

가

FlatDisk

4

1.

가

가

2.

2.1 .[1][2]



[2.1]



[2.2]

2.2

가

3.

1. RAM DISK 가
2. zlib
3. /
4. 3
- 5.

4.

4-1.

가 . (2.4.X API [3] [4]) .

```

static int _blocknum[      ];
static int _blocksize [    ];
static int _sectsize [    ];

int flatdisk_open(struct inode *inodep, struct file *filp)
{
    int minor = MINOR(inodep->i_rdev);
    _blocknum[minor] =      ;
    _blocksize[minor] =    ;
    _sectsize[minor] =     ;
    MOD_INC_USE_COUNT;
    return 0;
}

int flatdisk_release(struct inode *inodep, struct file *filp)
{
    MOD_DEC_USE_COUNT;
    return 0;
}

int flatdisk_ioctl(struct inode *inodep, struct file *filp, unsigned int cmd, unsigned
long arg)
{
    int minor = MINOR(inodep->i_rdev);
    switch (cmd) {
    case BLKGETSIZE:
        return _blocknum[minor] * _blocksize [minor] / _sectsize[minor]; //
    case BLKSSZGET: // block size of media
        return _blocksize[minor];
    case BLKFLSBUF: // flush
        if (!capable(CAP_SYS_ADMIN))
            return -EACCES;
        destroy_buffers(inodep->i_rdev);
        break;
    default:
        return -EINVAL;
    }
    return 0;
}

void flatdisk_request( request_queue_t * rq )
{
    int minor, off, size;
    while(1) {
        INIT_REQUEST;
        minor = MINOR(CURRENT_DEV);
        off = CURRENT->sector * _sectsize[minor];
        size = CURRENT->current_nr_sectors * _sectsize[minor];
        switch(CURRENT->cmd) {
        case READ: case READA :
            <      :offset:size>
            break;
        case WRITE:
            <      :offset:size >
            break;
        default:
            end_request (0);
            continue;
        }
        end_request (1);
    }
}

static struct block_device_operations flatdisk_fops = {
    open:flatdisk_open,
    release:flatdisk_release,
    ioctl:flatdisk_ioctl,
};

int init_module(void)
{
    flatdisk_major = register_blkdev (0, DEVICE_NAME, &flatdisk_fops);
    if (flatdisk_major < 0)
        return flatdisk_major;
    blk_init_queue(BLK_DEFAULT_QUEUE(flatdisk_major), flatdisk_request);
    blk_size[MAJOR_NR] = _blocknum;
}

```

```

blksize_size[MAJOR_NR] = _blocksize;
hardsect_size[MAJOR_NR] = _sectsize;
return 0;
}
void cleanup_module(void)
{
    unregister_blkdev (MAJOR_NR, DEVICE_NAME);
    blk_cleanup_queue(BLK_DEFAULT_QUEUE(flatdisk_major));
    blk_size[MAJOR_NR] = NULL;
    blksize_size[MAJOR_NR] = NULL;
    hardsect_size[MAJOR_NR] = NULL;
}

```

```

open, release, ioctl
request,
3가
.
open
,
,
read write
가
.
가
, major
0
. (
major
major
.)

```

```

#insmod < >

#mknod < > b < major > < minor >

#mk2fs < >
ext2

#mount < > < >

```

flatdisk

```
1 - My Profile (192,168,1,3)
File Edit Transfer Fonts Options Tools View Window Help
[root@localhost os.term.flatdisk]# insmod flatdisk_m.o
[root@localhost os.term.flatdisk]# mknod /dev/flatdisk b 254 0
[root@localhost os.term.flatdisk]# mke2fs /dev/flatdisk
mke2fs 1.27 (8-Mar-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
128 inodes, 1024 blocks
51 blocks (4.98%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
128 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 35 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
[root@localhost os.term.flatdisk]# mount /dev/flatdisk /mnt/flatdisk
[root@localhost os.term.flatdisk]# ls /mnt/flatdisk
lost+found/
[root@localhost os.term.flatdisk]#
```

printk

major

(/var/log/messages

.)

