

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М.В. Ломоносова
Факультет вычислительной математики и кибернетики

ДОКУМЕНТАЦИЯ К ЗАДАНИЮ
ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ИГРЫ «ЖИЗНЬ»

студента 205 учебной группы факультета ВМК МГУ
Бугаевского Владимира Михайловича

Преподаватель: Герасимов Сергей Валерьевич

ВАРИАНТ: 2b

Москва, 2014

Оглавление

1	Алфавитный указатель структур данных	1
1.1	Структуры данных	1
2	Список файлов	3
2.1	Файлы	3
3	Структуры данных	5
3.1	Структура <code>msg_</code>	5
3.1.1	Подробное описание	5
3.1.2	Поля	5
3.1.2.1	<code>mtype</code>	5
3.1.2.2	<code>op</code>	6
4	Файлы	7
4.1	Файл <code>life-client.c</code>	7
4.1.1	Подробное описание	7
4.1.2	Функции	7
4.1.2.1	<code>client_check_partition</code>	7
4.1.2.2	<code>handler</code>	8
4.1.2.3	<code>main</code>	8
4.1.2.4	<code>quit_client</code>	8
4.1.2.5	<code>rcv_server_message</code>	8
4.1.2.6	<code>snd_server_message</code>	8
4.2	Файл <code>life-server.c</code>	9
4.2.1	Подробное описание	10
4.2.2	Функции	10
4.2.2.1	<code>handler</code>	10
4.2.2.2	<code>main</code>	10
4.2.2.3	<code>rcv_client_message</code>	10
4.2.2.4	<code>rcv_worker_message</code>	11
4.2.2.5	<code>server_add</code>	12
4.2.2.6	<code>server_clear</code>	12

4.2.2.7	<code>server_init</code>	12
4.2.2.8	<code>server_next_generation</code>	12
4.2.2.9	<code>server_quit</code>	12
4.2.2.10	<code>server_snap</code>	13
4.2.2.11	<code>server_start</code>	13
4.2.2.12	<code>server_stop</code>	13
4.2.2.13	<code>server_waiting_worker</code>	13
4.2.2.14	<code>snd_client_message</code>	13
4.2.2.15	<code>snd_worker_info</code>	13
4.2.2.16	<code>snd_worker_message</code>	14
4.3	Файл <code>life-worker.c</code>	14
4.3.1	Подробное описание	15
4.3.2	Функции	16
4.3.2.1	<code>main</code>	16
4.3.2.2	<code>rcv_server_message</code>	16
4.3.2.3	<code>rcv_worker_info</code>	16
4.3.2.4	<code>sem_down</code>	16
4.3.2.5	<code>sem_up</code>	16
4.3.2.6	<code>snd_server_message</code>	16
4.3.2.7	<code>worker_add</code>	17
4.3.2.8	<code>worker_clear</code>	17
4.3.2.9	<code>worker_count_neighbours</code>	17
4.3.2.10	<code>worker_define_partners</code>	17
4.3.2.11	<code>worker_del</code>	17
4.3.2.12	<code>worker_init</code>	17
4.3.2.13	<code>worker_is_ready</code>	18
4.3.2.14	<code>worker_quit</code>	18
4.3.2.15	<code>worker_snap</code>	18
4.3.2.16	<code>worker_start</code>	18
4.3.2.17	<code>worker_update_map</code>	18
4.3.2.18	<code>worker_update_memory</code>	18
4.4	Файл <code>life.h</code>	18
4.4.1	Подробное описание	19
4.4.2	Макросы	19
4.4.2.1	<code>worker_being_ready</code>	19
4.4.3	Функции	20
4.4.3.1	<code>quit_message</code>	20
4.4.3.2	<code>write_log</code>	21

Алфавитный указатель	22
----------------------	----

Глава 1

Алфавитный указатель структур данных

1.1 Структуры данных

Структуры данных с их кратким описанием.

