Capgemini AS France
Telecom & Média
ALM
Tour Europlaza
20, Avenue André Prothin
92927 La Défense
FRANCE

Tél. : +33(0)1 49 00 40 00
Fax : +33(0)1 47 78 45 52

HOW TO INSTALL DEV.TESSERACT C++ TO JS?

Référence :	CNED_ADAPT /0019
Version :	1.0
Date :	22/01/2016
Statut :	En cours
Usage :	Interne
Auteur :	Zidane EL MRANI
Type :	SFS : Spécification fonctionnelle du système

APPROBATION DU DOCUMENT			
Organisme ou entreprise	**Nom (fonction)**	**Date**	**Visa**
Capgemini France			

DIFFUSION				
Destinataire	**Organisme ou entreprise**	**Nombre**	**Pour action**	**Pour info**
Equipe projet	CNED			

MISES A JOUR			
Version	**Date**	**Auteur**	**Motifs**
1.0	22/01/2016	Zidane EL MRANI	Création

DEPOT	
URL	**Administrateur**

Table of contents

1. INTRODUCTION 5
2. REQUIREMENTS: 6
2.1 TESSERACT SOURCE FILES 6
2.2 GIT 6
2.3 PORTABLE EMSCRIPTEN 7
2.4 CYGWIN (YOU MAY NEED IT FOR COMPILATION IN WINDOWS) 8
2.5 MICROSOFT VISUAL STUDIO (PREFERABLY 2011, OTHERWISE YOU WILL NEED TO UPDATE THE SOURCES) 9
3. COMPILING THE LIBRARIES 10
3.1 WRITE THE FILE "COMPILE.VAR" 10
3.2 COMPILE THE LIBRARIES USING GCC 12
3.3 COMPILE THE LIBRARIES USING EMCC (EMSCRIPTEN) 12
3.4 WRITE THE BATCH FILE TO COMPILE JPEG, PNG, TIFF AND ZLIB 12
3.5 GENERATE THE GLOBAL MAKEFILE USING GCC 13
3.6 GENERATE THE GLOBAL MAKEFILE USING EMCC 15
4. COMPILE TESSERACT C/C++ 17

1. INTRODUCTION

The national distance learning center offers its users a text editor to create documents that will be adapted to the user's preferences within a project named "CNED Adapt".

This project is a web platform set to provide documents for users who need specific adaptation and modifications for a better visibility and readability. Typical users include people suffering from dyslexia for example.

Users can add already existing documents, either in HTML page format, PDF or even more EPUB (eBooks). The documents will be displayed in accordance with the display rules inherent to the chosen profile settings.

This document is a step-by-step guide aimed at software developers to get them started with Tesseract, a library used in this project. It represents a guideline to how to install Tesseract and compile it using EMSCRIPTEN from C/C++ to JavaScript.

2. REQUIREMENTS:

Here is a list of all the software and files you will need in order to compile TESSERACT:

2.1 TESSERACT SOURCE FILES

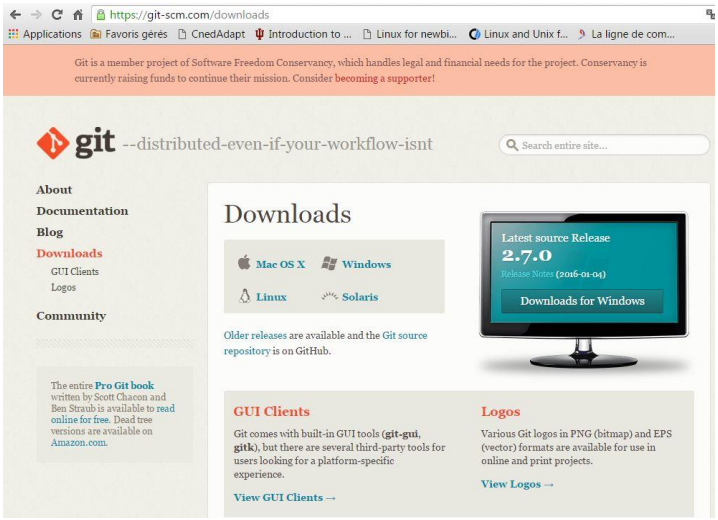
TESSERACT C/C++ sources files can be obtained from the links listed in the table below.