4-2.

Jean-loup Gailly Mark Adler가

Zlib[5][6]

```
#define _O_RDONLY 0
#define _O_WRONLY 1
#define _O_CREAT 1000
#define _O_TRUNC 2000

extern void * sys_call_table[];

asm linkage int (*s_open)(const char *fname, int flags, int mod);
asm linkage int (*s_read)(int h, void *buf, int len);
asm linkage int (*s_write)(int h, const void *buf, int len);
```

```

asmlinkage int (*s_close)(int h);
asmlinkage long (*s_lseek)(int h, long offset, int origin);

int _fputc(int c, _FILE *stream)
{
    if (s_write((int)stream, &c, 1) != 1)
        return -1;
    return c;
}
FILE * __fdopen( int handle, const char *mode )
{
    return 0;
}
_FILE * _fopen( const char *filename, const char *mode )
{
    int i, h;
    for(i=0; mode[i]; i++) {
        if (mode[i] == 'r') {
            h = s_open(filename, _O_RDONLY, 0);
            return h < 0 ? (_FILE*)0 : (_FILE*)h;
        }
        if (mode[i] == 'w') {
            h = s_open(filename, _O_WRONLY, 00660);
            if (h < 0)
                h = s_open(filename, _O_WRONLY|_O_CREAT|_O_TRUNC, 00660);
            return h < 0 ? (_FILE*)0 : (_FILE*)h;
        }
    }
    return 0; // (_FILE*) s_open(filename, m);
}
int _fseek( _FILE *stream, long offset, int origin )
{
    return s_lseek((int)stream, offset, origin);
}
int _fread( void *buffer, int size, int count, _FILE *stream )
{
    return s_read((int)stream, buffer, size*count) / size;
}
int _fwrite( const void *buffer, int size, int count, _FILE *stream)
{
    return s_write((int)stream, buffer, size*count) / size;
}
int _fclose( _FILE *stream )
{
    s_close((int)stream);
    return 0;
}
long _ftell( _FILE * handle )
{
    return s_lseek((int)handle, 0, SEEK_CUR);
}
void init_fileptr(void)
{
    s_open = sys_call_table[__NR_open];
    s_read = sys_call_table[__NR_read];
    s_write = sys_call_table[__NR_write];
    s_close = sys_call_table[__NR_close];
    s_lseek = sys_call_table[__NR_lseek];
}

```

IO

IO

[7]

set_fs

KERNEL_DS

```

[12]      .(      2.2      vmalloc      [14]      kmalloc
.)

zlib

zlib      가      makefile [8][9][10]
      flatdisk      makefile

```

```

#
OBJECTS = adler32.o compress.o crc32.o deflate.o gzio.o inblock.o infcodes.o inffast.o
inflate.o inftrees.o infutil.o trees.o uncompr.o zfile.o zutil.o flatdisk.o

#
CFLAGS = -Wall -I /usr/src/linux/include -DMODULE -D__KERNEL__
CC = gcc
AR = ar
LD = ld

#
flatdisk : $(OBJECTS)
      $(LD) -r -o flatdisk_m.o $(OBJECTS)

#
adler32.o      : adler32.c
      $(CC) -c $(CFLAGS) adler32.c
compress.o    : compress.c
      $(CC) -c $(CFLAGS) compress.c
...

```

, ld [11] .

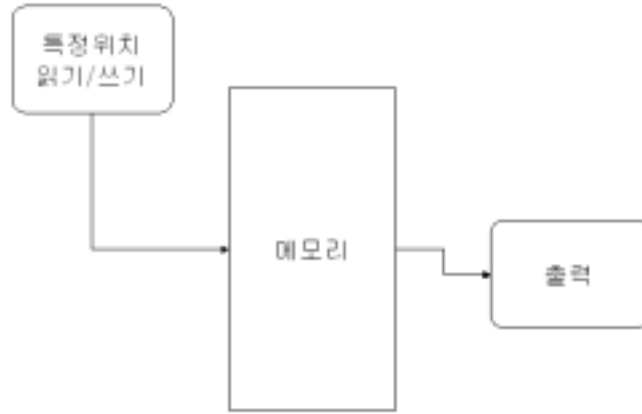
```

#ld -r -o <      > <      1> <      2> ... <      n>

```

4-3.

