

Server Operating Systems

Lecture 5

File Security

File Security

- File System Permissions
- Changing Permissions with the Command Line
- Identifying and Switching Users

Standard Unix Security Features

User Accounts and File Security

- Two lines of defense: login/passwords to access the system and **file permissions**.

Remote Access Control

- Firewall software such as Sun's SunScreen Secure Net and ipchains or iptables on Linux help restrict remote access.
- Shutdown daemons like FTP, rlogin, Telnet if they are not needed.

Security Check Tools

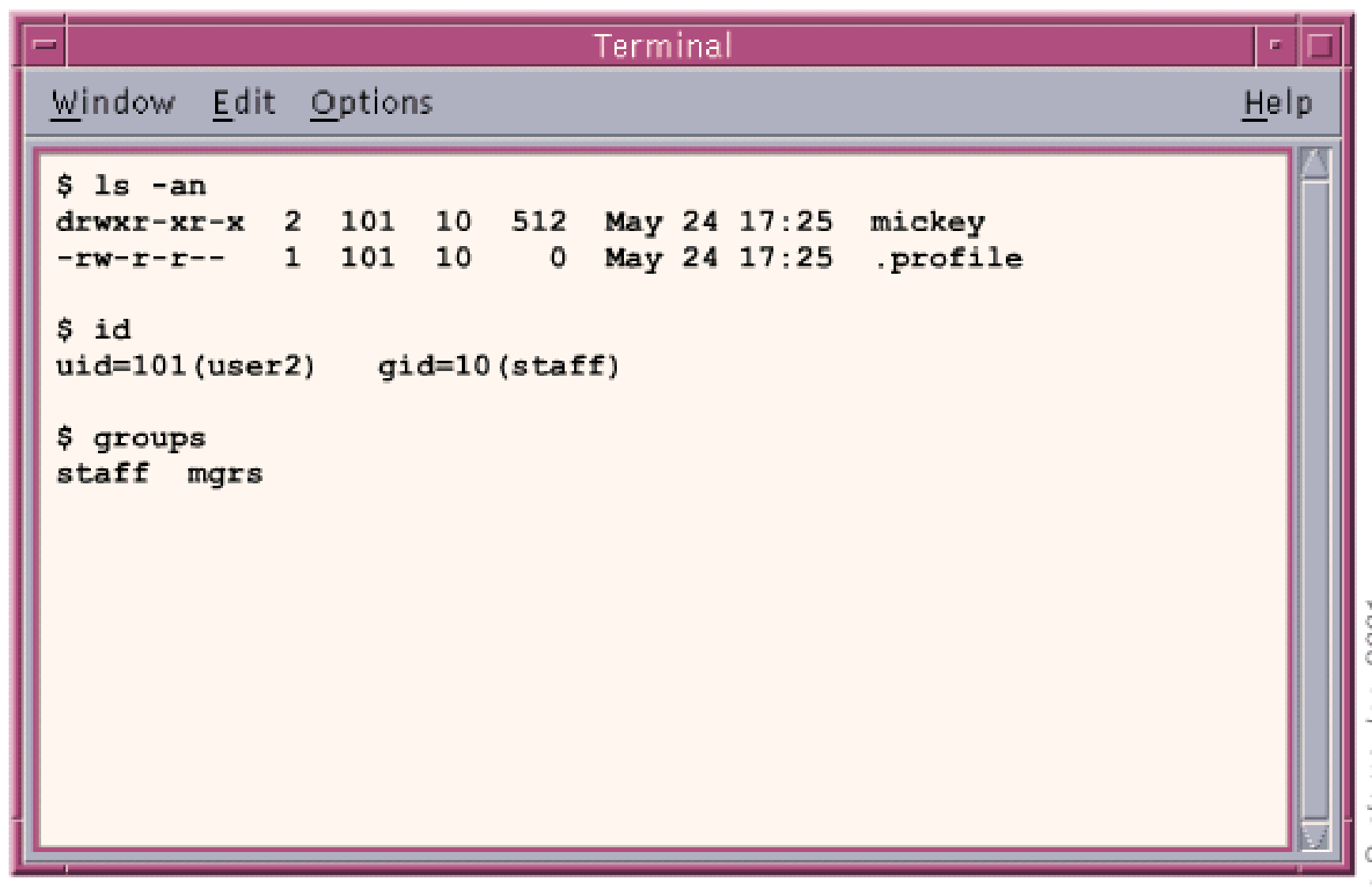
- Automatically check file system integrity.

