

# PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO

• WWW.PORTUGAL-A-PROGRAMAR.PT

EDIÇÃO #43 · DEZEMBRO 2013

ISSN 1647-0710



A PROGRAMAR

**LISTAS** DUPLAMENTE  
LIGADAS

**EXPANDE** O TEU MERCADO  
GLOBALIZANDO A TUA APP!

**INTRODUÇÃO** AO WEB2PY  
PARTE 2

**A FERRAMENTA** GENGETOPT

**IMPLEMENTAÇÃO** DE ÁRVORES  
DE VANTAGEM

ANÁLISES

**TÉCNOLOGIAS** DE PROGRAMAÇÃO  
DE JOGOS

COLUMNAS

OS PERIGOS DAS  
ESTRUTURAS MUTÁVEIS **C#**

COMUNIDADES

LEITOR DE QR CODE  
PARA WINDOWS PHONE **NETPONTO**

NO CODE

É A “LÍNGUA”  
DO SÉCULO XXI **PROGRAMAR**

## A ferramenta getopt

Os parâmetros de linha de comando constituem um poderoso mecanismo de interação, especialmente nas aplicações de modo consola/terminal que funcionam em modo de texto. De facto, através dos parâmetros da linha de comando, torna-se possível especificar múltiplos comportamentos e funcionalidades para uma mesma aplicação de modo consola/terminal, estando cada comportamento associado a um dado conjunto de parâmetros passado pela linha de comando. Por exemplo, o bem conhecido comando **ls**, cuja função primária no UNIX é a listar nomes de ficheiros e diretórios, disponibiliza na versão GNU mais de 50 parâmetros da linha de comando, desde o **-a** (mostra todos os ficheiros) ao **-x** (mostra a listagem por linhas). Curiosamente, existe ainda a opção **-X** (maiúscula) que lista os ficheiros por ordem alfabética das respetivas extensões.

Este artigo mostra como se usa e quais as mais-valias da ferramenta **getopt** para o tratamento semi-automatizado das opções passadas pela linha de comando em programas escritos em linguagem C que se destinam a ambientes linux.

### Acesso aos parâmetros da linha de comando na linguagem C

Na linguagem C, os parâmetros da linha de comando estão acessíveis através de dois parâmetros disponíveis na função **main**: o vetor de strings **argv** e o inteiro **argc**. O vetor **argv** contém uma string para cada conjunto de caracteres passados pela linha de comando, em que **argv[0]** representa o nome do programa em execução, **argv[1]** corresponde ao primeiro parâmetro passado linha de comando e assim sucessivamente. Por sua vez, o inteiro **argc** indica o número de strings existentes no vetor **argv**. O código da Listagem 1 exemplifica a iteração do vetor **argv**. A Listagem 2 apresenta o resultado da execução do código quando o programa é chamado da seguinte forma: **./mostraArgv um dois --tres**

Conforme indicado anteriormente, o primeiro elemento do vetor **argv** (**argv[0]**) corresponde sempre ao nome do programa que está a ser executado, como se denota na Listagem 2 (**./mostraArgv**).

```
int main(int argc, char *argv[])
{
    int i;
    for (i = 0; i < argc ; i++)
    {
        printf("argv[%d]=\"%s'\n", i, argv[i]);
    }
    return 0;
}
```

Listagem 1: iteração do vetor **argv** (**mostraArgv.c**)

```
./mostraArgv um dois --tres
argv[0] = './mostraArgv'
argv[1] = 'um'
argv[2] = 'dois'
argv[3] = '--tres'
```

Listagem 2: exemplo da execução do programa **mostraArgv**

### Opções em formato curto e opções em formato longo

Em ambiente UNIX, é comum as aplicações de modo consola disponibilizarem as opções da linha de comando em dois formatos: formato curto e formato longo. Uma opção curta está limitada a uma letra precedida de um sinal menos (e.g., **-a**). Por sua vez, uma opção longa é composta por uma palavra precedida por dois sinais menos (por exemplo, **--all**). O formato longo é obviamente mais explícito. É frequente ainda uma opção ser disponibilizada no formato curto e no formato longo. Exemplo disso é a opção **-a** e **--all** do comando **ls**. Uma opção pode requerer elementos adicionais, como, por exemplo, o nome de um ficheiro.

Para além das opções e dos respetivos parâmetros, uma aplicação pode ainda solicitar argumentos livres, isto é, que não estejam diretamente associados a opções. É o caso do comando **ls** que pode receber uma lista com os nomes dos ficheiros a serem processados (**ls -l a.txt b.txt c.txt**).

