uOS

Операционная система реального времени

Версия 0.2
22 августа 2005

Сергей В. Вакуленко

Пожалуйста, присылайте сообщения об ошибках, вопросы и предложения по адресу `uos-list@vak.ru`. Информацию о последних версиях и инструкции по загрузке можно получить на домашней странице uOS – `http://www.vak.ru/uos/`.

uOS — Операционная система реального времени
Сверстано с помощью TEXinfo 1999-09-25.10

Содержание

1 Введение

Данное справочное руководство описывает версию 0.2 операционной системы **uOS**.

uOS представляет собой встраиваемую операционную систему для промышленных применений и систем реального времени.

Операционная система **uOS** распространяется бесплатно, на условиях лицензии GPL (см. Приложение C [GPL], страница 44) с небольшим дополнением (см. Приложение B [Условия распространения], страница 43). Документация по системе **uOS** доступна на условиях лицензии GFDL (см. Приложение D [GFDL], страница 51). На домашней странице **uOS** (`http://www.vak.ru/uos/`) Вы найдете самую свежую информацию.

Пожалуйста, присылайте сообщения об ошибках, вопросы и предложения по адресу `uos-list@vak.ru`.

1.1 Характеристики uOS

uOS представляет собой переносимую масштабируемую операционную систему реального времени с вытесняющей многозадачностью.

Операционная система **uOS** может применяться в промышленных и коммуникационных системах с самым широким диапазоном ресурсов, от 8-битных микроконтроллеров с 16 килобайтами ПЗУ и 2 килобайтами ОЗУ, до 32-битных микропроцессоров. Система поддерживает неограниченное количество задач, приоритетов и ресурсов.

Система **uOS** построена по модульному принципу. Базовый модуль ядра занимает около 2 килобайт ПЗУ и 200 байт ОЗУ. Набор используемых модулей может наращиваться в соответствии с потребностями конкретного применения. В перечень модулей входят драйверы устройств, диспетчер памяти, сетевые протоколы.

При разработке основной упор делался на простоту и эффективность реализации, а также переносимость. С целью облегчения переноса на другие архитектуры микропроцессоров машинно-зависимая часть ядра **uOS** выделена в отдельный блок. См. Глава 6 [Перенос], страница 41. В данной версии доступны переносы на следующие архитектуры:

- Atmel AVR. Используется компилятор GCC 3.2.
- Samsung ARM7TDMI (S3C45x0) в режиме Thumb. Используется компилятор GCC 3.0.4.
- Intel 80x86, под управлением MS-DOS. Используется компилятор Borland Turbo C/C++.
- Intel i386. Используется компилятор GCC 3.3 и загрузчик Grub 0.97.
- Linux 386, в виде отдельной задачи. Может применяться для отладки машинно-независимых частей разрабатываемых систем: алгоритмов обработки данных, диспетчеров памяти, сетевых протоколов.

В данной версии **uOS** имеются следующие драйверы устройств:

- Таймер. Реализован для AVR, Thumb, 80x86, i386 и Linux.
- Асинхронный порт. Реализован для AVR, Thumb и Linux (консоль).
- Неразрушаемая память (NVRAM). Реализован для AVR.
- Сторожевой таймер. Реализован для AVR.
- Контроллер Ethernet Cirrus CS8900A. Реализован для AVR. Используется восьмибитный режим работы контроллера.
- Сетевой протокол SLIP для асинхронного порта (RFC 1055). Реализован для AVR и Thumb.
- Эмулятор сетевого контроллера. Реализован для Linux (/dev/tap0).

- Буквенно-цифровой индикатор ЖКИ 16x2. Реализован для AVR.
- Приемник инфракрасного дистанционного управления Irman (`http://www.evation.com/irman/`). Реализован для AVR.
- Управление скоростью вращения мотора постоянного тока методом ШИМ. Реализован для AVR.

Поддержка сетевых протоколов включает следующие модули:

- Протокол IP+ICMP (RFC ...).
- Протокол ARP (RFC ...).
- Протокол UDP (RFC ...).
- Протокол TCP (RFC ...) - пока не реализован.
- Модуль таблицы маршрутизации.
- Протокол SNMP версии 1.0 (RFC ...), включая модули поддержки базового набора параметров MIB2 (RFC 1213).

Дополнительные модули:

- Динамическое выделение памяти.
- Управление буферами ввода-вывода.
- Вычисление контрольных сумм: CRC8-ATM, CRC8-Dallas, CRC16-CCITT, CRC16-IP.
- Генератор случайных чисел.
- Библиотека символьного ввода-вывода: printf, putchar, getchar.

1.2 Основные концепции

Ядро системы оперирует объектами двух основных типов: 'задача' и 'ресурс'.

Задача представляет собой поток управления (thread). Каждая задача имеет отдельный стек.

В процессе выполнения задача может *захватывать* необходимые ресурсы. При попытке захватить ресурс, занятый другой задачей, задача *блокируется* до момента *освобождения* ресурса. Таким образом, каждая задача может находиться в одном из двух состояний: выполняемом или заблокированном.

Каждая задача имеет целочисленную характеристику - *приоритет*. Если в выполняемом состоянии находится более одной задачи, будет выполняться задача с более высоким (большим) приоритетом. Приоритет задается при создании задачи и может изменяться по ходу выполнения. Нулевой, самый низкий приоритет присваивается фоновой задаче.

Для облегчения отладки каждая задача имеет также *имя* - текстовую строку.

Ресурс представляет собой метод взаимодействия задач. Можно считать ресурсы обобщением семафоров и почтовых ящиков. Кроме захвата и освобождения ресурсов, задачи имеют возможность обмениваться сообщениями. Задача (или несколько задач) может *ожидать сообщения* от ресурса, при этом задача блокируется. Другая задача может *послать сообщение* ресурсу, при этом все ожидающие сообщения задачи переходят в выполняемое состояние и получают посланное сообщение. Посылающая задача не блокируется. Если ни одна задача не ждет сообщения, оно теряется.

В качестве сообщения используется произвольный указатель, обычно ссылающийся на структуру данных, содержащую требуемую информацию.

Этот же механизм сообщений применяется для обработки *аппаратных прерываний*. При захвате ресурса задача может присвоить ему номер аппаратного прерывания, и ожидать сообщения. При возникновении прерывания задача получит сообщение (пустое).

Для каждого аппаратного прерывания, требующего обслуживания, следует создать отдельную задачу. Такая задача обычно в бесконечном цикле ожидает сообщения о прерывании, выполняет необходимые действия с аппаратурой и посылает сообщения другим ресурсам, содержащие результаты обработки.

2 Ядро системы

2.1 Старт системы

При старте системы вызывается функция пользователя `uos_init()`, которая посредством `task_create()` создает необходимое количество задач. После завершения функции `uos_init()` запускается планировщик задач, открываются прерывания и наиболее приоритетная из созданных задач получает управление.

2.1.1 Функция `uos_init()`

```
#include <kernel/uos.h>

void uos_init (void);
```

Функция пользователя, вызываемая системой на этапе инициализации. Ваша программа должна содержать ровно одну функцию `uos_init()`. Главная её цель — создать необходимое количество задач. К этому моменту механизм синхронизации задач еще не функционирует, поэтому единственная функция ядра, которую можно (и нужно) вызывать — это `task_create()`. Функция `uos_init()` не должна вызывать никакие другие функции ядра (`task_xxx()`, `lock_xxx()`) ни непосредственно, ни посредством вызова других модулей. В частности, нельзя инициализировать и обращаться к модулю управления памятью (`mem_alloc()`и пр.).

Во время работы `uos_init()` прерывания запрещены.

После завершения функции `uos_init()` запускается планировщик задач, открываются прерывания и наиболее приоритетная из созданных задач получает управление.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];

void hello (void *data)
{
    debug_puts ("Hello, World!\n");
    uos_halt ();
}

void uos_init (void)
{
    task_create (hello, 0, "hello", 1, stack, sizeof (stack), 0);
}
```

2.1.2 Функция uos_halt()

```
#include <kernel/uos.h>

void uos_halt (void);
```

Завершение работы отлаживаемой системы. Используется при отладке про управлением инструментальной операционной системы, например MS-DOS или Linux. При работе в целевом процессоре не выполняет никаких действий.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];

void hello (void *data)
{
    debug_puts ("Hello, World!\n");
    uos_halt ();
}

void uos_init (void)
{
    task_create (hello, 0, "hello", 1, stack, sizeof (stack), 0);
}
```

2.2 Задачи

Задача представляет собой поток управления (thread). Каждая задача имеет отдельный стек.

В процессе выполнения задача может *захватывать* необходимые ресурсы. При попытке захватить ресурс, занятый другой задачей, задача *блокируется* до момента *освобождения* ресурса. Таким образом, каждая задача может находиться в одном из двух состояний: выполняемом или заблокированном.

Каждая задача имеет целочисленную характеристику - *приоритет*. Если в выполняемом состоянии находится более одной задачи, будет выполняться задача с более высоким (большим) приоритетом. Приоритет задается при создании задачи и может изменяться по ходу выполнения. Нулевой, самый низкий приоритет присваивается фоновой задаче.

Для облегчения отладки каждая задача имеет также *имя* - текстовую строку.

2.2.1 Тип `task_t`

```
#include <kernel/uos.h>

typedef struct _task_t {
    ...
} task_t;

task_t *task_create (void (*func)(void*), void *arg, char *name,
    int priority, char *stack, int stacksz);
void task_exit (void *message);
void task_delete (task_t *task, void *message);
void *task_wait (task_t *task);
int task_stack_avail (task_t *task);
char *task_name (task_t *task);
int task_priority (task_t *task);
void task_set_priority (task_t *task, int priority);
```

Структура, описывающая задачу (поток управления). Каждая задача выполняется независимо от других. Операционная система производит переключение между задачами, всякий раз выбирая для выполнения задачу, имеющую наивысший *приоритет* (priority). Для упрощения отладки каждой задаче присваивается *имя* (name). Имеется специальная задача с именем “idle” и приоритетом 0, получающая управление при отсутствии готовых к выполнению задач.

Каждая задача имеет отдельный *стек* (stack), использующийся при выполнении для хранения адресов возврата и параметров вызываемых функций. В стеке также сохраняется содержимое регистров процессора при переключении задач. Память для стека содержится в структуре `task_t`.

Для работы с задачами применяются следующие функции:

Функция	**Описание**
‘`task_create`’	Создание задачи
‘`task_exit`’	Завершение текущей задачи
‘`task_delete`’	Принудительное завершение задачи
‘`task_wait`’	Ожидание завершения задачи
‘`task_stack_avail`’	Вычисление размера стека, не использованного задачей
‘`task_name`’	Запрос имени задачи
‘`task_priority`’	Запрос приоритета задачи
‘`task_set_priority`’	Установка приоритета задачи

2.2.2 Функция `task_create()`

```
#include <kernel/uos.h>

task_t *task_create (void (*func)(void*), void *arg, char *name,
    int prio, char *stack, int stacksz);
```

Создание новой задачи (потока) и помещение ее в очередь готовых к выполнению задач. Переключение задач не происходит.

Параметры

`func` Функция, которая будет вызвана при первом переключении на новую задачу. Не должна возвращать управление. Для завершения задачи следует применять функцию `task_exit()`.

`arg` Данное значение будет передано в качестве аргумента при вызове функции `func`.

`name` Текстовая строка, обозначающая "имя" данной задачи. Используется при выдаче отладочных сообщений. Доступно посредством функции `task_name`.

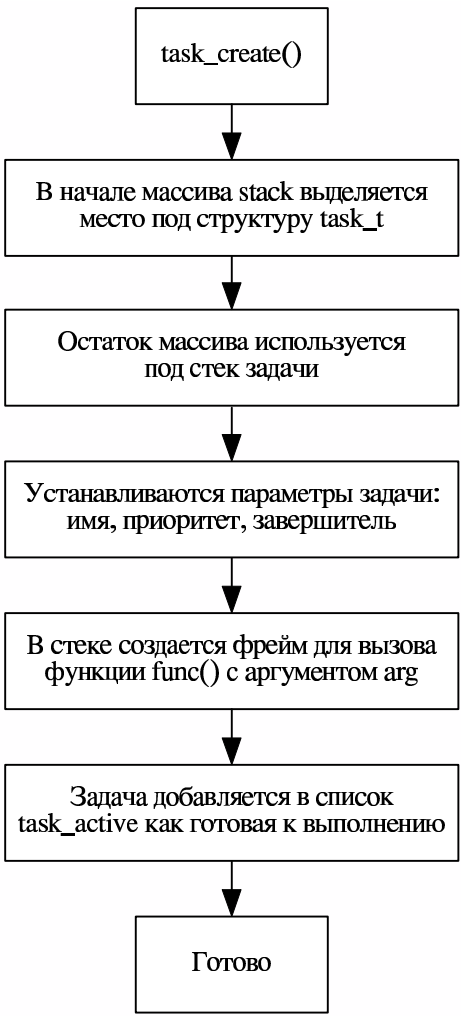
`prio` Приоритет задачи, положительное целое значение. Задача с большим значением `prio` имеет более высокий приоритет.

`stack` Массив памяти, в которой будут размещаться структура данных задачи (`task_t`) и стек.

`stacksz` Размер массива `stack` в байтах. Минимальное значение зависит от процессора и выполняемой задачи. Например, для процессора AVR, рекомендуемый размер составляет 200-300 байтов. При отладке рекомендуется задать размер стека с запасом, затем с помощью функции `task_stack_avail` определить требуемый расход памяти для каждой задачи, и при компоновке реальной системы использовать оптимизированные значения. Для экономии стека не рекомендуется исмользовать в функциях локальные массивы и вызов `alloca()`.

Возвращаемое значение

Возвращает указатель на структуру `task_t`, расположенную в массиве stack.



Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];

void hello (void *data)
{
    debug_puts ("Hello, World!\n");
    uos_halt ();
}

void uos_init (void)
{
    task_create (hello, 0, "hello", 1, stack, sizeof (stack), 0);
}
```

2.2.3 Функция task_exit()

```
#include <kernel/uos.h>

void task_exit (void *status);
```

Завершение текущей задачи со статусом `status`. Происходит переключение задач.

Если некая задача ожидает посредством функции `task_wait()` завершения текущей задачи, она получит `status` в качестве возвращаемого значения.

Параметры

`status` Произвольный указатель, передаваемый в качестве сообщения функции `task_wait()`.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

task_t *task2;
lock_t finalizer;
char stack1 [400], stack2 [400];

void main1 (void *data)
{
    void *msg;

    /* Задача 1 имеет более высокий приоритет и стартует раньше */
    debug_puts ("Task 1: waiting for task2\n");
    msg = task_wait (task2);
    debug_printf ("Task 1: task 2 returned '%s'\n", msg);
    uos_halt ();
}

void main2 (void *data)
{
    debug_puts ("Task 2: returning 'Hello'\n");
    task_exit ("Hello");
}

void uos_init (void)
{
    task_create (main1, 0, "task1", 2, stack1, sizeof (stack1), 0);
    task2 = task_create (main2, 0, "task2", 1, stack2, sizeof (stack2),
        &finalizer);
}
```

2.2.4 Функция `task_delete()`

```
#include <kernel/uos.h>

void task_delete (task_t *task, void *status);
```

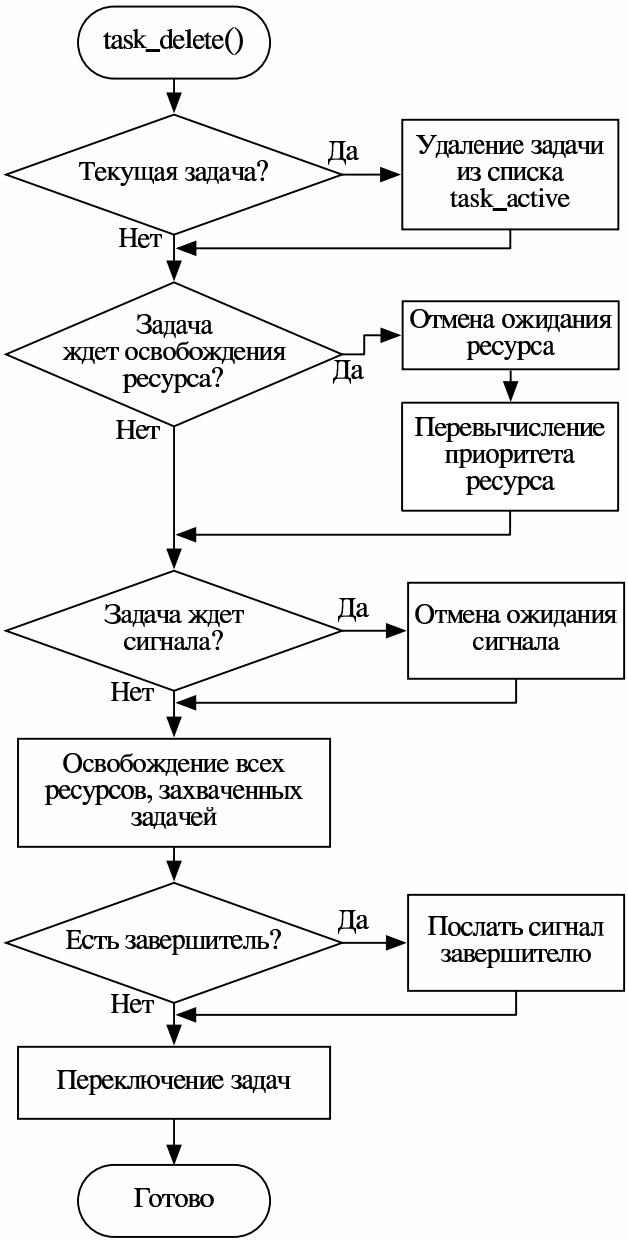
Принудительное завершение указанной задачи со статусом `status`. Если завершаемая задача является текущей, происходит переключение задач.

Если некая задача ожидает посредством функции `task_wait()` завершения текущей задачи, она получит `status` в качестве возвращаемого значения.

Параметры

`task` Указатель на структуру данных завершаемой задачи.

`status` Произвольный указатель, передаваемый в качестве сообщения функции `task_wait()`.


task_delete()
Текущая задача?
Да
Удаление задачи из списка task_active
Нет
Задача ждет освобождения ресурса?
Да
Отмена ожидания ресурса
Перевычисление приоритета ресурса
Нет
Задача ждет сигнала?
Да
Отмена ожидания сигнала
Нет
Освобождение всех ресурсов, захваченных задачей
Есть завершитель?
Да
Послать сигнал завершителю
Нет
Переключение задач
Готово

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

task_t *task2;
lock_t finalizer, event;
char stack1 [400], stack2 [400], stack3 [400];

void main1 (void *data)
{
    void *msg;

    /* Задача 1 имеет самый высокий приоритет и стартует раньше.
     * Ждем завершения задачи 2. */
    debug_puts ("Task 1: waiting for task2\n");
    msg = task_wait (task2);
    debug_printf ("Task 1: task 2 returned '%s'\n", msg);
    uos_halt ();
}

void main2 (void *data)
{
    /* Задача 2 стартует второй и ждет некоего события. */
    debug_puts ("Task 2: waiting for some event\n");
    lock_wait (&event);
}

void main3 (void *data)
{
    /* Задача 2 стартует последней и завершает задачу 2. */
    debug_puts ("Task 3: killing task 2\n");
    task_delete (task2, "Killed");
    task_exit (0);
}

void uos_init (void)
{
    task_create (main1, 0, "task1", 3, stack1, sizeof (stack1), 0);
    task2 = task_create (main2, 0, "task2", 2, stack2, sizeof (stack2),
        &finalizer);
    task_create (main3, 0, "task3", 1, stack3, sizeof (stack3), 0);
}
```

2.2.5 Функция `task_wait()`

```
#include <kernel/uos.h>

void *task_wait (task_t *task);
```

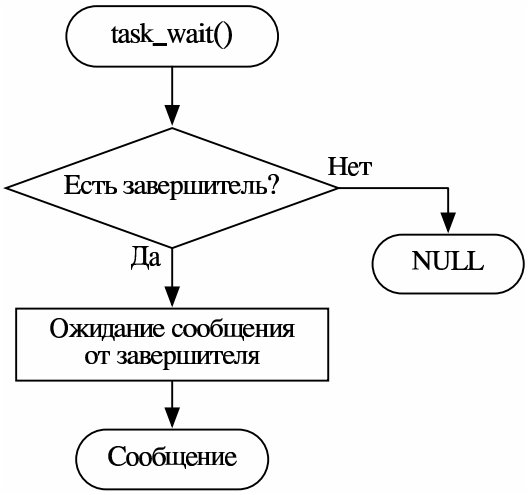
Ожидание завершения указанной задачи. Текущая задача останавливается и происходит переключение задач.

Параметры

`task` Задача, завершения которой ожидается.

Возвращаемое значение

Возвращает статус завершения задачи, установленный вызовами `task_exit()` или `task_delete()`.



Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

task_t *task2;
lock_t finalizer;
char stack1 [400], stack2 [400];

void main1 (void *data)
{
    void *msg;

    /* Задача 1 имеет более высокий приоритет и стартует раньше */
    debug_puts ("Task 1: waiting for task2\n");
    msg = task_wait (task2);
    debug_printf ("Task 1: task 2 returned '%s'\n", msg);
    uos_halt ();
}

void main2 (void *data)
{
    debug_puts ("Task 2: returning 'Hello'\n");
    task_exit ("Hello");
}

void uos_init (void)
{
    task_create (main1, 0, "task1", 2, stack1, sizeof (stack1), 0);
    task2 = task_create (main2, 0, "task2", 1, stack2, sizeof (stack2),
        &finalizer);
}
```

2.2.6 Функция task_stack_avail()

```
#include <kernel/uos.h>

int task_stack_avail (task_t *task);
```

Запрос размера неиспользованной части стека указанной задачи. Переключение задач не происходит.

Вычислить размера стека, необходимый для конкретной задачи, довольно непросто. Рекомендуется при отладке задавать размер стека с запасом, затем с помощью функции `task_stack_avail` определить требуемый расход памяти для каждой задачи, и при компоновке реальной системы использовать оптимизированные значения. Для экономии стека не рекомендуется исмользовать в функциях локальные массивы и вызов `alloca()`.

Параметры

`task` Задача, размер стека которой требуется опросить.

Возвращаемое значение

Возвращает положительное целое значение — размер в байтах неиспользованной части стека указанной задачи.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];
task_t *task;

void hello (void *data)
{
    int unused;

    unused = task_stack_avail (task);
    debug_printf ("Stack %d bytes\n", sizeof (stack));
    debug_printf ("Unused %d bytes\n", unused);
    uos_halt ();
}

void uos_init (void)
{
    task = task_create (hello, 0, "hello", 1, stack, sizeof (stack), 0);
}
```

2.2.7 Функция task_name()

```
#include <kernel/uos.h>

char *task_name (task_t *task);
```

Запрос имени указанной задачи. Имя задается при создании задачи и используется для отладочной печати. Переключение задач не происходит.

Параметры

`task` Задача, имя которой требуется запросить.

Возвращаемое значение

Возвращает указатель на текстовую строку — имя задачи.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];
task_t *task;

void hello (void *data)
{
    debug_printf ("Task name = '%s'\n", task_name (task));
    uos_halt ();
}

void uos_init (void)
{
    task = task_create (hello, 0, "hello", 1, stack, sizeof (stack), 0);
}
```

2.2.8 Функция task_priority()

```
#include <kernel/uos.h>

int task_priority (task_t *task);
```

Запрос приоритета указанной задачи. Приоритет задается при создании задачи и может изменяться функцией `task_set_priority()`. Переключение задач не происходит.

Параметры

`task` Задача, приоритет которой требуется запросить.

Возвращаемое значение

Возвращает целое положительное значение — приоритет указанной задачи.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];
task_t *task;

void hello (void *data)
{
    debug_printf ("Task priority = %d\n", task_priority (task));
    uos_halt ();
}

void uos_init (void)
{
    task = task_create (hello, 0, "hello", 123, stack, sizeof (stack), 0);
}
```

2.2.9 Функция task_set_priority()

```
#include <kernel/uos.h>

void task_set_priority (task_t *task, int priority);
```

Изменение приоритета указанной задачи. Если новое значение приоритета превышает приоритет текущей задачи, может произойти переключение задач.

Параметры

`task` Задача, приоритет которой требуется изменить.

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>

char stack [400];
task_t *task;

void hello (void *data)
{
    debug_printf ("Task priority = %d\n", task_priority (task));
    task_set_priority (task, 456);
    debug_printf ("New priority = %d\n", task_priority (task));
    uos_halt ();
}

void uos_init (void)
{
    task = task_create (hello, 0, "hello", 123, stack, sizeof (stack), 0);
}
```

2.3 Ресурсы

Ресурс представляет собой метод взаимодействия задач. Можно считать ресурсы обобщением семафоров и почтовых ящиков. Рекомендуется использовать ресурсы для защиты структур данных, доступ к которым производится из нескольких задач (критические области).

Кроме захвата и освобождения ресурсов, задачи имеют возможность обмениваться сообщениями. Задача (или несколько задач) может *ожидать сообщения* от ресурса, при этом задача блокируется. Другая задача может *послать сообщение* ресурсу, при этом все ожидающие сообщения задачи переходят в выполняемое состояние и получают посланное сообщение. Посылающая задача не блокируется. Если ни одна задача не ждет сообщения, оно теряется.

В качестве сообщения используется произвольный указатель, обычно ссылающийся на структуру данных, содержащую требуемую информацию.

