plug-in — дополнение

middle man — посредник

Глава 11

Лёгкий IRC

Вот и пришло время для приложения. До сих пор мы видели только разрозненные части. Мы видели как писать последовательный код, как порождать процессы, как регистрировать процессы и т.д. Теперь мы соберём все эти части в одно работающее целое.

В этой главе мы создадим простую IRC-подобную программу. Мы не будем придерживаться настоящего IRC протокола. Вместо этого мы придумаем наш собственный совершенно другой и не совместимый ни с чем протоколHYPERLINK \l "id.4c3e30fd4f52"[1]HYPERLINK \l "id.4c3e30fd4f52". С точки зрения пользователя наша программа *является* реализацией IRC, хотя нижележащая реализация гораздо проще, чем это могло бы ожидаться, т. к. мы используем сообщения Эрланга как основу для межпроцессорных сообщений. Это полностью устраняет разбор сообщений и значительно упрощает дизайн.

Наша программа является программой на *чистом* Эрланге, которая совершенно не использует библиотеки OTP и минимально использует стандартные библиотеки. Так что, к примеру, у неё полностью самодостаточная клиент-серверная архитектура и форма восстановления после ошибок, основанная на явном манипулировании связей. Причина неиспользования библиотек в том, что я хочу вносить вам на рассмотрение по одной концепции за раз и показывать, что мы можем достичь с одним языком и минимальным использованием библиотек. Мы будем писать код как набор компонентов. Каждый компонент прост, но вместе они работают достаточно сложно. Мы можем заставить большую часть сложностей убраться, используя библиотеки OTP, так что позднее в этой книге мы покажем более правильные способы организации кода, основанном на общий библиотеках OTP для построения деревьев клиент-серверов и наблюдения (супервизоров).

Рис.11. 1: Структура процесса

**

Наше приложение построено из пяти компонентов. Структура этих компонентов показана на Рис.11. 1. Рисунок показывает три клиентских узла (предполагается, что они на других машинах) и один серверный узел (тоже на другой машине). Эти компоненты выполняют следующие функции:

Интерфейс с пользователем — это графическое приложение, которое используется для отправки сообщений и отображения полученных сообщений . Сообщения отправляются чат-клиенту.

Чат-клиент («С» на рисунке) — разбирается с сообщениями от пользовательского приложения и отправляет их к контроллеру группы для текущей группы. Принимает сообщения от контроллера группы и отправляет их к пользовательскому приложению.

Контроллер группы («G» на рисунке) — управляет одной чат-группой. Если контроллеру посылается сообщение, то он рассылает это сообщение всем участникам в данной группе. Он отслеживает новых участников, которые присоединились к группе и участников, которые покинули группу. Контроллер завершается, если в группе не осталось участников.

Чат-сервер («S» на рисунке) — отслеживает контроллеров группы. Чат-сервер нужен только когда новый участник пытается присоединиться к группе. Чат-сервер существует в единственном экземпляре, в то время как контроллеры групп создаются для каждой активной группы.

Посредник («M» на рисунке) — обеспечивает транспортировку данных в системе. Если процесс C посылает сообщение к M, оно попадёт к G (см. Рис.11. 1). Процесс M скрывает низкоуровневый интерфейс сокетов между двумя машинами. Главным образом процесс M прячет физическую границу между машинами за какой-то абстракцией. Это значит, что на основе передачи сообщений Эрланга можно построить целое приложение и не заботиться о подробностях нижележащей инфраструктуры связи.

11.1 Диаграммы последовательности сообщений

Если у нас много параллельных процессов, то очень легко потерять нить происходящего. Чтобы помочь нам понять, что происходит, мы можем нарисовать диаграмму последовательности сообщений (MSD), которая показывает взаимодействие между различными процессами.

Рис.11. 2: Прохождение сообщений, участвующих в передаче текстового сообщения

**

Диаграмма последовательности сообщений на Рис.11. 2 показывает последовательность сообщений, которые пересылаются, когда пользователь напечатает строку в поле ввода. Это приводит к отправке сообщения к чат-контроллеру (C), за которым следует сообщение к одному из посредников (M1), затем через M2 к контроллеру группы (G). На этапе между посредниками происходит двоичное кодирование сообщений Эрланга.

MSD даёт хорошее представление того, что происходит. Если вы будете глазеть на MSD и на код программы достаточно долго, то вы сможете убедить себя в том, что этот код реализует именно ту последовательность передачи сообщений, которая изображена на диаграмме.

