Smally

P. Baillehache

June 7, 2020

Contents

1 Interface **1**

2 Code **3**
2.1 smally.c 3
2.2 smally-inline.c 5

3 Makefile **6**

4 Unit tests **7**

5 Unit tests output **8**

Introduction

Smally library is a C library to compress and decompress data.

It provides an implementation of the LZ77 algorithm.

It uses the `PBErr` and `GSet` libraries.

1 Interface

```
// *************** CRYPTIC.H ***************
#ifndef CRYPTIC_H
#define CRYPTIC_H

// ================= Include ==================
#include <stdlib.h>
```

```c
#include <stdio.h>
#include <stdbool.h>
#include "pberr.h"
#include "gset.h"

// ================= Define ==================

#define SMALLY_DEFAULT_OP_MODE SmallyOpMode_LZ77

// ================= Data structures ==================

// Operating mode
typedef enum SmallyOpMode {

  SmallyOpMode_LZ77

} SmallyOpMode;

// Structure for the Feistel cipher
typedef struct Smally {

  // Operating mode
  SmallyOpMode mode;

} Smally;

// ================ Functions declaration ====================

// Static constructor for a Feistel cipher,
// 'keys' is a GSet of null terminated strings, all the same size
// 'fun' is the ciphering function of the form
// void (*fun)(char* src, char* dest, char* key, unsigned long len)
// 'src', 'dest' have same length 'len'
// 'key' may be of any length
#if BUILDMODE != 0
static inline
#endif
Smally SmallyCreateStatic(void);

// Function to free the memory used by the static Smally
void SmallyFreeStatic(Smally* that);

// Get the operating mode of the Smally 'that'
#if BUILDMODE != 0
static inline
#endif
SmallyOpMode SmallyGetOpMode(const Smally* const that);

// Set the operating mode of the Smally 'that' to 'mode'
#if BUILDMODE != 0
static inline
#endif
void SmallySetOpMode(
  Smally* const that,
   SmallyOpMode mode);

// Function to compress a file 'fpIn' with the Smally 'that'
// Save the result in the file 'fpOut'.
void SmallyCompressFile(
      Smally* that,
  FILE* const fpIn,
  FILE* const fpOut);
```

```
// Function to decompress a file 'fpIn' with the Smally 'that'
// Save the result in the file 'fpOut'.
void SmallyDecompressFile(
      Smally* that,
  FILE* const fpIn,
  FILE* const fpOut);

// ================ inliner ====================

#if BUILDMODE != 0
#include "smally-inline.c"
#endif

#endif
```

2 Code

2.1 smally.c

```
// *************** CRYPTIC.C ***************

// ================= Include ==================
#include "smally.h"
#if BUILDMODE == 0
#include "smally-inline.c"
#endif

// ================ Functions implementation ==================

// Function to free the memory used by the static Smally
void SmallyFreeStatic(
  Smally* that) {

#if BUILDMODE == 0

  if (that == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'that' is null");
    PBErrCatch(SmallyErr);

  }

#endif

  // Nothing to do

}

// Function to compress a file 'fpIn' with the Smally 'that'
// Save the result in the file 'fpOut'.
void SmallyCompressFile(
      Smally* that,
  FILE* const fpIn,
  FILE* const fpOut) {
```

```
#if BUILDMODE == 0

  if (that == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'that' is null");
    PBErrCatch(SmallyErr);

  }

  if (fpIn == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'fpIn' is null");
    PBErrCatch(SmallyErr);

  }

  if (fpOut == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'fpOut' is null");
    PBErrCatch(SmallyErr);

  }

#endif

}

// Function to decompress a file 'fpIn' with the Smally 'that'
// Save the result in the file 'fpOut'.
void SmallyDecompressFile(
      Smally* that,
  FILE* const fpIn,
  FILE* const fpOut) {

#if BUILDMODE == 0

  if (that == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'that' is null");
    PBErrCatch(SmallyErr);

  }

  if (fpIn == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'fpIn' is null");
```

```
    PBErrCatch(SmallyErr);

  }

  if (fpOut == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'fpOut' is null");
    PBErrCatch(SmallyErr);

  }

#endif

}
```

