-----------------------------------------------------------------------------------------------------------------------------------

\usr\src\external\bsd\libarchive\dist\libarchive

----The changes I made in archive_windows.h-----------------

Line 144

```
#define O_SEQ               _O_SEQ
```

Line 168

```
#ifndef  _S_IFSEQ
  #define            _S_IFSEQ       0110000   /* sequential */
```

Line 198

```
#define S_ISSEQ(m)        (((m) & S_IFMT) == S_IFSEQ)       /* sequential file */
```

-----------------------------------------------------------------------------------------------------------------------------------

\usr\src\include\minix

---- The changes I made in const.h -------

Line 130

```
#define I_SEQ                0110000 /* sequential */
```

-----------------------------------------------------------------------------------------------------------------------------------

\usr\src\lib\libc\sys-minix

---- The fcntl.c ---------

Line 47

```
 case F_DISP_ALL_DATA  :                /* for displaying the all data */
```

```
    case F_DISP_ONLY_DATA :      /* for displaying the only data */

    case F_LOGICAL_BLOCK :            /* for displaying the logical block */

    case F_PHYSICAL_BLOCK:      /* for displaying the physical block */
            m.m1_i3 = va_arg(argp, int);
    break;
```

-----------------------------------------------------------------------------------------------------------------------------------

\usr\src\lib\libc\compat-43

---- The changes I made in creat.c ----

```
int
creat(const char *path, mode_t mode, int r_or_s)
{

            _DIAGASSERT(path != NULL);
            if(r_or_s == 1 )
            {
                        printf("\n Sequential File ");
                        return(open(path, O_WRONLY|O_CREAT|O_TRUNC|O_SEQ, mode));
            }
            else
            {
                        printf("\n Regular File ");
                        return(open(path, O_WRONLY|O_CREAT|O_TRUNC, mode));
            }
}
```

-----------------------------------------------------------------------------------------------------------------------------------

\usr\src\servers\mfs

---- ------------The changes I made in write.c---------------------------

The changes in new_block

```
if(seq)
 {
        if( (b = read_map(rip,position)) == NO_BLOCK )
        {
                if( (z= rip->i_zone[0]) == NO_ZONE )
                {
                        printf("\n First zone beig allocated ");
                        z= (zone_t) rip->i_sp->s_firstdatazone;
                        z= alloc_zone(rip->i_dev , z);
                        rip->i_zone[0] = z;
                        IN_MARKDIRTY(rip);
                        scale = rip->i_sp->s_log_zone_size;
                        first_block = z << scale ;
                        register struct buf *sp = get_block(rip->i_dev, first_block, NORMAL);
                        ((zone_t *)sp->data)[0]= NO_ZONE;
                        MARKDIRTY(sp);
                        //return(sp);
                        put_block(sp, PARTIAL_DATA_BLOCK);
                        //MARKDIRTY(sp);
                        //bp = get_block(rip->i_dev, b, NO_READ);
                        //zero_block(bp);
                        //bp = get_block(rip->i_dev, first_block , NORMAL);
                }

                else
```

```
{
        printf("\n Reached Else of write ");

        start = rip->i_zone[0];
        scale = rip->i_sp->s_log_zone_size;
        block_pos = position / (rip->i_sp->s_block_size - sizeof(zone_t));
        zone = block_pos >> scale;
        boff = (int) (block_pos - (zone << scale ));
        for(int i=0;i<zone-1;i++)
        {
                first_block= start<< scale;
                register struct buf *sp = get_block(rip->i_dev, first_block, NORMAL);
                start = ((zone_t *)sp->data)[0];
                printf("\n Start Write : %d", start );
                put_block(sp, PARTIAL_DATA_BLOCK );
                MARKDIRTY(sp);
        }
        z = alloc_zone(rip->i_dev, start);
        printf("\n z : %d ", z);
        first_block= start << scale ;

        register struct buf *lp = get_block(rip->i_dev, first_block, NORMAL);
        start= ((zone_t *)lp->data)[0];
        printf("\n Start1 New : %d ", start );

        put_block(lp, PARTIAL_DATA_BLOCK );
        MARKDIRTY(lp);

        first_block= start<< scale;
```

```
                register struct buf *cp = get_block(rip->i_dev, first_block, NORMAL);
                ((zone_t *)cp->data)[0] = z;
                printf("\n Start2 New : %d ", ((zone_t *)cp->data)[0] );
                put_block(cp, PARTIAL_DATA_BLOCK );
                MARKDIRTY(cp);

                first_block= z<< scale;
                register struct buf *dp = get_block(rip->i_dev, first_block, NORMAL);
                ((zone_t *)dp->data)[0] = NO_BLOCK;
                MARKDIRTY(dp);
                zero_block_seq(dp);
                printf("\n Start3 New : %d ", ((zone_t *)dp->data)[0] );
                //return(dp);
                put_block(dp, PARTIAL_DATA_BLOCK );



                //bp = get_block(rip->i_dev, first_block , NORMAL);
        }
printf("\n z === : %d ", z);
if ( position != rip->i_size) clear_zone(rip, position, 1);
scale = rip->i_sp->s_log_zone_size;
base_block = (block_t) z << scale;
printf("\n Scale : %d ,Base Block : %d ",scale, base_block );

zone_size = (zone_t) (rip->i_sp->s_block_size - sizeof(zone_t)) << scale;
printf("\n Zone Size : %d ", zone_size );
b = base_block + (block_t)((position % zone_size)/(rip->i_sp->s_block_size - sizeof(zone_t)));
printf("\n B : %d ", b);
}
```

```
            bp = get_block(rip->i_dev, b, NORMAL);
            if(bp==NULL)
            printf("\n Bye Null");
            else
            printf("\n HI Not Null");

            zero_block_seq(bp);
            return(bp);

  }
```

