###########

<honig@sprynet.com>

##: 43184

####### # ### ####### ########## ########

#####, ###/1 ### #### # ################### ### #### ##### ####### ## ######## ### ### #### ##### #########

2013#11#13 07:52:45 # ###.

##

#################################? ########## ###############

##

1. ## 1
2. #### 1
3. ## 2
4. ## 2
5. ##### ### 2
6. ##### ####### 2
7. ###/###/#386/####/########## 3
8. ######## ######### ########### #### (####### = 8 ###) 3

1.

########## ##? ##################### ############# ## ##### #######(1) ################# ######## ###################?

2.

#########################

1. ######################### #########################

2. ############################# ###########################

#####################

2.1.

(####) ###################### #### ########################## ######## ################ (## 0.25 #####)

2.2.

##############*######*(1) ############# ########### # ##### ##################*####* ######## ###### ######

######:

```
tcpdump -c 4000 -s 10000 -w dumpfile.bin
```

##4000 ############*dumpfile.bin* ####### ###### 10,000 ###################

3.

#############

1. ##### #### ##### #############################
2. ### ####### #######
3. ######### ####### ### ###(1) ### ####® ########### ####y ##################### ############## ######## ########################## ###########################
4. ######## ################################# ############# (7.18) ##### ##### ####### 93% (6.7)# ######## 29% (2.1) ################

```
% tcpdump -c 4000 -s 10000 -w ipsecdemo.bin
% uliscan ipsecdemo.bin

Uliscan 21 Dec 98
L=8 256 258560
Measuring file ipsecdemo.bin
Init done
Expected value for L=8 is 7.1836656
6.9396 -------------------------------------------------------
6.6177 ----------------------------------------------------
6.4100 --------------------------------------------------
2.1101 -----------------
2.0838 -----------------
2.0983 -----------------
```

4.

####################### ## ############# ########### ########### (#############) ############## ############################# ###################### ####### ################## ########################## ########################

5. #####

############ ######## (######## ######## ######## ##########) # ## #4 # ## #6 ######## #6 ########## ##### ######## ## (####) ####################### #### ### #################### ###### ### ######## ########################### ##### # 2 ###################

6. #####

####### ########### ##### ######################## ############ IPSEC ######################## ## ######/######### ######(8) ##### ##### ###############

####### # ##### ############## ####### #################

7. ###/###/#386/####/##########

########## #######(1) ################################## ### ######(8) ############/######## ####
#####

```
device bpf
```

8. ######## ######### ########### #### (####### = 8 ###)

####### ###################

```
/*
 ULISCAN.c   ---blocksize of 8

 1 Oct 98
 1 Dec 98
 21 Dec 98       uliscan.c derived from ueli8.c

 This version has // comments removed for Sun cc

 This implements Ueli M Maurer's "Universal Statistical Test for Random
 Bit Generators" using L=8

 Accepts a filename on the command line; writes its results, with other
 info, to stdout.

 Handles input file exhaustion gracefully.

 Ref: J. Cryptology v 5 no 2, 1992 pp 89-105
 also on the web somewhere, which is where I found it.

 -David Honig
 honig@sprynet.com

 Usage:
 ULISCAN filename
 outputs to stdout
*/

#define L 8
#define V (1<<L)
#define Q (10*V)
#define K (100   *Q)
#define MAXSAMP (Q + K)

#include <stdio.h>
#include <math.h>

int main(argc, argv)
int argc;
char **argv;
{
 FILE *fptr;
 int i,j;
 int b, c;
 int table[V];
 double sum = 0.0;
 int iproduct = 1;
 int run;
```

```
extern double   log(/* double x */);

printf("Uliscan 21 Dec 98 \nL=%d %d %d \n", L, V, MAXSAMP);

if (argc < 2) {
  printf("Usage: Uliscan filename\n");
  exit(-1);
} else {
  printf("Measuring file %s\n", argv[1]);
}

fptr = fopen(argv[1],"rb");

if (fptr == NULL) {
  printf("Can't find %s\n", argv[1]);
  exit(-1);
}

for (i = 0; i < V; i++) {
  table[i] = 0;
}

for (i = 0; i < Q; i++) {
  b = fgetc(fptr);
  table[b] = i;
}

printf("Init done\n");

printf("Expected value for L=8 is 7.1836656\n");

run = 1;

while (run) {
  sum = 0.0;
  iproduct = 1;

  if (run)
    for (i = Q; run && i < Q + K; i++) {
      j = i;
      b = fgetc(fptr);

      if (b < 0)
        run = 0;

      if (run) {
        if (table[b] > j)
          j += K;

        sum += log((double)(j-table[b]));

        table[b] = i;
      }
    }

  if (!run)
    printf("Premature end of file; read %d blocks.\n", i - Q);

  sum = (sum/((double)(i - Q))) /  log(2.0);
  printf("%4.4f ", sum);

  for (i = 0; i < (int)(sum*8.0 + 0.50); i++)
    printf("-");

  printf("\n");
```

```
    /* refill initial table */
    if (0) {
      for (i = 0; i < Q; i++) {
        b = fgetc(fptr);
        if (b < 0) {
          run = 0;
        } else {
          table[b] = i;
        }
      }
    }
  }
}
```