CS360 Lecture notes -- Prsize: recursive directory traversal

- Jim Plank
- Directory: **/blugreen/homes/plank/cs360/notes/Prsize**
- Lecture notes: **http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/Prsize/lecture.html**

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). It is a good program as it illustrates using **opendir/readdir/closedir**, **stat**, recursion, building path names, and finding hard links.

First, I wrote **prsize1.c**. This prints the total size of all files in the current directory. It is a simple use of stat and **opendir/readdir/closedir**. Test it out on the directory **test1**. Go into a clean directory of your own, and do the following:

```
UNIX> cp ~huangj/cs360/notes/Prsize/*.c .
UNIX> cp ~huangj/cs360/notes/Prsize/makefile .
UNIX> make
...
UNIX> setenv PRDIR `pwd`
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize1
2074
UNIX> ls -la
drwxr-xr-x  3 huangj         512 Sep 23 10:22 .
drwxr-xr-x  7 huangj        1024 Sep 23 10:37 ..
drwxr-xr-x  2 huangj         512 Sep 23 10:22 d1
-rw-r--r--  1 huangj          11 Sep 23 10:22 f1
-rw-r--r--  1 huangj          15 Sep 23 10:22 f2
UNIX> dc
512 1024 + 512 + 11 + 15 + p
2074
q
UNIX>
```

The "**setenv**" line sets it up so that you can call **prsize1** from any directory. So, as you can see from the "**ls -l**" and the "**dc**", it sums up the size from all the files in the directory "**test1**". Now, the next step we'd like to take is to get the program to sum up the sizes of all files *reachable* from the current directory. To do this, we need to make the program recursive. Instead of putting all our code in the **main()** routine, we'll instead bundle it into a function, and call that function. **Prsize2.c** does this. It provides the same functionality as **prsize1.c**, except that it makes a call to **get_size()** to find the size. Note there is no recursion yet -- that is for prsize3.c. If you test **prsize2**, you'll see that it does the same thing as **prsize1**.

```
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize2
2074
UNIX>
```

Now, we want to make **prsize2** recursive. 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. This is done in **prsize3.c**. Try it out on the **~huangj/cs360/notes/Prsize/test1** directory:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize3
prsize: Too many open files
UNIX>
```

So, what's happening? Well, to check, I put a print statement into **prsize3a.c** to see when it's making the recursive calls:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize3a
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
UNIX>
```

Now you can see what's happening. When enumerating files in ".", you come across the file ".". This is a directory, so you make a recursive call on it. This goes into an infinite loop until you run out of open file discriptors at which point **opendir()** fails. To fix this, you need to check and see whether or not you are trying to make a recursive call to the "." directory. You need to check for ".." as well. **Prsize4.c** puts in this code. Now try it out:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize4
Couldn't stat f3
prsize: No such file or directory
UNIX>
```

Ok, now what's the problem? Well, the program is trying to stat **f3** in the directory **d1**, but it's not working in the directory **d1**. In other words, **prsize3** is called from the directory **~huangj/cs360/notes/Prsize/test1**, and makes the call "**exists = stat("f3", &buf)**". Of course stat is going to return -1, because there is no file **f3** in the directory. Instead, we need to look for "**d1/f3**". In other words, our code has a bug -- we need to be looking for **fn/de->d_name** in **get_size()**, and not just **de->d_name**. **Prsize5.c** makes this change.

```
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize5
3115
```

So, this looks ok, except there's still something wrong:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> ls -la
total 5
drwxr-xr-x  3 huangj         512 Sep 23 10:22 .
drwxr-xr-x  7 huangj        1024 Sep 23 10:37 ..
drwxr-xr-x  2 huangj         512 Sep 23 10:22 d1
-rw-r--r--  1 huangj          11 Sep 23 10:22 f1
```

```
-rw-r--r--  1 huangj          15 Sep 23 10:22 f2
UNIX> ls -la d1
total 3
drwxr-xr-x  2 huangj         512 Sep 23 10:22 .
drwxr-xr-x  3 huangj         512 Sep 23 10:22 ..
-rw-r--r--  1 huangj          17 Sep 23 10:22 f3
UNIX> dc
512 1024 + 512 + 11 + 15 + 17 + p
2091
512 1024 + 512 + 11 + 15 + 512 + 512 + 17 + p
3115
q
UNIX>
```

As you can see, **prsize5** is counting **d1** and **d1/.** as separate files, and adding both of their sizes into the total. Same for **.** and **d1/..**

