CS450-Programming Assignment 4

Yuyang Luo Jiaqian Yan

Document of Design:

- First, we divide the disk address entries into two parts. The right most 8 bites in an entry record the number of blocks register to this entry; and the left most 24 bites save the address for the first block.
- Because of compatibility to the system call, we didn't modify readi() and writei() functions in fs.c. Then we just modify the bmap() function to make the system support access to the Extent file type.
  - When we call bmap(): if the given inode type is the original file type, we will just move to the original code; if the given inode type is Extent file, we will jump to the new function, Ebmap(), to access the block on disk.
  - Ebmap() will call Eseek() first to get the correct the position of the block on disk. Then, it will map the block into buffer.
  - If Eseek() finds the target block, which is not existed in the file, Ebmap() will use Eballoc() to register a block which is next to the last block we had in this file. If Eballoc() is failed, the original balloc() will be used to register a block into a new disk address entry.

Additional works:

- First because the disk size of xv6 is 512KB, and there are some other applications in this disk, we can only use about 250 blocks from it so that we can never reach the limit 0xFF for a continuous block. That is why we limit each one to 50 blocks so that the over limit files will be split.
- In writei(), it will check the offset before writing anything. If the offset is larger than the max size of the original file type, nothing will be write. So we also changed the way to check the extent file type and make sure it will write successfully.
- In addition, we also add the function Eitrunc() which use bfree() on every block which is no longer need in Extent files. (Specifically use command rm to remove the Extent file)
- Typically, when user create a new file, it highly possible does not know how large the file is. So we didn't alloc all block at the same time, and we just alloc the disk space dynamically as we need to avoid waste of disk space.

Differ files:

- defs.h: give the function name to the system
- fcntl.h: add O_EXTENT flag
- fs.c: modify bmap(), balloc(), itrunc() functions to provide access, alloc, and release

block on disk. And implement lseek().

- fs.h: add three macros so that we can get information from disk address table entry.
- ls.c: modify application of ls, which allows to dump information of Extent file.
- stat.h: modify the system call so that it will dump information about each extent of an extent based file.
- sysfile.c: add seek function to the system so that it is able to seek
- syscall.c: add system call to the system
- syscall.h: add system call to the system
- user.h: add system call to the system
- usys.S: add system call to the system

The manual page for the system call lseek():

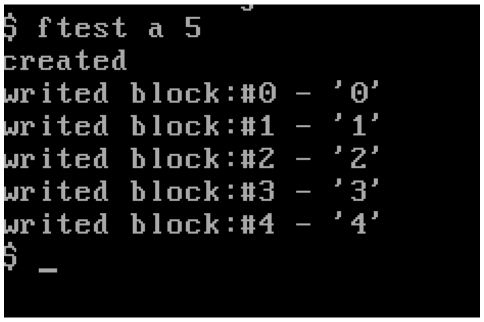
- This function receives two parameters, the file description and the offset size.
- First the system call will check if the offset and the file type are illegal.
- It returns -1, if there are errors
- It returns 0 and set the new offset into the file structure, if the input file parameters are legal.

Test:

First, implementation of test program: “ftest.c”:

```c
#include "types.h"
#include "user.h"
#include "fcntl.h"

char get_idx_char(int idx)
{
    int id =idx% 62;
    if(id<10)
        return '0'+id;
    if(id<36)
        return 'a'+(id-10);
    return 'A'+(id-36);
}

int main(int argc, char *argv[])
{
    if(argc<3)
    {
        printf(2,"please input file name and length\n");
        exit();
    }
    char buff[512]={0};
    int len = atoi(argv[2]);
    int fd = open(argv[1], O_EXTENT|O_RDWR|O_CREATE);
    printf(2,"created\n");
    int i;
    for(i=0;i<len;i++)
    {
        int j;
        char x= get_idx_char(i);
        for (j=0;j<sizeof(buff);j++)
        {
            buff[j]=x;
        }
        write(fd,buff,sizeof(buff));
        printf(2,"writed block:#%d - \'%c\'\n",i, x);
    }
    exit();
}
```

