

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

По дисциплине «Системное программное обеспечение»

Вариант №2

Доска обсуждений

Студент:

Михайлова Алена Андреевна

Группа:

P34112

Преподаватель:

Кореньков Юрий Дмитриевич

Санкт-Петербург

2021 г.

1. Текст задания

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. В режиме сервера линейно отображается журнал всех входящих и исходящих сообщений, завершение программы-сервера выполняется по нажатию ключевой клавиши (например, Q).

При запуске в режиме клиента программе в качестве аргументов командной строки также передается имя пользователя и адрес сервера. Дискуссионная доска представляет собой дерево сообщений, узлы которого можно разворачивать или сворачивать для просмотра ответов на интересующее сообщение. Для отправки своего сообщения в дереве выбирается существующее, под которым появится отправляемый ответ. Предусмотреть прокрутку дерева, если оно не помещается на экране, а также индикацию появления новых ответов, в том числе в неразвёрнутых ветках. Собственное сообщение вводится в отдельной строке в нижней части окна, не блокируя работу дерева.

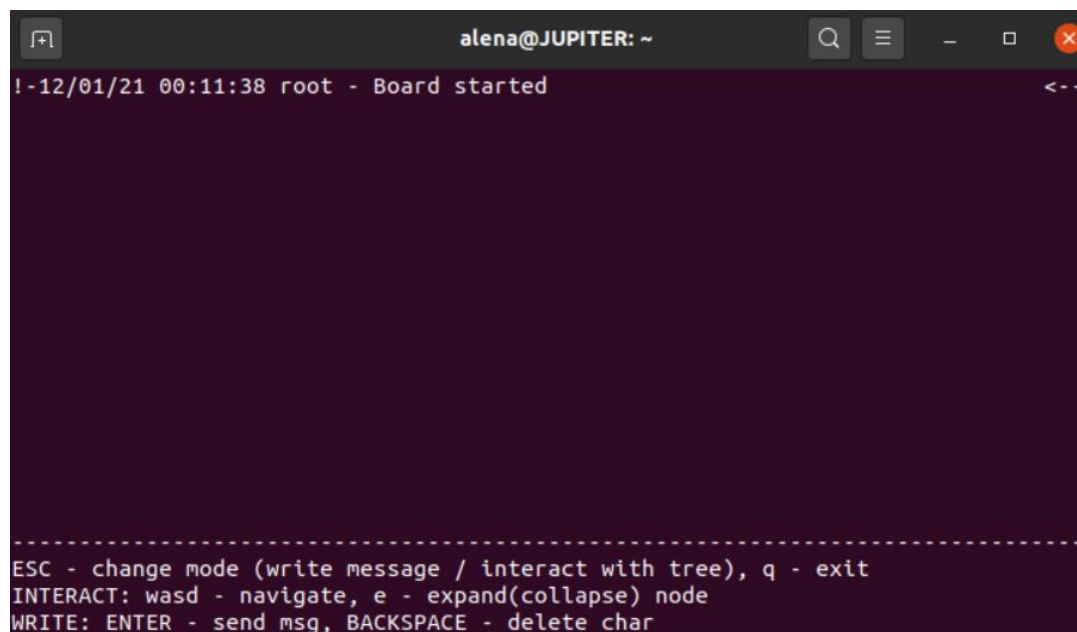
2. Описание работы

Запуск программы:

- Для запуска серверной части исполняемый файл нужно запустить с одним аргументом: `server`
- Для запуска клиентской части исполняемый файл нужно запустить с тремя аргументами: `client <port number> <username>`

Примеры работы:

1) Внешний вид доски



```
alena@JUPITER: ~  
!-12/01/21 00:11:38 root - Board started  
  
-----  
ESC - change mode (write message / interact with tree), q - exit  
INTERACT: wasd - navigate, e - expand(collapse) node  
WRITE: ENTER - send msg, BACKSPACE - delete char
```

2) Отправка сообщения

```
alena@JUPITER: ~  
!-12/01/21 00:11:38 root - Board started  
!-12/01/21 00:12:40 alena - Hello everyone! This is the board!! WOW!!!  
  
-----  
ESC - change mode (write message / interact with tree), q - exit  
INTERACT: wasd - navigate, e - expand(collapse) node  
WRITE: ENTER - send msg, BACKSPACE - delete char
```

3) После снятия индикации нового сообщения

```
alena@JUPITER: ~  
+-12/01/21 00:11:38 root - Board started  
+-12/01/21 00:12:40 alena - Hello everyone! This is the board!! WOW!!!  
  
-----  
ESC - change mode (write message / interact with tree), q - exit  
INTERACT: wasd - navigate, e - expand(collapse) node  
WRITE: ENTER - send msg, BACKSPACE - delete char
```