Copy in your machine the TESSERACT folder corresponding to the version you are looking for: Each version contains the C/C++ sources **AND** a portable EMSCRIPTEN.

Library	*Link*
Giflib-5.1.1	http://downloads.sourceforge.net/giflib/giflib-5.1.1.tar.bz2 or https://github.com/pcwalton/giflib/archive/master.zip
Jpeg_9	https://github.com/winlibs/libjpeg/archive/master.zip
Leptonica-1.70	http://www.leptonica.com/source/leptonica-1.70.tar.gz
Leptonica-1.71	http://www.leptonica.com/source/leptonica-1.71.tar.gz
Leptonica-1.72	http://www.leptonica.com/source/leptonica-1.72.tar.gz
Libmng-2.0.3	http://downloads.sourceforge.net/libmng/libmng-2.0.3.tar.xz
Libpng-1.6.17	http://sourceforge.net/projects/libpng/files/libpng16/older-releases/1.6.17/libpng-1.6.17.tar.gz/download
Libwebp-0.4.3	http://pkgs.fedoraproject.org/repo/pkgs/libwebp/libwebp-0.4.3.tar.gz/08813525eeeffe7e305b4cbfade8ae9b/libwebp-0.4.3.tar.gz
Openjpeg-2.1	http://sourceforge.net/projects/openjpeg.mirror/files/2.1.0/openjpeg-2.1.0.tar.gz/download or https://github.com/uclouvain/openjpeg/releases/tag/version.2.1
Tesseract	https://github.com/tesseract-ocr/tesseract/archive/master.zip
Tiff-4.0.3	http://pkgs.fedoraproject.org/repo/pkgs/libtiff/tiff-4.0.3.tar.gz/051c1068e6a0627f461948c365290410/tiff-4.0.3.tar.gz
Zlib-1.2.8	http://prdownloads.sourceforge.net/libpng/zlib-1.2.8.tar.gz?download or https://github.com/madler/zlib/archive/master.zip

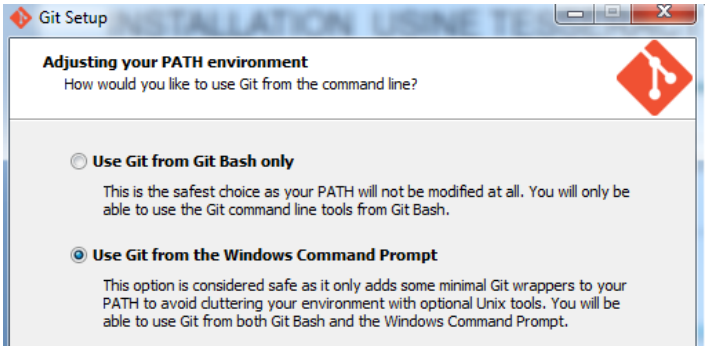
2.2 GIT

GIT can be downloaded from the following link: https://git-scm.com/downloads



Choose the version you need bases on your OS.

If you are using Windows, when installing GIT, choose: **"Use GIT from the Windows Command Prompt"**



2.3 PORTABLE EMSCRIPTEN

A portable version of EMSCRIPTEN is normally contained in the folder with TESSERACT C/C++ source files. If it is not the case, download it from the following link: https://s3.amazonaws.com/mozilla-games/emscripten/releases/emsdk-1.30.0-full-64bit.exe this link will download the 1.30.0 version of EMSCRIPTEN; you just need to launch the setup application afterwards.

If you want to use the latest version, use this link: https://s3.amazonaws.com/mozilla-games/emscripten/releases/emsdk-1.35.0-portable-64bit.zip. In this case, you will need to install EMSCRIPTEN by yourself using the command prompt. Go to the folder containing the portable EMSCRIPTEN, and then use the command **/emcmdprompt.bat** to set the environment.

