Dedicatorias etc

[Año]

Yihui Xie

[Nombre de la compañía]

[Fecha]

A Minimal Book Example



Indice

Contenido

[Prerequisites 0](#_Toc86609921)

[Abstract 0](#_Toc86609922)

[disclaimer 0](#_Toc86609923)

[Introduction 1](#_Toc86609924)

[Literature 2](#_Toc86609925)

[Literature 2](#_Toc86609926)

[Especificacion de argumentos 3](#_Toc86609927)

[sistemas nix 3](#_Toc86609928)

[sistemas windows 3](#_Toc86609929)

[ZOS y otros 3](#_Toc86609930)

[Conclusiones 3](#_Toc86609931)

[Definiciones 3](#_Toc86609932)

[Datos de entrada 3](#_Toc86609933)

[Opciones 3](#_Toc86609934)

[Flags 3](#_Toc86609935)

[Incoherencias 3](#_Toc86609936)

[Methods 3](#_Toc86609937)

[Applications 3](#_Toc86609938)

[Example one 3](#_Toc86609939)

[Example two 3](#_Toc86609940)

[Final Words 3](#_Toc86609941)

[Definiciones 3](#_Toc86609942)

[Parametro 4](#_Toc86609943)

[Argumento 4](#_Toc86609944)

[Opcion 4](#_Toc86609945)

[Flags 4](#_Toc86609946)

[Concatenacion 4](#_Toc86609947)

[Definicion 4](#_Toc86609948)

[Casos especiales 4](#_Toc86609949)

[help 4](#_Toc86609950)

[Acceso a las variables de entorno 4](#_Toc86609951)

[Valores booleanos 4](#_Toc86609952)

[Ejemplos de codigos 4](#_Toc86609953)

[Verbos 4](#_Toc86609954)

[Libreria cmdline 4](#_Toc86609955)

[Introduccion 5](#_Toc86609956)

[Enumeraciones 5](#_Toc86609957)

[Type 5](#_Toc86609958)

[Source 5](#_Toc86609959)

[Parametros 5](#_Toc86609960)

[Flags 5](#_Toc86609961)

[Excepciones 5](#_Toc86609962)

Abstract

Introduciion

A Minimal Book Example

Yihui Xie

2021-10-31

# Prerequisites

This is a *sample* book written in **Markdown**. You can use anything that Pandoc’s Markdown supports, e.g., a math equation .

The **bookdown** package can be installed from CRAN or Github:

install.packages("bookdown")  
# or the development version  
# devtools::install\_github("rstudio/bookdown")

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading #.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): <https://yihui.org/tinytex/>.

# Abstract

Este documento plantea una propuesta universal para la gestión de los argumentos pasados a un programa por línea de comandos para cualquier sistema operativo junto con el manual de uso para programas desarrollados en C++

# disclaimer

As my mother tongue is Spanish and not forgetting that it is one of the most widely spoken languages ​​in the world but for sack of accuracy this document has been written is Castillian

However, if someone is interested in the job or wants more information, I will be happy to translate the documents or help as much as possible.

In any case, for readability and maintainability reasons, I use English inside the code.

# Introduction

You can label chapter and section titles using {#label} after them, e.g., we can reference Chapter @ref(intro). If you do not manually label them, there will be automatic labels anyway, e.g., Chapter @ref(methods).

Figures and tables with captions will be placed in figure and table environments, respectively.

par(mar = c(4, 4, .1, .1))  
plot(pressure, type = 'b', pch = 19)



Here is a nice figure!

Reference a figure by its code chunk label with the fig: prefix, e.g., see Figure @ref(fig:nice-fig). Similarly, you can reference tables generated from knitr::kable(), e.g., see Table @ref(tab:nice-tab).

knitr::kable(  
 head(iris, 20), caption = 'Here is a nice table!',  
 booktabs = TRUE  
)

Here is a nice table!

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| --- | --- | --- | --- | --- |
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| 4.8 | 3.4 | 1.6 | 0.2 | setosa |
| 4.8 | 3.0 | 1.4 | 0.1 | setosa |
| 4.3 | 3.0 | 1.1 | 0.1 | setosa |
| 5.8 | 4.0 | 1.2 | 0.2 | setosa |
| 5.7 | 4.4 | 1.5 | 0.4 | setosa |
| 5.4 | 3.9 | 1.3 | 0.4 | setosa |
| 5.1 | 3.5 | 1.4 | 0.3 | setosa |
| 5.7 | 3.8 | 1.7 | 0.3 | setosa |
| 5.1 | 3.8 | 1.5 | 0.3 | setosa |

You can write citations, too. For example, we are using the **bookdown** package ([Xie 2020](#ref-R-bookdown)) in this sample book, which was built on top of R Markdown and **knitr** ([Xie 2015](#ref-xie2015)).

# Literature

Here is a review of existing methods.

# Literature

Placeholder

## Especificacion de argumentos

### sistemas nix

### sistemas windows

### ZOS y otros

## Conclusiones

## Definiciones

### Datos de entrada

### Opciones

### Flags

## Incoherencias

# Methods

We describe our methods in this chapter.

