Bedeutung und Aufbau von SunRPC (Praxisteil)

Johannes Bauer

KVBK - Konzepte von Betriebssystemkomponenten

14. Dezember 2004

Problemstellung und Ziel

- Aufbau eines Chat-Systems über SunRPC
- Ein Server, viele Clients
- Clients sehen jeweils Nachrichten aller eingeloggten Teilnehmer

Probleme mit SunRPC

- Client soll nur Client sein
- Wie bekommt der Client die Nachrichten dann?
- (Schlechte) Lösung: Polling

Semantik des Chatservers

Einloggen auf dem Chatserver

- Prozedur 1: LOGIN
- Parameter
  1. char Nickname[256];
- Resultat
  1. int Resultat;
     ERROR_INSANE_NICKNAME
     ERROR_ALREADY_LOGGED_IN
     LOGIN_SUCCESS

Semantik des Chatservers

Abrufen von Nachrichten

- Prozedur 2: FETCH
- Parameter
  1. char Nickname[256];
- Resultat
  1. int Resultat;
     ERROR_INSANE_NICKNAME
     ERROR_NOT_LOGGED_IN
     ERROR_NO_MESSAGES
     FETCH_SUCCESS
  2. char Line[256];

Semantik des Chatservers

Senden von Nachrichten

- Prozedur 3: SEND
- Parameter
  1. char Nickname[256];
  2. char Line[256];
- Resultat
  1. int Resultat;
     ERROR_INSANE_NICKNAME
     ERROR_INSANE_MESSAGE
     ERROR_NOT_LOGGED_IN
     SEND_SUCCESS

Programmierung

Holen der ersten Dateien

- `/home/cip/2003/sijsbaue/kvbkchat/grabfiles 1`

Programmierung

Holen der ersten Dateien

- `/home/cip/2003/sijsbaue/kvbkchat/grabfiles 1`
- ```
  cd kvbk_chat
  ls -l
  -rw-r--r-- joe users 509 Chat.x
  -rw-r--r-- joe users 893 Makefile
  -rw-r--r-- joe users 396 Definitions.h
  ```
  ```

Programmierung

Holen der ersten Dateien

- `/home/cip/2003/sijsbaue/kvbkchat/grabfiles 1`
- ```
  cd kvbk_chat
  ls -l
  -rw-r--r-- joe users 509 Chat.x
  -rw-r--r-- joe users 893 Makefile
  -rw-r--r-- joe users 396 Definitions.h
  ```
- `cat Chat.x`
  ```

Programmierung

Holen der ersten Dateien

- cat Chat.x

```
...
program CHAT_PROG {
    version PROGRAM_VERS {
        Login_Resultat LOGIN(Login_Parameter) = 1;
        Fetch_Resultat FETCH(Fetch_Parameter) = 2;
        Send_Resultat SEND(Send_Parameter) = 3;
    } = 1;
} = 222111;
```

Programmierung

Erstellen des RPC-Gerüsts

- ```
  make rpcgen
  rpcgen -a Chat.x
  ```
  ```

Programmierung

Erstellen des RPC-Gerüsts

- ```
  make rpcgen
  rpcgen -a Chat.x
  ```
- ```
  ls -l
  -rw-r--r-- joe users 1,2K Chat_client.c
  -rw-r--r-- joe users 1,3K Chat_clnt.c
  -rw-r--r-- joe users 2,5K Chat.h
  -rw-r--r-- joe users  672 Chat_server.c
  -rw-r--r-- joe users 2,6K Chat_svc.c
  -rw-r--r-- joe users  509 Chat.x
  -rw-r--r-- joe users 1,5K Chat_xdr.c
  -rw-r--r-- joe users  396 Definitions.h
  -rw-r--r-- joe users  893 Makefile
  -rw-r--r-- joe users 1,1K Makefile.Chat
  ```

Programmierung

Editieren des Clientprogramms (Chat_client.c)

```
1   result_1 = login_1(&login_1_arg, clnt);
2   if (result_1 == (Login_Resultat *) NULL) {
3       clnt_perror (clnt, "call failed");
4   }
5   result_2 = fetch_1(&fetch_1_arg, clnt);
6   if (result_2 == (Fetch_Resultat *) NULL) {
7       clnt_perror (clnt, "call failed");
8   }
9   result_3 = send_1(&send_1_arg, clnt);
10  if (result_3 == (Send_Resultat *) NULL) {
11      clnt_perror (clnt, "call failed");
12  }
```

Programmierung

Editieren des Clientprogramms (Chat_client.c)

```
1   strcpy(login_1_arg.Nickname, "Johannes");
2
3   result_1 = login_1(&login_1_arg, clnt);
4   if (result_1 == (Login_Resultat *) NULL) {
5       clnt_perror (clnt, "call failed");
6   }
7
8   if (result_1.Resultat != LOGIN_SUCCESS) {
9       printf("Fehler beim Login: %s\n", Chat_ErrMsgs[
            result_1.Resultat]);
10      exit(1);
11  }
```

Programmierung

Editieren des Clientprogramms (Chat_client.c)

```
1       char Buffer[256];
2
3       if (fgets(Buffer, 255, stdin) != NULL) {
4           // Eingelesener Text liegt im Buffer
5           //     -> Jetzt lossenden!
6       }
```

Programmierung

Test des Programms

- `make`

Programmierung

Test des Programms

- `make`
- `./client faui05a.informatik.uni-erlangen.de`

```
Hallo du da!
(Johannes): Hallo du da!
Wow, ich kann mich selber lesen!
(Johannes): Wow, ich kann mich selber lesen!
```