### Processamento manual/explícitos dos parâmetros da linha de comando

Para o programador da aplicação, o processamento dos parâmetros da linha de comando suportados por uma aplicação é frequentemente uma tarefa fastidiosa. De facto, o processamento manual dos parâmetros da linha de comando requer que seja implementado código para a validação das opções. Esse código deve incidir sobre a deteção de opções inválidas, seja porque não existem, seja porque falta um parâmetro complementar à opção, por exemplo, o nome de ficheiro sobre o qual se aplica a opção. Adicionalmente, quando são acrescentadas funcionalidades em novas versões da aplicação, é frequente o aumento do número de opções, requerendo atualizações e acréscimos ao código que trata do processamento das opções da linha de comando e dos respetivos parâmetros.

A Listagem 3 apresenta código rudimentar para o tratamento manual de opções da linha de comando. Concretamente, o código processa as opções **-a**, **--all** (equivalente à opção **-a**), **--help** e **-f**. A opção **-f** requer um nome de ficheiro como parâmetro adicional. A opção **--help**, que existe na maioria das aplicações, apresenta ajuda sucinta sobre a aplicação, no-meadamente sobre as opções da linha de comandos que disponibiliza. Apesar de lidar somente com quatro opções, e apenas uma delas ter um parâmetro adicional obrigatório, o

# A PROGRAMAR

## A FERRAMENTA GENGETOPT

código mostrado na Listagem 3 tem um tamanho apreciável e uma relativa complexidade devido às várias estruturas de controlo **if**. Importa ainda notar que o código apresentado não atende a todas as situações, pois não deteta a múltipla indicação de uma opção (e.g., **-a -a -a** ou **-a --all**), ou a indicação de uma opção em lugar do nome do ficheiro para a opção **-f** (e.g., **-a -f --all**).

O código não considera ainda a possibilidade de uma ou mais opções poderem ser obrigatórias, no sentido que a aplicação só deve prosseguir a execução caso as opções definidas como obrigatórias tenham sido indicadas pelo utilizador. Por fim, o acréscimo de mais opções à aplicação requer uma substancial extensão do código, diminuindo a legibilidade do mesmo e aumentando os custos de manutenção. Por exemplo, o acrescentar de uma nova opção requer não só escrever o código no ciclo **while** para tratamento da opção, mas também a atualização da função **Help**.

### Processamento assistido das opções da linha de comando - as funções getopt, getopt\_long e argp

Sendo o processamento dos parâmetros da linha de comando uma tarefa relativamente entediante, não é de estranhar que tenham surgido várias metodologias com o objetivo de reduzir o volume de código e trabalho do programador. Umas dessas metodologias assenta no uso da API (*Application Programming Interface*) **getopt** [getoptOnline]. Essa API assenta no uso da função **getopt** que recebendo uma string especialmente formatada com as possíveis opções curtas a suportar, devolve as opções que foram indicadas pelo utilizador. Por exemplo, para suportar as opções **-a** e **-f**, especifica-se a string “**af:**”, com os dois pontos a indicarem que a opção **-f** requer obrigatoriamente um parâmetro adicional. Um exemplo do uso da função **getopt** pode ser encontrado em [getoptExemplo]. Para suprir as limitações da função **getopt**, existe a função **getopt\_long** que suporta o processamento de opções curtas e longas, mas a custo de uma acrescida complexidade como se denota pelo protótipo da função mostrado na Listagem 4.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/** Programa que processa (sumariamente) as opções:
 * -a/-all, --help, -f <ficheiro>
 */
void Help(const char *msg){
    if( msg != NULL ){
        fprintf(stderr, "[ERRO]: %s\n", msg);
    }
    printf("Opções\n");
    printf("-a / -all      processa tudo\n");
    printf("-f nome        processa nomeficheiro\n");
    printf("--help         exibe ajuda sucinta\n");
}

int main(int argc, char *argv[]){
    int i;
    char Err_S[256];
    i=1; /* argv[0] --> nome do programa */
    while (i < argc){
        if (strcmp(argv[i], "-a") == 0) {
```

```
        printf("opção -a\n");
    }
    else if (strcmp(argv[i], "--all") == 0) {
        printf("opção --all\n");
    }
    else if(strcmp(argv[i], "-f") == 0) {
        /* opção -f requer ficheiro */
        i++;
        if (i == argc) {
            Help("Falta fich opção -f");
            exit(EXIT_FAILURE);
        }
        else {
            printf("-f fich=%s\n", argv[i]);
        }
    }
    else if (strcmp(argv[i], "--help") == 0) {
        printf("Opção --help\n");
        Help(NULL);
        exit(EXIT_SUCCESS);
    }
    else {
        sprintf(Err_S, "Erro '%s'", argv[i]);
        Help(Err_S);
        exit(EXIT_FAILURE);
    }
    i++;
}/* while */
return 0;
}
```

Listagem 3: processamento rudimentar de opções