Go to EMSDK's directory, and then list all the installed components using the following command: **./emsdk list**

You should get the following window under Windows:

```
D:\Emscripten\TESSERACT\SA_EMSCRIPTEN>emsdk list

The following precompiled tool packages are available for download:
     *      clang-e1.30.0-64bit          INSTALLED
            clang-e1.34.1-64bit
            node-0.12.2-32bit
     *      node-0.12.2-64bit            INSTALLED
            python-2.7.5.3-32bit
     *      python-2.7.5.3-64bit         INSTALLED
            java-7.45-32bit
     *      java-7.45-64bit              INSTALLED
            spidermonkey-37.0.1-64bit
            spidermonkey-nightly-2015-04-12-64bit
            git-1.9.4
     *      emscripten-1.30.0            INSTALLED
            emscripten-1.34.1
     *      vs-tool-0.9.4                INSTALLED
            crunch-1.03
            mingw-4.6.2-32bit

The following tools can be compiled from source:
            clang-tag-e1.34.3-32bit
            clang-tag-e1.34.4-32bit
            clang-tag-e1.34.3-64bit
            clang-tag-e1.34.4-64bit
            clang-incoming-32bit
            clang-incoming-64bit
            clang-master-32bit
            clang-master-64bit
            emscripten-tag-1.34.3-32bit
            emscripten-tag-1.34.4-32bit
            emscripten-tag-1.34.3-64bit
            emscripten-tag-1.34.4-64bit
            emscripten-incoming-32bit
            emscripten-master-32bit
            emscripten-incoming-64bit
            emscripten-master-64bit

The following precompiled SDKs are available for download:
    *       sdk-1.30.0-64bit             INSTALLED
            sdk-1.34.1-64bit

The following SDKs can be compiled from source:
            sdk-incoming-32bit
            sdk-incoming-64bit
            sdk-master-32bit
            sdk-master-64bit
            sdk-tag-1.34.3-32bit
            sdk-tag-1.34.4-32bit
            sdk-tag-1.34.3-64bit
            sdk-tag-1.34.4-64bit

Items marked with * are activated for the current user.

To access the historical archived versions, type 'emsdk list --old'

D:\Emscripten\TESSERACT\SA_EMSCRIPTEN>_
```

To activate EMSCRIPTEN -in case it is not-, use the following command under Windows: **./emsdk activate**

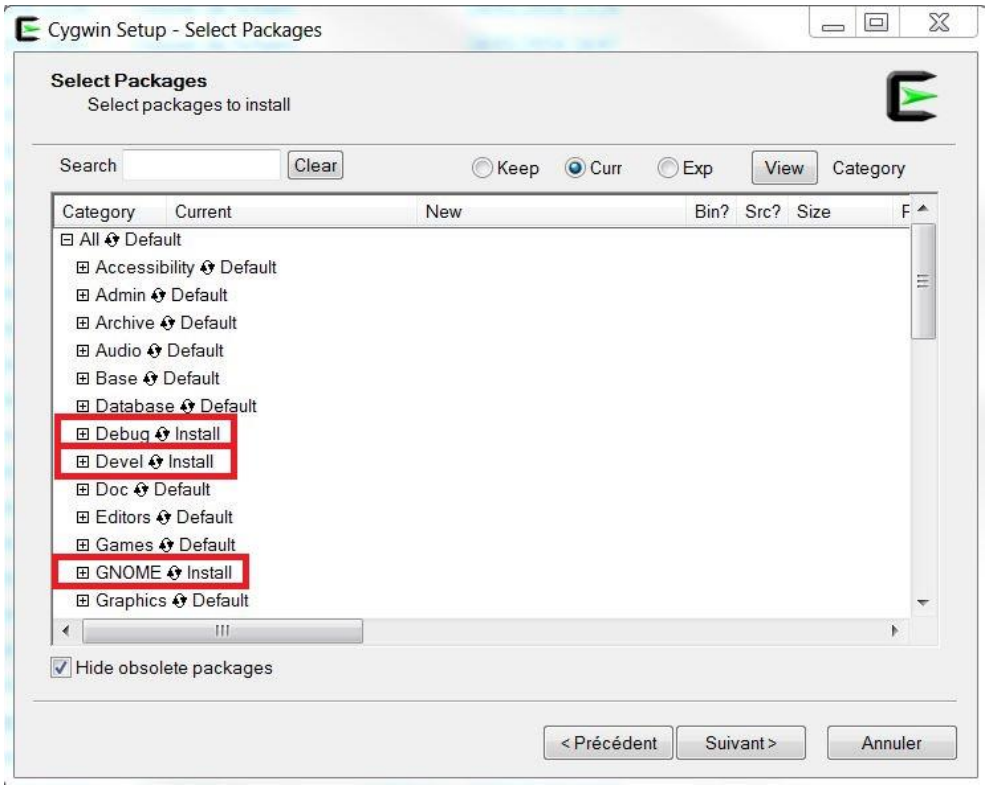
!! Be careful not to use the command "**./emsdk activate latest**" as described in EMSCRIPTEN's documentation. Make sure the 1.30.0 version is the one activated as there might be some issues with the most recent versions.