<code>msg_</code>	сообщение	5
-------------------	---------------------	---

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

life-client.c	7
life-server.c	9
life-worker.c	14
life.h	18

Глава 3

Структуры данных

3.1 Структура msg_

сообщение

```
#include <life.h>
```

Поля данных

- long `mtype`
тип сообщения
- int `op`
тип команды (операция)
- int `prm1`
первый параметр операции
- int `prm2`
второй параметр операции
- char `mtext` [STRSIZE]
текстовое содержание сообщения

3.1.1 Подробное описание

сообщение

Сообщения используются для обмена данными между

1. клиентом и сервером;
2. сервером и рабочими.

3.1.2 Поля

3.1.2.1 long msg_::mtype

тип сообщения

Это поле может быть равно:

- `pid_client`,
- `pid_server`,

- `pid_worker[i]`,
- `worker_being_ready`.

3.1.2.2 `int msg_::op`

тип команды (операция)

Это поле может быть равно:

- `O_ADD`
- `O_CLEAR`
- `O_START`
- `O_STOP`
- `O_SNAP`
- `M_ATTACH`
- `O_DEL`
- `O_QUIT` либо просто использоваться для передачи информации.

Объявления и описания членов структуры находятся в файле:

- [life.h](#)

Глава 4

Файлы

4.1 Файл life-client.c

```
#include "life.h"
```

Функции

- void `quit_client` (void)
- void `handler` (int signo)
- int `client_check_partition` (int `N`, int `K`)
- int `snd_server_message` (int op, int p1, int p2)
- ssize_t `rcv_server_message` (char c)
- int `main` (int argc, char *argv[])

Переменные

- pid_t `pid_server` = 0
идентификатор процесса-сервера
- pid_t `pid_client` = 0
идентификатор процесса-клиента
- key_t `key` = 0
IPC ключ для создания очереди сообщений
- int `msgid` = 0
идентификатор очереди сообщений

4.1.1 Подробное описание

Клиент принимает команды со стандартного потока ввода и пердает эти команды в понятном для сервера виде. Все типы операций описаны в заголовочном файле "life.h".

4.1.2 Функции

4.1.2.1 int client_check_partition (int N, int K)

Проверка на правильность разбиения.

Аргументы

in	N	число клеток "вселенной" по горизонтали
in	K	число процессов-рабочих

Возвращает

Новое число процессов-рабочих, если разбиение невозможно, иначе - старое число рабочих.

4.1.2.2 void handler (int signo)

Функция обработчик. Обработывает приход сигнала SIGTERM, свидетельствующий о неудачном запуске процесса-сервера "life-server".

Аргументы

in	signo	сигнал
----	-------	--------

4.1.2.3 int main (int argc, char * argv[])

Основная функция клиента. Здесь

1. производится чтение параметров N, M, K из командной строки,
2. проверяется частичная корректность входных параметров,
3. включает аппарат очереди сообщений IPC,
4. включает сервер и осуществляет обмен данными с сервером.

4.1.2.4 void quit_client (void)

Клиент завершает свою работу

4.1.2.5 ssize_t rcv_server_message (char c)

Принять сообщение от сервера.

Аргументы

in	c	включает флаг IPC_NOWAIT
----	---	--------------------------

Возвращает

При успешном завершении системный вызов возвращает действительную длину сообщения, скопированного в поле mtext. При ошибке возвращается -1, а в переменную errno записывается код ошибки.

4.1.2.6 int snd_server_message (int op, int p1, int p2)

Отправить сообщение серверу.

Аргументы

in	op	тип операции
in	p1	первый параметр операции (опционально)
in	p2	второй параметр операции (опционально)

Возвращает

При успешном завершении возвращает 0, а при ошибке — -1, и в переменную errno записывается код ошибки.