Когда я проектирую программу наподобие чата, я часто рисую множество MSD диаграмм — это помогает мне думать о том, что происходит. Я не большой любитель графических методов проектирования, но MDS диаграммы полезны для отображения того, что происходит в ряде параллельных процессов, которые обмениваются сообщениями для решения определённой проблемы.

А сейчас посмотрим на индивидуальные компоненты.

11.2 Пользовательский интерфейс

Рис.11. 3: Виджет ввода-вывода

**

Пользовательский интерфейс построен на базе простого виджета ввода-вывода. Этот виджет показан на Рис.11. 3. Код этого виджета достаточно длинный и в основном касается доступа к оконной системе посредством стандартной библиотеки gs. Т. к. мы пока не хотим прыгать в эту кроличью нору, то мы не покажем здесь соответствующий код (хотя вы найдёте этот код, начиная со страницы *17*). Интерфейс у виджета ввода-вывода следующий:

@spec io_widget:start(Pid) -> Widget

Создаёт новый виджет ввода-вывода. Возвращает Widget, который является PID, который может использоваться для общения с виджетом. Когда пользователь печатает что-либо в поле ввода виджета процессу, который вызвал эту функцию посылаются сообщения вида {Widget, State, Parse}. State — это переменная состояния, сохранённая в виджете, которая может устанавливаться пользователем. Parse — это результат разбора строки ввода пользовательским парсером.

@spec io_widget:set_title(Widget, Str)

Устанавливает заголовок в виджете.

@spec io_widget:set_state(Widget, State)

Устанавливает состояние виджета.

@spec io_widget:insert_str(Widget, Str)

Вставляет строку в основную область виджета.

@spec io_widget:set_handler(Widget, Fun)

Устанавливает парсер виджета в Fun (см. далее).

Виджет ввода-вывода может генерировать следующие сообщения:

{Widget, State, Parse}

Это сообщение отправляется, когда пользователь вводит строку в нижней области команд виджета. Parse — это результат разбора этой строки парсером, связанным с данным виджетом.

{Widget, destroyed}

Это сообщение отправляется, когда пользователь разрушает виджет посредством закрытия окна.

В общем, виджет ввода-вывода — это программируемая штучка. С ним можно связать парсер, который будет использоваться для разбора всех сообщений, которые вводятся в поле ввода виджета. Разбор делается вызовом функции Parse(Str). Эта функция может быть установлена вызовом set_handler(Widget, Parse).

Парсер по-умолчанию — это такая функция:

Parse(Str) -> Str end.

11.3 Клиентская часть

Клиентская часть программы чата состоит из трёх процессов: виджет ввода-вывода (о котором мы уже говорили), клиент чата (который организует взаимодействие между виджетом и посредником) и процесс посредника. В этой части мы сосредоточимся на клиенте чата.

Клиент чата

Мы запускаем чат-клиент вызовом start/0:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

```
start() ->

connect("localhost" , 2223, "AsDT67aQ" , "general" , "joe" ).
```

Он пытается подсоединиться к localhost на порт 2223 (это жестко зашито в код для тестовых целей). Функция connect/5 просто создаёт параллельный процесс, вызывая hander/5. А вот обработчику приходится выполнять несколько задач:

- он делает себя системным процессом, так что теперь он может перехватывать сигналы выхода
- он создаёт виджет ввода-вывода и устанавливает подсказку и заголовок этого виджета
- он порождает процесс соединения (который пытается соединиться с сервером)
- в конце он ждёт события соединения в disconnected/2 (прим. перев.: «Синее, а не бурое! А по описанию -- бурое, а не синее!..» (С) АБС)

Код для него:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK

"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

```
connect(Host, Port, HostPsw, Group, Nick) ->

spawn(fun() -> handler(Host, Port, HostPsw, Group, Nick) end).

handler(Host, Port, HostPsw, Group, Nick) ->

process_flag(trap_exit, true),

Widget = io_widget:start(self()),

set_title(Widget, Nick),

set_state(Widget, Nick),

set_prompt(Widget, [Nick, " > " ]),

set_handler(Widget, fun parse_command/1),

start_connector(Host, Port, HostPsw),

disconnected(Widget, Group, Nick).
```

В отключенном состоянии процесс либо получит сообщение {connected, MM}HYPERLINK \l "id.826360296f2c"[2]HYPERLINK \l "id.826360296f2c", после чего он посылает сообщение login к серверу и ждёт ответа на логин, либо виджет может быть разрушен, что приводит к всеобщему завершению. Соединяющийся процесс периодически шлёт сообщения о состоянии к чат-клиенту. Эти сообщения сразу же пересылаются к виджету ввода-вывода для показа.

