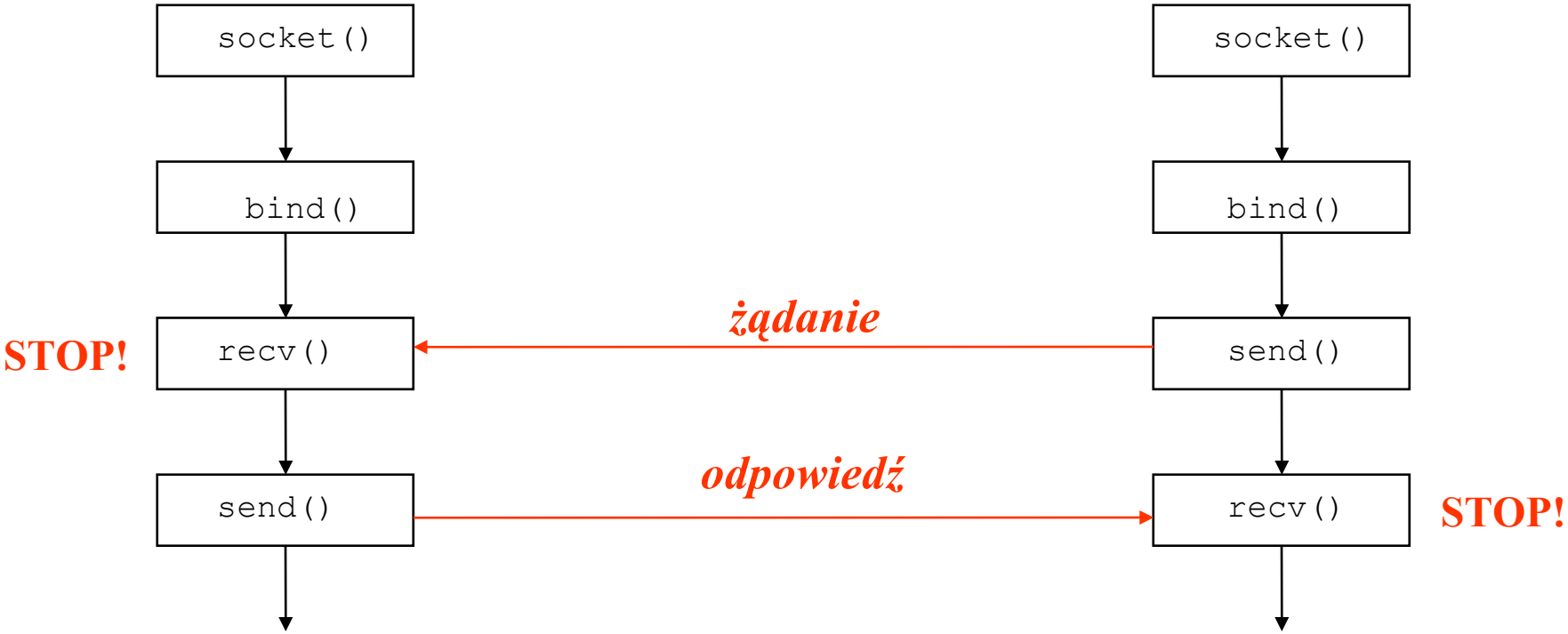
Gniazda BSD

komunikacja bezpołączeniowa

Użycie gniazd w transmisji bezpołączeniowej



recvfrom()

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen);
```

gdzie:

sockfd

- otwarte i związane gniazdo, z którego podejmiemy próbę odebrania komunikatu

buf

- wskaźnik na bufor, w którym zostanie umieszczony odebrany komunikat

len

- maksymalna dopuszczalna długość odebranego komunikatu

flags

- MSG_DONTWAIT – nie czekaj na nadejście komunikatu
- MSG_PEEK – odbierz komunikat, ale go nie usuwaj z kolejki

src_addr

- struktura gniazda, która zostanie wypełniona danymi nadawcy komunikatu

addrlen

- dana, która odbierze informację o długości struktury gniazda nadawcy

wynik:

faktyczna długość odebranego komunikatu albo -1 (→ errno)

sendto()

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

ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
               const struct sockaddr *dest_addr, socklen_t addrlen);
```