2.3.1 Тип `lock_t`

```
#include <kernel/uos.h>

typedef struct _lock_t {
    ...
} lock_t;

void lock_take (lock_t *lock);
void lock_release (lock_t *lock);
int lock_try (lock_t *lock);
void lock_signal (lock_t *lock, void *message);
void *lock_wait (lock_t *lock);
```

Для работы с ресурсами применяются следующие функции:

Функция	**Описание**
'`lock_take`'	Захват ресурса
'`lock_release`'	Освобождение ресурса
'`lock_try`'	Попытка захвата ресурса
'`lock_signal`'	Посылка сообщения
'`lock_wait`'	Ожидание сообщения

2.3.2 Функция `lock_take()`

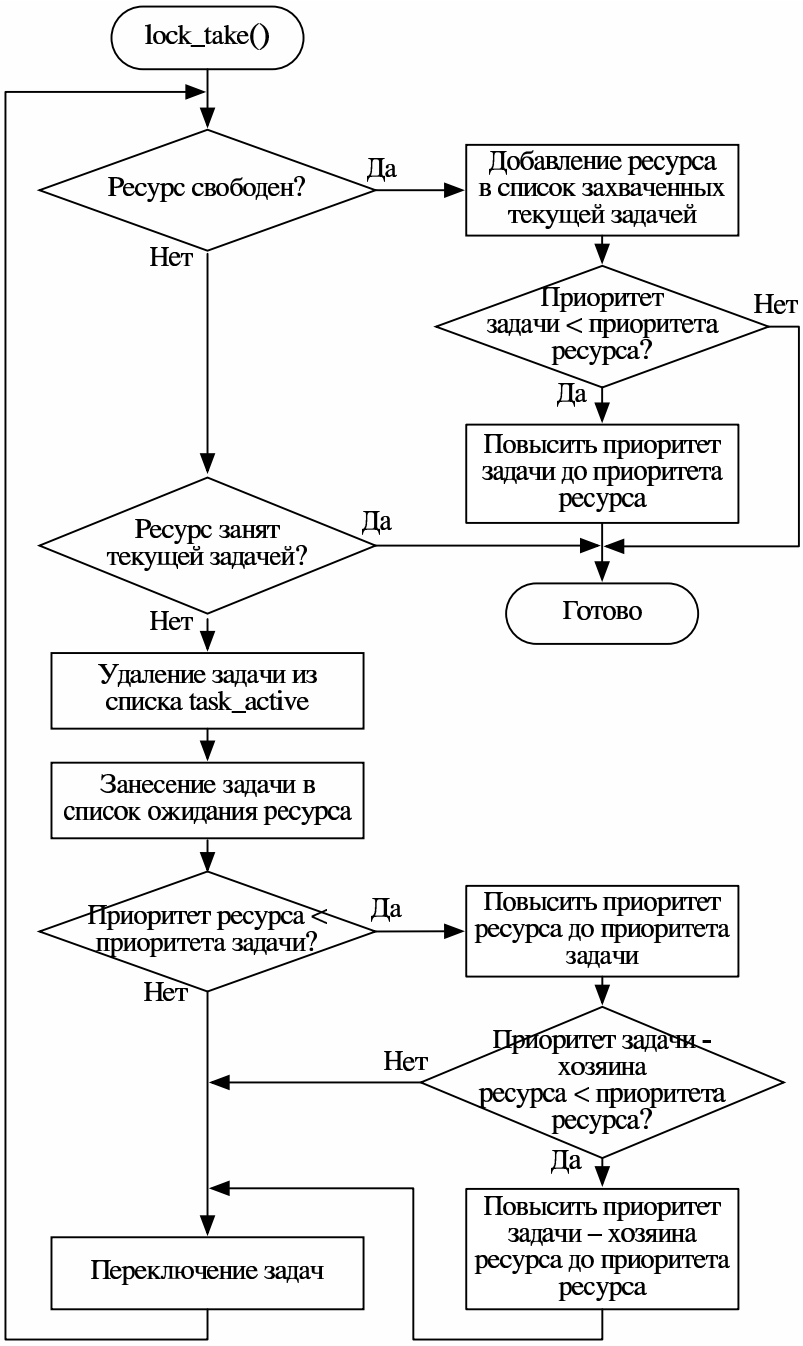
```
#include <kernel/uos.h>

void lock_take (lock_t *lock);
```

Захват ресурса. Если ресурс свободен, переключение задач не происходит. Если ресурс занят другой задачей, текущая задача блокируется до освобождения ресурса.

Если ресурс связан с аппаратным прерыванием (см. Раздел 2.4.1 [lock_take_irq], страница 31), обработка прерывания блокируется на время удержания ресурса. Отложенное прерывание будет обработано при освобождении ресурса вызовом `lock_release()` или `lock_wait()`.

Параметры

`lock` Ресурс, который требуется захватить.


lock_take()
Ресурс свободен?
Да
Нет
Добавление ресурса в список захваченных текущей задачей
Приоритет задачи < приоритета ресурса?
Нет
Да
Повысить приоритет задачи до приоритета ресурса
Ресурс занят текущей задачей?
Да
Нет
Готово
Удаление задачи из списка task_active
Занесение задачи в список ожидания ресурса
Приоритет ресурса < приоритета задачи?
Да
Нет
Повысить приоритет ресурса до приоритета задачи
Приоритет задачи - хозяина ресурса < приоритета ресурса?
Нет
Да
Повысить приоритет задачи – хозяина ресурса до приоритета ресурса
Переключение задач

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>
#include <timer/timer.h>

lock_t lock;
timer_t timer;
char stack1 [4000], stack2 [4000];

void main1 (void *data)
{
    /* Задача 1 имеет более высокий приоритет и стартует раньше */
    debug_puts ("Task 1: taking the lock\n");
    lock_take (&lock);

    debug_puts ("Task 1: sleeping 1 second\n");
    timer_delay (&timer, 1000);

    debug_puts ("Task 1: releasing the lock\n");
    lock_release (&lock);

    task_exit (0);
}

void main2 (void *data)
{
    /* Задача 2 приостанавливается до освобождения ресурса */
    debug_puts ("Task 2: taking the lock\n");
    lock_take (&lock);

    debug_puts ("Task 2: got the lock\n");
    lock_release (&lock);
    uos_halt ();
}

void uos_init (void)
{
    task_create (main1, 0, "task1", 2, stack1, sizeof (stack1), 0);
    task_create (main2, 0, "task2", 1, stack2, sizeof (stack2), 0);
    timer_init (&timer, 100, KHZ, 10);
}
```

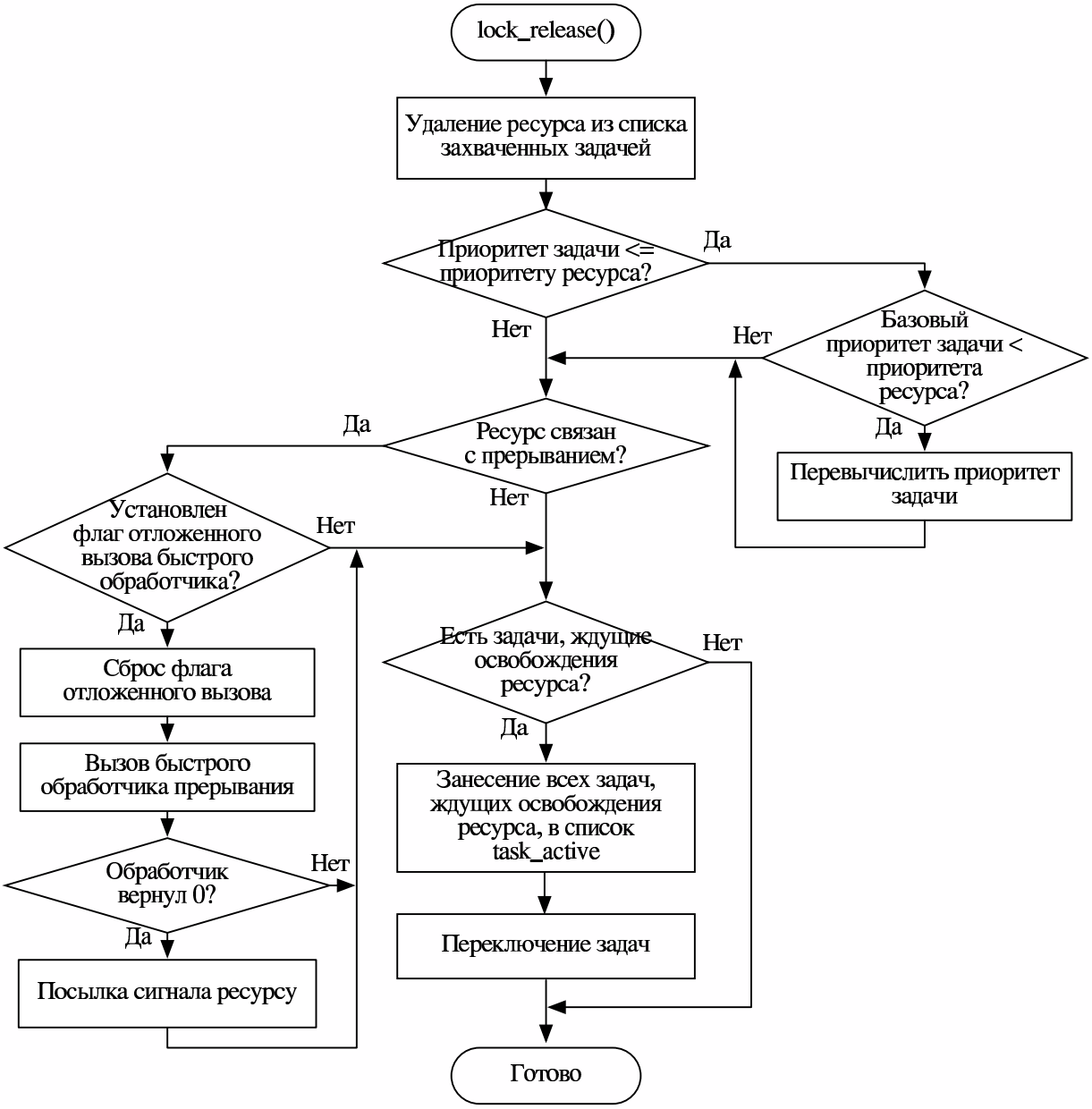
2.3.3 Функция `lock_release()`

```
#include <kernel/uos.h>

void lock_release (lock_t *lock);
```

Освобождение ресурса. Если ресурс связан с аппаратным прерыванием (см. Раздел 2.4.1 [lock_take_irq], страница 31), и за время удержания ресурса возникло прерывание, производится его обработка. Может произойти переключение задач.

Параметры

`lock` Ресурс, который требуется освободить.


lock_release()
Удаление ресурса из списка захваченных задачей
Приоритет задачи <= приоритету ресурса?
Да
Нет
Базовый приоритет задачи < приоритета ресурса?
Нет
Да
Перевычислить приоритет задачи
Ресурс связан с прерыванием?
Да
Нет
Установлен флаг отложенного вызова быстрого обработчика?
Нет
Да
Сброс флага отложенного вызова
Вызов быстрого обработчика прерывания
Обработчик вернул 0?
Нет
Да
Посылка сигнала ресурсу
Есть задачи, ждущие освобождения ресурса?
Нет
Да
Занесение всех задач, ждущих освобождения ресурса, в список task_active
Переключение задач
Готово

Пример

```
#include <runtime/lib.h>
#include <kernel/uos.h>
#include <timer/timer.h>

lock_t lock;
timer_t timer;
char stack1 [4000], stack2 [4000];

void main1 (void *data)
{
    /* Задача 1 имеет более высокий приоритет и стартует раньше */
    debug_puts ("Task 1: taking the lock\n");
    lock_take (&lock);

    debug_puts ("Task 1: sleeping 1 second\n");
    timer_delay (&timer, 1000);

    debug_puts ("Task 1: releasing the lock\n");
    lock_release (&lock);

    task_exit (0);
}

void main2 (void *data)
{
    /* Задача 2 приостанавливается до освобождения ресурса */
    debug_puts ("Task 2: taking the lock\n");
    lock_take (&lock);

    debug_puts ("Task 2: got the lock\n");
    lock_release (&lock);
    uos_halt ();
}

void uos_init (void)
{
    task_create (main1, 0, "task1", 2, stack1, sizeof (stack1), 0);
    task_create (main2, 0, "task2", 1, stack2, sizeof (stack2), 0);
    timer_init (&timer, 100, KHZ, 10);
}
```

2.3.4 Функция `lock_try()`

```
#include <kernel/uos.h>

int lock_try (lock_t *lock);
```

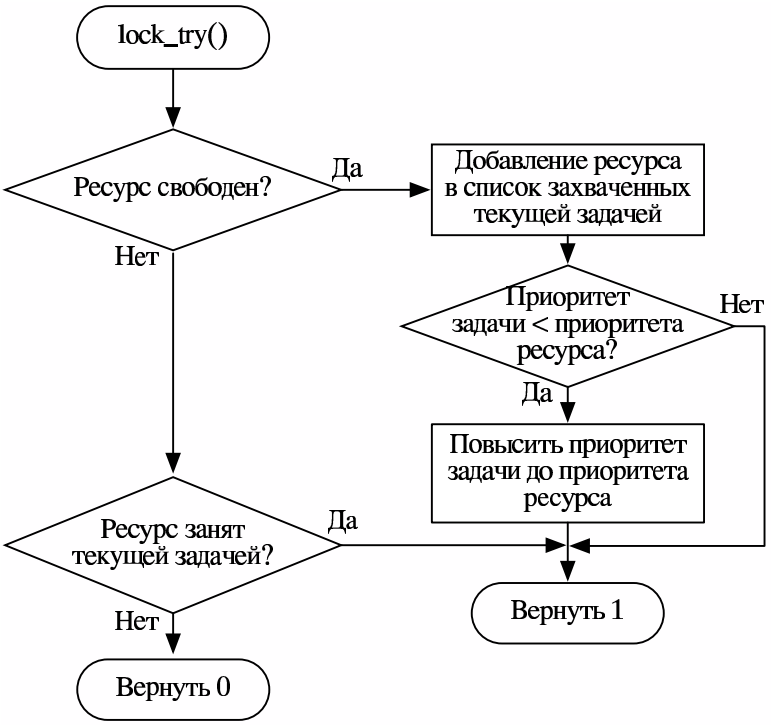
Попытка захвата ресурса. Если ресурс свободен, он захватывается. Переключение задач не происходит.

Параметры

`lock` Ресурс, который требуется захватить.

Возвращаемое значение

Возвращает 1 если ресурс успешно захвачен. В случае неуспеха возвращается 0.



Пример

2.3.5 Функция `lock_signal()`

```
#include <kernel/uos.h>

void lock_signal (lock_t *lock, void *message);
```

Посылка сообщения ресурсу. Может произойти переключение задач. Если ни одна задача не ждет сообщения, оно теряется.

Параметры

`lock` Ресурс, которому посылается сообщение.

`message` Сообщение — произвольный указатель.

Пример

2.3.6 Функция `lock_wait()`

