NATIONAL RESEARCH UNIVERSITY

Computer Architecture and Operating Systems
Lecture 10: Users, Groups, and Permissions

Andrei Tatarnikov

atatarnikov@hse.ru
@andrewt0301

Authentication and Access Control

The security concerns can be classified in two groups:

- **Authentication** – making sure that nobody can access the system without first proving that she has entry rights
- **Access control** – providing a mechanism for checking whether a user has the right to access a certain object and preventing access to objects as required

User Attributes

- Login Name
- Encrypted Password
- User ID (UID)
- Group ID (GID)
- Home Directory
- Comment
- Login Shell

```
tatarnikov@akos:~$ cat /etc/passwd | grep -C 1 tatarnikov
rdavydov:x:1000:1001::/home/rdavydov:/bin/bash
tatarnikov:x:1001:1002:,,,:/home/tatarnikov:/bin/bash
chgena:x:1002:1003:,,,:/home/chgena:/bin/bash
```

Root User

- User with UID = 0 is special
- It is typically named **root** (though this is not fixed)
- Processes run by **root** have no access control limitations (can do everything)

Group Attributes

- Group Name
- Encrypted Password
- Group Identifier (GID)
- User List

```
tatarnikov@akos:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog
sudo:x:27:tatarnikov,chgena,kanakhin,ejudge,nikita
```

Utilities to Manager Users and Groups

- **passwd** – set password
- **useradd** – add user
- **userdel** – delete user
- **usermod** – modify user
- **groupadd** – add group
- **groupdel –** delete group
- **groupmod** – modify group

To be covered in the workshop

Process Credentials

Attributes

- Real user ID and group ID
- Effective user ID and group ID
- Saved set-user-ID and saved set-group-ID
- File-system user ID and group ID (Linux-specific)
- Supplementary group IDs

Utilities

- su - run a command with substitute user and group ID
- sudo — execute a command as another user

Discretionary Access Control

- Model “user-group-others”
- If the process UID matches the file UID, the set of **user** rights is used
- If one of the process GIDs matches the file GID, the set of **group** rights is taken
- Otherwise the set of **other** rights is taken

Access Rights

- **r, w, x** — interpretation is different for folders and files
- Files:
  - **r** — right to read from file
  - **w** — right to write to file
  - **x** — right to execute a file
- Folders:
  - **r** — right to read the list of files
  - **w** — right to modify the list of files (create, delete, rename)
  - **x** — right to find the specified file name
    - E.g. "--x" means a users cannot see the list of file name, but can access specific files if he knows their names

Getting and Setting Permissions

- Utility “ls”

```
tatarnikov@akos:/home$ cd tatarnikov/hello/
tatarnikov@akos:~/hello$ ls -l
total 28
-rwxrwxr-x 1 tatarnikov tatarnikov 16696 Apr 12 15:52 hello
-rw-rw-r-- 1 tatarnikov tatarnikov    71 Apr 12 15:50 hello.c
-rw-rw-r-- 1 tatarnikov tatarnikov    56 Apr 12 15:51 Makefile
```

- Utility “chmod”

```
tatarnikov@akos:~/hello$ chmod o+w hello.c
tatarnikov@akos:~/hello$ ls hello.c -l
-rw-rw-rw- 1 tatarnikov tatarnikov 71 Apr 12 15:50 hello.c
```

Permission Groups

- **u** – Owner
- **g** – Group
- **o** – Others
- **a** – All Users

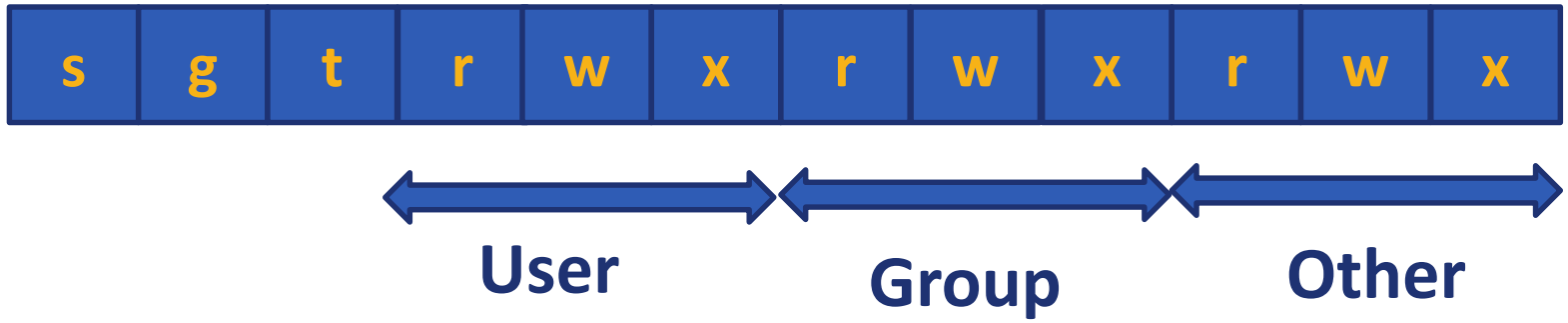
The permission assignment are: + (plus) and – (minus); these are used to tell the system whether to add or remove the specific permissions.

