(NULL, "login", &req, &ans)) != 0) {
             fprintf(stderr, "rlogind: getaddrinfo failed code %d\n",
                     code);
             exit(1);
     }



      ...



#ifndef DEBUG
     /* Disassociate server from controlling terminal. */
     ...
#endif



     /*                                                    */
     /* ans must contain at least one addrinfo, use        */
     /* the first.                                         */
     /*                                                    */
     f = socket(ans->ai_family, ans->ai_socktype, ans->ai_protocol);
      ...



     if (bind(f, ans->ai_addr, ans->ai_addrlen) < 0) {
             ...
     }



     listen(f, 5);
     for (;;) {
```

```
                int g, len = sizeof(from);



                g = accept(f, (struct sockaddr *) &from, &len);
                if (g < 0) {
                        if (errno != EINTR)
                                syslog(LOG_ERR, "rlogind: accept: %m");
                        continue;
                }
                if (fork() == 0) {
                        close(f);
                        doit(g, &from);
                }
                close(g);
        }



        /*                                                      */
        /* Free answers after use                               */
        /*                                                      */
        freeaddrinfo(ans);
        exit(0);
    }
```