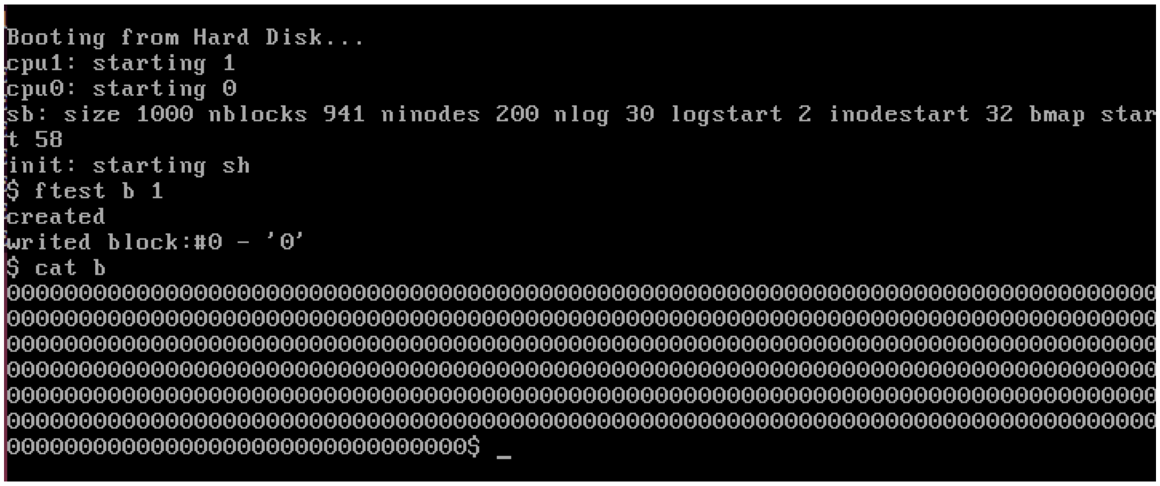
Run “ftest a 5” in xv6:



```
$ ftest a 5
created
writed block:#0 - '0'
writed block:#1 - '1'
writed block:#2 - '2'
writed block:#3 - '3'
writed block:#4 - '4'
$ _
```

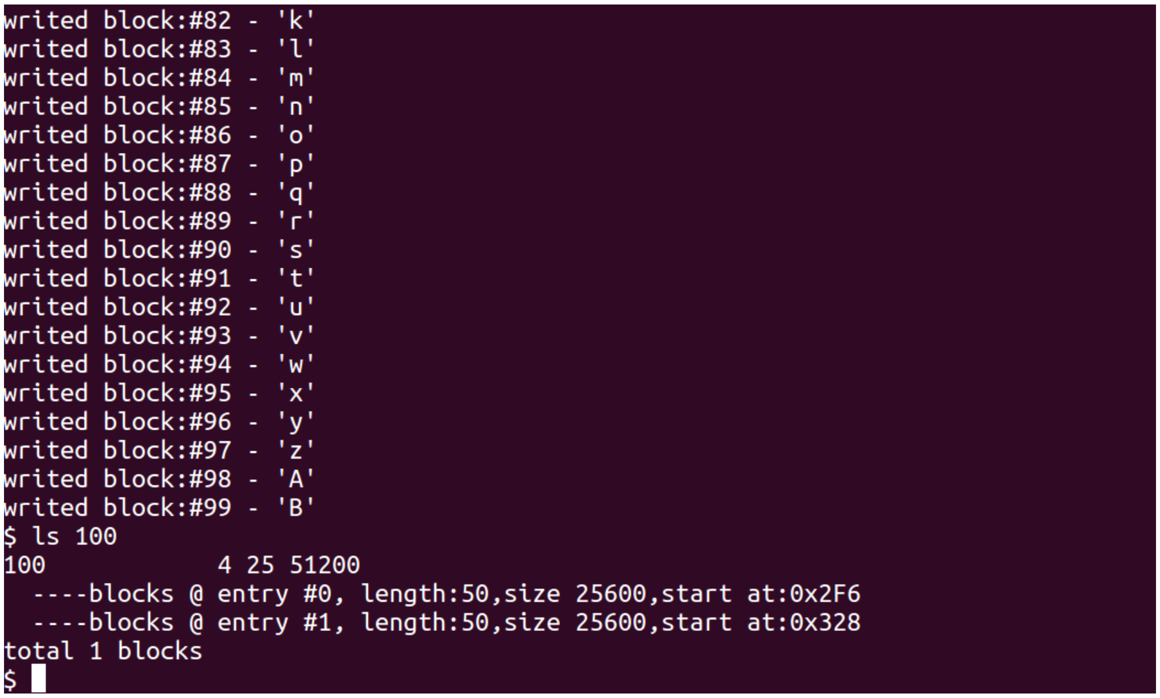
So, with this test, it created out file name “a” and write block for 5 times.

