

[КАК СТАТЬ АВТОРОМ](#)

Ничто не стареет так быстро, как будущее. И вот как его увидеть сейчас



Vass 27 марта 2009 в 17:18

Разбор опций командной строки в UNIX-подобных системах

Программирование*

Tutorial

Введение

Одной из важных задач любой программы, будь она консольной или графической является интерпретация *аргументов* командной строки. Формально аргументами называются все слова в командной строке(в том числе и имя самой команды) разбитые *разделителем* (как правило, это пробел и табуляция), кавычки же позволяют включать разделители в аргументы.

Аргументы можно подразделить на *опции* и *операнды*. Опции изменяют поведение программы или предоставляют ей дополнительную информацию. У опции могут быть свои аргументы, которые являются информацией необходимой только для этой опции.

POSIX

Стандарт POSIX описывает ожидаемое поведение программ в UNIX-подобных системах. При написании программы никто вас не заставит строго следовать стандарту, однако, это является хорошей идеей, так как это облегчит жизнь вашим пользователям. Вот основные правила касающиеся аргументов команды:

- в имени программы должно быть не менее 2 и не более 9 символов;

- имена программ должны быть написаны только строчными символами и цифрами;
- имя опции должно быть простым буквенно-цифровым символом. Опции с множеством цифр запрещены;
- все опции должны начинаться с символа "-";
- для опций без аргументов должна быть реализована возможность объединения опций (например **foo -a -b -c** и **foo -abc**);
- аргумент опции должен отделяться от нее пробелом;
- аргумент опции не может быть необязательным;
- если опции требуется множество значений аргумента, они должны передаваться в виде строки, разделенные запятыми или разделителем;
- опции должны идти перед операндами;
- аргумент "--" указывает на окончание всех опций;
- порядок опций не должен играть роли, кроме случаев когда опции взаимоисключающие, тогда побеждает последняя;
- порядок аргументов может иметь значение;
- программы читающие или записывающие именованные файлы, должны трактовать единственный аргумент "-" как стандартный ввод или стандартный вывод соответственно.

Длинные опции

В GNU программах также используются длинные опции, поведение которых не описано в POSIX, длинные опции начинаются с "--" Для этих опций в GNU также реализованы следующие соглашения:

- каждая короткая опция должна иметь свой вариант длинной опции;

- длинную опцию можно сократить до кратчайшей строки, обеспечивающей ее уникальность;
- Аргументы длинной опции отделяются либо разделителем, либо знаком "=".

Откуда берутся параметры в программе

как известно функция `main()` в C определяется так:

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

Здесь присутствует два параметра: **argc** определяет количество аргументов в командной строке, а **argv** хранит массив указателей на эти аргументы.

Следует отметить, что **argv[0]** – всегда имя команды, а **argv[argc] == NULL**, эти два факта могут оказаться полезными при разработке.

Разбор опций

В 80-х годах группа поддержки Unix заметила, что каждая программа Unix использует собственные методы разбора опций. Что послужило толчком к разработке функции **getopt()**, чтобы упростить написание кода, придерживающегося стандартных соглашений.

Функция GNU **getopt_long()**, является совместимой с **getopt()**, а также упрощает разбор длинных опций.

getopt

Объявление:

```
#include <unistd.h>

int getopt(int argc, char *argv[], const char *optstring);

extern char *optarg;
```

```
extern int optind, opterr, optopt;
```

Аргументы **argc** и **argv** передаются непосредственно от функции **main()**, а **optstring** является строкой символов опций. Если за какой либо буквой в строке следует двоеточие, значит эта опция принимает аргумент.

Для использования **getopt()** ее вызывают повторно в цикле, до тех пор, пока она не вернет **-1**. Каждый раз обнаружив действительный символ опции, функция возвращает этот символ. Если опция принимает аргумент, то указатель на него помещается в переменную **optarg**.

переменная **optind** хранит текущий индекс в **argv**. Когда переменная **opterr** не равна нулю (по умолчанию 0), **getopt()** сама выводит сообщения в случае недействительной опции или отсутствия аргумента. Если же **opterr** равен нулю, то в случае возникновения ошибки **getopt()** возвращает "?" или ":" в зависимости от того найдена недействительная опция или пропущен обязательный аргумент опции, в переменной **optopt** будет находиться обнаруженный недействительный символ. Следует заметить, что стандартная функция **getopt()** останавливается сразу как только найдет первый аргумент начинающийся не с символа "-", GNU вариант функции просматривает в поисках опций, всю командную строку. Поведение GNU функции можно изменить (но это выходит за рамки статьи).

