Measuring directory size using stat system call

Kaynak: http://web.eecs.utk.edu/~plank/plank/classes/cs360/

Recursive directory traversal

- This lecture covers the writing of a command **prsize**
- What **prsize** does is return the number of bytes taken up by all files reachable from the current directory (excluding soft links).
- Prsize illustrates using **opendir/readdir/closedir**, **stat**, recursion, building path names, and finding hard links.

prsize1

Open working (current) directory

Get inode information of selected file

Get file size from the inode infomation.

File Edit View Search Tools Documents Help

Open Save Undo

prsize1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>
main()
{
  DIR *d;
  struct dirent *de;
  struct stat buf;
  int exists;
  int total_size;

  d = opendir(".");
  if (d == NULL) {
    perror("prsize");
    exit(1);
  }
  total_size = 0;
  for (de = readdir(d); de != NULL; de = readdir(d)) {
    exists = stat(de->d_name, &buf);
    if (exists < 0) {
      fprintf(stderr, "Couldn't stat %s\n", de->d_name);
    } else {
      total_size += buf.st_size;
    }
  }
  closedir(d);
  printf("%d\n", total_size);
}
```

C Tab Width: 8 Ln 22, Col 4 INS

prsize1

We temporarily added current directory to PATH so that we can able to use ***prsize*** programs in other directories.

```
File  Edit  View  Search  Terminal  Help
abc:3_Prsize$ ./prsize1
367799
abc:3_Prsize$ prsize1
prsize1: command not found
abc:3_Prsize$ PATH=$PATH:$(pwd)
abc:3_Prsize$ prsize1
367799
abc:3_Prsize$ mkdir test1
abc:3_Prsize$ cd test1
abc:test1$ prsize1
8192
abc:test1$
```

prsize2 function defition

The program that lists the file sizes has been defined as a function. It will help us for recursive directory traversal.

prsize2.c ×

```c
int get_size(char *fn)
{
  DIR *d;
  struct dirent *de;
  struct stat buf;
  int exists;
  int total_size;

  d = opendir(fn);
  if (d == NULL) {
    perror("prsize");
    exit(1);
  }

  total_size = 0;

  for (de = readdir(d); de != NULL; de = readdir(d)) {
    exists = stat(de->d_name, &buf);
    if (exists < 0) {
      fprintf(stderr, "Couldn't stat %s\n", de->d_name);
    } else {
      total_size += buf.st_size;
    }

  closedir(d);
  return total_size;
}

main()
{
  printf("%d\n", get_size("."));
}
```

C ▾ Tab Width: 8 ▾ Ln 43, Col 2 INS

prsize3 : recursive implementation

- Whenever we encounter a directory, we want to find out the size of everything in that directory, so we will call get_size() recursively on that directory.
- S_ISDIR() checks if a file directory or not.
- If it is a directory, all the files it contains should be measured.
- But this results in the error you see below.

```
File Edit View Search Terminal Help
test:prsize3
prsize: Too many open files
test:
```

```c
int get_size(char *fn)
{
  DIR *d;
  struct dirent *de;
  struct stat buf;
  int exists;
  int total_size;

  d = opendir(fn);
  if (d == NULL) {
    perror("prsize");
    exit(1);
  }

  total_size = 0;

  for (de = readdir(d); de != NULL; de = readdir(d)) {
    exists = stat(de->d_name, &buf);
    if (exists < 0) {
      fprintf(stderr, "Couldn't stat %s\n", de->d_name);
    } else {
      total_size += buf.st_size;
    }
    /* Make the recursive call if the file is a directory */
    if (S_ISDIR(buf.st_mode)) {
      total_size += get_size(de->d_name);
    }
  }
  closedir(d);
  return total_size;
}

main()
{
  printf("%d\n", get_size("."));
}
```

C ▾ Tab Width: 8 ▾ Ln 22, Col 22 INS

prsize3a: analysis of error

- I put a print statement into prsize3a.c to see when it's making the recursive calls:
- When the program is run (./prsize3a) repeatedly calls the "." (current) directory.
- I other words it calls itself infinitely.
- This goes into an infinite loop until you run out of open file descriptors at which point opendir() fails.

File Edit View Search Tools Documents Help

Open Save Undo

prsize3a.c

