Exercise 1 (1/5)

- Write a directory monitoring program **monitor.c** which watches a specific directory (*path*) and reports the changes made to the directory. The changes should be watched are applicable to the directory itself, its sub-directories and all files inside the directory. The path (`path`) of the directory should be passed as a command line argument to the program.
- The report message should identify the type (event) of change and should also print the stat info of the changed item (file/folder). For example if file (f1.txt) is modified, it should prints a message "f1.txt is modified" and the stat info for this file should also be printed to stdout.
- The **monitor.c** program should be able to print the stat info of all files and folders in the path (`path`) once on the startup, and once before termination.
- We assume that we do not have files or folders inside the sub-folders of the watched directory *(path)*.

Exercise 1 (2/5)

- The user can terminate the program **monitor.c** by sending SIGINT signal (pressing Ctrl-c), and your program needs to handle this signal SIGINT such that it prints the stat info of all entries in the directory (*path*) before termination.
- Write another program **ex1.c** which will be responsible for making some changes to the watched directory. This program gets the path (`path`) of the watched directory as a command line argument.
- In **ex1.c**, add a function `find_all_hlinks(const char* source)` to find all hard links which refers to the same i-node of the *source* in the path (`path` which is passed as a command line argument) and it should print them to stdout with their inode numbers and absolute paths (Use *realpath* function).
- In **ex1.c**, Add a function `unlink_all(const char* source)` to remove all duplicates of a hard link. This function should keep only one hard link and then print the path of the last hard link to stdout.

Exercise 1 (3/5)

- In **ex1.c**, Add a function `create_sym_link(const char* source, const char* link)` which will create a symbolic link *link* to *source* in the path (`path` which is passed as a command line argument). Use *symlink* system call.
- Write a test script **ex1_test.sh** which includes an arbitrary number of commands for creating, modifying and removing files and folders in the path (*path*). It also may include commands for creating links and accessing files and folders.
- The program **monitor.c** should monitor the following changes:
    - IN_ACCESS (File was accessed)
    - IN_CREATE (File/directory created in watched directory)
    - IN_DELETE (File/directory deleted from watched directory)
    - IN_MODIFY (File was modified)
    - IN_OPEN (File or directory was opened)
    - IN_ATTRIB (Metadata changed)
- An example of a test script is in this gist.

Exercise 1 (4/5)

- You should run the program **monitor.c** first, then the program **ex1.c**. After that, you can run the **ex1_test.sh** script.
- The main function of the program **ex1.c** should:
  - Create a file **myfile1.txt** contains the text "Hello world." and create two hard links **myfile11.txt**, and **myfile12.txt** to it and add them to the watched directory.
  - Find all hard links to **myfile1.txt** and print their i-nodes, paths, and content.
  - Move the file **myfile1.txt** to another folder /**tmp**/**myfile1.txt**.
  - Modify the file **myfile11.txt** (its content). Did the **monitor.c** program reported an event for **myfile11.txt**? Justify your answer and add it to **ex1.txt**.
  - Create a symbolic link **myfile13.txt** in the watched directory to /**tmp**/**myfile1.txt**. Modify the file /**tmp**/**myfile1.txt** (its content). Did the **monitor.c** program reported an event for **myfile13.txt**? Justify your answer and add it to **ex1.txt**.