# Applications

Some *significant* applications are demonstrated in this chapter.

## Example one

## Example two

# Final Words

We have finished a nice book.

# Definiciones

Placeholder

## Parametro

## Argumento

## Opcion

## Flags

### Concatenacion

## Definicion

# Casos especiales

Placeholder

## help

## Acceso a las variables de entorno

## Valores booleanos

# Ejemplos de codigos

Placeholder

# Verbos

Por definicion los parametros son inmutables, lo cual no implica que se puedan leer.

* getXXX: Obtiene una referencia al parametro
* letXXX: Obtieneuna copia del parametro
* hasXXX: Indica siexiste o no
* addXXX: Permite incluir parametros

# Libreria cmdline

Placeholder

## Introduccion

## Enumeraciones

### Type

### Source

## Parametros

## Flags

# Excepciones

Si se produce un error durante el proceso de análisis de la línea de comandos o se produce un error en tiempo de ejecución se genera una excepción **CmdLineException**

Distinguimos los siguientes tipos:

CmdLineParameterException;El parámetro es erroneo CmdLineValueException;el valor asociado alparámetro no es valido de acuerdo con el tipo indicado

CmdLineNotFoundException;runtime\_error;Se ha solicitado un parámetro que no existe CmdLineInvalidTypeException;Se ha solicitado una conversion no valida de un parámetro

Además, para gestionar la solicitud de ayuda, se generan las siguientes excepciones:

* HelpRequested
* HelpDetailedRequested

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. <http://yihui.org/knitr/>.

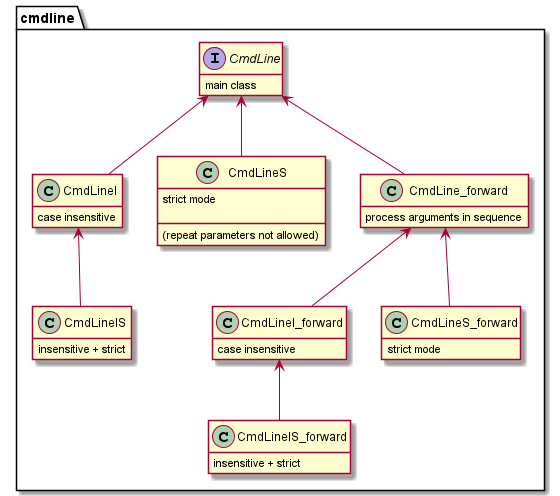
———. 2020. *Bookdown: Authoring Books and Technical Documents with r Markdown*. <https://github.com/rstudio/bookdown>.

Libreria

# Caracteristicas

# Clases

La clase base de la librería se denomina CmdLine y se implementa como singleton



Todas las instancias devuelve un objeto CmdLine

|  |  |  |
| --- | --- | --- |
|  |  |  |
| CmdLine |  |  |
| CmdLineI |  |  |
| CmdLineIS |  |  |
|  |  |  |
| CmdLineS |  |  |

# Excepciones

De acuerdo con los criterios de C++ los posibles errores generan excepciones.

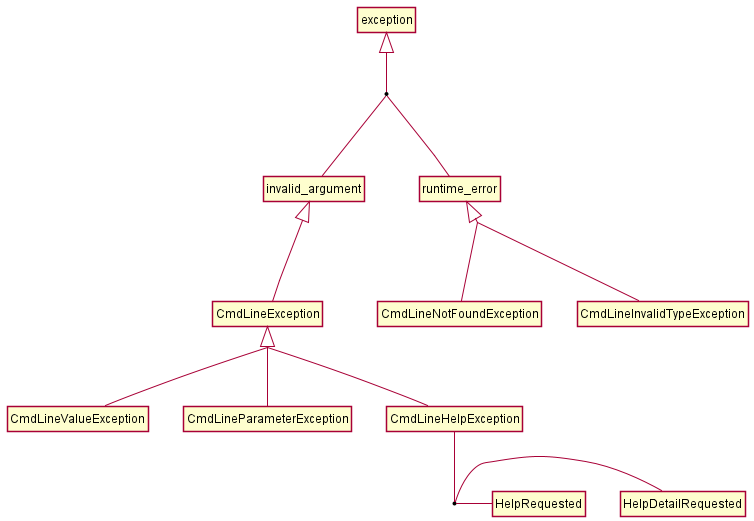
Considerando que la librería analiza la línea de comandos distinguimos dos tipos de excepciones:

* Errores en tiempo de desarrollo
* Entrada invalida de datos

Los primeros se consideran que solo pueden en tiempo de desarrollo por que ocurren cuando se intenta acceder a un parámetro que no existe o no esta definido; es decir, dado que el acceso se produce de manera programática solo puede darse en el caso de que se haya tecleado o definido de manera errónea ese valor en el programa.

Los segundos se producen en función de los datos introducidos por el usuario: nombre del parámetro incorrecto, valores erróneos, etc.