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

```
disconnected(Widget, Group, Nick) ->

receive
```

{connected, MM} ->

insert_str(Widget, "connected to server\nsending data\n"),

MM ! {login, Group, Nick},

wait_login_response(Widget, MM);

{Widget, destroyed} ->

exit(died);

{status, S} ->

insert_str(Widget, to_str(S)),

disconnected(Widget, Group, Nick);

Other ->

io:format("chat_client disconnected unexpected:\~p\~n" ,[Other]),

disconnected(Widget, Group, Nick)

end.

Сообщение {connected, MM} очевидно должно придти от соединяющегося процесса, который был создан вызовом start_connection(Host, Port, HostPsw). Этот вызов создаёт параллельный процесс, который в свою очередь периодически пытается соединиться с IRC сервером.

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

start_connector(Host, Port, Pwd) ->

S = self(),

spawn_link(fun() -> try_to_connect(S, Host, Port, Pwd) end).

try_to_connect(Parent, Host, Port, Pwd) ->

%% Parent is the Pid of the process that spawned this process

```
case lib_chan:connect(Host, Port, chat, Pwd, []) of

{error, _Why} ->

Parent ! {status, {cannot, connect, Host, Port}},

sleep(2000),

try_to_connect(Parent, Host, Port, Pwd);

{ok, MM} ->

lib_chan_mm:controller(MM, Parent),

Parent ! {connected, MM},

exit(connectorFinished)

end.
```

try_to_connect зацикливается навечно, пытаясь каждые две секунды подключиться к серверу. Если подключиться не удаётся, то он посылает сообщение о состоянии к чат-клиенту.

Замечание: в start_connection мы написали следующее:

```
S = self(),

spawn_link(fun() -> try_to_connect(S, ...) end)
```

Это не то же самое, что и здесь:

```
**

spawn_link(fun() -> try_to_connect(self(), ...) end)
```

Причина в том, что в первом фрагменте кода self() выполняется внутри родительского процесса. Во втором куске кода self() выполняется внутри порождённой функции, так что он возвращает идентификатор порождённого процесса, а не PID текущего процесса, как вы могли бы подумать. Это довольно распространённая причина для ошибок (и непонимания).

Если соединение установлено, то он посылает сообщение {connected, MM} к чат-клиенту. По прибытии этого сообщения чат-клиент посылает сообщение для логина к серверу (оба этих события происходят в disconnected/2) и ждёт ответа в wait_login_response/2:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK

*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

```
wait_login_response(Widget, MM) ->

receive

{MM, ack} ->

active(Widget, MM);

Other ->

io:format("chat_client login unexpected:\~p\~n" ,[Other]),

wait_login_response(Widget, MM)

end.
```

Если всё идёт по плану, то процесс должен получить подтверждающее сообщение (ack). (В нашем случае это единственно возможный ответ, т. к. пароль точно был правильным). После получения подтверждения эта функция вызывает active/2:

HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

```
active(Widget, MM) ->

receive

{Widget, Nick, Str} ->

MM ! {relay, Nick, Str},

active(Widget, MM);

{MM,{msg,From,Pid,Str}} ->

insert_str(Widget, [From,"@" ,pid_to_list(Pid)," " , Str, "\n" ]),
```

```
active(Widget, MM);

{'EXIT',Widget,windowDestroyed} ->

MM ! close;

{close, MM} ->

exit(serverDied);

Other ->

io:format("chat_client active unexpected:\~p\~n" ,[Other]),

active(Widget, MM)

end.
```

active/2 просто шлёт сообщения от виджета к группе (и наоборот) и отслеживает соединение с группой.

За исключением некоторых объявлений модулей и простейших процедур форматирования и разбора это завершает чат-клиент.

Полный код чат-клиента приведён на стр. ____

11.4 Серверная часть

Серверная часть программы сложнее, чем клиентская. Для каждого клиента чата есть соответствующий чат-контроллер, который организует взаимодействие чат-клиента с чат-сервером. Есть единственный чат-сервер, который знает обо всех сеансах чата в данный момент и ещё есть некоторое количество менеджеров групп (по одному на чат-группу), которые управляют отдельными чат-группами.