```
#include <kernel/uos.h>

void *lock_wait (lock_t *lock);
```

Ожидание сообщения. Текущая задача блокируется до момента, пока другая задача не пошлет сообщение ресурсу вызовом `lock_signal()`. Если ресурс был захвачен текущей задачей, на время ожидания он освобождается. Происходит переключение задач.

Для надежной доставки сообщений требуется, чтобы ожидающая задача предварительно захватила ресурс. Возможно ожидание сообщений на незахваченном ресурсе, но при этом надежная доставка сообщений не гарантируется.

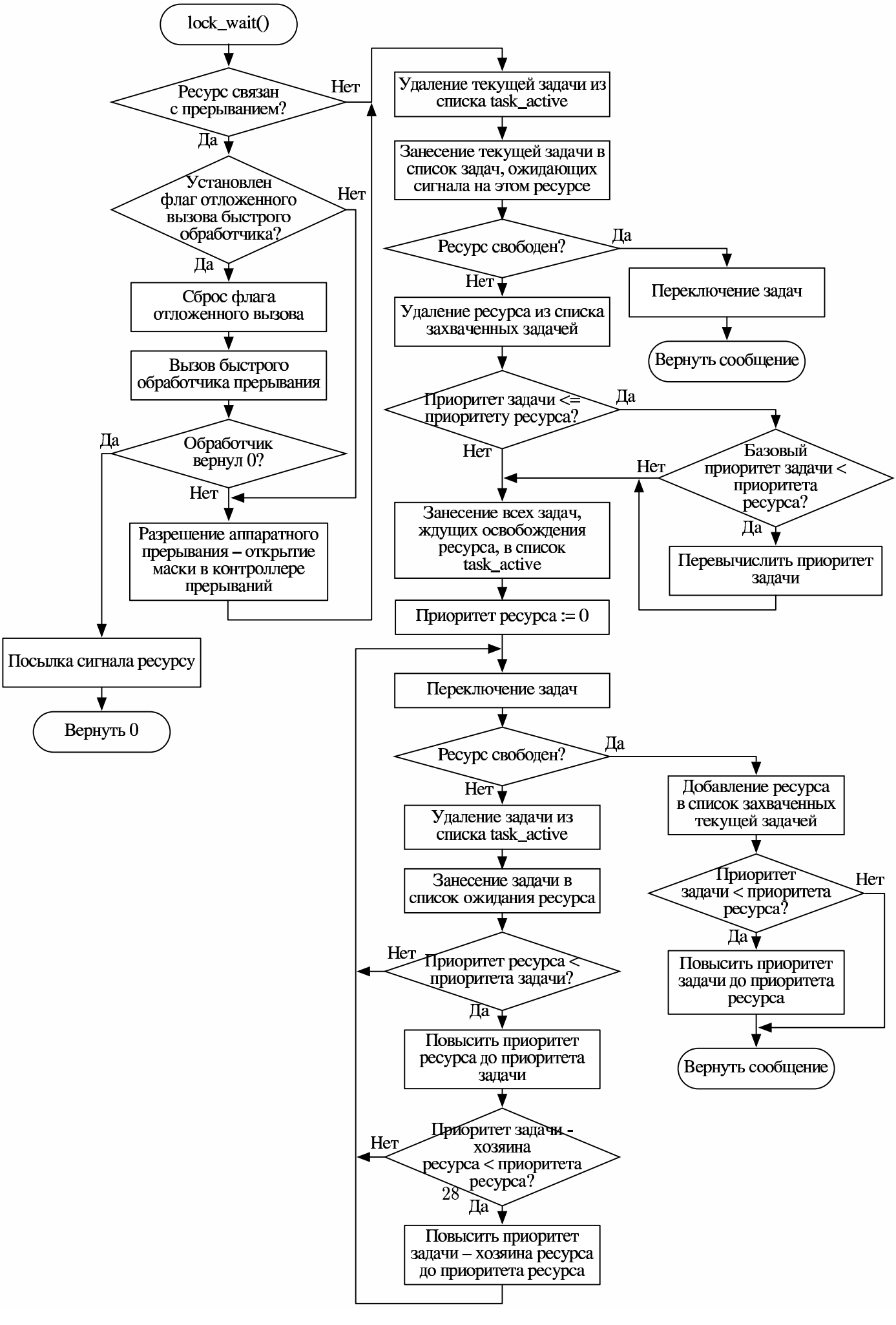
Если ресурс связан с аппаратным прерыванием (см. Раздел 2.4.1 [lock_take_irq], страница 31), и за время удержания ресурса возникло прерывание, производится его обработка, затем производится аппаратное разрешение (открытие маски) данного прерывания в контроллере прерываний.

Параметры

`lock` Ресурс, для которого ожидается сообщение.

Возвращаемое значение

Возвращает указатель-сообщение, посланное вызовом `lock_signal()`.


lock_wait()
Ресурс связан с прерыванием?
Нет
Да
Установлен флаг отложенного вызова быстрого обработчика?
Нет
Да
Сброс флага отложенного вызова
Вызов быстрого обработчика прерывания
Да
Обработчик вернул 0?
Нет
Разрешение аппаратного прерывания – открытие маски в контроллере прерываний
Посылка сигнала ресурсу
Вернуть 0
Удаление текущей задачи из списка task_active
Занесение текущей задачи в список задач, ожидающих сигнала на этом ресурсе
Ресурс свободен?
Да
Переключение задач
Вернуть сообщение
Нет
Удаление ресурса из списка захваченных задачей
Приоритет задачи <= приоритету ресурса?
Да
Базовый приоритет задачи < приоритета ресурса?
Нет
Да
Перевычислить приоритет задачи
Нет
Занесение всех задач, ждущих освобождения ресурса, в список task_active
Приоритет ресурса := 0
Переключение задач
Ресурс свободен?
Да
Добавление ресурса в список захваченных текущей задачей
Нет
Удаление задачи из списка task_active
Занесение задачи в список ожидания ресурса
Приоритет задачи < приоритета ресурса?
Нет
Да
Повысить приоритет задачи до приоритета ресурса
Вернуть сообщение
Нет
Приоритет ресурса < приоритета задачи?
Да
Повысить приоритет ресурса до приоритета задачи
Приоритет задачи - хозяина ресурса < приоритета ресурса?
Нет
Да
Повысить приоритет задачи – хозяина ресурса до приоритета ресурса

Пример

2.4 Прерывания

Для обработки аппаратных прерываний применяется механизм сообщений. При захвате ресурса задача может присвоить ему номер аппаратного прерывания, и ожидать сообщения. При возникновении прерывания задача получит сообщение (пустое).

После вызова `lock_take_irq()` аппаратное прерывание считается связанным с указанным ресурсом. Для каждого аппаратного прерывания, требующего обслуживания, следует создать отдельную задачу. Такая задача обычно в бесконечном цикле ожидает сообщения о прерывании, выполняет необходимые действия с аппаратурой и посылает сообщения другим ресурсам, содержащие результаты обработки.

Такой метод обработки прерываний обладает простотой и наглядностью, но имеет один недостаток: он требует переключения задач на каждое прерывание. Для тех случаев, когда нужна более быстрая реакция на прерывания, применяется дополнительный механизм *быстрой обработки*.

Быстрый обработчик представляет собой функцию, которая регистрируется при захвате ресурса прерывания, и вызывается ядром при наступлении прерывания, обеспечивая максимально высокую скорость реакции. Обслужив прерывание, быстрый обработчик принимает решение и возвращает ядру признак: следует ли посылать основной задаче-обработчику сообщение о прерывании. Таким образом, работа по обработке разделяется между основной задачей прерывания и быстрым обработчиком: срочные действия выполняет быстрый обработчик, а “медленные” - основная задача.

Пока ресурс захвачен функцией `lock_take()`, быстрый обработчик не может быть вызван "немедленно", так как это привело бы к конфликту совместного доступа к данным. В этом случае вызов быстрого обработчика откладывается до освобождения ресурса функциями `lock_release()` или `lock_wait()`.

Поскольку быстрый обработчик выполняется в обход механизма синхронизации задач, на него накладываются некоторые ограничения. Во избежание конфликта с другими задачами, он имеет право работать только с данными, защищенными ресурсом прерывания. Он не должен вызывать никакие функции ядра (`task_xxx()`, `lock_xxx()`) ни непосредственно, ни посредством вызова других модулей. В частности, он не может обращаться к модулю управления памятью (`mem_alloc()`и пр.). Подобные действия должны перекладываться на задачу-обработчик.

При возникновении прерывания выполняются следующие действия:

1. Аппаратный запрет прерывания (закрывается маска в контроллере прерываний).
2. При наличии быстрого обработчика:
   a. Если ресурс занят - вызов быстрого обработчика откладывается до освобождения ресурса. Обработка прерывания закончена.
   b. Вызов быстрого обработчика. Если обработчик вернул ненулевой код - обработка прерывания закончена.
3. Посылка сообщения ресурсу.
4. Переключение задач.

Аппаратное разрешение прерывания (открытие маски в контроллере прерываний) производится при вызове функции `lock_wait()`.

```
#include <kernel/uos.h>

typedef int (*handler_t) (void*);

void lock_take_irq (lock_t *lock, int irq, handler_t func, void *arg);
void lock_release_irq (lock_t *lock);
```

Для работы с прерываниями применяются следующие функции:

Функция	**Описание**
'lock_take_irq'	Захват прерывания
'lock_release_irq'	Освобождение прерывания
'lock_wait'	Ожидание прерывания

2.4.1 Функция lock_take_irq()

```
#include <kernel/uos.h>

void lock_take_irq (lock_t *lock, int irq, handler_t func, void *arg);
```

Захват ресурса и привязка его к аппаратному прерыванию с указанным номером. Если ресурс свободен, переключение задач не происходит. Если ресурс занят другой задачей, текущая задача блокируется до освобождения ресурса.

После вызова `lock_take_irq()` аппаратное прерывание считается связанным с указанным ресурсом. При возникновении прерываний будет вызван быстрый обработчик, а ресурс будет получать сообщение. При последующих захватах ресурса функцией `lock_take()` вызов быстрого обработчика откладывается на время удержания ресурса. Отложенное прерывание будет обработано при освобождении ресурса вызовом `lock_release()`, или `lock_wait()`.

Для каждого аппаратного прерывания, требующего обслуживания, следует создать отдельную задачу. Такая задача должна работать по следующему алгоритму:

1. захватить прерывание вызовом `lock_take_irq()`
2. инициализировать обслуживаемую аппаратуру
3. в цикле ожидать прерывания вызовом `lock_wait()`
4. выполнить необходимые действия с аппаратурой, сохранить данные и т.п.
5. при необходимости послать сообщения другим ресурсам вызовом `lock_signal()`
6. перейти к п.3, продолжив цикл ожидания прерывания

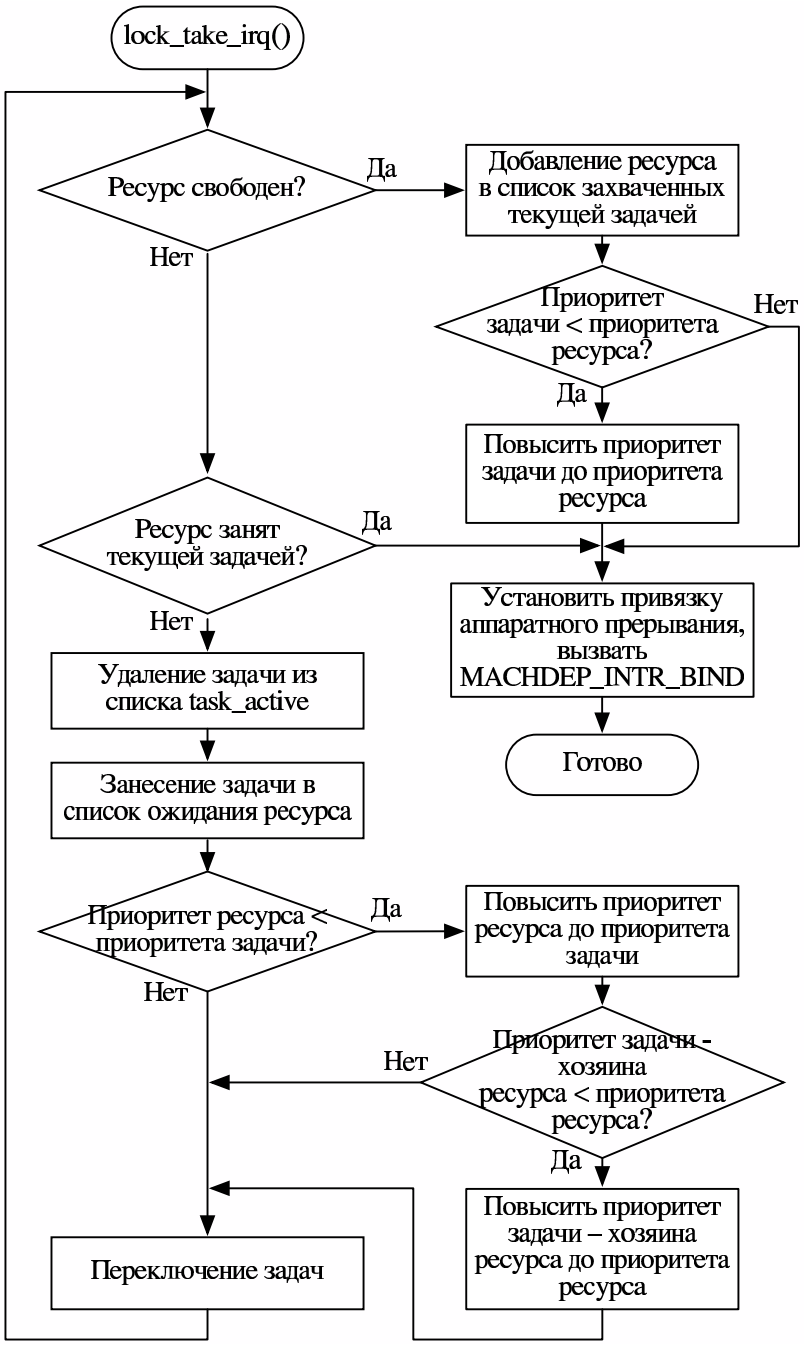
Параметры

`lock` Ресурс, который требуется захватить.

`irq` Номер аппаратного прерывания, который будет привязан к данному ресурсу.

`func` Указатель на функцию – быстрый обработчик прерывания (см. Раздел 2.4.3 [handler_t], страница 35). Необязательный параметр. Если нет необходимости использовать быстрый обработчик, следует указать параметр `func` равным `0`.

`arg` Аргумент, передаваемый быстрому обработчику прерывания (см. Раздел 2.4.3 [handler_t], страница 35).


lock_take_irq()
Ресурс свободен?
Да
Нет
Добавление ресурса в список захваченных текущей задачей
Приоритет задачи < приоритета ресурса?
Нет
Да
Повысить приоритет задачи до приоритета ресурса
Ресурс занят текущей задачей?
Да
Нет
Установить привязку аппаратного прерывания, вызвать MACHDEP_INTR_BIND
Готово
Удаление задачи из списка task_active
Занесение задачи в список ожидания ресурса
Приоритет ресурса < приоритета задачи?
Да
Нет
Повысить приоритет ресурса до приоритета задачи
Приоритет задачи - хозяина ресурса < приоритета ресурса?
Нет
Да
Повысить приоритет задачи – хозяина ресурса до приоритета ресурса
Переключение задач

Пример

2.4.2 Функция `lock_release_irq()`

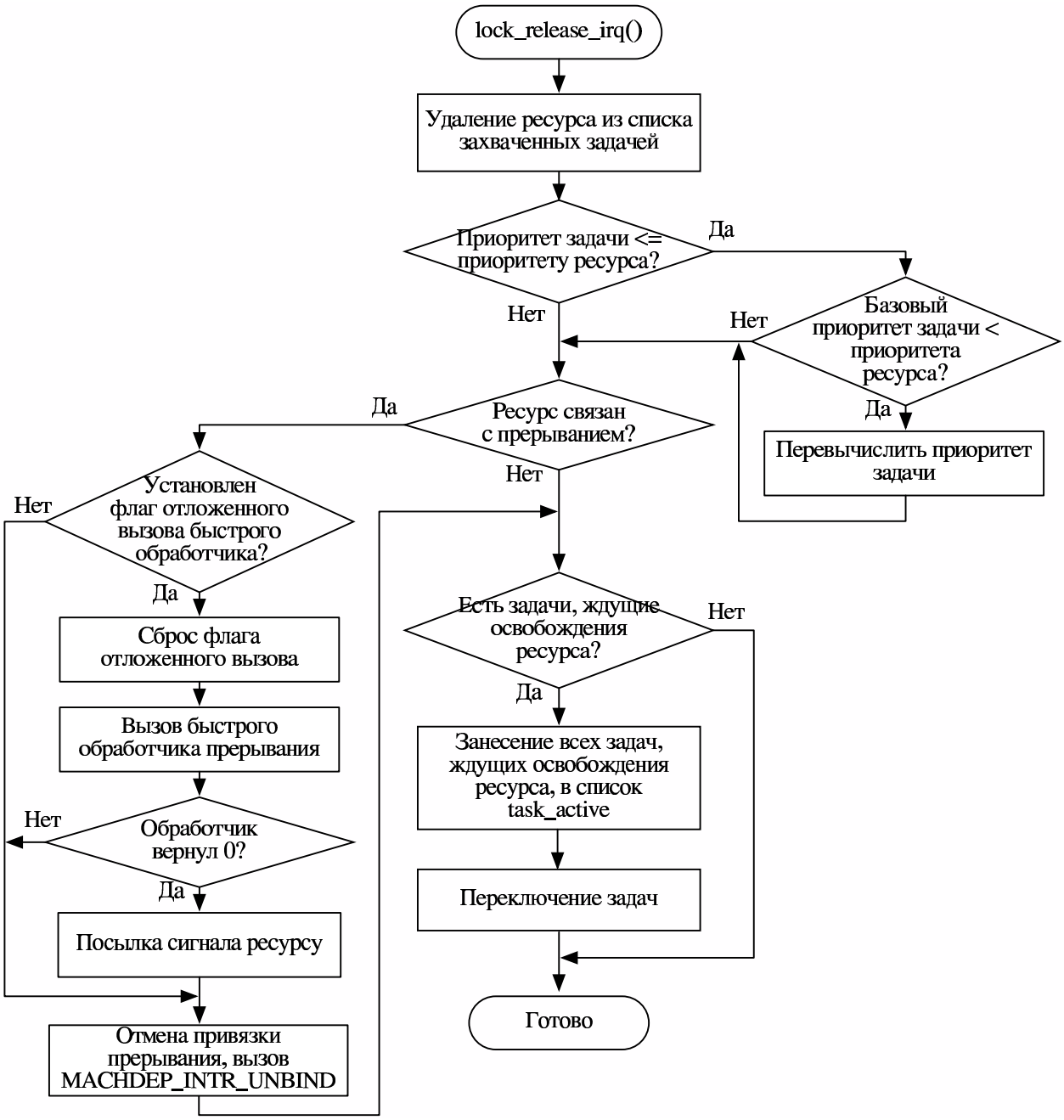
```
#include <kernel/uos.h>

