МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Информационные технологий и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Курсовая работа
по курсу "Операционные системы"
3 семестр

Задание 8
Проектирование консольной клиент-серверной игры

Студент: Черкашин Андрей Викторович
Группа: М8О-206Б-20
Преподаватель: Соколов А. А.
Дата: 29.12.21
Оценка: 5
Подпись: _____

Москва, 2021

1. Постановка задачи

Задание:

На основе любой из выбранных технологий:

- Pipes
- Sockets
- Сервера очередей
- И другие

Создать собственную игру более, чем для одного пользователя. Игра может быть устроена по принципу: клиент-клиент, сервер-клиент.

Вариант 8: «Быки и коровы» (угадывать необходимо числа). Общение между сервером и клиентом необходимо организовать при помощи pipe'ов. При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Должна быть реализована функция поиска игры, то есть игрок пытается войти в игру не по имени, а просто просит сервер найти ему игру.

2. Основная часть

Общая информация о pipe-ах

Концептуально канал представляет собой соединение между двумя процессами, так что стандартный вывод одного процесса становится стандартным вводом другого процесса. В операционной системе UNIX каналы полезны для связи между связанными процессами (межпроцессное взаимодействие).

- Канал является односторонней связью, т. е. Мы можем использовать канал так, чтобы один процесс записывал в канал, а другой — из канала. Он открывает канал, который является областью основной памяти, которая рассматривается как «виртуальный файл» .
- Канал может использоваться процессом создания, а также всеми его дочерними процессами для чтения и записи. Один процесс может записывать в этот «виртуальный файл» или канал, а другой связанный процесс может читать из него.
- Если процесс пытается прочитать до того, как что-то записано в канал, процесс приостанавливается до тех пор, пока что-то не будет записано.
- Системный вызов pipe находит первые две доступные позиции в таблице открытых файлов процесса и распределяет их по концам канала для чтения и записи.

Именнованые каналы

Именованные каналы во многом работают так же, как и обычные каналы, но все же имеют несколько заметных отличий.

- Именованные каналы существуют в виде специального файла устройства в файловой системе.

- Процессы различного происхождения могут разделять данные через такой канал.
- Именованный канал остается в файловой системе для дальнейшего использования и после того, как весь ввод/вывод сделан.

Интересное свойство именнованных каналов (блокировка)

Если FIFO открыт для чтения, процесс его блокирует до тех пор, пока какой-нибудь другой процесс не откроет FIFO для записи. Аналогично для обратной ситуации. Если такое поведение нежелательно, то может быть использован флаг O_NONBLOCK в системном вызове open(), чтобы отменить действие блокирования.

Описание программы

В программе используются именно именнованые каналы, потому что это позволяет работать клиенту и серверу раздельно. Как подключается клиент? Все просто, при запуске сервера создается канал, к которому подключается клиент при запуске, когда он подключился, сервер по этому каналу высылает новые каналы по которым клиент подключется к своему потоку на сервере, а при этом канал, по которому он подключался остается активным, чтобы можно было подключиться другим клиентам. При подключении клиента у него появляется промптер, указывающий, что он может начать работу с сервером и отправлять ему команды. Когда промптер > клиент может запросить список текущих игр, создать игру по количеству игроков (если нужно, он может указать имя игры, если нет, то имя будет сгенерировано автоматически). Также клиент может присоединиться к игре по имени или без него, во втором случае сервер найдет свободную комнату и перекинет игрока в эту комнату. Когда игроков не хватает, высвечивается

надпись Waiting for players… Ожидание реализовано с помощью отправки серверу запроса и получения ответа, в котором указано состояние игры. Когда комната заполнена, то промптер меняется на ‘название игы’# у того, чей ход на данный момент. Все остальные в комнате получают надпись об ожидании хода. Если кто-то угадал загаданное число, выводится сообщение о выигрыше или проигрыше, после этого клиент может выйти из игры с помощью команды q. Далее для него доступны все основные действия, описанные выше.

Общение клиент-сервер построено на request-reply, т.е. клиент отправляет запрос, сервер ему отвечает.

Игры хранятся на сервере в виде массива, созданного глобально. Аналогично хранится массив игроков. Иногда бывает такое, что к одному участку общей памяти обращаются разные потоки. Во избежания ошибок нужно было блокировать участок памяти, когда один из потоков обращается к нему. Для этого было принято решение использовать семафор.

3. Тестирование программы

Server

```
slash@avm:~/Desktop/OS_KP$ ./server
/tmp/bulls_and_cows_sr1
nodes created
game /tmp/bulls_and_cows_sw1 /tmp/bulls_and_cows_sr1
IDX: 0
THREAD STARTED
/tmp/bulls_and_cows_sr2
nodes created
game /tmp/bulls_and_cows_sw2 /tmp/bulls_and_cows_sr2
IDX: 1
THREAD STARTED
/tmp/bulls_and_cows_sr3
nodes created
game /tmp/bulls_and_cows_sw3 /tmp/bulls_and_cows_sr3
IDX: 2
THREAD STARTED
/tmp/bulls_and_cows_sr4
nodes created
game /tmp/bulls_and_cows_sw4 /tmp/bulls_and_cows_sr4
IDX: 3
THREAD STARTED
game = *game0 [2\2] num: 1234
mystery: 5342 b: 0 c 3
game = a [2\2] num: 1234
mystery: 6937 b: 1 c 1
game = *game0 [2\2] num: 5342
mystery: 5342 b: 4 c 4
game = a [2\2] num: 6937
mystery: 6937 b: 4 c 4
1.Gamename: a(a) PC: 0
OK
Games_count 1
list reply: [*game0[2\2]
]
OK
Games_count 1
list reply: [*game0[2\2]
]
OK
Games_count 1
```

```
list reply: [*game0[2\1]
]
0.Gamename: *game0(*game0) PC: 0
OK
Games_count 0
list reply: [NO GAMES AVAILABLE
]
Exited
Exited
Exited
Exited
^C
```

Client1

```
slash@avm:~/Desktop/OS_KP$ ./client
> create 2
*game0# 1234
bulls: 0, cows: 3
You are looser! Hehe
*game0 loser# q
> l
*game0[2\1]
> exit
```

Client2

```
slash@avm:~/Desktop/OS_KP$ ./client
> create 2 a
a# 1234
bulls: 1, cows: 0
You are looser! Hehe
a loser# q
> q
> exit
```

Client3

```
> join
*game0 2 2
*game0# 5342
bulls: 4, cows: 0
Congratulations!

You are the WINNER!!!
*game0 winner# q
> l
```

```
NO GAMES AVAILABLE
> exit
```

Client4

```
> join a
a 2 2
a# 6937
bulls: 4, cows: 0
Congratulations!

You are the WINNER!!!
a winner# q
> l
*game0[2\2]
> q
> l
*game0[2\2]
> exit
```

4. Выводы

Использование именнованых каналов позволило существенно упростить подключение пользователей к серверу. Передача сообщений через pipe оказалась очень удобной и простой, однако выполнение манипуляций над общими участками памяти заставила меня прибегнуть к использованию семафора.

Эта работа научила меня работать с многопоточным сервером, который выделяет поток на каждого клиента. Это оказалось достаточно удобно, ведь мы получаем сообщения на потоке только от конкретного клиента, а не выстраиваем очередь из сообщений. Так же я получил опыт использвания семафора, и узнал, что будет, если блокировкой общей памяти пренебречь.