


Модуль os в Python, доступ к функциям ОС



lgmachinery.ru

Выгода до 600 000р. при покупке в лизинг!

Доставка по РФ • Запчасти • Сервис

Получить предложение

РЕКЛАМА

[Стандартная библиотека Python3.](#) / Модуль os в Python, доступ к функциям ОС

Интерфейсы операционной системы

[Модуль os](#) обеспечивает портативный способ использования функциональных возможностей, зависящих от операционной системы.

- если нужно просто прочитать или записать файл, то лучше воспользоваться [встроенной функцией open\(\)](#),
- для различных манипуляций с путями, то удобнее будет использовать [модуль os.path](#) или [pathlib](#),
- если необходимо прочитать все строки в файлах, указанных в командной строке, посмотрите на [модуль fileinput](#),
- для создания временных файлов и каталогов смотрите [модуль tempfile](#),
- для операций с файлами и каталогами (копирование, перемещение, создание, удаление) используйте [модуль shutil](#).

Примечания о доступности функций модуля os:

- Конструкция всех зависящих от Python модулей, встроенных в операционную систему, такова, что при наличии одинаковых функциональных возможностей он использует один и тот же интерфейс. Например, [функция os.stat\(path\)](#) возвращает статистическую информацию о пути в том же формате, который произошел от интерфейса POSIX.
- Расширения, характерные для конкретной операционной системы, также доступны через [модуль os](#), но их использование является угрозой переносимости ПО между системами.
- Все функции, принимающие пути или имена файлов, принимают как [байтовые строки](#), так и [строковые объекты](#) и приводят к объекту одного типа, если возвращается путь или имя файла.

Заметка. Все функции в этом модуле поднимают исключение OSError или их подклассы в случае недопустимых или недоступных имен и путей к файлам или других аргументов, которые имеют правильный тип, но не принимаются операционной системой.

os.error:

Исключение os.error это псевдоним для [встроенного исключения OSError](#).

os.name:

os.name это имя импортируемого модуля, зависящего от операционной системы. В настоящее время зарегистрированы следующие имена: 'posix', 'nt', 'java'.

Смотрите также [sys.platform](#) имеет более тонкую детализацию, [os.uname\(\)](#) дает системно-зависимую информацию о версии системы. [Модуль platform](#) содержит подробные проверки идентичности системы.

Имена файлов, аргументы командной строки и переменные окружения.

В Python имена файлов, аргументы командной строки и переменные окружения представлены с использованием [строкового типа](#). В некоторых системах, перед передачей их операционной системе необходимо декодирование строк в байты и обратно. Python использует кодировку файловой системы для выполнения этого преобразования (смотрите [sys.getfilesystemencoding\(\)](#)).

В некоторых системах преобразование с использованием кодировки файловой системы может завершиться ошибкой. В этом случае Python использует [обработчик ошибок кодирования](#) surrogateescape. Это означает, что некодировемые байты заменяются символом Unicode U + DCxx при декодировании, и они снова преобразуются в исходный байт при кодировании.

Код файловой системы должна гарантировать успешное декодирование всех байтов ниже 128. Если кодировка файловой системы не обеспечивает эту гарантию, функции API могут вызывать [ошибки UnicodeErrors](#).

Вверх

[Управление переменной средой окружения системы в Python](#)

Управление переменными средами `environment` из кода Python. Переменные среды обычно используются для значений конфигурации, таких как пути поиска, расположение файлов и т.д.

[Представление пути в файловой системе](#)

Функция `os.fsencode()` кодирует имя файла `filename` в виде пути. Функция `os.fsdecode()` декодирует имя файла `filename` в виде пути.

[Извлечение/установка `uid`, `gid` и `pid` процесса](#)

Функции для различных манипуляций с `uid`, `gid` и `pid` процесса. Чаще всего они используются авторами демонов или специальных системных программ, которым необходимо изменять уровень разрешений, а не запускаться от имени пользователя `root`.

[Наследование файловых дескрипторов](#)

Файловый дескриптор имеет "наследуемый" флаг, который указывает, может ли файловый дескриптор наследоваться дочерними процессами. Начиная с Python-3.4, файловые дескрипторы, созданные Python, по умолчанию не наследуются.