void lock_release_irq (lock_t *lock);
```

Освобождение ресурса, отмена привязки аппаратного прерывания. Прерывание блокируется (запрещается). Может произойти переключение задач.

Следует заметить, что функция `lock_release()` не отменяет привязку и не блокирует прерывание.

Параметры

`lock` Ресурс, который требуется освободить.


lock_release_irq()
Удаление ресурса из списка захваченных задачей
Приоритет задачи <= приоритету ресурса?
Да
Нет
Базовый приоритет задачи < приоритета ресурса?
Нет
Да
Перевычислить приоритет задачи
Ресурс связан с прерыванием?
Да
Нет
Установлен флаг отложенного вызова быстрого обработчика?
Нет
Да
Сброс флага отложенного вызова
Вызов быстрого обработчика прерывания
Обработчик вернул 0?
Нет
Да
Посылка сигнала ресурсу
Отмена привязки прерывания, вызов MACHDEP_INTR_UNBIND
Есть задачи, ждущие освобождения ресурса?
Нет
Да
Занесение всех задач, ждущих освобождения ресурса, в список task_active
Переключение задач
Готово

Пример

2.4.3 Тип `handler_t`

```
#include <kernel/uos.h>

typedef int (*handler_t) (void *arg);
```

Тип – указатель на функцию быстрой обработки прерываний.

Стандартный метод обработки прерываний требует переключения задач на каждое прерывание. Для тех случаев, когда нужна более быстрая реакция на прерывания, применяется дополнительный механизм быстрой обработки.

Быстрый обработчик представляет собой функцию, которая регистрируется при захвате ресурса прерывания, и вызывается ядром при наступлении прерывания, обеспечивая максимально высокую скорость реакции. Обслужив прерывание, быстрый обработчик принимает решение и возвращает ядру признак: следует ли посылать основной задаче-обработчику сообщение о прерывании. Таким образом, работа по обработке разделяется между основной задачей прерывания и быстрым обработчиком: срочные действия выполняет быстрый обработчик, а “медленные” - основная задача.

Поскольку быстрый обработчик выполняется в обход механизма синхронизации задач, на него накладываются некоторые ограничения. Во избежание конфликта с другими задачами, он имеет право работать только с данными, защищенными ресурсом прерывания. Он не должен вызывать никакие функции ядра (`task_xxx()`, `lock_xxx()`) ни непосредственно, ни посредством вызова других модулей. В частности, он не может обращаться к модулю управления памятью (`mem_alloc()`и пр.). Подобные действия должны перекладываться на задачу-обработчик.

Параметры

`arg` Аргумент, передаваемый быстрому обработчику прерывания. Задается при захвате прерывания функцией `lock_take_irq()` (см. Раздел 2.4.1 [lock_take_irq], страница 31).

Пример

3 Системная библиотека

TODO

4 Модули

4.1 Драйвер таймера

```
#include <dev/timer.h>

typedef struct _timer_t {
    ...
} timer_t;

