Project 4 Help

SPL, SNU

Overview

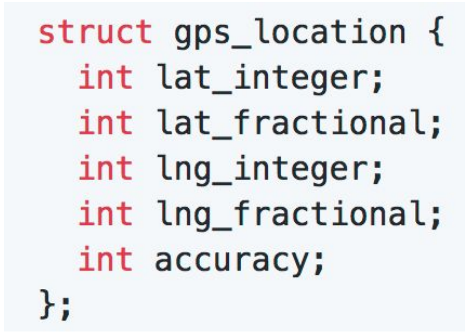
- Geo-tagged file system (based on ext2 or ext4)
  - Attach a GPS tag to each regular file
- Access control with the tags
  - Files are accessible from the location where they are recently created/modified

Key Challenges

- Modify physical representation of inode
    - to embed GPS coordinates
- Add GPS-related inode operations and implement them for ext2(ext4) regular files
    - set_gps_location
    - get_gps_location
- Modify access control mechanism to realize location-based access control

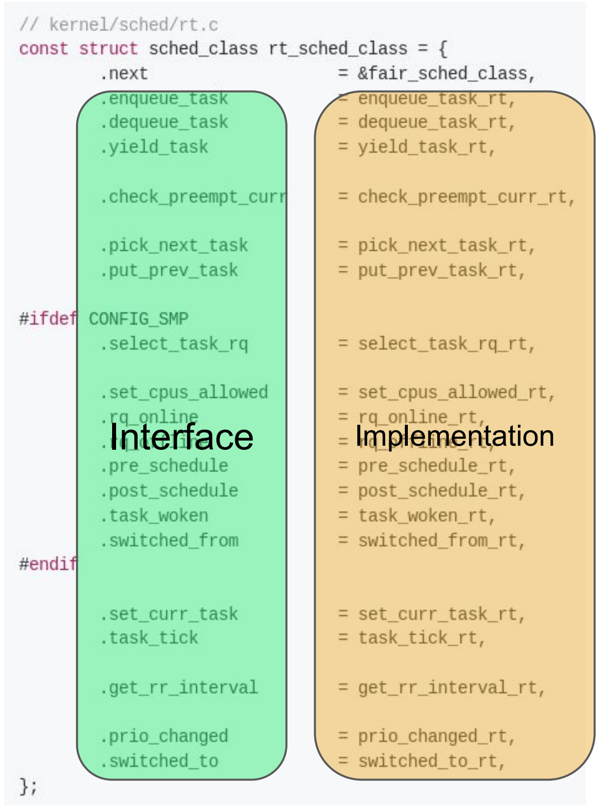
Add GPS-related fields to inode structure

- fs/ext2/ext2.h(or fs/ext4/ext4.h)
- There are two structs for ext2 inode.
  - **inode in the memory**
  - **inode on the disk**
- Add 5 fields.
- Pay attention to endianness of the fields of the **physical inode**.
  - You may get a hint from other fields in the structures.



```
struct gps_location {
  int lat_integer;
  int lat_fractional;
  int lng_integer;
  int lng_fractional;
  int accuracy;
};
```