```
total_size = 0;

for (de = readdir(d); de != NULL; de = readdir(d)) {
  exists = stat(de->d_name, &buf);
  if (exists < 0) {
    fprintf(stderr, "Couldn't stat %s\n", de->d_name);
  } else {
    total_size += buf.st_size;
  }
  /* Make the recursive call if the file is a directory */
  if (S_ISDIR(buf.st_mode)) {
    printf("Making recursive call on directory %s\n", de->d_name);
    total_size += get_size(de->d_name);
  }

closedir(d);
return total_size;

main()
```

C Tab Width: 8 Ln 38, Col 42 INS

File Edit View Search Terminal Help

```
Making recursive call on directory .
Making recursive call on directory .
Making recursive call on directory .
Making recursive call on directory .
Making recursive call on directory .
Making recursive call on directory .
prsize: Too many open files
```

prsize4 : excluding "." and ".."directories

- If we exclude "." and ".." inside recursive loop program doen't go into infinite loop.

File Edit View Search Tools Documents Help

Open Save Undo

prsize4.c

```c
for (de = readdir(d); de != NULL; de = readdir(d)) {
  exists = stat(de->d_name, &buf);
  if (exists < 0) {
    fprintf(stderr, "Couldn't stat %s\n", de->d_name);
  } else {
    total_size += buf.st_size;
  }
  /* Make the recursive call if the file is a directory and is not
     . or .. */
  if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 &&
      strcmp(de->d_name, "..") != 0) {
    total_size += get_size(de->d_name);
  }
}
closedir(d);
return total_size;
```

C Tab Width: 8 Ln 3

File Edit View Search Terminal Help

```
test4:ls -la
total 20
drwxr-xr-x 2 root root 4096 Mar 28 21:44 .
drwx------ 6 bilg bilg 4096 Mar 28 21:43 ..
-rw-r--r-- 1 root root    9 Mar 28 21:44 f1.txt
-rw-r--r-- 1 root root   18 Mar 28 21:44 f2.txt
-rw-r--r-- 1 root root   51 Mar 28 21:44 f3.txt
test4:prsize4
8270
test4:
```

prsize4 : path problem

- We solved infinite loop problem but there remains another problem with file paths.
- It is necessary to write full path starting from «.» (working directory) before doing a recursive call.

```
File Edit View Search Terminal Help
test4:touch altklasor/deneme.txt
test4:ls
altklasor  f1.txt  f2.txt  f3.txt
test4:prsize4
Couldn't stat deneme.txt
prsize: No such file or directory
test4:cd altklasor
altklasor:ls
deneme.txt
altklasor:
```

prsize5: solution to path problem

256 character file name + 2 character (/ and null) + file path(strlen(fn))

Print the file name to *s variable* together with file path.

File Edit View Search Tools Documents Help

Open Save Undo

prsize5.c

```c
char *s;
d = opendir(fn);
if (d == NULL) {
  perror("prsize");
  exit(1);
}
s = (char *) malloc(sizeof(char)*(strlen(fn)+258));

for (de = readdir(d); de != NULL; de = readdir(d)) {
  /* Look for fn/de->d_name */
  sprintf(s, "%s/%s", fn, de->d_name);
  exists = stat(s, &buf);
  if (exists < 0) {
    fprintf(stderr, "Couldn't stat %s\n", s);
  } else {
    total_size += buf.st_size;
  }
  if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 &&
      strcmp(de->d_name, "..") != 0) {
    total_size += get_size(s);
  }
}
closedir(d);
```

Saving file '/home/bilg/Documents/h... C Tab Width: 8 Ln 20, Col 11 INS

prsize5: repeating files problem

- Prsize5 computed the size of directory as 20558.
- However, the files with some inode numbers are taken into account two times.
- What we need is for **prsize** to be able to recognize hard links, and only count them once.
- How do you recognize whether two files are links to the same disk file?
- You use the inode number. This is held in **buf.st_ino**.

