9/5/2021

Kookaburra

The complete guide.

AZ Software
GITHUB: AZPRODUCTIONS

Table of contents

1. Getting Started

If you've never coded before, programming can seem difficult. For that reason, we created this E-Book to help new coders. For more experienced developers, this guide may not be necessary.

This book contains multiple chapters, sub-chapters and extra tips/tricks from the team who created Kookaburra. Let's code!

1.1 Download Kookaburra.

When downloading Kookaburra, you can chose between getting it from the Microsoft Store or downloading it via GitHub. On Linux you can download it via GitHub using ***Curl*** or ***Wget***. In this book we exclusively use Windows examples, but you should be able to follow along as a Linux user. So open your favourite browser and head over to 'https://www.github.com/AZProductions/Kookaburra/releases' or 'https://www.github.com/AZProductions/Kookaburra/releases/latest' to download the latest stable build. Click on the little dropdown icon and download the most suitable version for your computer.

If you don't know which version you should download, newer computers support x64. If you have an older computer or don't know if the computer is x64 based, just pick the x86 version.

Some browsers like Edge will stop you from opening the file. These popups are just warnings to tell the user to not download random files off the internet. The official Kookaburra files are safe to use. Feel free to scan it with your own Antivirus for your own piece of mind. Just select **keep** and open the file. Windows *SmartScreen* will popup, and again, this is a false positive. Click on *Read more*, and select run. Kookaburra should start up. If you're not so keen with these popups, we recommend you to download it in the **Microsoft Store**. *You can also manually build the code using **.Net**, it takes longer but then you can verify that the binaries haven't been tampered with. We won't be covering that in this book.*

1.2 Getting the basics right.

If you haven't already, start Kookaburra by double clicking the recently downloaded file named **'KookaburraShell'**. The Kookaburra CLI *(command-line interface)* will appear. Let's get you through the basics. At the top of the terminal, you can see welcome messages, errors alerts and warnings. The line where your cursor is blinking, is called the *input field*. You can start off by typing the command `help`, this opens the help dialog. In the help dialog you can view a list of possible commands with their corresponding descriptions. Press *enter* to return to the input field. Use the ***arrow key*** pointing **up** to see the last typed command, now press the ***arrow key*** pointing **down** to clear the input field. Type the command `-dt` to enter the desktop directory of you computer. You can create a file using the mkfile command. Type `mkfile test.txt`, a file will appear on the desktop. This example only uses a couple of commands, if you want to see the full list, head over to 'https://github.com/AZProductions/Kookaburra#-cli-commands'.

The second way of using Kookaburra is by coding **'.kookaburra'** files. Start off by creating a new file, by opening Notepad. *(You can use any other program to open it, this is just a simple way of doing it.)* Click on the menu header called **'File'** and select **'Save As'**. A window will appear, give the file the name **'helloworld.kookaburra'**. Change the file type to other and save the file in a folder of your choice. Now start by copy-pasting the following code.

```
print "Hello World!"
app.read()
```

Save the file *(ctrl+s)*, and open it with Kookaburra by dragging the HelloWorld file on to KookaburraShell. A window will popup with the text **'Hello World!'**. Congratulations, you've created your first Kookaburra program! Head over to the second chapter, to learn what the code does.

1.3 Customizing Kookaburra.

You can add custom commands by editing the **'custom_commands.txt'** file, which is located in the *AppData* directory. Open the Kookaburra settings directory by pressing win + R *(Windows key + R)* and typing **'%appdata%/kookaburra'**. Press *enter*, file explorer will appear with the folder open. Double click on the **'custom_commands.txt'** file. Add a new line and copy-paste the following `paint=mspaint`. In the CLI type **'paint'**, and Microsoft Paint will open.

You can also change the default text editer with the **'text_editer.txt'** file. Simply open it up and change it to `c:\program files\windows nt\accessories\wordpad.exe|%arg%` if you want to edit it in Wordpad. If you want to change it back to Notepad, replace the text in the file with `notepad.exe|%arg%`. By default on Linux it uses Nano, you can of course change it to vim or any other text/code editor. Kookaburra currently doesn't support custom start-up messages, but it will come in the upcoming release. Kookaburra's code is open-source so you can make your own 'custom version' of Kookaburra. **For a more indepth view, visit chapter 4.**

---

2. Kookaburra scripting

In chapter one we created our first Kookaburra program, in this chapter its all about scripting. Lets start!

2.1 Functions and values.

Kookaburra scripts are read line by line, it doesn't have events. Here is a list of all the functions and values currently in Kookaburra. Keep in mind that it's a fast evolving language, the syntax will be gradually expanded, so make sure to download the newest version of this book.