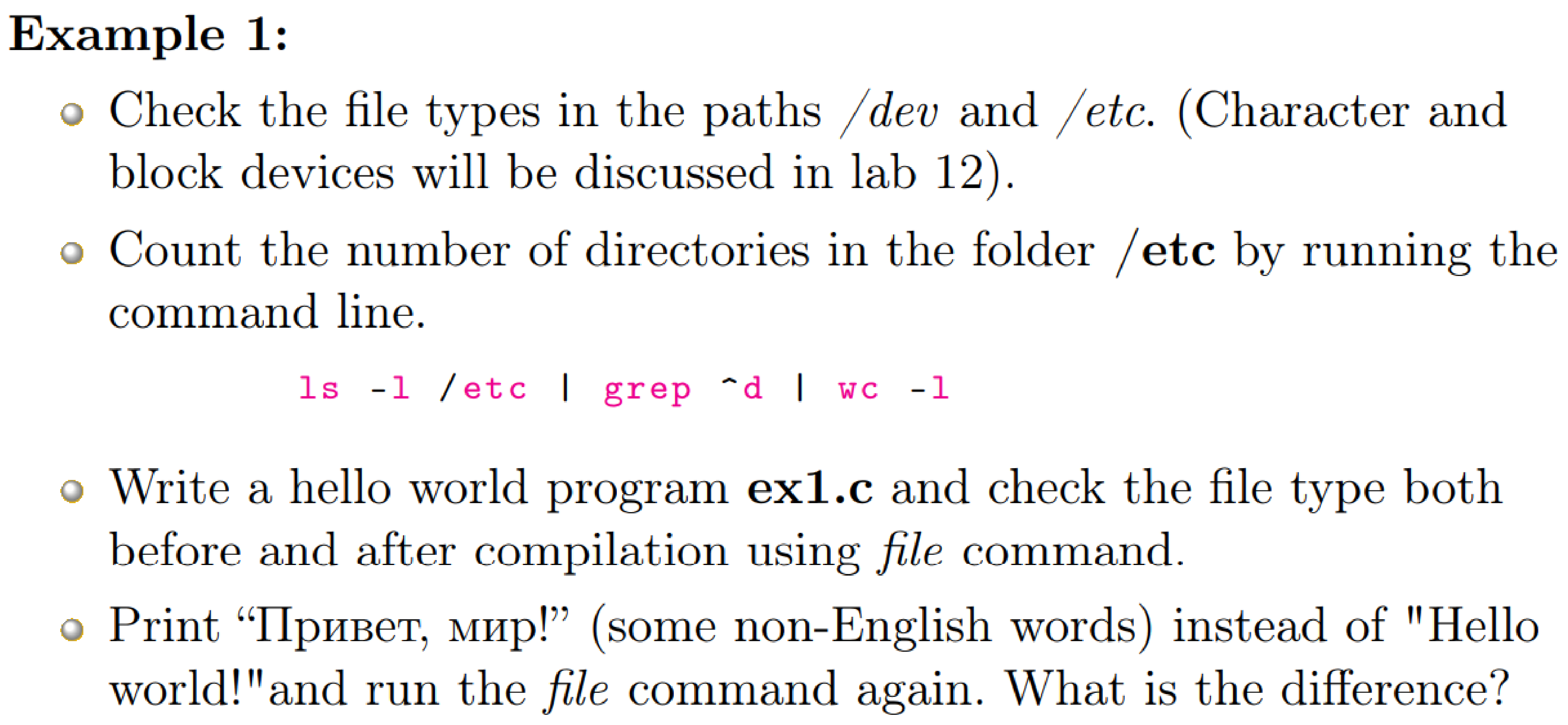
Exercise 1 (5/5)

  - Remove all duplicates of hard links to *myfile11.txt* only in the watched directory. After this, print the stat info of the kept hard link. Check the number of hard links and explain the difference. Add the answer to **ex1.txt**.
- **Hint:** Check the manual page of *inotify* at https://man7.org/linux/man-pages/man7/inotify.7.html.
- Submit **monitor.c**, **ex1.c**, **ex1.txt** and a script **ex1.sh** to run the program.
- **Hint:** Use **inotify** to track the changes to the directory entries.
- **Hint:** Use *link* system call to create hard links. Check this manual page.

Exercise 2 (Bonus exercise) P.S check next slide

- Implement the example 1 in slide 6 and add the commands and answers as comments to **ex21.sh**. Submit the script **ex21.sh** only.
- Implement the example 2 in slide 8 and add the commands and answers as comments to **ex22.sh**. Submit the script **ex22.sh**.
- Implement the example 3 in slides 11 and 12 and add the commands and answers as comments to **ex23.sh**. Submit the script **ex23.sh** and **gen.sh**.
- Implement the example 4 in slide 14 and add the commands and answers as comments to **ex24.sh**. Submit the script **ex24.sh**.
- Implement the example 5 in slide 17 and add the commands and answers as comments to **ex25.sh**. Submit the script **ex25.sh**.