2.1 가
1:1 . 가
가 , 가

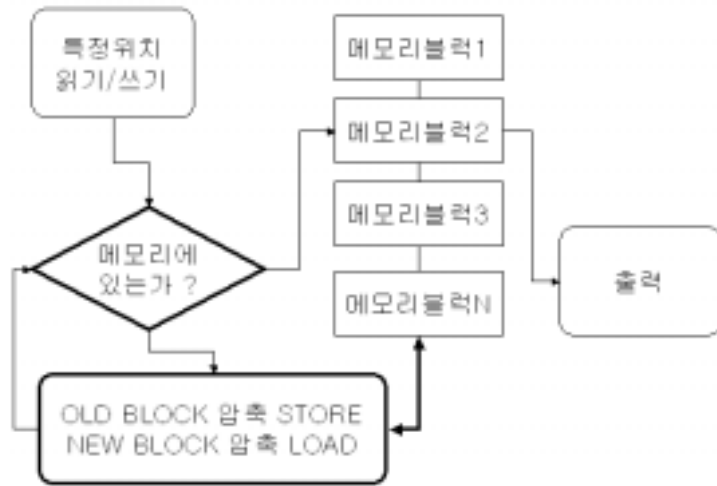


[3.1]

RAM DISK

3.1

3.2



[3.2]

가 가

가 Cache LRU (Last
Recently Used) [13]

```
#define ZBLOCK (64*1024) // 64K
#define ZCACHE 4

typedef struct {
    unsigned char * buffer;
    int updatecnt;
    int id;
    int dirty;
} s_zblock;

s_zblock zblock[ZCACHE];

static int globalcnt = 1;

static int flatdisk_findblock(int id)
{
    int i;
    for(i=0; i<ZCACHE; i++) {
        if (zblock[i].buffer != 0l && zblock[i].id == id) {
            zblock[i].updatecnt = globalcnt++;
            return i;
        }
    }
    return -1;
}

static void flatdisk_store(int id, void *buff)
{
    char fname[128];

    sprintf(fname, "%s/z%03d.bin", basedir, id);
    z_compress(fname, buff, ZBLOCK);
}

static void flatdisk_restore(int id, void *buf)
{
    char fname[128];

    sprintf(fname, "%s/z%03d.bin", basedir, id);
    z_uncompress(fname, buf, ZBLOCK);
}

static void flatdisk_flush()
{
    int i;
    for(i=0; i<ZCACHE; i++) {
        if (zblock[i].buffer != 0l && zblock[i].dirty == 1) {
            flatdisk_store(zblock[i].id, zblock[i].buffer);
            zblock[i].dirty = 0;
        }
    }
}

static void flatdisk_destroy()
{
    int i;
    for(i=0; i<ZCACHE; i++) {
        if (zblock[i].buffer != 0l) {
            z_free(zblock[i].buffer);
        }

        zblock[i].dirty = 0;
        zblock[i].buffer = 0;
        zblock[i].id = 0;
    }
}
}
```

```

static int flatdisk_alloccblock(int id)
{
    int i, sel=0;

    for(i=0; i<ZCACHE; i++) {
        if (zblock[i].buffer == 0) {
            sel = i;
            break;
        }
        if (zblock[i].updatecnt < zblock[sel].updatecnt) {
            sel = i;
        }
    }

    if (zblock[sel].buffer != 0 && zblock[sel].dirty == 1) {
        flatdisk_store(zblock[sel].id, zblock[sel].buffer);
    }

    if (zblock[sel].buffer == 0) {
        zblock[sel].buffer = z_alloc(ZBLOCK);
        memset(zblock[sel].buffer, 0, ZBLOCK);
    }

    zblock[sel].updatecnt = globalcnt++;
    zblock[sel].id = id;
    zblock[sel].dirty = 0;

    flatdisk_restore(id, (void*) zblock[sel].buffer);

    return sel;
}
static int flatdisk_getblock(int id)
{
    int i;
    i = flatdisk_findblock(id);
    if (i == -1)
        i = flatdisk_alloccblock(id);
    return i;
}
void flatdisk_read(void * buf, int off, int len)
{
    int rest, i, seg;

    rest = off % ZBLOCK;

    if (rest + len > ZBLOCK) {
        seg = ZBLOCK - rest;
        flatdisk_read(buf, off, seg);
        flatdisk_read((char*)buf + seg, off + seg, len - seg);
        return ;
    }

    i = flatdisk_getblock(off / ZBLOCK);
    memcpy(buf, zblock[i].buffer + rest, len);
}
void flatdisk_write(void * buf, int off, int len)
{
    int rest, i, seg;

    rest = off % ZBLOCK;

    if (rest + len > ZBLOCK) {
        seg = ZBLOCK - rest;
        flatdisk_write(buf, off, seg);
        flatdisk_write((char*)buf + seg, off + seg, len - seg);
        return ;
    }

    i = flatdisk_getblock(off / ZBLOCK);
    memcpy(zblock[i].buffer + rest, buf, len);
    zblock[i].dirty = 1;
}