4) Ответ на сообщение другим пользователем

```
alena@JUPITER: ~  
!-12/01/21 00:11:38 root - Board started  
!-12/01/21 00:12:40 alena - Hello everyone! This is the board!! WOW!!! <--  
!-12/01/21 00:13:56 not_alena - YEAH! CAN YOU IMAGINE THAT WE CAN CHA  
  
-----  
ESC - change mode (write message / interact with tree), q - exit  
INTERACT: wasd - navigate, e - expand(collapse) node  
WRITE: ENTER - send msg, BACKSPACE - delete char
```

5) Демонстрация скролла доски вправо

```
alena@JUPITER: ~  
21 00:11:38 root - Board started  
01/21 00:12:40 alena - Hello everyone! This is the board!! WOW!!! <--  
12/01/21 00:13:56 not_alena - YEAH! CAN YOU IMAGINE THAT WE CAN CHAT HERE??  
  
-----  
ESC - change mode (write message / interact with tree), q - exit  
INTERACT: wasd - navigate, e - expand(collapse) node  
WRITE: ENTER - send msg, BACKSPACE - delete char
```

6) Демонстрация скролла доски вниз

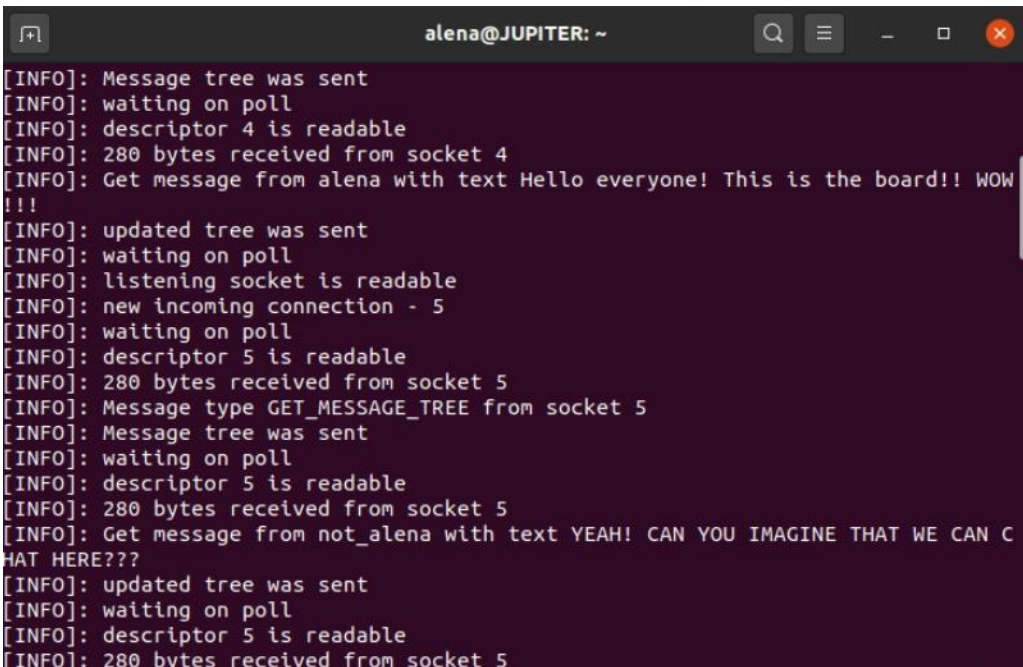
```
alena@JUPITER: ~  
+-12/01/21 00:14:24 not_alena - s  
+-12/01/21 00:14:25 not_alena - d  
+-12/01/21 00:14:26 not_alena - f  
+-12/01/21 00:14:27 not_alena - gh  
+-12/01/21 00:14:28 not_alena - j  
+-12/01/21 00:14:28 not_alena - k  
+-12/01/21 00:14:29 not_alena - l  
+-12/01/21 00:14:30 not_alena - z  
+-12/01/21 00:14:30 not_alena - x  
+-12/01/21 00:14:31 not_alena - c  
+-12/01/21 00:14:31 not_alena - v  
+-12/01/21 00:14:32 not_alena - v  
+-12/01/21 00:14:32 not_alena - b  
+-12/01/21 00:14:33 not_alena - n  
+-12/01/21 00:14:34 not_alena - n  
  
-----  
ESC - change mode (write message / interact with tree), q - exit  
INTERACT: wasd - navigate, e - expand(collapse) node  
WRITE: ENTER - send msg, BACKSPACE - delete char
```

7) Завершение клиентской программы по нажатию на q



```
alena@JUPITER: ~  
Hope not to see you again :)
```

8) Логи на стороне сервера



```
alena@JUPITER: ~  
[INFO]: Message tree was sent  
[INFO]: waiting on poll  
[INFO]: descriptor 4 is readable  
[INFO]: 280 bytes received from socket 4  
[INFO]: Get message from alena with text Hello everyone! This is the board!! WOW  
!!!  
[INFO]: updated tree was sent  
[INFO]: waiting on poll  
[INFO]: listening socket is readable  
[INFO]: new incoming connection - 5  
[INFO]: waiting on poll  
[INFO]: descriptor 5 is readable  
[INFO]: 280 bytes received from socket 5  
[INFO]: Message type GET_MESSAGE_TREE from socket 5  
[INFO]: Message tree was sent  
[INFO]: waiting on poll  
[INFO]: descriptor 5 is readable  
[INFO]: 280 bytes received from socket 5  
[INFO]: Get message from not_alena with text YEAH! CAN YOU IMAGINE THAT WE CAN C  
HAT HERE???  
[INFO]: updated tree was sent  
[INFO]: waiting on poll  
[INFO]: descriptor 5 is readable  
[INFO]: 280 bytes received from socket 5
```