```
int getopt (int argc, char *const *argv,
           const char *options);

int getopt_long (int argc, char *const *argv,
                 const char *shortopts,
                 const struct option *longopts,
                 int *indexptr);

error_t argp_parse (const struct argp *argp,
                    int argc, char **argv,
                    unsigned flags, int *arg_index,
```

Listagem 4: protótipos das funções getopt, getopt\_long e argp

A função **argp** é uma outra forma de automatizar algum do processamento dos parâmetros da linha de comando, similar ao **getopt** e ao **getopt\_long**, embora com mais alguma funcionalidade. Ao leitor interessado recomenda-se a consulta de [argp2010].

### Processamento automatizado dos parâmetros da linha de comando: o gerador de código gengetopt

O maior entrave ao uso das funções **getopt** e **getopt\_long** deve-se à sua relativa complexidade e à necessidade de se recorrer a um formato pouco amigável para a especificação dos parâmetros a tratar. O objetivo da ferramenta **gengetopt** (*generate getopt*) é precisamente libertar o programador da interação direta com as pouco amigáveis funções da família **getopt** [gengetoptOnline]. Para o efeito, o **gengetopt** constrói, a partir de um simples ficheiro de configuração fornecido pelo programador, o código em linguagem C para interagir com o **getopt/getopt\_long**. Através do ficheiro de configuração, o programador expressa, no formato próprio e relativamente

# A PROGRAMAR

## A FERRAMENTA GENGETOPT

mente simples do **gengetopt**, os parâmetros da linha de comando reconhecidos que devem ser aceites como válidos pela aplicação. Quando executado com o ficheiro de configuração, o **gengetopt** cria um ficheiro de código C e o respetivo ficheiro .h que devem ser integrados no projeto da aplicação. Na sua utilização mais simples e mais usual de validação dos parâmetros da linha de comando, o programador apenas tem que chamar a função **cmdline\_parser** criada pelo **gengetopt**. Para além de proceder à validação das opções da linha de comando, a função **cmdline\_parser** preenche a estrutura do tipo **gengetopt\_args\_info**, estrutura criada pelo **gengetopt** e cujos campos refletem a configuração expressa pelo programador.

### Exemplo basico.ggo

A listagem do ficheiro de texto **basico.ggo** (Listagem 5) exibe um exemplo básico de ficheiro de configuração **gengetopt**. Embora não seja obrigatório, é usual o nome de um ficheiro de configuração **gengetopt** ter a extensão **.ggo**. Na primeira parte do ficheiro são descritos o pacote de software (**package**), a versão (**version**), o propósito do software (**purpose**) e ainda uma descrição do software (**description**). As opções propriamente ditas são identificadas através da palavra-chave **option**. Concretamente, cada opção curta/longa é descrita por uma linha iniciada por **option**. Finalmente, o símbolo **#** serve para iniciar um comentário que se prolonga até ao fim da linha.

No exemplo **basico.ggo** são definidas as opções **-a/-all** e a opção **-f/-ficheiro** através das respetivas linhas **option**. Para o efeito, e após a palavra-chave **option**, cada linha identifica o nome da opção longa entre aspas (uma opção longa não pode ter espaços no nome, embora admita o símbolo **\_**), seguido da letra empregue para a opção curta ou de um sinal menos (**-**) que indica que a opção não tem versão curta. Segue-se a descrição da opção, através de texto delimitado por aspas. Esta descrição é empregue pelo **gengetopt** para gerar o texto de ajuda sucinta disponibilizado através da opção **--help**. A configuração que segue a descrição da opção depende do tipo de opção. No caso do par de opção **-a/-all** é apenas especificado que se trata de uma opção optional, isto é, não obrigatória. Por omissão, uma opção é considerada obrigatória (**required**) pelo **gengetopt**, isto é, se a opção não for especificada na execução da aplicação, a execução da aplicação termina com uma mensagem de erro a indicar a falta da opção obrigatória.

A opção **-f/-ficheiro** é um pouco mais elaborada, dado que requer que seja acrescentado, logo a seguir à opção, uma string correspondente a um nome de ficheiro. Essa informação é transmitida na configuração do **gengetopt** através da indicação **string** (tipo de elemento a acrescentar). Por sua vez, a especificação **typestr="filename"** serve para indicar que o valor a especificar após a opção se destina a ser interpretado pela aplicação como o nome de um ficheiro. Importa notar que o **gengetopt** não produz código para a validação do nome indicado como nome de ficheiro, sendo a especificação **typestr="filename"** somente empregue para a documentação acessível através da opção **--help**.