This is a drag. To be clearer, look in **test2**:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test2
UNIX> ls -la
drwxr-xr-x  2 huangj         512 Sep 23 10:26 .
drwxr-xr-x  7 huangj        1024 Sep 23 10:37 ..
-rw-r--r--  2 huangj          11 Sep 23 10:22 f4
-rw-r--r--  2 huangj          11 Sep 23 10:22 f4-hard-link
UNIX> $PRDIR/prsize5
1558
UNIX> dc
512 1024 + 11 + 11 + p
1558
q
UNIX>
```

The files **f4** and **f4-hard-link** are links to the same file. However, **prsize5** counts them as being different. So, 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**.

Now, the way we check for duplicate inodes is to maintain a rb-tree of inodes that we have seen so far. Before adding in the size of any file, we check to see if its inode is in the rb-tree. If so, we do nothing. Otherwise, we add in the size, and put the inode into the rb-tree. As inodes are ints, we can use **jrb_insert_int** and **jrb_find_int** to access and modify the red-black tree. The code is in **prsize6.c**.

```
UNIX> cd ~huangj/cs360/notes/Prsize/test2
UNIX> $PRDIR/prsize6
1547
UNIX> cd ~huangj/cs360/notes/Prsize/test1
UNIX> $PRDIR/prsize6
2091
```

Now, soft links present a small problem. Look at the **test3** directory.

```
UNIX> cd ~huangj/cs360/notes/Prsize/test3
UNIX> ls -la
drwxr-xr-x  2 huangj         512 Sep 23 10:26 .
drwxr-xr-x  7 huangj        1024 Sep 23 10:37 ..
-rw-r--r--  1 huangj          11 Sep 23 10:22 f5
lrwxrwxrwx  1 huangj           2 Sep 23 10:26 f5-soft-link -> f5
lrwxrwxrwx  1 huangj           1 Sep 23 10:24 soft-link-to-. -> .
UNIX> $PRDIR/prsize6
Couldn't stat ./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-t
```

So, what has happened? Since we're using **stat()**, **prsize6** doesn't recognize soft links, and thus we have the same infinite loop problem as before. It should be clear what we want -- instead of traversing the link to ".", we want **prsize** to count the size of the link itself (2 bytes for **f5-soft-link** and 1 byte for **soft-link-to-.**). Thus, all we need to do in **prsize7.c** is use **lstat()** instead of **stat()**. This gives information about the soft link itself, instead of the file to which the link points:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test3
UNIX> ls -la
drwxr-xr-x  2 huangj         512 Sep 23 10:26 .
drwxr-xr-x  7 huangj        1024 Sep 23 10:37 ..
-rw-r--r--  1 huangj          11 Sep 23 10:22 f5
lrwxrwxrwx  1 huangj           2 Sep 23 10:26 f5-soft-link -> f5
lrwxrwxrwx  1 huangj           1 Sep 23 10:24 soft-link-to-. -> .
UNIX> $PRDIR/prsize7
1550
UNIX> dc
512 1024 + 11 + 2 + 1 + p
1550
UNIX>
```

Finally, there's one more bug in this program. It has to do with open file descriptors. Try **prsize7** on the **test4** directory:

```
UNIX> cd ~huangj/cs360/notes/Prsize/test4
UNIX> $PRDIR/prsize7
prsize: Too many open files
UNIX>
```

What's going on? To figure it out, I put in a print statement at each call to **get_size** in **prsize7a.c**.

```
UNIX> cd ~huangj/cs360/notes/Prsize/test4
UNIX> $PRDIR/prsize7a
Testing .
Testing ./1
Testing ./1/2
Testing ./1/2/3
Testing ./1/2/3/4
Testing ./1/2/3/4/5
...
```

```
Testing ./1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19/20/21/22/23/24/25/26/27/28/29/30/31/32/33/34/35/36/37/38/39/40/41/42/43/44/45/46/47/48/49
prsize: Too many open files
UNIX>
```

What's happening is that the recursive calls to **get_size()** are made in between the **opendir()** and **closedir()** calls. That means that each time we make a recursive call, we add one to the number of open files. As Unix only allows a finite number of open files to be held by any one process, we get an error if we make too many nested recursive calls. The solution to this is to make sure that there are no open files when we make the recursive call. How do we do this? When enumerating the files in a directory, we put all directories into a dlist, and then after closing the directory file, we traverse the list and make the recursive calls. Note that we need to do a **strdup()** when we put the directories into the dlist. Why? Think it over, or see what happens when you don't do it, and you try run the program on the **test5** directory.

The correct and final version of prsize is in **prsize8.c**.

```
UNIX> cd ~huangj/cs360/notes/Prsize/test4
UNIX> $PRDIR/prsize8
33792
UNIX> cd test5
$PRDIR/prsize8
2656
UNIX>
```