[Создание дескриптора файла, чтение, запись и его закрытие](#)

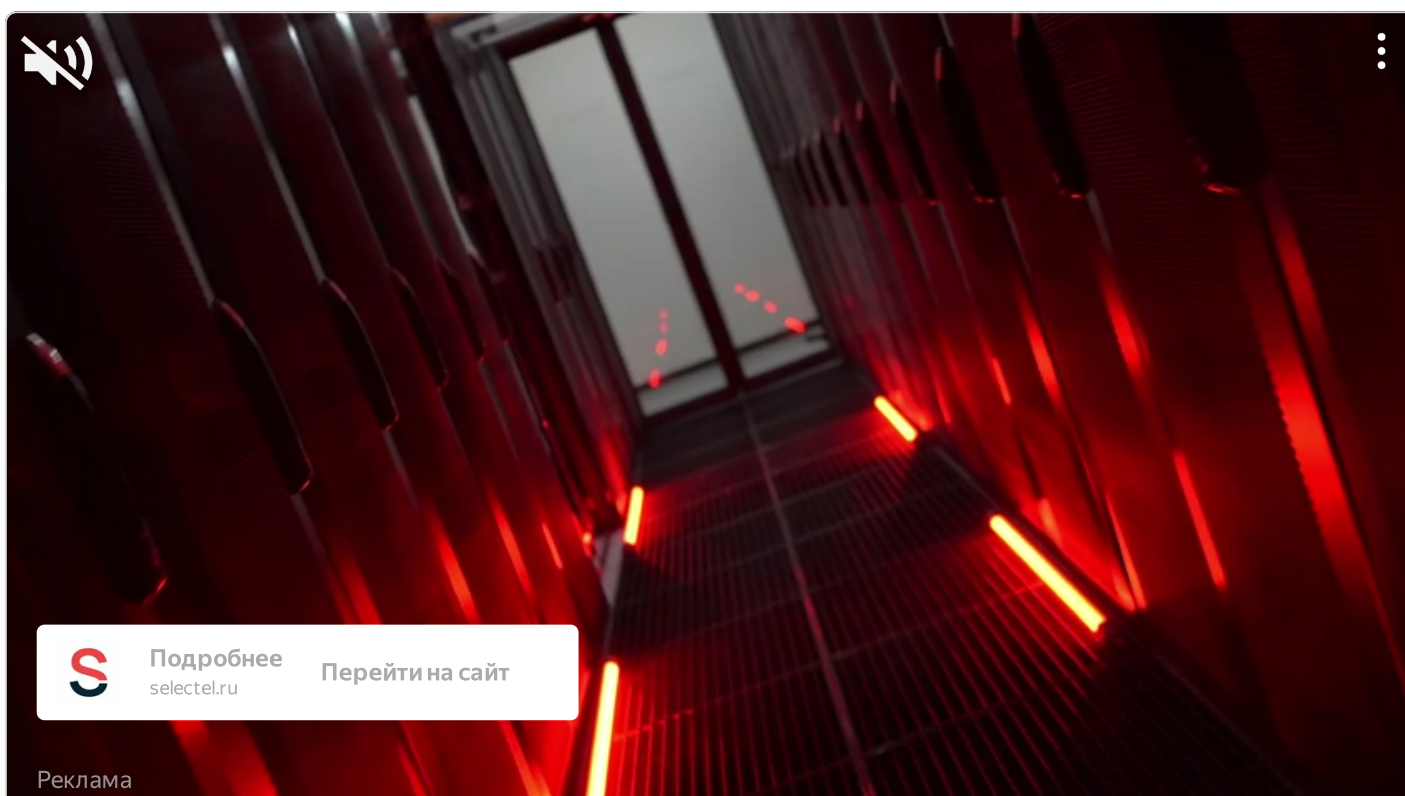
Создание файлового объекта средствами модуля `os`. Чтение, запись и закрытие файлового дескриптора, изменение прав доступа к нему. Получение статистики файлового дескриптора.

[Функция `listdir\(\)` модуля `os` в Python](#)

Функция `listdir()` модуля `os` возвращает список, содержащий имена файлов и директорий в каталоге, заданном путем `path`. Список приведен в произвольном порядке и не содержит специальных обозначений

[Функция `walk\(\)` модуля `os` в Python](#)

Функция `walk()` модуля `os` генерирует имена файлов в дереве каталогов, обходя дерево сверху вниз или снизу вверх. Для каждого каталога в дереве с корнем в вершине каталога `top`, включая саму вершину `top`, она выдает тройной кортеж `(dirpath, dirnames, filenames)`.