Example 1 in slide 6:



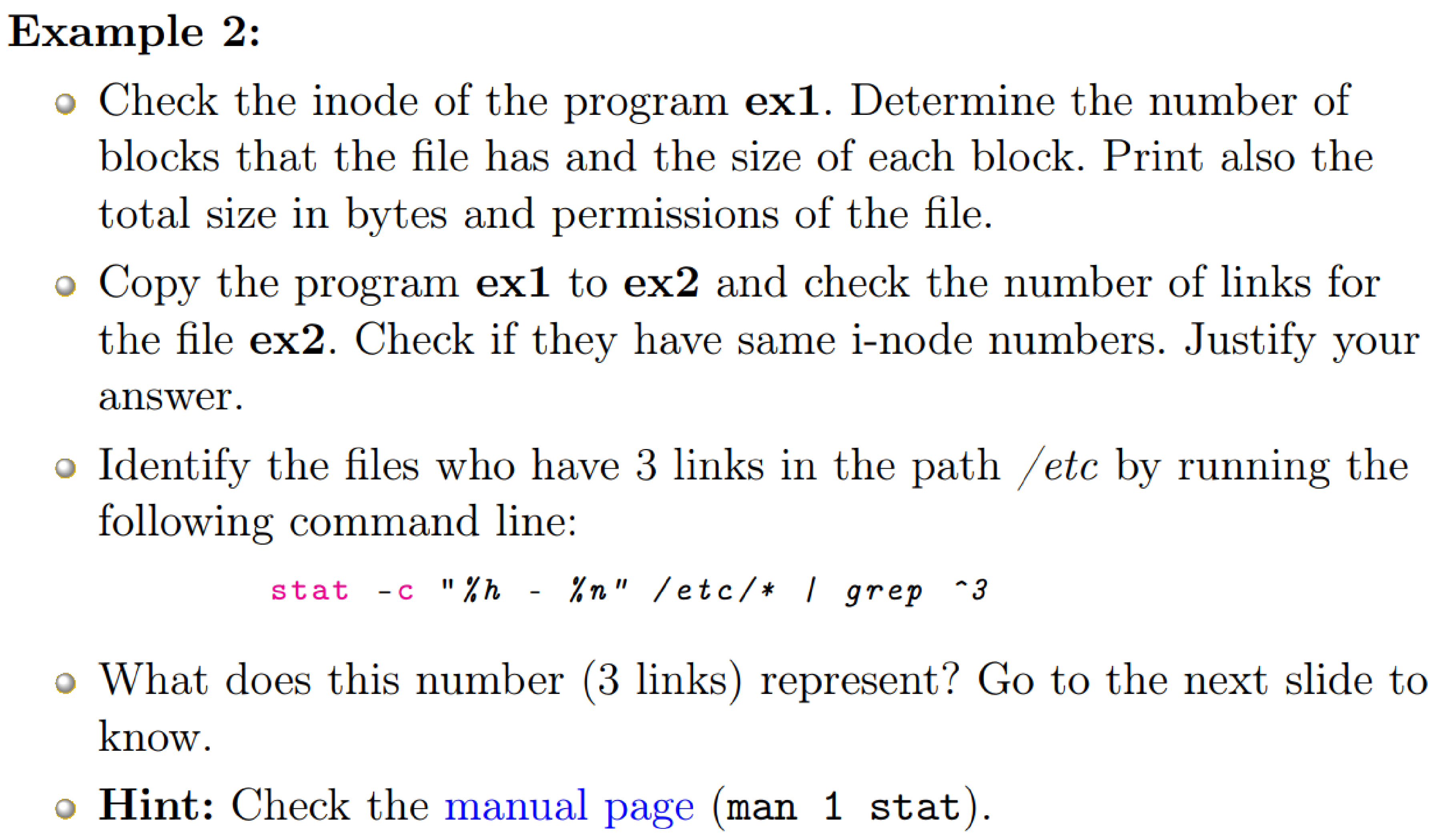
Example 1:

- Check the file types in the paths */dev* and */etc*. (Character and block devices will be discussed in lab 12).
- Count the number of directories in the folder **/etc** by running the command line.

```
ls -l /etc | grep ^d | wc -l
```

- Write a hello world program **ex1.c** and check the file type both before and after compilation using *file* command.
- Print “Привет, мир!” (some non-English words) instead of "Hello world!"and run the *file* command again. What is the difference?