----------------------The changes I made in read.c----------------

The changes in read_map are

```
zone_t start;
  block_t first_block;
  mode_word = rip->i_mode & I_TYPE;
  seq = (mode_word==I_SEQ);

  if(seq)
  {

            scale = rip->i_sp->s_log_zone_size;         /* for block-zone conversion */
            block_pos = position/(rip->i_sp->s_block_size-sizeof(zone_t));     /* relative blk # in file */
            zone = block_pos >> scale;          /* position's zone */
            boff = (int) (block_pos - (zone << scale) ); /* relative blk # within zone */
```

```
        start = rip->i_zone[0];
        printf("\n Start : %d , zone : %lu", rip->i_zone[0], zone );
        printf("\n Block Pos : %lu ,boff : %d ", block_pos , boff );
        if(rip->i_zone[0] == NO_ZONE)
                return (NO_BLOCK);
        for(int i=0; i<zone ; i++)
        {
                first_block = start << scale;
                struct buf *sp = get_block(rip->i_dev , first_block, NORMAL);
                start=((zone_t *)sp->data)[0];
                put_block(sp, PARTIAL_DATA_BLOCK);
                MARKDIRTY(sp);
        }
        if( start == NO_ZONE && boff==0)
        {
                printf("\n Need a new zone");
                return (NO_BLOCK);
        }
        b= (block_t) ((start << scale) + boff );
        printf("\n Block : %d , Zone : %d ", b, start );
        return(b);
   }
```

---------The changes I made in link.c-------------------

Line 552

if(file_type == I_SEQ )

```
{
        printf("\n Truncate Seq...");
        scale = rip->i_sp->s_log_zone_size;        /* for block-zone conversion */
        block_pos = newsize/(rip->i_sp->s_block_size-sizeof(zone_t));    /* relative blk # in file */
    last_block = rip->i_size /(rip->i_sp->s_block_size-sizeof(zone_t));
        zone = block_pos >> scale;         /* position's zone */
        boff = (int) (block_pos - (zone << scale) ); /* relative blk # within zone */
        max_zones = last_block >> scale;
        // Now Editing
        for(int i=0; i<zone ; i++)
        {
                first_block = start << scale;
                struct buf *sp = get_block(rip->i_dev , first_block, NORMAL);
                start=((zone_t *)sp->data)[0];
                put_block(sp, PARTIAL_DATA_BLOCK);
                MARKDIRTY(sp);
                j=i;
        }
        for(int i=j; i<max_zones ; i++)
        {
                first_block = start << scale;
                struct buf *sp = get_block(rip->i_dev , first_block, NORMAL);
                temp=((zone_t *)sp->data)[0];
                free_zone(rip->i_dev,start);
                put_block(sp, PARTIAL_DATA_BLOCK);
                MARKDIRTY(sp);
                start=temp;
        }
        IN_MARKDIRTY(rip);
        return(OK);
  }
```

--------The changes I made in proto.h-----------

Line 113

```
void zero_block_seq(struct buf *bp);
```

----------------------------------------------------------------------------------------------------------------------------------

\usr\src\sys\sys

----The changes I have made in stat.h are :

Line 143

```
#define _S_IFSEQ  0110000                    /* sequential */
```

Line 158

```
#define S_IFSEQ  _S_IFSEQ
```

Line 176

```
#define S_ISSEQ(m)       (((m) & _S_IFMT) == _S_IFSEQ)   /* sequential file */
```

----The changes I have made in fcntl.h are :

Line 99

```
#define O_SEQ        020000       /* Sequential flag */
```

Line 130

```
#define F_DISP_ALL_DATA            12   /* for displaying the all data */
#define F_DISP_ONLY_DATA   13   /* for displaying the only data */
#define F_LOGICAL_BLOCK           14   /* for displaying the logical block */
#define F_PHYSICAL_BLOCK   15   /* for displaying the physical block */
```

Line 206

```
int         creat(const char *, mode_t, int); // Added a int variable to creat
```

-----------------------------------------------------------------------------------------------------------------------------------------

\usr\src\sys\lib\libsa

----The changes I made in minixfs3.h

Line 164

```
#define I_SEQ                0110000 /* sequential file */
```