4.2 Файл life-server.c

```
#include "life.h"
```

Функции

- ssize_t rcv_worker_message (char c)
- ssize_t rcv_client_message (char c)
- int snd_worker_message (int i, char c)
- int snd_worker_info (int i, char c)
- int snd_client_message (char msg[])
- ssize_t server_waiting_worker (char c)
- void server_init (void)
- void server_add (int x, int y, char c)
- void server_clear (void)
- void server_start (void)
- void server_next_generation (void)
- void server_stop (void)
- void server_snap (void)
- void server_quit (void)
- void handler (int signo)
- int main (int argc, char *argv[])

Переменные

- int N
число клеток во "вселенной" по горизонтали
- int M
число клеток во "вселенной" по вертикали
- int K
число процессов-рабочих
- FILE * logfile
- pid_t pid_server = 0
идентификатор процесса-сервера
- pid_t pid_client = 0
идентификатор процесса-клиента
- pid_t * pid_worker
массив идентификаторов процессов-рабочих
- int * pid_worker_map
карта распараллеливания столбцов "вселенной" между рабочими

- `key_t key`
IPC ключ для очереди сообщений, семафоров и разделяемой памяти
- `int msgid`
идентификатор очереди сообщений
- `int * semid`
массив идентификаторов семафоров для каждого сегмента разделяемой памяти
- `int * shmid`
массив идентификаторов разделяемой памяти для каждой из границ областей "вселенной".
- `int width`
ширина одной полосы
- `int steps = 0`
число поколений, которых предстоит еще построить
- `int counter = 0`
число уведомлений, пришедших от рабочих

4.2.1 Подробное описание

Сервер принимает команды от клиента и проводит распараллеливание операций по процессам-рабочим. Все типы операций описаны в заголовочном файле "life.h".

4.2.2 Функции

4.2.2.1 void handler (int signo)

Функция обработчик. Обработывает приход сигнала SIGTERM, свидетельствующий о неудачном запуске одного из процессов-рабочих "life-server".

Аргументы

in	signo	сигнал
----	-------	--------

4.2.2.2 int main (int argc, char * argv[])

Основная функция сервера. Сервер

1. получает параметры для вселенной,
2. осуществляет обмен данными с клиентом.

4.2.2.3 ssize_t rcv_client_message (char c)

Принять сообщение от клиента.

Аргументы

in	c	включает флаг IPC_NOWAIT
----	---	--------------------------

Возвращает

При успешном завершении системный вызов возвращает действительную длину сообщения, скопированного в поле mtext. При ошибке возвращается -1, а в переменную errno записывается код ошибки.

4.2.2.4 ssize_t rcv_worker_message (char c)

Принять сообщение от рабочего.

Аргументы

in	c	включает флаг IPC_NOWAIT
----	---	--------------------------

Возвращает

При успешном завершении системный вызов возвращает действительную длину сообщения, скопированного в поле mtext. При ошибке возвращается -1, а в переменную errno записывается код ошибки.

4.2.2.5 void server_add (int x, int y, char c)

Сервер отправляет рабочему с командой добавить/удалить клетку во/из вселенную/ой.

Аргументы

in	x	номер строки вселенной
in	y	номер строки вселенной
in	c	выбор операции: 1 - добавить клетку, 0 - удалить клетку

4.2.2.6 void server_clear (void)

Сервер отправляет сообщения рабочим с командой очистить вселенную

4.2.2.7 void server_init (void)

Инициализация сервера. Сервер

1. динамически выделяет память под массивы, описанные в глобальной области кода;
2. подключает очередь сообщений;
3. создает файлов "worker-left" и "worker-right", отвечающих за семафоры и разделяемую память;
4. вызывает K рабочих и отправляет им информационное сообщение.

4.2.2.8 void server_next_generation (void)

Сервер отправляет сообщения рабочим с командой построить следующее поколение

4.2.2.9 void server_quit (void)

Сервер завершает свою работу:

1. посылает сообщения рабочим с командой завершить работу;
2. удаляет разделяемую память и семафоры;
3. освобождение динамической памяти;
4. отключает очередь сообщений;
5. отправляет уведомление клиенту.

4.2.2.10 void server_snap (void)

Сервер отправляет сообщения рабочим с командой сделать скриншот текущего состояния вселенной, а затем пересылает его клиенту

4.2.2.11 void server_start (void)

Сервер устанавливает счетчик поколений "steps"

4.2.2.12 void server_stop (void)

Сервер сбрасывает счетчик поколений "steps"

4.2.2.13 ssize_t server_waiting_worker (char c)

Сервер ожидает подтверждения того, что рабочий закончил выполнение команды

Аргументы

in	c	включает флаг IPC_NOWAIT
----	---	--------------------------

Возвращает

При успешном завершении системный вызов возвращает действительную длину сообщения, скопированного в поле mtext. При ошибке возвращается -1, а в переменную errno записывается код ошибки.