For more details, check this website: https://kripken.github.io/emscripten-site/docs/getting_started/downloads.html#installation-instructions

2.4 CYGWIN (you may need it for compilation in Windows)

To download CYGWIN, go to https://cygwin.com/install.html and choose the executable file based on your OS version.

Make sure to install these packages: **Devel, Debug and GNOME**.



2.5 MICROSOFT VISUAL STUDIO *(PREFERABLY 2011, OTHERWISE YOU WILL NEED TO UPDATE THE SOURCES)*

You can install Microsoft Visual Studio from Microsoft website for the latest version: https://www.microsoft.com/france/visual-studio/essayez/telecharger/visual-studio.aspx#telechargezVS

If you wish to install Microsoft visual studio 2011, use this link:https://www.microsoft.com/fr-fr/download/details.aspx?id=26830

3. COMPILING THE LIBRARIES

Here is a list of all the libraries required:

	LIB_GIF_5_1_1	
	LIB_JPEG_9	UT_JPEG
	LIB_LEPTONICA_1_70	UT_LEPTONICA
	LIB_LEPTONICA_1_71	
	LIB_LEPTONICA_1_72	
	LIB_MNG_2_0_3	
	LIB_OPENJPEG_2_1	
	LIB_PNG_1_6_17	UT_PNG
DLL_TESSERACT	LIB_TESSERACT	UT_TESSERACT
	LIB_TIFF_4_0_3	UT_TIFFLIB
	LIB_ WEBP_ENC_0_4_3	
	LIB_WEBP_0_4_3	
DLL_ZLIB_1_2_8	LIB_ZLIB_1_2_8	UT_ZLIB

The libraries highlighted in green will not be used when compiling with 'emcc'.

!! Each library must be compiled separately

DLL=Dynamic Link Library
UT=Unit Test
LIB= Normal library

3.1 WRITE THE FILE "COMPILE.VAR"

This file will be included in the 'makefile' files during the compilation whether EMSCRIPTEN or GCC is used. Hence, the need to write two different files: one per compiler.

The table below shows an example: The file should contain at least the compiler's name, the default flags and the directories to the different files that will be used,

GCC

```
################################
# ENVIRONEMENT VARIABLE
################################

COMPILER_NAME = gcc
# CROSS_COMPILE = mips-linux-
# MACHINE = I86
# ARCH = x86_64
CPP = $(CROSS_COMPILE)g++
CXX = $(CROSS_COMPILE)g++
CC   = $(CROSS_COMPILE)gcc
LINKER = $(CROSS_COMPILE)g++
RANLIB = $(CROSS_COMPILE)ranlib
AR_EXE = $(CROSS_COMPILE)ar r

# STAGING_DIR = /var/lib
UNAME = uname -r
KERNEL_DIR                                    =
${STAGING_DIR}/lib/modules/$UNAME/build
VERSION = (1.0)
RELEASE_DIR = /var/lib
ARC_EXE = tgz
WINDRES = res
```

