CreSP/I

\kri-sp-y

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

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 **dec**oder

decodes the data encoded by acmp.

```
#include “common.h”

int main()
{
    num tblSz = GET(), i = NUL;
    byte *table = calloc(MVAL, sizeof(byte)), buf = GET();

    for(; i < tblSz && !EF; i ++, buf = GET()) table[i] = buf;

    for(; !EF; buf = GET()) PUT(table[buf]);

    return 0;
}
```

a2f

audio **to f**ile

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 /* The amount of bytes to skip in stdin */
4  #define WCOR 1 /* The correction for getting an accurate minimal crestlength*/
5  #define THRSH 50 /* The threshold to which it is considered part of a crest */
6  #define RCONST 0.445 /* If a value is greater than this constant, it is
7                          rounded up */
8
9  num bLen = sizeof(byte);
10
11 num rounded(double di, double dv)
12 {
13     double d = di / dv;
14     num n = d;
15     double f = d - n;
16
17     return n + (f > RCONST);
18 }
19
20 num getWave()
21 {
22     num cLen = NUL;
23
24     while(GET() < THRSH && !EF);
25     while(GET() >= THRSH && !EF) cLen ++;
26
27     return rounded(cLen, bLen);
28 }
29
30 int main( void )
31 {
32     byte buf;
33
34     fseek(stdin, SKIP, SEEK_SET);
35     getWave();
36     bLen = getWave() + WCOR;
37
38     while(!EF && getWave() != LEND);
39     for(buf = getWave(); !EF; buf = getWave()) PUT(buf);
40
41     PUT(buf);
42
43     return 0;
44 }
```

f2a

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

encodes stdin to CreSP/I 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 200 /* The amperage */
6  #define LCOR 1 /* Correction constant for genWave */
7
8  void genWave(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) 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;
```