Displaying File System Permissions

Permissions determine which users can access files and directories in the file system and what those users can do with them.

Get a “long listing” of a directory to see file permissions.

User & Group ID

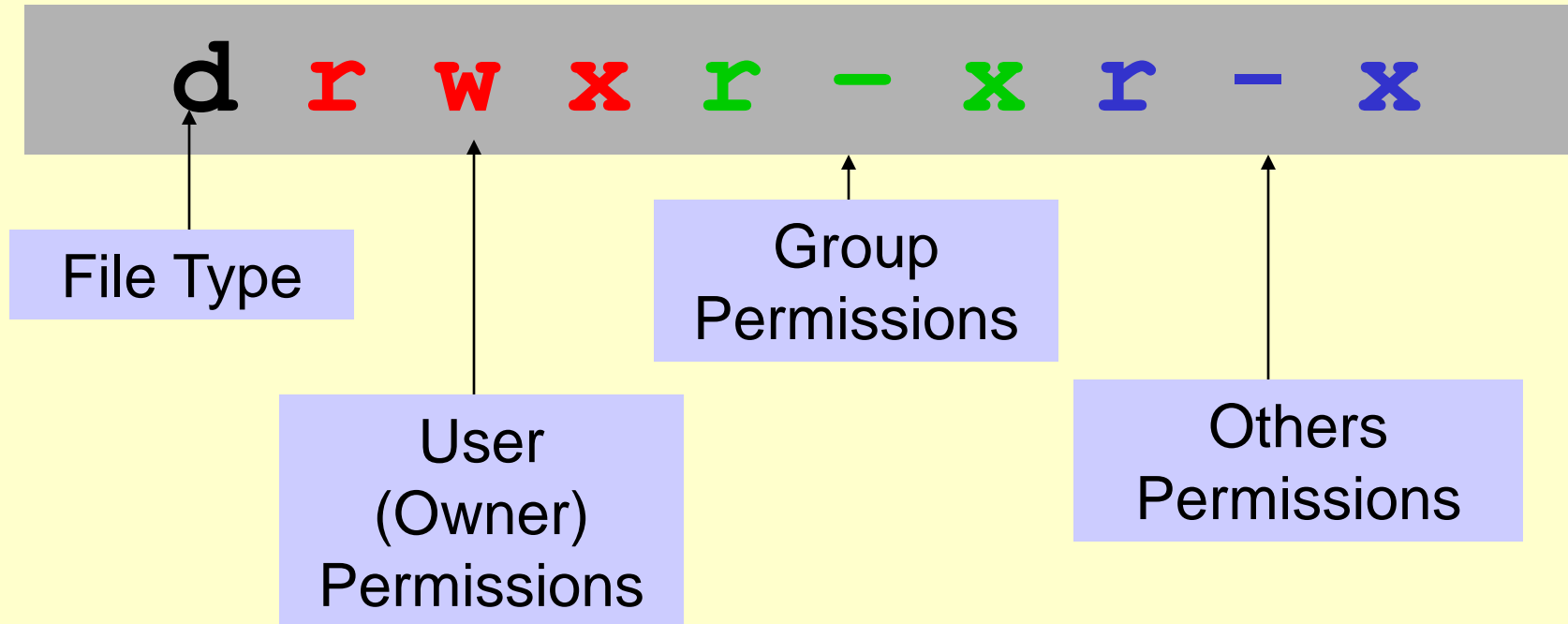
A screenshot of a terminal window titled "Terminal". The window has a menu bar with "Window", "Edit", "Options", and "Help". The terminal content shows the output of three commands: "ls -an", "id", and "groups".

```
$ ls -an
drwxr-xr-x  2  101  10  512  May 24 17:25  mickey
-rw-r-r--   1  101  10    0  May 24 17:25  .profile

$ id
uid=101(user2)   gid=10(staff)

$ groups
staff mgrs
```

Permission Categories



Permissions Categories

File Type

- Usually a ‘-‘ for normal files, or ‘d’ for directory.

User (Owner)

- User who created or owns the file.

Group

- Group the user belongs to (set by sys. admin.).

Others

- Everyone else (outside of user and group) who has access to the system.

Permission Types

Each permission category has three permission types: *read*, *write*, and *execute*.

- They are signified in the long listing by ‘*r*’, ‘*w*’, and ‘*x*’ in that order for each category of permissions.
- A ‘-’ indicates that that type of permission is denied.