EMCC

```
################################
# ENVIRONEMENT VARIABLE
################################

COMPILER_NAME = cygwin
# CROSS_COMPILE = mips-linux-
# MACHINE = I86
# ARCH = x86_64

#CPP = $(CROSS_COMPILE)em++
#CXX = $(CROSS_COMPILE)em++
#CC   = $(CROSS_COMPILE)emcc
#LINKER = $(CROSS_COMPILE)em++
#RANLIB = $(CROSS_COMPILE)emcc
#AR_EXE = $(CROSS_COMPILE)emcc

CPP          =          $(CROSS_COMPILE)python
"D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1
.30.0\em++"
CXX          =          $(CROSS_COMPILE)python
"D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1
.30.0\em++"
CC               =      $(CROSS_COMPILE)python
"D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1
```

```
RM = rm -f
MDIR = mkdir -p
RMDIR= rmdir
COPY= cp
INCS =  -I"../../../"
CXXINCS =  -I"../../../"

#Some flags that are used for cross compilation
TARGET_32BITS = -m32
TARGET_64BITS = -m64
TARGET_INTEL8086 =

#These are the default flags
DEFLIBS = $(CXXINCS) -O3 -W -fexceptions -pthread   -Wl,-rpath,/usr/local/lib   -Wl,-rpath,./ -pthread -lrt -lpthread -ldl
DEFAUT_CXXFLAGS  =  $(CXXINCS)  -O3  -W  -fexceptions -pthread
DEFAUT_CFLAGS = $(INCS) -O3 -W -pthread
DEFAUT_CXXFLAGS_DBG = -O0 -g -pg -gstabs+ -DDEBUG -W -pthread
DEFAUT_CFLAGS_DBG = -O0 -g -DDEBUG -Wall -pthread
DEFAULT_UNICODE_FLAGS =

#some defines for standardization and quick change
DEBUG_EXT = d
UNICODE_EXT = u
DUNICODE_EXT = ud
LIB_EXT = .a
DLL_EXT = .so
EXE_EXT =
PLUGIN_EXT = .plug

#Allow us to add some switch in the makefiles depending on the OS
ifneq ($(strip $(shell gcc -v 2>&1 |grep "cygwin")),)
      CYGWIN = true
endif
ifneq ($(strip $(shell gcc -v 2>&1 |grep "darwin")),)
      MACOSX = true
endif
```

```
.30.0\emcc"
LINKER        =        $(CROSS_COMPILE)python "D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1.30.0\em++"
RANLIB        =        $(CROSS_COMPILE)python "D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1.30.0\emcc"
AR_EXE        =        $(CROSS_COMPILE)python "D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1.30.0\emcc"

# STAGING_DIR = /var/lib
UNAME = uname -r
KERNEL_DIR                                    = ${STAGING_DIR}/lib/modules/$UNAME/build
VERSION = (1.0)
RELEASE_DIR = /var/lib
ARC_EXE = tgz
WINDRES = res
RM = rm -f
MDIR = mkdir
RMDIR= del /F /S /Q
COPY= copy
INCS =  -I"../../../"
CXXINCS =  -I"../../../"

#Some flags that are used for cross compilation
TARGET_32BITS =
TARGET_64BITS =
TARGET_INTEL8086 =

#These are the default flags
DEFLIBS  =  $(CXXINCS)  -W  -fexceptions  -L/usr/lib/e2fsprogs -luuid -lpthread -ldl
DEFAUT_CXXFLAGS = $(CXXINCS) -W -fexceptions -Os
DEFAUT_CFLAGS = $(INCS) -W -Os
DEFAUT_CXXFLAGS_DBG = -gstabs+
DEFAUT_CFLAGS_DBG = -gstabs+
DEFAULT_UNICODE_FLAGS =
DEFAULT_LINKER_FLAGS = -Os

#some defines for standardization and quick change
DEBUG_EXT = d
UNICODE_EXT = u
DUNICODE_EXT = ud
LIB_EXT = .bc
DLL_EXT = .so
EXE_EXT = .html
PLUGIN_EXT = .plug

#Allow us to add some switch in the makefiles depending on the OS
ifneq ($(strip $(shell gcc -v 2>&1 |grep
```

