Project 6

- Check svn for updated files / tests
    - https://svn.cs.umd.edu/repos/geekos/project0
  - New pfat.c to allow cp etc.
- ACLs:

Project 6

- Aka Project #5 Part 2:
  - Doubly Important to finish Project 5
- User ID
- Set Uid Bit (”sticky” bit)
- Access Control Lists

User Identifier

- Integer (well, almost)
- syscall.c:
  - Sys_GetUid()
  - Sys_SetUid(int uid)
    - Start as root, change to any other user ID
    - Not exactly linux like (where you login)
    - But not so different either

Sticky bit

- SYS_SET_SET_UID
  - Only user and owner can set
  - Suppose a file has it set, and is writable, executable by another user
    - What is the risk?
- No need to implement a new syscal to read, but you need to update all Stat...

Access Control Lists

- Project 5 : GOSFSfileNode
- This changes: ACLs are there:

```
typedef struct {
    char name[64];                      /* name of file */
    int size;                           /* size of the file */
    unsigned int isUsed:1;              /* is entry active */
    unsigned int isDirectory:1;         /* is this file a directory */
    unsigned int isSetUid:1;            /* set user id bit */
    int blocks[10];                     /* ... */
    struct VFS_ACL_Entry acls[VFS_MAX_ACL_ENTRIES];
                                        /* ACL entries */
} GOSFSfileNode;
```

•ACL system call

- Sys_SetAcl(char *name, int uid, int permissions)
  - Refers to ONE entry at a time
  - Can be used to add / remove permissions
  - Every {file,dir} has one and only one owner
    - Set ower will change owner
  - Owner can only be changed
  - Use 0 to remove, (Owner stays)
  - Example

VFS

- syscall.c
  - Well, thats the easy part
  - Sys_SetAcl(char *name, int uid, int permissions)
      - Refers to ONE entry at a time
  - Check
- You will have to add to VFS
- Part of the functionality is there:
  - Structures (fileio.h, vfs.h)
  - But will have to add code
      - Check vfs.h TODO
      - Add to mount point operations (why?)

fileio.h

```
/*
 * An entry in an Access Control List (ACL).
 * Represents a set of permissions for a particular user id.
 */
struct VFS_ACL_Entry {
    uint_t uid:28;
    uint_t permission:4;
};

/*
 * Generic structure representing the metadata for a directory entry.
 * This is filled in by the Stat() and FStat() VFS functions.
 */
struct VFS_File_Stat {
    int size;
    int isDirectory:1;
    int isSetuid:1;
    struct VFS_ACL_Entry acls[VFS_MAX_ACL_ENTRIES];
};
```

fileio.h (2)

- Why define O_OWNER as 1?

```
/*
 * File permissions.
 * These are used as flags for Open() VFS function.
 * O_READ and O_WRITE are also used in the permissions
 * field of struct VFS_ACL_Entry.
 */
#define O_CREATE        0x1    /* Create the file if it doesn't exist. */
#define O_READ          0x2    /* Open file for reading. */
#define O_WRITE         0x4    /* Open file for writing. */
/* conditional to support inclusion from user tools outside geekos */
#ifndef O_EXCL
#define O_EXCL          0x8    /* Don't create file if it already exists. */
#endif

/*
 * rest of File Permissions for the
 * field of struct VFS_ACL_Entry.
*/
#define O_OWNER         0x1  /* Is owner, there is always one and only owner */
#define O_EXECUTE       0x8  /* Can execute file */
```