



# GASFileSystem

A custom Linux file system designed and implemented as a kernel module. Project for CS334 Operating System.

[Task description here.](#)

## Contributors

SID	Name	Contributions	Contribution Rate
12111624	<a href="#">GuTaoZi</a>	Document, file system maker, scripts	33.33%
12110524	<a href="#">Artanisax</a>	Layout, annotation, VFS interface	33.33%
12112012	<a href="#">ShadowStorm</a>	Layout, annotation, environment	33.33%

## Project Structure

```
GAS_Filesystem
├── CHANGELOG.md
├── doc
│   ├── Project Report.pdf
│   └── Project Proposal.pdf
├── LICENSE
├── README.md
└── src
    ├── kern
    │   ├── bitmap.c
    │   ├── def.c
    │   ├── gas_dir.c
    │   ├── gas.h
    │   ├── gas_inode.c
    │   ├── gas_itree.c
    │   ├── gas_itree.h
    │   ├── gas_namei.c
    │   ├── gas_namei.h
    │   ├── gas_page.c
    │   ├── gas_super.c
    │   └── Makefile
```

```
├─ makefs
│   ├── bitmap.c
│   ├── bitmap.h
│   ├── Makefile
│   └── mkfs.c
└─ test
    ├── build_km_fs.sh
    ├── format_vdisk.sh
    ├── load_mount.sh
    ├── total.sh
    └── umount_rmmod.sh
```

## Environment Setup

GAS file system implement the VFS interfaces of ext2 for `Linux-3.13.0-170`. Following are some steps to set up the environment.

1. Prepare a Linux distribution that can use Linux 3.13.0-170 as the kernel.

Check whether the distribution supports this kernel:

```
sudo apt-cache search linux-image
```

If the available kernel lists contains the kernel version we want, then it's OK to run GAS file system in this environment.

Here we choose `Ubuntu 14.04.6 LTS`, you can download the image [here](#).

2. Install Linux kernel

You can check the available kernel versions using command:

```
sudo apt-cache search linux-image
```

To install Linux-3.13.0-170 kernel, run the following command:

```
sudo apt-get install linux-image-unsigned-3.13.0-170-generic
```

You can change the kernel version according to the available version lists, as long as it's before Linux-3.15.

3. Switch Linux kernel

Check the current kernel list:

```
dpkg --get-selections | grep linux-image
```

Open `/etc/default/grub` and edit:

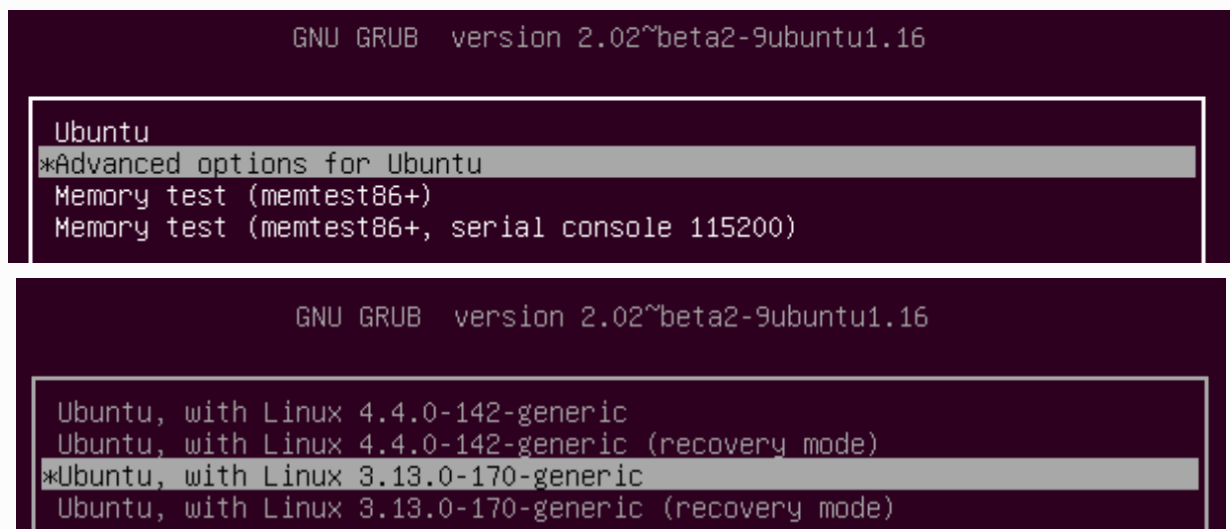
```
...
GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=-1
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="text"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical
locale=en_US"
...
```

Update grub.cfg:

```
sudo update-grub
```

#### 4. Reboot and select kernel version

After rebooting, select **ubuntu advanced options**, and switch to the proper Linux kernel.\



```
GNU GRUB  version 2.02~beta2-9ubuntu1.16

Ubuntu
*Advanced options for Ubuntu
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)

GNU GRUB  version 2.02~beta2-9ubuntu1.16

Ubuntu, with Linux 4.4.0-142-generic
Ubuntu, with Linux 4.4.0-142-generic (recovery mode)
*Ubuntu, with Linux 3.13.0-170-generic
Ubuntu, with Linux 3.13.0-170-generic (recovery mode)
```

You can check if the kernel version by running `uname -a` .

#### 5. Clone project repository

After installing and configuring some software packages like git, clone this repo using git:

```
git clone https://github.com/GuTaoZi/GAS_Filesystem.git
```

GAS file system only includes the kernel headers, no other dependency needed.

