Local Sockets

Carlos Alberto Santos de Souza
201110006250
Sistemas Operacionais 2016-1

1. Entendendo IPC

- *Interprocess Communication* são formas de comunicação entre processos.
- Os processos podem enviar ou receber porções de dados através de variáveis compartilhadas, arquivos compartilhados ou até mesmo através de sockets que usam uma rede.

2. Socket

- É uma ferramenta do SO que permite a comunicação entre processos tanto para processos residentes num mesmo computador quanto para processos em computadores diferentes.
- Nesta apresentação trabalharemos com *sockets locais* que representam a comunicação ente processos numa mesma máquina física.

3. Como funciona?

- Sockets funcionam como compartimentos que enviam/recebem mensagens.
- Eles são vinculados à programas que leem os dados que são recebidos, como também podemos escrever nestes sockets afim de que outros programas possam ler estes dados.
- No código que será mostrado usamos o padrão de cliente e servidor. Onde o Cliente envia dados e o lado servidor lê esses dados.

4. Bibliotecas

- <sys/socket.h> -> usada para a criar e manipular os sockets.
- <sys/un.h> -> possui a struct do sockaddr_un para armazenar as informações do socket.

5. Funções importantes

- int socket(int domain, int type, int protocol);
- int bind(int socket, const struct sockaddr *address, socklen_t address_len);
- int listen(int socket, int backlog);
- int accept (int socket, struct sockaddr *address, socklen_t *address_len);
- int connect(int *socket*, const struct sockaddr **address*, socklen_t *address_len*);

6. Implementação

Função main do servidor

```
int main (int argc, char* const argv[]) {
  const char* const socket_name = argv[1];

 int socket_fd; struct sockaddr_un name;
  int client_sent_quit_message;
 /* Cria o socket.  */
  socket_fd = socket (PF_LOCAL, SOCK_STREAM, 0);
  /* Indica que este é um servidor  */
  name.sun_family = AF_LOCAL;
  strcpy (name.sun_path, socket_name);
  bind (socket_fd, &name, SUN_LEN (&name));
  /* Ouve a conexão  */
  listen (socket_fd, 5);
/* Aceite conexões repetidamente, execute todas as tarefas para cada cliente.
  Continue até o cliente mandar uma mensagem com o nome "quit".  */
  do {
    struct sockaddr client_name;
    socklen_t client_name_len;
    int client_socket_fd;
   /* Aceite uma conexão.  */
    client_socket_fd = accept (socket_fd, &client_name, &client_name_len);
    /* Manipule uma conexão  */
    client_sent_quit_message = server (client_socket_fd);
    /* Feche uma conexão.  */
    close (client_socket_fd);
} while (!client_sent_quit_message);

 /* Remova o arquivo de socket.  */
  close (socket_fd);
  unlink (socket_name);
 return 0;
}
```

6. Implementação

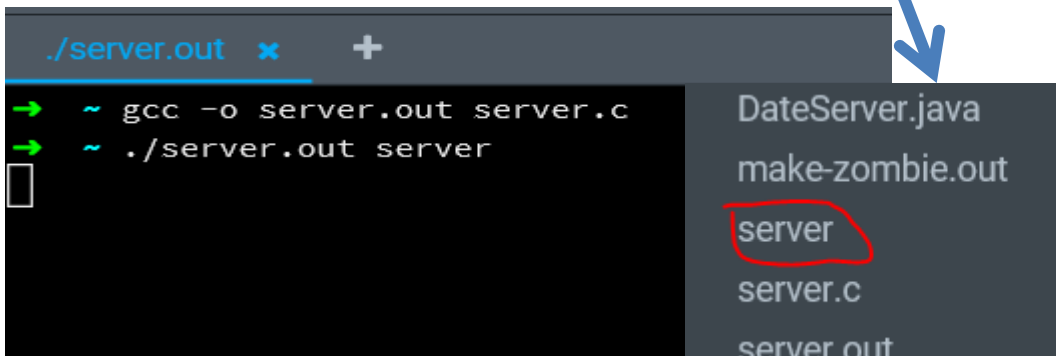
Função server() do servidor

```
int server (int client_socket) {
  while (1) {
    int length;
    char* text;
/* Primeiro, leia o tamanho da mensagem do socket.
Se o valor for 0, o cliente fecha a conexão.  */
  if (read (client_socket, &length, sizeof (length)) == 0){
      return 0;
  }
          /* Alocar um buffer para manipular um texto.  */
          text = (char*) malloc (length);
          /* Leia o texto vc mesmo, e implima-o.  */
         read (client_socket, text, length);
          printf ("%s\n", text);

      /* Se o cliente mandar a mensagem "quit", nós terminaremos.  */
      if (!strcmp(text, "quit")){
        return 1;
      }
       /* Libere o buffer.  */
          free (text);

    }
}
```

File descriptor



6. Implementação

função main do cliente

```c
int main (int argc, char* const argv[]) {
  const char* const socket_name = argv[1];
  const char* const message = argv[2];
  int socket_fd;
   struct sockaddr_un name;
/* Cria o socket.  */
   socket_fd = socket (PF_LOCAL, SOCK_STREAM, 0);
   /* Guarda o nome do servidor no endereço do socket.  */
   name.sun_family = AF_LOCAL;
   strcpy (name.sun_path, socket_name);
   /* Conecta o socket.  */
   connect (socket_fd, &name, SUN_LEN (&name));
   /* Escreve o texto no arquivo descritor do socket */
   write_text (socket_fd, message);
   close (socket_fd);
   return 0;
}
```

```c
/* Escreve um TEXTO no socket nomeado pelo pelo descrito
void write_text (int socket_fd, const char* text) {
/* Escreve o numero de bytes que estão no array incluin
 int length = strlen (text) + 1;
 write (socket_fd, &length, sizeof (length));
 /* Escreve a string.  */
 write (socket_fd, text, length);
}
```

```
./server.out   ~
➜  ~ gcc -o client.out client.c
➜  ~ ./client.out server "Olá"
➜  ~
```

```
./server.out   ~
➜  ~ gcc -o server.out server.c
➜  ~ ./server.out server
Olá
```

6. Referências

- **Advanced Linux Programming.** Disponível em: http://advancedlinuxprogramming.com/alp-folder/. Acesso em: 16 ago. 2016.

- **The Single UNIX .** Disponível em: http://pubs.opengroup.org/onlinepubs/7908799/xns/listen.html. Acesso em: 18 ago. 2016.

Gist (gitHub)

- https://github.com/CarlosAlbertoUFS/SOProjects/tree/master/Trabalho03