	"cygwin")),) CYGWIN = true endif ifneq ($(strip $(shell gcc -v 2>&1 |grep "darwin")),) MACOSX = true endif

3.2 COMPILE THE LIBRARIES USING GCC

After completing the previous task, it is time to compile each of the libraries.

First, 'makefile' files need to be written / generated. To do so, use **GENMAKE**.

(Cf: https://users.cs.duke.edu/~chase/genmake.html and http://www.robertnz.net/genmake.htm)

For each library, you should have 4 different 'makefile' files at least, corresponding to your OS version (Check if your OS is a 32bit or 64bit version, and adapt the files)

The table below summarizes the content of each file (in particular, the main flags used)

Extension Example	Meaning	DNDEBUG	D_DEBUG	D_CONSOLE (UT) D_USRDLL (DLL) D_LIB (LIB)	D_UNICODE	DUNICODE	Default Flags	
							CXX and C	UNICODE
LIB_JPEG_9.mak	*basic*	X		X				
LIB_JPEG_9d.mak	*debug*		X	X			X	
LIB_JPEG_9u.mak	*unicode*	X		X	X	X		X
LIB_JPEG_9ud.mak	*dunicode*		X	X	X	X	X	X

Here is the meaning of each flag:

Flag	Meaning
DNDEBUG	Turn off asserts as mandated by the C standards
D_DEBUG	Turn on asserts
D_CONSOLE	Used for UT files
D_USRDLL	The predefined processor is the user DLL (Dynamic Link library)
D_LIB	Used when calling the normal library (LIB)
D_UNICODE	Used by Windows headers
DUNICODE	Used by C-runtime / MFC(Microsoft foundation class) headers

3.3 COMPILE THE LIBRARIES USING EMCC *(EMSCRIPTEN)*

The same steps listed above must be completed when compiling with EMSCRIPTEN.

The 'makefile' files will be generated using GENMAKE, but this time we will have only one file in output (the basic file).

3.4 WRITE THE BATCH FILE TO COMPILE JPEG, PNG, TIFF AND ZLIB

As Windows is used, a batch file containing the commands to compile each of the following libraries {TIFF, ZLIB, JPEG and MNG} should be written.

Each of these files contains:

- The chosen compiler: in our case emcc (i.e. EMSCRIPTEN)
- The flags: typically [-o2 –DNDEBUG –D_LIB]
- The files to include: all the libraries specified above
- The path to both the input (library's directory) and the output (a newly created folder)

- The output files

When the compilation is done, you should obtain the linked bitcode (*.bc) generated by the 'make'.

