Wayne State University
CSC 4421 – Winter 2017
Computer Operating Systems: Lab
Lab 2: Intro to C/System Calls
Instructors: Sec. 1: David Warnke
Sec. 2: Rui Chen
Points Possible: 100
Due Date: 2/6/17

Goals

The purpose of this lab is to help you learn more about some frequently used system calls.

Collaboration Policy

This is an individual assignment. You may help each other verbally with the correct execution of Linux/C commands. You may not type anything in another student's session, nor provide any code or answers to the questions below.

Task 1: Printing File Names using Dirent Library: (50points)

Here we want to access a Linux directory using C. To do so we use the structs and functions made available in the dirent library. Below we see the function prototypes for the functions opendir, readdir, and closedir which have been implemented in the dirent library. DIR is simple type that is available from the library.

```
SYNOPSIS
#include <dirent.h>
DIR *opendir ( const char *dirname );
struct dirent *readdir (DIR * dir );
int closedir (DIR * dir ) ;
```

Dirent structure is defined as follows:

```
struct dirent {
ino_t d_ino ; /* file serial number */
off_t d_off ; /* offset to the next dirent */
unsigned short d_reclen ; /* length of this record */
unsigned char d_ type ; /* type of file */
char d_name [ 256 ] ; /* filename */
} ;
```

The following program displays the filenames contained in a directory whose pathname is passed as a command-line argument.

- **Type in and complete the following program on the computer.** Save it as shownames.c.
- Compile the program as shownames.out and execute it by giving it a directory as a command line argument. For example type the following into the terminal: `./shownames.out ~/CSC4421`. If you typed that then the program below will receive values of: argc==2, argv[0]=="./shownames.out", argv[1]=="~/CSC4421".

shownames.c:

```
#include <dirent.h>
#include <errno.h>
#include <stdio.h>
int main (int argc, char *argv[])
{
        struct dirent *direntp;
        // define a pointer with type directory stream (DIR):
        ________

        // check the number of arguments (argc). It should be two. If it
        // is not two, then print an error message and exit the program:
        if (__________)
        {
           fprintf (stderr, "Usage: %s directory name \n", argv[0]);
           return 1;
        }

        // The following lines should open the directory given by
        //   argument argv[1] (by using opendir).
        // Store it in your defined directory stream variable.
        // The rest checks if the system call returned a proper value
        //   if not an error message is printed and the program closed
        if ( ________ == NULL)
        {
                perror ( "Failed to open directory" );
                return 1;
        }

        // Read all the entries in this directory and store them in
        // direntp. readdir will read one entry at a time and increment
        // automatically. This is why it is in a while loop.
        // Then, print all the file names (using the struct from readdir).
        while (_______ != NULL)
                printf ("%s \n" , _______ );

        // close the defined directory stream.
        while ( ( closedir (_______) == -1) && ( errno == EINTR) );

        return 0;
}
```

Task 2: Printing the time that the file path was created (50 points)

The stat system call can access a file by name and retrieve file status information. It is given by: `int stat (const char *pathname, struct stat *buf);` This system call returns a stat structure (both the function and struct are called stat), which contains the following fields:

```
struct stat {
dev_t st_dev; /* ID of device containing file */
ino_t st_ino; /* inode number */
mode_t st_mode; /* protection */
nlink_t st_nlink; /* number of hard links */
uid_t st_uid; /* user ID of owner */
gid_t st_gid; /* group ID of owner */
dev_t st_rdev; /* device ID (if special file) */
off_t st_size; /* total size, in bytes */
blksize_t st_blksize; /*blocksize for file system I/O */
blkcnt_t st_blocks; /* number of 512B blocks allocated */
time_t st_atime; /* time of last access */
time_t st_mtime; /* time of last modification */
time_t st_ctime; /* time of last status change */
};
```

On success, zero is returned. On error, -1 is returned, and **errno** is set appropriately. The following program displays the time that the file path was last accessed.

- **Type in and complete the following program on the computer.** Save it as printaccess.c . **Compile the program, then execute it by giving it a path name as an argument.**

printaccess.c:

```
#include <stdio.h>
#include <time.h>
#include <sys/stat.h>
int main (int argc, char *argv[])
{

   // define an instance of struct stat (don't use a pointer).
   ________

   // Use the stat function store the status of the file in your stat struct
   // stat takes a pointer, so pass your struct by reference with &
   if ( _____ == -1)
   {
      // use perror to print error message then exit program
      ______
   }
   // print the last access time for the file using defined instance of
   // struct stat. ctime requires a pointer, hence the & below.
   printf ("%s last accessed at %s", argv[1], ctime(&_____.st_atime));
}
```