Example 2 in slide 8:



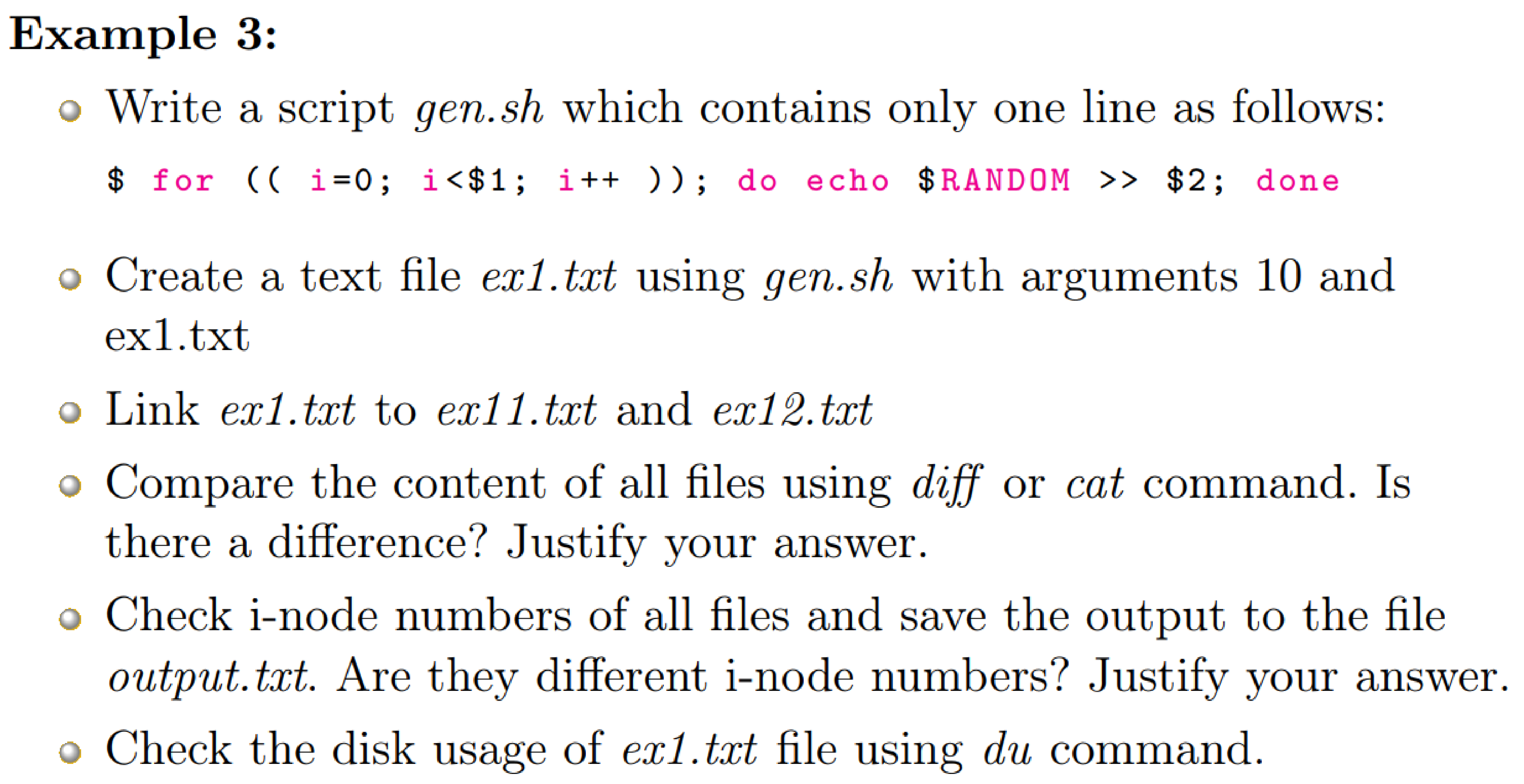
Example 2:

- Check the inode of the program **ex1**. Determine the number of blocks that the file has and the size of each block. Print also the total size in bytes and permissions of the file.
- Copy the program **ex1** to **ex2** and check the number of links for the file **ex2**. Check if they have same i-node numbers. Justify your answer.
- Identify the files who have 3 links in the path */etc* by running the following command line:

```
stat -c "%h - %n" /etc/* | grep ^3
```

- What does this number (3 links) represent? Go to the next slide to know.
- **Hint:** Check the manual page (`man 1 stat`).

