Branch: master ▾ Find file Copy path

cs-35l / assignment6 / randmain.c


prithvikannan Homework portion

c9d1e57 on Nov 7

1 contributor

Raw Blame History

148 lines (127 sloc) 3.7 KB

```
/* Generate N bytes of random output.  */

/* When generating output this program uses the x86-64 RDRAND
   instruction if available to generate random numbers, falling back
   on /dev/urandom and stdio otherwise.

   This program is not portable.  Compile it with gcc -mrdrnd for a
   x86-64 machine.

   Copyright 2015, 2017 Paul Eggert

   This program is free software: you can redistribute it and/or
   modify it under the terms of the GNU General Public License as
   published by the Free Software Foundation, either version 3 of the
   License, or (at your option) any later version.

   This program is distributed in the hope that it will be useful, but
   WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
   General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.  */

#include "randcpuid.h"
#include <errno.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

static bool
writebytes(unsigned long long x, int nbytes)
{
    int ndigits = nbytes * 2;
    do
    {
        if (putchar("0123456789abcdef"[x & 0xf]) < 0)
            return false;
        x >>= 4;
        ndigits--;
    } while (0 < ndigits);

    return 0 <= putchar('\n');
}

/* Main program, which outputs N bytes of random data.  */
int main(int argc, char **argv)
{
    /* Check arguments.  */
    bool valid = false;
```

```
52      long long nbytes;
53      if (argc == 2)
54      {
55          char *endptr;
56          errno = 0;
57          nbytes = strtoll(argv[1], &endptr, 10);
58          if (errno)
59              perror(argv[1]);
60          else
61              valid = !*endptr && 0 <= nbytes;
62      }
63      if (!valid)
64      {
65          fprintf(stderr, "%s: usage: %s NBYTES\n", argv[0], argv[0]);
66          return 1;
67      }
68
69      /* If there's no work to do, don't worry about which library to use.  */
70      if (nbytes == 0)
71          return 0;
72
73      /* Now that we know we have work to do, arrange to use the
74       appropriate library.  */
75      void *library;
76      char *errorLoading;
77      unsigned long long (*rand64)(void);
78
79      if (rdrand_supported())
80      {
81          /* Hardware lib: randlibhw */
82          library = dlopen("randlibhw.so", RTLD_LAZY);
83          if (!library)
84          {
85              fprintf(stderr, "Error when opening dynamic library.\n");
86              return 1;
87          }
88
89          rand64 = dlsym(library, "rand64");
90          errorLoading = dlerror();
91          if (errorLoading != NULL)
92          {
93              fprintf(stderr, "Error when loading library.\n");
94              return 1;
95          }
96      }
97      else
98      {
99          /* Software lib: randlibsw */
100         library = dlopen("randlibsw.so", RTLD_LAZY);
101         if (!library)
102         {
103             fprintf(stderr, "Error when opening dynamic library.\n");
104             return 1;
105         }
106
107         rand64 = dlsym(library, "rand64");
108         errorLoading = dlerror();
109         if (errorLoading != NULL)
110         {
111             fprintf(stderr, "Error when loading library.\n");
112             return 1;
113         }
114     }
115
116     int wordsize = sizeof rand64();
117     int output_errno = 0;
```

```
118
119     do
120     {
121         unsigned long long x = rand64();
122         int outbytes = nbytes < wordsize ? nbytes : wordsize;
123         if (!writebytes(x, outbytes))
124         {
125             output_errno = errno;
126             break;
127         }
128         nbytes -= outbytes;
129     } while (0 < nbytes);
130
131     if (dlclose(library))
132     {
133         fprintf(stderr, "Error when closing dynamic library.\n");
134         return 1;
135     }
136
137     if (fclose(stdout) != 0)
138         output_errno = errno;
139
140     if (output_errno)
141     {
142         errno = output_errno;
143         perror("output");
144         return 1;
145     }
146
147     return 0;
148 }
```