Assignment 1:

Bilal Waraich

Operating Systems

Problem 1.1:

strace:

A screenshot of a computer program

Description automatically generated

ltrace:

A screenshot of a computer program

Description automatically generated

1. The top three system calls:
2. openat: used to open a file relative to a directory file descriptor
3. mmap: maps files or devices into an address space in memory
4. close: closes an open file descriptor.

The top three library calls:

1. fwrite: write to a binary stream
2. fputc: output a character into a stream
3. \_\_freading: indicate whether a stream is used for reading

Problem 1.2:

1. *int open(const char \*path, int oflag, ...)* could fail due to the file that is being opened not existing.

*int close(int fildes)* could fail due to there being an interruption by a signal while it is being executed.

1. The value of errno does not change if no error occurs. Instead it retains its previous value and is only modified when there is an error.

Problem 1.3:

#define \_DEFAULT\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

extern char \*\*environ;

static int verbose = 0;

static int isenv(const char \*arg) {

return strchr(arg, '=') != NULL;

}

static void set(char \*arg) {

if (verbose) {

fprintf(stderr, "setenv:\t%s\n", arg);

}

if (putenv(arg) == -1) {

perror("putenv");

}

}

static void unset(char \*arg) {

if (verbose) {

fprintf(stderr, "unset:\t%s\n", arg);

}

if (unsetenv(arg) == -1) {

perror("unsetenv");

}

}

static void show(char \*env[]) {

for (int i = 0; env[i]; i++) {

puts(env[i]);

}

}

static void exec(char \*argv[]) {

if (verbose) {

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

for (int i = 0; argv[i]; i++) {

fprintf(stderr, "arg[%d]= '%s'\n", i, argv[i]);

}

}

execvp(argv[0], argv);

perror("execvp");

\_exit(1);

}

int main(int argc, char \*argv[]) {

int opt, nargc = 0;

char \*nargv[argc];

while ((opt = getopt(argc, argv, "vu:")) != -1) {

switch (opt) {

case 'v':

verbose = 1;

break;

case 'u':

unset(optarg);

break;

default: /\* '?' \*/

fprintf(stderr, "usage: env [-v] [-u name] [name=value ...] command [arg ...]]\n");

exit(EXIT\_FAILURE);

}

}

for (int i = optind; i < argc; i++) {

if (isenv(argv[i])) {

set(argv[i]);

} else {

nargv[nargc++] = argv[i];

}

}

nargv[nargc] = NULL;

if (nargv[0] == NULL) {

show(environ);

} else {

exec(nargv);

}

return EXIT\_SUCCESS;

}