

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Курсовой проект по курсу
«Операционные системы»**

Студент: Свиридов Степан Юрьевич
Группа: М8О-206Б-22
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Постановка задачи

Цель работы

Целью работы является:

- Приобретение практических навыков в использовании знаний, полученных в течении курса
- Проведение исследования в выбранной предметной области

Задание

Консоль-серверная игра. Необходимо написать консоль-серверную игру. Необходимо написать 2 программы: сервер и клиент. Сначала запускается сервер, а далее клиенты соединяются с сервером. Сервер координирует клиентов между собой. При запуске клиента игрок может выбрать одно из следующих действий (возможно больше, если предусмотрено вариантом):

- Создать игру, введя ее имя
- Присоединиться к одной из существующих игр по имени игры

Вариант №7:

«Быки и коровы» (угадывать необходимо слова). Общение между сервером и клиентом необходимо организовать при помощи pipe. При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Если кто-то из игроков вышел из игры, то игра должна быть продолжена.

Общие сведения о программе

Программа состоит из двух файлов — server.c и user.c. Запускается server.c, в котором указывается количество игроков и генерируется слово для игры. Затем каждый игрок делает свой ход, а слово, которое он ввел обрабатывает программа user.c и результат обработки посылает в server.c, который оценивает результативность хода каждого игрока и печатает ее. Всего дается 5 попыток, побеждает тот игрок, который первым отгадал слово.

Исходный код

server.c :

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <fcntl.h>
#include <string.h>
#include <pthread.h>

#define PIPE_READ 0
```

```

#define PIPE_WRITE 1
#define MAX_PLAYERS 10
#define MAX_ATTEMPTS 5

typedef struct {
    int fd[2];
    char word[4];
} Player;

char* generateWord() { //функция для генерации рандомного слова
    srand(time(NULL));
    char* word = (char*)malloc(5 * sizeof(char));
    for (int i = 0; i < 4; i++) {
        word[i] = 'a' + rand() % 26;
    }
    word[4] = '\0';
    return word;
}

int main() {
    int numPlayers;
    int numConnectedPlayers = 0;
    Player players[MAX_PLAYERS];

    printf("Введите количество игроков (максимум %d): ", MAX_PLAYERS);
    scanf("%d", &numPlayers);
    numConnectedPlayers = numPlayers;

    if (numPlayers > MAX_PLAYERS) {
        printf("Превышено максимальное количество игроков\n");
        return 1;
    }
    else {
        printf("Итак, будут играть %d игроков\n", numPlayers);
    }

    printf("\n");
    char* word = generateWord();
    printf("Загаданное слово - %s\n", word);
    printf("Загадано слово из 4 букв, ваша задача угадать его, все просто!\n");
    printf("Все игроки ходят по очереди, всего будет 5 попыток, если за 5 попыток никто так и не угадает слово,\n");
    printf("то это будет означать проигрыш. Успехов!\n");
    bool first_iter = true;
    int number_to_exit[5] = {-1, -1, -1, -1, -1};
    int i;

    for (i = 0; i < MAX_ATTEMPTS; i++) {
        int j;
        int k;

        if (!first_iter) {
            printf("Желает ли кто-то выйти из игры? (-1, если никто): ");
            scanf("%d", &number_to_exit[i]);
            if (!(number_to_exit[i] > 0 && number_to_exit[i] <= numPlayers) && number_to_exit[i] != -1) {
                printf("Нет такого игрока\n");
            }
        }
    }
}

```

```

    }
    if (number_to_exit[i] != -1 && (number_to_exit[i] > 0 && number_to_exit[i] <= numPlayers)){
        numConnectedPlayers -= 1;
        printf("Осталось игроков: %d\n", numConnectedPlayers);
    }
}

if (numConnectedPlayers == 0) {
    printf("Все игроки ливнули\n");
    return 1;
}

for (j = 1; j <= numPlayers; j++) {
    if (j != number_to_exit[0] && j != number_to_exit[1] && j != number_to_exit[2] && j !=
number_to_exit[3] && j != number_to_exit[4]) {
        printf("Игрок %d делает ход: ", j);
        scanf("%s", players[j].word);
    }
}