4.2.2.14 int snd_client_message (char msg[])

Отправить сообщение клиенту.

Аргументы

in	msg	текстовое содержание сообщения
----	-----	--------------------------------

Возвращает

При успешном завершении возвращает 0, а при ошибке — -1, и в переменную errno записывается код ошибки.

4.2.2.15 int snd_worker_info (int i, char c)

Отправить информационное сообщение рабочему. Сообщение содержит:

1. номер рабочего;
2. число клеток полосы, обрабатываемой рабочим, по вертикали;
3. число клеток полосы, обрабатываемой рабочим, по горизонтали.

Аргументы

in	i	номер рабочего
in	c	включает флаг IPC_NOWAIT

Возвращает

При успешном завершении возвращает 0, а при ошибке — -1, и в переменную errno записывается код ошибки.

4.2.2.16 int snd_worker_message (int i, char c)

Отправить рабочему сообщение.

Аргументы

in	i	номер рабочего
in	c	включает флаг IPC_NOWAIT

Возвращает

При успешном завершении возвращает 0, а при ошибке — -1, и в переменную errno записывается код ошибки.

4.3 Файл life-worker.c

```
#include "life.h"
```

Функции

- int worker_is_ready (void)
- ssize_t rcv_server_message (char c)
- ssize_t rcv_worker_info (void)
- int snd_server_message (void)
- void worker_define_partners (int i)
- void worker_init (void)
- void worker_quit (void)
- void worker_add (int x, int y)
- void worker_del (int x, int y)
- void worker_clear (void)
- int worker_count_neighbours (int x, int y)
- void sem_down (int i)
- void sem_up (int i)
- void worker_update_map (void)
- void worker_update_memory (void)
- void worker_start (void)
- int worker_snap (int i)
- int main (int argc, char *argv[])

Переменные

- `int M = -1`
число клеток области по вертикали
- `int N = -1`
число клеток области по горизонтали
- `int K = -1`
число процессов-рабочих
- `int id_worker = -1`
индекс рабочего
- `int id_collab_left = -1`
индекс левого соседа рабочего
- `int id_collab_right = -1`
индекс правого соседа рабочего
- `pid_t pid_worker = -1`
идентификатор процесса-рабочего
- `pid_t pid_server = -1`
идентификатор процесса-сервера
- `char ** map_state_curr = NULL`
карта текущего состояния "вселенной".
- `char ** map_state_prev = NULL`
карта последнего смоделированного состояния "вселенной".
- `key_t key = 0`
IPC-ключ
- `int msgid`
идентификатор очереди сообщений
- `int semid [4]`
массив идентификаторов семафоров
 1. `semid[0]` - правая граница левого соседа рабочего;
 2. `semid[1]` - левая граница рабочего;
 3. `semid[2]` - правая граница рабочего;
 4. `semid[3]` - левая граница правого соседа рабочего;
- `int shmid [4]`
массив идентификаторов разделяемой памяти
 1. `semid[0]` - правая граница левого соседа рабочего;
 2. `semid[1]` - левая граница рабочего;
 3. `semid[2]` - правая граница рабочего;
 4. `semid[3]` - левая граница правого соседа рабочего;
- `struct sembuf sops [4]`
массив для управления семафорами
- `char * shmad [4]`
массив указатель на начало адресного пространства разделяемой памяти

4.3.1 Подробное описание

Рабочий принимает команды от сервера и производит моделирование очередного поколения. Все типы операций описаны в заголовочном файле "life.h".

4.3.2 Функции

4.3.2.1 `int main (int argc, char * argv[])`

Основная функция рабочего. Сервер

1. получает количество процессов-рабочих;
2. осуществляет обмен данных с сервером.

4.3.2.2 `ssize_t rcv_server_message (char c)`

Принять сообщение от сервера.

