Linux Users and File Permissions

Some administration:

- Lab 4 will be assigned next Thursday
- `/etc/passwd`, `/etc/shadow`, `/etc/group` and you.

Topics:

- Users
- Files
- Users, Groups, and Others (everyone else)
- basic and advanced permissions
- sudo, root, and permissions
- File Access Control Lists (ACL)

GPG last words (for now)

- Backup your `~/.gnupg` folder!
- You can export all public keys at once!
- Check your grades!
- I'm almost done exclaiming!!!
- alright, we're good now

Passwords, passwd, and you

some notes:

- `passwd` - the command to reset user passwords
- `/etc/passwd` - the file that stores user information, but NOT their password OR hashed password
- `/etc/shadow` - the file that stores user password hashes (with salts!)
- `/etc/group` - the file that stores group information including group members

Everything is a file

File Type	ID	Description
Normal	`-`	normal file
Directory	`d`	normal directory
Hard link	`-`	additional name for existing file
Symbolic link	`l`	shortcut to a file or directory
Socket	`s`	used to pass data between two processes
Named pipe	`p`	socket but users cannot work directly with pipes
Character device	`c`	processes character hardware communications
Block device	`b`	processes block hardware communications

Users

In Linux, a user is an entity that can manipulate files (and some other things. . .)

Users have some of the following properties:

- Real name <- for everyone else
- User name <- for sysadmin / the user
- User ID <- (UID) what the computer actually works with
- Group(s) <- Similar to users, the computer doesnt care about t he group name, it checks a GID
- password(?)

Files belong to users (and groups)

Linux file permissions are associated with the following three groups of users:

- the **U**ser that owns the file
- the **G**roup that the file belongs to
- **O**ther users (everyone else, sometimes **W**orld)
- You can change the owner and group with `chown` and `chgrp` respectively.
- You can check members of groups via the `/etc/group` file with `cat`.

Running processes have users and groups too!

This mean a process has the same permissions as the user that the process is running as!

UID's and processes

Three types of UID's

1. **Real UID**: UID of parent process (whoever is running the process)
2. **Effective UID**: Processes can gain or ched permissions by changing their UID, think `passwd`
3. **Saved UID**: The UID that is available to the process via some other means.

Types of permissions

3 basic types, **R**ead, **W**rite, and e**X**ecute

These are applied to each of the three groups defined previously.