Print, Figlets and Colors

The most used function in Kookaburra is `print`. It's used to print text on the CLI. By default it makes a new line every time you print, you can add an `@` in front of the string to print it not as a new line. Example:

```
print "hello"
#prints the text 'hello'as a new line.

new Rule()
#creates a rule. (a spacer between the two examples.) Read about rules in
'Grids Barcharts and more.'

print @"he"
print @"llo"
#prints the text 'hello', because of the '@' it doesn't print as a new line.
```

Kookaburra supports the basic Windows Colors. It uses the colors when printing text on the screen, but also when rendering **Barcharts, Grids and Rules**. Just add these after `app.color =`

```
Black
Blue
Cyan
DarkBlue
DarkCyan
DarkGray
DarkGreen
DarkMagenta
DarkRed
DarkYellow
Gray
Green
Magenta
Red
White
Yellow
```

Maybe in the near future Kookaburra will support the longer list of colors from the **Spectre.Console** library. We are definitely looking out for a solution, it's a game breaking change from 16 to 256 colors.

Titles, windows sizes and debug messages

You can customize Kookaburra scripts, by changing the title and window size. Changing the windows size is only supported in Windows. Unfortunately you can't change the window icon at runtime, you can create a shortcut on windows with a custom icon. Just open the propeties tab in the shortcut, the icon will also be applied in the running window. In this instance it's Kookaburra. You can start changing the title by typing `app.title = "this is the title"`, it supports strings and of course binding.

2.2 Binding

One of the most useful features in Kookaburra is binding, which allows you to display realtime/user dependent text into strings. All the functions that support the *Kookaburra Format System*, support binding. There are two types of binding, **Element binding** and **String Binding**. Element binding is used for live text or generated data, which is user/pc dependent. For example, you can print 'Hello AZ!'. AZ is the username of the computer that can vary between computers. Here's an example:

```
print "Hello {environment.username}!"
# You use '{}' these brackets when doing Element binding.
```

With String binding you can put parts of strings in strings. String binding uses the **'<>'** instead of **'()'**.

```
string test = "Hello"
print "<string.test> Michael!"
#prints 'Hello Micheal!'
```

2.3 Grids, Barcharts and Rules.

One of the newest additions to Kookaburra are Grids, Barcharts and Rules. Let's start with Rules. Rules are basically spacers to separate items in the CLI. Creating a new Rule is simple, just type: `new Rule()`. You can add text in the rule, which supports the *Kookaburra Format System*. `new Rule("Hello {environment.username}!")` Barcharts are just as easy, start off by creating it with `new Barchart("Test Message")`. Then Add Rows with `Barchart.Add("test", 24)` The first value is a string, and the second is the value you give it.

```
app.debug-off
app.color = white
new Barchart("Test Message")
app.color = Green
Barchart.Add("test1", 20)
app.color = red
Barchart.Add("test2", 24)
app.color = blue
Barchart.Add("test3", 45)
Barchart.Display()
app.read()
```

Grids are also just as easy to implement. Start by typing `new Grid()` to create it. Then add columns and rows by using **AddColumn** and **AddRow** functions. Here's an example:

```
app.debug-off
app.clear
new Grid()
app.color = red
Grid.AddColumn("This is a grid!")
app.color = blue
```

```
Grid.AddColumn("I'm a column")
Grid.AddRow("You can add", "multiple rows!")
Grid.Display()
```

The last little feature we've added is the calendar. Type `new Calendar(2021, 7)` to see a little calendar. It currently doesn't have a purpose. Perhaps in the future we can add calendar items.

2.4 Dialogs and Images.

The newest version of Kookaburra includes Dialogs and Images, which are easy to setup. Let's start with images, with images you can render normal **.png** or **.jp**(e)**g** files into the console window. Type `new image("file.png", "12")`, the first argument is the file location and the second argument is a number which desides the maxmimum width of the image in the terminal. You can use dialogs to simplify user input. Here's an example:

```
string example = dialog.yesno("Is four plus four eight?")
# prints True or False depending on the user input.
print example
```

Here's a list with all dialogs:

```
# You can input (Y)es or (N)o
string yesno = dialog.yesno("Do you want to update?")
# Numbers only
string numbers = dialog.numerical("How many apples do you want?")
# Colors only
string colors = dialog.color("What is your favorite color?")
# Just like 'app.read()' but the input is hidden with * characters, just like
a login form.
string password = dialog.secret("What is the password?")
```

2.5 Extended importing features.

With Kookaburra you can also import libraries pre-made by **us** and **contributors**. Currently we have a short list of supported libraries. You can create files using `import FileIO`. Type `new filewriter(location, value)`. The import is necessary. The second library `import net` is currently under development and not yet fully functional. We've only implemented local ip to Element binding. Example: `print "your local ip is: {Net.IP}!"`.