gdzie:

sockfd

- otwarte gniazdo, przez które podejmiemy próbę wysłania komunikatu

buf

- wskaźnik na bufor zawierający nadawany komunikat

len

- długość nadawanego komunikatu

flags

- MSG_DONTROUTE – wyślij wiadomość rozgłoszeniową w sieci lokalnej

dest_addr

- dane odbiorcy komunikatu

addrlen

- długości struktury danych odbiorcy

wynik:

faktyczna długość nadanego komunikatu albo -1 (→ errno)

Napiszemy tandem programowy służący do wymiany prostych komunikatów tekstowych:

- program *listener* będzie nasłuchiwał na wskazanym porcie komunikatów nadchodzących od *speakera*
- *listener* zakończy pracę po odebraniu specjalnego komunikatu
- *speaker* będzie wysyłał dowolne komunikaty do *listenera* identyfikowanego adresem IP i numerem portu

listener.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>

void die(char *msg) {
   perror(msg);
   exit(1);
}
```

```
int main(int argc, char *argv[]) {
   char line[128];
   int  listener_port;
   int  sock;
   struct sockaddr_in HERE;
   struct sockaddr_in THERE;
   int  len, slen;

   if(argc != 2) {
      fprintf(stderr,
              "Usage: %s listener_port\n", argv[0]);
      return 1;
   }
```

```
listener_port = strtol(argv[1], NULL, 10);
printf("Listening from port %d\n", listener_port);
sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sock == -1)
   die("socket");
HERE.sin_family = AF_INET;
HERE.sin_port = htons(listener_port);
HERE.sin_addr.s_addr = htonl(INADDR_ANY);
memset(HERE.sin_zero, '\0', 8);

if(bind(sock,(struct sockaddr *)&HERE,
    sizeof(HERE)) == -1) die("bind");
```

```
    while((len = recvfrom(sock, line, sizeof(line),
                          0,
                          (struct sockaddr*)&THERE,
                          &slen)) != -1) {
        line[len] = '\0';
        printf("Message [%s] received from %s\n",
                line, inet_ntoa(THERE.sin_addr));
        if(strcmp(line,"END") == 0) {
           puts("Terminating message received.");
           break;
        }
   }
   close(sock);
   return 0;
}
```

speaker.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>

void die(char *msg) {
   perror(msg);
   exit(1);
}
```

```
int main(int argc, char *argv[]) {
   char line[128];
   char listener_addr[128];
   int  listener_port;
   int  sock;
   struct sockaddr_in THERE;
   struct hostent *hent;

   if(argc != 3) {
      fprintf(stderr,
      "Usage: %s listener_addr listener_port\n",
      argv[0]);
      return 1;
   }
```

```
strcpy(listener_addr, argv[1]);
listener_port = strtol(argv[2], NULL, 10);
printf("Talking to: %s:%d\n",
        listener_addr, listener_port);
sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sock == -1)
   die("socket");
hent = gethostbyname(listener_addr);
if(hent == NULL)
   die("gethostbyname");
```

```
THERE.sin_family = AF_INET;
THERE.sin_port=htons(listener_port);
THERE.sin_addr=*((struct in_addr*)hent->h_addr);
memset(THERE.sin_zero, '\0', 8);

puts("Enter messages, Ctrl-D to stop...");
while(fgets(line, sizeof(line), stdin)) {
   line[strlen(line) - 1] = '\0';
   if(sendto(sock, line, strlen(line), 0,
     (struct sockaddr *)&THERE, sizeof(THERE))
       == -1)
         die("sendto");
      printf("Message [%s] sent\n", line);
}
```

```
   if(sendto(sock, "END", 3, 0,
      (struct sockaddr *)&THERE, sizeof(THERE))
      == -1) die("sendto");
   close(sock);
   puts("Terminating message sent.");
   return 0;
}
```