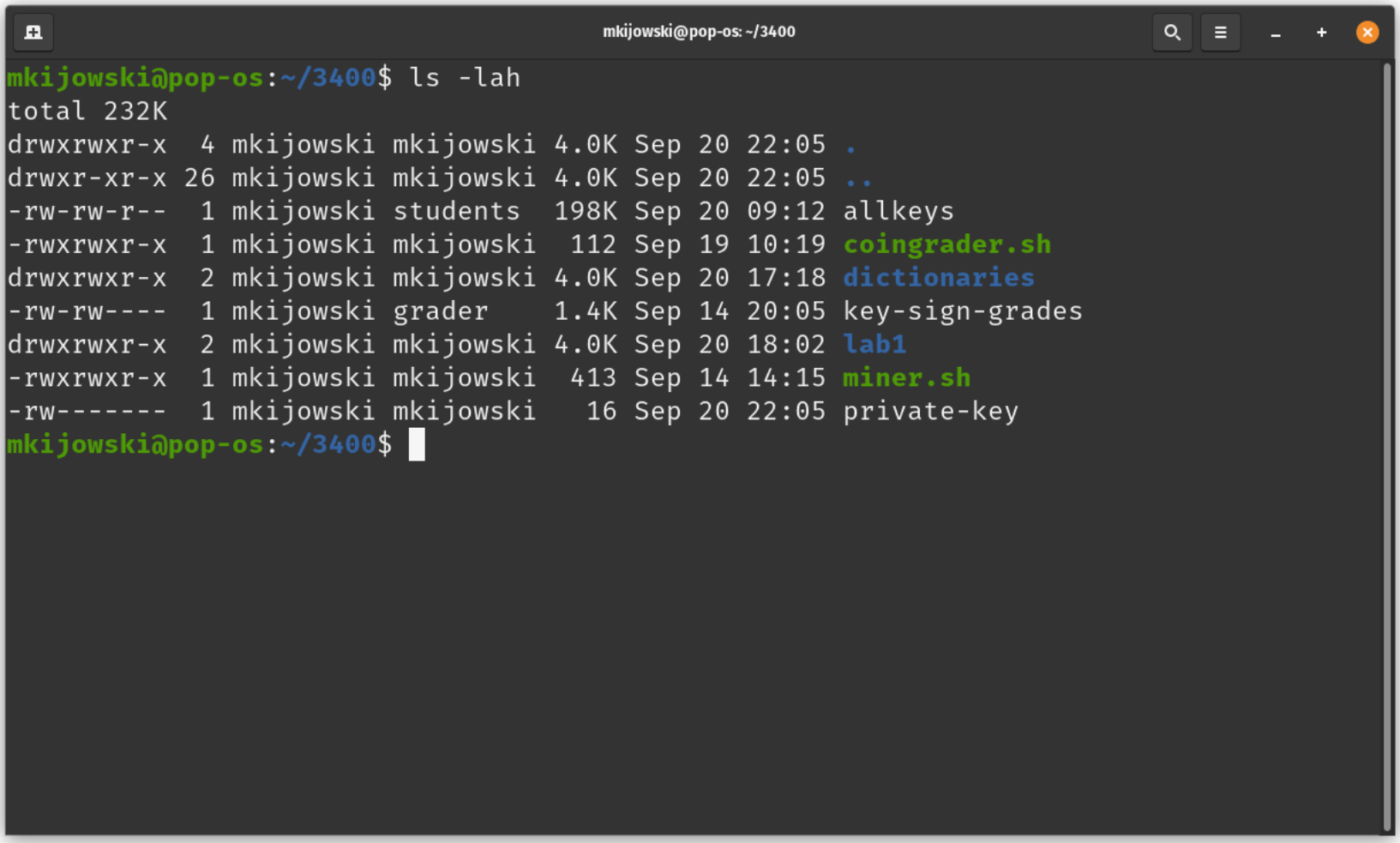

Figure 1: permissions

Effect of permissions on Files

Permission	Character	Description
Read	`-`	The file is not readable
	`r`	The file is readable
Write	`-`	The file cannot be changed or modified
	`w`	The file can be changed or modified
Execute	`-`	The file cannot be executed
	`x`	The file can be executed
	`s`	In the `users` triplet, `setuid`. If found in the `group` triplet, `setgid`. `s` implies executable but also will execute with the

Effect of permissions on Directories (folders)

Permission	Character	Description
Read	`-`	Directory contents cannot be listed
	`r`	Directory contents can be listed
Write	`-`	Directory contents cannot be altered
	`w`	Directory content can be altered (files can be added or removed)
Execute	`-`	Directory cannot be changed to (cannot `cd`)
	`x`	Directory can be changed to
	`s`	If in `user` triplet, `setuid` (which does

root, sudo, and security

the `root` user is the highest level of access on a system.

`sudo` or "super-user do" allows you to execute commands as the `root` user.

You can allow/deny access to the full `sudo` command in most systems with the sudo group in `/etc/group`

You can specify command specific use of `sudo` with the `visudo` command. This allows specified users and groups access to use all or a subset of available commands with sudo permissions.

For security reasons we want to restrict the use of root and sudo as much as possible!

- limit access to sudo to only those that need it
- limit use of sudo to only the commands that need elevated privileges

Principle of Least Privilege (PoLP)

Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job. -Jerome Saltzer, Communications of the ACM

- Very useful principle
- Nearly impossible in practice (too much complexity)
- still useful across all areas of cyber security

CIA

- We can protect ***Confidentiality*** by restricting `others` access to files and limiting access to groups.
- We can protect ***Integrity*** by limiting `write` access to files.
- We can ensure ***Availability*** by ensuring users that need access to files are in the proper groups.

Limitations of basic Linux File Permissions

Files can only have one owner and one group.

ACL's allow for finer permissions settings per file.

`setfacl` and `getfacl` allow for additional user/group triplets.

- `setfacl -m u:<username>:<triplet> <filename>`
- `setfacl -m g:<groupname>:<triplet> <filename>`