Numeric Values for Permissions

- r = 4
- w = 2
- x = 1

A sample permission string would be **chmod 640 file1**, which means that the owner has read and write permissions, the group has read permissions, and all other user have no rights to the file.

Permissions Bits



- Total 12 bits (9 main + 3 additional)
- **0777** — full access for everyone
- **0664** — read/write permissions for owned and group, others read only
- **0700** — only owner (user) has permissions

Advanced Permissions

- **_** – no special permissions
- **d** – directory
- **l** – file or directory is a symbolic link
- **s** –indicates the *setuid*/*setgid* permissions (if defined is shown in the read portion of the owner or group permissions).
- **t** – indicates the sticky bit permissions (if defined shown in the executable portion of the all users permissions)

Setuid/Setgid Special Permissions

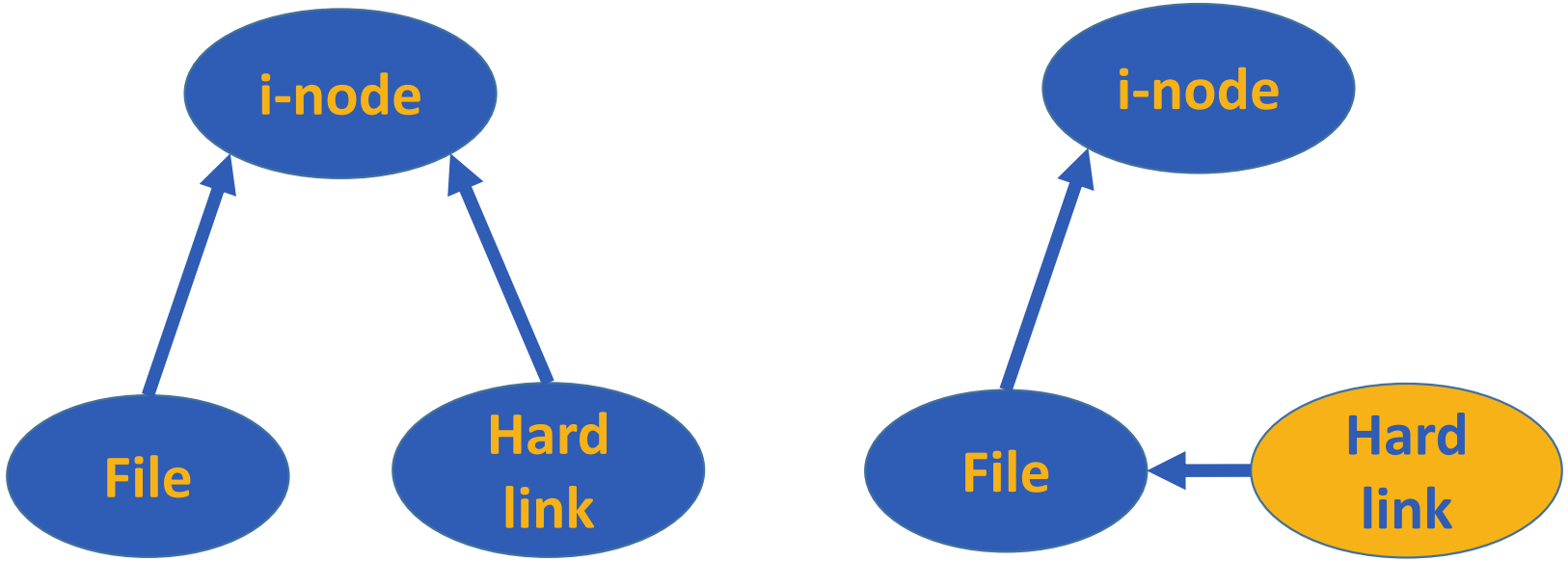
- Used to tell the system to run an executable as the owner with the owner's permissions
- Must be used with care (incorrectly assigned permissions to a file owned by root can open system to intrusion)
- Assigned in the following way: ***chmod g+s file***

Sticky Bit Special Permissions

- Useful in shared environments
- When assigned to the permissions on a directory, only file owner can rename or delete files
- Assigned in the following way: ***chmod +t dir1***

Links

- Links (hard links)
- Symbolic links (soft links)



Links: Example

```
tatarnikov@akos:~$ mkdir links
tatarnikov@akos:~$ cd links/
tatarnikov@akos:~/links$ nano myfile.txt
tatarnikov@akos:~/links$ ln myfile.txt hardlink
tatarnikov@akos:~/links$ ln myfile.txt -s softlink
tatarnikov@akos:~/links$ ls -li
total 8
1030979 -rw-rw-r-- 2 tatarnikov tatarnikov 19 May 24 05:33 hardlink
1030979 -rw-rw-r-- 2 tatarnikov tatarnikov 19 May 24 05:33 myfile.txt
1030978 lrwxrwxrwx 1 tatarnikov tatarnikov 10 May 24 05:33 softlink -> myfile.txt
```

Any Questions?

```
            .text
__start:    addi t1, zero, 0x18
            addi t2, zero, 0x21
cycle:      beq t1, t2, done
            slt t0, t1, t2
            bne t0, zero, if_less
            nop
            sub t1, t1, t2
            j cycle
            nop
if_less:    sub t2, t2, t1
            j cycle
done:       add t3, t1, zero
```