---

3. Distributing Kookaburra scripts

This is an advanced chapter, not essential for beginners to understand.

3.1 Idea

To convert source code into machine code, there are 2 options. An **Interpreter** translates just one statement of the program at a time into machine code. A **Compiler** scans the entire program and translates the whole of it into machine code at once. Kookaburra is an Interpreter. To distribute it, you need to have both **Kookaburrashell** and the **.kookaburra** file. This is a hassle when deploying your program. To prevent this, we can bundle the files into a single exe file. We've made a short blog about it.

3.2 Coding

To work around the issue of publishing Kookaburra, we can bundle both the file and the **Kookaburra** framework. When running the program, it will extract both Kookaburra and the file into a folder, which is included in the **exe**, and run Kookaburra with the script. In this tutorial we are going to use **Visual Studio 2019** and **.Net 5**. Start off by creating a new Console Application. Then add the KookaburraShell.exe and script in resources. Remember to set ***Copy to output directory*** to **Copy Always**. Then copy-paste the following code.

```
string path =
System.Reflection.Assembly.GetExecutingAssembly().Location.Replace("TestApp.d
ll", "");
try
{
    Directory.CreateDirectory(path + "TestApp/");
    File.Copy(@"Resources\KookaburraShell.exe", path + "TestApp/src.exe");
    File.Copy(@"Resources\TestApp.kookaburra", path +
"TestApp/TestApp.kookaburra");
}
catch { }
ProcessStartInfo startInfo = new ProcessStartInfo();
startInfo.FileName = path + "TestApp/src.exe";
startInfo.Arguments = path + "TestApp/TestApp.kookaburra";
Process.Start(startInfo);
```

You can replace **TestApp** with the name of your program. Using this method you can manipulate Kookaburra and add Icons and more. Publish the app by clicking Publish in the ***Solution Explorer***. Now you can distribute your favourite program to whoever you want. There many other methods of doing this, you can experiment with different frameworks and programming languages.

---

4. Customizing Kookaburra

Since version 0.7.5 you can customize Kookaburra using the **conf.txt**. Head over to the Kookaburra config directory. These are the default values for the conf.txt file.

```
# KookaburraShell - Conf.txt
show_startup=true
startup_location=default
FR=false
force_rainbow=false
```

All lines starting with a # are comments. `show_startup` determines if Kookaburra shows the startup text. `startup_location` sets the default location when starting kookaburra, this needs to be a valid path. Use `default` to open Kookaburra in the current user directory. `FR` is used to check if Kookaburra is running for the first time. `force_rainbow`, forces the input bar to be displayed in rainbow colors.

---

5. Mastering the CLI

You can speed up your workflow by mastering the CLI. Here's a cheat sheet.

5.1 Command list.

Command	description
`cp`	Copy files from one place to another.
`rm`	Delete files.
`rmdir`	Delete folders.
`mkdir`	Make folders.
`mkfile`	Make files.
`cd`	Change to a directory or file.
`cd ..`	Go back a directory.
`ls` or `dir` or `directory`	Displays all the files and folders in the current directory.
`clear` or `cls`	Clears the console window.
`whoami`	Shows the pc name and username.
`drives`	List all drives in your computer.
`browse`	Opens file-explorer in the current directory. You can also use `explore` or `explorer`.
`ipconfig`	Show all internet settings from your computer.
`download`	Download files from an internet server.
`password`	Generate passwords.
`tree`	Renders a detailed list of all the files and folders in a directory.
`winreset`	Resets the window height and width. (Windows only)
`hash`	Get's the MD5, SHA256 and SHA1 from the specified file. (`hash test.txt`)

`tos`	Opens the Terms of Service.
`exit`	Exits the console.
`env`	Gets the Env data and writes it to the screen. *(For testing purposes)*
`restart`	Restarts the console.
`restart -p`	Restarts the console without doing Package Check.
`kbconfig` or `settings`	Open the settings file in the default text editor.
`drives`	Displays all active drives with additional info.
`explorer` or `explore` or `browse` or `view`	Opens file explorer in the current directory.
`mv`	Moves the selected directory or file to the specified directory or file.
`mv -o`	Moves the selected directory or file to the specified directory or file overriding it.
`ipconfig`	Shows the current internet drives, chipsets and local ip information.
`start`	Starts the specified file.
`sound.play`	Plays specified **.wav** file(s).
`download`	Downloads specified file from an internet location.
`tip`	Shows a random tip.
`whoami`	Displays the MachineName + UserName.
`edit`	Opens the specified file in the default text editor. *(text_editor.txt)*
`pwd`	Prints the working directory.
`uname`	Prints the username.
`mname`	Prints the machinename.
`tree` or `list`	Displays the current folder/file structure in a detailed tree.