```
package "gengetopt_basico"
version "1.0"
purpose "Exemplificar gengetopt"
description "Exemplo básico do gengetopt"

# Definição das opções
option "all" a "opção -a / -all" optional
option "ficheiro" f "requer nome de ficheiro"
typestr="filename" string optional
```

Listagem 5: ficheiro de configuração **basico.ggo**

### Execução do gengetopt

Se o **gengetopt** não estiver disponível no sistema, é esperável que possa ser facilmente instalado através do sistema de gestão de software da distribuição de linux em uso, sobretudo no caso de se estar a usar uma distribuição popular de linux (Ubuntu, Fedora, etc.). Por exemplo, numa distribuição **Ubuntu**, a instalação é efetuada através da execução de **sudo apt-get install gengetopt**.

Finalizado o ficheiro de configuração, executa-se o **gengetopt** da seguinte forma: **gengetopt < basico.ggo**. A necessidade de se especificar o ficheiro de configuração **.ggo** através do redirecionamento de entrada (**< basico.ggo**) deve-se ao facto do **gengetopt**, por omissão, processar a entrada padrão (**stdin**). Caso se pretenda evitar o redirecionamento de entrada, deve-se especificar na execução do **gengetopt** a opção **-i** (ou a versão longa **--input**) seguida do nome do ficheiro de configuração a processar (**gengetopt -i basico.ggo** ou **gengetopt --input=basico.ggo**).

A execução do **gengetopt** leva à criação de dois ficheiros de código fonte: **cmdline.c** e **cmdline.h**. Estes ficheiros contêm o código (ficheiro **cmdline.c**), as estruturas e os protótipos (ficheiro **cmdline.h**) criadas pelo **gengetopt** para processamento das opções da linha de comando suportadas pela aplicação. Através da opção **-F nome** (ou **--file-name nome**) é possível indicar ao **gengetopt nome** como alternativa a **cmdline** para a criação dos ficheiros de código fonte. Por exemplo, executando-se **gengetopt -i basico.ggo -F params**, o **gengetopt** cria os ficheiros **params.c** e **params.h** em lugar dos ficheiros **cmdline.c** e **cmdline.h**.

### Utilização do código gerado pelo gengetopt

Para integrar a funcionalidade de processamento das opções especificadas na linha de comando, o programador da aplicação deve chamar a função **cmdline\_parser** que foi criada pelo **gengetopt**. Esta função recebe como parâmetros, para além de **argc** e **argv**, o endereço de memória de uma estrutura do tipo **struct gengetopt\_args\_info**. O protótipo da função **cmdline\_parser** e a declaração da estrutura **gengetopt\_args\_info** são mostrados na Listagem 6.

```
struct gengetopt_args_info{
    const char *help_help;
    const char *version_help;
    const char *all_help;
    char *ficheiro_arg;
    char *ficheiro_orig;
    const char *ficheiro_help;
```

# A PROGRAMAR

## A FERRAMENTA GENGETOPT

```
unsigned int help_given;
unsigned int version_given;
unsigned int all_given;
unsigned int ficheiro_given;
} ;

int cmdline_parser (int argc, char **argv,
    struct gengetopt_args_info *args_info);
```

Listagem 6: estrutura struct gengetopt\_args\_info e protótipo da função cmdline\_parser

O código da função main (Listagem 7) limita-se a chamar a função **cmdline\_parser**, sem ter em consideração os valores da **struct gengetopt\_args\_info** que são preenchidos pela função. Com essa utilização simples, o programador está somente a fazer uso da validação de parâmetros. A compilação da aplicação, requer que sejam primeiros criados os ficheiros de código objeto **main1.o** e **cmdline.o** a partir dos respetivos ficheiros de código fonte C, sendo os ficheiros de código objetos posteriormente *linkados* para a criação do ficheiro executável. A execução do **gengetopt**, bem como os passos de compilação e ligação necessários à criação do executável (**main1.exe**) da aplicação encontram-se na Listagem 8.