9) Завершение работы серверной программы по нажатию на кнопку q



```
alena@JUPITER: ~  
[INFO]: waiting on poll  
[INFO]: some key was pressed  
Goodbye, my dear friend...  
[INFO]: connections were closed  
alena@JUPITER:~$
```

3. Аспекты реализации и результаты

Модель представления данных

Доска обсуждений представляет из себя дерево.

Структура дерева:

```
Struct  
tree {  
    struct tree_node *start; //ссылка на корень (первый узел)  
    size_t used; //сколько узлов использовано  
    size_t size; //размер дерева в узлах  
};
```

Структура узла дерева:

```
struct
tree_node
{
    int id;
    int sibling_id;
    int child_id;
    int parent_id;
    time_t creation_time;
    char username[20];
    char content[256];
};
```

При добавлении нового сообщения от клиента серверу посылается следующая небольшая структура:

```
struct
message
{
    char username[20];
    char content[256];
    int parent_id;
};
```

Сервер же отправляет клиенту целые узлы дерева.

Серверная часть

Сервер был реализован в один поток с использованием механизма «poll», который предоставляет механизм одновременного управления вводом и выводом для набора файловых дескрипторов.

Изначально в массив дескрипторов добавляется два:

```
//сокет для прослушивания новых подключений
fds[0].fd = listen_sd;
fds[0].events = POLLIN;
//дескриптор stdin, чтобы прослушивать нажатие на кнопки (для осуществления
выхода из программы)
fds[1].fd = 0;
fds[1].events = POLLIN;
```

Далее, в ходе работы poll, подключаются клиентские соединения, которые мы тоже кладем в массив структура, чтобы в дальнейшем отслеживать, когда в них появляется информация для считывания (приходит запрос от клиента).

Чтобы получить дерево целиком, клиенту необходимо отправить сообщение с parent_id = -1, так как таким id может обладать только корневое.

После добавления нового сообщения каким-то из клиентов сервер отправляет всем клиентам по два сообщения: сообщение, где обновился child/sibling id в ходе добавления нового сообщения и само новое сообщение.

При отключении клиента возникает event, poll «просыпается» и на выходе получает отрицательное значение считанных байтов, что сигнализирует об отключении, в ходе чего сервер переупаковывает массивы файловых дескрипторов сокетов.

Клиентская часть

Для реализации клиентской части также был использован poll. В массив структур добавляется два дескриптора: сокет для подключения к серверу (и получения сообщений от него) и stdin для реализации взаимодействия пользователя с псевдографикой.

Логика взаимодействия сервера реализована следующим образом: при инициализации клиента отправляется запрос на сервер с сообщением, где parent_id = -1, то есть клиент запрашивает полное дерево доски сообщений. После его получения клиент обновляет локальную копию доски обсуждений и занимается его отрисовкой.

После этого клиент может начинать отправлять свои сообщения, отвечая на сообщения из доски обсуждений. При нажатии на ENTER после ввода нового сообщения, оно отправляется на сервер, после чего мы получаем обновленные узлы, обновляем в соответствии с этим локальную версию дерева и перерисовываем его.

Реализация отрисовки была несколько сложнее. Первое, с чем пришлось столкнуться – это с необходимостью перевода терминала в неканонический режим, который позволяет драйверу терминала не ожидать окончания ввода строки в виде нажатия на ENTER, а сразу отправлять символ в stdin программы.

Для отрисовки элементов были использованы управляющие кодовые последовательности терминала ANSI, которые позволяют перемещать курсор, очищать терминал и делать всякие другие вещи.

Но сложнее всего было реализовать логику по сворачиванию/разворачиванию узлов и прокрутку доски при ее необходимости. Для этого был добавлен массив структур visible_status, число элементов в котором равно количеству элементов в дереве обсуждений, там для каждого узла хранится информация о том, является ли новым это сообщение (если да, то его нужно отмечать «!») и о том, нужно ли этот элемент отображать (свернута или развернута ветка). Также есть массив draw_order, который является порядком отрисовки элементов в данный момент. Массив draw_order строится рекурсивно (так как это дерево), учитывая массив visible. Для реализации прокрутки в приложении хранится состояние (struct coordinates *start_with), отражающее, с какого элемента необходимо отрисовывать доску.

4. Выводы

Открытие №1: существование механизма poll.

Открытие №2: что в poll можно в том числе захватить stdin (так как это файл) и с помощью этого реализовывать реакцию на пользовательский ввод.

Открытие №3: управляющие кодовые последовательности терминала ANSI, про которые я вообще до момента написания лабы даже не слышала.