2.2 smally-inline.c

```
// *************** CRYPTIC-INLINE.C ***************

// ================ Functions implementation ====================

// Static constructor for a Feistel cipher,
// 'keys' is a GSet of null terminated strings, all the same size
// 'fun' is the ciphering function of the form
// void (*fun)(char* src, char* dest, char* key, unsigned long len)
// 'src', 'dest' have same length 'len'
// 'key' may be of any length
#if BUILDMODE != 0
static inline
#endif
Smally SmallyCreateStatic(void) {

  // Declare a Smally and set the properties
  Smally c = {

    .mode = SMALLY_DEFAULT_OP_MODE

  };

  // Return the Smally
  return c;

}

// Get the operating mode of the Smally 'that'
#if BUILDMODE != 0
static inline
#endif
SmallyOpMode SmallyGetOpMode(
  const Smally* const that) {

#if BUILDMODE == 0

  if (that == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
```

```
      SmallyErr->_msg,
      "'that' is null");
    PBErrCatch(SmallyErr);

  }

#endif

  // Return the operating mode
  return that->mode;

}

// Set the operating mode of the Smally 'that' to 'mode'
#if BUILDMODE != 0
static inline
#endif
void SmallySetOpMode(
  Smally* const that,
   SmallyOpMode mode) {

#if BUILDMODE == 0

  if (that == NULL) {

    SmallyErr->_type = PBErrTypeNullPointer;
    sprintf(
      SmallyErr->_msg,
      "'that' is null");
    PBErrCatch(SmallyErr);

  }

#endif

  // Set the operating mode
  that->mode = mode;

}
```

3 Makefile

```
# Build mode
# 0: development (max safety, no optimisation)
# 1: release (min safety, optimisation)
# 2: fast and furious (no safety, optimisation)
BUILD_MODE?=0

all: pbmake_wget main smally

# Automatic installation of the repository PBMake in the parent folder
pbmake_wget:
if [ ! -d ../PBMake ]; then wget https://github.com/BayashiPascal/PBMake/archive/master.zip; unzip master.zip; rm -f

# Check code style
style:
cbo *.h *.c

# Makefile definitions
```

```
MAKEFILE_INC=../PBMake/Makefile.inc
include $(MAKEFILE_INC)

# Rules to make the executable
repo=smally
$($(repo)_EXENAME): \
$($(repo)_EXENAME).o \
$($(repo)_EXE_DEP) \
$($(repo)_DEP)
$(COMPILER) `echo "$($(repo)_EXE_DEP) $($(repo)_EXENAME).o" | tr ' ' '\n' | sort -u` $(LINK_ARG) $($(repo)_LINK_ARG)

$($(repo)_EXENAME).o: \
$($(repo)_DIR)/$($(repo)_EXENAME).c \
$($(repo)_INC_H_EXE) \
$($(repo)_EXE_DEP)
$(COMPILER) $(BUILD_ARG) $($(repo)_BUILD_ARG) `echo "$($(repo)_INC_DIR)" | tr ' ' '\n' | sort -u` -c $($(repo)_DIR)/$

# Rules to make the tool
smally: \
main-smally.o \
$($(repo)_EXE_DEP) \
$($(repo)_DEP)
$(COMPILER) `echo "$($(repo)_EXE_DEP) main-smally.o" | tr ' ' '\n' | sort -u` $(LINK_ARG) $($(repo)_LINK_ARG) -o smal

main-smally.o: \
main-smally.c \
$($(repo)_INC_H_EXE) \
$($(repo)_EXE_DEP)
$(COMPILER) $(BUILD_ARG) $($(repo)_BUILD_ARG) `echo "$($(repo)_INC_DIR)" | tr ' ' '\n' | sort -u` -c main-smally.c

install:
cp smally ~/Tools/smally

testSmally:
smally -lz77 -out test.sma -compress main.c && smally -lz77 -out test.txt -decompress test.sma && diff main.c test.tx
```

4 Unit tests

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

void UnitTestAll() {

  printf("UnitTestAll OK\n");

}

int main() {

  UnitTestAll();

  // Return success code
  return 0;

}
```

5 Unit tests output

```
UnitTestAll OK
```