Аргументы

<code>in</code>	<code>c</code>	включает флаг <code>IPC_NOWAIT</code>
-----------------	----------------	---------------------------------------

Возвращает

При успешном завершении системный вызов возвращает действительную длину сообщения, скопированного в поле `mtext`. При ошибке возвращается -1, а в переменную `errno` записывается код ошибки.

4.3.2.3 `ssize_t rcv_worker_info (void)`

Принять информационное сообщение от сервера.

Возвращает

При успешном завершении системный вызов возвращает действительную длину сообщения, скопированного в поле `mtext`. При ошибке возвращается -1, а в переменную `errno` записывается код ошибки.

4.3.2.4 `void sem_down (int i)`

Опустить семафор.

Аргументы

<code>in</code>	<code>i</code>	номер семафора
-----------------	----------------	----------------

4.3.2.5 `void sem_up (int i)`

Поднять семафор.

Аргументы

<code>in</code>	<code>i</code>	номер семафора
-----------------	----------------	----------------

4.3.2.6 `int snd_server_message (void)`

Отправить сообщение серверу.

Возвращает

При успешном завершении возвращает 0, а при ошибке — -1, и в переменную errno записывается код ошибки.

4.3.2.7 void worker_add (int x, int y)

Рабочий добавляет клетку в свою область "вселенной".

Аргументы

in	x	номер строки
in	y	номер столбца

4.3.2.8 void worker_clear (void)

Рабочий освобождает свою область "вселенной".

4.3.2.9 int worker_count_neighbours (int x, int y)

Посчитать количество соседей у данной клетки.

Аргументы

in	x	номер строки
in	y	номер столбца

Возвращает

число соседей

4.3.2.10 void worker_define_partners (int i)

Определить индексы соседей рабочего.

Аргументы

in	i	индекс рабочего
----	---	-----------------

4.3.2.11 void worker_del (int x, int y)

Рабочий удаляет клетку из своей области "вселенной".

Аргументы

in	x	номер строки
in	y	номер столбца

4.3.2.12 void worker_init (void)

Инициализация рабочего. Рабочий

1. подключает очередь сообщений, семафоры и разделяемую память;
2. динамически выделяет память под массивы, описанные в глобальной области кода;
3. очищает таблицу текущего состояния.

4.3.2.13 `int worker_is_ready (void)`

Рабочий сообщает о том, что он выполнил операцию, посланную сервером.

Возвращает

При успешном завершении возвращает 0, а при ошибке — -1, и в переменную `errno` записывается код ошибки.

4.3.2.14 `void worker_quit (void)`

Рабочий завершает свою работу:

1. отключает разделяемую память, семафоры и очередь сообщений;
2. освобождение динамической памяти;
3. отправляет уведомление серверу.

4.3.2.15 `int worker_snap (int i)`

Сделать скриншот строки

Аргументы

in	i	номер строки
----	---	--------------

4.3.2.16 `void worker_start (void)`

Построить очередное поколение.

4.3.2.17 `void worker_update_map (void)`

Обновить карту последнего сгенерированного поколения.

4.3.2.18 `void worker_update_memory (void)`

Обновить разделяемую память, соответствующую границам рабочего.