Чат-контроллер

Чат-контроллер — это дополнение (plug-in) для lib_chan, дистрибутивному набору, основанному на сокетах. Мы встречали его в главе 10.5, lib_chan, на стр. ____. lib_chan нуждается в конфигурационном файле и модуле дополнении.

Конфигурационный файл для системы чата следующий:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"_HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"chatHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf".HYPERLINK

"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"conf

{port, 2223}.

{service, chat, password,"AsDT67aQ" ,mfa,mod_chat_controller,start,[]}.

Если вы посмотрите назад на код chat_client.erl, вы увидите, что номер порта, имя сервиса и пароль согласуются с информацией из конфигурационного файла.

Модуль чат-контроллера очень прост:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"modHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"chatHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"_HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"controllerHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"erl

```
-module(mod_chat_controller).

-export([start/3]).

-import(lib_chan_mm, [send/2]).

start(MM, , ) ->

process_flag(trap_exit, true),

io:format("mod_chat_controller off we go ...\~p\~n" ,[MM]),

loop(MM).

loop(MM) ->

receive

{chan, MM, Msg} ->

chat_server ! {mm, MM, Msg},

loop(MM);

{'EXIT', MM, _Why} ->
```

chat_server ! {mm_closed, MM};

Other ->

io:format("mod_chat_controller unexpected message =\~p (MM=\~p)\~n" ,

[Other, MM]),

loop(MM)

end.

Этот код будет принимать только два сообщения. Когда клиент соединяется он получит произвольное сообщение и просто отправит его к чат-серверу. С другой стороны, если сеанс завершается по какой-либо причине, он получит сообщение о выходе и затем скажет чат-серверу, что клиент умер.

Чат-сервер

Чат-сервер — это зарегистрированный процесс, называемый (что неудивительно) chat_server. Вызов chat_server:start/0 запускает и регистрирует сервер, а он запускает lib_chan.

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"serverHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"erl

start() ->

start_server(),

lib_chan:start_server("chat.conf").

start_server() ->

register(chat_server,

spawn(fun() ->

process_flag(trap_exit, true),

Val= (catch server_loop([])),

io:format("Server terminated with:\~p\~n" ,[Val])

end)).

Серверный цикл прост. Он ждёт сообщения {login, Group, Nick}HYPERLINK \l "id.6415a23dbd50" [3]HYPERLINK \l "id.6415a23dbd50" от посредника с PID, равным Channel. Если есть контроллер чат-группы для этой группы, то он просто посылает сообщение о логине к контроллеру группы, а иначе он запускает нового контроллера группы.

Чат-сервер — это единственный процесс, который знает PID-ы всех контроллеров групп, так что, когда делается новое соединение к системе, к чат-серверу приходит запрос на поиск идентификатора процесса контроллера группы.

Сам по себе сервер прост:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"serverHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"erl

server_loop(L) ->

receive

{mm, Channel, {login, Group, Nick}} ->

case lookup(Group, L) of

{ok, Pid} ->

Pid ! {login, Channel, Nick},

server_loop(L);

error ->

Pid = spawn_link(fun() ->

chat_group:start(Channel, Nick)

end),

server_loop([{Group,Pid}|L])

```
end;

{mm_closed, _} ->

server_loop(L);

{'EXIT', Pid, allGone} ->

L1 = remove_group(Pid, L),

server_loop(L1);

Msg ->

io:format("Server received Msg=\~p\~n" ,

[Msg]),

server_loop(L)

end.
```

Код для манипуляций списком групп включает в себя несколько простых подпрограмм для обработки списков:

```
HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"chatHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"serverHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"erl

lookup(G, [{G,Pid}|_]) -> {ok, Pid};

lookup(G, [_|T])

-> lookup(G, T);

lookup(_,[])

-> error.

remove_group(Pid, [{G,Pid}|T]) -> io:format("\~p removed\~n" ,[G]), T;

remove_group(Pid, [H|T])
```

-> [Hlremove_group(Pid, T)];

remove_group(_, [])

-> [].

Менеджер группы

К текущему моменту всё, что осталось — это менеджер группы. Важнейшая часть этого — диспетчер.

