1. wget http://web.cs.ucla.edu/classes/fall18/cs35L/assign/coreutils-with-bug.tar.gz
2. tar –xvf coreutils-with-bug.tar.gz to unzip the file
3. I then used cd to get into the coreutils-with-bug folder
4. I then followed this by the ./configure command

The ./configure command stopped at several places and was slow, but everything was still properly configured.

1. To build this version of coreutils as is, I ran the make and then make install command. The following error output was given:

In file included from **utimecmp.c:41:0**:

**utimens.h:2:5:** **error:** conflicting types for '**futimens**'

 int futimens (int, char const \*, struct timespec const [2]);

**^**

In file included from **utimecmp.h:25:0**,

                 from **utimecmp.c:25**:

**/usr/include/sys/stat.h:373:12:** **note:** previous declaration of '**futimens**' was here

 extern int futimens (int \_\_fd, const struct timespec \_\_times[2]) \_\_THROW;

**^**

make[3]: \*\*\* [utimecmp.o] Error 1

make[3]: Leaving directory `/w/home.17/cs/ugrad/enika/coreutils-with-bug/lib'

make[2]: \*\*\* [all] Error 2

make[2]: Leaving directory `/w/home.17/cs/ugrad/enika/coreutils-with-bug/lib'

make[1]: \*\*\* [all-recursive] Error 1

make[1]: Leaving directory `/w/home.17/cs/ugrad/enika/coreutils-with-bug'

make: \*\*\* [all] Error 2

The main error is the red colored error message which states: “conflicting types for ‘futimens’”. ‘futimens’ seems to have been declared in a header file called utimens.h. The main error seems to be that another function named futimens has also been declared with the same exact parameter names as the original futimens declared in utimens.h. Thus, this redefinition of futimens will create an error. Additionally, futimens function has two parameters int and char const\*, which are not valid declarations.

To solve this a problem, a patch can be applied. To download the patch:

Wget <https://web.cs.ucla.edu/classes/fall18/cs35L/assign/coreutils.diff>

Then within coreutils-with-bug:

Patch –p0 < coreutils.diff

I found out from the man patch page that –p0 was needed since the patch file is run from the root directory of coreutils-with-bug.

1. In the coreutils-with-bug folder (the patch folder), I ran the following commands:

$ tmp=$(mktemp -d)

$ cd $tmp

$ touch -d '1918-11-11 11:00 GMT' wwi-armistice

$ touch now

$ sleep 1

$ touch now1

$ TZ=UTC0 /u/cs/ugrad/enika/coreutils-with-bug/src/ ls -lt --full-time wwi-armistice now now1

This resulted in the output:

-rw-r--r-- 1 enika csugrad 0 1918-11-11 11:00:00.000000000 +0000 wwi-armistice

-rw-r--r-- 1 enika csugrad 0 2018-10-26 00:03:02.854254861 +0000 now1

-rw-r--r-- 1 enika csugrad 0 2018-10-26 00:02:41.470510975 +0000 now

As can be seen by the first line with 1918, ls –t mishandles the timestamp of files far in the past.

Then I navigated into gdb using

Gdb /u/cs/ugrad/enika/coreutils-with-bug/src/ls

Within the tmp directory

After this I typed in gdb info functions to see all the possible function headers. The first function I noticed which may cause a problem was the sort\_files() function within main. It only makes sense to check this function since the error lies in the sorting of the files.

Then my first breakpoint was set at sort\_files() using the following command:

break sort\_files()

After I set the breakpoint I ran the command:

Run –lt

Once the breakpoint was reached I stepped into the function sort\_files() by using the s command multiple times to observe the kind of output I was getting:

If(! Setjmp (failed\_strcoll))

Switch (sort\_type)

Switch(time\_type)

Func = sort\_reverse ? rev\_cmp\_mtime: compare\_mtime;

Break;

Break;

Qsort (files, files\_index, sizeof \*files, func);

From the above output, it is evident that the sort\_type is sort\_time and the time\_type is time\_mtime.

To find the value of func, I typed in p func and found out that the value of func is compare\_mtime. To set a breakpoint at compare\_mtime, I used the following command:

Break compare\_mtime

Continue

This results in the output:

static int compare\_mtime (V a, V b) { return cmp\_mtime (a, b, xstrcoll); }

From here we have to look into cmp\_mtime() function. This time I typed in the command:

list cmp\_mtime

This gave me the following result:

static inline int

cmp\_mtime (struct fileinfo const \*a, struct fileinfo const \*b,

           int (\*cmp) (char const \*, char const \*))

{

  int diff = timespec\_cmp (get\_stat\_mtime (&b->stat),

                           get\_stat\_mtime (&a->stat));

  return diff ? diff : cmp (a->name, b->name);

}

This is nothing but the function cmp\_mtime. From the above output, I realized that I need to also look at the timespec\_cmp function. Before that, however, I set a breakpoint at cmp\_mtime and looked at the local variables to see if there was anything strange. There wasn’t, so again I typed in:

list timespec\_cmp

This results in the output:

  /\* Return negative, zero, positive if A < B, A == B, A > B, respectively.

44   Assume the nanosecond components are in range, or close to it.  \*/

45 static inline int

46 timespec\_cmp (struct timespec a, struct timespec b)

47 {

48   int diff = a.tv\_sec - b.tv\_sec;

49   return diff ? diff : a.tv\_nsec - b.tv\_nsec;

50 }

Then we set another breakpoint at timespec\_cmp:

Break timespec\_cmp

Continue

Next

|  |  |
| --- | --- |
|  |  |

To find the values of the arguments a and b, I used info args:

a = {tv\_sec = -1613826000, tv\_nsec = 0}

b = {tv\_sec = 1493339426, tv\_nsec = 137535585}

Then to find the value of the local variable diff, I used info locals:

Diff = 1187801870

However, this result of diff seems incorrect since a.tv\_sec is negative and b.tv\_sec is positive, meaning that diff should be a negative number (A < B so the result shoule be negative). Thus there must be some kind of integer overflow resulting in this output.

To fix the bug, I went out of gdb and into emacs within the coreutils-with-bugs/lib/timespec.h file. I modified the timespec\_cmp function as can be seen by lab4.diff. Essentially, I modified it so that if a.tv\_sec is less than b.tv\_sec then, a -1 will be returned. If the opposite case is true, then 1 will be returned. Finally, if the two values are equal, the nanosecond values of a and b will be compared.

After timespec.h has been modified, I generated the diff file inside of coreutils-with-bug/lib with the following command:

Diff –u timespec.h timespec.h > lab4.diff

2018-10-25  Enika Biswas  <enika@lnxsrv06.seas.ucla.edu>

        \* timespec.h:

           Fxed the integer overflow error by returning a positive 1 if a.tv\_sec is bigger

        than b.tv\_sec and -1 if the opposite case is true. If they are equal, then we

        compare the nanosecond values of a and b.