пример программы с использованием getopt()

Любезно предоставил @ iv_s

```
1. #include <unistd.h>
2. #include <stdlib.h>
3. #include <stdio.h>
4.
5. int main(int argc, char **argv) {
```

```
6.     if(argc == 1) { // если запускаем без аргументов, выводим справку
7.         printf(«getopt test\n»);
8.         printf(«usage:\n»);
9.         printf(" opts -a n -b m -o s\n");
10.        printf(«example:\n»);
```

[Все потоки](#)[Разработка](#)[Администрирование](#)[Дизайн](#)[Менеджмент](#)[Маркетинг](#)[Научпоп](#)

```
13.        return 0;
14.    }
15.    char *opts = «a:b:o:»; // доступные опции, каждая принимает аргумент
16.    int a, b; // тут храним числа
17.    char op; // а тут оператор
18.    int opt; // каждая следующая опция попадает сюда
19.    while((opt = getopt(argc, argv, opts)) != -1) { // вызываем getopt пока она не
    // вернет -1
20.        switch(opt) {
21.            case 'a': // если опция -a, преобразуем строку с аргументом в число
22.                a = atoi(optarg);
23.                break;
24.            case 'b': // тоже для -b
25.                b = atoi(optarg);
26.                break;
27.            case 'o': // в op сохраняем оператор
```

```
28.         op = optarg[0];
29.         break;
30.     }
31. }
```

◆ +66 👁 27K 📖 93 ➡

```
33.     case '+': // если оператор + складываем, и т.д.
34.         printf("%d + %d = %d\n", a, b, a + b);
35.         break;
36.     case '-':
37.         printf("%d - %d = %d\n", a, b, a - b);
38.         break;
39.     case '*':
40.         printf("%d * %d = %d\n", a, b, a * b);
41.         break;
42.     case '/':
43.         printf("%d / %d = %d\n", a, b, a / b);
44.         break;
45. }
46. return 0;
47. }
```

* This source code was highlighted with Source Code Highlighter.

getopt_long()

Объявление:

```
#include <getopt.h>
```

```
int getopt_long(int argc, char *argv[], const char *optstring, const struct  
option *longopts, int *longindex);
```

первые три аргумента те же, что и в **getopt()**, **longopts** является указателем на массив длинных опций, **longindex** указывает на переменную, в которую помещается индекс обнаруженной длинной опции в **longopts**, если в это нет необходимости может быть **NULL**.

Структура **option** определена следующим образом:

```
struct option  
{  
    const char *name;  
    int has_arg;  
    int *flag;  
    int val;  
}
```

name – имя опции без предшествующих черточек;

has_arg – как понятно из названия, переменная описывает имеет ли длинная опция аргумент, может принимать три значения:

- 0 – не принимает аргумент;
- 1 – обязательный аргумент;
- 2 – необязательный аргумент.

flag – если этот указатель равен **NULL**, то **getopt_long()** возвращает значение поля **val**, иначе она возвращает 0, а переменная на которую указывает **flag** заполняется значением **val**; **val** – обычно содержит некоторую символьную константу, если длинная опция соответствует короткой, то эта константа должна быть такой же как и та что появляется в аргументе **optstring**.

Важно заметить, что последний элемент массива **longopts**, должен быть заполнен нулями.

Пример программы с использованием getopt_long()

Любезно предоставил @ shuffle

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <getopt.h>
5.
6. void usage(char *name)
7. {
8.     printf(«usage: %s\n \t-h this message\n \t-c [config file]\n \t--
       help this message\n \t--config=config_file\n», name);
```



```
9. return;
10. }
11.
12. int main (int argc, char *argv[])
13. {
14.     int c;
15.     while (1){
16.         static struct option long_opt[] = {
17.             {«help», 0, 0, 'h'},
18.             {«config», 1, 0, 'c'},
19.             {0,0,0,0}
20.         };
21.         int optIdx;
22.
23.         if((c = getopt_long(argc, argv, «c:h», long_opt, &optIdx)) == -1)
24.             break;
25.         switch( c ){
26.             case 'h':
27.                 usage(argv[0]);
28.                 return(-1);
29.
30.             case 'c':
```