Run “ftest b 1” & “cat b”



```
Booting from Hard Disk...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ ftest b 1
created
writed block:#0 - '0'
$ cat b
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000$ _
```

From the above result, we can see that the cat can be used directly with extent files.

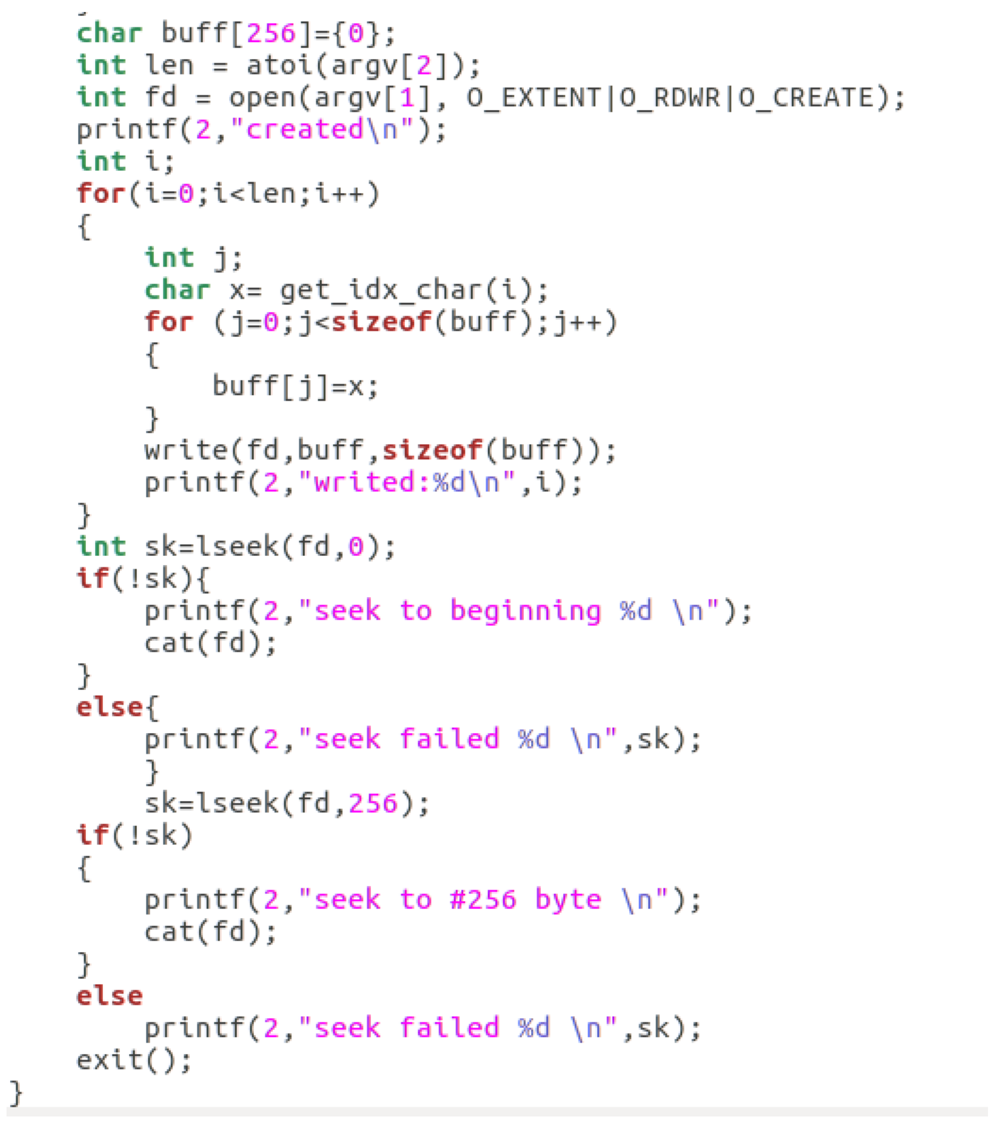


```
writed block:#82 - 'k'
writed block:#83 - 'l'
writed block:#84 - 'm'
writed block:#85 - 'n'
writed block:#86 - 'o'
writed block:#87 - 'p'
writed block:#88 - 'q'
writed block:#89 - 'r'
writed block:#90 - 's'
writed block:#91 - 't'
writed block:#92 - 'u'
writed block:#93 - 'v'
writed block:#94 - 'w'
writed block:#95 - 'x'
writed block:#96 - 'y'
writed block:#97 - 'z'
writed block:#98 - 'A'
writed block:#99 - 'B'
$ ls 100
100            4 25 51200
  ----blocks @ entry #0, length:50,size 25600,start at:0x2F6
  ----blocks @ entry #1, length:50,size 25600,start at:0x328
total 1 blocks
$ 
```