void timer_init (timer_t *t, int prio, unsigned long khz,
unsigned char msec_per_tick);
void timer_delay (timer_t *t, unsigned long msec);
unsigned long timer_milliseconds (timer_t *t);
unsigned short timer_days (timer_t *t);
unsigned char timer_passed (timer_t *t, unsigned long t0, unsigned short msec);
```

TODO

Для работы с таймером применяются следующие функции:

‘`timer_init`’	Инициализация таймера
‘`timer_delay`’	Задержка выполнения текущей задачи
‘`timer_milliseconds`’	Запрос времени в миллисекундах
‘`timer_days`’	Запрос времени в сутках
‘`timer_passed`’	Проверка временного события

4.2 Драйвер асинхронного порта

TODO

4.3 Драйвер неразрушаемой памяти

TODO

5 Практический подход

5.1 Установка текстов uOS

Информацию о последних версиях и инструкции по загрузке можно получить на домашней странице **uOS** (`http://www.vak.ru/uos/`).

Система разработки **uOS** проверена и работает в следующих операционных системах:

- Linux
- FreeBSD

После распаковки дистрибутива uOS Вы получите следующую структуру каталогов:

Каталог	**Содержимое каталога**
'arch'	Часть системы, зависящая от архитектуры целевого процессора
'demos'	Демонстрационные примеры
'doc'	Документация
'drivers'	Драйверы устройств
'kernel'	Ядро иперационной системы
'lib'	Библиотека полезных функций
'net'	Реализация TCP/IP
'sys'	Файлы заголовков (include)

(Описать процесс конфигурирования и компиляции. Параметры Makefile. Компиляция в DOS.)

5.2 Создание проекта

TODO

5.3 Удаленная отладка с GDB

TODO

5.4 Применение assert

TODO (описать параметр NDEBUG)

5.5 Оптимизация памяти

TODO

5.6 Пример реального проекта

5.7 Особенности архитектуры AVR

Распределение регистров AVR:

Регистр	**Описание**
R0	Временный регистр
R1	Всегда содержит 0
R2...R17, R28-R29	Сохраняемые регистры
R18...R27, R30-R31	Несохраняемые регистры

Передача параметров при вызове функции:

Вызов	**Параметр 1**	**Параметр 2**	**Параметр 3**	**Параметр 4**
f (p8, p8)	R25	R24	—	—
f (p8, p8, p8, p8)	R25	R24	R23	R22
f (p16, p16)	R25:24	R23:22	—	—
f (p16, p16, p16)	R25:24	R23:22	R21:20	—
f (p32, p32)	R25:24:23:22	R21:20:19:18	—	—
f (p32, p32, p32)	R25:24:23:22	R21:20:19:18	R17:16:15:14	—
f (p8, p16, p32)	R25	R24:23	R22:21:20:19	—

(старший байт слева, млдший справа)

Возвращаемое значение:

Размер	**Регистры**
p8	R25:24 (R25=0)
p16	R25:24
p32	R25:24:23:22

Параметры оператора `asm`:

Символ	**Описание**	**Диапазон**
a	Простые старшие регистры	R16...R23
b	Пары-указатели Y-Z	Y, Z
d	Старшие регистры	R16...R31
e	Пары-указатели X-Z	X, Y, Z
I	6-битная целая константа	0...63
J	6-битная отрицательная целая константа	-63...0
l	Младшие регистры	R0...R15
M	8-битная целая константа	0...255
O	Целая константа 8, 16, 24	8, 16, 24
r	Любой регистр	R0...R31
w	Специальные пары	R24, R26, R28, R30

При прерывании, состояние регистров сохраняется в стеке текущей задачи:

Стек	
—	
—	
PC младший байт	<– Значение SP до прерывания
PC старший байт	
R31	
SREG	— Бит I равен 1 (прерывания разрешены)
R30	
R29	
...	
R1	
R0	
—	<– Новое значение SP

5.8 Особенности архитектуры ARM

Распределение регистров ARM:

Регистр	**Применение**	**Описание**
R0...R3, R12	Volatile	Несохраняемые регистры
R4...R10	Non-volatile	Сохраняемые регистры
R11	Frame pointer	Указатель фрейма в стеке
R13 (SP)	Stack pointer	Указатель стека
R14 (LR)	Link register	Адрес возврата
R15 (PC)	Program counter	Счетчик команд

При вызове функции первые четыре параметра передаются в регистрах R0...R3. Остальные помещаются в стек в обратном порядке, последний параметр имеет больший адрес. Функция возвращает значение в регистре R0 или регистрах R0...R1 (в случае `long long` или `double`).

Параметры оператора `asm`:

Символ	**Режим ARM**	**Режим Thumb**
m	Операнд в памяти	Операнд в памяти
r	Операнд в регистре	Операнд в регистре
g	Операнд в памяти или в регистре	Операнд в памяти или в регистре
f	Регистр с плавающей точкой	Регистр с плавающей точкой
l	R0...R12, LR, SP, PC	R0...R7
h	—	R8...R12, LR, SP, PC
b	—	R0...R7, SP
k	—	SP

При прерывании, состояние регистров сохраняется в стеке текущей задачи:

Стек	
—	
—	<– Значение SP до прерывания
PC	
CPSR	
LR	
R12	
R11	
...	
R1	
R0	<– Новое значение SP
—	

6 Перенос на другие архитектуры

Определение структуры фрейма в стеке
Построение фрейма
Переключение задач
Обработчики прерываний
Запрет и восстановление прерываний
Разрешение прерываний
Привязка прерываний
Отсутствие активности
Останов
TODO

Приложение А История изменений

TODO

А.1 Изменения в версии 0.1

- TODO

А.2 Изменения в версии 0.2

- TODO

Приложение B Условия распространения

Операционная система uOS является свободным программным обеспечением. Она распространяется на условиях Стандартной общественной лицензии GNU (см. Приложение C [GPL], страница 44), с одним дополнением: при связывании файлов с файлами uOS получаемый в результате бинарный файл не обязан подчиняться требованиям лицензии GPL.

В других словах:

1. Вы можете применять uOS для разработки встраиваемых систем. На распространение таких систем (с бинарным кодом uOS внутри) не накладывается никаках ограничений. В частности, Вы не обязаны выдавать покупателю Вашей системы её исходные тексты. Данный пункт относится *только* к распространению системы в виде выполняемых кодов.
2. Если Вы хотите распространять модифицированную версию uOS или другую инструментальную систему, в состав которой входит uOS (либо ее существенная часть), в виде текстов или объектных файлов, Вы обязаны делать это на условиях лицензии GPL, т.е. с предоставлением полных исходных текстов.

Приложение C Стандартная общественная лицензия GNU

Версия 2, июнь 1991 г.
Автор перевода Елена Тяпкина (tiapkina@hotmail.com), март 2002 г.

This is an unofficial translation of the GNU General Public License into Russian. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL—only the original English text of the GNU GPL does that. However, we hope that this translation will help Russian speakers understand the GNU GPL better.

Настоящий перевод Стандартной Общественной Лицензии GNU на русский язык не является официальным. Он не публикуется Free Software Foundation и не устанавливает имеющих юридическую силу условий для распространения программного обеспечения, которое распространяется на условиях Стандартной Общественной Лицензии GNU. Условия, имеющие юридическую силу, закреплены исключительно в аутентичном тексте Стандартной Общественной Лицензии GNU на английском языке. Я надеюсь, что настоящий перевод поможет русскоязычным пользователям лучше понять содержание Стандартной Общественной Лицензии GNU.

Текст GNU GPL на английском языке вы можете прочитать здесь (http://www.gnu.org/copyleft/gpl.html).

Преамбула

Большинство лицензий на программное обеспечение лишает Вас права распространять и вносить изменения в это программное обеспечение. Стандартная Общественная Лицензия GNU, напротив, разработана с целью гарантировать Вам право совместно использовать и вносить изменения в свободное программное обеспечение, т.е. обеспечить свободный доступ к программному обеспечению для всех пользователей. Условия настоящей Стандартной Общественной Лицензии применяются к большей части программного обеспечения Free Software Foundation, а также к любому другому программному обеспечению по желанию его автора. (К некоторому программному обеспечению Free Software Foundation применяются условия Стандартной Общественной Лицензии GNU для Библиотек). Вы также можете применять Стандартную Общественную Лицензию к разработанному Вами программному обеспечению.

Говоря о свободном программном обеспечении, мы имеем в виду свободу, а не безвозмездность. Настоящая Стандартная Общественная Лицензия разработана с целью гарантировать Вам право распространять экземпляры свободного программного обеспечения (и при желании получать за это вознаграждение), право получать исходный текст программного обеспечения или иметь возможность его получить, право вносить изменения в программное обеспечение или использовать его части в новом свободном программном обеспечении, а также право знать, что вы имеете все вышеперечисленные права.

Чтобы защитить Ваши права, мы вводим ряд ограничений с тем, чтобы никто не имел возможности лишить Вас этих прав или обратиться к Вам с предложением отказаться от этих прав. Данные ограничения налагают на Вас определенные обязанности в случае, если вы распространяете экземпляры программного обеспечения или модифицируете программное обеспечение.

Например, если вы распространяете экземпляры такого программного обеспечения за плату или бесплатно, вы обязаны передать новым обладателям все права в том же объеме, в каком они принадлежат Вам. Вы обязаны обеспечить получение новыми обладателями программы ее исходного текста или возможность его получить. Вы также обязаны ознакомить их с условиями настоящей Лицензии.

Для защиты Ваших прав мы: (1) оставляем за собой авторские права на программное обеспечение и (2) предлагаем Вам использовать настоящую Лицензию, в соответствии с условиями которой вы вправе воспроизводить, распространять и/или модифицировать программное обеспечение.

Кроме того, для защиты как нашей репутации, так и репутации других авторов программного обеспечения, мы уведомляем всех пользователей, что на данное программное обеспечение никаких гарантий не предоставляется. Те, кто приобрел программное обеспечение, с внесенными в него третьими лицами изменениями, должны знать, что они получают не оригинал, в силу чего автор оригинала не несет ответственности за ошибки в работе программного обеспечения, допущенные третьими лицами при внесении изменений.

Наконец, программное обеспечение перестает быть свободным в случае, если лицо приобретает на него исключительные права. Недопустимо, чтобы лица, распространяющие свободное программное обеспечение, могли приобрести исключительные права на использование данного программного обеспечения и зарегистрировать их в Патентном ведомстве. Чтобы избежать этого, мы заявляем, что обладатель исключительных прав обязан предоставить любому лицу права на использование программного обеспечения либо не приобретать исключительных прав вообще.

Ниже изложены условия воспроизведения, распространения и модификации программного обеспечения.

Условия воспроизведения, распространения и модификации

0. Условия настоящей Лицензии применяются ко всем видам программного обеспечения или любому иному произведению, которое содержит указание правообладателя на то, что данное произведение может распространяться на условиях Стандартной Общественной Лицензии. Под термином “Программа” далее понимается любое подобное программное обеспечение или иное произведение. Под термином “произведение, производное от Программы” понимается Программа или любое иное производное произведение в соответствии с законодательством об авторском праве, т.е. произведение, включающее в себя Программу или ее часть, как с внесенными в ее текст изменениями, так и без них и/или переведенную на другой язык. (Здесь и далее, понятие “модификация” включает в себя понятие перевода в самом широком смысле). Каждый приобретатель экземпляра Программы именуется в дальнейшем “Лицензиат”.

   Действие настоящей Лицензии не распространяется на осуществление иных прав, кроме воспроизведения, распространения и модификации программного обеспечения. Не устанавливается ограничений на запуск Программы. Условия Лицензии распространяются на выходные данные из Программы только в том случае, если их содержание составляет произведение, производное от Программы (независимо от того, было ли такое произведение создано в результате запуска Программы). Это зависит от того, какие функции выполняет Программа.

1. Лицензиат вправе изготовлять и распространять экземпляры исходного текста Программы в том виде, в каком он его получил, без внесения в него изменений на любом носителе, при соблюдении следующих условий: на каждом экземпляре

помещен знак охраны авторского права и уведомление об отсутствии гарантий; оставлены без изменений все уведомления, относящиеся к настоящей Лицензии и отсутствию гарантий; вместе с экземпляром Программы приобретателю передается копия настоящей Лицензии.

Лицензиат вправе взимать плату за передачу экземпляра Программы, а также вправе за плату оказывать услуги по гарантийной поддержке Программы.

2. Лицензиат вправе модифицировать свой экземпляр или экземпляры Программы полностью или любую ее часть. Данные действия Лицензиата влекут за собой создание произведения, производного от Программы. Лицензиат вправе изготовлять и распространять экземпляры такого произведения, производного от Программы, или собственно экземпляры изменений в соответствии с пунктом 1 настоящей Лицензии при соблюдении следующих условий:
   a. файлы, измененные Лицензиатом, должны содержать хорошо заметную пометку, что они были изменены, а также дату внесения изменений;
   b. при распространении или публикации Лицензиатом любого произведения, которое содержит Программу или ее часть или является производным от Программы или от ее части, Лицензиат обязан передавать права на использование данного произведения третьим лицам на условиях настоящей Лицензии, при этом Лицензиат не вправе требовать уплаты каких-либо лицензионных платежей. Распространяемое произведение лицензируется как одно целое;
   c. если модифицированная Программа при запуске обычно читает команды в интерактивном режиме, Лицензиат обязан обеспечить вывод на экран дисплея или печатающее устройство сообщения, которое должно включать в себя:
      - знак охраны авторского права;
      - уведомление об отсутствии гарантий на Программу (или иное, если Лицензиат предоставляет гарантии);
      - указание на то, что пользователи вправе распространять экземпляры Программы в соответствии с условиями настоящей Лицензии, а также на то, каким образом пользователь может ознакомиться с текстом настоящей Лицензии. (Исключение: если оригинальная Программа является интерактивной, но не выводит в своем обычном режиме работы сообщение такого рода, то вывод подобного сообщения произведением, производным от Программы, в этом случае не обязателен).

   Вышеуказанные условия применяются к модифицированному произведению, производному от Программы, в целом. В случае если отдельные части данного произведения не являются производными от Программы, являются результатом творческой деятельности и могут быть использованы как самостоятельное произведение, Лицензиат вправе распространять отдельно такое произведение на иных лицензионных условиях. В случае если Лицензиат распространяет вышеуказанные части в составе произведения, производного от Программы, то условия настоящей Лицензии применяются к произведению в целом, при этом права, приобретаемые сублицензиатами на основании Лицензии, передаются им в отношении всего произведения, включая все его части, независимо от того, кто является их авторами.

   Целью настоящего пункта 2 не является заявление прав или оспаривание прав на произведение, созданное исключительно Лицензиатом. Целью настоящего пункта является обеспечение права контролировать распространение произведений, производных от Программы, и составных произведений, производных от Программы.

   Размещение произведения, которое не является производным от Программы, на одном устройстве для хранения информации или носителе вместе с Программой или

произведением, производным от Программы, не влечет за собой распространения условий настоящей Лицензии на такое произведение.

3. Лицензиат вправе воспроизводить и распространять экземпляры Программы или произведения, которое является производным от Программы, в соответствии с пунктом 2 настоящей Лицензии, в виде объектного кода или в исполняемой форме в соответствии с условиями п.п.1 и 2 настоящей Лицензии при соблюдении одного из перечисленных ниже условий:
   a. к экземпляру должен прилагаться соответствующий полный исходный текст в машиночитаемой форме, который должен распространяться в соответствии с условиями п.п. 1 и 2 настоящей Лицензии на носителе, обычно используемом для передачи программного обеспечения, либо
   b. к экземпляру должно прилагаться действительное в течение трех лет предложение в письменной форме к любому третьему лицу передать за плату, не превышающую стоимость осуществления собственно передачи, экземпляр соответствующего полного исходного текста в машиночитаемой форме в соответствии с условиями п.п. 1 и 2 настоящей Лицензии на носителе, обычно используемом для передачи программного обеспечения, либо