```
File Edit View Search Terminal Help
test4:prsize5
20558
test4:ls -lai
total 24
1840938 drwxr-xr-x 3 root root 4096 Mar 28 22:06 .
1840034 drwx------ 6 bilg bilg 4096 Mar 28 22:18 ..
1839719 drwxr-xr-x 2 root root 4096 Mar 28 22:06 altklasor
1841007 -rw-r--r-- 1 root root    9 Mar 28 21:44 f1.txt
1841008 -rw-r--r-- 1 root root   18 Mar 28 21:44 f2.txt
1841009 -rw-r--r-- 1 root root   51 Mar 28 21:44 f3.txt
test4:ls -lai altklasor
total 8
1839719 drwxr-xr-x 2 root root 4096 Mar 28 22:06 .
1840938 drwxr-xr-x 3 root root 4096 Mar 28 22:06 ..
1840932 -rw-r--r-- 1 root root    0 Mar 28 22:06 deneme.txt
test4:
```

prsize6: solution to repeating files problem

- In order to prevent re-use of the same folders, the inode number of each file included in the calculation can be kept in a BST and used for control purposes.
- The red-black tree in the libfdr library is used to keep the inode list.
- It prevents previously used inode for getting the size of a file to be added to the tree again

File Edit View Search Tools Documents Help

Open Save Undo

prsize6.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <sys/stat.h>
#include "jrb.h"

int get_size(char *fn, JRB inodes)
{
  DIR *d;
  struct dirent *de;
  struct stat buf;
  int exists;
  int total_size;
  char *s;

  d = opendir(fn);
  if (d == NULL) {
    perror("prsize");
    exit(1);
  }

  total_size = 0;
  s = (char *) malloc(sizeof(char)*(strlen(fn)+258));

  for (de = readdir(d); de != NULL; de = readdir(d)) {
    /* Look for fn/de->d_name inodes*/
    sprintf(s, "%s/%s", fn, de->d_name);
    exists = stat(s, &buf);
```

C Tab Width: 8 Ln 12, Col 1 INS

prsize6: solution to repeating files problem

prsize6.c ×

```
  s = (char *) malloc(sizeof(char)*(strlen(fn)+258));

  for (de = readdir(d); de != NULL; de = readdir(d)) {
    /* Look for fn/de->d_name inodes*/
    sprintf(s, "%s/%s", fn, de->d_name);
    exists = stat(s, &buf);
    if (exists < 0) {
      fprintf(stderr, "Couldn't stat %s\n", s);
    } else {
      if (jrb_find_int(inodes, buf.st_ino) == NULL) {
        total_size += buf.st_size;
        jrb_insert_int(inodes, buf.st_ino, JNULL);
      }
    }
    if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 &
        strcmp(de->d_name, "..") != 0) {
      total_size += get_size(s, inodes);
    }
  }
  closedir(d);
  free(s);
  return total_size;
}

main()
{
  JRB inodes;
  inodes = make_jrb();
  printf("%d\n", get_size(".", inodes));
}
```

C ▾ Tab Width: 8 ▾ Ln 12, Col 1 INS

- Search for the inode in tree. Insert if not in the tree.
- Inode is added as key to prevent repetition.
- We don't use value (value=JNULL)

prsize7.c: soft (symbolic) link problem

We used the lstat () system call for soft links.

prsize7.c

```c
s = (char *) malloc(sizeof(char)*(strlen(fn)+258));

for (de = readdir(d); de != NULL; de = readdir(d)) {
  /* Look for fn/de->d_name */
  sprintf(s, "%s/%s", fn, de->d_name);
  exists = lstat(s, &buf);
  if (exists < 0) {
    fprintf(stderr, "Couldn't stat %s\n", s);
  } else {
    if (jrb_find_int(inodes, buf.st_ino) == NULL) {
```

The result using stat()