[Функция `scandir\(\)` модуля `os` в Python](#)

Функция `scandir()` модуля `os` возвращает итератор объектов `os.DirEntry`, соответствующих записям в каталоге, заданном путем `path`. Записи приводятся в произвольном порядке, а специальные символы `'.'` и `'..'` не включены.

[Объект `DirEntry\(\)` модуля `os` в Python](#)

Объект `DirEntry()` модуля `os` получается в результате работы функции `os.scandir()`. Методы объекта `DirEntry()` предоставляют пути к файлу и другие атрибуты файлов, расположенных в сканируемом каталоге.

[Функция `stat\(\)` модуля `os` в Python](#)

Функция `stat()` модуля `os` получает статистическую информацию файла или дескриптора файла. Выполняет эквивалент системного вызова `stat()`. Функция `stat()` может поддерживать указание дескриптора файла и не следовать символическим сс

[Вверх](#)

[Объект stat_result в Python, результаты выполнения os.stat\(\)](#)

Объект `stat_result` модуля `os` имеет атрибуты, которые примерно соответствуют членам структуры системного вызова `stat`.

[Функция lstat\(\) модуля os в Python](#)

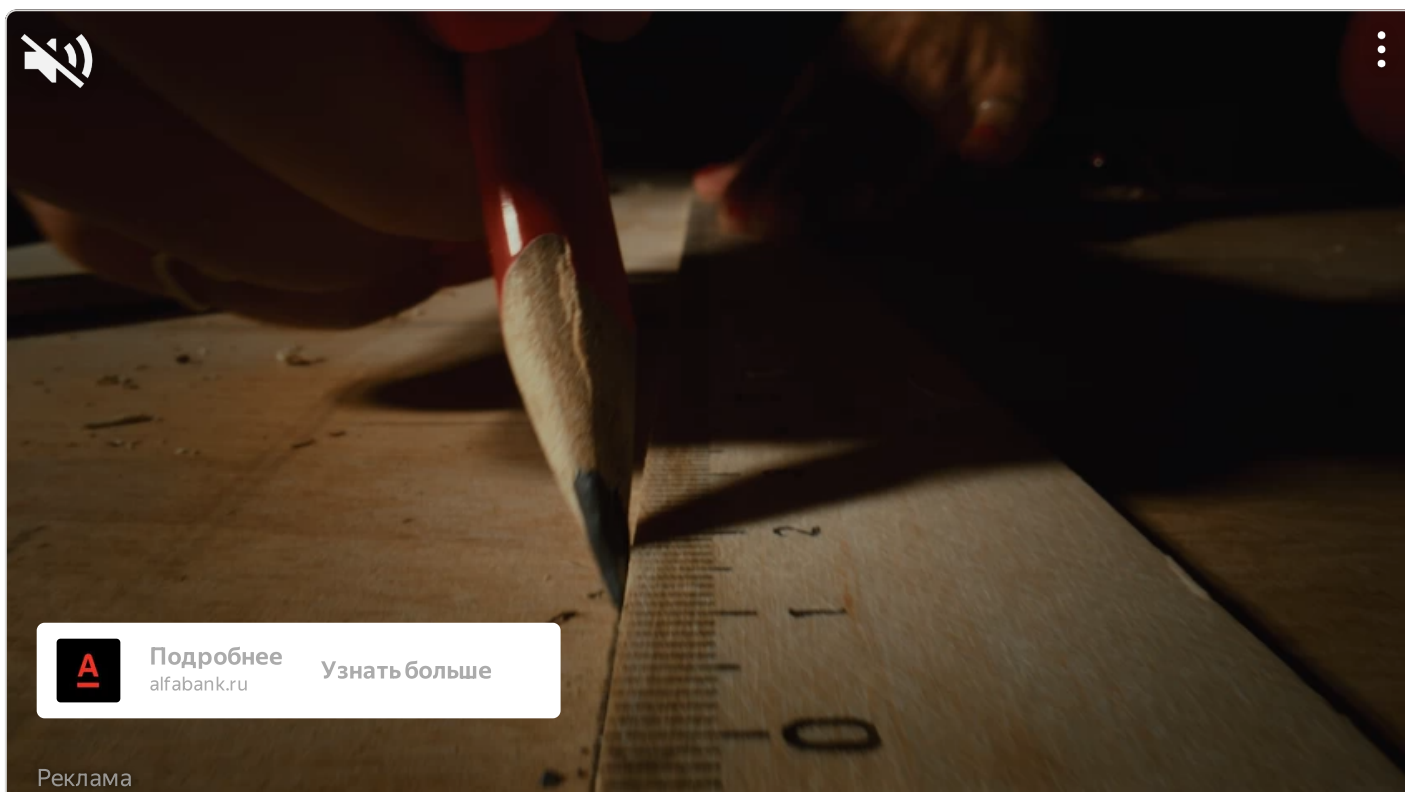
Функция `lstat()` модуля `os` выполняет эквивалент системного вызова `lstat()` для данного пути. Похож на `os.stat()`, но не следует по символическим ссылкам.