La estructura de las excepciones se muestra en el siguiente diagrama:



# Linea de comandos

La línea de comandos es la forma de ejecutar cualquier programa en un ordenador, ya sea de forma directa, mediante un terminal, o a través de iconos y otras maneras mas o menos amigables. Esta línea de comandos está compuesta por, el nombre y ubicación del programa a ejecutar junto con un conjunto de parámetros que permiten adecuar la ejecución concreta del programa a las condiciones particulares esa ejecución, además de indicar sobe que objeto u objetos debe operar el programa.

Por poner únicamente un ejemplo, el procesador de textos de Microsoft®, WinWord® 365 admite 17 parámetros diferentes en la línea de comandos.

## Taxonomia de la línea de comandos

Como se ha indicado, a continuación del nombre del programa se pueden indicar un conjunto de datos, a los que denominaremos de manera general parámetros o argumentos que modifican o definen el comportamiento predefinido del programa para esa ejecución, estos parámetros solo pueden ser de los siguientes tipos:

1. **Argumentos**: Objetos que se deben procesar
2. **Flags**: Indicadores que activan o desactivan alguna funcionalidad del programa
3. **Opciones**: Datos que cambian o definen determinados componentes predefinidos del programa
4. **Definiciones**: Información que se añade al programa para su posible uso pero que no están predefinidos en el mismo

### Programa ejemplo

Pongamos por ejemplo un programa, no muy útil a efectos prácticos, al que llamaremos collector .

La funcionalidad de este programa es recolectar un conjunto de ficheros de un conjunto de directorios y copiarlos en otro directorio, durante el proceso además, el programa busca dentro de los ficheros ciertas palabras clave, y si existen, las cambia por otras definidas por el usuario. Además, si no se le indica ningún directorio de entrada, y no se indica ninguna lista de ficheros, lee los datos que se introduzcan por consola y si no se le indica un directorio de salida, el resultado lo escribe, con un nombre predeterminado en el directorio actual.

El programa se puede configurar con los siguientes parámetros:

|  |  |  |
| --- | --- | --- |
| Parámetro | Defecto | Descripción |
| In | Consola | Indica el/los directorios donde buscar ficheros |
| Out | Directorio actual | Indica donde guardar las copias |
| file | Collector.txt | Nombre del fichero a generar |
| Overwrite | Si | Sobrescribir un fichero si ya existe en el directorio de salida |
| Pattern | \*.\* | Mascara de ficheros a procesar |
| Verbose | Si | Mostrar o no información de progreso del programa |
| Summary | Si | Mostrar resumen del resultado de la ejecución |
| Define | N/A | Conjunto de pares palabra clave/valor que se deben sustituir |

De acuerdo con las convenciones actuales la definición del programa podría ser parecido a esta:

Método de parámetros largos \*nix (--nombre):

Collector [–-overwrite -–verbose --summary --out dir\_output --file file\_name

[--in dir\_input]\* [--pattern pattern]\* [--define clave=valor]\* ...

Método de parámetros cortos \*nix (-letra)

Collector [–svw -o dir\_output -f file\_name [-i dir\_input]\* -p pattern]\* [-d clave=valor]\* ...

Método de parámetros Windows:

Collector [/s/v/W /o dir\_output /f:file\_name [/i dir\_input]\* [/p pattern]\* [/d:clave=valor]\* ...

## Analisis

Tomando como referencia el programa anterior, tenemos

Argumentos: Los puntos suspensivos

Flags: overwrite, summary, verbose

Opciones: in, out, pattern

Definiciones: define

Veamos las diferencias entre cada uno de estos tipos:

* Los argumentos son los objetos que debe procesar el programa, por lo tanto son dependientes de cada programa.
* Los flags únicamente indican la activación/desactivación de la funcionalidad correspondiente
* Las opciones necesitan información adicional, pero también están definidas en el programa
* Las definiciones también necesitan información adicional, indicada por el signo igual (=) pero a diferencia de las opciones, no están definidas en el programa

Es decir, si ejecutamos alguno de los siguientes comandos:

Collector -x

Collector -v -o

Collector /j opcion

El programa deberá emitir algún mensaje de error indicando que el parámetro no existe o que le faltan datos. Sin embargo, el siguiente comando:

collector -d autor=yo\_mismo

El programa no puede identificar, a priori, si ‘autor’ será útil o no en el programa.

Por otro lado, los siguientes comandos son equivalentes:

collector –-verbose -w

collector –-overwrite

collector

Puesto que los flags verbose y overwrite están activados por defecto

Esto obliga a incluir otras opciones en el programa, las que inhabilitan las funcionalidades activas por defecto en el programa, en nuestro caso asumiremos que son:

-m | --mute para no mostrar la información de progreso

-n | --nooverwrite para evitar sobreescribir ficheros existentes

-r | --nosummary para no emitir el resumen

Con lo que el programa completo se define ahora como:

Collector [–mnrsvw -o dir\_output -f file\_name [-i dir\_input]\* -p pattern]\* [-d clave=valor]\* ...

## Analisis

Tomando como referencia el programa anterior, tenemos

1. En función del sistema operativo el modo de indicar los parámetros difiere
2. No se puede identificar que es un flag o una opción
3. No se puede conocer directamente si se está activando o desactivando un flag
4. El uso de parámetros largos es proclive a errores de escritura

Ademas hay que tener en cuenta que el programa debe