4.4 Файл `life.h`

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/sem.h>
#include <sys/shm.h>
#include <signal.h>
#include <time.h>
```

Структуры данных

- struct `msg_`
сообщение

Макросы

- #define `O_ADD` 1
добавить клетку из "вселенной".
- #define `O_CLEAR` 2
очистить "вселенную".
- #define `O_START` 3
начать процесс моделирования
- #define `O_STOP` 4
остановить процесс моделирования
- #define `O_SNAP` 5
сделать скриншот текущего состояния "вселенной".
- #define `O_DEL` 6
удалить клетку из "вселенной".
- #define `O_QUIT` 13
завершить работу
- #define `STRSIZE` 4096
длина текстового сообщения
- #define `worker_being_ready` 15
тип сообщения

Функции

- void `write_log` (FILE *f, char msg[])
- void `quit_message` (char str[])

Переменные

- struct `msg_` message

4.4.1 Подробное описание

Сервер принимает команды от клиента и проводит распараллеливание операций по процессам-работчим. Все типы операций описаны в заголовочном файле "life.h".

4.4.2 Макросы

4.4.2.1 #define worker_being_ready 15

тип сообщения

Данный тип сообщений используется для подтверждения работчим того, что он выполнил команду, посланную сервером.

4.4.3 Функции

4.4.3.1 void quit_message (char str[])

Сообщить об аварийном завершении работы и завершить работу.

Аргументы

in	str	текстовое сообщение
----	-----	---------------------

4.4.3.2 void write_log (FILE * f, char msg[])

Записать сообщение в лог-файл.

Аргументы

in	f	лог-файл
in	msg	текстовое сообщение

Предметный указатель

- client_check_partition
 - life-client.c, 7
- handler
 - life-client.c, 8
 - life-server.c, 10
- life-client.c, 7
 - client_check_partition, 7
 - handler, 8
 - main, 8
 - quit_client, 8
 - rcv_server_message, 8
 - snd_server_message, 8
- life-server.c, 9
 - handler, 10
 - main, 10
 - rcv_client_message, 10
 - rcv_worker_message, 10
 - server_add, 12
 - server_clear, 12
 - server_init, 12
 - server_next_generation, 12
 - server_quit, 12
 - server_snap, 12
 - server_start, 13
 - server_stop, 13
 - server_waiting_worker, 13
 - snd_client_message, 13
 - snd_worker_info, 13
 - snd_worker_message, 14
- life-worker.c, 14
 - main, 16
 - rcv_server_message, 16
 - rcv_worker_info, 16
 - sem_down, 16
 - sem_up, 16
 - snd_server_message, 16
 - worker_add, 17
 - worker_clear, 17
 - worker_count_neighbours, 17
 - worker_define_partners, 17
 - worker_del, 17
 - worker_init, 17
 - worker_is_ready, 17
 - worker_quit, 18
 - worker_snap, 18
 - worker_start, 18
 - worker_update_map, 18
 - worker_update_memory, 18
- life.h, 18
 - quit_message, 20
 - worker_being_ready, 19
 - write_log, 21
- main
 - life-client.c, 8
 - life-server.c, 10
 - life-worker.c, 16
- msg_, 5
 - mtime, 5
 - op, 6
- mtime
 - msg_, 5
- op
 - msg_, 6
- quit_client
 - life-client.c, 8
- quit_message
 - life.h, 20
- rcv_client_message
 - life-server.c, 10
- rcv_server_message
 - life-client.c, 8
 - life-worker.c, 16
- rcv_worker_info
 - life-worker.c, 16
- rcv_worker_message
 - life-server.c, 10
- sem_down
 - life-worker.c, 16
- sem_up
 - life-worker.c, 16
- server_add
 - life-server.c, 12
- server_clear
 - life-server.c, 12
- server_init
 - life-server.c, 12
- server_next_generation
 - life-server.c, 12
- server_quit
 - life-server.c, 12
- server_snap
 - life-server.c, 12

server_start
 life-server.c, [13](#)
server_stop
 life-server.c, [13](#)
server_waiting_worker
 life-server.c, [13](#)
snd_client_message
 life-server.c, [13](#)
snd_server_message
 life-client.c, [8](#)
 life-worker.c, [16](#)
snd_worker_info
 life-server.c, [13](#)
snd_worker_message
 life-server.c, [14](#)

worker_add
 life-worker.c, [17](#)
worker_being_ready
 life.h, [19](#)
worker_clear
 life-worker.c, [17](#)
worker_count_neighbours
 life-worker.c, [17](#)
worker_define_partners
 life-worker.c, [17](#)
worker_del
 life-worker.c, [17](#)
worker_init
 life-worker.c, [17](#)
worker_is_ready
 life-worker.c, [17](#)
worker_quit
 life-worker.c, [18](#)
worker_snap
 life-worker.c, [18](#)
worker_start
 life-worker.c, [18](#)
worker_update_map
 life-worker.c, [18](#)
worker_update_memory
 life-worker.c, [18](#)
write_log
 life.h, [21](#)