```
#include <stdio.h>
#include <stdlib.h>
#include "cmdline.h"

int main(int argc, char *argv[]) {
    struct gengetopt_args_info ArgsInfo;
    if (cmdline_parser(argc, argv, &ArgsInfo)) {
        fprintf(stderr, "Erro: execução de
                           cmdline_parser\n");
        exit(EXIT_FAILURE);
    }
    return 0;
}
```

Listagem 7: main1.c

```
gengetopt < basico.ggo
gcc -Wall -W -c cmdline.c
gcc -Wall -W -c main1.c
gcc -Wall -W cmdline.o main1.o -o main1.exe
```

Listagem 8: execução do gengetopt e criação do ficheiro executável main1.exe

Apesar de ainda não ter nenhuma funcionalidade implementada pelo programador, a aplicação **main1.exe** já deteta opções inválidas, isto é, opções não especificadas no ficheiro de configuração. Por exemplo, a execução **./main1.exe -j** leva ao término abrupto do programa com uma mensagem de erro a indicar que a opção **-j** é inválida. Igualmente, caso seja especificada a opção **-f/-ficheiro** sem que seja seguida por uma string (que seria interpretada como nome de ficheiro), a aplicação termina reportando que a opção **-f/-ficheiro** requer um argumento. A aplicação suporta ainda as opções **-h/-help** e **-V/-version**. Estas opções são automaticamente criadas pelo **gengetopt** a partir da informação fornecida no ficheiro de configuração **.ggo**. A saída produzida pela opção **--help** é mostrada na Listagem 9.

```
./main1.exe --help
RevistaProgramar 1.0
```

```
Exemplificar gengetopt
Usage: RevistaProgramar [OPTIONS] ...
Exemplo básico do gengetopt
-h, --help                  Print help and exit
-V, --version                Print version and exit
-a, --all                   opção -a / -all
-f, --ficheiro=filename     requer nome de ficheiro
```

Listagem 9: saída da execução de **./main1.exe --help**

### Utilização dos parâmetros passados pela linha de comando

Obviamente, o uso do **gengetopt** não se esgota na validação de parâmetros. O programador necessita de ter acesso a informação que permita determinar quais foram as opções indicadas na linha de comando, e no caso de opções com parâmetros, qual o valor dos parâmetros. Essa informação é disponibilizada pelos vários campos da **struct gengetopt\_args\_info** (Listagem 6). De facto, para cada opção especificada no ficheiro de configuração **.ggo**, o **gengetopt** cria um conjunto de campos na **struct gengetopt\_args\_info**. Por exemplo, para a opção **--all**, existem os campos **unsigned int all\_given** e **const char \*all\_help**. O primeiro apresenta um valor inteiro positivo caso a opção **-a/-all** tenha sido indicada na linha de comando, ao passo que o campo **all\_help** contém o texto de ajuda definido para a opção **-a/-all**.

Por sua vez, a opção **-f/-ficheiro**, que requer um argumento string adicional, está representada por quatro campos na **struct gengetopt\_args\_info**: **unsigned int ficheiro\_given**, **const char \*ficheiro\_help**, **char \*ficheiro\_arg** e **char \*ficheiro\_orig**. Os dois primeiros campos tem funcionalidade análoga aos anteriormente vistos para a opção **-a/-all**. O campo **char \*ficheiro\_arg** representa o parâmetro adicional requerido pela opção, ao passo que o campo **char \*ficheiro\_orig** representa a string que foi passada na linha de comando. Neste caso, e por via do parâmetro adicional para **-f/-ficheiro** ter sido declarado como string, os campos **ficheiro\_arg** e **ficheiro\_orig** tem o mesmo valor.

Através do uso dos campos da estrutura **struct gengetopt\_args\_info**, o programador pode determinar quais as opções que foram indicadas e levar a aplicação a agir em concordância com o especificado na linha de comando. Assim, para detetar se uma determinada opção foi indicada, a aplicação testa o campo **OPCAO\_given**, em que **OPCAO** corresponde ao nome da opção. Caso a opção tenha sido definida como tendo um parâmetro, a aplicação pode aceder ao parâmetro através do campo **OPCAO\_arg**. A listagem **main2.c** exemplifica o uso dos campos **OPCAO\_given** e **OPCAO\_arg**. A aplicação **main2.exe** pode ser compilada da seguinte forma: **gcc -Wall -Wextra cmdline.c main2.c -o main2.exe**

```
#include <stdio.h>
#include <stdlib.h>
#include "cmdline.h"
```

# A PROGRAMAR

## A FERRAMENTA GENGETOPT

```
int main(int argc, char *argv[]) {
    struct gengetopt_args_info ArgsInfo;
    if (cmdline_parser(argc, argv, &ArgsInfo)) {
        fprintf(stderr, "Erro: execução de
                           cmdline_parser\n");
        exit(EXIT_FAILURE);
    }
    if (ArgsInfo.all_given) {
        printf("-a/-all\n");
    }
    if (ArgsInfo.ficheiro_given)
    {
        printf("-f/--ficheiro com '%s'\n",
               ArgsInfo.ficheiro_arg);
    }
    printf("Finito!\n");
    return 0;
}
```

Listagem 10: main2.c

### Explorando as capacidades do gengetopt

O **gengetopt** possui um vasto leque de funcionalidades para além das empregues no exemplo **basico.ggo**. O ficheiro de configuração **funcionalidades.ggo** apresenta mais algumas das funcionalidades do **gengetopt** (Listagem 11).

