Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных систем и технологий»

**Семейство протоколов TCP/IP. Сокеты в UNIX и работа с ними**

Лабораторная работа №11 по учебной дисциплине «Операционные системы»

По направлению 09.03.01 «Информатика и вычислительная техника»

Отчёт

Принял преподаватель:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Выполнил студент группы 9091:

\_\_\_\_\_\_\_ Ковалев А.Д.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

Великий Новгород

2021

**Цель работы:** познакомиться с механизмами работы сокетов в UNIX.

**Исходный текст программы**

**Client.c**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/time.h>

#include <string.h>

#include <stdio.h>

#include <errno.h>

#include <unistd.h>

#include <stdlib.h>

#include <time.h>

#define UDP\_BUFFER\_LEN 65507

void fill\_random\_nums(int \*nums, int n, int min, int max)

{

srand((unsigned)(time(0)));

for (int i = 0; i < n; i++)

{

nums[i] = min + rand() % (max - min + 1);

}

}

void print\_nums(int \*nums, int c)

{

for (int i = 0; i < c; i++)

{

printf("%d ", nums[i]);

}

printf("\n");

}

long timedifference(struct timeval t0, struct timeval t1)

{

return (t1.tv\_sec - t0.tv\_sec) \* 1000000 + (t1.tv\_usec - t0.tv\_usec);

}

int main(int argc, char \*argv[])

{

size\_t max\_n = UDP\_BUFFER\_LEN / sizeof(int);

int n = 0, min, max;

while (n <= 0 || n > max\_n)

{

printf("Enter array length(<=%ld): ", max\_n);

scanf("%d", &n);

}

printf("Enter minimum: ");

scanf("%d", &min);

printf("Enter maximum: ");

scanf("%d", &max);

int sockfd;

char sendline[n \* sizeof(int) / sizeof(char)], recvline[n \* sizeof(int) / sizeof(char)];

struct sockaddr\_in servaddr, cliaddr;

sockfd = socket(PF\_INET, SOCK\_DGRAM, 0);

bzero(&cliaddr, sizeof(cliaddr));

cliaddr.sin\_family = AF\_INET;

cliaddr.sin\_port = htons(0);

cliaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

bind(sockfd, (struct sockaddr \*)&cliaddr, sizeof(cliaddr));

char \*sep = strchr(argv[1], ':');

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(atoi(sep + 1));

sep[0] = 0;

inet\_aton(argv[1], &servaddr.sin\_addr);

fill\_random\_nums((int \*)sendline, n, min, max);

print\_nums((int \*)sendline, n);

struct timeval start, end;

int s = sendto(sockfd, sendline, sizeof(sendline), 0, (struct sockaddr \*)&servaddr, sizeof(servaddr));

gettimeofday(&start, 0);

recvfrom(sockfd, recvline, sizeof(recvline), 0, (struct sockaddr \*)NULL, NULL);

gettimeofday(&end, 0);

print\_nums((int \*)recvline, n);

printf("Response time: %ldns\n", timedifference(start, end));

close(sockfd);

}

**Server.c**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <string.h>

#include <stdio.h>

#include <errno.h>

#include <unistd.h>

#include <limits.h>

#include <stdlib.h>

#ifdef \_\_linux\_\_

#define UDP\_BUFFER\_LEN 65507

#endif

int compare\_ints(const void \*a, const void \*b)

{

return (\*((int \*)b) - \*((int \*)a));

}

int main(int argc, char \*argv[])

{

size\_t maxlen = UDP\_BUFFER\_LEN;

int sockfd;

char line[maxlen];

struct sockaddr\_in servaddr, cliaddr;

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(atoi(argv[1]));

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

sockfd = socket(PF\_INET, SOCK\_DGRAM, 0);

if (bind(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)

{

servaddr.sin\_port = 0;

if (bind(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)

{

perror(NULL);

close(sockfd);

exit(1);

}

}

socklen\_t servlen = sizeof(servaddr);

getsockname(sockfd, (struct sockaddr \*)&servaddr, &servlen);

printf("Listening on port: %d\n", ntohs(servaddr.sin\_port));

while (1)

{

socklen\_t clilen = sizeof(cliaddr);

int n = recvfrom(sockfd, line, maxlen, 0, (struct sockaddr \*)&cliaddr, &clilen);

qsort(line, n \* sizeof(char) / sizeof(int), sizeof(int), compare\_ints);

sendto(sockfd, line, n, 0, (struct sockaddr \*)&cliaddr, clilen);

}

}

**Вывод:** В процессе выполнения лабораторной работы я на практике познакомился с работой сокетов в UNIX.