for (k = 1; k <= numPlayers; k++) {
    first_iter = false;
    if (k != number_to_exit[0] && k != number_to_exit[1] && k != number_to_exit[2] && k !=
number_to_exit[3] && k != number_to_exit[4]) {
        if (pipe(players[k].fd) == -1) {
            perror("pipe");
            break;
        }
        int pid;
        pid = fork();
        if (pid == -1) {
            printf("Не удалось создать нового игрока\n");
            break;
        }
        else if (pid == 0) {
            close(players[k].fd[PIPE_READ]);
            if (dup2(players[k].fd[PIPE_WRITE], STDOUT_FILENO) == -1) {
                perror("dup2");
                exit(1);
            }
            close(players[k].fd[PIPE_WRITE]);
            execl("./user", "./user", word, players[k].word, NULL);
            perror("execl");
            return 1;
        }
        else {
            close(players[k].fd[PIPE_WRITE]);
            char buffer[100];
            ssize_t bytesRead = read(players[k].fd[PIPE_READ], buffer, sizeof(buffer));
            if (bytesRead == -1) {
                perror("read");
                exit(1);
            }
            int bulls = atoi(buffer);
            char *stat_2 = buffer;
            while (*stat_2 != ' ') {

```

```

        stat_2++;
    }
    stat_2++;
    int cows = atoi(stat_2);
    printf("Попытка игрока %d:: Быки: %d, Коровы: %d\n", k, bulls, cows);
    if (bulls == 4) {
        printf("Игрок %d победил!\n", k);
        return 0;
    }
    if (i == MAX_ATTEMPTS - 1) {
        printf("\n");
        printf("К сожалению вы проиграли( Никто не смог угадать слово\n");
        return 0;
    }
    wait(NULL);
}
}
}
}
}

```

user.c :

```

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <string.h>

```

void countBullsAndCows(char *guess, char *target, int *bulls, int *cows) { //функция для подсчета коров и быков

```

    int i, j;
    *bulls = 0;
    *cows = 0;

    for (i = 0; i < strlen(guess); i++) {
        if (guess[i] == target[i]) {
            (*bulls)++;
        } else {
            for (j = 0; j < strlen(target); j++) {
                if (guess[i] == target[j]) {
                    (*cows)++;
                    break;
                }
            }
        }
    }
}

```

```

int main (int argc, char *argv[]) {
    int bulls, cows;
    countBullsAndCows(argv[2], argv[1], &bulls, &cows);
    dprintf(STDOUT_FILENO, "%d %d\n", bulls, cows);
    return 0;
}

```

}

Демонстрация работы программы

Введите количество игроков (максимум 10): 3

Итак, будут играть 3 игроков

Загаданное слово - vjw

Загадано слово из 4 букв, ваша задача угадать его, все просто!

Все игроки ходят по очереди, всего будет 5 попыток, если за 5 попыток никто так и не угадает слово,

то это будет означать проигрыш. Успехов!

Игрок 1 делает ход: vjqw

Игрок 2 делает ход: vkwq

Игрок 3 делает ход: jvbw

Попытка игрока 1:: Быки: 2, Коровы: 1

Попытка игрока 2:: Быки: 2, Коровы: 0

Попытка игрока 3:: Быки: 0, Коровы: 3

Желает ли кто-то выйти из игры? (-1, если никто): 2

Осталось игроков: 2

Игрок 1 делает ход: dnak

Игрок 3 делает ход: wjvw

Попытка игрока 1:: Быки: 0, Коровы: 0

Попытка игрока 3:: Быки: 1, Коровы: 3

Желает ли кто-то выйти из игры? (-1, если никто): -1

Игрок 1 делает ход: fdnk

Игрок 3 делает ход: adds

Попытка игрока 1:: Быки: 0, Коровы: 0

Попытка игрока 3:: Быки: 0, Коровы: 0

Желает ли кто-то выйти из игры? (-1, если никто): -1

Игрок 1 делает ход: adad

Игрок 3 делает ход: vjqt

Попытка игрока 1:: Быки: 0, Коровы: 0

Попытка игрока 3:: Быки: 3, Коровы: 0

Желает ли кто-то выйти из игры? (-1, если никто): 1

Осталось игроков: 1

Игрок 3 делает ход: vjw

Попытка игрока 3:: Быки: 4, Коровы: 0

Игрок 3 победил!

Выводы

Я научился создавать простые консоль-серверные игры и строить общение между клиентами и сервером с помощью `pipe`'ов. В целом работа мне показалась весьма интересной, но в то же время довольно трудной.