1. Билет №???

Socket программы

1.1. Socketpair

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>

#define BUF_SIZE 50
#define CHILD_NUM 3

int main(int argc, char ** argv)
{
    int fd[2];
    char buf[BUF_SIZE];
    pid_t child_pid[CHILD_NUM];

    if (socketpair(AF_UNIX, SOCK_DGRAM, 0, fd) < 0)
    {
        perror("socketpair() failed");
        exit(1);
    }

    for (size_t i = 0; i < CHILD_NUM; i++)
    {
        if ((child_pid[i] = fork()) == -1)
        {
            perror("Can't fork\n");
            exit(1);
        }
```

```
30          if (child_pid[i] == 0)
31          {
32              //close(fd[1]);
33              sprintf(buf, "%d", getpid());
34              write(fd[0], buf, sizeof(buf));
35              printf("Child wrote %s\n", buf);
36              //sleep(1);
37              read(fd[0], buf, sizeof(buf));
38              printf("Child %d read %s\n", getpid(), buf);
39             // close(fd[0]);
40
41              return EXIT_SUCCESS;
42          }
43          else
44          {
45              //close(fd[0]);
46              read(fd[1], buf, sizeof(buf));
47              printf("Parent read %s\n", buf);
48              sprintf(buf, "%s %d", buf, getpid());
49              write(fd[1], buf, sizeof(buf));
50              printf("Parent wrote %s\n", buf);
51          }
52      }
53
54      //close(fd[0]);
55      //close(fd[1]);
56
57      return EXIT_SUCCESS;
58  }
```

1.2. AF_UNIX + SOCK_DGRAM + без bind у клиента

client

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

```c
#include <sys/socket.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <sys/un.h>

#define BUF_SIZE 50

int main(int argc, char ** argv)
{
    char buf[BUF_SIZE];

    if (argc != 2)
    {
        perror("args not enough\n");
        return EXIT_FAILURE;
    }
    sprintf(buf, "%d %s", getpid(), argv[1]);

    int sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
    if (sockfd == -1)
    {
        perror("socket() failed\n");
        return EXIT_FAILURE;
    }

    struct sockaddr sa;
    sa.sa_family = AF_UNIX;
    strcpy(sa.sa_data, "socket.soc");

    if (sendto(sockfd, buf, sizeof(buf), 0, &sa, sizeof(sa)) == -1)
    {
        perror("sendto() failed\n");
        close(sockfd);
        return EXIT_FAILURE;
    }
```

```
40
41      close(sockfd);
42
43      return EXIT_SUCCESS;
44  }
```

server

```
1   #include <sys/types.h>
2   #include <sys/socket.h>
3   #include <stdlib.h>
4   #include <stdio.h>
5   #include <errno.h>
6   #include <unistd.h>
7   #include <string.h>
8
9   #define BUF_SIZE 50
10
11  int sockfd;
12
13  void signal_handler(int signal)
14  {
15      printf("\nCaught signal = %d\n", signal);
16      unlink("socket.soc");
17      close(sockfd);
18      printf("\nServer exiting.\n");
19      exit(0);
20  }
21
22  int main()
23  {
24    if ((signal(SIGINT, signal_handler) == SIG_ERR)) {
25          perror("Can't attach handler\n");
26          return EXIT_FAILURE;
27      }
28
29    char buf[BUF_SIZE];
```

```
30
31    sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
32    if (sockfd == -1)
33    {
34      perror("socket() failed");
35          return EXIT_FAILURE;
36    }
37
38    struct sockaddr sa;
39    sa.sa_family = AF_UNIX;
40    strcpy(sa.sa_data, "socket.soc");
41
42    if (bind(sockfd, &sa, sizeof(sa)) < 0)
43    {
44      perror("bind() failed");
45      unlink("socket.soc");
46      close(sockfd);
47      return EXIT_FAILURE;
48    }
49
50    int bytes;
51    while (1)
52    {
53      bytes = recvfrom(sockfd, buf, sizeof(buf), 0, NULL, NULL);
54      if (bytes == -1)
55      {
56        perror("recvfrom() failed");
57        unlink("socket.soc");
58        close(sockfd);
59        return EXIT_FAILURE;
60      }
61
62      printf("\nreceived: %s\n", buf);
63    }
64
65    return EXIT_SUCCESS;
66  }
```

1.3. AF_UNIX + SOCK_DGRAM + bind у клиента

client