   c. к экземпляру должна прилагаться полученная Лицензиатом информация о предложении, в соответствии с которым можно получить соответствующий исходный текст. (Данное положение применяется исключительно в том случае, если Лицензиат осуществляет некоммерческое распространение программы, при этом программа была получена самим Лицензиатом в виде объектного кода или в исполняемой форме и сопровождалась предложением, соответствующим условиям пп.b п.3 настоящей Лицензии).

   Под исходным текстом произведения понимается такая форма произведения, которая наиболее удобна для внесения изменений. Под полным исходным текстом исполняемого произведения понимается исходный текст всех составляющих произведение модулей, а также всех файлов, связанных с описанием интерфейса, и сценариев, предназначенных для управления компиляцией и установкой исполняемого произведения. Однако, в качестве особого исключения, распространяемый исходный текст может не включать того, что обычно распространяется (в виде исходного текста или в бинарной форме) с основными компонентами (компилятор, ядро и т.д.) операционной системы, в которой работает исполняемое произведение, за исключением случаев, когда исполняемое произведение сопровождается таким компонентом.

   В случае если произведение в виде объектного кода или в исполняемой форме распространяется путем предоставления доступа для копирования его из определенного места, обеспечение равноценного доступа для копирования исходного текста из этого же места удовлетворяет требованиям распространения исходного текста, даже если третьи лица при этом не обязаны копировать исходный текст вместе с объектным кодом произведения.

4. Лицензиат вправе воспроизводить, модифицировать, распространять или передавать права на использование Программы только на условиях настоящей Лицензии. Любое воспроизведение, модификация, распространение или передача прав на иных условиях являются недействительными и автоматически ведут к расторжению настоящей Лицензии и прекращению всех прав Лицензиата, предоставленных ему настоящей Лицензией. При этом права третьих лиц, которым Лицензиат в соответствии с настоящей Лицензией передал экземпляры Программы или права на нее, сохраняются в силе при условии полного соблюдения ими настоящей Лицензии.
5. Лицензиат не обязан присоединяться к настоящей Лицензии, поскольку он ее не подписал. Однако только настоящая Лицензия предоставляет право распространять

или модифицировать Программу или произведение, производное от Программы. Подобные действия нарушают действующее законодательство, если они не осуществляются в соответствии с настоящей Лицензией. Если Лицензиат внес изменения или осуществил распространение экземпляров Программы или произведения, производного от Программы, Лицензиат тем самым подтвердил свое присоединение к настоящей Лицензии в целом, включая условия, определяющие порядок воспроизведения, распространения или модификации Программы или произведения, производного от Программы.

6. При распространении экземпляров Программы или произведения, производного от Программы, первоначальный лицензиар автоматически передает приобретателю такого экземпляра право воспроизводить, распространять и модифицировать Программу в соответствии с условиями настоящей Лицензии. Лицензиат не вправе ограничивать каким-либо способом осуществление приобретателями полученных ими прав. Лицензиат не несет ответственности за несоблюдение условий настоящей Лицензии третьими лицами.

7. Лицензиат не освобождается от исполнения обязательств в соответствии с настоящей Лицензией в случае, если в результате решения суда или заявления о нарушении исключительных прав или в связи с наступлением иных обстоятельств, не связанных непосредственно с нарушением исключительных прав, на Лицензиата на основании решения суда, договора или ином основании возложены обязательства, которые противоречат условиям настоящей Лицензии. В этом случае Лицензиат не вправе распространять экземпляры Программы, если он не может одновременно исполнить условия настоящей Лицензии и возложенные на него указанным выше способом обязательства. Например, если по условиям лицензионного соглашения сублицензиатам не может быть предоставлено право бесплатного распространения экземпляров Программы, которые они приобрели напрямую или через третьих лиц у Лицензиата, то в этом случае Лицензиат обязан отказаться от распространения экземпляров Программы.

   Если любое положение настоящего пункта при наступлении конкретных обстоятельств будет признано недействительным или неприменимым, настоящий пункт применяется за исключением такого положения. Настоящий пункт применяется в целом при прекращении вышеуказанных обстоятельств или их отсутствии.

   Целью данного пункта не является принуждение Лицензиата к нарушению патента или заявления на иные права собственности или к оспариванию действительности такого заявления. Единственной целью данного пункта является защита неприкосновенности системы распространения свободного программного обеспечения, которая обеспечивается за счет общественного лицензирования. Многие люди внесли свой щедрый вклад в создание большого количества программного обеспечения, которое распространяется через данную систему в надежде на ее длительное и последовательное применение. Лицензиат не вправе вынуждать автора распространять программное обеспечение через данную систему. Право выбора системы распространения программного обеспечения принадлежит исключительно его автору.

   Настоящий пункт 7 имеет целью четко определить те цели, которые преследуют все остальные положения настоящей Лицензии.

8. В том случае если распространение и/или использование Программы в отдельных государствах ограничено соглашениями в области патентных или авторских прав, первоначальный правообладатель, распространяющий Программу на условиях настоящей Лицензии, вправе ограничить территорию распространения Программы, указав только те государства, на территории которых допускается распространение Программы без ограничений, обусловленных такими соглашениями. В этом случае

такое указание в отношении территорий определенных государств признается одним из условий настоящей Лицензии.

9. Free Software Foundation может публиковать исправленные и/или новые версии настоящей Стандартной Общественной Лицензии. Такие версии могут быть дополнены различными нормами, регулирующими правоотношения, которые возникли после опубликования предыдущих версий, однако в них будут сохранены основные принципы, закрепленные в настоящей версии.

   Каждой версии присваивается свой собственный номер. Если указано, что Программа распространяется в соответствии с определенной версией, т.е. указан ее номер, или любой более поздней версией настоящей Лицензии, Лицензиат вправе присоединиться к любой из этих версий Лицензии, опубликованных Free Software Foundation. Если Программа не содержит такого указания на номер версии Лицензии Лицензиат вправе присоединиться к любой из версий Лицензии, опубликованных когда-либо Free Software Foundation.

10. В случае если Лицензиат намерен включить часть Программы в другое свободное программное обеспечение, которое распространяется на иных условиях, чем в настоящей Лицензии, ему следует испросить письменное разрешение на это у автора программного обеспечения. Разрешение в отношении программного обеспечения, права на которое принадлежат Free Software Foundation, следует испрашивать у Free Software Foundation. В некоторых случаях Free Software Foundation делает исключения. При принятии решения Free Software Foundation будет руководствоваться двумя целями: сохранение статуса свободного для любого произведения, производного от свободного программного обеспечения Free Software Foundation и обеспечение наиболее широкого совместного использования программного обеспечения.

ОТСУТСТВИЕ ГАРАНТИЙНЫХ ОБЯЗАТЕЛЬСТВ

11. ПОСКОЛЬКУ НАСТОЯЩАЯ ПРОГРАММА РАСПРОСТРАНЯЕТСЯ БЕСПЛАТНО, ГАРАНТИИ НА НЕЕ НЕ ПРЕДОСТАВЛЯЮТСЯ В ТОЙ СТЕПЕНИ, В КАКОЙ ЭТО ДОПУСКАЕТСЯ ПРИМЕНИМЫМ ПРАВОМ. НАСТОЯЩАЯ ПРОГРАММА ПОСТАВЛЯЕТСЯ НА УСЛОВИЯХ “КАК ЕСТЬ”. ЕСЛИ ИНОЕ НЕ УКАЗАНО В ПИСЬМЕННОЙ ФОРМЕ, АВТОР И/ИЛИ ИНОЙ ПРАВООБЛАДАТЕЛЬ НЕ ПРИНИМАЕТ НА СЕБЯ НИКАКИХ ГАРАНТИЙНЫХ ОБЯЗАТЕЛЬСТВ, КАК ЯВНО ВЫРАЖЕННЫХ, ТАК И ПОДРАЗУМЕВАЕМЫХ, В ОТНОШЕНИИ ПРОГРАММЫ, В ТОМ ЧИСЛЕ ПОДРАЗУМЕВАЕМУЮ ГАРАНТИЮ ТОВАРНОГО СОСТОЯНИЯ ПРИ ПРОДАЖЕ И ПРИГОДНОСТИ ДЛЯ ИСПОЛЬЗОВАНИЯ В КОНКРЕТНЫХ ЦЕЛЯХ, А ТАКЖЕ ЛЮБЫЕ ИНЫЕ ГАРАНТИИ. ВСЕ РИСКИ, СВЯЗАННЫЕ С КАЧЕСТВОМ И ПРОИЗВОДИТЕЛЬНОСТЬЮ ПРОГРАММЫ, НЕСЕТ ЛИЦЕНЗИАТ. В СЛУЧАЕ ЕСЛИ В ПРОГРАММЕ БУДУТ ОБНАРУЖЕНЫ НЕДОСТАТКИ, ВСЕ РАСХОДЫ, СВЯЗАННЫЕ С ТЕХНИЧЕСКИМ ОБСЛУЖИВАНИЕМ, РЕМОНТОМ ИЛИ ИСПРАВЛЕНИЕМ ПРОГРАММЫ, НЕСЕТ ЛИЦЕНЗИАТ.

12. ЕСЛИ ИНОЕ НЕ ПРЕДУСМОТРЕНО ПРИМЕНЯЕМЫМ ПРАВОМ ИЛИ НЕ СОГЛАСОВАНО СТОРОНАМИ В ДОГОВОРЕ В ПИСЬМЕННОЙ ФОРМЕ, АВТОР И/ИЛИ ИНОЙ ПРАВООБЛАДАТЕЛЬ, КОТОРЫЙ МОДИФИЦИРУЕТ И/ИЛИ РАСПРОСТРАНЯЕТ ПРОГРАММУ НА УСЛОВИЯХ НАСТОЯЩЕЙ ЛИЦЕНЗИИ, НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ПЕРЕД ЛИЦЕНЗИАТОМ ЗА УБЫТКИ, ВКЛЮЧАЯ ОБЩИЕ, РЕАЛЬНЫЕ, ПРЕДВИДИМЫЕ И КОСВЕННЫЕ УБЫТКИ (В ТОМ ЧИСЛЕ УТРАТУ ИЛИ ИСКАЖЕНИЕ ИНФОРМАЦИИ, УБЫТКИ, ПОНЕСЕННЫЕ ЛИЦЕНЗИАТОМ ИЛИ ТРЕТЬИМИ

ЛИЦАМИ, НЕВОЗМОЖНОСТЬ РАБОТЫ ПРОГРАММЫ С ЛЮБОЙ ДРУГОЙ ПРОГРАММОЙ И ИНЫЕ УБЫТКИ). АВТОР И/ИЛИ ИНОЙ ПРАВООБЛАДАТЕЛЬ В СООТВЕТСТВИИ С НАСТОЯЩИМ ПУНКТОМ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ДАЖЕ В ТОМ СЛУЧАЕ, ЕСЛИ ОНИ БЫЛИ ПРЕДУПРЕЖДЕНЫ О ВОЗМОЖНОСТИ ВОЗНИКНОВЕНИЯ ТАКИХ УБЫТКОВ.

Приложение D Лицензия GNU на свободную документацию

Версия 1.1, март 2000 г.
Автор перевода Елена Тяпкина (tiapkina@hotmail.com), август 2001 г.

This is an unofficial translation of the GNU Free Documentation License (GFDL) into Russian. It was not published by the Free Software Foundation, and does not legally state the distribution terms for works that uses the GFDL—only the original English text of the GFDL does that. However, we hope that this translation will help Russian speakers understand the GFDL better.

Настоящий перевод Лицензии GNU на Свободную Документацию (GFDL) на русский язык не является официальным. Он не публикуется Free Software Foundation и не устанавливает имеющих юридическую силу условий для распространения произведений, которые распространяются на условиях GFDL. Условия, имеющие юридическую силу, закреплены исключительно в аутентичном тексте GFDL на английском языке. Я надеюсь, что настоящий перевод поможет русскоязычным пользователям лучше понять содержание GFDL.

Текст GFDL на английском языке вы можете прочитать здесь (http://www.gnu.org/copyleft/fdl.html).

0. Преамбула

Цель настоящей Лицензии — сделать свободными справочник, руководство пользователя или иные документы в письменной форме, т.е. обеспечить каждому право свободно копировать и распространять как с изменениями, так и без изменений, за вознаграждение или бесплатно указанные документы. Настоящая Лицензия также позволяет авторам или издателям документа сохранить свою репутацию, не принимая на себя ответственность за изменения, сделанные третьими лицами.

Настоящая Лицензия относится к категории “copyleft”. Это означает, что все произведения, производные от документа, должны быть свободными в соответствии с концепцией “copyleft”. Настоящая Лицензия дополняет General Public License GNU, которая является лицензией “copyleft”, разработанной для свободного программного обеспечения.

Настоящая Лицензия разработана для применения ее к документации на свободное программное обеспечение, поскольку свободное программное обеспечение должно сопровождаться свободной документацией. Пользователь должен обладать теми же правами в отношении руководства пользователя, какими он обладает в отношении свободного программного обеспечения. При этом действие настоящей Лицензии не распространяется только на руководство пользователя. Настоящая Лицензия может применяться к любому текстовому произведению независимо от его темы или от того, издано ли данное произведение в виде печатной книги или нет. Настоящую Лицензию рекомендуется применять для произведений справочного или обучающего характера.

1. Сфера действия, термины и их определения

Условия настоящей Лицензии применяются к любому руководству пользователя или иному произведению, которое в соответствии с уведомлением, помещенным правообладателем, может распространяться на условиях настоящей Лицензии. Далее под

термином “Документ” понимается любое подобное руководство пользователя или произведение. Лицо, которому передаются права по настоящей Лицензии, в дальнейшем именуется “Лицензиат”.

“Модифицированная версия Документа” — любое произведение, содержащее Документ или его часть, скопированные как с изменениями, так и без них и/или переведенные на другой язык.

“Второстепенный раздел” — имеющее название приложение или предисловие к Документу, в котором отражено исключительно отношение издателей или авторов Документа к его содержанию в целом, либо к вопросам, связанным с содержанием Документа. Второстепенный раздел не может включать в себя то, что относится непосредственно к содержанию Документа. (Например, если часть Документа является учебником по математике, во Второстепенном разделе не может содержаться что-либо имеющее отношение непосредственно к математике). Во Второстепенных разделах могут быть затронуты вопросы истории того, что составляет содержание или что связано с содержанием Документа, а также правовые, коммерческие, философские, этические или политические взгляды относительно содержания Документа.

“Неизменяемые разделы” — определенные Второстепенные разделы, названия которых перечислены как Неизменяемые разделы в уведомлении Документа, определяющем лицензионные условия.

“Текст, помещаемый на обложке” — определенные краткие строки текста, которые перечислены в уведомлении Документа, определяющем лицензионные условия, как текст, помещаемый на первой и последней страницах обложки.

“Прозрачный” экземпляр Документа — экземпляр Документа в машиночитаемой форме, представленный в формате с общедоступной спецификацией при условии, что документ может просматриваться и редактироваться непосредственно с помощью общедоступных текстовых редакторов или общедоступных программ для векторной или растровой графики (в случае, если в документе содержатся изображения векторной или растровой графики). Указанный формат должен обеспечить ввод текста Документа в программы форматирования текста или автоматический перевод Документа в различные форматы, подходящие для ввода текста Документа в программы форматирования текста. Экземпляр Документа, представленный в ином формате, разметка которого затрудняет или препятствует внесению в Документ последующих изменений пользователями, не является Прозрачным. Такой экземпляр документа называется “Непрозрачным”.

Форматы, в которых может быть представлен Прозрачный экземпляр Документа, включают простой формат ASCII без разметки, формат ввода Texinfo, формат ввода LaTeX, SGML или XML с использованием общедоступного DTD, а также соответствующий стандартам простой формат HTML, предназначений для внесения модификаций человеком. “Непрозрачные” форматы включают в себя PostScript, PDF, форматы, которые можно прочитать и редактировать только с помощью текстовых редакторов, права на использование которых свободно не передаются, форматы SGML или XML, для которых DTD или инструменты для обработки не являются общедоступными, а также генерируемый машиной HTML, который вырабатывается некоторыми тесктовыми редакторами исключительно в целях вывода.

“Титульный лист” — для печатной книги собственно титульный лист, а также следующие за ним страницы, которые должны содержать сведения, помещаемые на титульном листе в соответствии с условиями настоящей Лицензии. Для произведений, формат которых не предполагает наличие титульного листа, под Титульным листом понимается текст, который помещен перед началом основного текста произведения, после его названия, напечатанного наиболее заметным шрифтом.

2. Копирование без внесения изменений

Лицензиат вправе воспроизводить и распространять экземпляры Документа на любом носителе за вознаграждение или безвозмездно при условии, что каждый экземпляр содержит текст настоящей Лицензии, знаки охраны авторских прав, а также уведомление, что экземпляр распространяется в соответствии с настоящей Лицензией, при этом Лицензиат не вправе предусматривать иные лицензионные условия дополнительно к тем, которые закреплены в настоящей Лицензии. Лицензиат не вправе использовать технические стредства для воспрепятствования или контроля за чтением или последующим изготовлением копий с экземпляров, распространяемых Лицензиатом. Лицензиат вправе получать вознаграждение за изготовление и распространение экземпляров Документа. При распространении большого количества экземпляров Документа Лицензиат обязан соблюдать условия пункта 3 настоящей Лицензии.