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"groupHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"erl

group_controller([]) ->

exit(allGone);

group_controller(L) ->

receive

{C, {relay, Nick, Str}} ->

foreach(fun({Pid,_}) -> Pid ! {msg, Nick, C, Str} end, L),

group_controller(L);

{login, C, Nick} ->

controller(C, self()),

C ! ack,

self() ! {C, {relay, Nick, "I'm joining the group" }},

group_controller([{C,Nick}lL]);

{close,C} ->

{Nick, L1} = delete(C, L, []),

self() ! {C, {relay, Nick, "I'm leaving the group" }},

group_controller(L1);

Any ->

io:format("group controller received Msg=\~p\~n" , [Any]),

group_controller(L)

end.

Аргумент L в group_controller(L) — это список имён и идентификаторов процессов посредников {Pid, Nick}.

Когда менеджер группы получает сообщение {relay, Nick, Str}, он просто рассылает его всем процессам в группе. Если приходит сообщение {login, C, Nick}, он добавляет кортеж {C, Nick} в список рассылки. *Важно* упомянуть вызов lib_chan_mm:controller/2. Этот вызов устанавливает управляющий процесс посредника в контроллер группы, что означает, что *все сообщения к сокету, управляемому посредником, будут посланы к контроллеру группы* — это, вероятно, главная часть для понимания — как работает весь этот код.

Всё, что остаётся — это код, который запускает сервер группы:

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"groupHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"erl

-module(chat_group).

-import(lib_chan_mm, [send/2, controller/2]).

-import(lists, [foreach/2, reverse/2]).

-export([start/2]).

start(C, Nick) ->

process_flag(trap_exit, true),

controller(C, self()),

C ! ack,

self() ! {C, {relay, Nick, "I'm starting the group" }},

group_controller([{C,Nick}]).

и функция delete/3, вызываемая из цикла диспетчера процесса

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"groupHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"erl

delete(Pid, [{Pid,Nick}|T], L) -> {Nick, reverse(T, L)};

delete(Pid, [H|T], L)

-> delete(Pid, T, [H|L]);

delete(_, [], L)

-> {"????" , L}.

11.5 Запуск приложения

Приложение целиком располагается в каталоге pathto/code/socket_dist. Оно также использует некоторые библиотечные модули из каталога pathto/code.

Для запуска приложения получите исходные коды веб-сайта этой книги и распакуйте их в какой-нибудь каталог. (Здесь мы предполагаем, что это каталог /home/joe/erlbook). Откройте окно терминала и выполните следующие команды:

$ cd /home/joe/erlbook/code

/home/joe/erlbook/code $ make

...

/home/joe/erlbook/code $ cd socket_dist

/home/joe/erlbook/code/socket_dist $ make chat_server

...

Это запустит чат-сервер. А теперь нам надо открыть другое терминальное окно и запустить тест клиента:

$ cd /home/joe/erlbook/code/socket_dist

/home/joe/erlbook/code/socket_dist $ make chat_client

...

Запуск make chat_client выполняет функцию chat_client:test(). Это на самом деле создаёт четыре окна, которые подключаются к тестовой группе, названной «general». На Рис.11. 4 мы можем увидеть снимок экрана, показывающий как выглядит система после выдачи этих команд.

Рис.11. 4: Снимок экрана, показывающий четыре окна, подключенные к одной группе

**

Для развёртывания системы в Интернете всё, что нам надо сделать — это поменять пароль и порт на что-нибудь подходящее и разрешить входящие соединения на порт, который мы выбрали.

11.6 Исходные коды программы чата

Итак, мы завершили описание программы чата. При описании программы мы разбили её на несколько маленьких фрагментов и опустили некоторую часть кода. Этот раздел содержит весь код в одном месте, что облегчает его чтение. Если у вас есть трудности с пониманием, что делает та или иная часть кода, обратитесь к описанию, изложенному ранее в этой главе.

