```c
/** @file
 *
 * minimal example filesystem using high-level API
 *
 * Compile with:
 *
 *     gcc -Wall ASE-fuse.c -o ASE-fuse -D_FILE_OFFSET_BITS=64 -lfuse
 *
 */

#include <fuse.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <stddef.h>
#include <assert.h>
#include <dirent.h>

struct file {
        char filename[MAXNAMLEN];
        char contents[4096];
        int size;
} file;

struct directory {
        struct file files[10];
        int entry_count;
} dir;

static void *simple_init()
{
        /* create/fill directory */
        dir.entry_count=1;
        strcpy(dir.files[0].filename, "test1");
        dir.files[0].size=10;
        memcpy(dir.files[0].contents, "ABCDEFGHIJKLMNOPQRSTUVWXYZ", 10);
        return NULL;
}

static int simple_getattr(const char *path, struct stat *stbuf)
{
        int res = 0;

        memset(stbuf, 0, sizeof(struct stat));
        if (strcmp(path, "/") == 0) {
                stbuf->st_mode = S_IFDIR | 0755;
                stbuf->st_nlink = 2;
        } else if (strcmp(path+1, dir.files[0].filename) == 0) {
                stbuf->st_mode = S_IFREG | 0444;
                stbuf->st_nlink = 1;
                stbuf->st_size = dir.files[0].size;
        } else
                res = -ENOENT;

        return res;
}

static int simple_getdir(const char *path, fuse_dirh_t h, fuse_dirfil_t filler)
{
        if (strcmp(path, "/") != 0)
                return -ENOENT;

        filler(h, ".", 0);
        filler(h, "..", 0);
        filler(h, dir.files[0].filename, 0);

        return 0;
}

static int simple_open(const char *path, int flags)
{
        if (strcmp(path+1, dir.files[0].filename) != 0)
```



```c
                return -ENOENT;

        if ((flags & O_ACCMODE) != O_RDONLY)
                return -EACCES;

        return 0;
}

static int simple_read(const char *path, char *buf, size_t size, off_t offset)
{
        size_t len;

        if(strcmp(path+1, dir.files[0].filename) != 0)
                return -ENOENT;

        len = dir.files[0].size;
        if (offset < len) {
                if (offset + size > len)
                        size = len - offset;
                memcpy(buf, dir.files[0].contents + offset, size);
        } else
                size = 0;

        return size;
}

static struct fuse_operations simple_oper = {
        .getattr        = simple_getattr,
        .getdir         = simple_getdir,
        .open           = simple_open,
        .read           = simple_read,
//      .mkdir          =
//      .unlink         =
//      .rmdir          =
//      .rename         =
//      .chmod          =
};

int main(int argc, char *argv[])
{
        struct fuse_args args = FUSE_ARGS_INIT(argc, argv);

        simple_init();
        return fuse_main(args.argc, args.argv, &simple_oper);
}
```

```
1
2  1) Setup 1 machine with pcocc
3   pcocc template list
4   pcoco alloc -c2 XXXXX
5   pcocc ssh vm0
6
7  2) Install fuse, fuse-devel + needed tools (gcc, ...)
8
9  3) compile ASE-fuse.c
10
11 4) Mount your new filesystem
12  mkdir /mnt/ASE
13  ./ASE-fuse /mnt/ASE
14
15 5) Check content of /mnt/ASE
16  ls -l /mnt/ASE
17
18 6) Check file content
19
20 7) Modify source to
21  - add a new file: size = 5 bytes, content = 01234
22  - add mtime support with modification time set to current time
23  - implemente rename method and test it with mv
24  - add unix right support and implement chmod method and test it with chmod
25
26
27
```