```
package "RevistaProgramar"
version "2.0"
purpose "Exemplificar gengetopt"
description "Exemplo do gengetopt"
# Definição das opções
option "obrigatoria" o "obrigatoria sem parâmetro"
option "opcional" - "opcional sem parâmetro" optional
option "flag" - "flag (estado inicial: off)" flag off
option "preenchido" p "string com valor por omissão"
                           string default="AAA"
option "argopcional" - "opção com argumento opcional"
                           float argoptional default="3.14"
option "numero" n "requer numero inteiro
(optional)"
                           long optional
option "multi" m "opção que pode ser especificada
mais do que uma vez" optional multiple(1-3)
option "multi2" - "opção múltipla até 4 vezes com
parâmetro string" string optional multiple(1-4)
option "enumerada" - "Uma opção cujo possíveis
valores são especificados por values"
values="zero","um","dois" default="zero" optional
```

Listagem 11: ficheiro de configuração funcionalidades.ggo

De seguida são analisadas algumas das funcionalidades que constam do ficheiro funcionalidades.ggo (Listagem 11).

**-o--obrigatoria:** opção obrigatória (por omissão, uma opção é do tipo *required*, isto é, é obrigatória)

**-opcional:** opção opcional (uso da palavra-chave *optional*). A opção não tem versão curta

**--flag:** opção do tipo **flag** que apenas tem dois estados: **on** e **off**. Neste caso está a **off** por omissão. Se for especificado **--flag** na linha de comando, o valor de **flag\_arg** da estrutura **struct gengetopt\_args\_info** passa a **on**

**-p--preenchido:** opção obrigatória do tipo **string** preenchida com o valor por omissão "AAA". O **gengetopt** requer que o valor por omissão deve ser sempre indicado entre aspas.

**--argopcional:** opção que tem um argumento do tipo **double** que é opcional, sendo ainda definido o valor por omissão de "3.14".

**-n--numero:** opção que requer um número do tipo **long** quando é especificada. O campo **numero\_arg** da estrutura **struct gengetopt\_args\_info** é do tipo **long**. Para além dos tipos de dados **long**, **string** e **double**, o **gengetopt** suporta ainda **int**, **short**, **float**, **longdouble** e **longlong**

**-m--multi:** opção que pode ser especificada múltiplas vezes na mesma linha de comando. No exemplo (**multiple(1-3)**), pode ser indicada entre uma a três vezes. Nesta forma sem parâmetro, a opção **multiple** é empregue nalguns programas para reforçar o nível de verbosidade das mensagens de depuração (**-v** algumas mensagens de debug, **-vv** mais mensagens de debug, etc.). O campo **unsigned int multi\_given** da estrutura **struct gengetopt\_args\_info** devolve o número de vezes que a opção foi especificada

**--multi2:** Opção múltipla mas que requer parâmetro. Neste caso, a opção **--multi2** pode ser indicada com um a quatro parâmetros (**multiple(1-4)**) do tipo **string** através da seguinte forma **--multi2=str1,str2,str3** ou **--multi2=str1 --multi2=str2 --multi2=str3**. Em termos de código, o valor dos vários parâmetros está acessível através do vetor de strings **char \*\* multi2\_arg** da estrutura **struct gengetopt\_args\_info**, sendo o número de elementos do vetor indicado pelo campo **unsigned int multi2\_given**.

**--enumerada:** opção cujo possíveis valores são especificados pelo campo "values". Neste caso, os valores possíveis são "zero", "um" ou "dois", sendo "zero" o valor por omissão.

A Listagem 12 mostra a estrutura **struct gengetopt\_args\_info** que foi gerada pelo **gengetopt** a partir do ficheiro de configuração **funcionalidades.ggo**.

### Opções avançadas: *groups* e *mode*

#### Grupos

Para além da possibilidade de configuração individual de cada opção, o **gengetopt** suporta ainda a definição de grupo de opções. Para o **gengetopt**, um grupo de opções é um conjunto de opções que estão em exclusão mútua, isto é, na execução da aplicação apenas pode ser especificada uma das opções pertencente a um dado grupo.

A definição de um grupo chamado **NomeGrupo** faz-se através de **defgroup NomeGrupo**. Cada opção pertencente ao

# A PROGRAMAR

## A FERRAMENTA GENETOPT

grupo é definida pela palavra-chave **groupoption** em que o parâmetro **group** indica o nome do grupo a que pertence. Para além da necessidade de indicação do grupo a que pertence, a definição de uma opção via **groupoption** é similar à definição de uma opção simples, com exceção que a opção não pode ser definida como obrigatória, pois tal contradiz o propósito de um grupo. Importa ainda referir que caso um grupo seja definido como obrigatório (**required**), então uma e uma só das opções do grupo terá que ser especificada aquando da execução da aplicação. A listagem mostra o ficheiro **grupos.ggo** que exemplifica a definição de grupos (**grupo\_1** e **grupo\_2**), bem como da opção independente "**normal1**". As linhas iniciadas pela palavra-chave **section** correspondem a divisões que apenas têm efeito na documentação acessível através da opção **--help**, com uma separação visual por secções. Um exemplo de utilização da funcionalidade de grupos é mostrado na Listagem 13, com o ficheiro de configuração **grupos.ggo**.