[Функция access\(\) модуля os в Python](#)

Функция `access()` проверяет доступ к пути `path` для реальных `uid`/`gid`. Эту процедуру можно использовать в среде `suid`/`sgid` для проверки, имеет ли вызывающий пользователь указанный доступ к пути `path`.

[Функция chdir\(\) модуля os в Python](#)

Функция `chdir()` модуля `os` изменяет текущий рабочий каталог.



[Функция chmod\(\) модуля os в Python](#)

Функция `chmod()` модуля `os` изменяет режим доступа к файлу или директории, указанного в `path`.

[Функция chown\(\) модуля os в Python](#)

Функция `chown()` изменяет владельца и идентификатор группы пути на числовые значения `uid` и `gid`.

[Функция chroot\(\) модуля os в Python](#)

Функция `chroot()` модуля `os` изменяет корневой каталог текущего процесса на путь файловой системы `path`.

[Функция getcwd\(\) модуля os в Python](#)

Функция `getcwd()` вернет строку, а функция `getcwdb()` вернет строку байтов представляющую текущий рабочий каталог.

[Функция link\(\) модуля os в Python](#)

Функция `link()` модуля `os` создает жесткую ссылку, указывающую на `src` с именем `dst`.

[Функция mkdir\(\) модуля os в Python](#)

Функция `mkdir()` модуля `os` создает каталог с именем `path` с режимом доступа к нему `mode`. Режим `mode` устанавливается последними 3 цифрами восьмеричного представления режима.

[Функция makedirs\(\) модуля os в Python](#)

Функция `makedirs()` модуля `os` рекурсивно создает все промежуточные каталоги, если они не существуют. Функция работает подобно `os.mkdir()`, но создает все каталоги промежуточного уровня.

[Функция symlink\(\) модуля os в Python](#)

Функция `symlink()` модуля `os` создает символическую ссылку, указывающую на `src` с именем `dst`.

[Функция readlink\(\) модуля os в Python](#)

[Вверх](#)

Функция `readlink()` модуля `os` вернет строку, представляющую путь, на который указывает символическая ссылка.

[Функция `remove\(\)` модуля `os` в Python](#)

Функция `remove()` модуля `os` удаляет путь `path` к файлу. Если путь является каталогом, возникает исключение `IsADirectoryError`. Функция `os.remove()` семантически идентична `os.unlink()`.

[Функция `removedirs\(\)` модуля `os` в Python](#)

Функция `removedirs()` модуля `os` удаляет каталоги рекурсивно. Если конечный каталог успешно удален, `os.removedirs()` пытается последовательно удалить каждый родительский каталог, указанный в пути, до появления ошибки. Появления ошибки обычно означает, что родительский каталог не пуст.

[Функция `rename\(\)` модуля `os` в Python](#)

Функция `rename()` модуля `os` переименовывает файл или каталог с именем `src` в `dst`. Если имя `dst` существует, операция, в ряде случаев, завершится с подклассом исключения `OSError`.

[Функция `renames\(\)` модуля `os` в Python](#)

Функция `renames()` модуля `os` рекурсивно переименовывает пустые директории или переименовывает конечный файл.

[Функция `replace\(\)` модуля `os` в Python](#)

Функция `replace()` модуля `os` переименовывает файл или пустой каталог с исходным именем `src` в `dst`.

[Функция `rmdir\(\)` модуля `os` в Python](#)

Функция `rmdir()` модуля `os` удаляет путь к каталогу `path`. Если директория `path` не существует или не является пустым каталогом, то возникает исключение.

