register\_chrdev, unregister\_chrdev - register a driver major number

**SYNOPSIS**

**#include <[linux/fs.h](file:///\\\\usr\\include\\linux\\fs.h)>**

**int register\_chrdev(unsigned int***major***, const char\****name***,** **struct file\_operations\****ops***);**  
**int unregister\_chrdev(unsigned int***major***, const char \****name***);** 

**DESCRIPTION**

The **register\_chrdev** function associates a character major number with set of driver entry points. The file\_operations structure contains pointers to functions that the the driver uses to implement the kernel interface to the driver.

связывает major номер символа с набором точек входа драйвера. Структура file\_operations содержит указатели на функции, которые драйвер использует для реализации интерфейса ядра с драйвером.

The paramter *major* is the character major number assigned to the device driver and to be mapped to the function table. The *name* parameter is a short name for the device and is displayed in the The */proc/devices* list. It also must exactly match the name passed to **unregister\_chrdev** function when releasing the functions.

A device driver module may register as many different major numbers as it supports, though this is not typically done.

The **unregister\_chrdev** function releases the major number, and is normally called in the module\_cleanup function to remove the driver from the kernel. 

*major* параметр - это *major* номер символа, назначаемый драйверу устройства и сопоставляемый с таблицей функций. Параметр name представляет собой краткое имя устройства и отображается в списке / proc / devices. Он также должен точно соответствовать имени, переданному функции unregister\_chrdev при освобождении функций.

Модуль драйвера устройства может зарегистрировать столько разных основных номеров, сколько он поддерживает, хотя обычно это не делается.

Функция unregister\_chrdev освобождает старший номер и обычно вызывается в функции module\_cleanup для удаления драйвера из ядра.

**RETURN VALUE**

On success, **register\_chrdev** returns 0 if *major* is a number other then 0, otherwise Linux will choose a major number and return the chosen value.

If there is an error, one of the following codes is returned instead:

-EINVAL

The specified number is not valid (> MAX\_CHRDEV)

-EBUSY

The major number is busy

The **unregister\_chrdev** function will return 0 if successful, or **-EINVAL** if the major number is not registered with the matching name. 

В случае успеха, register\_chrdev возвращает 0, если major - это число, отличное от 0, в противном случае Linux выберет старший номер и вернет выбранное значение.

В случае ошибки возвращается один из следующих кодов:

-EINVAL

Указанный номер недействителен (> MAX\_CHRDEV)

-EBUSY

Основной номер занят

Функция unregister\_chrdev вернет 0 в случае успеха или -EINVAL, если основной номер не зарегистрирован с соответствующим именем.

# NAME

register\_chrdev\_region - register a range of device numbers

# SYNOPSIS

**int register\_chrdev\_region(dev\_t***from***, unsigned***count***, const char \****name***);**

# ARGUMENTS

*dev\_t from*

the first in the desired range of device numbers; must include the major number.

первый в желаемом диапазоне номеров устройств; должен включать major номер.

*unsigned count*

the number of consecutive device numbers required

необходимое количество последовательных номеров устройств

*const char \* name*

the name of the device or driver.

# DESCRIPTION

Return value is zero on success, a negative error code on failure.

Возвращаемое значение равно нулю в случае успеха, отрицательный код ошибки при ошибке.

*/\*\**

*\* alloc\_chrdev\_region() - register a range of char device numbers* *зарегистрирует диапазон номеров символьного устройства*

*\* @dev: output parameter for first assigned number*

*\* @baseminor: first of the requested range of minor numbers*

*\* @count: the number of minor numbers required*

*\* @name: the name of the associated device or driver*

*\**

*\* Allocates a range of char device numbers. The major number will be*

*\* chosen dynamically, and returned (along with the first minor number)*

*\* in @dev. Returns zero or a negative error code.*

Выделяет диапазон номеров символьных устройств. Старший номер будет выбран динамически и возвращен (вместе с первым младшим номером) в @dev. Возвращает ноль или отрицательный код ошибки.

*\*/*

int [**alloc\_chrdev\_region**](https://elixir.bootlin.com/linux/latest/ident/alloc_chrdev_region)([**dev\_t**](https://elixir.bootlin.com/linux/latest/ident/dev_t) \*dev, unsigned baseminor, unsigned count,

const char \*name)

{

struct [**char\_device\_struct**](https://elixir.bootlin.com/linux/latest/ident/char_device_struct) \*[**cd**](https://elixir.bootlin.com/linux/latest/ident/cd);

[**cd**](https://elixir.bootlin.com/linux/latest/ident/cd) = [**\_\_register\_chrdev\_region**](https://elixir.bootlin.com/linux/latest/ident/__register_chrdev_region)(0, baseminor, count, name);

if ([**IS\_ERR**](https://elixir.bootlin.com/linux/latest/ident/IS_ERR)([**cd**](https://elixir.bootlin.com/linux/latest/ident/cd)))

return [**PTR\_ERR**](https://elixir.bootlin.com/linux/latest/ident/PTR_ERR)([**cd**](https://elixir.bootlin.com/linux/latest/ident/cd));

\*dev = [**MKDEV**](https://elixir.bootlin.com/linux/latest/ident/MKDEV)([**cd**](https://elixir.bootlin.com/linux/latest/ident/cd)->[**major**](https://elixir.bootlin.com/linux/latest/ident/major), [**cd**](https://elixir.bootlin.com/linux/latest/ident/cd)->baseminor);

return 0;

}