The result using lstat()

```
test4:ln -s f1.txt softf1
test4:ls
altklasor  f1.txt  f2.txt  f3.txt  softf1
test4:prsize6
12366
test4:prsize7
12372
test4:ls -li
total 16
1839719 drwxr-xr-x 2 root root 4096 Mar 28 22:06 altklasor
1841007 -rw-r--r-- 1 root root    9 Mar 28 21:44 f1.txt
1841008 -rw-r--r-- 1 root root   18 Mar 28 21:44 f2.txt
1841009 -rw-r--r-- 1 root root   51 Mar 28 21:44 f3.txt
1841073 lrwxrwxrwx 1 root root    6 Mar 29 00:36 softf1 -> f1.txt
test4:
```

prsize7a: printing visited paths

```
File  Edit  View  Search  Terminal  Help
abc:1_Prsize$ ./directory 10
abc:1_Prsize$ ./prsize7a
Testing .
Testing ./d10
Testing ./d10/d9
Testing ./d10/d9/d8
Testing ./d10/d9/d8/d7
Testing ./d10/d9/d8/d7/d6
Testing ./d10/d9/d8/d7/d6/d5
Testing ./d10/d9/d8/d7/d6/d5/d4
Testing ./d10/d9/d8/d7/d6/d5/d4/d3
Testing ./d10/d9/d8/d7/d6/d5/d4/d3/d2
Testing ./d10/d9/d8/d7/d6/d5/d4/d3/d2/d1
Testing ./test1
413002
```

- *Program çalıştırıldığı klasörden ("./") başlayıp tüm alt klasörlere aynı işlemi uyguluyor.*
- *Bir alt klasöre gidilirken önceki klasör açık kalmasıdır.*

```
Open ▾   directory  ~/Documents/h11/1_Prsize   Save
#!/bin/sh
x=$1
while [ $x -gt 0 ]
do
        mkdir d$x
        cd d$x
        echo "test" > f$x
        x=`expr $x - 1`
done
sh ▾  Tab Width: 8 ▾  Ln 1, Col 8  INS
```

prsize8: closing a directorybefore visiting another

- To reduce the number of files opened from a process, the directory can be closed before going to subfolders.
- In this implementation, subdirectories in the visited directory are added to the list.
- Then directory is closed and the subdirectories are visited.
- The same is true for every visited directory.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <sys/stat.h>
#include "jrb.h"
#include "dllist.h"

int get_size(char *fn, JRB inodes)
{
  DIR *d;
  struct dirent *de;
  struct stat buf;
  int exists;
  int total_size;
  char *s;
  Dllist directories, tmp;

  d = opendir(fn);
  if (d == NULL) {
    perror("prsize");
    exit(1);
  }

  total_size = 0;
  s = (char *) malloc(sizeof(char)*(strlen(fn)+258));

  directories = new_dllist();
  for (de = readdir(d); de != NULL; de = readdir(d)) {
    /* Look for fn/de->d_name */
```

C ▾ Tab Width: 8 ▾ Ln 54, Col 34 INS

prsize8: closing a directorybefore visiting another

Add to the list if there are subdirectories

Close directory before going to another one

```
  directories = new_dllist();
  for (de = readdir(d); de != NULL; de = readdir(d)) {
    /* Look for fn/de->d_name */
    sprintf(s, "%s/%s", fn, de->d_name);
    exists = lstat(s, &buf);
    if (exists < 0) {
      fprintf(stderr, "Couldn't stat %s\n", s);
    } else {
      if (jrb_find_int(inodes, buf.st_ino) == NULL) {
        total_size += buf.st_size;
        jrb_insert_int(inodes, buf.st_ino, JNULL);
      }
    }
    if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 &&
        strcmp(de->d_name, "..") != 0) {
      dll_append(directories, new_jval_s(strdup(s)));
    }
  }
  closedir(d);
  dll_traverse(tmp, directories) {
     total_size += get_size(tmp->val.s, inodes);
     /* This keeps the program from overgrowing its memory */
     free(tmp->val.s);
  }

  /* As does this */
  free_dllist(directories);
  free(s);
  return total_size;
}

main()
{
  JRB inodes;
  inodes = make_jrb();
  printf("%d\n", get_size(".", inodes));
```

Clean after getting size

C ▾ Tab Width: 8 ▾ Ln 67, Col 14 INS