```
struct genetopt_args_info{
    const char *help_help;
    const char *version_help;
    const char *obrigatoria_help;
    const char *opcional_help;
    int flag_flag;
    const char *flag_help;
    char *ficheiro_arg;
    char *ficheiro_orig;
    const char *ficheiro_help;
    char *preenchido_arg;
    char *preenchido_orig;
    const char *preenchido_help;
    long numero_arg;
    char *numero_orig;
    const char *numero_help;
    float argopcional_arg;
    char *argopcional_orig;
    const char *argopcional_help;
    char *enumerada_arg;
    char *enumerada_orig;
    const char *enumerada_help;
    unsigned int multi_min;
    unsigned int multi_max;
    const char *multi_help;
    char **multi2_arg;
    char **multi2_orig;
    unsigned int multi2_min;
    unsigned int multi2_max;
    const char *multi2_help;
    unsigned int help_given;
    unsigned int version_given;
    unsigned int obrigatoria_given;
    unsigned int opcional_given;
    unsigned int flag_given;
    unsigned int ficheiro_given;
    unsigned int preenchido_given;
    unsigned int numero_given;
    unsigned int argopcional_given;
    unsigned int enumerada_given;
    unsigned int multi_given;
    unsigned int multi2_given;
};

};
```

Listagem 12: estrutura struct **genetopt\_args\_info** criada pelo **genetopt** a partir da configuração **funcionalidades.ggo**

```
section "grupo_1"
defgroup "grupo_1" groupdesc="definição do grupo_1"
required
```

```
groupoption "opA_grupo1" - "opção opA do grupo 1"
            group="grupo_1" string argoptional
groupoption "opB_grupo1" - "opção opB do grupo 1"
            group="grupo_1" string multiple
groupoption "opC_grupo1" - "opção opC do grupo 1"
            group="grupo_1" string default="BBB"

section "grupo_2"
defgroup "grupo_2" groupdesc="definição do grupo 2"
groupoption "op1_grupo2" - "opção op1 do grupo 2"
            group="grupo_2" float argoptional
groupoption "op2_grupo2" - "opção op2 do grupo 2"
            group="grupo_2" int multiple(1-3)
groupoption "op3_grupo2" - "opção op3 do grupo 2"
            group="grupo_2" string default="CCC"

# Opções independentes
section "opções independentes"
option "normal1" n "Opção normal que não pertence a
nenhum grupo" double default="2.71828"
```

Listagem 13: ficheiro de configuração **grupos.ggo**

### Modos

O **genetopt** suporta ainda outra forma de agregação de opções, através da funcionalidade de "modes". Como o nome sugere, a funcionalidade de **modo** permite a definição de vários modos de execução da aplicação. Assim, por exemplo, é possível definir um **modo A** que implementa determinado tipo de funcionalidade, um **modo B** que implementa outro tipo de funcionalidade e um **modo C** que executa outro tipo de operações. Os modos são mutuamente exclusivos, no sentido em que a execução da aplicação com a indicação de uma ou mais opções de um modo é incompatível com o uso de opções de qualquer outro modo. Mesmo com a existência de modos, o **genetopt** continua a suportar opções independentes (não pertencentes a modos nem a grupos) que são definidas da forma habitual.

A definição de um modo faz-se com a palavra-chave **defmodo** atribuindo-se um nome ao modo. A definição de uma opção pertencente a um modo é feita com a declaração **modeoption**, usando-se o parâmetro **mode=nomeModo** para associação da opção ao modo **nomeModo**. A Listagem 14 **modes.ggo** exemplifica a definição de modos e de opções independentes.