ls on extent files, and dump information of each extent of an extent based file in addition to file size.

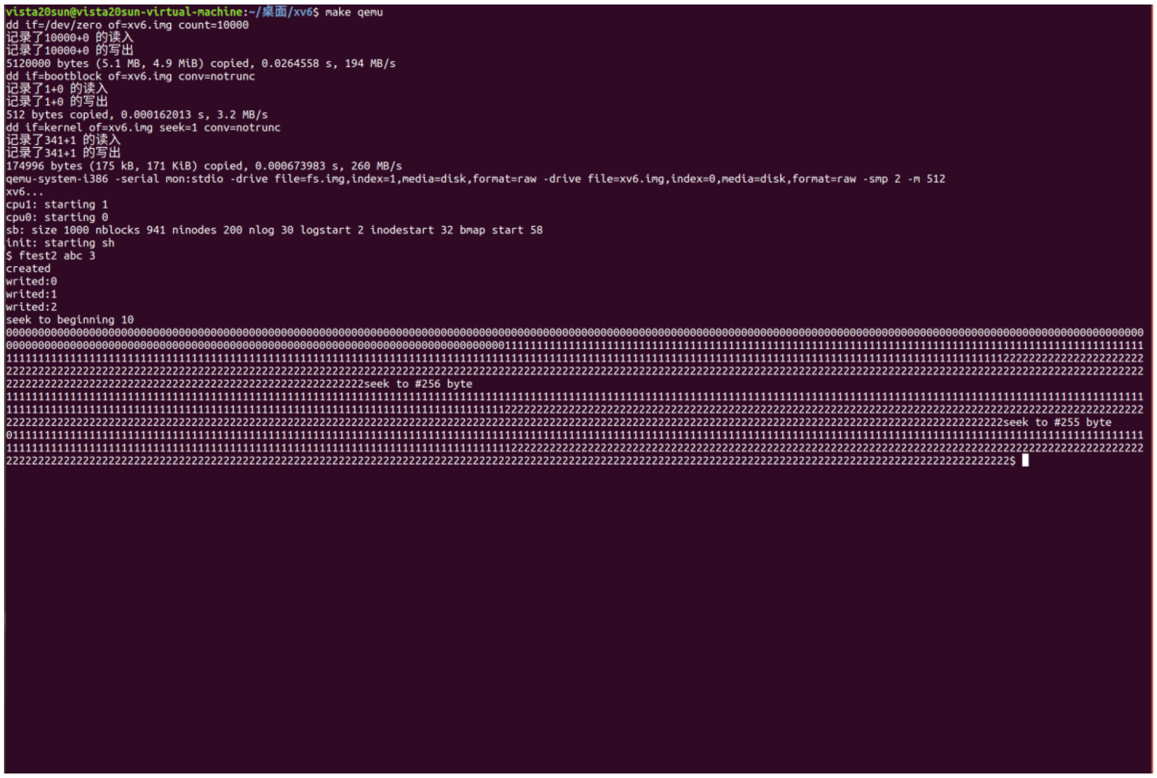
Then, implementation of test program: “ftest2.c”:

```
#include "types.h"
#include "user.h"
#include "fcntl.h"

char get_idx_char(int idx)
{
    int id =idx% 62;
    if(id<10)
        return '0'+id;
    if(id<36)
        return 'a'+(id-10);
    return 'A'+(id-36);
}

void
cat(int fd)
{
  int n;
  char buf[512];
  while((n = read(fd, buf, sizeof(buf))) > 0) {
    if (write(1, buf, n) != n) {
      printf(1, "cat: write error\n");
      exit();
    }
  }
  if(n < 0){
    printf(1, "cat: read error\n");
    exit();
  }
}

int main(int argc, char *argv[])
{
    if(argc<3)
    {
        printf(2,"please input file name and length\n");
        exit();
    }
```



```
    char buff[256]={0};
    int len = atoi(argv[2]);
    int fd = open(argv[1], O_EXTENT|O_RDWR|O_CREATE);
    printf(2,"created\n");
    int i;
    for(i=0;i<len;i++)
    {
        int j;
        char x= get_idx_char(i);
        for (j=0;j<sizeof(buff);j++)
        {
            buff[j]=x;
        }
        write(fd,buff,sizeof(buff));
        printf(2,"writed:%d\n",i);
    }
    int sk=lseek(fd,0);
    if(!sk){
        printf(2,"seek to beginning %d \n");
        cat(fd);
    }
    else{
        printf(2,"seek failed %d \n",sk);
        }
        sk=lseek(fd,256);
    if(!sk)
    {
        printf(2,"seek to #256 byte \n");
        cat(fd);
    }
    else
        printf(2,"seek failed %d \n",sk);
    exit();
}
```

Test again with “ftest2.c”, run “ftest2 abc 3”



```
vista20sun@vista20sun-virtual-machine:~/桌面/xv6$ make qemu
dd if=/dev/zero of=xv6.img count=10000
记录了10000+0 的读入
记录了10000+0 的写出
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0264558 s, 194 MB/s
dd if=bootblock of=xv6.img conv=notrunc
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 0.000162013 s, 3.2 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
记录了341+1 的读入
记录了341+1 的写出
174996 bytes (175 kB, 171 KiB) copied, 0.000673983 s, 260 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ftest2 abc 3
created
writed:0
writed:1
writed:2
seek to beginning 10
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000001111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111112222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
22222222222222222222222222222222222222222222222222222222seek to #256 byte
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111112222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222seek to #255 byte
0111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222$ 
```

From the test, we can see that it created 3 blocks, and kept filling them with numbers in order for 256 bytes, which is (n*256). Until the blocks were all full, it started to seek the file at #256 byte. So, the test run in the right way which meets what we excepted.

Also could use “ls” & “cat” command on the file generated by test program to see the file details.