

Pobieranie argumentów wiersza polecenia

Lista argumentów

- Lista argumentów zawiera cały wiersz poleceń, łącznie z nazwą programu i wszystkimi dostarczonymi argumentami.

Przykłady:

```
$ cat plik
$ ls -l /tmp
```

- Standard ANSI definiuje dwa parametry, które można przekazywać funkcji `main()`: `argc` i `argv`. Pozwalają one przekazywać programom C/C++ argumenty z wiersza wywołania:

nazwa programu

```
test plik1 plik2 plik3
```

argumenty, które chcemy przekazać do programu

- Aby mieć dostęp do argumentów przekazywanych w wierszu wywołania program, nagłówek funkcji `main()` musi mieć postać:

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

- Parametr `argc` zawiera liczbę argumentów (ang. *argument count*) przekazywanych do programu. Jego wartość wynosi co najmniej 1, ponieważ nazwa programu jest również traktowana jako argument.
 - Parametr `argv` (ang. *argument value*) to wskaźnik do tablicy wskaźników odsyłających do napisów będących argumentami z wiersza wywołania.
- Przykład: programu drukuje argumenty przekazane w wierszu wywołania

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Wartosc argc to: %d\n", argc);
    printf("Nazwa programu to: %s\n", argv[0]);
    printf("Wartosci argumentow wiersza polecenia:\n");
    { int i;
      for (i = 1; i < argc; i++)
          printf("arg %i to %s\n", i, argv[i]);
    }
    return 0;
}
```

- *Wywołanie programu:*

```
$ ./test 5 plik1 plik2
Wartosc argc to: 4
Nazwa programu to: ./test
Wartosci argumentow wiersza polecenia:
arg 1 to 5
arg 2 to plik1
arg 3 to plik2
```

- Można również wykorzystać fakt, że tablica z argumentami jest zakończona wskaźnikiem pustym:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Wartosc argc to: %d\n", argc);
    printf("Nazwa programu to: %s\n",argv[0]);
    printf("Wartosci argumentow wiersza polecenia:\n");
    while (*++argv != NULL)
        printf("Argument: %s\n", *argv);
    return 0;
}
```

Opcje w wierszu polecenia

- Oczekiwane przez program argumenty można podzielić na dwie grupy: opcje oraz pozostałe argumenty.

Krótkie opcje

- Przykłady:

```
tar -cvf archiwum.tar katalog-z-plikami
lub
tar -c -v -f archiwum.tar katalog-z-plikami

ps -fu nowak      lub      ps -f -u nowak
```

- Obsługa opcji

```
#include <unistd.h>

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

extern int optind;          /* wartość początkowa 1 */
extern char *optarg;
extern optopt;
extern opterr;             /* wartość początkowa 1 */
```

- Każde wywołanie funkcji `getopt()` obsługuje jedną opcję. Funkcja zwraca kod rozpoznanej opcji. Po zakończeniu opcji zwraca `-1`. W przypadku nierozpoznanej opcji zwraca znak zapytania (`?`). Jeśli argument `optstring` zaczyna się od dwukropka (`:`), funkcja zwraca `:`, jeśli po opcji wymagającej argumentu nie podano `go`.
- Argumenty:
 - `argc` – ile argumentów do analizy
 - `argv` – tablica z argumentami do analizy
 - `optstring` – jakie opcje (z ewentualnymi argumentami) są obsługiwane; jeśli opcja wymaga argumentu, po nazwie opcji należy umieścić dwukropek (`:`); kolejność umieszczania opcji nie ma znaczenia
- Zmienne związane z `getopt`:
 - `optint` – wskazuje następny element w tablicy argumentów wiersza polecenia; wartość początkowa wynosi 1; po zakończeniu przetwarzania opcji wskazuje pierwszy argument do dalszego przetwarzania
 - `optarg` – wskazuje argument związany z opcją, o ile opcja go wymaga
 - `optopt` – w przypadku nieznanego opcji `getopt` zwraca `'?'`, `optopt` zawiera kod tej opcji; jeśli pominięto argument w opcji wymagającej argumentu, `getopt` zwraca `':'`, `optopt` zawiera kod tej opcji (tak dzieje się wtedy, jeśli `optstring` zaczyna się od `:`)
 - `opterr` – jeśli 1 (prawda), `getopt` w przypadku nierozpoznanej opcji automatycznie wyświetla komunikat o błędzie na wyjściu `stderr`; jeśli 0 (fałsz) komunikat nie jest wyświetlany

