Lab 11

Jarred Parr and Alexander Fountain

1. a. `stat(1)` is a command line program and `stat(3)` is a library function which invokes that same functionality but inside of a running executable whose data can be captured and used for other purposes.

   b. This program reads a from `stdin` the name of a file and then uses `stat` to report the mode and type of file that it is.

   c.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <errno.h>

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

   if (argc < 2) {
      printf ("Usage: filename required\n");
      exit(1);
   }

   if (stat (argv[1], &statBuf) < 0) {
      perror ("huh?  there is ");
      exit(1);
   }

   if ((statBuf.st_mode & S_IFMT) == S_IFDIR) {
      printf("%s is a directory\n", argv[1]);
   } else {
      printf("%s is not a directory\n", argv[1]);
   }
   return 0;
}
```

```
lab11 ♡ l
total 36K
drwxr-xr-x  3 ghost ghost 4.0K Apr 10 19:22 .
drwxr-xr-x 13 ghost ghost 4.0K Apr 10 19:15 ..
-rwxr-xr-x  1 ghost ghost  17K Apr 10 19:21 a.out
drwxr-xr-x  2 ghost ghost 4.0K Apr 10 19:21 hello
-rw-r--r--  1 ghost ghost  519 Apr 10 19:22 sample1.c
lab11 ♡ stat sample1.c
  File: sample1.c
  Size: 519             Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d      Inode: 18752385    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   ghost)   Gid: ( 1000/   ghost)
Access: 2019-04-10 19:23:26.325340945 -0400
Modify: 2019-04-10 19:22:04.442223580 -0400
Change: 2019-04-10 19:22:04.445556904 -0400
 Birth: -
lab11 ♡ stat hello
  File: hello
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 803h/2051d      Inode: 18752383    Links: 2
Access: (0755/drwxr-xr-x)  Uid: ( 1000/   ghost)   Gid: ( 1000/   ghost)
Access: 2019-04-10 19:21:45.605606567 -0400
Modify: 2019-04-10 19:21:45.605606567 -0400
Change: 2019-04-10 19:21:45.605606567 -0400
 Birth: -
lab11 ♡ ./a.out sample1.c
sample1.c is not a directory%
lab11 ♡ ./a.out hello
hello is a directory%
lab11 ♡
```

2. a. This program opens a pointer to the current directory (denoted by `"."` and performs a while loop and reads the filename of each value in the directory and prints out the values to the console.

   b.

```
[fountaia@eos01 lab11]$ ls -l
total 24
-rwxr-xr-x 1 fountaia users 16896 Apr 11 08:34 a.out
-rw-r--r-- 1 fountaia users   393 Apr 11 08:34 sample2.c
[fountaia@eos01 lab11]$ ./a.out
.
Size: 28 bytes
..
Size: 64 bytes
sample2.c
Size: 393 bytes
a.out
Size: 16896 bytes
[fountaia@eos01 lab11]$
```

3. a. Depth first search tree traversal. This is evident in the layout featuring the sizes of subfolders of the parent first, then the parent, which validates that it accesses the root node and its data before moving onward.

b. As evidenced by the man page, the block size for `du` is `1024`.

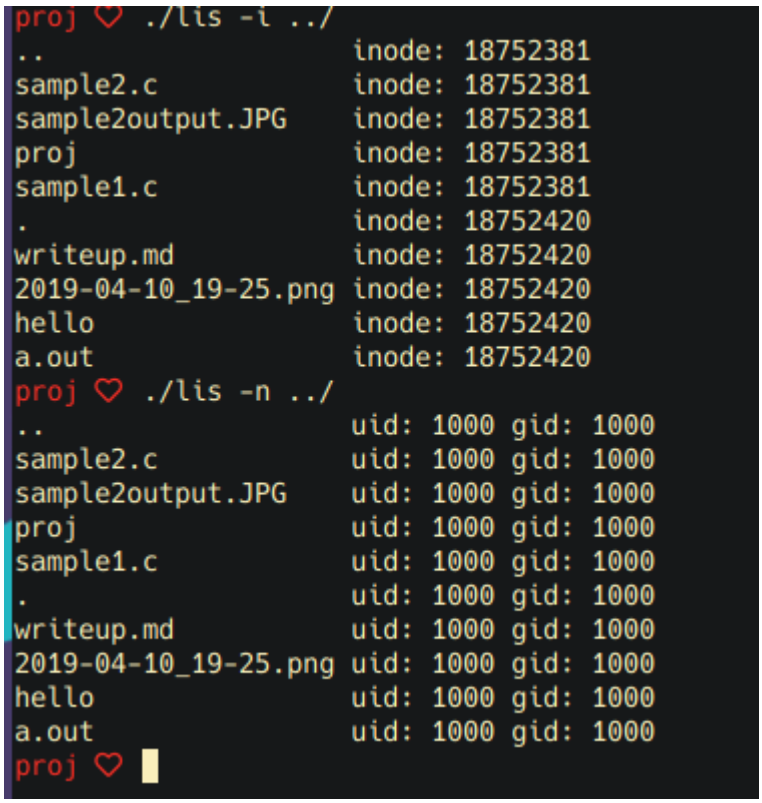
c. Reporting data size in terms of the block size allows the user to get an idea not only how many blocks the system is utilizing for its storage, but also can inform the user of this number as well in a succinct way.

Programming Assignment