Example 3 in slide 11:



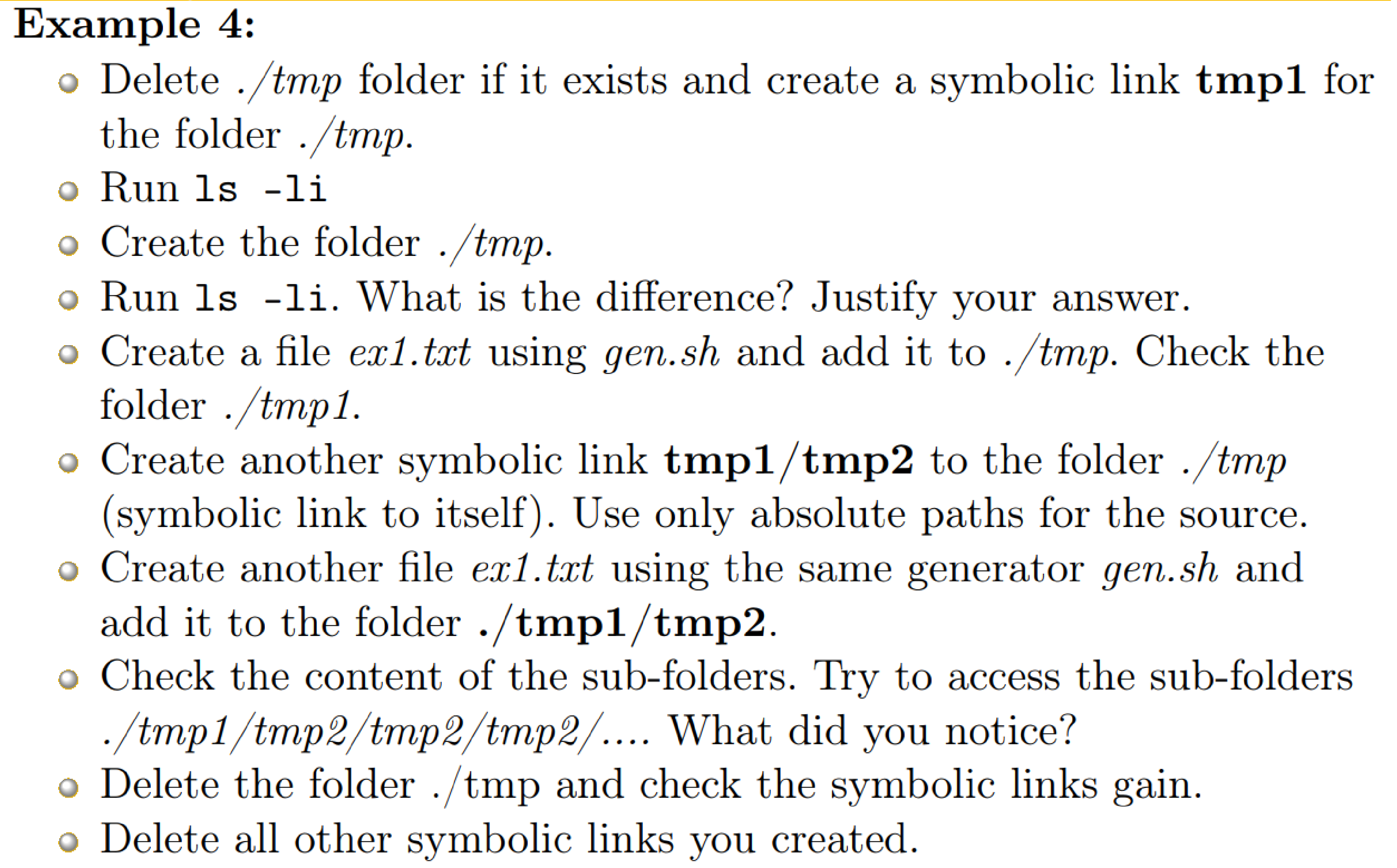
Example 3:

- Write a script *gen.sh* which contains only one line as follows:

```
$ for (( i=0; i<$1; i++ )); do echo $RANDOM >> $2; done
```

- Create a text file *ex1.txt* using *gen.sh* with arguments 10 and ex1.txt
- Link *ex1.txt* to *ex11.txt* and *ex12.txt*
- Compare the content of all files using *diff* or *cat* command. Is there a difference? Justify your answer.
- Check i-node numbers of all files and save the output to the file *output.txt*. Are they different i-node numbers? Justify your answer.
- Check the disk usage of *ex1.txt* file using *du* command.