[Функция `strerror\(\)` модуля `os` в Python](#)

Функция `strerror()` модуля `os` возвращает сообщение об ошибке, соответствующее коду ошибки, которая появляется при сбое в коде приложения.

[Функция `supports_dir_fd` модуля `os` в Python](#)

Функция `supports_dir_fd` модуля `os` возвращает объект `set`, указывающий, какие функции в модуле `os` принимают дескриптор открытого файла для своего параметра `dir_fd`.

[Функция `supports_effective_ids` модуля `os` в Python](#)

Функция `supports_effective_ids` модуля `os` возвращает множество `set`, указывающее, разрешает ли функция `os.access()` указывать `True` для своего параметра `effective_ids` на локальной платформе.

[Функция `supports_fd` модуля `os` в Python](#)

Функция `supports_fd` модуля `os` возвращает объект `set`, указывающий, какие функции в модуле `os` позволяют указывать параметр пути в качестве дескриптора открытого файла на локальной платформе.

[Функция `supports_follow_symlinks` модуля `os` в Python](#)

Функция `os.supports_follow_symlinks()` возвращает множество `set`, которое указывает, какие функции в модуле `os` принимают `False` для их параметра `follow_symlinks` на локальной платформе.

[Функция `truncate\(\)` модуля `os` в Python](#)

Функция `truncate()` модуля `os` обрезает файл, соответствующий пути `path`, так, чтобы он имел длину не более `length` байтов.

[Функция `utime\(\)` модуля `os` в Python](#)

Функция `utime()` модуля `os` устанавливает/изменяет время доступа к файлу и время изменения файла, указанного в `path`.

[Манипулирование списком контроля доступа ACL в Linux](#)

Примером использования расширенных атрибутов ОС Linux является реализация списков контроля доступа POSIX ACL.

[Функция `abort\(\)` модуля `os` в Python](#)

Функция `abort()` модуля `os` генерирует сигнала `SIGABRT` для текущего процесса.

[Функция `exec*\(\)` модуля `os` в Python](#)

[Вверх](#)

Все эти функции выполняют новую программу, заменяя текущий процесс. Они ничего не возвращают. В Unix новый исполняемый файл загружается в текущий процесс и будет иметь тот же идентификатор процесса, что и вызывающая программа. Ошибки будут сообщаться как исключения `OSError`.

[Функция `popen\(\)` модуля os в Python](#)

Функция `popen()` модуля `os` откроет канал для чтения или записи стандартного ввода-вывода запущенной команды `cmd`.

[Функция `system\(\)` модуля os в Python](#)

Функция `system()` модуля `os` выполняет команду `command` в оболочке (subshell). Это реализуется путем вызова стандартной функции языка Си `system()` и имеет те же ограничения.

[Функция `exit\(\)` модуля os в Python](#)

Функция `_exit()` модуля `os` осуществляет выход из процесса со статусом `n`, без вызова обработчиков очистки, сброса буферов `stdio` и т. д.

[Функция `fork\(\)` модуля os в Python](#)

Функция `fork()` модуля `os` форкает дочерний процесс. Возвращает `0` в дочернем элементе и идентификатор дочернего процесса в родительском элементе.

[Функция `kill\(\)` модуля os в Python](#)

Функция `kill()` модуля `os` посылает сигнал `sig` на процессу `pid`.

[Функция `spawn\(\)` модуля os в Python](#)

Функции `os.spawn*()` запускают программу, расположенную по указанному пути в файловой системе в новом процессе.

[Функция `umask\(\)` модуля os в Python](#)

Функция `umask()` модуля `os` устанавливает текущий `umask` пользователя в числовое значение `mask` и возвращает предыдущий `umask`.

[Функция `uname\(\)` модуля os в Python](#)

Функция `os.uname()` возвращает информацию, идентифицирующую текущую операционную систему. Возвращаемое значение - это объект с пятью атрибутами

[Функция `wait\(\)` модуля os в Python](#)

Функция `os.wait()` возвращает идентификатор процесса и код завершения, упакованный в 16-битовое значение. Младший байт представляет номер сигнала, прекратившего выполнение процесса, а старший - код состояния, возвращенный процессом по его завершении.

[Функция `waitpid\(\)` модуля os в Python](#)

Функция `waitpid()` модуля `os` в Unix: ждет завершения дочернего процесса, заданного идентификатором процесса `pid`, и возвращает кортеж, содержащий его идентификатор процесса и индикацию состояния выхода, закодированную как для `os.wait()`.

