CreSP/I

\'kris-pē

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

How CreSP/I Works

The **Cre**stlength **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 the recording.

Bytes encoded in CreSP are decoded by dividing the crestlength by the minimal crestlength and subtracting one.

Programs Provided

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

acmp

pre-**a**udio **co**m**p**utation

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

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

adec

acmp **dec**oder

decodes the data encoded by acmp.

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

a2f

audio **to f**ile

decodes raw unsigned 8-bit pcm through CreSP and writes the product to stdout

```
1  #include "common.h"
2
3  #define SKIP 200 /* The amount of bytes to skip in stdin */
4  #define WCOR 1 /* The correction for getting an accurate minimal crestlength*/
5  #define THRSH 0x90 /* The threshold to which it is considered part of a
6                        crest*/
7  #define RCNST 0.445 /* If a value is greater than this constant, it is rounded
8                         up */
9
10 num bLen = sizeof(byte);
11
12 num dvid(dbl a, dbl b) /* Divide and round */
13 {
14     dbl d = a / b;
15     num n = d;
16     dbl f = d - n;
17
18     return n + (f > RCNST);
19 }
20
21 num getW()
22 {
23     num cLen = NUL;
24
25     while(GET() < THRSH && !EF);
26     while(GET() >= THRSH && !EF) cLen ++;
27
28     return dvid(cLen, bLen);
29 }
30
31 int main( void )
32 {
33     byte buf;
34
35     getW();
36     bLen = getW() + WCOR;
37
38     while(!EF && getW() != LEND);
39     for(buf = getW(); !EF; buf = getW()) PUT(buf);
40
41     PUT(buf);
42
43     return NUL;
44 }
```

f2a

file **to** **a**udio

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

```
1  #include "common.h"
2
3  #define TLEN 1 /* The length of the trough */
4  #define HLEN 20000 /* The length of the header */
5  #define AMP 0xFF /* The amperage */
6  #define LCOR 1 /* Correction constant for genWave */
7
8  void genW(num len)
9  {
10     num tLen = TLEN;
11
12     while(tLen--) 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) genW(NUL);
21
22     genW(LEND);
23
24     for(; !EF; buf = GET()) genW(buf);
25
26     return NUL;
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;
13 typedef double dbl;
```