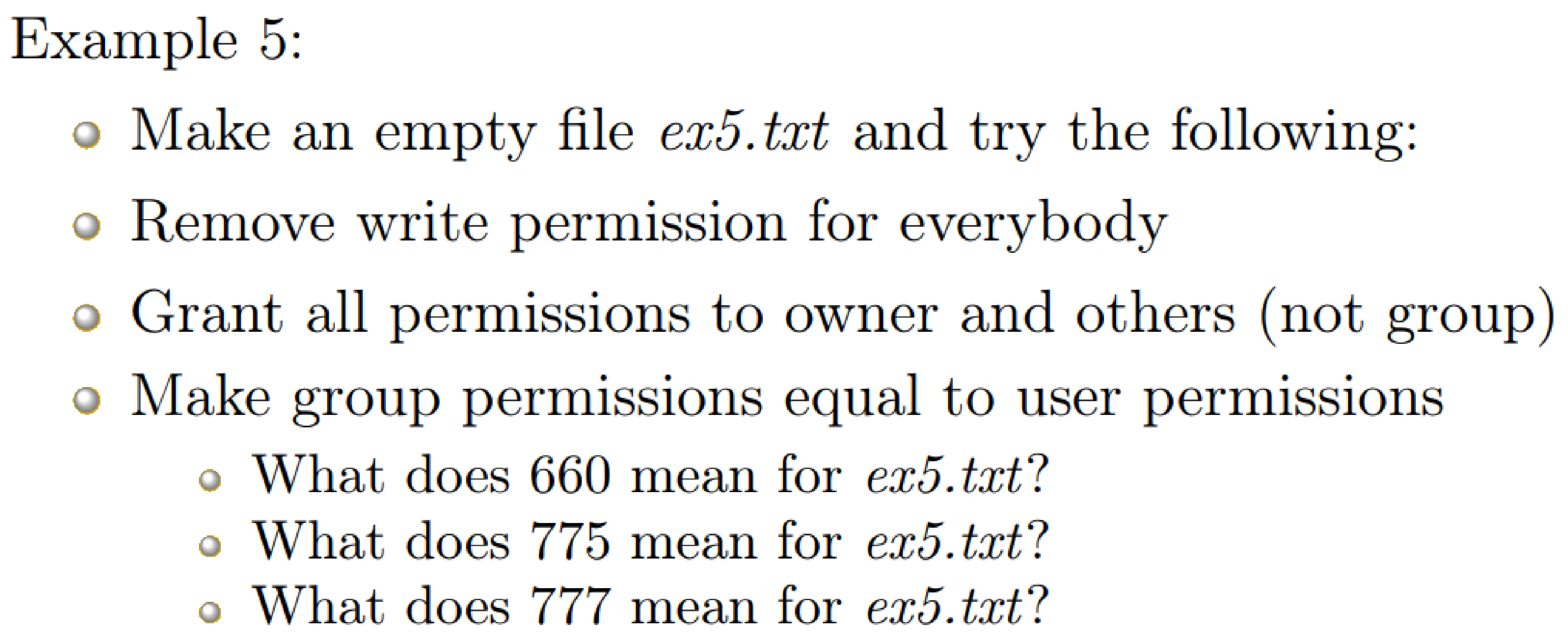
Example 4 in slide 14:



Example 4:

- Delete *./tmp* folder if it exists and create a symbolic link **tmp1** for the folder *./tmp*.
- Run `ls -li`
- Create the folder *./tmp*.
- Run `ls -li`. What is the difference? Justify your answer.
- Create a file *ex1.txt* using *gen.sh* and add it to *./tmp*. Check the folder *./tmp1*.
- Create another symbolic link **tmp1/tmp2** to the folder *./tmp* (symbolic link to itself). Use only absolute paths for the source.
- Create another file *ex1.txt* using the same generator *gen.sh* and add it to the folder **./tmp1/tmp2**.
- Check the content of the sub-folders. Try to access the sub-folders *./tmp1/tmp2/tmp2/tmp2/....* What did you notice?
- Delete the folder ./tmp and check the symbolic links gain.
- Delete all other symbolic links you created.

Example 5 in slide 17:



Example 5:

- Make an empty file *ex5.txt* and try the following:
- Remove write permission for everybody
- Grant all permissions to owner and others (not group)
- Make group permissions equal to user permissions
    - What does 660 mean for *ex5.txt*?
    - What does 775 mean for *ex5.txt*?
    - What does 777 mean for *ex5.txt*?