Code C présentés en Sys 1 - Amphi 8 Protocoles Applicatifs en TCP et UDP

Guillaume Didier

12 novembre 2024

1 Implémentation de `open_clientfd` et `open_listenfd`

Listing 2 et 1 d'après *Computer Systems: A Programmer's Perspective, 3rd edition*, Randal E. Bryant et Dave O'Hallaron.

```c
/* open_listenfd - Open and return a listening socket on port. This
 *     function is reentrant and protocol-independent.
 *
 *     On error, returns:
 *       -2 for getaddrinfo error
 *       -1 with errno set for other errors.
 */
int open_listenfd(const char *port) {
    struct addrinfo hints = {0}, *listp, *p;  int listenfd = -1, rc, optval = 1;

    hints.ai_socktype = SOCK_STREAM; // Accept connections
    hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG; // … on any local IP address
    hints.ai_flags |= AI_NUMERICSERV; // if the port number is already a number

    // Get a list of potential server addresses
    if (( rc = getaddrinfo(NULL, port, &hints, &listp)) != 0){
        fprintf(stderr, "getaddrinfo failed (port %s): %s\n", port, gai_strerror(rc));
        return -2;
    }

    for (p = listp; p; p = p->ai_next){ // Walk the list for one that we can bind to
        listenfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol); // Create a socket fd
        if (listenfd < 0) {
            continue; // Socket failed, try the next
        }

        /* Eliminates "Address already in use" error from bind */
        setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, (const void *)&optval, sizeof(int));

        // Bind the descriptor to the address
        if (bind(listenfd, p->ai_addr, p->ai_addrlen) == 0) {
            break; // Success
        }

        // not success, close and try next
        if (close(listenfd) < 0) { // Close failed, that's very wrong
            fprintf(stderr, "open_listenfd close failed: %s\n", strerror(errno));
            return -1;
        }
    }

    freeaddrinfo(listp); // Clean up

    if (!p) {
        return -1; // No address worked
    }

    // Make it a listening socket ready to accept connection requests
    if (listen(listenfd, LISTENQ) < 0) { // Error
        close(listenfd);
        return -1;
    }

    return listenfd; // Call accept on listenfd to serve your clients their cidre
}
```

Listing 1: Implémentation de `open_listenfd`

```c
/* open_clientfd - Open connection to server at <hostname, port> and
 *     return a socket descriptor ready for reading and writing. This
 *     function is reentrant and protocol-independent.
 *
 *     On error, returns:
 *       -2 for getaddrinfo error
 *       -1 with errno set for other errors.
 */
#define LISTENQ 1024 /* Second argument to listen() */
int open_clientfd(const char *hostname, const char *port) {
    int clientfd = -1, rc;  struct addrinfo hints = {0}, *listp, *p;

    hints.ai_socktype = SOCK_STREAM; // Open a connection
    hints.ai_flags = AI_NUMERICSERV; // ... using a numeric port arg.
    hints.ai_flags |= AI_ADDRCONFIG; // Recommended for connections

    // Get a list of potential server addresses
    if ((rc = getaddrinfo(hostname, port, &hints, &listp)) != 0) {
        fprintf(stderr, "getaddrinfo failed (%s:%s): %s\n", hostname, port, gai_strerror(rc));
        return -2;
    }

    // Walk the list for one that we can successfully connect to
    for (p = listp; p; p = p->ai_next) {
        // Create a socket fd
        clientfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
        if (clientfd < 0){
            continue; // Socket failed, try the next one
        }

        // Connect to the server
        if (connect(clientfd, p->ai_addr, p->ai_addrlen) != -1) {
            break; // Success
        }

        // Connect failed, try another
        if (close(clientfd) < 0) {
            fprintf(stderr, "open_clientfd: close failed: %s\n", strerror(errno));
            return -1;
        }
    }

    freeaddrinfo(listp); // Clean up

    if (!p) { // All connects failed
        return -1;
    }

    return clientfd; // The last connect succeeded - we can ask for cidre and get it !
}
```

Listing 2: Implémentation de `open_clientfd`