Laboratory Exercise 4 Solutions

cpe 357 Fall 2010

Due to me by (or before) 4:00pm, Monday, November 1st.
The Laboratory Exercise is to be done individually.

Problems

There are no written exercises this week. Stay tuned for next week.

Laboratory Exercise

1. There's one task for this week: Write a program, `mypwd.c`, that performs the same task as `/bin/pwd` *without* using the `getcwd(3)` function described by POSIX. I want you to traverse the directories and do it yourself. The technique is described in Stevens, Section 4.22.

   If there are multiple paths to (possible names for) any partiular directory, return the one that appears earliest in each directory that is traversed.

   Your program should produce the same results as `/bin/pwd`. In case of error, call `perror()` with the string "mypwd" to report the error and exit. If the pathname is too long (see below), simply print "path too long" and exit. Similarly, if for whatever reason, you are unable to determine the present working directory (e.g. it's been unlinked), print "cannot get current directory."

   Just to make life easier, you may assume you will get no paths deeper than `PATH_MAX`, defined in `limits.h`. If `PATH_MAX` is undefined, you may define it to be 2048 characters.

Tricks and Tools

You will probably want to look into Stevens, Chapter 4, and:

`opendir(3)`	open a directory for reading
`readdir(3)`	read a directory entry
`rewinddir(3)`	rewind a directory to the beginning
`closedir(3)`	close a directory
`stat(2)`	get file status
`lstat(2)`	get file status

Remember, in an environment where there is more than one filesystem mounted, it is possible, though unlikely, that you could discover that the i-node of the top of a mounted filesystem is the same as the one you're looking for. (Remember, inode numbers are only unique within a filesystem.) If this happens, your program might get confused. Even though the inode number may be the same on another device, the device number won't. Check both and you'll be in good shape.

For the Laboratory Exercises: Submit the program described above to the `lab04` directory of `pn-cs357` via handin.

Solutions

Here are two versions of the program, a complete iterative version on page 2, and a smaller recursive version on page 5.

The program `mypwd.c`.

```
/*
 * mypwd:
 *
 * Author: Dr. Phillip Nico
 *         Department of Computer Science
 *         California Polytechnic State University
 *         One Grand Avenue.
 *         San Luis Obispo, CA  93407  USA
 *
 * Email:   pnico@csc.calpoly.edu
 *
 * Revision History:
 *         $Log:$
 */

#include <dirent.h>
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

#ifndef PATH_MAX
#define PATH_MAX 2048
#endif

#define TOO_LONG_ERROR "path too long\n"

void getfileinfo(const char *one, ino_t *inode, dev_t *dev) {
  /* return the inode and device for the given file, or
   * bail if it doesn't exist.
   */
  struct stat sb;
  if ( !inode || !dev ) {
    fprintf(stderr,"%s:  NULL pointer argument\n",__FUNCTION__);
    exit(-1);
  }
  if ( -1 == lstat(one,&sb) ) {
    perror(one);
    exit(-1);
  }
  *inode = sb.st_ino;
  *dev   = sb.st_dev;
}

int atroot() {
  /* return true if we're at the top of the tree.
   * we determine this by looking at both the inode and
```

```
  * device numbers for "." and ".." and checking to see
  * whether they're the same.
  */
 ino_t self_i, parent_i;
 dev_t self_d, parent_d;

 getfileinfo(".",&self_i,&self_d);
 getfileinfo("..",&parent_i,&parent_d);
             /* I'm my own grand-pa... */
 return ((self_i == parent_i) &&                                    60
         (self_d == parent_d) );

}

int main(int argc, char *argv[]){
 /* reconstruct an absolute path to the current directory working
  * backwards to the root.  Backwards in the string, too, because
  * we'll get our information from the end.
  */
 char *head;                                                        70
 char buffer[PATH_MAX + 1];
 DIR  *here;
 ino_t self_inode, other_inode;
 dev_t self_dev, other_dev;
 struct dirent *dp;
 int len;

 head = buffer+PATH_MAX+1;
 *--head = '\0';
 *--head = '/';                                                     80

 /* now work our way back up. */
 while ( !atroot() ) {
  /* first, find out who I am */
  getfileinfo(".",&self_inode, &self_dev);

  /* move on up, and start looking for ourselves in our parent */
  chdir("..");

  /* open the directory */                                          90
  if ( NULL == ( here = opendir(".")) ) {
   perror(".");
   exit(-1);
  }

  /* look for something that has our inode and device */
  while ( NULL != (dp=readdir(here)) ) {
   getfileinfo(dp->d_name, &other_inode, &other_dev);
   if ( ( self_inode == other_inode ) &&
        ( self_dev   == other_dev   ) )                             100
       break;
  }
```

```
  /* check to see if we found it; bail if not */
  if ( dp == NULL ) {
    /* didn't find it! (Help! We're lost!)*/
    fprintf(stderr,"cannot get current directory\n");
    exit(-1);
  }

  /* make room */
  len   = strlen(dp->d_name);
  head -= len + 1;

  if ( head >= buffer ) {
    /* copy it over */
    *head = '/';
    strncpy(head+1,dp->d_name,len);
  }

  if ( -1 == closedir(here)) {  /* close the directory */
    perror(".");
    exit(-1);
  }

 }

 if ( head < buffer )          /* too long */
   printf(TOO_LONG_ERROR);
 else
   printf("%s\n",head);

 return 0;
}
```

The program `small.c`.

```
/*
 * A small version of mypwd.  It doesn't do the error checking
 * it should, but it does demonstrate that this can be done fairly
 * densely.
 */

#include <dirent.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

void recurse(dev_t dev, ino_t ino) {
  struct stat sb;
  struct dirent *d;
  DIR *dir;

  chdir("..");
  lstat(".",&sb);

  if ( (dev == sb.st_dev) && (ino == sb.st_ino)  )
    return;

  recurse(sb.st_dev,sb.st_ino);

  for(dir=opendir(".");(d=readdir(dir)); ) {
    lstat(d->d_name,&sb);
    if ( (sb.st_ino == ino) && (sb.st_dev == dev) )
      break;
  }

  chdir(d->d_name);
  printf("/%s",d->d_name);
}

int main(int argc, char *argv[]){
  struct stat sb;

  lstat(".",&sb);
  recurse(sb.st_dev,sb.st_ino);
  printf("\n");
  return 0;
}
```