```
31.         printf(«option 'c' selected, filename: %s\n», optarg);
32.         return(1);
33.
34.     default:
35.         usage(argv[0]);
36.         return(-1);
37.     }
38. }
39.
40. return(0);
41. }
```

* This source code was highlighted with Source Code Highlighter.

Заключение

В статье рассмотрены основы использования функций разбора аргументов командной строки в UNIX-подобных системах. Этот материал может быть более обширен, но любой заинтересовавшийся человек в состоянии узнать и изучить все тонкости самостоятельно.

Статья подготовлена по материалам книги Арнольда Роббинса «Linux программирование в примерах» ISBN 5-9579-0059-1

Теги: `unix`, `getopt`, `getopt_long`, разбор опций, арнольд роббинс

Хабы: Программирование

Редакторский дайджест



Присылаем лучшие статьи раз в месяц

Электронная почта



71

Карма



Рейтинг

Vasiliy Sorokin @Vass

C++ Developer

Реклама

 Комментарии 34

ПОХОЖИЕ ПУБЛИКАЦИИ





18 апреля в 22:44

Города, инверсии и логистика: разбор задач для QA-инженеров

 +22  1.7K  14  0



3 апреля в 17:00

Краткая история Dell UNIX

 +37  7.5K  30  4 +4

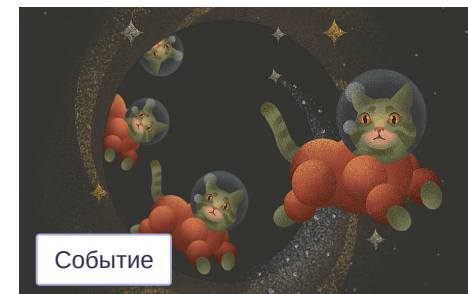
14 сентября 2016 в 03:19

Обзор пакетов Node.js для разбора опций командной строки

 +21  19K  77  34 +34

МИНУТОЧКУ ВНИМАНИЯ

Разместить



Хотите рассказать о себе в наших социальных сетях?

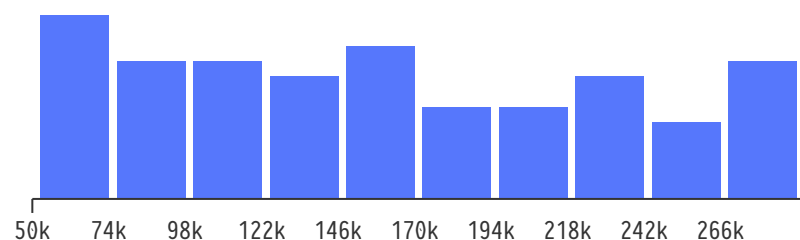
Как студенту погнаться за двумя зайцами и не отбросить копыта

Конкурс технических статей
Технотекст 2021

СРЕДНЯЯ ЗАРПЛАТА В IT

156 251 ₽/мес.

– средняя зарплата во всех IT-специализациях по данным из 9 082 анкет, за 1-ое пол. 2022 года. Проверьте «в рынке» ли ваша зарплата или нет!



Проверить свою зарплату

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 20:15

Римские и средневековые доспехи. Что лучше?

♦ +76 👁 9.6K 📖 39 💬 20 +20

вчера в 14:03

Чтобы найти хороших разработчиков, заставьте их читать чужой код

♦ +61 👁 13K 📖 80 💬 40 +40

вчера в 17:03

Как изменилась стандартная библиотека Python за последние годы

♦ +47 👁 5.5K 📖 52 💬 7 +7

вчера в 16:00

Используем клиентский процессор по максимуму. Часть 1: Rust + WebAssembly

♦ +40 👁 2.6K 📖 46 💬 2 +2

вчера в 20:06

О бедном Арсаним замолвите слово

♦ +34 👁 4.1K 📖 22 💬 6 +6

Реклама



ru.hexlet.io

Профессия Java-разработчик на Хекслете за 10 месяцев

Вводные курсы бесплатно

Курсы Java-разработки с нуля.

Под руководством опытных менторов. Онлайн обучение.

[Узнать больше](#)

ЧИТАЮТ СЕЙЧАС

NVIDIA открыла исходный код видеодрайверов для Linux

 3.6K  6 **+6**

В России начали строить фабрику для выпуска 28-нм чипов. Но все не так просто

 84K  615 **+615**

Римские и средневековые доспехи. Что лучше?

 9.6K  21 **+21**

Директор Apple по машинному обучению Ян Гудфеллоу уволился из-за невозможности теперь всегда работать удаленно

 26K  34 **+34**

Google возвращает цифровой кошелек Google Wallet

4.3K 3 +3

Реклама

Хабр Карьера

Ваша зарплата выше
или ниже рынка?
Узнайте
в калькуляторе

→

Перейти на Хабр Карьеру

Ваш аккаунт

- Войти
- Регистрация

Разделы

- Публикации
- Новости
- Хабы
- Компании
- Авторы
- Песочница

Информация

- Устройство сайта
- Для авторов
- Для компаний
- Документы
- Соглашение
- Конфиденциальность

Услуги

- Реклама
- Тарифы
- Контент
- Семинары
- Мегапроекты



Настройка языка

0 сайте

Техническая поддержка

Вернуться на старую версию

© 2006–2022, Habr