# Usage Guide

The shell commands to test GAS file system is integrated in shell scripts in `GAS_Filesystem/src/test`, you can execute the scripts or try commands yourself.

## Test scripts

`total.sh` : integrating all the following scripts (unmount immediately after successfully mounting)

`build_km_fs.sh` : build the kernel module and file system maker

`format_vdisk.sh` : create a virtual disk

`load_mount.sh` : load kernel module, mount virtual disk

`umount_rmmod.sh` : unmount virtual disk, remove module

## Step-by-step building

1. Build kernel module

Run `make` in `GAS_Filesystem/src/kern`

2. Build file system maker

Run `make` in `GAS_Filesystem/src/makefs`

3. Create and format a 4MB virtual disk

```
dd if=/dev/zero of=vdisk bs=1M count=4
../tools/mkfs.gas vdisk
```

4. Load kernel module and mount the virtual disk

```
insmod ../kern/gas.ko
mount -o loop -t gas vdisk /mnt/gas
```

5. GAS file system is running in `/mnt/gas/` !

```
ls -al /mnt/gas
```

You can try some file operations in this directory.

6. Unmount the virtual disk and remove GAS file system from kernel modules

```
umount /mnt/gas
rmmod gas
```

# File System Layout

All the meta data are stored in little endian in the corresponding structures:

## Super block

```
struct gas_super_block
{
    __le32 s_magic;           // magic number
    __le32 s_blocksize;       // block size
    __le32 s_bam_blocks;      // block alloc map block
    __le32 s_iam_blocks;      // inode alloc map block
    __le32 s_inode_blocks;    // inodes per block
    __le32 s_nblocks;         // number of blocks
    __le32 s_ninodes;         // number of inodes
};
```

## Inode

```
struct gas_inode
{
    __le16 i_mode;            // file type: file, directory, symlink etc.
    __le16 i_nlink;           // number of hard links references
    __le32 i_uid;             // Owner user id
    __le32 i_gid;             // Owner group id
    __le32 i_size;            // inode size
    __le32 i_atime;           // Access time
    __le32 i_mtime;           // Modify time
    __le32 i_ctime;           // Last change time
    __le32 i_blkaddr[9];      // 6 direct, single + double + triple indirect block address
};
```

## Directory entry

```
struct gas_dir_entry
{
    char de_name[GAS_MAX_NAME_LEN]; // dentry name (max length 60)
    __le32 de_inode;                 // dentry inode number
};
```

# VFS Interfaces

This project implemented the following interfaces of VFS, so you can directly use basic file operations like `cd`, `ls`, `touch`, `cat`, `echo` etc. under the directory `/mnt/gas`.

## Address space

```
const struct address_space_operations gas_a_ops = {
    .readpage = gas_readpage,
    .readpages = gas_readpages,
    .writepage = gas_writepage,
    .writepages = gas_writepages,
    .write_begin = gas_write_begin,
    .write_end = gas_write_end,
    .bmap = gas_bmap,
    .direct_IO = gas_direct_io,
};
```

## File operations

```
const struct file_operations gas_file_ops = {
    .llseek = generic_file_llseek,
    .read = do_sync_read,
    .aio_read = generic_file_aio_read,
    .write = do_sync_write,
    .aio_write = generic_file_aio_write,
    .mmap = generic_file_mmap,
    .splice_read = generic_file_splice_read,
    .splice_write = generic_file_splice_write
};
```

## Inode: file, directory, symlink

```
const struct inode_operations gas_file_inode_ops = {
    .getattr = gas_getattr,
};

const struct inode_operations gas_dir_inode_ops = {
    .create = gas_create,
    .lookup = gas_lookup,
    .link = gas_link,
    .unlink = gas_unlink,
    .symlink = gas_symlink,
    .mknod = gas_mknod,
    .mkdir = gas_mkdir,
    .rmdir = gas_rmdir,
    .getattr = gas_getattr,
};

const struct inode_operations gas_symlink_inode_ops = {
    .readlink = generic_readlink,
    .follow_link = page_follow_link_light,
    .put_link = page_put_link,
    .getattr = gas_getattr,
};
```

# File System Maker

The GAS file system maker is [/GAS\\_Filesystem/src/makefs/mkfs.c](#) .

This file system maker will open the virtual disk and initialize super block, block allocate map, inode allocate map and inode list, then it will create the root directory inode and do block cache synchronization.

Also this maker will print some basic information of the file system running on this virtual disk.

By execute `mkfs.gas` , you can format a virtual disk file into GAS file system style.

```
./mkfs.gas vdisk_name
```

```
=====
GASFILESYS
=====
```

```
=====GAS File Sys Info=====
```

```
Disk size = 4194304
```

```
Number of Blocks = 1024
```

```
BAM blocks = 1
```

```
IAM blocks = 1
```

```
inode blocks = 4
```

```
Number of inodes = 256
```

```
Data block starts at 7 block
```

```
=====
GAS: Initializing super block ...
```

```
GAS: Initializing block alloc map ...
```

```
GAS: Initializing iNode alloc map ...
```

```
GAS: Initializing iNode list ...
```

```
Device write complete
```

```
total 8
```

```
drwxr-xr-x 2 root root 520 Jun 15 04:34 .
```

```
drwxr-xr-x 3 root root 4096 Jun 14 22:41 ..
```


## Changelog

See [CHANGELOG.md](#).


# TODOs


 GAS File System


 Layout Component

 Super block


 Inode


 Directory entry


 Tree structure

 VFS interfaces

 Page


 File operations


 Address space operations


 Inode operations

 File

 Link

 Directory

 Annotations

 Project report

 Project video