Чат-клиент

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"chatHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"clientHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_client.erl"erl

```
-module(chat_client).

-import(io_widget,

[get_state/1, insert_str/2, set_prompt/2, set_state/2,

set_title/2, set_handler/2, update_state/3]).
```

```
-export([start/0, test/0, connect/5]).

start() ->

connect("localhost" , 2223, "AsDT67aQ" , "general" , "joe" ).

test() ->

connect("localhost" , 2223, "AsDT67aQ" , "general" , "joe" ),

connect("localhost" , 2223, "AsDT67aQ" , "general" , "jane" ),

connect("localhost" , 2223, "AsDT67aQ" , "general" , "jim" ),

connect("localhost" , 2223, "AsDT67aQ" , "general" , "sue" ).

connect(Host, Port, HostPsw, Group, Nick) ->

spawn(fun() -> handler(Host, Port, HostPsw, Group, Nick) end).

handler(Host, Port, HostPsw, Group, Nick) ->

process_flag(trap_exit, true),

Widget = io_widget:start(self()),

set_title(Widget, Nick),

set_state(Widget, Nick),

set_prompt(Widget, [Nick, " > " ]),

set_handler(Widget, fun parse_command/1),

start_connector(Host, Port, HostPsw),

disconnected(Widget, Group, Nick).

disconnected(Widget, Group, Nick) ->

receive

{connected, MM} ->

insert_str(Widget, "connected to server\nsending data\n" ),

MM ! {login, Group, Nick},

wait_login_response(Widget, MM);

{Widget, destroyed} ->

exit(died);

{status, S} ->
```

```
insert_str(Widget, to_str(S)),
disconnected(Widget, Group, Nick);
Other ->
io:format("chat_client disconnected unexpected:\~p\~n" ,[Other]),
disconnected(Widget, Group, Nick)
end.
wait_login_response(Widget, MM) ->
receive
{MM, ack} ->
active(Widget, MM);
Other ->
io:format("chat_client login unexpected:\~p\~n" ,[Other]),
wait_login_response(Widget, MM)
end.
active(Widget, MM) ->
receive
{Widget, Nick, Str} ->
MM ! {relay, Nick, Str},
active(Widget, MM);
{MM,{msg,From,Pid,Str}} ->
insert_str(Widget, [From,"@" ,pid_to_list(Pid)," " , Str, "\n" ]),
active(Widget, MM);
{'EXIT',Widget,windowDestroyed} ->
MM ! close;
{close, MM} ->
exit(serverDied);
Other ->
io:format("chat_client active unexpected:\~p\~n" ,[Other]),
```

```
active(Widget, MM)
end.
start_connector(Host, Port, Pwd) ->
S = self(),
spawn_link(fun() -> try_to_connect(S, Host, Port, Pwd) end).
try_to_connect(Parent, Host, Port, Pwd) ->
%% Parent is the Pid of the process that spawned this process
case lib_chan:connect(Host, Port, chat, Pwd, []) of
{error, _Why} ->
Parent ! {status, {cannot, connect, Host, Port}},
sleep(2000),
try_to_connect(Parent, Host, Port, Pwd);
{ok, MM} ->
lib_chan_mm:controller(MM, Parent),
Parent ! {connected, MM},
exit(connectorFinished)
end.
sleep(T) ->
receive
after T -> true
end.
to_str(Term) ->
io_lib:format("\~p\~n" ,[Term]).
parse_command(Str) -> skip_to_gt(Str).
skip_to_gt(">" ++ T) -> T;
skip_to_gt([_|T])
-> skip_to_gt(T);
skip_to_gt([])
```

-> exit("no >").

Конфигурация lib_chan

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"_HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"chatHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat.conf"conf

{port, 2223}.

{service, chat, password,"AsDT67aQ" ,mfa,mod_chat_controller,start,[]}.

Чат-контроллер

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"modHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"chatHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"_HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"controllerHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/mod_chat_controller.erl"erl

-module(mod_chat_controller).

-export([start/3]).

-import(lib_chan_mm, [send/2]).

start(MM, ,) ->

process_flag(trap_exit, true),

io:format("mod_chat_controller off we go ...\~p\~n" ,[MM]),

loop(MM).

```
loop(MM) ->

receive

{chan, MM, Msg} ->

chat_server ! {mm, MM, Msg},

loop(MM);

{'EXIT', MM, _Why} ->

chat_server ! {mm_closed, MM};

Other ->

io:format("mod_chat_controller unexpected message =\~p (MM=\~p)\~n" ,

[Other, MM]),

loop(MM)

end.
```