```
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>

#define DLIST_LEN 1024
#define MIN(x, y) ((x) < (y)) ? (x) : (y)

int comp(char* n1, char* n2) {
  size_t l1 = strlen(n1);
  size_t l2 = strlen(n2);
  size_t min = MIN(l1, l2);

  for (int i = 0; i < min; ++i) {
    if (n1[i] != n2[i]) {
      char ch1 = n1[i] - 'a';
      char ch2 = n2[i] - 'a';
      if (ch1 < ch2) return 1;
    }
  }

  return l1 == min ? 1 : 0;
}

void merge(char* dlist[], int l, int m, int r) {
  int s1 = (m - 1) - l;
  int s2 = r - m;

  char *L[s1], *R[s2];
  for (int i = 0; i < s1; ++i) {
    L[i] = dlist[i];
  }

  for (int i = m; i < s2; ++i) {
    R[i] = dlist[i];
  }

  int i, j, k;
  for (i = 0, j = 0, k = l; i < s1 && j < s2; ++k) {
    if (comp(L[i], R[i])) {
```

```
      dlist[k] = L[i];
      ++i;
    } else {
      dlist[k] = R[j];
      ++j;
    }
  }

  while (i < s1) {
    dlist[k] = L[i];
    ++i;
    ++k;
  }

  while (j < s1) {
    dlist[k] = L[j];
    ++j;
    ++k;
  }
}

void sort(char* dlist[], int l, int r) {
  int m = l + (r - l) / 2;

  sort(dlist, l, m);
  sort(dlist, m, r);

  merge(dlist, l, m, r);
}

int main(int argc, char** argv) {
  DIR *dir_ptr;
  struct dirent *entry_ptr;
  struct stat stat_buf;
  char dlist[DLIST_LEN];

  int opt;

  char* directory = argv[2];
  if (stat(directory, &stat_buf) < 0) {
    perror("Invalid input supplied");
    return -1;
  }

  int i = 0;
  while ((opt = getopt(argc, argv, "n:i")) != -1) {
    switch(opt) {
      case 'n':
        dir_ptr = opendir(directory);
        while ((entry_ptr = readdir(dir_ptr))) {
          struct stat st;
          char entry[DLIST_LEN];
          stat((entry_ptr->d_name), &st);
```

```
          printf("%-20s uid: %d gid: %d\n", entry_ptr->d_name, st.st_uid, st.st_gid);
          snprintf(entry, "%-20s uid: %d gid: %d\n", entry_ptr->d_name, st.st_uid,
st.st_gid, DLIST_LEN);
          dlist[i] = entry;
          ++i;
        }
        sort(dlist, 0, i);
        for (int j = 0; j < i - 1; ++j) {
            printf("%s", dlist[i]);
        }
        break;
      case 'i':
        dir_ptr = opendir(directory);
        while ((entry_ptr = readdir(dir_ptr))) {
          struct stat st;
          stat((entry_ptr->d_name), &st);
          printf("%-20s inode: %lu\n", entry_ptr->d_name, st.st_ino);
          snprintf(entry, "%-20s inode: %lu\n", entry_ptr->d_name, st.st_ino DLIST_LEN);
          dlist[i] = entry;
          ++i;
        }
        sort(dlist, 0, i);
        for (int j = 0; j < i - 1; ++j) {
            printf("%s", dlist[i]);
        }
        break;
      case '?':
        printf("invalid option specified");
        break;
      default:
        dir_ptr = opendir("./");
        while ((entry_ptr = readdir(dir_ptr))) {
          struct stat st;
          stat((entry_ptr->d_name), &st);
          printf("%-20s\n", entry_ptr->d_name);
        }
    }
  }
}
```


proj ♡ ./lis -i ../
.. inode: 18752381
sample2.c inode: 18752381
sample2output.JPG inode: 18752381
proj inode: 18752381
sample1.c inode: 18752381
. inode: 18752420
writeup.md inode: 18752420
2019-04-10_19-25.png inode: 18752420
hello inode: 18752420
a.out inode: 18752420
proj ♡ ./lis -n ../
.. uid: 1000 gid: 1000
sample2.c uid: 1000 gid: 1000
sample2output.JPG uid: 1000 gid: 1000
proj uid: 1000 gid: 1000
sample1.c uid: 1000 gid: 1000
. uid: 1000 gid: 1000
writeup.md uid: 1000 gid: 1000
2019-04-10_19-25.png uid: 1000 gid: 1000
hello uid: 1000 gid: 1000
a.out uid: 1000 gid: 1000
proj ♡