Below, a sample of the batch file:

```
mkdir out
set CC_COMPILER=python "D:\DEV\TESSERACT\SA_EMSCRIPTEN\emscripten\1.30.0\emcc"
set CC_FLAGS=-O2 -DNDEBUG -D_LIB
set CC_INCLUDES=-I ./ -I ../ -I ../libpng-1.6.17/ -I ../jpeg-9/ -I ../zlib-1.2.8/
set PATH_TO_PNG=../libpng-1.6.17/
set PATH_TO_OUT=out/

%CC_COMPILER% -c %PATH_TO_PNG%png.c -o %PATH_TO_OUT%png.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngset.c -o %PATH_TO_OUT%pngset.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngget.c -o %PATH_TO_OUT%pngget.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngread.c -o %PATH_TO_OUT%pngread.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngpread.c -o %PATH_TO_OUT%pngpread.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngrtran.c -o %PATH_TO_OUT%pngrtran.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngrutil.c -o %PATH_TO_OUT%pngrutil.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngerror.c -o %PATH_TO_OUT%pngerror.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngmem.c -o %PATH_TO_OUT%pngmem.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngrio.c -o %PATH_TO_OUT%pngrio.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngwio.c -o %PATH_TO_OUT%pngwio.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngtrans.c -o %PATH_TO_OUT%pngtrans.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngwrite.c -o %PATH_TO_OUT%pngwrite.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngwtran.c -o %PATH_TO_OUT%pngwtran.o %CC_FLAGS% %CC_INCLUDES%
%CC_COMPILER% -c %PATH_TO_PNG%pngwutil.c -o %PATH_TO_OUT%pngwutil.o %CC_FLAGS% %CC_INCLUDES%

set ALL_OBJ=%PATH_TO_OUT%png.o %PATH_TO_OUT%pngset.o %PATH_TO_OUT%pngget.o %PATH_TO_OUT%pngread.o
%PATH_TO_OUT%pngpread.o %PATH_TO_OUT%pngrtran.o %PATH_TO_OUT%pngrutil.o %PATH_TO_OUT%pngerror.o
%PATH_TO_OUT%pngmem.o %PATH_TO_OUT%pngrio.o %PATH_TO_OUT%pngwio.o %PATH_TO_OUT%pngtrans.o
%PATH_TO_OUT%pngwrite.o %PATH_TO_OUT%pngwtran.o %PATH_TO_OUT%pngwutil.o
rem echo %ALL_OBJ%
%CC_COMPILER% %ALL_OBJ% -o out/LIBPNG_1_6_17.bc %CC_INCLUDES%

%CC_COMPILER% -c ../libpng-1.6.17/pngtest.c -o out/pngtest.o %CC_FLAGS% %CC_INCLUDES%

%CC_COMPILER% out/LIBPNG_1_6_17.bc out/ZLIB_1_2_8.bc out/LIBJPEG_9.bc out/pngtest.o -o UT_PNG.html
%CC_FLAGS% %CC_INCLUDES% --preload-file pngtest.png
```

3.5 GENERATE THE GLOBAL MAKEFILE USING GCC

Once all the previous steps are completed, the global makefile using 'gcc' compiler should be generated using GENMAKE. It should be named **CTBS_2D.gcc.mak.**

The generated file will contain the path to the project, the different extensions used and several commands.

Below, a sample:

```
#######################################
#    WORKSPACE MAKEFILE BY GENMAKE    #
#       INTENDED FOR GNU MAKE         #
#######################################
PATH_TO_PROJECT = gcc
DEBUG_EXT = d
UNICODE_EXT = u
DUNICODE_EXT = ud
.PHONY: all all-before all-after clean clean-custom depend dist

all :
	make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8.mak
```

```
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8d.mak
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8u.mak
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8ud.mak
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64.mak
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64d.mak
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64u.mak
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64ud.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8d.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8u.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8ud.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64d.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64u.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64ud.mak
REA32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8.mak
        make -i -C gcc/ -f UT_ZLIB.mak
DGA32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8d.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8d.mak
        make -i -C gcc/ -f UT_ZLIBd.mak
REU32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8u.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8u.mak
        make -i -C gcc/ -f UT_ZLIBu.mak
DGU32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8ud.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8ud.mak
        make -i -C gcc/ -f UT_ZLIBud.mak
FUL32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8.mak
        make -i -C gcc/ -f UT_ZLIB.mak
FUA32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8.mak
        make -i -C gcc/ -f UT_ZLIB.mak
FUU32 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8u.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8u.mak
        make -i -C gcc/ -f UT_ZLIBu.mak
REA64 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64.mak
        make -i -C gcc/ -f UT_ZLIBx64.mak
DGA64 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64d.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64d.mak
        make -i -C gcc/ -f UT_ZLIBx64d.mak
REU64 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64u.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64u.mak
        make -i -C gcc/ -f UT_ZLIBx64u.mak
DGU64 :
        make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64ud.mak
        make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64ud.mak
        make -i -C gcc/ -f UT_ZLIBx64ud.mak
```

```
FUL64 :
	make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64.mak
	make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64.mak
	make -i -C gcc/ -f UT_ZLIBx64.mak
FUA64 :
	make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64.mak
	make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64.mak
	make -i -C gcc/ -f UT_ZLIBx64.mak
FUU64 :
	make -i -C ../PROJECT/gcc -f LIB_ZLIB_1_2_8x64u.mak
	make -i -C ../PROJECT/gcc -f DLL_ZLIB_1_2_8x64u.mak
	make -i -C gcc/ -f UT_ZLIBx64u.mak
clean: clean-custom
	make -i clean -C gcc/ -f UT_ZLIB.mak
depend:
	make -i depend -C gcc/ -f UT_ZLIB.mak
dist:
	make -i -C gcc/ -f UT_ZLIB.mak dist
```