```
1  #include <sys/types.h>
2  #include <sys/socket.h>
3  #include <stdlib.h>
4  #include <stdio.h>
5  #include <errno.h>
6  #include <unistd.h>
7  #include <string.h>
8  #include <sys/un.h>
9
10 #define BUF_SIZE 50
11
12 int main(int argc, char **argv)
13 {
14     char buf[BUF_SIZE];
15
16     if (argc != 2)
17     {
18         perror("args not enough\n");
19         return EXIT_FAILURE;
20     }
21     sprintf(buf, "%d %s", getpid(), argv[1]);
22
23     int sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
24     if (sockfd == -1)
25     {
26         perror("socket() failed\n");
27         return EXIT_FAILURE;
28     }
29
30     struct sockaddr sa;
31     sa.sa_family = AF_UNIX;
32     strcpy(sa.sa_data, "socket.soc");
```

```
33      socklen_t len = sizeof(sa);
34
35      char name[20];
36      sprintf(name, "%d.soc", getpid());
37
38      struct sockaddr ca;
39      ca.sa_family = AF_UNIX;
40      strcpy(ca.sa_data, name);
41      if (bind(sockfd, &ca, sizeof(ca)) == -1)
42      {
43          perror("bind() failed\n");
44          close(sockfd);
45          return EXIT_FAILURE;
46      }
47
48      if (sendto(sockfd, buf, sizeof(buf), 0, &sa, len) == -1)
49      {
50          perror("sendto() failed\n");
51          unlink(name);
52          close(sockfd);
53          return EXIT_FAILURE;
54      }
55      if (recvfrom(sockfd, buf, sizeof(buf), 0, NULL, NULL) == -1) //&sa, &
            len
56      {
57          perror("recvfrom() failed\n");
58          unlink(name);
59          close(sockfd);
60          return EXIT_FAILURE;
61      }
62      printf("\nreceived: %s\n", buf);
63      unlink(name);
64      close(sockfd);
65      return EXIT_SUCCESS;
66  }
```

server

```
1  #include <sys/types.h>
2  #include <sys/socket.h>
3  #include <stdlib.h>
4  #include <stdio.h>
5  #include <errno.h>
6  #include <unistd.h>
7  #include <string.h>
8
9  #define BUF_SIZE 50
10
11 int sockfd;
12
13 void signal_handler(int signal)
14 {
15     printf("\nCaught signal = %d\n", signal);
16     unlink("socket.soc");
17     close(sockfd);
18     printf("\nServer exiting.\n");
19     exit(0);
20 }
21
22 int main(int argc, char **argv)
23 {
24     if ((signal(SIGINT, signal_handler) == SIG_ERR))
25     {
26         perror("Can't attach handler\n");
27         return EXIT_FAILURE;
28     }
29     char buf[BUF_SIZE];
30     sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
31
32     if (sockfd == -1)
33     {
34         perror("socket() failed");
35         return EXIT_FAILURE;
36     }
```

```
37
38      struct sockaddr sa;
39      sa.sa_family = AF_UNIX;
40      strcpy(sa.sa_data, "socket.soc");
41
42      if (bind(sockfd, &sa, sizeof(sa)) == -1)
43      {
44          perror("bind() failed");
45          unlink("socket.soc");
46          close(sockfd);
47          return EXIT_FAILURE;
48      }
49
50      int bytes;
51      while (1)
52      {
53          struct sockaddr ca;
54          socklen_t len = sizeof(ca);
55
56          bytes = recvfrom(sockfd, buf, sizeof(buf), 0, &ca, &len);
57          if (bytes == -1)
58          {
59              perror("recvfrom() failed");
60              unlink("socket.soc");
61              close(sockfd);
62              return EXIT_FAILURE;
63          }
64          printf("\nreceived: %s\n", buf);
65          sprintf(buf, "%s %d", buf, getpid());
66
67          if (sendto(sockfd, buf, sizeof(buf), 0, &ca, len) == -1)
68          {
69              perror("sendto() failed");
70              unlink("socket.soc");
71              close(sockfd);
72              return EXIT_FAILURE;
73          }
74          printf("sent: %s\n", buf);
```

```
75        }
76        return EXIT_SUCCESS;
77    }
```