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

Глава 1

Алфавитный указатель структур данных

1.1 Структуры данных

Структуры данных с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

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	чило процессов-рабочих

Возвращает

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

4.1.2.2 void handler (int signo)

Функция обработчик. Обрабатывает 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 данных 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 процесам-рабочим. Все типы операций описаны в заголовочном файле "life.h".

4.2.2 Функции

4.2.2.1 void handler (int signo)

Функция обработчик. Обрабатывает 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 данных 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_neigbours (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

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.

4.3.2.3 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

in	i	номер семафора

4.3.2.5 void sem_up (int i)

Поднять семафор.

Аргументы

in	i	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_neigbours (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, соотвествующую 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 процесам-рабочим. 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 рабочим 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