```

```
}

```

zblock read, write 가
가 가 ,
dirty , , wirt dirty 1 .

4-4.

2.4.X

2.2.X

. (2.4.X, 2.2X 가 가 .)

```
#if ((LINUX_VERSION_CODE>>8)&0xFF) == 2 && ((LINUX_VERSION_CODE>>16)&0xFF) == 2
#define _VER_2_2
#else
#if ((LINUX_VERSION_CODE>>8)&0xFF) != 4 || ((LINUX_VERSION_CODE>>16)&0xFF) != 2
# error
#endif

```

가 . (2.2.x memcpy, memset export
가 .)

5.

```
#insmod flatdisk_m.o <-fsize > <-fbase >
```

가 ,
ramdisk 가 , 가 . ext2
ext3 , flatdisk

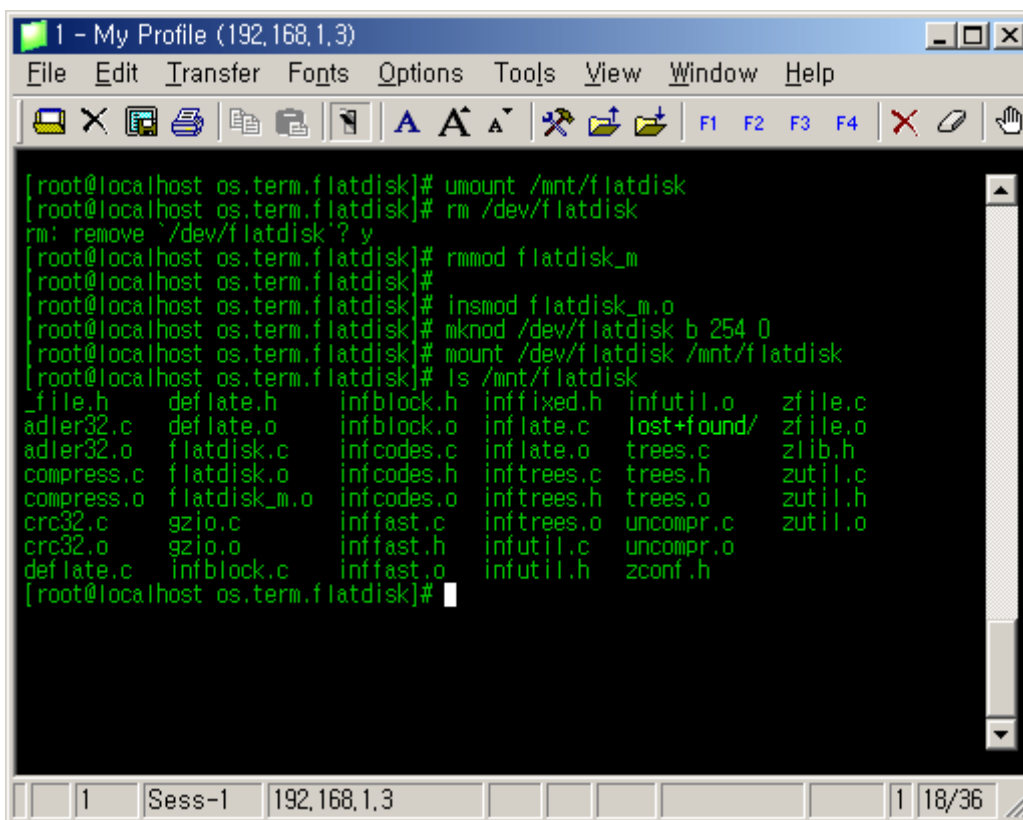
가

```
[root@localhost os.term.flatdisk]# ls
_file.h  deflate.h  infblock.h  inffixed.h  infutil.o  zconf.h
adler32.c  deflate.o  infblock.o  inflate.c  makefile  zfile.c
adler32.o  flatdisk.c  infcodes.c  inflate.o  temp.bin  zfile.o
compress.c  flatdisk.o  infcodes.h  inftrees.c  trees.c  zlib.h
compress.o  flatdisk_m.o  infcodes.o  inftrees.h  trees.h  zutil.c
crc32.c  gzio.c  inffast.c  inftrees.o  trees.o  zutil.h
crc32.o  gzio.o  inffast.h  infutil.c  uncompr.c  zutil.o
deflate.c  infblock.c  inffast.o  infutil.h  uncompr.o
[root@localhost os.term.flatdisk]# cp *.* /mnt/flatdisk
[root@localhost os.term.flatdisk]# du -b /mnt/flatdisk
12288 /mnt/flatdisk/lost+found
510976 /mnt/flatdisk
[root@localhost os.term.flatdisk]# du -b /home/noerror/disk
135168 /home/noerror/disk
[root@localhost os.term.flatdisk]# diff . /mnt/flatdisk
Only in /mnt/flatdisk: lost+found
Only in .: makefile
[root@localhost os.term.flatdisk]# df -k
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/hda6 2719716 2189948 391612 85% /
none 257900 0 257900 0% /dev/shm
/dev/flatdisk 1003 499 453 53% /mnt/flatdisk
```

```
[root@localhost os.term.flatdisk]#
```