5.2 Shortcut list.

Shortcut	Description
`-c:/`, it works with all drives.	Automatically gets the root of the set directory.
`-env`	The location of the configuration directory of kookaburra.
`-r`	Goes to the root of the current directory.
`-cookies`	The directory that serves as a common repository for Internet cookies.
`-desktop` or `-dt`	The logical Desktop rather than the physical file system location.

`-favorites`	The directory that serves as a common repository for the user's favorite items.
`-fonts`	A virtual folder that contains fonts.
`-history`	The directory that serves as a common repository for Internet history items.
`-personal`	The directory that serves as a common repository for documents. This member is equivalent to MyDocuments.
`-programs`	The directory that contains the user's program groups.
`-recent`	The directory that contains the user's most recently used documents.
`-resources`	The file system directory that contains resource data.
`-st`	The directory that contains the Start menu items.
`-startup`	The directory that corresponds to the user's Startup program group. The system starts these programs whenever a user logs on or starts Windows.
`-system`	The System directory.
`-templates`	The directory that serves as a common repository for document templates.
`-windows`	The Windows directory or SYSROOT. This corresponds to the %windir% or %SYSTEMROOT% environment variables.
`-c`	Clears location.
`AdminTools`	The file system directory that is used to store administrative tools for an individual user. The Microsoft Management Console (MMC) will save customized consoles to this directory, and it will roam with the user.
`ApplicationData`	The directory that serves as a common repository for application-specific data for the current roaming user. A roaming user works on more than one computer on a network. A roaming user's profile is kept on a server on the network and is loaded onto a system when the user logs on.
`CDBurning`	The file system directory that acts as a staging area for files waiting to be written to a CD.
`CommonAdminTools`	The file system directory that contains administrative tools for all users of the computer.
`CommonApplicationData`	The directory that serves as a common repository for application-specific data that is used by all users.
`CommonDesktopDirectory`	The file system directory that contains files and folders that appear on the desktop for all users.
`CommonDocuments`	The file system directory that contains documents that are common to all users.

`CommonMusic`	The file system directory that serves as a repository for music files common to all users.
`CommonOemLinks`	This value is recognized in Windows Vista for backward compatibility, but the special folder itself is no longer used.
`CommonPictures`	The file system directory that serves as a repository for image files common to all users.
`CommonProgramFiles`	The directory for components that are shared across applications.
`CommonProgramFilesX86`	The Program Files folder.
`CommonPrograms`	A folder for components that are shared across applications.
`CommonStartMenu`	The file system directory that contains the programs and folders that appear on the Start menu for all users.
`CommonStartup`	The file system directory that contains the programs that appear in the Startup folder for all users.
`CommonTemplates`	The file system directory that contains the templates that are available to all users.
`CommonVideos`	The file system directory that serves as a repository for video files common to all users.
`DesktopDirectory`	The directory used to physically store file objects on the desktop. Do not confuse this directory with the desktop folder itself, which is a virtual folder.
`InternetCache`	The directory that serves as a common repository for temporary Internet files.
`LocalApplicationData`	The directory that serves as a common repository for application-specific data that is used by the current, non-roaming user.
`LocalizedResources`	The file system directory that contains localized resource data.
`MyComputer`	The My Computer folder. When passed to the Environment.GetFolderPath method, the MyComputer enumeration member always yields the empty string ("") because no path is defined for the My Computer folder.
`MyDocuments`	The My Documents folder. This member is equivalent to Personal.
`MyMusic`	The My Music folder.
`MyPictures`	The My Pictures folder.
`MyVideos`	The file system directory that serves as a repository for videos that belong to a user.
`NetworkShortcuts`	A file system directory that contains the link objects that may exist in the My Network Places virtual folder.
`PrinterShortcuts`	The file system directory that contains the link objects that can

	exist in the Printers virtual folder.
`ProgramFiles`	The program files directory.
`ProgramFilesX86`	The x86 Program Files folder.
`SendTo`	The directory that contains the Send To menu items.
`StartMenu`	The directory that contains the Start menu items.
`SystemX86`	The Windows System folder.
`UserProfile`	The user's profile folder. Applications should not create files or folders at this level; they should put their data under the locations referred to by ApplicationData.

***Most of the Shortcuts in this list are from the Microsoft docs.**

6. The End

6.1 Good luck

Congrats, you've reached the end of this e-book. You're officialy a ***Kookaburra Pro***! Thank you for checking it out, were open for suggestions, let us know via the Issues section in Github https://github.com/AZProductions/Kookaburra/issues/new/choose. Good luck on your coding journey! **Happy coding!**

6.2 Special mentions

- Special thanks to the `spectre.console` library, it speeds up the process of making Kookaburra by a ton!
- We've enjoyed the tools supplied from the teams over at **Github** and **Microsoft**.