```
package "RevistaProgramar"
version "1.0"
purpose "Exemplificar genetopt/ modos"
description "Exemplo de modos do genetopt"
versiontext "Patrício R. Domingues - Revista Programar"

# opções "independentes"
section "opções independentes"
option "normal1" n "Opção normal que não pertence a
nenhum modo" double default="2.71828"
option "normal2" i "Outra opção que não pertence a
nenhum modo" optional int default="20"

section "definição de modos"
# modo_1
defmode "modo_1" modeDESC="modo 1"
modeoption "opt_A" - "opção A do modo 1" mo-
```

```
de="modo_1" string optional  
modeoption "opt_B" - "opção B do modo 1" mo-  
de="modo_1" string typestr="filename" optional de-  
fault="modo1.txt"  
modeoption "opt_C" - "opção C do modo 1" mo-  
de="modo_1" long argoptional default="5"  
  
defmode "modo_2" modeDESC="modo 2"  
modeoption "opt_X" - "opção X do modo 2" longlong  
default="123456789" mode="modo_2"  
modeoption "opt_Y" - "opção Y do modo 2" string de-  
fault="modo2.txt" mode="modo_2"  
modeoption "opt_Z" - "opção Z do modo 2" string  
multiple(1-2) mode="modo_2"
```

Listagem 14: ficheiro de configuração modes.ggo

### Notas finais

O utilitário **gengetopt** simplifica significativamente a gestão das opções da linha de comando de um programa escrito em linguagem C. Em lugar de desenvolver código à medida das opções da linha de comando que pretende que sejam suportadas pela aplicação, o programador define as necessidades da aplicação através do ficheiro de configuração **.ggo** a passar ao **gengetopt**. A intuitiva linguagem de configuração do **gengetopt** e as funcionalidades que a ferramenta disponibiliza possibilitam ao programador o criar de soluções poderosas e facilmente extensíveis para o processamento das opções passadas pela linha de comando.

O **gengetopt** é suportado para ambiente UNIX, nomeadamente para distribuições de Linux. Embora exista um porte do **gengetopt** para ambiente Windows [**gengetoptWindows**], está um pouco desatualizado, dado que corresponde à versão 2.20, ao passo que a versão corrente do gengetopt é a 2.22.6. Acresce-se que dado o **gengetopt** gerar código dependente das funções **getopt** e **getopt\_long** leva a que também seja necessário o porte dessas funções para ambiente Windows. Isso ocorre, por exemplo, no sistema Cygwin [**cygwinOnline**].

### Saber mais

Este artigo apresentou o gengetopt e algumas das suas funcionalidades. Para um conhecimento mais aprofundado da ferramenta gengetopt, como por exemplo, das opções da linha de comando suportadas pela própria ferramenta, recomenda-se a leitura do sítio web [**gengetoptOnline**] e da própria página do manual do **gengetopt**, acessível através de **man gengetopt**. Recomenda-se ainda uma análise atenta ao código **.c** e **.h** gerado pelo **gengetopt**.

## AUTOR

Escrito por Patrício Domingues

Patrício Domingues é professor do Departamento de Engº Informática da Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria (IPLLeiria). Leciona, entre outras, a disciplina de Programação Avançada ao curso de Licenciatura em Engenharia Informática, onde tem o privilégio de trabalhar com os colegas docentes Gustavo Reis e Vítor Carreira.

**De facto, através dos parâmetros da linha de comando, torna-se possível especificar múltiplos comportamentos e funcionalidades para uma mesma aplicação de modo consola/terminal**

### Bibliografia

- [**argp2010**] Ben Asselstine, “Step-by-Step into Argp”, 2010. <http://bit.ly/argp2010>
- [**cygwinOnline**] Cygwin Online, <http://cygwin.com/>
- [**gengetoptOnline**] Documentação online do gengetopt. <http://bit.ly/gengetopt>
- [**gengetoptWindows**] Porte do gengetopt para windows. <http://bit.ly/gengetoptwin32>
- [**getoptOnline**] Documentação online do getopt. <http://bit.ly/getopt>
- [**getoptExemplo**] Exemplo do uso da função getopt. <http://bit.ly/getoptExemplo>