To be able to access directories the ‘*r*’ and ‘*x*’ permissions must be set.

- So, when you want to copy a file from a directory, or move files to or from a directory, ‘*r*’ and ‘*x*’ must be set on the directory. Also, you’ll need ‘*r*’ permission on the file being moved or copied.

File Protection Summary

<u>Access</u>	<u>Result for normal users</u>
---	No activity of any sort . Cannot even see the file.
r--	File can be displayed or copied.
--x	Executable program files can be executed.
r-x	Files can be displayed, copied and executed. This is required to run a script file.
rwX	File can be moved, modified, copied, deleted and executed.

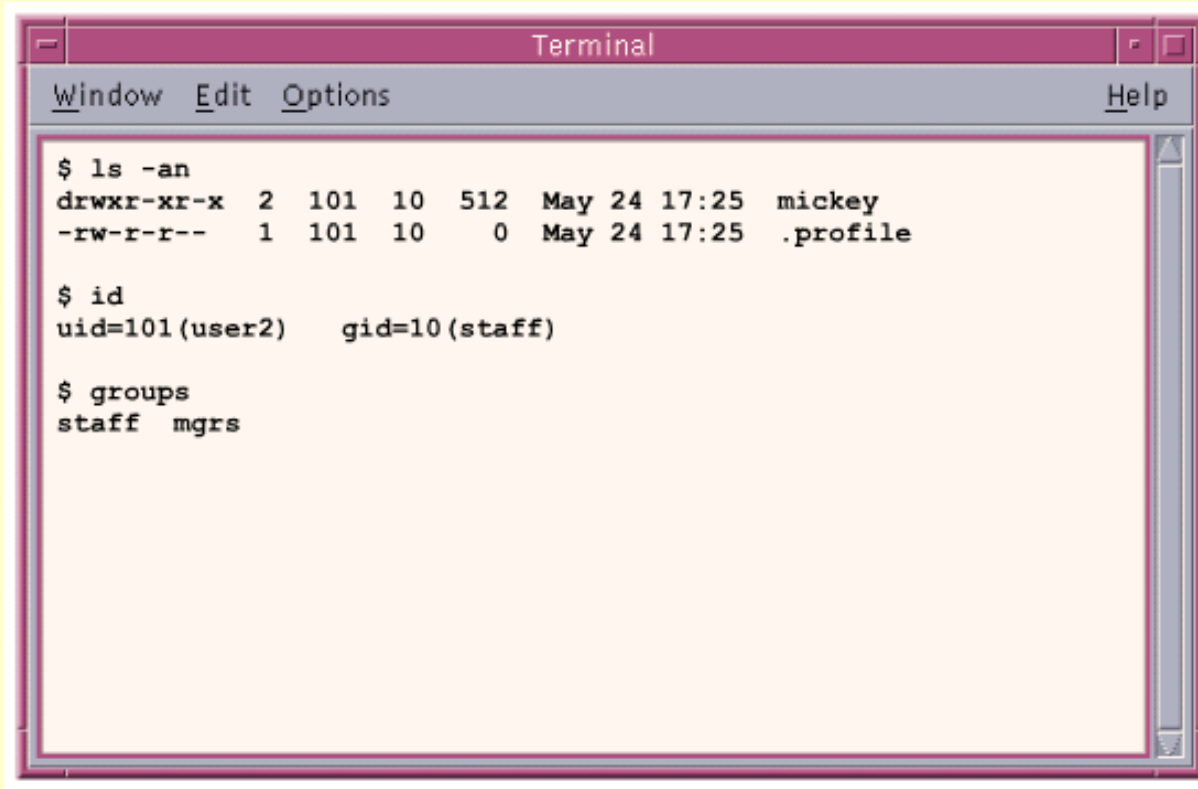
Directory Protection Summary

<u>Access</u>	<u>Result for normal users</u>
---	No activity of any sort within the directory or any of its subdirectories.
r--	Can list names of files in directory, but does not reveal any of their attributes. (size, ownership etc)
--x	Lets users run programs in the directory that they already know the names of, but hides all others.
r-x	Lets users run programs in the directory, and list contents of directory, but does not allow them to create or delete files.
rwX	Lets users work with programs, list contents, and create and delete files.