Przykład:

Załóżmy, że program można wywołać następująco: prog -cvf dane.tar dane_kat

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    int opcja;
    int kod_powrotu=0;
    char optstring[] = ":cvf:"; /* przetwarzane opcje */

    while ( (opcja = getopt(argc,argv,optstring)) != -1 )
        switch ( opcja ) {
            case 'c' :
                fprintf(stderr,"przetwarzam -c\n");
                break;
            case 'v' :
                fprintf(stderr,"przetwarzam -v\n");
                break;
            case 'f' :
                fprintf(stderr,"przetwarzam -f '%s'\n",optarg);
                break;
            case ':': /* brakuje argumentu w opcji */
                fprintf(stderr,"opcja -%c wymaga argumentu\n",optopt);
                kod_powrotu = 1; /* Wystapil blad */
                break;
            case '?':
            default :
                fprintf(stderr, "opcja -%c nie znana - ignoruje\n", optopt);
                kod_powrotu = 1; /* Wystapil blad */
                break;
        }

    printf("Pozostale argumenty:\n");
    for ( ; optind < argc; ++optind )
        printf("argv[%d] = '%s'\n",optind,argv[optind]);

    return kod_powrotu;
}
```

Opcje długie

Opcje długie rozpoczynają się od znaków `--`. Przykład:

```
ps --help
ps --user nowak
```

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

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

- Przykład obsługi długich opcji: Mitchell, Oldham, Samuel, *Linux. Programowanie dla zaawansowanych*, str. 19.

```
/* *****
 * Code listing from "Advanced Linux Programming," by CodeSourcery LLC *
 * Copyright (C) 2001 by New Riders Publishing *
 * See COPYRIGHT for license information. *
 * ***** */

#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>

/* The name of this program. */
const char* program_name;

/* Prints usage information for this program to STREAM (typically
   stdout or stderr), and exit the program with EXIT_CODE. Does not
   return. */

void print_usage (FILE* stream, int exit_code)
{
    fprintf (stream, "Usage: %s options [ inputfile ... ]\n", program_name);
    fprintf (stream,
            "  -h --help           Display this usage information.\n"
            "  -o --output filename Write output to file.\n"
            "  -v --verbose        Print verbose messages.\n");
    exit (exit_code);
}
```

```
/* Main program entry point.  ARGV contains number of argument list
   elements; ARGV is an array of pointers to them.  */

int main (int argc, char* argv[]) {
    int next_option;

    /* A string listing valid short options letters.  */
    const char* const short_options = "ho:v";
    /* An array describing valid long options.  */
    const struct option long_options[] = {
        { "help",      0, NULL, 'h' },
        { "output",    1, NULL, 'o' },
        { "verbose",   0, NULL, 'v' },
        { NULL,        0, NULL, 0   } /* Required at end of array.  */
    };

    /* The name of the file to receive program output, or NULL for
       standard output.  */
    const char* output_filename = NULL;
    /* Whether to display verbose messages.  */
    int verbose = 0;

    /* Remember the name of the program, to incorporate in messages.
       The name is stored in argv[0].  */
    program_name = argv[0];

    do {
        next_option = getopt_long (argc, argv, short_options,
                                   long_options, NULL);

        switch (next_option)
        {
            case 'h': /* -h or --help */
                /* User has requested usage information.  Print it to standard
                   output, and exit with exit code zero (normal termination).  */
                print_usage (stdout, 0);

            case 'o': /* -o or --output */
                /* This option takes an argument, the name of the output file.  */
                output_filename = optarg;
                break;

            case 'v': /* -v or --verbose */
                verbose = 1;
                break;

            case '?': /* The user specified an invalid option.  */
                /* Print usage information to standard error, and exit with exit
                   code one (indicating abnormal termination).  */
                print_usage (stderr, 1);

            case -1: /* Done with options.  */
                break;

            default: /* Something else: unexpected.  */
                abort ();
        }
    }
}
```

```
while (next_option != -1);

/* Done with options.  OPTIND points to first non-option argument.
   For demonstration purposes, print them if the verbose option was
   specified.  */
if (verbose) {
    int i;
    for (i = optind; i < argc; ++i)
        printf ("Argument: %s\n", argv[i]);
}

/* The main program goes here.  */

return 0;
}
```

- **Przykład użycia:**

```
$ ./program --help
Usage: ./program options [ inputfile ... ]
-h --help           Display this usage information.
-o --output filename Write output to file.
-v --verbose        Print verbose messages.
```