Ex-4

Implementation of select() and getpeername() system calls

The select system call monitors three sets of independent file descriptors. The file descriptors to be monitored are specified in the three file descriptor sets pointed by the second, third and fourth parameters to the select call.

Syntax:

int select(int maxfdp, fd\_set \* readset, fd\_set \* writeset, fd\_set \* exceptset, struct timeval \* timeout);

****maxfdp****: the total number of file descriptors being monitored, which is one greater than the maximum value of file descriptors in all file descriptor sets, because file descriptors are counted from 0;

****readfds****, writefds, exceptset: point to the set of descriptors corresponding to readable, writable, and exception events, respectively.

****timeout****: used to set the timeout of the select function, that is, tell the kernel how long to wait at maximum. timeout == NULL means wait for infinite time.

#include <sys/select.h>

#include <sys/time.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

int main() {

fd\_set rd;

struct timeval tv;

int err;

FD\_ZERO( & rd);

FD\_SET(0, & rd);

tv.tv\_sec = 5;

tv.tv\_usec = 0;

err = select(1, & rd, NULL, NULL, & tv);

if (err == 0) // Timeout

{

printf("select timeout!\n");

} else if (err == -1) // Failure

{

printf("fail to select!\n");

} else // Successful

{

printf("data is available!\n");

}

return 0;

}

Getpeername()

getpeername - get name of connected peer

**SYNOPSIS**

getpeername(s, name, namelen)

int s;

struct sockaddr \*name;

int \*namelen;

**DESCRIPTION**

Getpeername returns the name of the peer connected to socket s. The namelen parameter should be initialized to indicate the amount of

space pointed to by name. On return it contains the actual size of the name returned (in bytes). The name is truncated if the buffer pro-

vided is too small.

**DIAGNOSTICS**

A 0 is returned if the call succeeds, **-1** if it fails.

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

int main() {

int sockfd;

struct sockaddr\_in server\_addr, peer\_addr;

socklen\_t peer\_addr\_len;

char peer\_ip[INET\_ADDRSTRLEN];

int peer\_port;

// Create a socket

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0) {

perror("socket");

return 1;

}

// Specify server address

memset(&server\_addr, 0, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(12345);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

// Connect to the server

if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("connect");

close(sockfd);

return 1;

}

// Get peer address

peer\_addr\_len = sizeof(peer\_addr);

if (getpeername(sockfd, (struct sockaddr \*)&peer\_addr, &peer\_addr\_len) < 0) {

perror("getpeername");

close(sockfd);

return 1;

}

// Convert the IP address to a string

inet\_ntop(AF\_INET, &peer\_addr.sin\_addr, peer\_ip, sizeof(peer\_ip));

peer\_port = ntohs(peer\_addr.sin\_port);

// Print peer address

printf("Connected to %s:%d\n", peer\_ip, peer\_port);

// Close the socket

close(sockfd);

return 0;

}