

# CreSP/I

*\kri-sp-y\*

The **Cre**stlength **S**torage **P**rotocol/**I**nterface

Copyright (c) 2016, Charles Thompson  
<[chmithbiz@gmail.com](mailto:chmithbiz@gmail.com)>

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

**THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE**

# How CreSP/I Works

The **C**restlength **S**torage **P**rotocol and **I**nterface encodes bytes by creating a square wave in which the length of the crest determines the value. The encoded data starts with a leader which gives the minimum crestlength (a crestlength that is equivalent to the value zero). The leader is ended with the maximum crestlength (255x the length of the minimum crestlength) and henceforth the data is read until the end of recording.

In case it has not been inferred already, each byte is equal to the crestlength divided by the minimum crestlength.

## Programs Provided

- **acmp** - optimizes data for CreSP/I storage.
- **adec** - processes data that has been optimized for CreSP/I storage; undoes the work of acmp.
- **a2f** - reads CreSP/I encoded data from raw unsigned 8-bit PCM files.
- **f2a** - encodes data into CreSP/I and writes it to raw unsigned 8-bit PCM files.

# acmp

## pre-audio computation

optimizes data for CreSP/I by representing the most common values with the lowest values

```
1  #include "common.h"
2
3  int main( void )
4  {
5      num *frqTbl = calloc(MVAL, sizeof(num)), tblSz = MVAL, top = NUL, i;
6      byte *outTbl = calloc(MVAL, sizeof(byte)), buf = GET();
7
8      for(;!EF; buf = GET())
9      {
10         frqTbl[buf] ++;
11         if(frqTbl[buf] > frqTbl[top]) top = buf;
12     }
13
14     for(;tblSz && frqTbl[top] != NUL;tblSz --)
15     {
16         outTbl[MVAL - tblSz] = top, frqTbl[top] = 0;
17
18         for(i = NUL; i < MVAL; i ++) if(frqTbl[i] > frqTbl[top]) top = i;
19     }
20
21     PUT(tblSz = MVAL - tblSz);
22
23     for(i = NUL; i < tblSz; i ++)
24     {
25         PUT(outTbl[i]);
26         frqTbl[outTbl[i]] = i;
27     }
28
29     fseek(stdin, 0, SEEK_SET);
30
31     for(buf = GET(); !EF; buf = GET()) PUT(frqTbl[buf]);
32
33     return 0;
34 }
```

# adec

## acmp decoder

decodes the data encoded by acmp.

```
1  #include "common.h"
2
3  int main()
4  {
5      num tblSz = GET(), i = NUL;
6      byte *table = calloc(MVAL, sizeof(byte)), buf = GET();
7
8      for(; i < tblSz && !EF; i ++, buf = GET()) table[i] = buf;
9
10     for(; !EF; buf = GET()) PUT(table[buf]);
11
12     return 0;
13 }
```

# a2f

## audio to file

reads raw unsigned 8-bit pcm from stdin and writes the data obtained through CreSP/I to stdout

```
1  #include "common.h"
2
3  #define SKIP 200
4  #define CLENI 1
5  #define WCOR 1
6  #define THRSH 50
7
8  num bLen = 1;
9
10 num rounded(double di, double dv)
11 {
12     double d = di / dv;
13     num n = d;
14     double f = d - n;
15
16     return n + (f > 0.445);
17 }
18
19 num getWave()
20 {
21     num cLen = CLENI;
22
23     while(GET() < THRSH && !EF);
24     while(GET() >= THRSH && !EF) cLen ++;
25
26     return rounded(cLen, bLen) - WCOR;
27 }
28
29 int main( void )
30 {
31     byte buf;
32
33     fseek(stdin, SKIP, SEEK_SET);
34     getWave();
35     bLen = getWave() + WCOR;
36
37     while(!EF && getWave() != LEND);
38     for(buf = getWave(); !EF; buf = getWave()) PUT(buf);
39
40     PUT(buf);
41
42     return 0;
43 }
```

# f2a

## file to audio

encodes stdin to CreSP/I and writes the raw unsigned 8-bit PCM to stdout

```
1  #include "common.h"
2
3  #define TLEN 1
4  #define HLEN 20000
5  #define AMP 200
6  #define LCOR 2
7
8  void genWave(num len)
9  {
10     num tLen = TLEN;
11
12     while(--tLen + LCOR) PUT(NUL);
13     while(--len + LCOR) PUT(AMP);
14 }
15
16 int main( void )
17 {
18     num hLen = HLEN, buf = GET();
19
20     while(--hLen) genWave(NUL);
21
22     genWave(LEND);
23
24     for(; !EF; buf = GET()) genWave(buf);
25
26     return 0;
27 }
```

# common.h

## common inclusions and definitions

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define PUT putchar
5  #define GET getchar
6  #define EF feof(stdin)
7  #define NUL 0
8  #define LEND 255
9  #define MVAL 256
10
11 typedef unsigned int num;
12 typedef unsigned char byte;
```