Чат-сервер

HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"DownloadHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl" HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"socketHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"distHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"/HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"chatHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"serverHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl".HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_server.erl"erl

```
-module(chat_server).

-import(lib_chan_mm, [send/2, controller/2]).

-import(lists, [delete/2, foreach/2, map/2, member/2,reverse/2]).

-compile(export_all).

start() ->

start_server(),

lib_chan:start_server("chat.conf" ).
```

```
start_server() ->

register(chat_server,

spawn(fun() ->

process_flag(trap_exit, true),

Val= (catch server_loop([])),

io:format("Server terminated with:\~p\~n" ,[Val])

end)).

server_loop(L) ->

receive

{mm, Channel, {login, Group, Nick}} ->

case lookup(Group, L) of

{ok, Pid} ->

Pid ! {login, Channel, Nick},

server_loop(L);

error ->

Pid = spawn_link(fun() ->

chat_group:start(Channel, Nick)

end),

server_loop([{Group,Pid}|L])

end;

{mm_closed, _} ->

server_loop(L);

{'EXIT', Pid, allGone} ->

L1 = remove_group(Pid, L),

server_loop(L1);

Msg ->

io:format("Server received Msg=\~p\~n" ,

[Msg]),
```

```
server_loop(L)

end.

lookup(G, [{G,Pid}|_]) -> {ok, Pid};

lookup(G, [_|T])

-> lookup(G, T);

lookup(_,[])

-> error.

remove_group(Pid, [{G,Pid}|T]) -> io:format("\~p removed\~n" ,[G]), T;

remove_group(Pid, [H|T])

-> [H|remove_group(Pid, T)];

remove_group(_, [])

-> [].
```

Чат-группы

HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"DownloadHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl" HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"socketHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"distHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"/HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"chatHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"*HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"groupHYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl".HYPERLINK "http://media.pragprog.com/titles/jaerlang/code/socket_dist/chat_group.erl"erl

```
-module(chat_group).

-import(lib_chan_mm, [send/2, controller/2]).

-import(lists, [foreach/2, reverse/2]).

-export([start/2]).

start(C, Nick) ->

process_flag(trap_exit, true),

controller(C, self()),
```

```
C ! ack,
self() ! {C, {relay, Nick, "I'm starting the group" }},
group_controller([{C,Nick}]).
delete(Pid, [{Pid,Nick}|T], L) -> {Nick, reverse(T, L)};
delete(Pid, [H|T], L)
-> delete(Pid, T, [H|L]);
delete(_, [], L)
-> {"????" , L}.
group_controller([]) ->
exit(allGone);
group_controller(L) ->
receive
{C, {relay, Nick, Str}} ->
foreach(fun({Pid,_}) -> Pid ! {msg, Nick, C, Str} end, L),
group_controller(L);
{login, C, Nick} ->
controller(C, self()),
C ! ack,
self() ! {C, {relay, Nick, "I'm joining the group" }},
group_controller([{C,Nick}|L]);
{close,C} ->
{Nick, L1} = delete(C, L, []),
self() ! {C, {relay, Nick, "I'm leaving the group" }},
group_controller(L1);
Any ->
io:format("group controller received Msg=\~p\~n" , [Any]),
group_controller(L)
end.
```