가 1/4

가 . Diff
(가 가 makefile
lost+found .)



```
1 - My Profile (192,168,1,3)
File Edit Transfer Fonts Options Tools View Window Help
[root@localhost os.term.flatdisk]# umount /mnt/flatdisk
[root@localhost os.term.flatdisk]# rm /dev/flatdisk
rm: remove `/dev/flatdisk'? y
[root@localhost os.term.flatdisk]# rmdir /dev/flatdisk
[root@localhost os.term.flatdisk]# rmmmod flatdisk_m
[root@localhost os.term.flatdisk]# insmod flatdisk_m.o
[root@localhost os.term.flatdisk]# mknod /dev/flatdisk b 254 0
[root@localhost os.term.flatdisk]# mount /dev/flatdisk /mnt/flatdisk
[root@localhost os.term.flatdisk]# ls /mnt/flatdisk
_file.h  deflate.h  infblock.h  infixed.h  infutil.o  zfile.c
adler32.c  deflate.o  infblock.o  inflate.c  lost+found/  zfile.o
adler32.o  flatdisk.c  infcodes.c  inflate.o  trees.c      zlib.h
compress.c  flatdisk.o  infcodes.h  inftrees.c  trees.h      zutil.c
compress.o  flatdisk_m.o  infcodes.o  inftrees.h  trees.o      zutil.h
crc32.c     gzio.c     inffast.c  inftrees.o  uncompr.c   zutil.o
crc32.o     gzio.o     inffast.h  infutil.c  uncompr.o
deflate.c  infblock.c  inffast.o  infutil.h  zconf.h
[root@localhost os.term.flatdisk]#
```

```
[root@localhost os.term.flatdisk]# umount /mnt/flatdisk
[root@localhost os.term.flatdisk]# rm /dev/flatdisk
rm: remove `/dev/flatdisk'? y
[root@localhost os.term.flatdisk]# rmmmod flatdisk_m
[root@localhost os.term.flatdisk]#
[root@localhost os.term.flatdisk]# insmod flatdisk_m.o
[root@localhost os.term.flatdisk]# mknod /dev/flatdisk b 254 0
[root@localhost os.term.flatdisk]# mount /dev/flatdisk /mnt/flatdisk
```

```

[root@localhost os.term.flatdisk]# ls /mnt/flatdisk
_file.h  deflate.h  infblock.h  infix.h  infutil.o  zfile.c
adler32.c  deflate.o  infblock.o  inflate.c  lost+found/  zfile.o
adler32.o  flatdisk.c  infcodes.c  inflate.o  trees.c      zlib.h
compress.c  flatdisk.o  infcodes.h  inftrees.c  trees.h      zutil.c
compress.o  flatdisk_m.o  infcodes.o  inftrees.h  trees.o      zutil.h
crc32.c    gzio.c     inffast.c  inftrees.o  uncompr.c    zutil.o
crc32.o    gzio.o     inffast.h  infutil.c  uncompr.o
deflate.c  infblock.c  inffast.o  infutil.h  zconf.h
[root@localhost os.term.flatdisk]#

```

6.

가 .

, 가 ,

.

가 가

, 가 . (

, .)

가, (,

가) 가

.

가 ,

가 .

7.

[1] Understanding the linux kernel, chapter 12, o'reilly

[2]

http://doc.kldp.org/Translations//html/The_Linux_Kernel-KLDP/tlk9.html

[3] Understanding the linux kernel, chapter 13, o'reilly

[4] Mini RAM Disk

<http://www.linuxkernel.net/moduleprog/lkp/source/minird.c>

- <http://www.linuxkernel.net/moduleprog/lkp/doc/v1.01/lkp.pdf>
- [5] ZLIB
<http://www.gzip.org/zlib/>
- [6] ZLIB
<http://user.chollian.net/~oriqpt/materials/old/zlib/ZLIB.htm>
- [7] System Call Spy
<http://www.linuxkernel.net/moduleprog/lkp/source/syscall.c>
- [8] How to write a Makefile
http://vertigo.hsrl.rutgers.edu/ug/make_help.html
- [9] Makefile Getting Started
<http://oucsace.cs.ohiou.edu/~bhumphre/makefile.html>
- [10] Makefile Tutorial
<http://www.opussoftware.com/tutorial/TutMakefile.htm>
- [11] LD , AIX 4.3
[http://www.kimseoyoung.com/AIX/Aix/Manual/usr/share/man/info/ko_KR/a_doc
_lib/cmds/aixcmds3/ld.htm](http://www.kimseoyoung.com/AIX/Aix/Manual/usr/share/man/info/ko_KR/a_doc_lib/cmds/aixcmds3/ld.htm)
- [12] Getting Hold of Memory, Linux Device Drivers, 2nd Edition
<http://www.xml.com/ldd/chapter/book/ch07.html>
- [13] Understanding the linux kernel, chapter 10, o'reilly
- [14] vmalloc
http://members.tripod.lycos.co.kr/shpark75/tutorial/linux1/linux7_9.htm