Лицензиат вправе сдавать экземпляры Документа в прокат на условиях, определенных в предыдущем абзаце, или осуществлять публичный показ экземпляров Документа.

3. Тиражирование

Если Лицензиат издает печатные экземпляры Документа в количестве свыше 100, и в соответствии с уведомлением Документа, определяющем лицензионные условия, Документ должен содержать Текст, помещаемый на обложке, Лицензиат обязан издавать экземпляры Документа в обложке с напечатанными на ней ясно и разборчиво соответствующими Текстами, помещаемыми на обложке: Тексты, помещаемые на первой странице обложки — на первой странице, Тексты, помещаемые на последней странице — соответственно на последней. Также на первой и последней странице обложки экземпляра Документа должно быть ясно и разборчиво указано, что Лицензиат является издателем данных экземпляров. На первой странице обложки должно быть указано полное название Документа без пропусков и сокращений, все слова в названии должны быть набраны шрифтом одинакового размера. Лицензиат вправе поместить прочие сведения на обложке экземпляра. Если при издании экземпляров Документа изменяются только сведения, помещенные на обложке экземпляра, за исключением названия Документа, и при этом соблюдаются требования настоящего пункта, такие действия приравниваются к копированию без внесения изменений.

Если объем текста, который должен быть помещен на обложке экземпляра, не позволяет напечатать его разборчиво, Лицензиат обязан поместить разумную часть текста непосредственно на обложке, а остальной текст на страницах Документа, следующих сразу за обложкой.

Если Лицензиат издает или распространяет Непрозрачные экземпляры Документа в количестве свыше 100, Лицезиат обязан к каждому такому экземпляру приложить Прозрачный экземпляр этого Документа в машиночитаемой форме или указать на каждом Непрозрачном экземпляре Документа адрес в компьютерной сети общего пользования, где содержится Прозрачный экземпляр без каких-либо добавленных материалов, полный текст которого каждый пользователь компьютерной сети общего пользования вправе бесплатно, не называя своего имени и не регистрируясь, записать в память компьютера с использованием общедоступных сетевых протоколов. Во втором случае Лицензиат обязан предпринять разумные шаги с тем, чтобы доступ к Прозрачному экземпляру Документа по указанному адресу сохранялся по крайней мере в течение одного года после последнего распространения Непрозрачного экземпляра Документа данного тиража, независимо от того, было ли распространение осуществлено Лицензиатом непосредственно или через агентов или розничных продавцов.

Прежде чем начать распространение большого количества экземпляров Документа Лицензиату заблаговременно следует связаться с авторами Документа, чтобы они имели

возможность предоставить Лицензиату обновленную версию Документа. Лицензиат не обязан выполнять данное условие.

4. Внесение изменений

Лицензиат вправе воспроизводить и распространять Модифицированные версии Документа в соответствии с условиями пунктов 2 и 3 настоящей Лицензии, при условии что Модифицированная версия Документа публикуется в соответствии с настоящей Лицензией. В частности, Лицензиат обязан передать каждому обладателю экземпляра Модифицированной версии Документа права на распространение и внесение изменений в данную Модифицированную версию Документа, аналогично правам на распространение и внесение изменений, которые передаются обладателю экземпляра Документа. При распространении Модифицированных версий Документа Лицензиат обязан:

A. поместить на Титульном листе и на обложке при ее наличии название Модифицированной версии, отличающееся от названия Документа и названий предыдущих версий. Названия предыдущих версий при их наличии должны быть указаны в Документе в разделе “История”. Лицензиат вправе использовать название предыдущей версии Документа с согласия издателя предыдущей версии;

B. указать на Титульном листе в качестве авторов тех лиц, которые являются авторами изменений в Модифицированной версии, а также не менее пяти основных авторов Документа либо всех авторов, если их не более пяти;

C. указать на Титульном листе наименование издателя Модифицированной версии, с указанием, что он является издателем данной Версии;

D. сохранить все знаки охраны авторского права Документа;

E. поместить соответствующий знак охраны авторского права на внесенные Лицензиатом изменения рядом с прочими знаками охраны авторского права;

F. поместить непосредственно после знаков охраны авторского права уведомление, в соответствии с которым каждому предоставляется право использовать Модифицированную Версию в соответствии с условиями настоящей Лицензии. Текст уведомления приводится в приложении к настоящей Лицензии;

G. сохранить в уведомлении, указанном в подпункте G, полный список Неизменяемых разделов и Текста, помещаемого на обложке, перечисленных в уведомлении Документа;

H. включить в Модифицированную версию текст настоящий Лицензии без каких-либо изменений;

I. сохранить в Модифицированной версии раздел “История”, включая его название, и дополнить его пунктом, в котором указать так же, как данные сведения указаны на Титульном листе, название, год публикации, наименования новых авторов и издателя Модифицированной версии. Если в Документе отсутствует раздел “История”, Лицензиат обязан создать в Модифицированной версии такой раздел, указать в нем название, год публикации, авторов и издателя Документа так же, как данные сведения указаны на Титульном листе Документа и дополнить этот раздел пунктом, содержание которого описано в предыдущем предложении;

J. сохранить в Модифицированной версии адрес в компьютерной сети, указанный в Документе, по которому каждый вправе осуществить доступ к Прозрачному экземпляру Документа, а также адрес в компьютерной сети, указанный в Документе, по которому можно получить доступ к предыдущим версиям Документа. Адреса, по которым находятся предыдущие версии Документа, можно поместить в раздел “История”. Лицензиат вправе не указывать адрес произведения в компьютерной сети, которое было опубликовано не менее чем за четыре года до публикации самого

Документа. Лицензиат вправе не указывать адрес определенной версии в компьютерной сети с разрешения первоначального издателя данной версии;

K. сохранить без изменений названия разделов “Благодарности” или “Посвящения”, а также содержание и стиль каждой благодарности и/или посвящения;

L. сохранить без изменений названия и содержание всех Неизменяемых разделов Документа. Нумерация данных разделов или иной способ их перечисления не включается в состав названий разделов;

M. удалить существующий раздел Документа под названием “Одобрения”. Такой раздел не может быть включен в Модифицированную версию;

N. не присваивать существующим разделам Модифицированной версии название “Одобрения” или такие названия, которые повторяют название любого из Неизменяемых разделов.

Если в Модифицированную версию включены новые предисловия или приложения, которые могут быть определены как Второстепенные разделы и которые не содержат текст, скопированный из Документа, Лицензиат вправе по своему выбору определить все или некоторые из этих разделов как Неизменяемые. Для этого следует добавить их названия в список Неизменяемых разделов в уведомлении в Модифицированной версии, определяющем лицензионные условия. Названия данных разделов должны отличаться от названий всех остальных разделов.

Лицензиат вправе дополнить Модифицированную версию новым разделом “Одобрения” при условии, что в него включены исключительно одобрения Модифицированной версии Лицензиата третьими сторонами, например оценки экспертов или указания, что текст Модифицированной версии был одобрен организацией в качестве официального определения стандарта.

Лицензиат вправе дополнительно поместить на обложке Модифицированной версии Текст, помещаемый на обложке, не превышающий пяти слов для первой страницы обложке и 25 слов для последней страницы обложки. К Тексту, помещаемому на обложке, каждым лицом непосредственно или от имени этого лица на основании соглашения с ним может быть добавлено только по одной строке на первой и на последней страницах обложки. Если на обложке Документа Лицензиатом от своего имени или от имени лица, в интересах которого действует Лицензиат, уже был помещен Текст, помещаемый на обложке, Лицензиат не вправе добавить другой Текст. В этом случае Лицензиат вправе заменить старый текст на новый с разрешения предыдущего издателя, который включил старый текст в издание.

По настоящей Лицензии автор(ы) и издатель(и) Документа не передают право использовать их имена и/или наименования в целях рекламы или заявления или предположения, что любая из Модифицированных Версий получила их одобрение.

5. Объединение документов

Лицензиат с соблюдением условий п.4 настоящей Лицензии вправе объединить Документ с другими документами, которые опубликованы на условиях настоящей Лицензии, при этом Лицензиат должен включить в произведение, возникшее в результате объединения, все Неизменяемые разделы из всех первоначальных документов без внесения в них изменений, а также указать их в качестве Неизменяемых разделов данного произведения в списке Неизменяемых разделов, который содержится в уведомлении, определяющем лицензионные условия для произведения.

Произведение, возникшее в результате объединения, должно содержать только один экземпляр настоящей Лицензии. Повторяющиеся в произведении одинаковые Неизменяемые разделы могут быть заменены единственной копией таких разделов. Если

произведение содержит несколько Неизменяемых Разделов с одним и тем же названием, но с разным содержанием, Лицензиат обязан сделать название каждого такого раздела уникальным путем добавления после названия в скобках уникального номера данного раздела или имени первоначального автора или издателя данного раздела, если автор или издатель известны Лицензиату. Лицензиат обязан соответственно изменить названия Неизменяемых разделов в списке Неизменяемых разделов в уведомлении, определяющем лицензионные условия для произведения, возникшего в результате объединения.

В произведении, возникшем в результате объединения, Лицензиат обязан объединить все разделы “История” из различных первоначальных Документов в один общий раздел “История”. Подобным образом Лицензиат обязан объединить все разделы с названием “Благодарности” и “Посвящения”. Лицензиат обязан исключить из произведения все разделы под названием “Одобрения”.

6. Сборники документов

Лицензиат вправе издать сборник, состоящий из Документа и других документов, публикуемых в соответствии с условиями настоящей Лицензии. В этом случае Лицензиат вправе заменить все экземпляры настоящей Лицензии в документах одним экземпляром, включенным в сборник, при условии, что остальной текст каждого документа включен в сборник с соблюдением условий по осуществлению копирования без внесения изменений.

Лицензиат вправе выделить какой-либо документ из сборника и издать его отдельно в соответствии с настоящей Лицензией, при условии, что Лицензиатом в данный документ включен текст настоящей Лицензии и им соблюдены условия Лицензии по осуществлению копирования без внесения изменений в отношении данного документа.

7. Подборка документа и самостоятельных произведений

Размещение Документа или произведений, производных от Документа, с другими самостоятельными документами или произведениями на одном устройстве для хранения информации или носителе не влечет за собой возникновения Модифицированной версии Документа, при условии, что Лицензиат не заявляет авторских прав на осуществленный им подбор или расположение документов при их размещении. Такое размещение называется “Подборкой”, при этом условия настоящей Лицензии не применяются к самостоятельным произведениям, размещенным вышеуказанным способом вместе с Документом, при условии, что они не являются произведениями, производными от Документа.

Если условия пункта 3 настоящей Лицензии относительно Текста, помещаемого на обложке, могут быть применены к экземплярам Документа в Подборке, то в этом случае Текст с обложки Документа может быть помещен на обложке только собственно Документа внутри подборки при условии, что Документ занимает менее четвертой части объема всей Подборки. Если Документ занимает более четвертой части объема Подборки, в этом случае Текст с обложки Документа должен быть помещен на обложке всей Подборки.

8. Перевод

Перевод является одним из способов модификации Документа, в силу чего Лицензиат вправе распространять экземпляры перевода Документа в соответствии с пунктом 4 настоящей Лицензии. Замена Неизменяемых разделов их переводами может быть осуществлена только с разрешения соответствующих правообладателей, однако Лицензиат вправе в дополнение к оригинальным версиям таких Неизменяемых разделов включить в текст экземпляра перевод всех или части таких Разделов. Лицензиат вправе

включить в текст экземпляра перевод настоящей Лицензии при условии, что в него включен также и оригинальный текст настоящей Лицензии на английском языке. В случае разногласий в толковании текста перевода и текста на английском языке предпочтение отдается тексту Лицензии на английском языке.

9. Расторжение лицензии

Лицензиат вправе воспроизводить, модифицировать, распространять или передавать права на использование Документа только на условиях настоящей Лицензии. Любое воспроизведение, модификация, распространение или передача прав на иных условиях являются недействительными и автоматически ведут к расторжению настоящей Лицензии и прекращению всех прав Лицензиата, предоставленных ему настоящей Лицензией. При этом права третьих лиц, которым Лицензиат в соответствии с настоящей Лицензией передал экземпляры Документа или права на него, сохраняются в силе при условии полного соблюдения ими настоящей Лицензии.

10. Пересмотр условий лицензии

Free Software Foundation может публиковать новые исправленные версии GFDL. Такие версии могут быть дополнены различными нормами, регулирующими правоотношения, которые возникли после опубликования предыдущих версий, однако в них будут сохранены основные принципы, закрепленные в настоящей версии (смотри `http://www.gnu.org/copyleft/`).

Каждой версии присваивается свой собственный номер. Если указано, что Документ распространяется в соответствии с определенной версией, т.е. указан ее номер, или любой более поздней версией настоящей Лицензии, Лицензиат вправе присоединиться к любой из этих версий Лицензии, опубликованных Free Software Foundation (при условии, что ни одна из версий не является проектом Лицензии). Если Документ не содержит такого указания на номер версии Лицензии Лицензиат вправе присоединиться к любой из версий Лицензии, опубликованных когда-либо Free Software Foundation (при условии, что ни одна из версий не является Проектом Лицензии).

Порядок применения условий настоящей Лицензии к Вашей документации

Чтобы применить условия настоящей Лицензии к созданному Вами документу, Вам следует включить в документ текст настоящей Лицензии, а также знак охраны авторского права и уведомление, определяющее лицензионные условия, сразу после титульного листа документа в соответствии с нижеприведенным образцом:

> © имя (наименование) автора или иного правообладателя, год первого опубликования документа
>
> Каждый имеет право воспроизводить, распространять и/или вносить изменения в настоящий Документ в соответствии с условиями GNU Free Documentation License, Версией 1.1 или любой более поздней версией, опубликованной Free Software Foundation.
>
> Данный Документ содержит следующие Неизменяемые разделы (указать названия Неизменяемых разделов); данный документ содержит следующий Текст, помещаемый на первой странице обложки (перечислить); данный документ содержит следующий Текст, помещаемый на последней странице обложки (перечислить).

Копия настоящей Лицензии включена в раздел под названием “GNU Free Documentation License”.

Если документ не содержит Неизменяемых разделов, укажите “Данный документ не содержит Неизменяемых разделов”. Если документ не содержит Текста, помещаемого на первой или последней страницах обложки, укажите “Данный документ не содержит Текста, помещаемого на первой странице обложки”, соответственно укажите для последней страницы обложки.

Если Ваш документ содержит имеющие существенное значение примеры программного кода, мы рекомендуем Вам выпустить их отдельно в соответствии с условиями одной из лицензий на свободное программное обеспечение, например GNU General Public License, чтобы их можно было использовать как свободное программное обеспечение.

Указатель