Permissions Table

Permission	Symbol	File	Directory
Read	r	File can be displayed or copied. A copied file takes on a new owner. Cannot move or delete a file that has only read permission.	Contents can be listed with ls command. (Must also have execute permission to use ls command options.)
Write	w	File can be modified, moved and deleted, but only if the directory that it resides in has write permission.	Files can be added or deleted, but only if directory also has execute permission.
Execute	x	Scripts or executable files can be executed. (Scripts need read access too.)	Controls access to directory. A user can cd to the directory and list the contents. Files can be moved or copied to the directory.
None	-	Permission denied.	Permission denied.

User & Group ID



```
Terminal
Window Edit Options Help

$ ls -an
drwxr-xr-x  2 101 10 512 May 24 17:25 mickey
-rw-r-r--   1 101 10   0 May 24 17:25 .profile

$ id
uid=101(user2)  gid=10(staff)

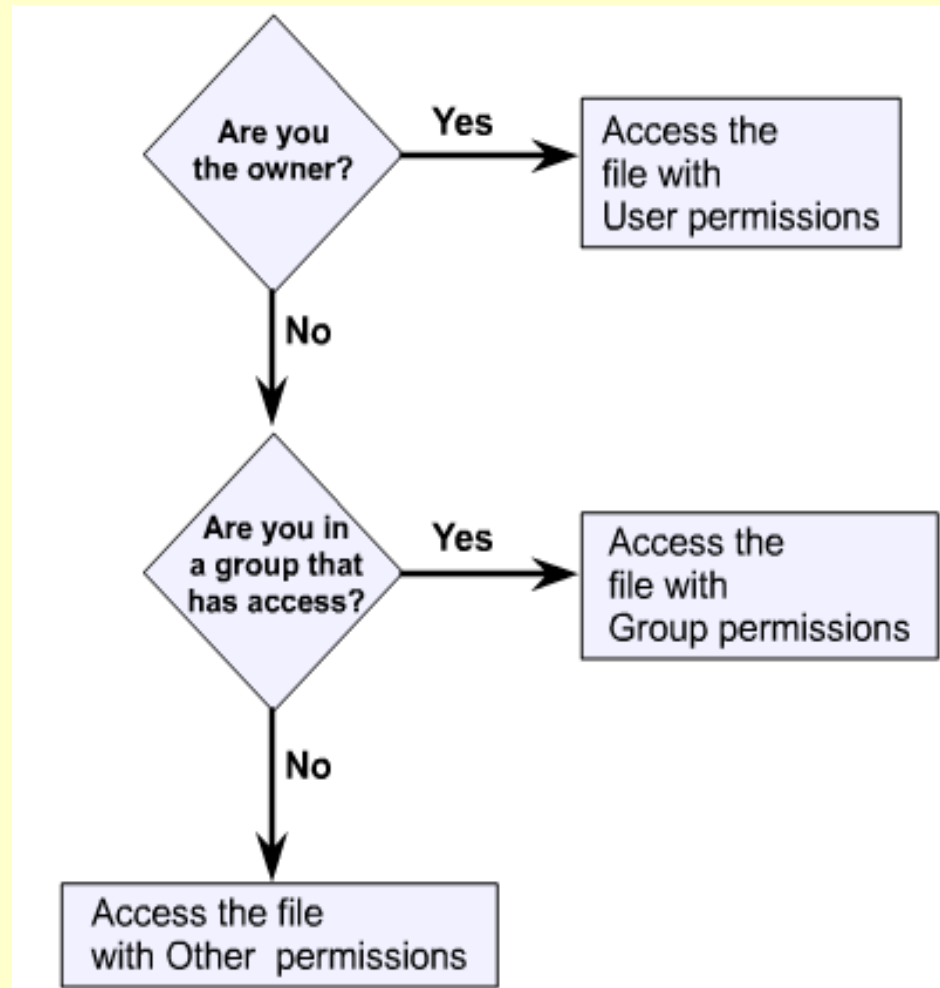
$ groups
staff mgrs
```

ls -n displays the UID and the GID

id displays numeric and alphabetic User ID and Group ID for your Effective User ID (EUID)

groups displays all of the groups you are a member of

Determining File and Directory Access



File Permissions Example

```
-rwxr-xr-- student year1 512 kevin.sh
```

If you are logged on as student, what can you do with the file?

If you are logged on as user5, who is also a member of the group year1, what can you do with the file?

If you are logged on as user007, what can you do with the file?