Виджет ввода-вывода

HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"DownloadHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl" HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"socketHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"*HYPERLINK*
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"distHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"/HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"ioHYPERLINK
*"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"*HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"widgetHYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl".HYPERLINK
"http://media.pragprog.com/titles/jaerlang/code/socket_dist/io_widget.erl"erl

```
-module(io_widget).

-export([get_state/1,

start/1, test/0,

set_handler/2,

set_prompt/2,

set_state/2,

set_title/2, insert_str/2, update_state/3]).

start(Pid) ->

gs:start(),

spawn_link(fun() -> widget(Pid) end).

get_state(Pid)

-> rpc(Pid, get_state).

set_title(Pid, Str)

-> Pid ! {title, Str}.

set_handler(Pid, Fun)

-> Pid ! {handler, Fun}.

set_prompt(Pid, Str)

-> Pid ! {prompt, Str}.

set_state(Pid, State)
```

```
-> Pid ! {state, State}.

insert_str(Pid, Str)

-> Pid ! {insert, Str}.

update_state(Pid, N, X) -> Pid ! {updateState, N, X}.

rpc(Pid, Q) ->

Pid ! {self(), Q},

receive

{Pid, R} ->

R

end.

widget(Pid) ->

Size = [{width,500},{height,200}],

Win = gs:window(gs:start(),

[{map,true},{configure,true},{title,"window" }|Size]),

gs:frame(packer, Win,[{packer_x, [{stretch,1,500}]},

{packer_y, [{stretch,10,120,100},

{stretch,1,15,15}]}]),

gs:create(editor,editor,packer, [{pack_x,1},{pack_y,1},{vscroll,right}]),

gs:create(entry, entry, packer, [{pack_x,1},{pack_y,2},{keypress,true}]),

gs:config(packer, Size),

Prompt = " > " ,

State = nil,

gs:config(entry, {insert,{0,Prompt}}),

loop(Win, Pid, Prompt, State, fun parse/1).

loop(Win, Pid, Prompt, State, Parse) ->

receive

{From, get_state} ->

From ! {self(), State},
```

```
loop(Win, Pid, Prompt, State, Parse);

{handler, Fun} ->

loop(Win, Pid, Prompt, State, Fun);

{prompt, Str} ->

%% this clobbers the line being input ...

%% this could be fixed - hint

gs:config(entry, {delete,{0,last}}),

gs:config(entry, {insert,{0,Str}}),

loop(Win, Pid, Str, State, Parse);

{state, S} ->

loop(Win, Pid, Prompt, S, Parse);

{title, Str} ->

gs:config(Win, [{title, Str}]),

loop(Win, Pid, Prompt, State, Parse);

{insert, Str} ->

gs:config(editor, {insert,{'end',Str}}),

scroll_to_show_last_line(),

loop(Win, Pid, Prompt, State, Parse);

{updateState, N, X} ->

io:format("setelemtn N=\~p X=\~p Satte=\~p\~n" ,[N,X,State]),

State1 = setelement(N, State, X),

loop(Win, Pid, Prompt, State1, Parse);

{gs,,destroy,,_} ->

io:format("Destroyed\~n" ,[]),

exit(windowDestroyed);

{gs, entry,keypress,,['Return'|]} ->

Text = gs:read(entry, text),

%% io:format("Read:\~p\~n",[Text]),
```

```
gs:config(entry, {delete,{0,last}}),
gs:config(entry, {insert,{0,Prompt}}),
try Parse(Text) of
Term ->
Pid ! {self(), State, Term}
catch
_: ->
self() ! {insert, " bad input\n** /h for help\n" }
end,
loop(Win, Pid, Prompt, State, Parse);
{gs,_,configure,[],[W,H,_,_]} ->
gs:config(packer, [{width,W},{height,H}]),
loop(Win, Pid, Prompt, State, Parse);
{gs, entry,keypress,_,_} ->
loop(Win, Pid, Prompt, State, Parse);
Any ->
io:format("Discarded:\~p\~n" ,[Any]),
loop(Win, Pid, Prompt, State, Parse)
end.
scroll_to_show_last_line() ->
Size
= gs:read(editor, size),
Height
= gs:read(editor, height),
CharHeight = gs:read(editor, char_height),
TopRow
= Size - Height/CharHeight,
if
```

```
TopRow > 0 -> gs:config(editor, {vscrollpos, TopRow});

true

-> gs:config(editor, {vscrollpos, 0})

end.

test() ->

spawn(fun() -> test1() end).

test1() ->

W = io_widget:start(self()),

io_widget:set_title(W, "Test window" ),

loop(W).

loop(W) ->

receive

{W, {str, Str}} ->

Str1 = Str ++ "\n" ,

io_widget:insert_str(W, Str1),

loop(W)

end.

parse(Str) ->

{str, Str}.
```

11.7 Упражнения

- улучшите графический виджет, добавив боковую панель для перечисления имён участников группы
- добавьте код для показа имён всех участников группы
- добавьте код для перечисления всех групп
- добавьте личные сообщения
- добавьте такой код, чтобы контроллер группы работал не на серверной машине, а на машине первого пользователя, который подключился к данной группе
- Посмотрите внимательно на диаграмму последовательности сообщений (Рис.11. 2), чтобы убедиться, что вы понимаете её и проверьте, что вы можете указать все сообщения из

диаграммы в программном коде

- нарисуйте свою собственную диаграмму последовательности сообщений, чтобы показать, как решается проблема в фазе логина (в оригинале «фаза логина проблемы»)

HYPERLINK \l "id.0b9a32f8a799"[1]HYPERLINK \l "id.0b9a32f8a799" Это облегчает нам жизнь и позволяет сосредоточиться на приложении вместо низкоуровневых деталей протокола.

HYPERLINK \l "id.64ae7d66b7a1"[2]HYPERLINK \l "id.64ae7d66b7a1" MM означает middle man — посредник. Это процесс, который используется для связи с сервером.

HYPERLINK \l "id.3da754ac9620"[3]HYPERLINK \l "id.3da754ac9620" Nick — это прозвище пользователя