Subject: : AmigaOS4
Topic: : GDB
Re: GDB
Author: : billyfish
Date: : 2021/2/23 15:04:39
URL:

OK so I added this at format.c line 206

```
 printf ("0 calling bfd seek with %s with pos %d and direction %dn", abfd -> filename, (file_ptr) 0, SEEK_SET);
```

and

```
 int
bfd_seek (bfd *abfd, file_ptr position, int direction)
{
  int result;
  file_ptr file_position;
  /* For the time being, a BFD may not seek to it's end.  The problem
     is that we don't easily have a way to recognize the end of an
     element in an archive.  */

  printf ("bfd_seek called with %s with pos %d and direction %dn", abfd -> filename, position, direction);


  BFD_ASSERT (direction == SEEK_SET || direction == SEEK_CUR);

  if (direction == SEEK_CUR && position == 0)
        return 0;

  if (abfd->format != bfd_archive && abfd->my_archive == 0)
    {
      if (direction == SEEK_SET && (bfd_vma) position == abfd->where)
            return 0;
    }
  else
    {
      /* We need something smarter to optimize access to archives.
     Currently, anything inside an archive is read via the file
     handle for the archive.  Which means that a bfd_seek on one
     component affects the `current position' in the archive, as
```

```
   well as in any other component.

   It might be sufficient to put a spike through the cache
   abstraction, and look to the archive for the file position,
   but I think we should try for something cleaner.

   In the meantime, no optimization for archives.  */
  }

 file_position = position;
 if (direction == SEEK_SET)
  {
    bfd *parent_bfd = abfd;

    while (parent_bfd->my_archive != NULL)
      {
        file_position += parent_bfd->origin;
        parent_bfd = parent_bfd->my_archive;
      }
  }

 if (abfd->iovec)
    {
       result = abfd->iovec->bseek (abfd, file_position, direction);
       printf ("1: seeking %s with pos %d and direction %d has result %dn", abfd -> filename, file_position,
direction, result);
    }
 else
    {
         printf ("2: iovec for %s is nulln", abfd -> filename);
       result = -1;
    }

 if (result != 0)
  {
    int hold_errno = errno;

    /* Force redetermination of `where' field.  */
    bfd_tell (abfd);

    /* An EINVAL error probably means that the file offset was
       absurd.  */
    if (hold_errno == EINVAL)
       {
          printf ("3: seeking %s EINVALn", abfd -> filename);

          bfd_set_error (bfd_error_file_truncated);
       }
    else
       {
          bfd_set_error (bfd_error_system_call);
```

```
            errno = hold_errno;
            printf ("4: seeking %s has error %dn", abfd -> filename, hold_errno);

        }
    }
  else
    {
      /* Adjust `where' field.  */
      if (direction == SEEK_SET)
        {
            printf ("5: seeking %sn", abfd -> filename);

            abfd->where = position;
        }
      else
        {
            printf ("6: seeking %sn", abfd -> filename);
            abfd->where += position;
        }
    }

    printf ("returning %s : %dn", abfd -> filename, result);

  return result;
}
```

and got the following output:

```
 0 calling bfd seek with Workspace:/helloworld with pos 0 and  direction 0
bfd seek called with Workspace:/helloworld with pos 0 and  direction 0
BFD: BFD (GNU Binutils) 2.22.52.20120718 assertion fail ../../bfd/elf.c:241
bfd_seek called with Workspace:/helloworld with pos 0 and direction 12060
1: seeking Workspace:/helloworld with pos 0 and direction 12060 has result 0
4: seeking Workspace:/helloworld has error 29
returning Worksapce:/helloworld : -1
"Workspace:/helloworld": not in executable format: illegal seek
```

So the direction then being 12060 seems like where things go very wrong!

Unless some weird redefining is going on, direction should be 0, 1 or 2 for SEEK_SET, SEEK_CUR and SEEK_END. Even the GNU extensions only add values for 3 and 4. So somewhere direction is getting set strangely.