3.6 GENERATE THE GLOBAL MAKEFILE USING EMCC

The global makefile using emcc needs to be generated by using GENMAKE. It should be named **CTBS_2D.emcc.mak**

Here is a sample:

```
#######################################
#    WORKSPACE MAKEFILE BY GENMAKE    #
#       INTENDED FOR GNU MAKE         #
#######################################
PATH_TO_PROJECT = emcc
DEBUG_EXT = d
UNICODE_EXT = u
DUNICODE_EXT = ud
.PHONY: all all-before all-after clean clean-custom depend dist
all :
	make -i -C emcc/ -f LIB_ZLIB_1_2_8.mak
	make -i -C emcc/ -f UT_ZLIB.mak
	make -i -C emcc/ -f LIB_JPEG_9.mak
	make -i -C emcc/ -f UT_JPEG.mak
	make -i -C emcc/ -f LIB_TIFF_4_0_3.mak
	make -i -C emcc/ -f UT_TiffLib.mak
	make -i -C emcc/ -f LIB_OPENJPEG_2_1.mak
	make -i -C emcc/ -f LIB_PNG_1_6_17.mak
	make -i -C emcc/ -f UT_PNG.mak
	make -i -C emcc/ -f LIB_WEBP_0_4_3.mak
	make -i -C emcc/ -f LIB_WEBP_ENC_0_4_3.mak
	make -i -C emcc/ -f LIB_LEPTONICA_1_70.mak
	make -i -C emcc/ -f LIB_LEPTONICA_1_71.mak
	make -i -C emcc/ -f LIB_LEPTONICA_1_72.mak
	make -i -C emcc/ -f UT_LEPTONICA.mak
	make -i -C emcc/ -f LIB_TESSERACT.mak
	make -i -C emcc/ -f UT_TESSERACT.mak
clean: clean-custom
	make -i clean -C emcc/ -f LIB_PNG_1_6_17.mak
	make -i clean -C emcc/ -f LIB_JPEG_9.mak
	make -i clean -C emcc/ -f LIB_TIFF_4_0_3.mak
	make -i clean -C emcc/ -f LIB_LEPTONICA_1_70.mak
```

```
	make -i clean -C emcc/ -f LIB_OPENJPEG_2_1.mak
	make -i clean -C emcc/ -f LIB_WEBP_0_4_3.mak
	make -i clean -C emcc/ -f LIB_WEBP_ENC_0_4_3.mak
	make -i clean -C emcc/ -f UT_ZLIB.mak
	make -i clean -C emcc/ -f UT_JPEG.mak
	make -i clean -C emcc/ -f UT_TiffLib.mak
depend:
	make -i depend -C emcc/ -f UT_ZLIB.mak
	make -i depend -C emcc/ -f UT_JPEG.mak
	make -i depend -C emcc/ -f UT_TiffLib.mak
dist:
	make -i -C emcc/ -f UT_ZLIB.mak dist
	make -i -C emcc/ -f UT_JPEG.mak dist
	make -i -C emcc/ -f UT_TiffLib.mak dist
```

4. COMPILE TESSERACT C/C++

Once all the previous tasks are done, go to your project's directory and use the following command:

```
make –f CTBS_2D.gcc.mak
```

This command will generate JS & HTML files.