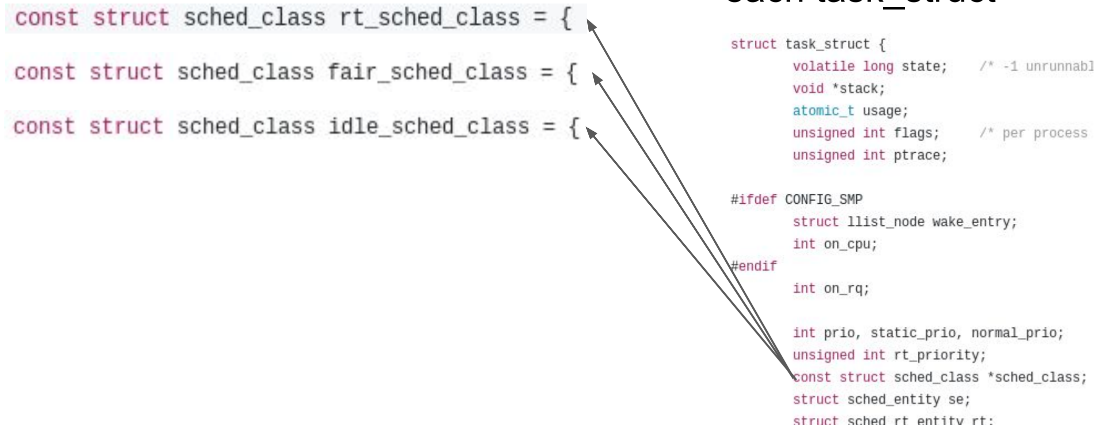
Recall from Project 3...



```
// kernel/sched/rt.c
const struct sched_class rt_sched_class = {
        .next                   = &fair_sched_class,
        .enqueue_task           = enqueue_task_rt,
        .dequeue_task           = dequeue_task_rt,
        .yield_task             = yield_task_rt,

        .check_preempt_curr     = check_preempt_curr_rt,

        .pick_next_task         = pick_next_task_rt,
        .put_prev_task          = put_prev_task_rt,

#ifdef CONFIG_SMP
        .select_task_rq         = select_task_rq_rt,

        .set_cpus_allowed       = set_cpus_allowed_rt,
        .rq_online              = rq_online_rt,
        .rq_offline             = rq_offline_rt,
        .pre_schedule           = pre_schedule_rt,
        .post_schedule          = post_schedule_rt,
        .task_woken             = task_woken_rt,
        .switched_from          = switched_from_rt,
#endif

        .set_curr_task          = set_curr_task_rt,
        .task_tick              = task_tick_rt,

        .get_rr_interval        = get_rr_interval_rt,

        .prio_changed           = prio_changed_rt,
        .switched_to            = switched_to_rt,
};
```

Multiple implementation sets

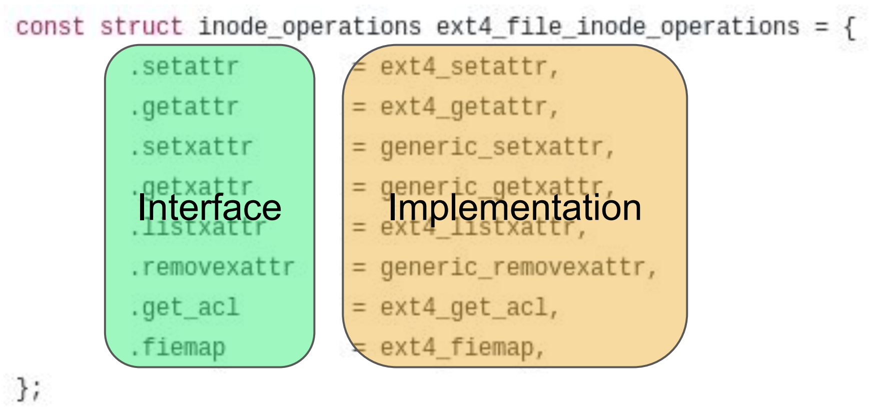
Pointing from each task_struct



```
const struct sched_class rt_sched_class = {

const struct sched_class fair_sched_class = {

const struct sched_class idle_sched_class = {
```

```
struct task_struct {
        volatile long state;    /* -1 unrunnabl
        void *stack;
        atomic_t usage;
        unsigned int flags;     /* per process
        unsigned int ptrace;

#ifdef CONFIG_SMP
        struct llist_node wake_entry;
        int on_cpu;
#endif
        int on_rq;

        int prio, static_prio, normal_prio;
        unsigned int rt_priority;
        const struct sched_class *sched_class;
        struct sched_entity se;
        struct sched_rt_entity rt;
```