[Определение состояния процесса](#)

Следующие функции принимают в качестве параметра `status` код состояния процесса, возвращаемый `os.system()`, `os.wait()` или `os.waitpid()`. Они могут быть использованы для определения диспозиции процесса.

[Константы для поддержки операций с путями](#)

Значения используемые для поддержки операций с путями.

[Генератор случайных байтов на основе модуля os в Python](#)

Генерация случайных байтов операционной системой.

[Функция `startfile\(\)` модуля os в Python](#)

Функция `startfile()` модуля `os` запускает файл в Windows с помощью связанного с ним приложения на основе расширения.

[Функция `times\(\)` модуля os в Python](#)

Функция `times()` модуля `os` возвращает текущее время глобального процесса.

[Функции `getloadavg\(\)` и `cpu_count\(\)` модуля os в Python](#)

[Вверх](#)

В этом разделе представлены функции определения количества ядер сервера и уровня загрузки системы (load average).

[Функция waitstatus to exitcode\(\) модуля os в Python](#)

Функция waitstatus_to_exitcode() модуля os преобразует статус ожидания в код выхода.

Содержание раздела:

- [ОБЗОРНАЯ СТРАНИЦА РАЗДЕЛА](#)
- [Управление переменной средой окружения системы](#)
- [Представление пути в файловой системе](#)
- [Извлечение/установка uid, gid и pid процесса](#)
- [Наследование файловых дескрипторов](#)
- [Создание дескриптора файла, чтение, запись и его закрытие](#)
- [Функция listdir\(\) модуля os](#)
- [Функция walk\(\) модуля os](#)
- [Функция scandir\(\) модуля os](#)
- [Объект DirEntry\(\) модуля os](#)
- [Функция stat\(\) модуля os](#)
- [Объект stat result, результаты выполнения os.stat\(\)](#)
- [Функция lstat\(\) модуля os](#)
- [Функция access\(\) модуля os](#)
- [Функция chdir\(\) модуля os](#)
- [Функция chmod\(\) модуля os](#)
- [Функция chown\(\) модуля os](#)
- [Функция chroot\(\) модуля os](#)
- [Функция getcwd\(\) модуля os](#)
- [Функция link\(\) модуля os](#)
- [Функция mkdir\(\) модуля os](#)
- [Функция makedirs\(\) модуля os](#)
- [Функция symlink\(\) модуля os](#)
- [Функция readlink\(\) модуля os](#)
- [Функция remove\(\) модуля os](#)
- [Функция removedirs\(\) модуля os](#)
- [Функция rename\(\) модуля os](#)
- [Функция renames\(\) модуля os](#)
- [Функция replace\(\) модуля os](#)
- [Функция rmdir\(\) модуля os](#)
- [Функция strerror\(\) модуля os](#)
- [Функция supports_dir_fd модуля os](#)
- [Функция supports_effective_ids модуля os](#)
- [Функция supports_fd модуля os](#)
- [Функция supports_follow_symlinks модуля os](#)
- [Функция truncate\(\) модуля os](#)
- [Функция utime\(\) модуля os](#)
- [Манипулирование списком контроля доступа ACL в Linux](#)
- [Функция abort\(\) модуля os](#)
- [Функция exec*\(\) модуля os](#)
- [Функция popen\(\) модуля os](#)
- [Функция system\(\) модуля os](#)
- [Функция exit\(\) модуля os](#)
- [Функция fork\(\) модуля os](#)
- [Функция kill\(\) модуля os](#)
- [Функция spawn\(\) модуля os](#)
- [Функция umask\(\) модуля os](#)
- [Функция uname\(\) модуля os](#)

Вверх

- [Функция wait\(\) модуля os](#)
- [Функция waitpid\(\) модуля os](#)
- [Определение состояния процесса](#)
- [Константы для поддержки операций с путями](#)
- [Генератор случайных байтов на основе модуля os](#)
- [Функция startfile\(\) модуля os](#)
- [Функция times\(\) модуля os](#)
- [Функции getloadavg\(\) и cpu count\(\) модуля os](#)
- [Функция waitstatus to exitcode\(\) модуля os](#)

ХОЧУ ПОМОЧЬ
ПРОЕКТУ

