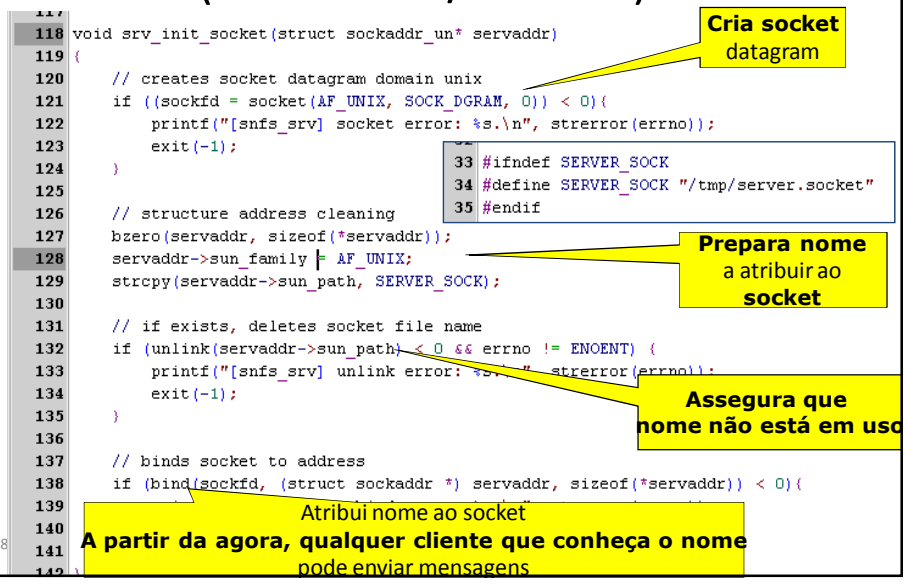
Exemplo de Sockets UNIX/Datagram: Projecto SO (SNFS)

SNFS: Inicialização do servidor (snfs-server/server.c)




```
118 void srv_init_socket(struct sockaddr_un* servaddr)
119 {
120     // creates socket datagram domain unix
121     if ((sockfd = socket(AF_UNIX, SOCK_DGRAM, 0)) < 0){
122         printf("[snfs_srv] socket error: %s.\n", strerror(errno));
123         exit(-1);
124     }
125
126     // structure address cleaning
127     bzero(servaddr, sizeof(*servaddr));
128     servaddr->sun_family = AF_UNIX;
129     strcpy(servaddr->sun_path, SERVER_SOCK);
130
131     // if exists, deletes socket file name
132     if (unlink(servaddr->sun_path) < 0 && errno != ENOENT) {
133         printf("[snfs_srv] unlink error: %s.\n", strerror(errno));
134         exit(-1);
135     }
136
137     // binds socket to address
138     if (bind(sockfd, (struct sockaddr *) servaddr, sizeof(*servaddr)) < 0){
139
140
141
142 }
```

```
33 #ifndef SERVER_SOCK
34 #define SERVER_SOCK "/tmp/server.socket"
35 #endif
```

SNFS: Inicialização do servidor (snfs-server/server.c)

```
118 void srv_init_socket(struct sockaddr_un* servaddr)
119 {
120     // creates socket datagram domain unix
121     if ((sockfd = socket(AF_UNIX, SOCK_DGRAM, 0)) < 0){
122         printf("[snfs_srv] socket error: %s.\n", strerror(errno));
123         exit(-1);
124     }
125
126     // structure address cleaning
127     bzero(servaddr, sizeof(*servaddr));
128     servaddr->sun_family = AF_UNIX;
129     strcpy(servaddr->sun_path, SERVER_SOCK);
130
131     // if exists, deletes socket file name
132     if (unlink(servaddr->sun_path) < 0 && errno != ENOENT) {
133         printf("[snfs_srv] unlink error: %s.\n", strerror(errno));
134         exit(-1);
135     }
136
137     // binds socket to address
138     if (bind(sockfd, (struct sockaddr *) servaddr, sizeof(*servaddr)) < 0){
139
140
141
```

```
33 #ifndef SERVER_SOCK
34 #define SERVER_SOCK "/tmp/server.socket"
35 #endif
```

Cria socket datagram

Prepara nome a atribuir ao **socket**

Assegura que nome não está em uso

Atribui nome ao socket
A partir da agora, qualquer cliente que conheça o nome pode enviar mensagens

SNFS: Cliente (snfs_api.c)

```
63 static int remote_call(snfs_msg_req_t *req, int reqsz, snfs_msg_res_t *res,
64     int ressz, int to_all_servers)
65 {
66     int status;
67
68     req->sn = (ne
69
70     // sends the request to the server
71     status = sendto(Cli_sock, (void*)req, reqsz, 0,
72         (struct sockaddr *)&Serv_addr, sizeof(Serv_addr));
73     if (status
74       //printf
75         printf
76         return -1;
77     }
78
79     // waits for an answer
80     status = recvfrom(Cli_sock, res, ressz, 0, NULL, NULL);
81     if (status < 0) {
82         printf("[snfs_api] recvfrom error: %s.\n", strerror(errno));
83         return -1;
84     }
85     if (status == 0) {
86         printf("[snfs_api] server is closed.\n");
87         return -1;
88     }
89     if (res->sn != req->sn) {
90       printf("[snfs_api] received response to wrong request serial number.\n");
91       return -1;
92     }
93
94     return status;
95 }
```

Cliente um socket (Cli_sock).
remote_call envia pedido 'req' ao(s) servidor(es) e **espera pela resposta (que coloca em 'res')**

Envia o pedido

Serv_addr é variável global com o nome do servidor.
Inicializada na função snfs_init.

Espera pela resposta

Cofirma que nº de série da resposta é o mesmo do pedido

8/28/2003

SNFS: Cliente (snfs_api.c)

```
63 static int remote_call(snfs_msg_req_t *req, int reqsz, snfs_msg_res_t *res,
64    int ressz, int to_all_servers)
65 {
66    int status;
67
68    req->sn = (ne
69
70    // sends the request to the server
71    status = sendto(Cli_sock, (void*)req, reqsz, 0,
72       (struct sockaddr *)&Serv_addr, sizeof(Serv_addr));
73    if (status
74      //printi
75       printf
76       return -1;
77    }
78
79    // waits for an answer
80    status = recvfrom(Cli_sock, res, ressz, 0, NULL, NULL);
81    if (status < 0) {
82       printf("[snfs_api] recvfrom error: %s.\n", strerror(errno));
83       return -1;
84    }
85    if (status == 0) {
86       printf("[snfs_api] server is closed.\n");
87       return -1;
88    }
89    if (res->sn != req->sn) {
90      printf("[snfs_api] received response to wrong request serial number.\n");
91      return -1;
92    }
93
94    return status;
95 }
```

Cliente um socket (Cli_sock).
remote_call envia pedido 'req' ao(s) servidor(es) e **espera pela resposta (que coloca em 'res')**

Envia o pedido

Serv_addr é variável global com o nome do servidor.
Inicializada na função snfs_init.

Espera pela resposta

Cofirma que nº de série da resposta é o mesmo do pedido

8/28/2003