Similar for inode_operations



```
const struct inode_operations ext4_file_inode_operations = {
        .setattr        = ext4_setattr,
        .getattr        = ext4_getattr,
        .setxattr       = generic_setxattr,
        .getxattr       = generic_getxattr,
        .listxattr      = ext4_listxattr,
        .removexattr    = generic_removexattr,
        .get_acl        = ext4_get_acl,
        .fiemap         = ext4_fiemap,
};
```

```
const struct inode_operations ext3_dir_inode_operations = {
const struct inode_operations ext4_file_inode_operations = {
const struct inode_operations ext4_special_inode_operations = {
```

```
struct inode {
        umode_t                 i_mode;
        unsigned short          i_opflags;
        kuid_t                  i_uid;
        kgid_t                  i_gid;
        unsigned int            i_flags;

#ifdef CONFIG_FS_POSIX_ACL
        struct posix_acl        *i_acl;
        struct posix_acl        *i_default_acl;
#endif

        const struct inode_operations   *i_op;
        struct super_block      *i_sb;
```

Make a new syscall

- int set_gps_location(struct gps_location __user *loc)

```
struct gps_location {
  int lat_integer;
  int lat_fractional;
  int lng_integer;
  int lng_fractional;
  int accuracy;
};
```

- Latitude = lat_integer + lat_frac*10^-6
- Longitude = lng_integer+lng_frac*10^-6
- 0 <= lat_frac, lng_frac <= 999,999
- -90 <=latitude <= 90
- -180 <= longitude <= 180

Define GPS-related inode operations

- Add the following two function pointer fields to the **struct inode_operations** structure in **include/linux/fs.h**
  - int (*set_gps_location)(struct inode *);
  - int (*get_gps_location)(struct inode *, struct gps_location *);

And implement them for ext2(ext4)

- Implement the set/get functions for ext2(ext4).
  - set_gps_location: copy the current device location to the inode
  - get_gps_location: copy the inode location to the buffer
- Register the functions with ext2(ext4) file inode operations.

Update location info

- GPS info of regular files should be updated whenever they are created or modified
  - Use set_gps_location operation
- look at
  - fs/ - for file system code
  - fs/ext2/ - for ext2 specific code
  - fs/ext4/ - for ext4 specific code

Access control

- Files of the modified ext2(4) can be only accessible from the location where they are recently created/modified.
- There is an inode operation related to access control. You can use it.
- Compare the geo-tag and current location.
  - You cannot use float or double operations.
  - You should consider accuracy of the geo-tag
  - Compare the values with your own algorithm. Document any assumptions or approximations on README.md

Be careful!

- Current device location is shared mutable state, so you should use proper synchronization mechanism when accessing the state.
- Never access the memory by user-space addresses directly. Refer to guides and provided links in Project 1 help document(Linux Kernel Exploration Guide for OS projects).
- For parameters in struct gps_location in set_gps_location system call, make sure they are in appropriate range.

Testing with the modified file system

- To test your code, you should create a your modified ext2 file system. You will use mke2fs.
- You need to modify ext2 inode structure to make mke2fs use your modified ext2.
  - e2fsprogs/lib/ext2fs/ext2_fs.h
  - There are **two structs you should modify**.
  - Make two structures compatible with each other. And they need to match with the inode layout in your kernel.

About submission (IMPORTANT!)

- Make sure your branch name: *proj4*
- Don't be late!
  - TA will not grade the commits after the deadline.
- Slides and Demo
  - Send it to the TA's email (os-tas@spl.snu.ac.kr) before the deadline.
  - os-tas@spl.snu.ac.kr
  - Title: **[OS-ProjX] TeamX slides&demo submission**
  - File name: **TeamX-slides.ppt(.pdf)**, **TeamX-demo.mp4(.avi….)**
- Check for format : slides title / demo name / branch name and directory name
- Please aggregate your demo videos (=submit only one video!)

Announcement

- Design Review
  - Team1~7 ⇒ Kyungtae Kim: heaven **at** snu **dot** ac **dot** kr
    - Available schedule
      - Fri, 13:00~17:30
      - Tue, 13:00~16:00
  - Team8~14 ⇒ Sungwoo Cho: pigbug419 **at** gmail **dot** com
    - Available schedule
      - Mon, 10:00~
      - Tue, 10:00~15:25, 17:00~
- Check your source code before submission

Thank you for your listening!

Happy coding!