#include "randcpuid.h"

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <dlfcn.h>

/\* Main program, which outputs N bytes of random data. \*/

int

main (int argc, char \*\*argv)

{

/\* Check arguments. \*/

bool valid = false;

long long nbytes;

if (argc == 2)

{

char \*endptr;

errno = 0;

nbytes = strtoll (argv[1], &endptr, 10);

if (errno)

perror (argv[1]);

else

valid = !\*endptr && 0 <= nbytes;

}

if (!valid)

{

fprintf (stderr, "%s: usage: %s NBYTES\n", argv[0], argv[0]);

return 1;

}

/\* If there's no work to do, don't worry about which library to use. \*/

if (nbytes == 0)

return 0;

/\* Now that we know we have work to do, arrange to use the

appropriate library. \*/

void (\*initialize) (void);

unsigned long long (\*rand64) (void);

void (\*finalize) (void);

void\* software;

void\* hardware;

char\* error;

if (rdrand\_supported ())

{

hardware = dlopen("randlibhw.so", RTLD\_NOW);

if(hardware == NULL){

printf("Failed to open");

exit(1);

}

software = NULL;

//initialize = hardware\_rand64\_init;

initialize = dlsym(hardware, "hardware\_rand64\_init");

error = dlerror();

if (error){

printf("Failed to locate function and initialize");

exit(1);

}

//rand64 = hardware\_rand64;

rand64 = dlsym(hardware, "hardware\_rand64");

error = dlerror();

if(error){

printf("Failed to locate function");

exit(1);

}

//finalize = hardware\_rand64\_fini;

finalize = dlsym(hardware, "hardware\_rand64\_fini");

error = dlerror();

if(error){

printf("Failed to locate function and finalize");

exit(1);

}

}

else

{

software = dlopen("randlibsw.so", RTLD\_NOW);

if(software == NULL){

printf("Failed to open");

exit(1);

}

hardware = NULL;

//initialize = software\_rand64\_init;

initialize = dlsym(software, "software\_rand64\_init");

error = dlerror();

if (error){

printf("Failed to locate function and initialize");

exit(1);

}

//rand64 = software\_rand64;

rand64 = dlsym(software, "software\_rand64");

error = dlerror();

if(error){

printf("Failed to locate function");

exit(1);

}

//finalize = software\_rand64\_fini;

finalize = dlsym(software, "software\_rand64\_fini");

error = dlerror();

if(error){

printf("Failed to locate function and finalize");

exit(1);

}

}

initialize ();

int wordsize = sizeof rand64 ();

int output\_errno = 0;

do

{

unsigned long long x = rand64 ();

size\_t outbytes = nbytes < wordsize ? nbytes : wordsize;

if (fwrite (&x, 1, outbytes, stdout) != outbytes)

{

output\_errno = errno;

break;

}

nbytes -= outbytes;

}

while (0 < nbytes);

if (fclose (stdout) != 0)

output\_errno = errno;

if (output\_errno)

{

errno = output\_errno;

perror ("output");

finalize ();

return 1;

}

finalize ();

return 0;

}

randlibhw.so: randlibhw.c

$(CC) $(CFLAGS) randlibhw.c -shared -fPIC -o randlibhw.so

randlibsw.so: randlibsw.c

$(CC) $(CFLAGS) randlibsw.c -shared -fPIC -o randlibsw.so

randmain: randcpuid.o randmain.o

$(CC) $(CFLAGS) -ldl -Wl,-rpath=$PWD -o randmain randcpuid.o randmain.o