OSU Capstone Project SDK

OSU Capstone Project SDK for the 2021-2022 school year

Note: See new section on Debian 5/24/2022

Introduction

The *pdfpagesapp* is a CPP application that will render individual pdf pages into tiff's, png's, or jpeg's. In addition
to the application, a static library is also supplied named *libpdf_library.a*. Both the application and the library
were built using the C++ 20 standard.

Archive Contents

The table below lists the files and their description that are provided in the OSU_capstone_mm-dd-yyyy.tgz archive file.

To untar the file, use the following command:

```
tar -xzvf OSU_capstone_mm-dd-yyyy.tgz
```

File	Description		
cpp-env-debian.Dockerfile	Dockerfile for Debian Bullseye. See: [Dockerfile for Debian Bullseye development container] (#Dockerfile for Debian Bullseye development container)	libpdf_library.a	Static library which renders PDF pages to files.
log4cpp.cfg	Log4cpp config file. Must be put in same directory as pdfpagesapp.		
main.cpp	Example main method showing how to call the library. This is the pdfpagesapp main method.		
Mako-brochure.pdf	Test PDF file.		
Mako-brochure_output.json	Sample JSON output from the tool. This is always produced when you run the tool.		
	Sample JSON input that is passed to the		

Mako-brochure_input.json	RenderPageProcessor. Note: You can also see this in the output when you run the tool.
PageProcessor.h	Example base class for "processors". Embraces decorator pattern.
pdfpagesapp	Debian - render PDF pages to file application.
pdfpagesapp_centos	!COMING SOON! Centos - Render PDF pages to file application.
README.pdf	PDF version of the README file.
README.md	Mark-down version of the README file.
RenderPageProcessor.h	Example implementation class of the PageProcessor class (see PageProcessor.h). This is the real file I am using to render pdf pages to files.

Run the Application

To run the application, on a Centos7 OS, simply run from the command-line:

```
./pdfpagesapp --source path/to/pdf/file.pdf
```

Note: make sure you place the log4cpp.cfg in the same folder or the application will error out.

This will render all the pages to the current folder using defaults for optional parameters. If the file has 2 pages,
you will see the following files in the current folder:

```
ls
file_CMYK_1.tif
file_CMYK_2.tif
file.json
```

The *file.json* file contains JSON results which at HP we pass back to a web service via REST.

Alternately, if you choose to render as RGB and PNG, you will see:

```
ls
file_RGB_1.png
file_RGB_2.png
file.json
```

To see the optional arguments, simply call the app like the following and you will see the usage info:

```
./pdfpagesapp --help

Running the app...
Usage: ./pdfpagesapp <option(s)>

Options:
    -h,--help       Show this help message
    --source SOURCE     (Required) Specify the source PDF to render.
    --destination DEST  (Optional) Specify the destination folder where to
output the rendered pages. Default is current folder.
    --dpi DPI           (Optional) Specify the dpi to render at. Typically 300
or 600. Default is 300.
    --depth DEPTH       (Optional) Specify the bit depth to render at. Valid
options: 8 or 16 (bit). Default is 8.
    --format FORMAT     (Optional) Specify the output format. Valid options:
tiff, png, jpeg, or all. Default is tiff.
    --color COLOR       (Optional) Specify the color space to render. Valid
options: Gray, CMYK, CMY, or RGB. Default is CMYK.
```

Troubleshooting

I suspect you'll need to install additional libraries on your Centos7 system to make this app work. These should be easy
to acquire using yum. Here's the libraries being used as reported on my system:

```
ldd ./pdfpagesapp
linux-vdso.so.1 =>  (0x00007fff27d9d000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007f9680956000)
libuuid.so.1 => /lib64/libuuid.so.1 (0x00007f9680751000)
libssl.so.10 => /lib64/libssl.so.10 (0x00007f96804df000)
libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007f968007c000)
libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f967fd74000)
libm.so.6 => /lib64/libm.so.6 (0x00007f967fa72000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f967f85c000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f967f640000)
libc.so.6 => /lib64/libc.so.6 (0x00007f967f272000)
/lib64/ld-linux-x86-64.so.2 (0x00007f9680b5a000)
libgssapi_krb5.so.2 => /lib64/libgssapi_krb5.so.2 (0x00007f967f025000)
libkrb5.so.3 => /lib64/libkrb5.so.3 (0x00007f967ed3c000)
libcom_err.so.2 => /lib64/libcom_err.so.2 (0x00007f967eb38000)
libk5crypto.so.3 => /lib64/libk5crypto.so.3 (0x00007f967e905000)
libz.so.1 => /lib64/libz.so.1 (0x00007f967e6ef000)
libkrb5support.so.0 => /lib64/libkrb5support.so.0 (0x00007f967e4df000)
libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007f967e2db000)
libresolv.so.2 => /lib64/libresolv.so.2 (0x00007f967e0c1000)
libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f967de9a000)
libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f967dc38000)
```

Using the Static Library

To use the libpdf_library.a, refer to the main.cpp file supplied which shows you how to run it. I recommend using cmake
so in the CMakeLists.txt file, you would add it in like:

```
target_link_libraries(my_app libpdf_library.a)
```

Make sure you supply the log4cpp.cfg file in the running folder.

Dockerfile for Debian Bullseye development container

Here's how to build the Debian container used to build the app.

```
docker build -t clion/remote-cpp-env:1.5 -f src/Docker/cpp-env-
debian.Dockerfile .
docker run -d --cap-add sys_ptrace -p127.0.0.1:2222:22 --name clion_remote_env
clion/remote-cpp-env:1.5
```

Contact Info

If you have any issues, which I suspect you might, don't hesitate to email me at adam.bernosky@hp.com