Directory Permissions Example

```
drwxr----- student year1 512 myfolder
```

which contains

```
-rwxr-xr-- student year1 512 kevin.sh
```

If you are logged on as student, what can you do with the directory?

If you are logged on as user5, who is also a member of the group year1, what can you do with the directory?

If you are logged on as user007, what can you do with the directory?

File and Directory Default Permissions

User Category	User (owner)	Group	Other
New File	Read/Write	Read	Read
New Directory	Read/Write/Execute	Read/Execute	Read/Execute

New File: **-rw-r--r--**

New Directory: **drwxr-xr-x**

This means that files that you download from the net do NOT have execute permission set by default, even if they are executables. Why?

A

Changing Permissions with the Command Line

A

The **chmod** command can be used by a file's owner or the superuser to change its permissions.

There are two modes **chmod** works in:

- *Symbolic* (or *relative*)
- *Octal* (or *absolute*).

Symbolic chmod Mode

Change the permissions for a group of users *relative* to their current permissions.

chmod *mode filename*

mode consists of three parts:

- **WHO**: The category of users to work with (**u**, **g**, **o**, or **a**).
- **OP**: Operator to use (**=**, **+**, **-**).
- **PERMISSIONS**: Permissions to set, add or subtract (**r**, **w**, **x**).

Symbolic chmod Mode

File: `-rw-r--r--` `wilson.txt`

Apply: `chmod u+x,o-r wilson.txt`

Will give: `-rwxr-----` `wilson.txt`

What about:

`chmod u-r,go+r-w+x wilson.txt`

Will give: `--wxr-xr-x` `wilson.txt`

Octal chmod Mode

Numerically change a file's permissions for all categories simultaneously.

```
chmod octal_mode filename
```

```
chmod -R octal_mode directory_name
```

- Changes the permissions on the directory and all files in it at once.

Octal chmod Mode Values

Octal Value	Permissions
4	Read
2	Write
1	Execute

Octal Value	Sum of Permission Category Values	Corresponding Permissions	Definition
7	$4 + 2 + 1$	r w x	Read, Write and Execute
6	$4 + 2 + 0$	r w -	Read and Write
5	$4 + 0 + 1$	r - x	Read and Execute
4	$4 + 0 + 0$	r - -	Read only
3	$0 + 2 + 1$	- w x	Write and Execute
2	$0 + 2 + 0$	- w -	Write only
1	$0 + 0 + 1$	- - x	Execute only
0	$0 + 0 + 0$	- - -	No permissions

Octal chmod Mode

File: `-rw-r--r--` `notes5.txt`

Apply: `chmod 715 notes5.txt`

Will give: `-rwx--xr-x` `notes5.txt`

What about:

`chmod 070 work.txt`

Will give: `-----rwx----` `work.txt`

Changing the Default Permissions

The default permissions applied to files and directories can be changed with the **umask** command.

umask will display current mask

umask 722 will set mask

The mask is reset each time you login. It can be placed in an initialization file to keep it saved.

Changing Ownership and Groups

Using the **chown** command the owner of a file can be changed.

- On Linux only the superuser can do this.
- On other versions of UNIX, the current owner of the file or the superuser can do this.
- **chown [-R] *new_owner filename***

Change the group ownership of a file with the **chgrp** command.

- Only superuser can use this on Linux.
- Current owner or superuser can use this on other UNIXs.
- **chgrp [-R] *new_group filename***

Groups in Unix

There are two types of groups in Unix.

- *Primary Group*: The group assigned to a user by the sys. admin. when the user's account is created (stored in the /etc/passwd file).
 - Every user must have a primary group.
- *Secondary Group*: Other groups the user belongs to (stored in /etc/groups file).

Identifying Users on a Unix System

who gives detailed information on every user logged on to the system.

finger gives the same info as **who**, but can give more detailed information on individual users. It also gives this info whether the user is logged onto the system or not.

Switching to Another User Account

You can switch to another user's account with the **su** (“Switch User”) command.

su [-] *username*

- Once switched, you are subject to the same constraints as the user you are now logged in as.
- Type **exit** to log out as the new user.
- The ‘-‘ puts you in the new user's home directory and reads initialization files.
- If no ***username*** is given, root is assumed. The ‘-‘ can still be used (**su -**).

User Account Information

The *Real User ID* (**RUID**) is the account logged into the system originally with.

The *Effective User ID* (**EUID**) is the account currently being used.

The **who am i** command displays the RUID.

The **id** command shows information on the EUID.